IBM Tivoli Monitoring Agent Builder
Version 6.3

User's Guide



SC32-1921-14

IBM Tivoli Monitoring Agent Builder
Version 6.3

User's Guide



SC32-1921-14

Note
Before you use this information and the product it supports, read the information in "Notices" on page 657.

This edition applies to version 6.3 of IBM Tivoli Monitoring Agent Builder (product number 5724-C04) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2007, 2013.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures vii

Tables xi

Chapter 1. Introduction 1
About Eclipse 2
New in this release 2
About this documentation 2

Chapter 2. Getting started quick-reference guide 3

Chapter 3. Installing the Tivoli Monitoring Agent Builder 5
Requirements 5
Generate a software product compatibility report for Agent Builder 6
Authorization 6
Builder installation procedure. 6
Silent installation 7
Uninstalling Agent Builder 8
Silent uninstallation 9
Set the default browser in Agent Builder 9

Chapter 4. Starting Agent Builder . . . 11
Starting the Agent Builder 11

Chapter 5. Creating a basic agent . . . 13
Operating system requirements. 13
Starting the new agent wizard 13
Use the icon to start the new agent wizard . . . 14
Create a file to start the new agent wizard . . . 14
Creating and defining the agent 19
Naming your agent. 19
Defining initial data source 23
Data Sources 25
Selecting key attributes 26
Defining more data sources 27
Organizing the agent 29
Preparing your agent for modifications or for regeneration 34

Chapter 6. The Tivoli Monitoring Agent Editor 35
Default operating systems 37
Self-Describing Agent 38
Environment variables. 39
Watchdog information. 41
Cognos information 42
Generate Agent wizard link 44
Committing a version of the agent. 45
The Data Source Definition page 46
Copying data sources by using the Data Source Definition page 47
Runtime Configuration Information page 48
Agent XML Editor page 49
Saving your edits and changes 50
Committing a version of the agent. 51
Changing a subsequent version of your agent . . . 52
Changing the product code 52

Chapter 7. Editing data source and attribute properties 55
Creating attributes 56
Fields and options for defining attributes . . . 58
Attribute types 59
Numeric aspects of attributes 60
Specifying an enumeration for an attribute . . . 62
Copying attributes 62
Editing attributes 62
Removing attributes 63
Creating derived attributes 63
Interval specific calculations 66
Editing derived attributes. 67
Filtering attribute groups 67
Formula Editor 68
Changing the Formula Editor component view 70
Component types 70
Formula Editor common options 78
Formula Editor - Formula errors 80
Formula operators and functions 82
Specifying operating systems 88

Chapter 8. Monitoring a process. 89
Defining connections for process browsing 93

Chapter 9. Monitoring a Windows service. 97
Defining connections for service browsing. . . . 101

Chapter 10. Monitoring data from Windows Management Instrumentation (WMI) 105
Testing WMI attribute groups 111

Chapter 11. Monitoring a Windows Performance Monitor (Perfmon) . . . 113
Testing Perfmon attribute groups 118

Chapter 12. Monitoring data from a Simple Network Management Protocol (SNMP) 121
SNMP MIB errors 127
SNMP MIB Parsing options 128
Testing SNMP attribute groups 130

© Copyright IBM Corp. 2007, 2013

Chapter 13. Monitoring events from a Simple Network Management Protocol event sender. 133
SNMP Event Configuration properties 138
Testing SNMP event attribute groups 140

Chapter 14. Monitoring Java Management Extensions (JMX) MBeans 143
JMX configuration 156
JMX notifications 159
JMX monitors 159
Take Action commands for JMX monitors . . . 160
JMX operations 164
Starting and stopping JMX monitors. 165
Specific fields for Java Management Extensions (JMX) MBeans 165
Testing JMX attribute groups 169

Chapter 15. Monitoring data from a Common Information Model (CIM) . . 171
CIM configuration 174
Testing CIM attribute groups 174

Chapter 16. Monitoring a log file . . . 177
Log file parsing and separators 192
Testing log file attribute groups 193

Chapter 17. Monitoring an AIX Binary Log. 197

Chapter 18. Monitoring a Windows Event Log 201
Filtering by event type 206
Filtering by event source 207
Filtering by event identifier. 209

Chapter 19. Monitoring a command return code 213
Editing a command file definition 224
File Separation & Consolidation 225

Chapter 20. Monitor output from a script 227
Collecting script data from a remote system . . . 227
Script parsing and separators 227
Steps for monitoring output from a script 229

Chapter 21. Monitoring data from Java Database Connectivity (JDBC) 239
JDBC configuration 250
Stored procedures 251
Testing JDBC attribute groups 253

Chapter 22. Monitoring system availability by using Ping 255
Ping configuration files 257
Testing ping attribute groups. 259

Chapter 23. Monitoring HTTP availability and response time 261
HTTP tables 264
Monitoring a URL 267
Monitor `https://` URLs 269
Proxy server 269
HTTP configuration 269
Testing HTTP attribute groups. 270

Chapter 24. Monitoring data from a SOAP data source. 273
Specific fields for SOAP attributes 279
SOAP configuration 281
Testing SOAP attribute groups 282

Chapter 25. Monitoring data by using a socket 285
Sending socket information to the agent 291
Encoding of socket data 294
Socket errors 295
Socket configuration 295
Remote socket port connection 296
Sample script for socket 297
Testing socket attribute groups 298

Chapter 26. Use the Java API to monitor data. 301
Running the Java application 307
Generated sample Java application 308
Java API configuration 316
Testing Java application attribute groups 316

Chapter 27. Creating attribute groups from existing sources. 319
Joining two attribute groups 319
Manipulating attributes in joined attribute groups 322
Joined attributes 323
Creating a Filtered Attribute Group 324

Chapter 28. Creating a navigator group. 327

Chapter 29. Use of subnodes 331
Creating subnodes. 335
Creating a subnode by using the New Agent wizard 335
Creating a subnode from the data sources tree 339
Creating a subnode in the Agent Editor . . . 343
Subnode configuration 344

Chapter 30. Customizing agent configuration 361
Changing configuration properties by using the Agent Editor 369
Configuring a Windows remote connection . . . 372
Creating a user with Windows Management Instrumentation (WMI) permissions 373
Configuring a Secure Shell (SSH) remote connection 375

Chapter 31. Testing and debugging your agent 377
Attribute group testing 377
Attribute group testing - preferences 379
Attribute group testing - configuration 380
Full agent testing 382
Test Environment variables 386
Installing and testing an agent. 387
Installing the agent locally 387
Creating the agent image 389
Installing the package 390
Configuring and starting the agent 391
Agent post-generation and installation results . . 392
Uninstalling an agent. 400
Removing an agent by using the Tivoli Enterprise Portal 400
Removing an agent without using the Tivoli Enterprise Portal 401
Clearing the agent from the Tivoli Enterprise Portal 401

Chapter 32. Creating workspaces, Take Action commands, and situations 403
Creating situations, Take Action commands, and queries 403
Creating workspaces 403

Chapter 33. Importing application support files 409
Exporting and importing files for Tivoli Enterprise Monitoring Agents 409
Exporting and importing files for Tivoli System Monitor Agents. 414

Chapter 34. Event filtering and summarization 417
Controlling duplicate events 417
Viewing event filtering and summarization in the Tivoli Enterprise Portal 419

Chapter 35. Troubleshooting 425
Gathering product information for IBM Software Support 425
Agent Builder Trace logging 426
Logging configuration 428
Configuring and Tuning data collection 430
Data types 430
Environment variables 435
Informational, warning, and error messages overview 443
Message format 443
Agent messages 444
Agent trace logs 491
Setting RAS trace parameters 495
Problem classification. 497
Support information 532

Appendix A. Sharing project files . . . 533
Share a Solution Installer Project 535

Appendix B. Command-line options 537
Command - **generatelocal** 538
Command - **generatemappingfile** 538
Command - **generatezip** 539

Appendix C. Attributes reference . . . 541
Availability node 541
Performance Object Status node 545
Thread Pool Status attribute group 551
Event log attribute node. 554
Log File Summary. 556
AIX Binary Log attribute group 558
Monitor and Notification attribute groups 561
Counter Notifications. 562
Gauge Notifications 564
Registered Monitors 567
String Notifications 568
SNMP Event attribute groups 570
JMX Event attribute groups. 571
Ping attribute group 573
HTTP attribute groups 576
Discovery attribute groups 580
Take Action Status attribute group 582
Log File Status attribute group 585
Log File RegEx Statistics attribute group 589

Appendix D. Creating application support extensions for existing agents 595
Creating an Application Support Extension project 597
Adding support files to a project 599
Generating the Application Support Extension installation image 606
Installing your Application Support Extension . . 607
Converting a **Solution Install Project** to an Application Support Extension project 607

Appendix E. Cognos data model generation. 609
Prerequisites to generating a Cognos data model 609
Tivoli Data Warehouse 609
Tivoli Common Reporting 612
Framework Manager 612
Creating reports 613
Populating the ManagedSystem Table 618
Exporting reports and data models from Tivoli Common Reporting 620

Importing reports into Agent Builder 622
Installing reports from an agent package into Tivoli Common Reporting 624

Appendix F. Upgrading custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6.2 agents 625

Appendix G. ICU regular expressions 627

Appendix H. Creating Non-agent file bundles 631
Remote Deploy Bundle Editor 633
Adding commands to the bundle. 635
Adding prerequisites to the bundle 636
Adding files to the bundle 637
Generating the bundle 639
Creating deployable bundles for Tivoli Netcool/OMNIbus probes 641

Appendix I. Dynamic file name support 643

Appendix J. SNMP trap configuration 647

Appendix K. Take Action commands reference 651
SSHEXEC action 651

Appendix L. Documentation library 653
IBM Tivoli Monitoring library 653
Documentation for the base agents 654
Related publications 655
Other sources of documentation 655

Notices 657
Trademarks 658

Index 661

Figures

1. Create New Agent icon 14
2. Starting the wizard with the **File** > **New** > **IBM Tivoli Monitoring Agent** option. . . . 15
3. Starting the wizard with the File > New > Other option 16
4. Select a wizard page 17
5. Selecting the agent icon 18
6. Welcome window 19
7. Project page 20
8. General Information page 21
9. Agent Information page 22
10. Agent Initial Data Source page 24
11. Select key attributes page 27
12. Data Source Definition page: defining more monitors 28
13. Data Source Location page 29
14. Subnodes in the Navigator tree 32
15. Move existing data sources 33
16. Currently Defined Data Sources page 34
17. Choosing Reset from the Tivoli Monitoring perspective. 35
18. Agent Information page 36
19. Default Operating Systems page 38
20. Self-Describing Agent page 39
21. Environment Variables page 40
22. Environment Variable Information window 40
23. Watchdog Information page 41
24. Cognos Information page 43
25. Data Source Definition page for Cognos 44
26. Data Source Definition page 47
27. Runtime Configuration Information page 49
28. Agent XML Editor page 50
29. Updating the agent version after commit 52
30. Agent Product Code change 53
31. Agent support files invalidated 53
32. Attribute Information page 57
33. String attribute type. 59
34. Numeric attribute type. 60
35. Defining enumeration for an attribute. . . . 62
36. Attribute Information page 64
37. Attribute Information page with Derived Attribute Details tab open. 65
38. Derived Formula Editor window (default) 69
39. Derived Formula Editor view with formula hierarchy tree view 70
40. Edit the Selected Attribute window 71
41. Edit the Selected Attribute window with error 72
42. Edit the Selected Function page with warning 73
43. Edit the Selected Number page with a number selected.. 74
44. Edit the Selected Operator page. 75
45. Edit the Selected Operator page (for a conditional operator) 76
46. Edit the Selected Function page 77
47. Insert window example 78
48. Insert window example for `getenv` function 79
49. Remove function and its Arguments page 79
50. Empty formula example 80
51. Parsing error example 81
52. Insert window for fixing a formula manually 82
53. Operating systems where the agent is to run 88
54. Adding a server process 89
55. Process Monitor page 90
56. Browse the list of currently running processes 91
57. Search for a process. 92
58. Process Monitor page example 93
59. Select a connection type or connection template for process browsing 94
60. Edit connection properties for Tivoli Enterprise Portal Server Managed System 95
61. Adding a Windows service monitor 97
62. Service Monitor page 98
63. Browse the list of all services defined in the system 99
64. Search for a service 100
65. Service information 101
66. Select a connection type or connection template for service browsing 102
67. Edit connection properties for Tivoli Enterprise Portal Server Managed System . . 103
68. Adding WMI data 106
69. WMI Information page 107
70. List of classes with their associated attributes 108
71. Optionally specifying WMI search options 108
72. WMI Information page completed 109
73. WMI shown on Data Source Definition page in the Agent Editor 110
74. Global Windows Data Source Options window 111
75. WMI Test window 112
76. Adding Perfmon data. 113
77. Perfmon Information page 114
78. Browsing for Windows Performance Monitor objects 115
79. Performance Monitor Search window 115
80. Perfmon Information page with Object Name 116
81. Global Windows Data Source Options window 116
82. Perfmon Data Source Definition 118
83. Perfmon Test window. 119
84. Adding SNMP data 122
85. Simple Network Management Protocol Information page 123
86. List of objects 124
87. Selecting an object 125
88. Simple Network Management Protocol Information page 126
89. SNMP Data Source Definition 127
90. MIB Parsing window 129
91. SNMP Attribute Group Test selection 130
92. SNMP Test settings window 131
93. Create Connection Wizard 131

© Copyright IBM Corp. 2007, 2013

94. Adding SNMP events 134
95. Simple Network Management Protocol Event Information window 135
96. SNMP MIB Browser window 136
97. Select key attributes page 138
98. Runtime Configuration page 139
99. Test Event Settings window 141
100. Test Event Settings window that shows collected SNMP event data 142
101. Data Collection Status window 142
102. Adding JMX data 144
103. JMX Information page 145
104. JMX Agent-Wide options window. 146
105. JMX Browser with no connection selected 147
106. Connection Type page 148
107. JMX connection properties 149
108. Java Management Extensions (JMX) Browser window 150
109. JMX Information page 152
110. JMX Agent-Wide Options window 153
111. New attribute group in Agent Editor 154
112. JMX Agent-Wide Options window 155
113. Runtime Configuration tab of the Agent Editor 156
114. JMX Agent-Wide Options page 160
115. Attribute Information page 166
116. JMX Test window 169
117. Adding CIM data 171
118. Common Information Model (CIM) Information page 172
119. Common Information Model (CIM) Class Browser window 173
120. CIM Test Settings window 175
121. Adding a log file 177
122. Log File Information page 178
123. XML Browser window 179
124. Advanced Data Source Properties page, File Information 180
125. Advanced Data Source Properties page, Record Identification 183
126. Advanced Data Source Properties page, Filter Expression tab 184
127. The Event Information tab of the Advanced Data Source Properties page, ... 185
128. Attribute Information page 186
129. Advanced Log File Attribute Information page 187
130. Add Filter window 188
131. Add Filter example 1 190
132. Add Filter example 2 191
133. Example attribute value output when Agent parses a simple log file data row. 193
134. Example attribute value output when Agent parses a complex log file data row. 193
135. Parse Log window 194
136. Parse Log window that shows parsed log file attribute values 195
137. Data Collection Status window 195
138. Adding an AIX Binary Log 198
139. Binary Log Information page 199
140. Adding a Windows Event log 202
141. Windows Event Log page 203
142. Windows Event Log Bookmark Settings window 204
143. Global Windows Data Source Options window 205
144. Windows Event Log page 206
145. Event Source window. 207
146. Event Log Source Browser window 208
147. Windows Event Log window 209
148. Event Identifier window 210
149. Windows Event Log window 211
150. Adding data from a command 214
151. Command Return Code page 215
152. Command Information window 216
153. Return Code Definition window: Return code type. 217
154. Return Code Definition window: Return code value 218
155. Messages window 219
156. Message Definition window 219
157. Return Code Definition window completed 220
158. Command Information window completed 221
159. Return Code Definition window: Message text 222
160. Messages window 223
161. Return Code Definition window 223
162. Command Return Code page completed 224
163. Import Command File window 225
164. Example attribute value output when Agent parses complex script output. 229
165. Adding data from a script 230
166. Command List page 231
167. Command Information window for a script 233
168. Test Command window 234
169. Attribute Information page 236
170. Agent Editor Data Sources Definition page 237
171. Adding JDBC data 240
172. JDBC Information page 241
173. JDBC Browser window 242
174. Database Connection Wizard: JDBC Connectionspage 243
175. Database Connections Wizard: Connection Properties page 244
176. Java Database Connectivity (JDBC) Browser window 245
177. JDBC Information page 246
178. Data Source Definition page Attribute Group Information 247
179. Runtime Configuration Information page, Configuration for Java virtual machine (JVM) . 248
180. Attribute Information page 249
181. Test JDBC statement window 254
182. Adding Ping data 256
183. Ping Information window 257
184. Ping Test Settings window 259
185. Adding HTTP data 262
186. HTTP Information window 263
187. Attribute Information page 265
188. HTTP Test window 271
189. Adding HTTP data 274
190. SOAP Information page 275

191. SOAP Browser window 276
192. SOAP Browser window 277
193. SOAP Browser window 278
194. Attribute Information page 279
195. Test SOAP Collection window 283
196. Monitoring data with Socket 286
197. Socket Information window. 287
198. Global Socket Data Source Information page 288
199. Socket Error Code Definition window, 289
200. Messages window 289
201. Message Definition window 290
202. Test Socket Client window 298
203. Monitoring data with Java API. 302
204. Java API Information window 303
205. Global Java API Data Source Information page 304
206. Java API Error Code Definition window, 305
207. Messages window 305
208. Message Definition window 306
209. Sample agent structure 308
210. Test Java Client window 317
211. Attribute Group Information pageAttribute Group Information window. 322
212. Locating source attribute information 324
213. Adding existing data sources 325
214. Filter Information window 326
215. Selecting the initial source of monitoring data for a new agent 328
216. Providing the navigator group name and help text 329
217. Selecting the first source of monitoring data 330
218. Subnodes monitoring different systems 331
219. Subnode types in Navigator tree 332
220. Monitoring multiple subnode instances of the same subnode type 334
221. Example: data collection in a subnode 335
222. Data Source Location page - Creating a subnode 336
223. Subnode Information page 337
224. Initial Subnode Data Source page 338
225. Subnode Configuration Overrides page 339
226. Data Source Definition page 340
227. Defining a new subnode 341
228. Initial Subnode Data Source page 342
229. Subnode Configuration Overrides page 343
230. Agent Editor Data Source Definition page 344
231. Subnode Configuration Overrides window 346
232. Overriding custom configuration 347
233. Select Configuration Section window 348
234. Override configuration lists 349
235. Sections to override 350
236. SNMP Version 1 Properties expanded 351
237. Configuration property definitions in the Agent Builder 353
238. **Top** section with agent-level configuration for the **Agent Cfg** property 354
239. **Main** section with the agent-wide default value for the **Overridable Cfg** property. . . 355
240. **Example Subnode** section page with no subnode 356
241. **Example Subnode** section page with two subnode instances defined 357
242. Windows Management Instrumentation (WMI) Information page 358
243. Global Windows Data Source Options 359
244. **Insert Configuration Property** on the Log File Information page 362
245. Initial Configuration Properties window 363
246. Runtime Configuration Property window 364
247. Configuration Property Default Values window 365
248. Adding a configuration property choice 366
249. New configuration property displayed 367
250. Adding a second property to an existing section 368
251. Property inserted into the log file name 369
252. Runtime Configuration Information page in the Agent Editor 370
253. Editing a configuration section in the Agent Editor 371
254. Editing a configuration property in the Agent Editor 372
255. WMI Test window 378
256. WMI Test window that shows test log location. 379
257. Environment Variables window 381
258. Runtime Configuration window 381
259. Java Information window 382
260. Test Agent section of the Agent Editor, Agent Information page. 383
261. Agent Test menu. 383
262. Agent Test view with example subnode and navigator group highlighted. 384
263. Agent Test perspective 385
264. The Attribute Group Test view that shows more information (Performance Object Status) about data collections for the **Managed_URLs** and **Managed_Nodes** attribute groups . . . 386
265. Generate agent icon 388
266. Generate agent menu option 388
267. Create an agent image 390
268. Manage Tivoli Enterprise Monitoring Services window 396
269. Nodes for attribute groups you defined 397
270. Availability node 398
271. Performance Object Status node 399
272. Event log node 400
273. Setting the `sysadmin` user ID 404
274. Setting the `sysadmin` user ID (continued) 405
275. Setting the `sysadmin` user ID (continued) 406
276. Setting workspace properties 407
277. Setting workspace properties (continued) 408
278. Importing application support files 410
279. Importing situations 411
280. Selecting situations. 412
281. Take Actions window 413
282. Importing queries 414
283. Advanced Data Source Properties page. 419
284. Historical view and cache view when event filtering or summarization is not enabled . . 420

285. Historical view and cache view when **Only send summary events** is selected 421
286. Historical view and cache view when **Send all events** is selected 422
287. Historical view and cache view when **Send first event** is selected 423
288. Historical view and cache view when **Event threshold** is selected 424
289. IBM Java Logging Toolkit 429
290. Importing files 533
291. Importing files (continued) 534
292. Importing files (continued) 535
293. IBM Tivoli Monitoring Application Support Extension 596
294. Naming the project 597
295. IBM Tivoli Monitoring Application Support Extension 598
296. Naming the project 599
297. Importing support files 600
298. Import Information 601
299. Affinities list 602
300. Import Information complete 602
301. Importing situations 603
302. Importing queries 604
303. Import Take Action commands. 605
304. Agent support files 605
305. Application Support Extension Information window 606
306. Selecting agent project file 614
307. Selecting Publish Packages 615
308. Selecting Common Reporting 616
309. Selecting Report Studio 617
310. Report Studio 618
311. The Content Administration tab 621
312. The Content Administration tab with agent package listed 622
313. Importing report packages 623
314. Import Report Package window 624
315. Select a wizard window 631
316. **New Remote Deploy Project** window 632
317. **Remote Deploy Bundle Information** window 633
318. Remote Deploy Bundle Editor 634
319. Command window 636
320. New Prerequisite window 637
321. Adding files to the bundle or generating a bundle from the Navigator tree 638
322. Import Bundle Files window 639
323. Generate Final Remote Deploy Bundle window 640
324. Examples of configuration record types 2 and 3. 648

Tables

1. Quick-reference information for creating agents 3
2. Quick-reference information for other functions 4
3. Guidelines for installation and configuration 5
4. Location of installation log files 7
5. Minimum version of IBM Tivoli Monitoring required by the agent 23
6. Fields for editing data sources 55
7. Fields and options for defining attributes 58
8. Numeric attribute options 60
9. Valid format parameters for StringToTivoliTimestamp 84
10. StringToTivoliTimestamp examples 85
11. Fields on the Process Monitor page 90
12. SNMP Events configuration properties 139
13. Filter options 244
14. Supported SQL data types for use with a monitoring agent 249
15. Network Management configuration properties 259
16. HTML elements searched for objects to monitor 264
17. HTTP Attribute Information - Managed URLs 266
18. HTTP Attribute Information - URL Objects 267
19. URLs file entries 268
20. URL Monitoring configuration properties 269
21. Proxy Server configuration properties 269
22. Java configuration properties 270
23. SOAP Attribute Information 280
24. HTTP Server configuration properties 282
25. File types for supplemental files 290
26. Sample error code 293
27. Characters to encode in attribute values 294
28. Performance Object Status values 295
29. Socket configuration property 296
30. File types for supplemental files 306
31. Java trace level options 310
32. The data types of attribute fields and their IBM Tivoli Monitoring attribute type equivalents 310
33. Internal error codes for the agent 314
34. Changes to an agent that require modifications to the Java source 315
35. Java configuration properties 316
36. source attribute group one (single row) 320
37. source attribute group 2 (single row) 320
38. Resulting join 320
39. source attribute group one (single row) 320
40. source attribute group two (more than one row) 320
41. Resulting join 320
42. source attribute group one (more than 1 row) 321
43. source attribute group 2 (more than 1 row) 321
44. Resulting join (joining on Attribute3 and Attribute7) 321
45. Environment variables 386
46. Information to gather before you contact IBM Software Support 425
47. Environment variables 435
48. Trace log files for troubleshooting agents 493
49. Problems and solutions for installation and use of the Agent Builder 499
50. Problems and solutions for agents 510
51. Problems and solutions for Report Creation 529
52. Command quick-reference table 537
53. Required arguments 619
54. Optional arguments 619
55. Regular expression metacharacters 627
56. Regular expression operators 628
57. Replacement text characters 630
58. Flag options 630
59. Predefined Variables for Commands 636
60. Categories supported by the SNMP Data Provider 648
61. Severities supported by the SNMP Data Provider 649
62. Statuses supported by the SNMP Data Provider 649
63. Source IDs supported by the SNMP Data Provider 649

© Copyright IBM Corp. 2007, 2013

Chapter 1. Introduction

An overview of the IBM® Tivoli® Monitoring Agent Builder.

About Tivoli Monitoring Agent Builder

Tivoli Monitoring Agent Builder is a set of tools that are used to create agents, installation packages for the created agents, and application support extensions for existing agents.

Creating agents

Using the Tivoli Monitoring Agent Builder, you can quickly create, modify, and test an agent. Agents collect and analyze data about the state and performance of different resources, such as disks, memory, processor, or applications. The Builder creates a data provider that you can use to monitor three types of data:

Availability
Process and service availability and command return codes

Windows Event Log
Specific information from the Windows Event Log

External data sources
Data from external sources such as:

- Windows Management Instrumentation (WMI)
- Windows Performance Monitor (Perfmon)
- Simple Network Management Protocol (SNMP)
- SNMP Events
- Hypertext Transfer Protocol (HTTP)
- SOAP
- Java™ Database Connectivity (JDBC)
- Java application programming interface (API)
- Internet Control Message Protocol (ICMP) ping
- Java Management Extensions (JMX)
- Common Information Model (CIM), external scripts
- Log files
- Socket
- Java application programming interface (API)

Creating installation packages for agents

You can package agents for easy distribution by using a compressed image, which includes installation scripts to manage the installation of the agent and support files.

Creating custom support extensions for existing agents

Another feature of the builder is the ability to package and distribute custom support extensions for existing agents. Use this feature to develop new situations, queries, workspaces, and Take Actions for an existing Tivoli Monitoring V6.x agent. For example, if you have insight into how the retail industry uses IBM DB2® for an in-store database. You can create customized workspaces and situations that can then be given or sold to companies in this industry.

© Copyright IBM Corp. 2007, 2013

Creating reports for agents

You can use Agent Builder to generate a Cognos® data model which you can use to build Tivoli Common Reporting reports. These reports can be packaged as part of your agent image.

About Eclipse

The Tivoli Monitoring Agent Builder is an Eclipse application for the Eclipse 3.6.2 platform.

Eclipse is an open source framework for the construction of powerful software development tools and rich desktop applications. Technology providers can use the Eclipse plug-in framework to integrate technology on the desktop. These providers can then save time and money by focusing effort on delivering differentiation and value for their offerings. Eclipse is a multi-language, multi-platform, multi-vendor supported environment that an open source community of developers built. It is provided royalty-free by the Eclipse Foundation. Written in the Java language, Eclipse includes extensive plug-in construction toolkits and examples. Eclipse can be extended and run on a range of desktop operating systems that includes Windows, Linux, QNX, and Macintosh OS X. To see full details about Eclipse and the Eclipse Foundation, go to http://www.eclipse.org.

New in this release

For version 6.3 of the Tivoli Monitoring Agent Builder, the following enhancements were made since version 6.2.3.1:

- Full agent test: You can test the full agent in the Agent Builder without the need for a Tivoli Monitoring Infrastructure. This feature enhances the attribute group test function that was introduced in version 6.2.3.1. You can use full agent test to select an agent and test all of its attribute groups. For more information, see "Full agent testing" on page 382.
- New derived formula functions, `floor`, `ceiling`, `rate`, and `delta` are added. For more information, see "Formula operators and functions" on page 82. The `rate` and `delta` functions are in addition to the `rate` and `delta` attribute types that are already supported by Agent Builder.
- When you create a log file attribute group, you can search for log files to monitor not only with regular expressions in the file name but also with regular expressions in one subdirectory of the path. For more information, see Chapter 16, "Monitoring a log file," on page 177.
- Socket data sources: You can now configure your agent to accept data from a remote socket port. For more information, see "Remote socket port connection" on page 296.
- Ping data sources: When ping device definition files are changed, the agent reinitializes its configuration dynamically, without requiring a restart. For more information, see "Ping configuration files" on page 257.
- Agent Builder now runs with Java version 7. All of the Java data providers, including JMX, JDBC, HTTP, and Java API require at least Java version 5.
- Agent Builder now uses a newer version of the Global Secure Toolkit (GSKit) API for encryption in the script data provider. For more information, see the restriction noted at "Collecting script data from a remote system" on page 227.

About this documentation

The IBM Tivoli Monitoring Agent Builder User's Guide describes how to use the Tivoli Monitoring Agent Builder software to create, modify, debug, and package agents for use with Tivoli Monitoring products.

The User's Guide explains concepts that agent developers, system administrators, and application administrators must know to use the Tivoli Monitoring Agent Builder product. The guide also explains the integration of Tivoli Monitoring Agent Builder into the Tivoli environment.

Chapter 2. Getting started quick-reference guide

A quick-reference table of the procedures you can perform with Tivoli Monitoring Agent Builder

The two main objectives for using the Agent Builder are creating agents and creating application support extensions. Application support extensions are created by creating workspaces and situations to enhance one or more existing agents. The following tables include cross-references to the locations of further information about the procedures. The procedures are listed in a suggested order, but do not have to be followed in this order.

The following table contains the procedures for creating agents:

Table 1. Quick-reference information for creating agents

Goal	Refer to
1. Learn the preparation steps for creating an agent: • Review basic concepts about the Agent Builder and Eclipse. • Ensure that your environment meets requirements for software and authorization. • Install the Agent Builder. • Obtain a basic understanding of the types of data that the agent can monitor. • Organize data within the agent.	• Chapter 1, “Introduction,” on page 1 • “Requirements” on page 5 • “Authorization” on page 6 • “Builder installation procedure” on page 6 • Chapter 4, “Starting Agent Builder,” on page 11 • “Defining initial data source” on page 23 • Chapter 28, “Creating a navigator group,” on page 327 • Chapter 29, “Use of subnodes,” on page 331
2. Learn how to create an agent by using the Agent Builder wizard.	• Chapter 4, “Starting Agent Builder,” on page 11 • Chapter 5, “Creating a basic agent,” on page 13
3. Learn how to test and debug your created agent and the availability of your monitors.	• Chapter 31, “Testing and debugging your agent,” on page 377 • Appendix B, “Command-line options,” on page 537 • Chapter 35, “Troubleshooting,” on page 425 • “Preparing your agent for modifications or for regeneration” on page 34 • Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35.
4. Learn how to remove an agent that you created with the Agent Builder.	• “Uninstalling an agent” on page 400
5. Learn how to create workspaces and situations for your agent.	• Chapter 32, “Creating workspaces, Take Action commands, and situations,” on page 403 • Chapter 33, “Importing application support files,” on page 409
6. Learn how to create reports for your agent.	• Appendix E, “Cognos data model generation,” on page 609

The following table contains the procedures for creating other functions:

© Copyright IBM Corp. 2007, 2013

Table 2. Quick-reference information for other functions

Goal	Refer to
1. Learn how to create custom workspaces, situations, and queries.	• Chapter 32, "Creating workspaces, Take Action commands, and situations," on page 403
2. Learn how to package your application support extension.	• Appendix D, "Creating application support extensions for existing agents," on page 595
3. Learn how to build custom bundles.	• Appendix H, "Creating Non-agent file bundles," on page 631

Chapter 3. Installing the Tivoli Monitoring Agent Builder

Before you install the Tivoli Monitoring Agent Builder, check that the prerequisites summarized in Table 3 are met.

Note: For information about installing or modifying an *agent*, see "Installing and testing an agent" on page 387.

Table 3. Guidelines for installation and configuration

Goal	See
Verify that your environment meets the requirements.	"Requirements"
Ensure that you have appropriate authorization.	"Authorization" on page 6

Requirements

To install and run Agent Builder and agents that are created with Agent Builder, your system must meet certain requirements.

Installing Agent Builder

To install the Agent Builder, you must have:

- A computer with approximately 400 MB of space, plus the space that is required for the agents you develop.
- Agent Builder is supported on the Linux Intel 32-bit and x86 64-bit, Windows, and AIX® Operating Systems. For the most current information about system requirements, see the Software Product Compatibility Reports. For more information about running a software product compatibility report specific to Agent Builder, see "Generate a software product compatibility report for Agent Builder" on page 6.

  Note: The Linux Agent builder requires the installation of the latest 32-bit versions of the following packages:

  ```
  libstdc++
  libgcc
  libXp
  gtk2
  ```

  Further, Red Hat Enterprise Linux requires:

  ```
  compat-libstdc++-33
  libXtst
  ```

  and SUSE Enterprise Linux requires:

  ```
  compat-libstdc++
  xorg-x11-libs
  ```

  These packages are available on the Linux operating system installation media and Service Packs. Each package can have multiple prerequisites which all must be installed.

Running Agent Builder

To run Agent Builder help, on an AIX system, you must install an external web browser such as Firefox.

© Copyright IBM Corp. 2007, 2013

Note: After you install a web browser as the default browser on your AIX system, if Agent Builder does not display any help panels, you must set the default browser in Agent builder. For more information, see "Set the default browser in Agent Builder" on page 9.

Agent Builder agents; supported operating systems

See "Operating system requirements" on page 13 for information about the operating systems that are supported by the agents that are created with Agent Builder.

Generate a software product compatibility report for Agent Builder

Use this procedure to generate a Software product compatibility report for Agent Builder.

About this task

Software Product Compatibility Reports are available on the IBM Support Portal. The Software Product Compatibility Reports is a tool designed to allow you to easily generate custom reports about compatible IBM software combinations

Procedure

1. Go to the Software Product Compatibility Reports site at Software product compatibility reports.
2. Select **Create a report** under **In-depth reports**.
3. Enter `Tivoli Monitoring` in the **Full or partial product name:** field.
4. Click the search icon.
5. Select **Tivoli Monitoring** from the **Search results:** area.
6. Select a version from the **Version** list.
7. Under **Scope of report**, select **Product component**.
8. Under **Select component**, select **Agent Builder Server**.
9. Click **Submit**. The report is displayed in a new browser window.

Authorization

Before you can run Tivoli Monitoring Agent Builder, you must have Administrator authorization.

Administrator authorization is required for the following reasons:

- Ability to ensure a consistent running environment for the Agent Builder and the agents that are designed with it.
- Ability to access parts of the operating system, which requires Administrator authorization. The Agent Builder is installed on a development system and is designed for developers who require this type of access.

Builder installation procedure

Use this procedure to install the Tivoli Monitoring Agent Builder.

Before you begin

Before you install the Tivoli Monitoring Agent Builder, uninstall any previous versions. For more information about uninstalling, see ("Uninstalling Agent Builder" on page 8). None of your existing agent information is lost when you uninstall.

About this task

Procedure

1. Download and extract the Tivoli Monitoring Agent Builder V6.3 image.
2. Run the executable file for your operating system to start the installation:
   - On Windows: `setup.bat`
   - On UNIX: `setup.sh`

   Note: Run the installation program from the same user ID from which you intend to run the Agent Builder.
3. When the Tivoli Monitoring Agent Builder window opens, select your language, and click **OK**.
4. On the **Introduction** page, click **Next**.
5. On the **Software License Agreement** page, click **I accept the terms in the license agreement**, and click **Next**.
6. On the **Choose Install Folder** page, click one of the following options:
   - **Next** to install the Agent Builder to the directory specified in the **Where Would You Like to Install?** field.
   - **Restore Default Folder** to install the Agent Builder in a default directory.
   - **Choose** to select a different directory.

   Note: The directory name that you choose must not contain the following characters, having these characters in the directory path prevents the agent builder from starting:
   !
   #
   %
   ;
7. On the Pre-Installation Summary page, click **Install**.
8. On the Installing Tivoli Monitoring Agent Builder page, you can click **Cancel** if you want to cancel the installation. If you want to continue installing, wait for the **Install Complete** page to open. And then, click **Done**.

Results

On Windows systems, after the Agent Builder is installed, an option is added to the Start menu. Also, an Agent Builder icon is added to your desktop.

On UNIX systems, after the Agent Builder is installed, the Agent Builder executable file is named *Install_Location*/agentbuilder.

The log files for the Agent Builder installation are in the following locations:

Table 4. Location of installation log files

Operating System	Agent Builder installation log file location
Windows	*install_dir*\IBM_Tivoli_Monitoring_Agent_Builder_InstallLog.xml
Linux, AIX	*install_dir*/IBM_Tivoli_Monitoring_Agent_Builder_InstallLog.xml

The log is not written until you click **Done** because the log file traces actual installation.

Silent installation

You can install the product by using a silent installation method.

About this task

The silent installation options file, `installer.properties`, is included on the installation media at the root of the installation directory. You must modify this file to meet your needs.

Procedure

Start the silent installation by running the following command:

```
setup.[bat | sh] -i silent -f path/installer.properties
```

Where *path* is a fully qualified path to the `installer.properties` file (including the drive letter or UNC path name on Windows). The value that you choose for *path* cannot contain spaces.
Example of an Options file:

```
# ----------------------------------------------------------------------
# IBM Tivoli Monitoring Agent Builder
#
# (C) Copyright IBM Corporation 2009.  All rights reserved.
#
# Sample response file for silent install
#
# To use this file, use the following command:
#
# Windows:
#    setup.bat -i silent -f <path>\installer.properties
#
# Linux and AIX:
#    setup.sh -i silent -f <path>/installer.properties
#
# Where
#    <path> is a fully-quailfied path to the installer.properties
#    file (including the drive letter or UNC path name on Windows).
#    <path> cannot contain spaces.
# ----------------------------------------------------------------------


# ----------------------------------------------------------------------
# This property indicates that the license has been accepted
# ----------------------------------------------------------------------
# LICENSE_ACCEPTED=FALSE

# ----------------------------------------------------------------------
# This property specifies the install directory
#
# On Windows, the default is:
#     C:\\Program Files\\IBM\\ITM\\AgentBuilder
#
# On Linux and AIX, the default is:
#     /opt/IBM/ITM/AgentBuilder
# ----------------------------------------------------------------------
#USER_INSTALL_DIR=C:\\Program Files\\IBM\\ITM\\AgentBuilder
#USER_INSTALL_DIR=/opt/IBM/ITM/AgentBuilder
```

Uninstalling Agent Builder

Before you install Agent Builder, uninstall any earlier versions.

Procedure

- On AIX and Linux systems, run the following command:
  1. *INSTALL_DIR*`/uninstall/uninstaller`

     where *INSTALL_DIR* is the name of the directory where Agent Builder is installed.
- On Windows systems, do the following steps:

1. From the Windows Control Panel, select **Add/Remove Programs**.
2. Click **IBM Tivoli Monitoring Agent Builder**.
3. Click **Change/Remove**.

- On Windows 7 and Windows Server 2008 R2, do the following steps:
  1. Open Windows Programs and Features by selecting **Start** > **Control Panel** > **Programs** > **Programs and Features**.
  2. Select **IBM Tivoli Monitoring Agent Builder** from the list of installed programs.
  3. Click **Uninstall/Change**.
  4. Click **Uninstall** on the Uninstall IBM Tivoli Monitoring Agent Builder page.
  5. Click **Done** on the Uninstall Complete page.

  Note: On Windows 7 and Windows Server 2008 R2, you can also go to the Windows Programs and Features window by selecting **Start** > **Computer** > **Uninstall or change a program**. Then, continue from step b.

Silent uninstallation

You can use the silent uninstallation method to uninstall.

Procedure

Start the silent uninstallation by running the following command:

```
INSTALL_DIR/uninstall/uninstaller[.exe] -i silent
```

Set the default browser in Agent Builder

On some AIX and Linux systems, you might have to set the Agent Builder default browser so that help panels are displayed.

Procedure

1. Open the Agent Builder preferences panel by selecting **Window** > **Preferences**.
2. Select and expand the **General** node.
3. Select **Web Browser**.
4. Select **Use external web browser** if it is not already selected.
5. Select the web browser that you want to use from the available check boxes.
6. Optional: To add a web browser, use the following steps
   a. Click **New**.
   b. Enter a descriptive name for the browser in the **Name** field.
   c. Enter the full path to the browser executable file in the **Location** field.
   d. Click **OK**.
7. Click **OK**.

Chapter 4. Starting Agent Builder

Plan your agent and start Agent Builder.

The Tivoli Monitoring Agent Builder provides a wizard that can help you create an agent. Before you run the wizard, ensure that you make the following planning decisions:

- The project name (see "Naming your agent" on page 19)
- The types of monitoring data that you want your agent to collect (see "Defining initial data source" on page 23)

Creating your agent involves several steps:

1. Starting (launching) the Agent Builder
2. Creating an agent
3. Defining one or more types of data for your agent to monitor. When you indicate the types of data that the agent collects, the wizard guides you through the process of configuring each of the data collection methods. Types of data are availability, log events, or external data sources.
4. Testing your agent (see Chapter 31, "Testing and debugging your agent," on page 377)
5. Optionally add workspaces, situations, and queries (see Chapter 33, "Importing application support files," on page 409)

Restriction:

Agent builder supports the entry of only printable ASCII characters in display fields where you enter text. For example, attribute group names, attribute names, agent names, subnode names, help text. Printable ASCII characters means ASCII printable character codes 32 - 126. Japanese or any non-ASCII characters in the default (English) resource bundle are not supported. As a result the text displayed in the Tivoli Enterprise Portal is in printable ASCII characters only.

The reason for this restriction is that Agent Builder does not build language packs. Agent builder builds the default Java® resource bundle which means only printable ASCII characters can be entered in Agent Builder display fields.

This restriction does not apply to the actual data that the agent is processing.

Starting the Agent Builder

You can start Agent Builder in a number of ways.

Procedure

Start the Agent Builder by using one of the following methods

- Windows: Use one of the following methods
  - From a command-line type: *Install_Location*\agentbuilder.exe.
  - Select **Start** > **All Programs** > **IBM Tivoli Monitoring** > **Agent Builder**.
  - Click the **Agent Builder desktop icon**.
- All other supported operating systems, From a command-line type: *Install_Location*/agentbuilder

Note: When you initially run the Agent Builder, you are prompted for the location of your workspace directory. The files that create your agents are saved in that directory. You can designate any directory as your workspace.

© Copyright IBM Corp. 2007, 2013

Chapter 5. Creating a basic agent

Information about using the Agent Builder to create a basic agent

Use the following topics to learn how to create a basic agent with Agent Builder:

- "Operating system requirements"
- "Starting the new agent wizard"
- "Creating and defining the agent" on page 19
  - "Naming your agent" on page 19
  - "Defining initial data source" on page 23
  - "Organizing the agent" on page 29
- "Preparing your agent for modifications or for regeneration" on page 34

Operating system requirements

Agents that are created by agent builder are supported on various operating systems.

Agents that are created by the Agent Builder are supported on all operating systems that are supported by the OS monitoring agents, except z/OS® and i5/OS™. Supported operating systems are as follows:

- AIX
- HP-UX
- Linux
- Solaris
- Windows

For more detail, including version numbers, see the User's Guides for the Tivoli Monitoring Operating System agents (OS agents).

To run your monitoring agent, install the appropriate OS agent on the same computer where your monitoring agent runs. Install Tivoli Monitoring V6.2 or later in your environment. Tivoli Monitoring is not required to be on the same computer as your monitoring agent.

Note: Agent Builder browsers operate on the data sources and information accessible from the system on which the Agent Builder is run. Ensure that you run the Agent Builder on either of the following types of systems:

- A system that runs on the same level as the operating system and monitored applications for which you are developing the agent
- A system that connects to one that runs on the same level as the operating system and monitored applications for which you are developing the agent

Starting the new agent wizard

You can start the new agent wizard in a number of ways.

To start the wizard and create the agent, you can use either of the following methods:

- Use the **Create New Agent** icon on the toolbar. For more information, see "Use the icon to start the new agent wizard" on page 14.
- Use the **File** option on the Agent Builder menu bar. For more information, see "Create a file to start the new agent wizard" on page 14.

© Copyright IBM Corp. 2007, 2013

Note: You might not see the wizard from the new menu if you are not using the IBM Tivoli Monitoring perspective in Eclipse. To open the IBM Tivoli Monitoring perspective click **Window** > **Open Perspective** > **Other**. Then, select **IBM Tivoli Monitoring.** and click **OK**. The IBM Tivoli Monitoring perspective also opens when you end the wizard, along with the Project Explorer view, the Outline view, and the Problems view.

Use the icon to start the new agent wizard

You can start the new agent wizard by using a toolbar menu icon.

Procedure

Click the **Create New Agent** icon (which resembles a pencil) on the toolbar.
The Welcome window (Figure 6 on page 19) for creating an agent is displayed.

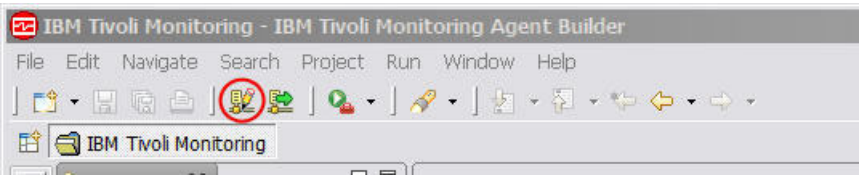


Figure 1. Create New Agent icon

What to do next

Click **Next** to choose a name for your agent. For more information, see ("Naming your agent" on page 19).

Create a file to start the new agent wizard

You can start the new agent wizard by creating a file.

About this task

Select one of the following ways to create a file:

Procedure

- From the Main menu, select **File** > **New** > **IBM Tivoli Monitoring Agent**

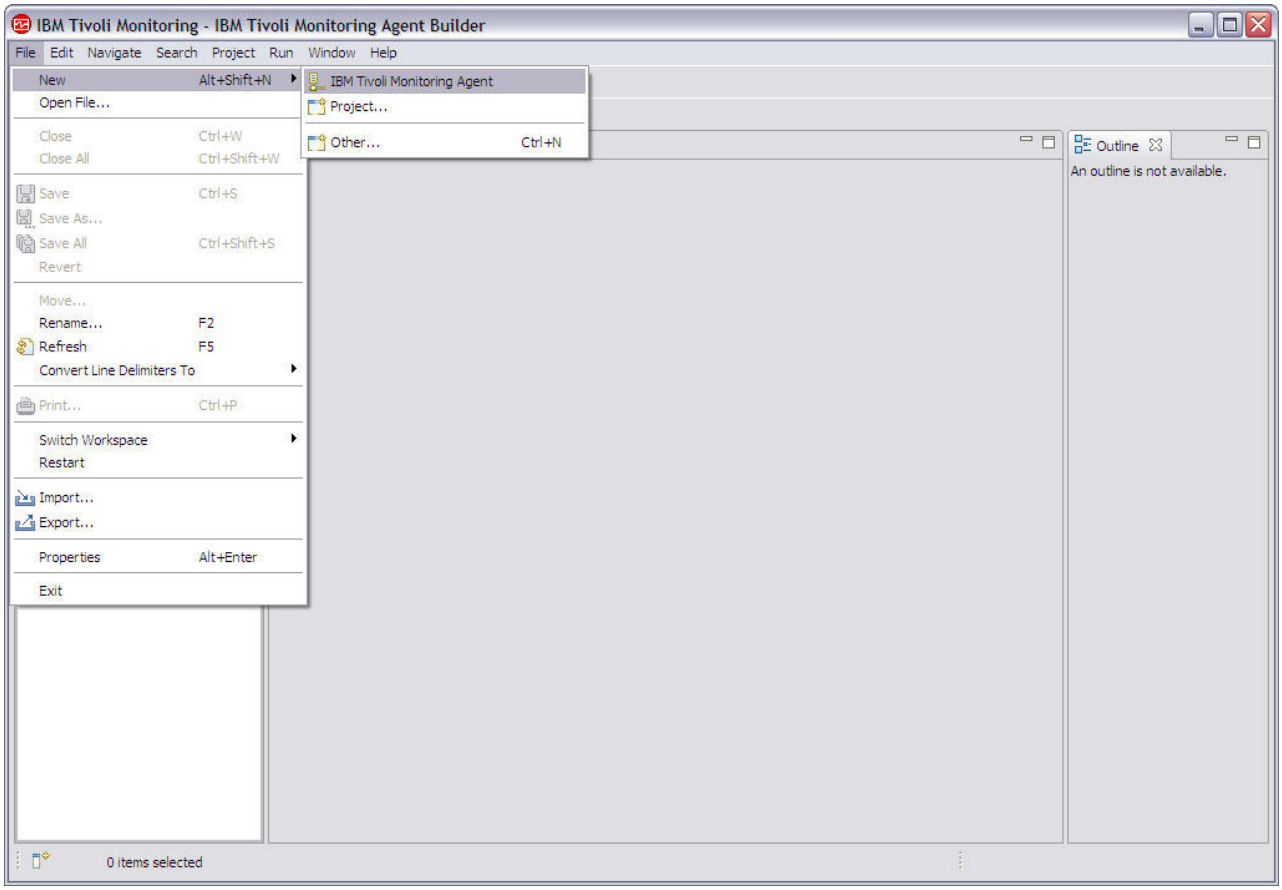


Figure 2. Starting the wizard with the **File** > **New** > **IBM Tivoli Monitoring Agent** *option*

The Welcome window Figure 6 on page 19 for creating an agent is displayed

- From the Main menu, select **File** > **New** > **Other**

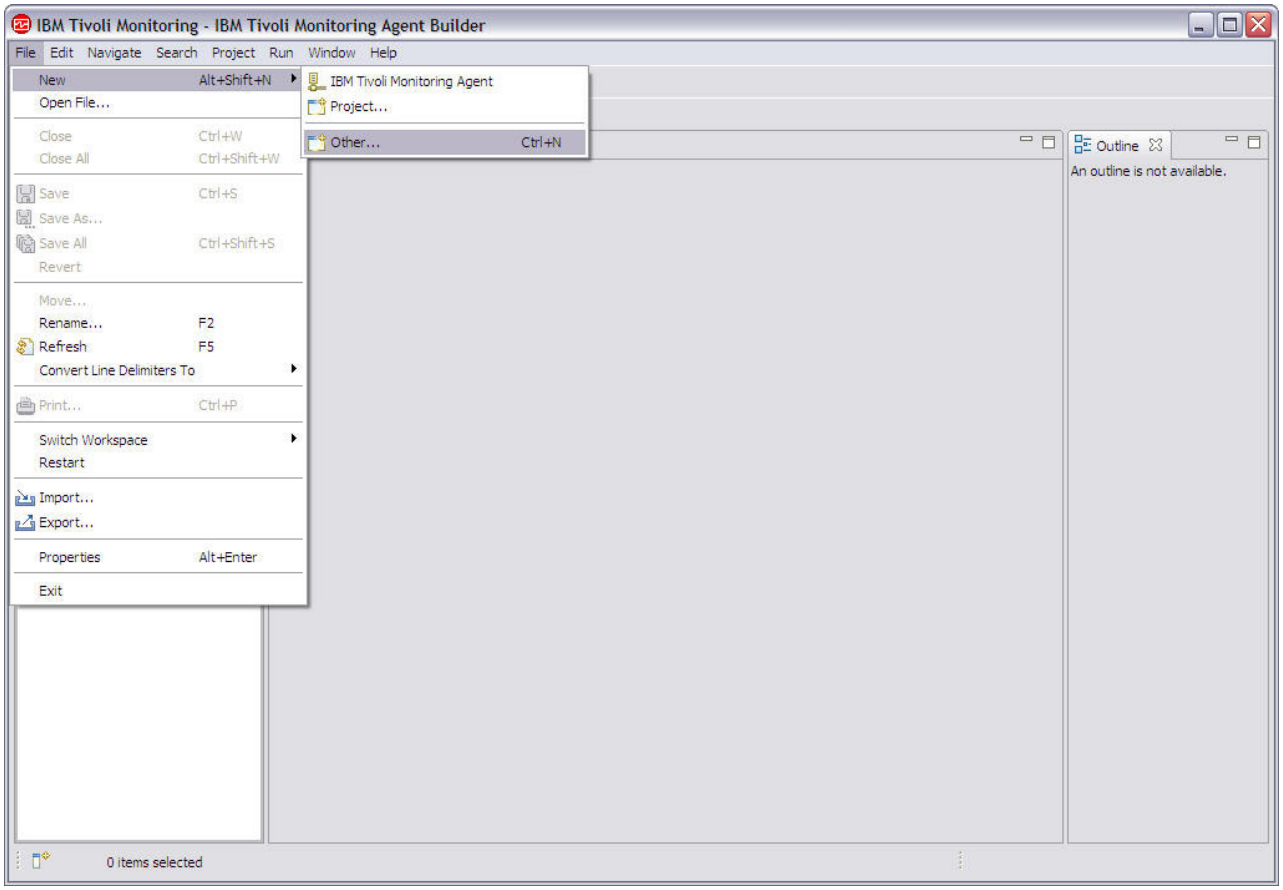


Figure 3. Starting the wizard with the File > New > Other option

1. When the Select a Wizard page is displayed, double-click the **IBM Tivoli Monitoring Wizards** folder of wizard options.

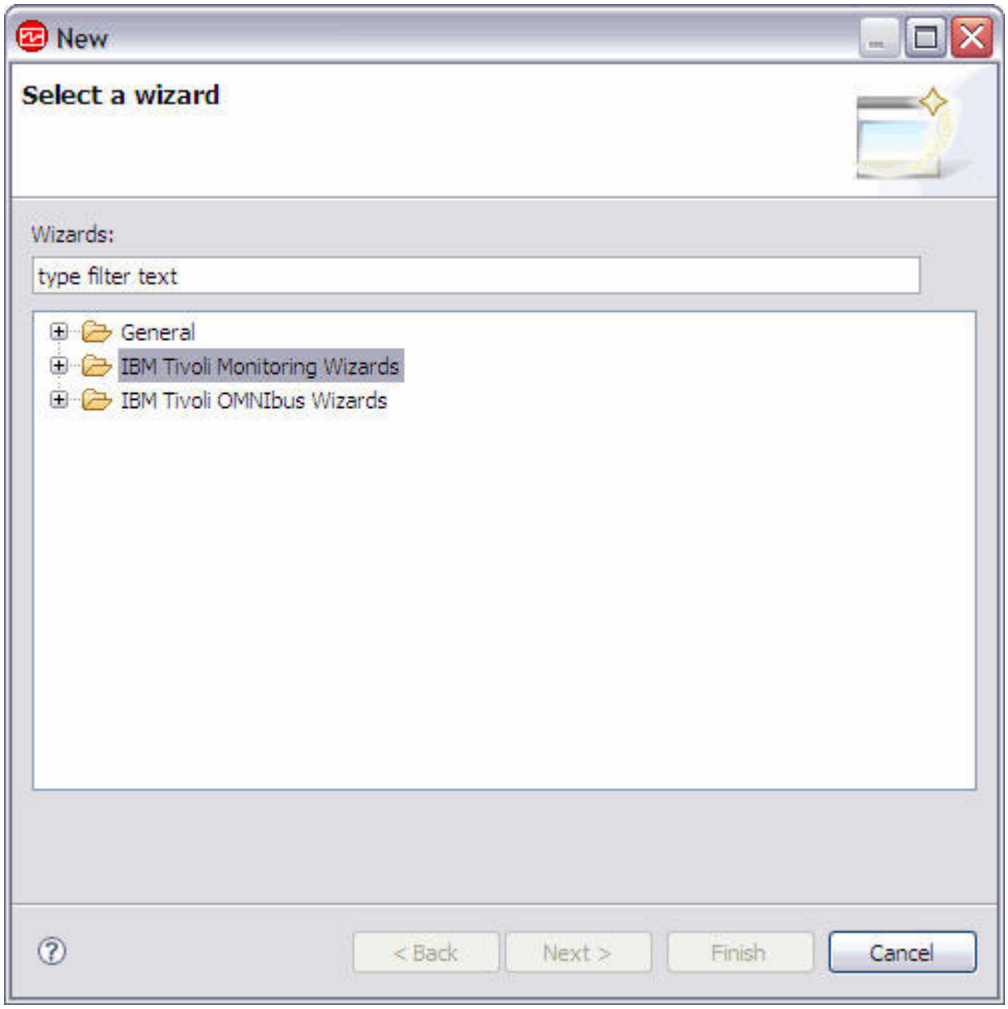


Figure 4. Select a wizard page

2. When the IBM Tivoli Monitoring Agent icon is displayed, double-click it.

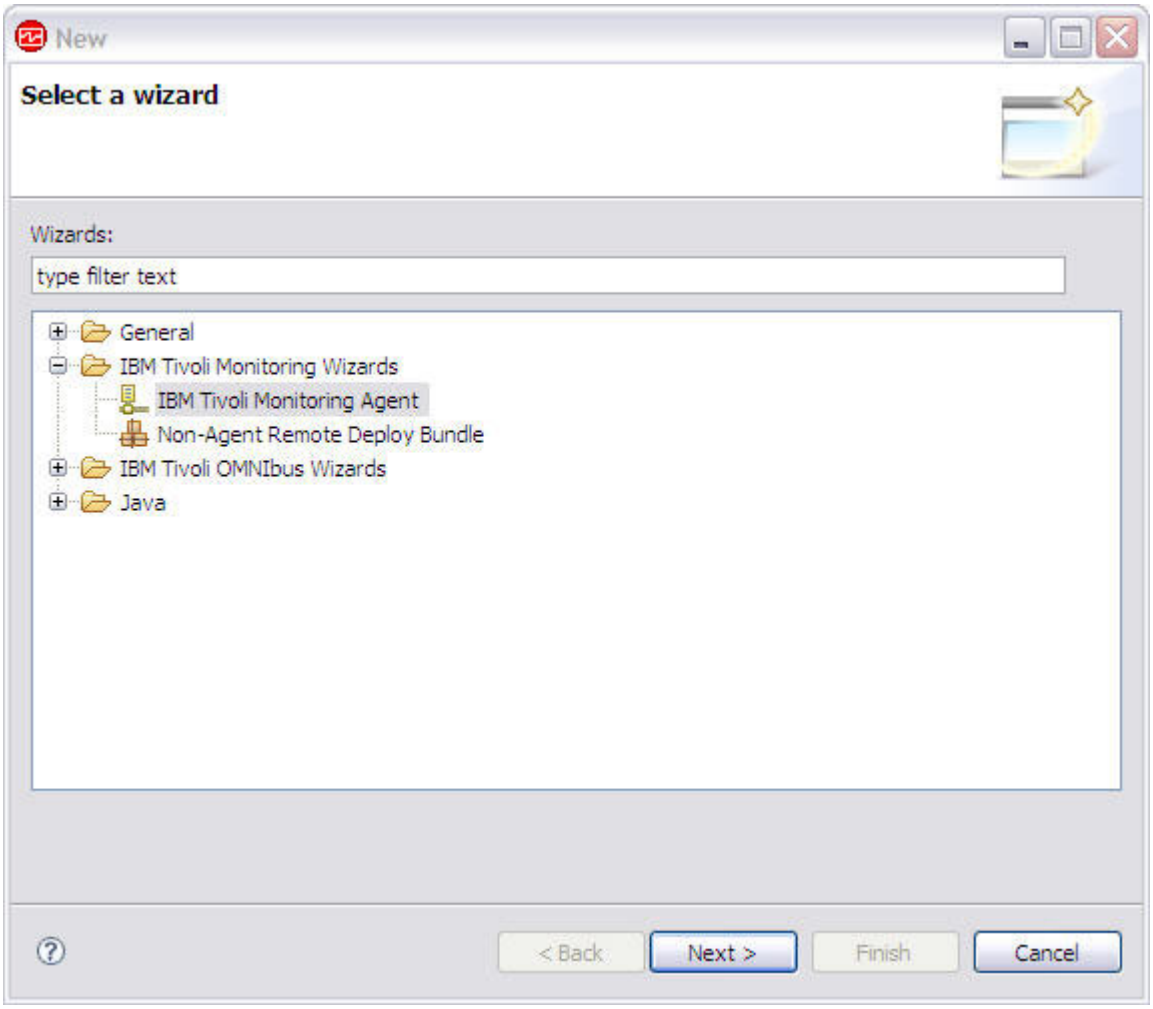


Figure 5. Selecting the agent icon

The Welcome window Figure 6 on page 19 for creating an agent is displayed.

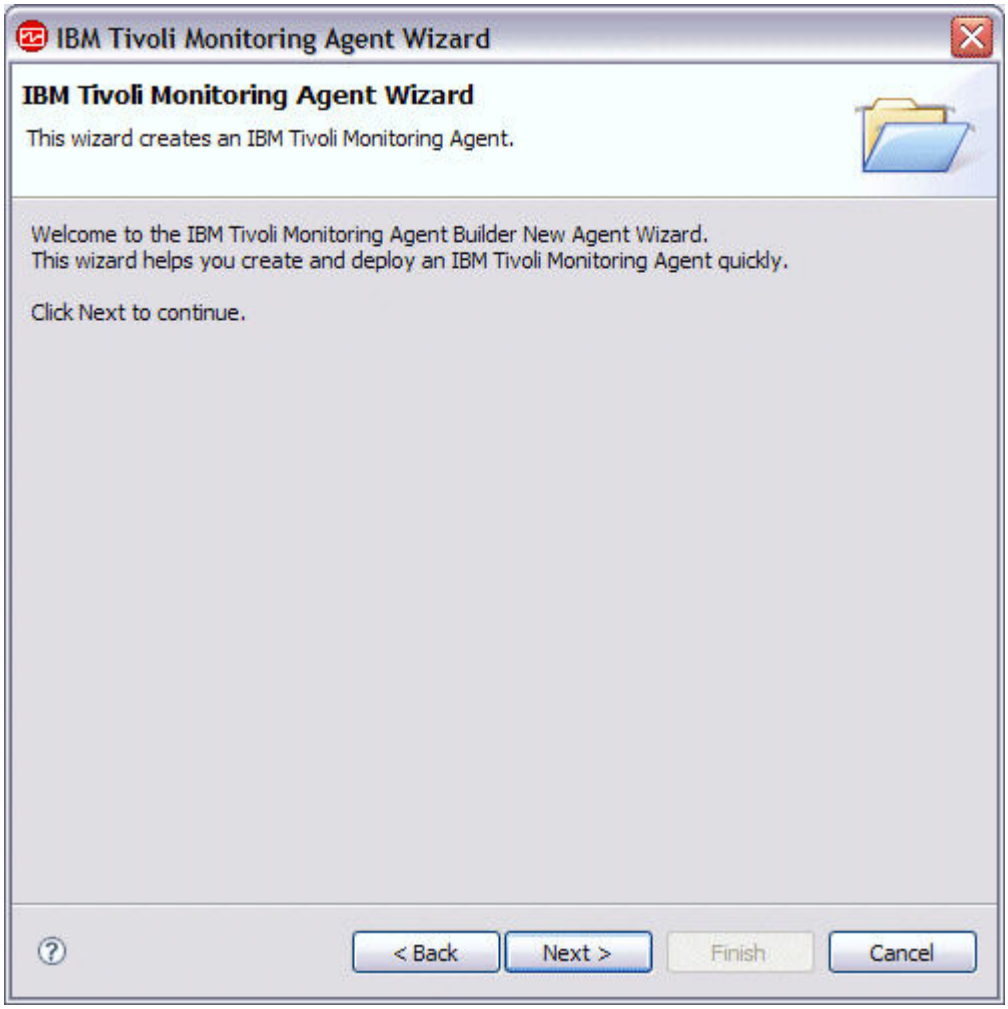


Figure 6. Welcome window

What to do next

Click **Next** to choose a name for your agent. For more information, see (“Naming your agent”).

Creating and defining the agent

Use these procedures and options to create and define the agent.

After you select a method to start the wizard, (“Starting the new agent wizard” on page 13), use the following procedures and options to create and define the agent:

1. Name your agent.
2. Define the data monitoring types by choosing one or more data monitoring categories and data sources for your new agent to monitor.
3. Selecting key attributes
4. Organize the data sources for the agent.

Naming your agent

Use the Project page to name your Agent.

Before you begin

Start the new agent wizard. For more information, see starting the wizard.

About this task

When you are ready to run the agent, the agent name that you designate is displayed in the (Tivoli Enterprise Portal).

Use the following steps to name your agent:

Procedure

1. After you start the new agent wizard, click **Next** in the Welcome window. Clicking **Next** displays the New IBM Tivoli Monitoring Agent Project page (Figure 7). On that page you can type a *project name* for the folder that is created in the Eclipse workspace to hold the necessary files for generating the agent.

   Note:

   a. Use the **Location** field to select a directory that is not the default directory for storing the project contents. If you want to put or use the project contents in another directory, clear the check box next to the **Use default** field. Then, click **Browse** to browse to your preferred directory.

   b. The Working Set function is an optional Eclipse feature that you can use to add resources to working sets. You can change how the Eclipse Navigator View displays resources by adding them to various working sets. For more information, see the Eclipse help.

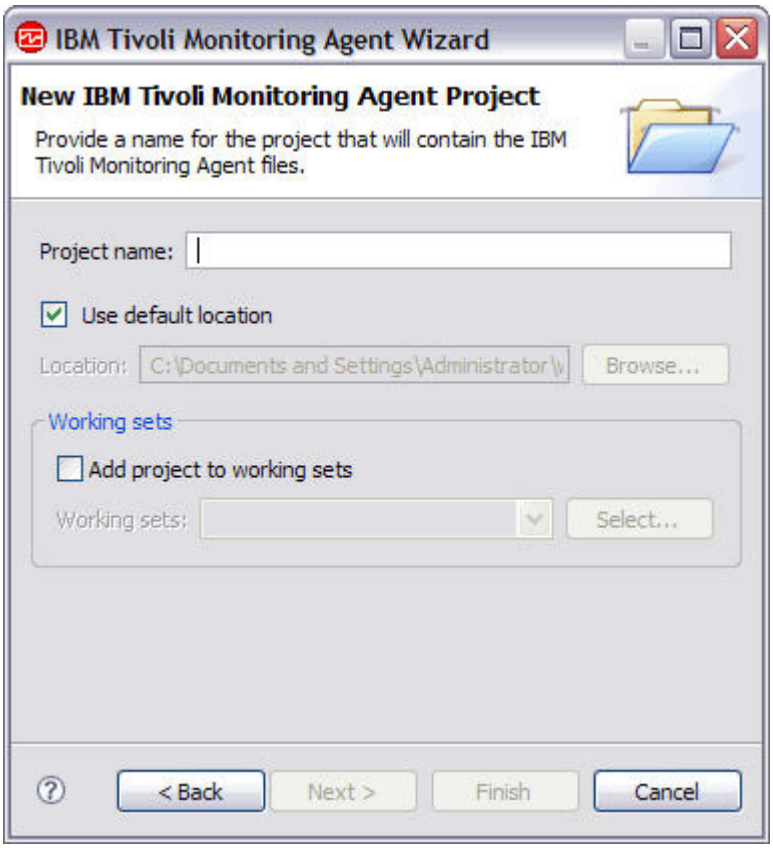


Figure 7. Project page

2. After you type the project name, click **Next** to see the General Information page.
   The project name is displayed in the **Display name** field. This name is displayed in the Tivoli

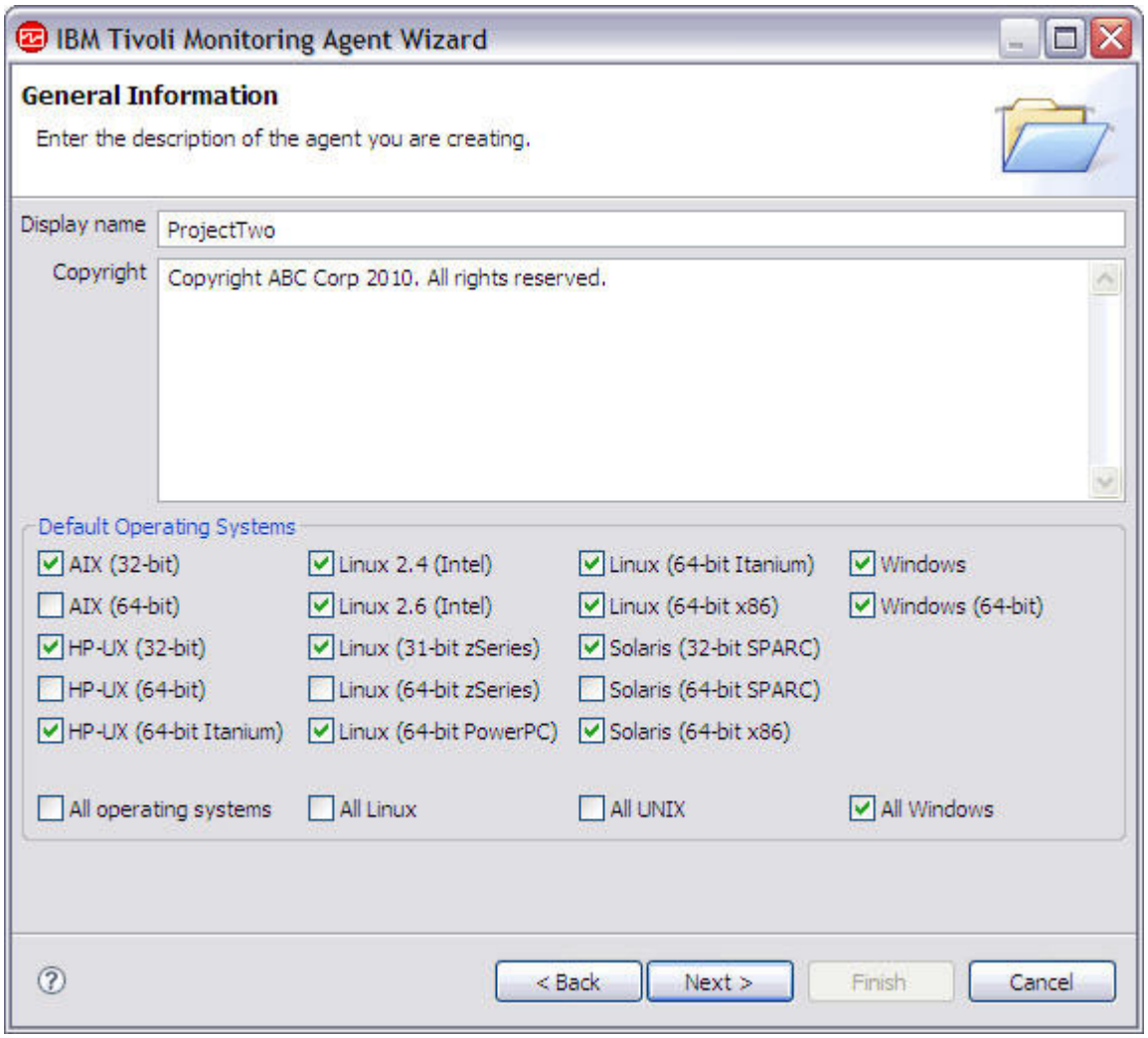


Figure 8. General Information page

   Enterprise Portal.
3. Type the copyright statement that you want to use for your new agents in the **Copyright** field (Figure 9 on page 22). This statement must meet your legal requirements for copyrights. This copyright statement is inserted into all of the files that are generated for the agent and can be edited when needed. After you supply the statement for the first time, the Agent Builder remembers your copyright statement. Agent Builder displays your copyright statement in the copyright area each time that you start the wizard to create an agent. For example, you might use the following format and contents: `Copyright ABC Corp 2009. All rights reserved.` If you are unsure what information you want to supply about the copyright, contact your enterprise legal department.
4. Select the operating systems for which you want your agent to be built.

   Note: If the 32-bit version of an operating system is selected, the agent can be deployed on a 64-bit operating system as a 32-bit process. However, in that case the 64-bit version of the operating system must not then be selected.
5. Click **Next** to display the Agent Information page.



Figure 9. Agent Information page

6. If you want to change the name of the agent, type a new name in the **Service name** field. The service name is the name of the service that is created on Windows. The service name is also the name of the service that is displayed in the Manage Tivoli Monitoring Services window. The name of the service is Monitoring Agent for *Name*, where *Name* is any name you choose consisting of letters, numbers, spaces, and underscores.
7. In the **Product code** field, type the registered product code for your new agent. A range of product codes is reserved for use with the Agent Builder. The allowed values are `K00-K99`,`K{0-2}{A-Z}`, and `K{4-9}{A-Z}`. These values are for internal use only and are not intended for agents that are to be shared or sold. If you are creating an agent to be shared with others, you must send a note to toolkit@us.ibm.com to reserve a product code. The request for a product code must include a description of the agent to be built. A product code is then assigned, registered, and returned to you. When you receive the three-letter product code, you are told how to enable the Agent Builder to use the assigned product code.
8. In the **Company identifier** field, type an alphabetic string that uniquely identifies the organization that develops the agent (`IBM` is reserved). A good candidate for this field can come from the URL of your company. For example, from a `mycompany.com` URL, use the text `mycompany`.
9. In the **Agent identifier** field, type an alphabetic string that uniquely identifies the agent For example, `DISKMON` might be the string for an agent that monitors disk usage. You can use the product code for the agent identifier.

   Note: The combined length of the **Agent identifier** field and the **Company identifier** field cannot exceed 11 characters.

   By default, Agent Builder sets the Agent identifier to be the same as the Product code.
10. In the **Version** field, type a three-digit number that identifies the agent version in the format *VRR*, where:

V = Version

R = Release

R = Release

The *VRR* entered is converted to a Tivoli Monitoring `VVRRMMFF` format in the following way: `0V.RR.00.00`.

In the agent editor, a **patch level** field is available. The **patch level** field can be used when you release a fix for an agent, without updating the version.

11. If you want your agent to support multiple instances, select the **Support multiple instances of this agent** check box.

    Some applications can be configured to run multiple instances of the application on a system at the same time. When building agents for these types of applications, it is often desirable to separate the management of each of the application instances. The separation means that the Tivoli Monitoring user can more easily determine when there are problems with one of the application instances. A similar condition happens if you are building an agent that uses a data provider from the **Data from a server** monitoring category. Because these data monitors can be used to monitor remote systems, it is possible to install the agent on one system and monitor several other systems. Each of these remote systems can be represented separately in Tivoli Monitoring and each can have different configuration values for communication. To have different configuration values, select **Support multiple instances of this agent**. This selection causes the Agent Builder to create a "template" agent. After the agent is installed, you can create and configure an instance of the agent for each instance of the monitored application.

12. In the **Minimum ITM Version** field, select the minimum version of IBM Tivoli Monitoring that the agent requires. See Table 5 which provides the reasons to use each level.

Table 5. Minimum version of IBM Tivoli Monitoring required by the agent

IBM Tivoli Monitoring level	Reason to use this level
V6.2.1	• The generated agent requires a minimum Tivoli Monitoring version of 6.2.1 to run successfully, and you cannot install on a version earlier than 6.2.1 (Figure 9 on page 22). If you select 6.2.1, you can **Allow any configuration property to be overridden in any subnode**. For more information, see ("Advanced subnode configuration" on page 348.) • This version supports 64-bit numeric attributes. • This version includes support for different default configuration values for Windows and for UNIX and Linux. For more information, see Default Value. The default minimum Tivoli Monitoring version is 6.2.1.
V6.2	By selecting 6.2, the resulting agent runs on Tivoli Monitoring, version 6.2 or later. However, other features that are listed earlier including 64-bit numeric attribute support are not available to the agent. Also, Subnode Configuration Overrides must be defined when the agent is built, see ("Configuring a subnode" on page 345).

What to do next

Click **Next** to define an initial data source for your agent. For more information, see "Defining initial data source"

Defining initial data source

Define initial data monitoring types by using the Agent Initial Data Source page.

Before you begin

Choose a name for your agent. For more information, see (“Naming your agent” on page 19).

About this task

Define the monitoring data categories and the data sources that your new agent is to monitor by using theAgent Initial Data Source page. For detailed information about each data source, see (“Data Sources” on page 25).

Procedure

1. After you enter the Agent Information, click **Next** on the Agent Information page (Figure 9 on page 22). The Agent Initial Data Source page opens, so you can define the first data source.

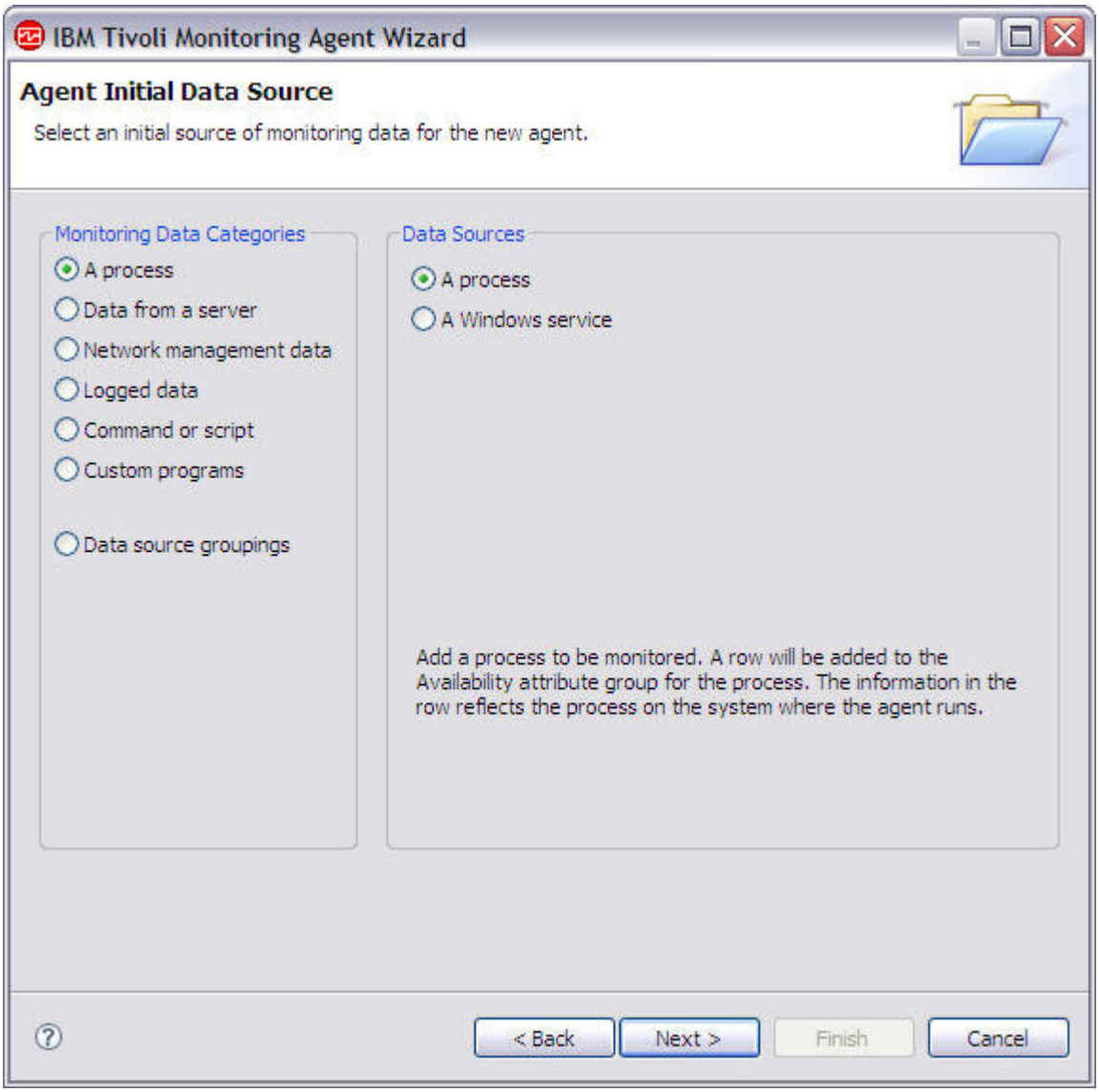


Figure 10. Agent Initial Data Source page

2. Select one of the **Monitoring Data Categories** and one of the **Data Sources**.
3. Click **Next**. The wizard guides you through the process of defining and configuring any of the data collection types that you specify.

4. Optional: Define more data sources from the Data Source Definition page after you define the initial data source. For more information, see (“Defining more data sources” on page 27).
5. If you defined a new data source that might return more than one data row, you are prompted to select key attributes. For more information, see (“Selecting key attributes” on page 26).
6. When you finish defining all of your data sources, click **Finish**.

What to do next

- If you want to add one or more data sources after you finish the wizard, you can add data sources by using the Agent Editor.
- Test and debug your agent.
- Generate and install your new agent.

Related concepts:

Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35
Use the Tivoli Monitoring Agent Editor to change, save, and commit a version of your agent.

Chapter 31, “Testing and debugging your agent,” on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

“Installing and testing an agent” on page 387
There are two methods for installing and testing the agents that you create with the Tivoli Monitoring Agent Builder.

Data Sources

Index list of data sources.

- Server process monitoring category
  - Process
  - Windows service
- Data from a server monitoring category
  - WMI
  - Perfmon
  - CIM
  - SNMP
  - SNMP Events
  - JDBC
  - JMX
  - HTTP
  - SOAP
- Network management data
  - Chapter 22, “Monitoring system availability by using Ping,” on page 255
- Logged data monitoring category
  - Log File
  - AIX Binary Log
  - Windows Event Log
- Command or script monitoring category
  - Command return code
  - Output from a script
- Custom programs monitoring category
  - Socket
  - Java API

- Existing data sources monitoring category
  - Join two data sources
  - “Creating a Filtered Attribute Group” on page 324
- Data source groupings monitoring category
  - Subnode Definition
  - Navigator group

Selecting key attributes

When an attribute group returns more than one data row, you must select key attributes.

About this task

When an attribute group can return more than one data row, each row represents an entity that is being monitored. Each time monitored data is sampled, IBM Tivoli Monitoring matches a row to the entity that is being monitored and to previous samples for that entity. This matching is done with key attributes. One or more attributes in the attribute group can be identified as key attributes. These key attributes, when taken together, distinguish one monitored entity from another. The key attributes do not change from one sample to the next for the same monitored entity.

Rate and delta attributes are calculated by comparing the current sample to the previous sample. Identical key attributes ensure that the agent is comparing values for the same monitored entity. Similarly, the summarization and pruning agent summarizes samples that have identical key attributes. In addition, any attribute that is set as a key attribute can also be used as a "Display item" in a situation.

You specify the details about your new data source, by using the information in (“Defining initial data source” on page 23). If it is possible for multiple data rows to be returned by the data source, you are prompted to select key attributes (Figure 11 on page 27).

Note: If the browser detected key attributes, those attributes are marked as key attributes and the Select key attributes page is not displayed.



Figure 11. Select key attributes page

Procedure

On the Select key attributes page, do one of the following steps:

- Click one or more attributes from the list that are the key attributes for this entity. To select more than one attribute, hold down the **Ctrl** key.
- If this attribute group can return only one row, click **Produces a single data row**. If this option is selected, no key attributes are necessary because only one monitored entity is ever reported in this attribute group.

Defining more data sources

You can add more data sources to your agent after you define the initial data source.

Procedure

1. Define more data sources after you define the initial data source, by any of the following methods:
   - From the Data Source Definition page select the agent, navigator group, or subnode, and then click **Add to Selected**
   - From the Data Source Definition page right-click on the agent, and select **Add Data Source** as shown in (Figure 12 on page 28)
   - You can also add more data sources after you complete the New Agent wizard. For more information, see (Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35). An advantage of using the Agent Editor is that you can save incremental changes as you make them.

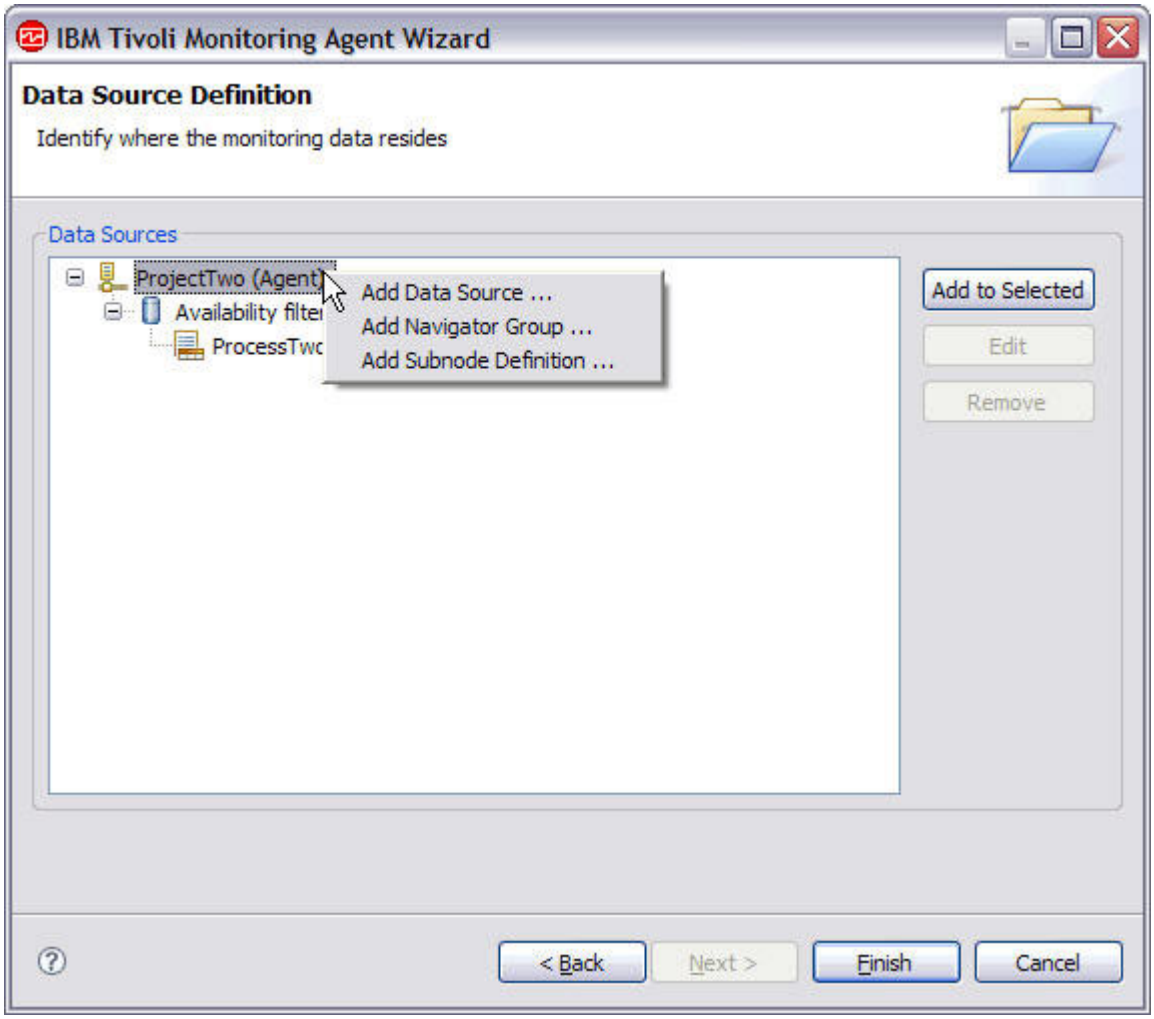


Figure 12. Data Source Definition page: defining more monitors

When you add a monitor by using the Agent Editor Data Source Definition page, the Data Source Location page is shown (Figure 13 on page 29). The Data Source Location is the same as the Agent Initial Data Source page (Figure 10 on page 24) except for the name of the page.

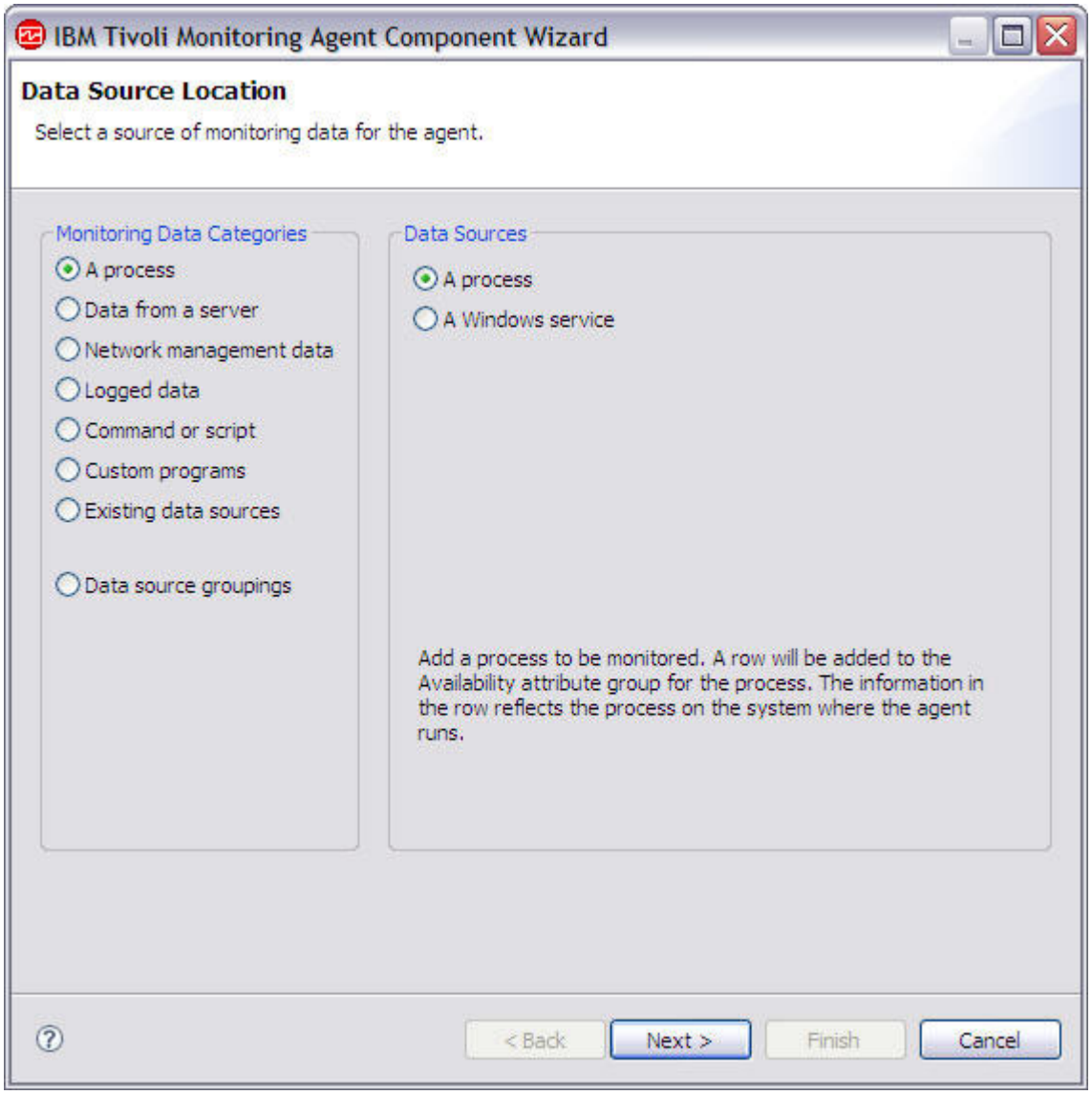


Figure 13. Data Source Location page

2. Select one of the **Monitoring Data Categories** and one of the **Data Sources**.
3. Click **Next**. The wizard guides you through the process of defining and configuring any of the data collection types that you specify.
4. Click **Finish** to add the new data source to the Data Source Definition page.

Organizing the agent

There are a number of ways you can organize the agent.

You can organize the data sources for the agent in the following ways:

- Navigator groups
- Subnodes

A navigator group is a way of grouping data sources in the Tivoli Enterprise Portal. By default, a separate navigation item is created in the Tivoli Enterprise Portal under your agent for each data source that you define. A query is associated with that navigator item so a default workspace displays data that is collected by that data source in a simple table view. If a data source is placed inside a navigator group,

a separate navigator item is no longer displayed under the agent. Workspaces that are created for the navigator group might display views that are constructed from all data sources that are contained in the navigator group. For more information, see (“Navigator groups”), and (Chapter 28, “Creating a navigator group,” on page 327).

A subnode is a portion of an agent that can be replicated as necessary, depending on the application or enterprise that is being monitored. For more information, see (“Subnodes”), and (Chapter 29, “Use of subnodes,” on page 331).

Navigator groups

Use Navigator groups to group several related data sources together so that workspaces can be created that show views of the related data sources

You might be able to collect file system data from more than one data source. It can be useful to create one workspace that shows views of all file system data from those different data sources.

Navigator groups are also a good way to hide data sources. You might decide that metrics collected from two data sources are most useful if the data sources are joined to create a combined, third data source. You want to see only the combined data in the Joined data source. You can create a navigator group that contains all three data sources and create a workspace that contains views to display only the combined data source. The two original data sources are effectively hidden from view in the Tivoli Enterprise Portal. See Chapter 27, “Creating attribute groups from existing sources,” on page 319 for information about joining data sources.

Note: When you group data sources in a navigator group, Tivoli Monitoring does not associate a query with the navigator group. It is assumed that you define a default workspace for the navigator group to display the data sources in a useful format.

The following criteria apply to navigator groups:

- A navigator group can be defined in the base agent or in a subnode.
- A navigator group cannot contain another Navigator group.

Subnodes

Use subnodes to monitor multiple applications from a single agent instance.

You can build a single agent that accomplishes the following tasks by using subnodes:

- Monitors each instance of an application that is running on a system instead of having to use separate instances of the agent, one per application instance.
- Monitors several different remote systems instead of having to use separate instances of the agent, one for each remote system.
- Monitors several different types of entities from one agent instead of having to build and deploy several different agents.
- Displays an additional level in the Tivoli Enterprise Portal physical Navigation tree that allows further grouping and customization.
- Defines further Managed System Lists allowing another level of granularity with situations.

An agent developer defines subnode types in the Agent Builder. Each type must correspond to a different type of entity that an agent can monitor. Attribute groups and attributes are added to the subnode type that describes the entity that is being monitored. When the agent is deployed and configured, one or more instances of each subnode type can be created. Each instance of a subnode must correspond to an instance of an application, a remote system, or whatever entity the subnode type was designed to monitor. All subnode instances of a single subnode type have attribute groups and workspaces that have an identical form. However, each subnode has data that comes from the particular entity that is being monitored.

The number of subnodes of each type is determined when the agent is configured. Some configuration data can apply to the agent as a whole, but other configuration data applies to only a single subnode. Configure each subnode differently from the other subnodes so that they do not monitor the exact same entity and display the exact same data.

A subnode is displayed within the agent in the physical Navigation tree in the Tivoli Enterprise Portal. Workspaces display the data that is defined by a subnode and situations can be distributed to one or more instances of a subnode. A managed system list is automatically created that contains all instances of the subnode, just like the Managed System List that is created for an agent.

Because agents built with Agent Builder create the subnode instances that are based on configuration values, these subnodes have the same life span as the agent. There is still just one heartbeat that is done for the agent, not a separate heartbeat for each subnode. Use subnodes to define monitoring for a number of systems or application instances and you can significantly increase the potential scale of the Tivoli Monitoring environment. The alternative is to use multiple agent instances, which can limit the potential scale of the Tivoli Monitoring environment.

Adding or removing a subnode requires reconfiguring the agent, which involves stopping and restarting it. For this reason, it can be good practice to define the agent as a multi-instance agent as well. By defining the agent as a multi-instance agent, you can manage portions of your environment separately.

Along with attribute groups in subnodes, an agent can define agent-level attribute groups that are located outside of a subnode. In the Tivoli Enterprise Portal Navigator tree, a subnode type is displayed under the agent name, and subnode instances are displayed under a subnode type. Subnodes are identified by a Managed System Name (MSN) just like agents, for example `94:Hill.cmn`.

For example, in the Navigator tree in Figure 14 on page 32, **Watching Over Our Friends** is an agent with three entities (**Boarders**, **Common Areas**, and **Kennel Runs**) and two subnode types (**Common Area** and **Kennel Run**). Two of these entities have subnode types that are defined for them (**Common Area** and **Kennel Run**). A subnode is not required for the third entity (**Boarder**), which is represented by a single row in a table at the base agent level. The Common Area subnode type has three subnode instances: `94:Hill:cmn`, `94:Meadow:cmn`, and `94:Tree:cmn` representing three common areas in the kennel. The Kennel Run subnode type has four subnode instances: `94:system1:run`, `94:system2:run`, `94:system4:run`, and `94:system5:run` representing four kennel runs.

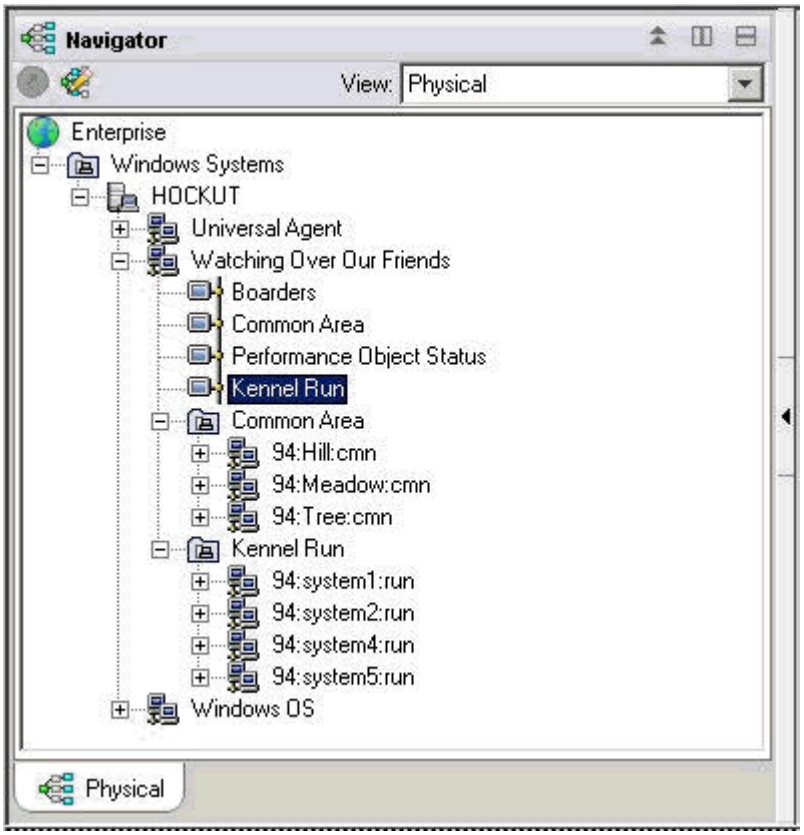


Figure 14. Subnodes in the Navigator tree

Moving existing data sources

Move an existing attribute group to a newly created navigator group or subnode.

Procedure

1. Begin creating a navigator group or subnode as described in Chapter 28, "Creating a navigator group," on page 327 or (Chapter 29, "Use of subnodes," on page 331).
   a. On the Navigator Group Data Source page, click **Existing data sources** in the **Monitoring Data Categories** area (Figure 15 on page 33).

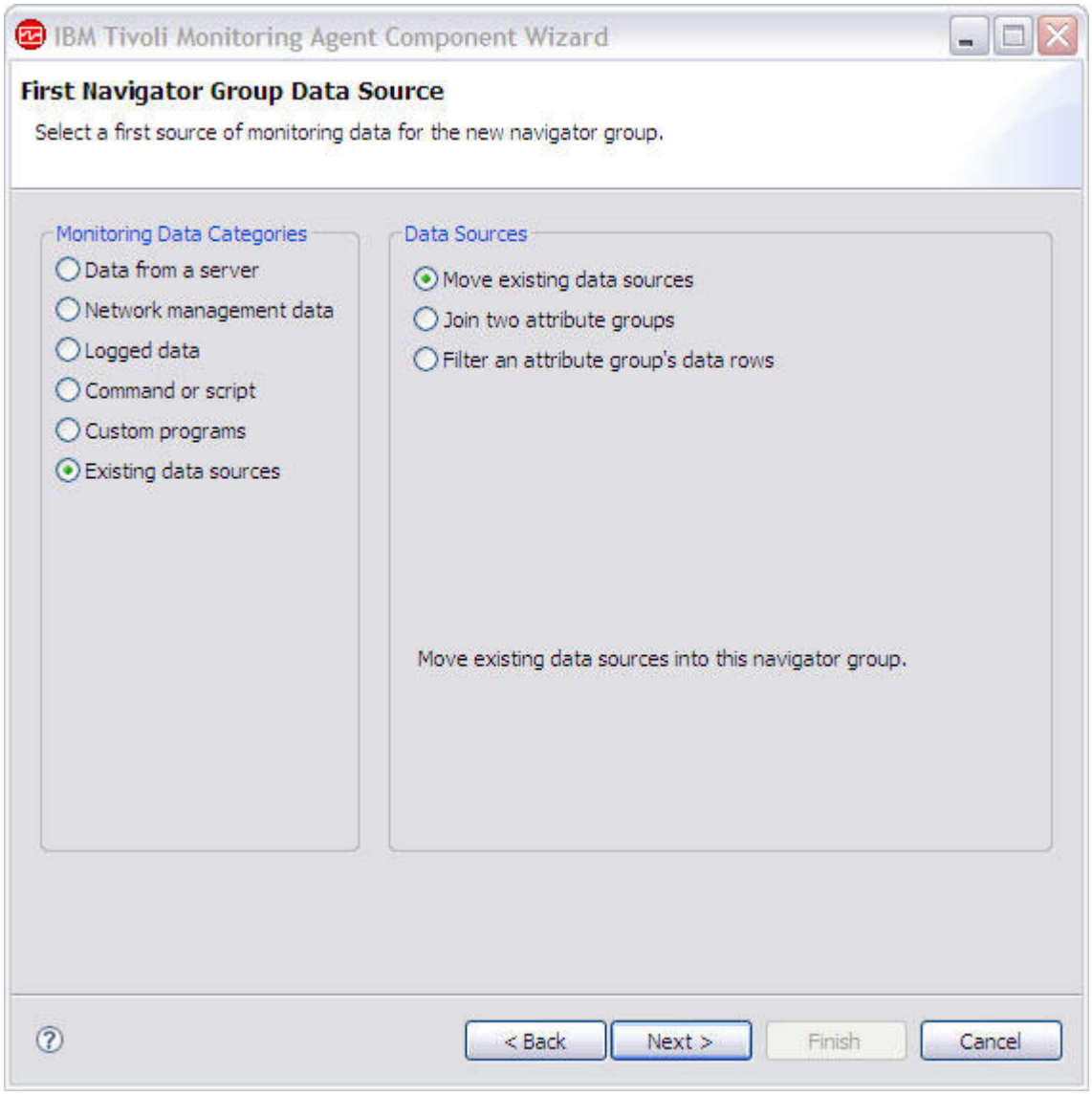


Figure 15. Move existing data sources

b. In the **Data Sources** area, click **Move existing data sources**.
c. Click **Next**.

2. On the Currently Defined Data Sources page, select one or more data sources (Figure 16 on page 34).



Figure 16. Currently Defined Data Sources page

3. Click **Finish**.

Results

The data source or data sources that are selected are moved into the new navigator group or subnode.

Note: When a subnode or navigator group is created, you can move attribute groups into it from the Data Sources tab of the Agent Editor. Move the attribute groups by dragging them from their original location to the navigator group or subnode in the navigation tree.

Preparing your agent for modifications or for regeneration

Modify and reinstall your agent.

To modify an existing agent definition, double-click the `itm_toolkit_agent.xml` file to start the IBM Tivoli Monitoring Agent Editor, see (Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35).

If you installed your agent before you modified it, you can uninstall the existing agent before you install the new one. Alternatively you can install the new agent directly over the existing agent without uninstalling it. To uninstall the existing agent first, see the procedure in (“Uninstalling an agent” on page 400), then regenerate and install the agent again as you did initially. If you choose to install over the existing agent, follow the steps in (Chapter 31, “Testing and debugging your agent,” on page 377).

If you change the agent version, when you install the new version, files in your Tivoli Monitoring installation that contain the old version number are overwritten. An exception is the file that contains your agent's configuration requirements (named according to the format: *product_code*_dd_*long_version*.xml). For example, for an agent with product code, 19 and version, 6.2.3, the file is called `19_dd_062300000.xml` along with the remote deployment bundle for your agent. Both of these files are added to the system alongside the old versions rather than replacing them.

Chapter 6. The Tivoli Monitoring Agent Editor

Use the Tivoli Monitoring Agent Editor to change, save, and commit a version of your agent.

Before you modify an agent, see "Preparing your agent for modifications or for regeneration" on page 34.

The IBM Tivoli Monitoring Agent Editor is a multi-page Eclipse editor that you can use to modify the properties of an existing IBM Tivoli Monitoring Agent. Each page in the editor corresponds to a specific function of the agent.

The list of available pages is shown in the Outline view beneath the Tivoli Monitoring Agent node. You can easily switch to another page by clicking a node in the Outline view. If the Outline view is missing, or hidden behind another view, you can reset the Tivoli Monitoring perspective. Reset the perspective by selecting **Window** > **Reset Perspective**. Alternatively right-click `IBM Tivoli Monitoring` and select **Reset** from the menu.

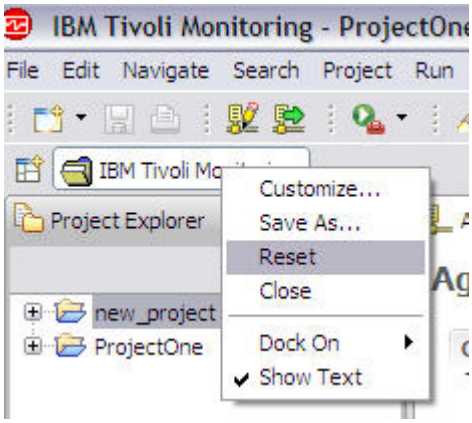


Figure 17. Choosing Reset from the Tivoli Monitoring perspective.

Note: For detailed information and procedures for creating an agent, see Chapter 5, "Creating a basic agent," on page 13.

The following pages are included in the Agent Editor:

- "Agent Information page"
- Data Source Definition page
- Runtime Configuration Information page
- Agent XML Editor page (itm_toolkit_agent.xml)

Note: When you view an Editor page, you can also switch to another page by clicking the tab for the page. Some pages show tabs only when they are selected in the Outline view. You can force a page to have a tab even when it is not selected. To force a page to have a tab, click the pin icon so that the pin in the icon points toward the page.

Agent Information page

The Agent Information page is the main page of the Tivoli Monitoring Agent Editor.

The Agent Information page (Figure 18 on page 36) contains the following information:

© Copyright IBM Corp. 2007, 2013

- General agent information
- Agent Content information
  - Default Operating Systems link
  - Self-Describing Agent link
  - Environment Variables link
  - Watchdog Information link
  - Cognos Information link
  - Data sources link
  - Runtime Configuration link
  - Outline View link
- Generate Agent Wizard link
- Commit Agent Version link

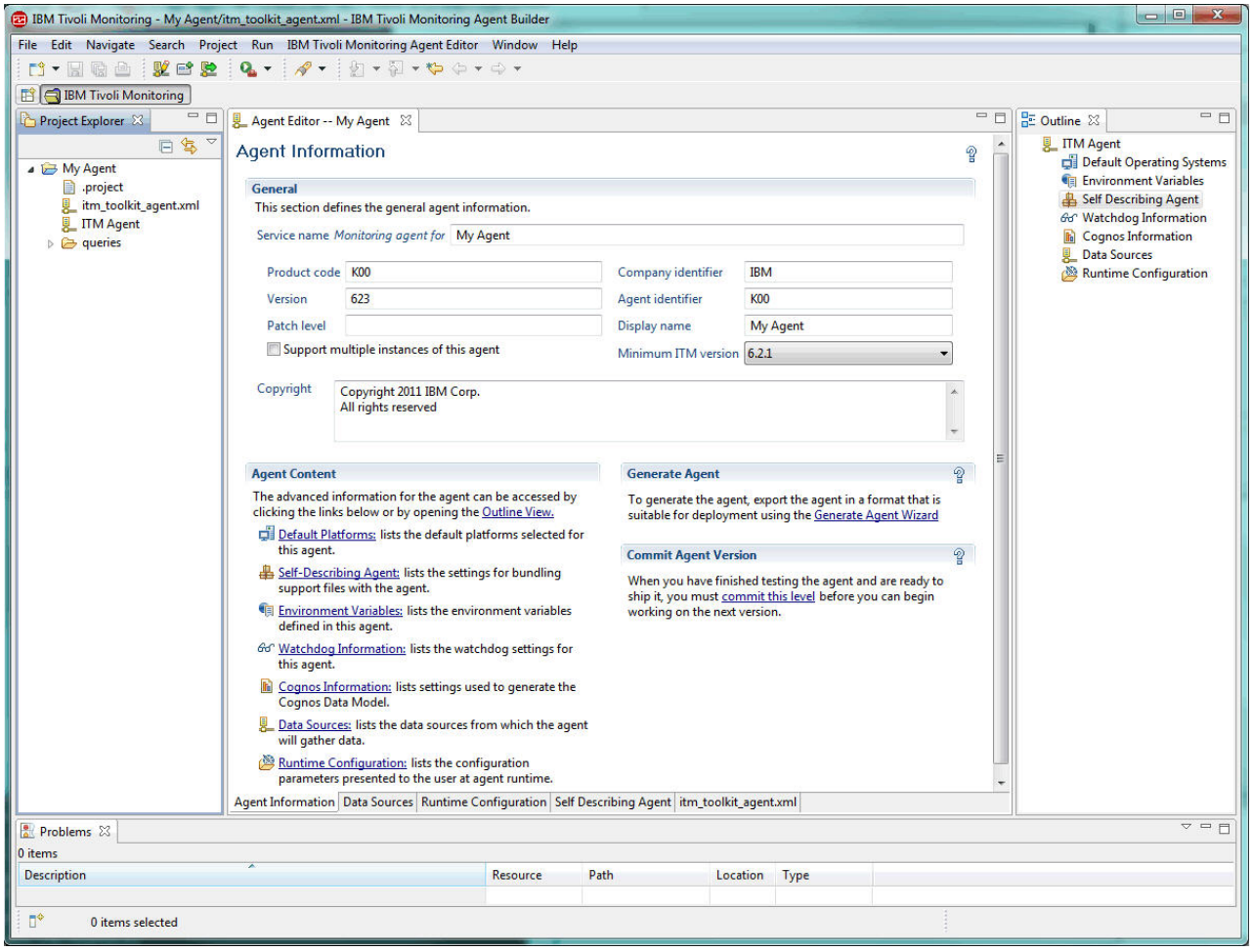


Figure 18. Agent Information page

General agent information

This general agent information was entered in the New Agent Wizard when the agent was created.

Default operating systems

Use the **Default Operating Systems** page to identify the operating systems for which your agent is built.

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.

About this task

Procedure

- To open the **Default Operating Systems** page (Figure 19 on page 38), click **Default Operating Systems** in the Agent Content section of the Agent Information page or the **Default Operating Systems** node in the Outline View.
- Use the **Default Operating Systems** page to identify the operating systems for which your agent is built.

  When you generate your agent, files unique to each of the operating systems you select here are added to your agent. Data sources that you add to your agent that are not specific to the Windows operating system are available on any of the selected operating systems. The operating systems on which any specific data source is available can be changed from this default selection. To change the Operating Systems available for a specific data source, use the Operating Systems pane of the Data Source Definition page. If default operating systems are not selected, operating systems must be selected for each specific data source on the Data Source Definition page.

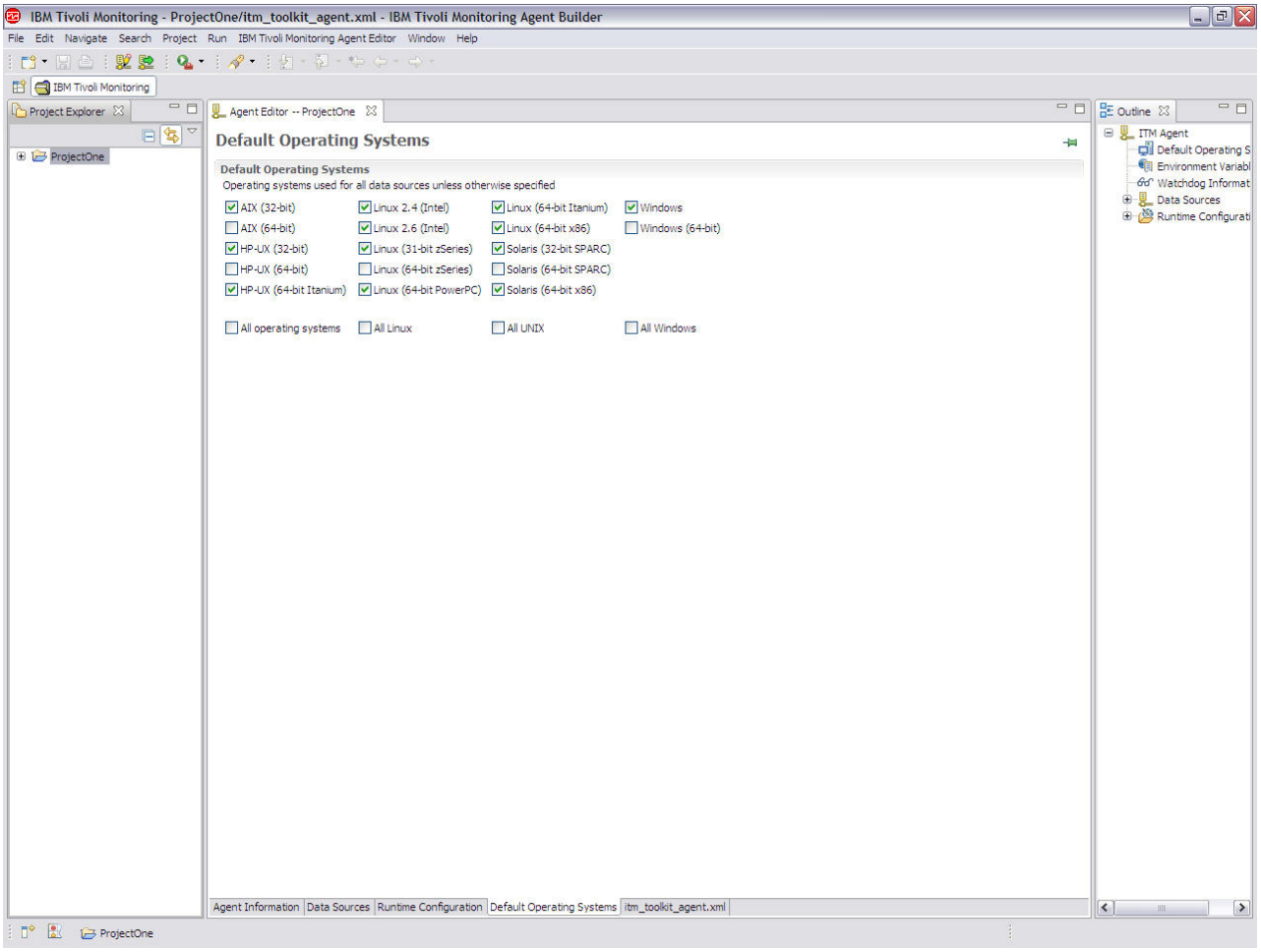


Figure 19. Default Operating Systems page

Self-Describing Agent

Use the Self-Describing Agent page to specify whether the agent's support files are bundled with the agent.

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35.

About this task

Procedure

To open the Self-Describing Agent page, click **Self-Describing Agent** in the Agent Content section of the Agent Information page or the **Self-Describing Agent** node in the Outline View.
Self-description is enabled by default for all new agents that are created with Agent Builder 6.2.3 or later. When self-description is enabled for an agent, application support packages are included in the agent image. The inclusion enables the agent to seed the support files for the Tivoli Enterprise Monitoring

Server, Tivoli Enterprise Portal Server, and the Tivoli Enterprise Portal Browser. For more information about self-describing agents, see the *IBM Tivoli Monitoring Installation and Setup Guide* and the *IBM Tivoli Monitoring Administrator's Guide*.

Note: You must have Tivoli Monitoring version 6.2.3 or later installed for the self-describing agent feature to work, and self-description must be enabled in Tivoli Monitoring. By default self-description is turned off in Tivoli Monitoring.

Note: Selecting the **Enable self-description for this agent** check box does not prevent your agent from working on previous versions of Tivoli Monitoring.

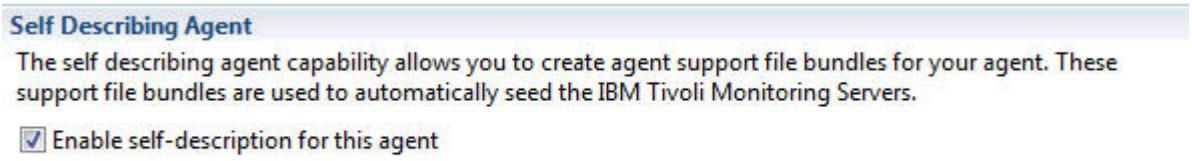


Figure 20. Self-Describing Agent page

Environment variables

Use the Environment Variables page to view and modify environment variables that are available to your agent while it is running.

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.

About this task

The environment variables can be defined by you, for access inside a script, or predefined variables that cause the agent to behave in a certain way. See Table 47 on page 435 for a list of predefined variables.

Procedure

1. To open the Environment Variables page, click **Environment Variables** in the **Agent Content** section of the Agent Information page. Alternatively, click **Environment Variables** node in the Outline view.

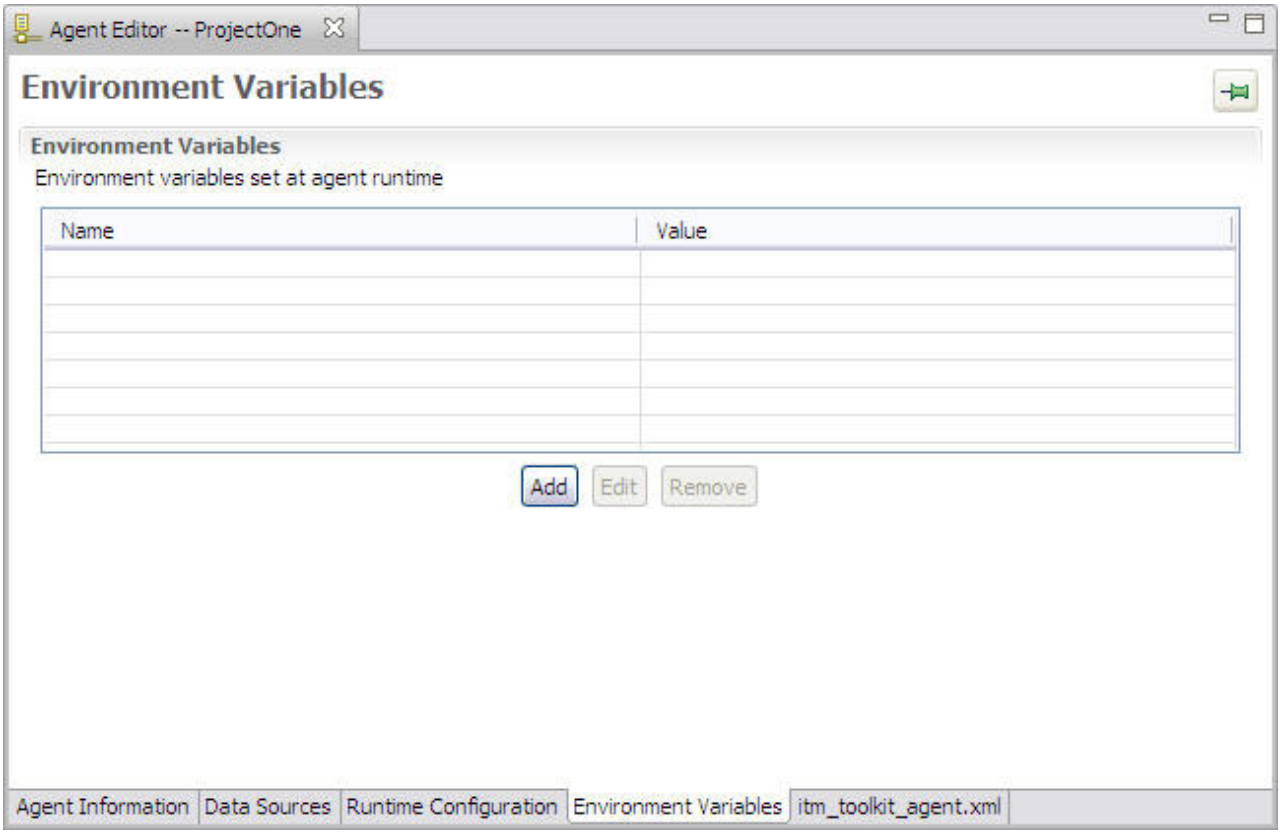


Figure 21. Environment Variables page

2. In the Environment Variables page (Figure 21), click **Add**.
3. In the Environment Variable Information window (Figure 22), complete the information as follows:

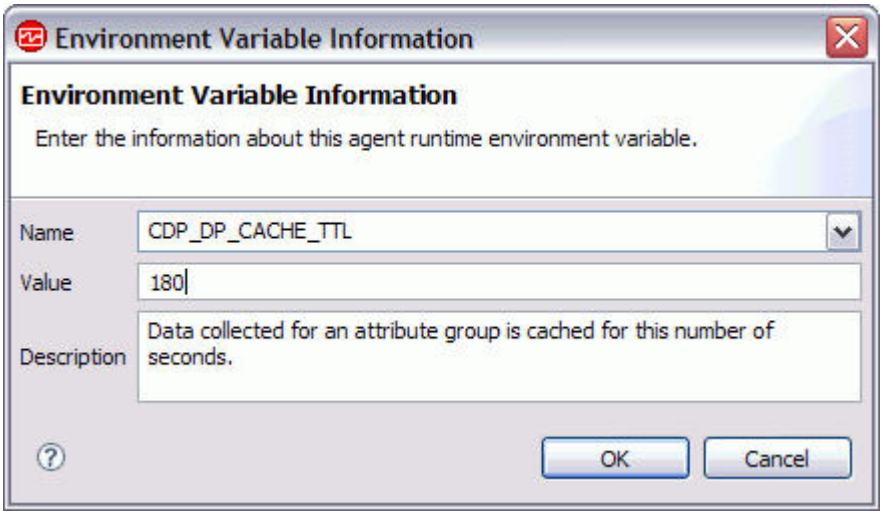


Figure 22. Environment Variable Information window

a. In the **Name** field, type a variable name or select a predefined name from the list.
b. In the **Value** field, type a value for the variable if you want to set a variable for the agent. If you do not enter a value, the agent propagates a value for the existing variable.

c. In the **Description** field, type a description of the variable, or keep the existing description of a predefined variable.
d. Click **OK**. The new variable is listed in the table on the Agent Information page.

Watchdog information

Use the Watchdog Information page to specify configuration information for the Agent Watchdog

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.

About this task

To open the Watchdog Information page (Figure 23), click **Watchdog Information** in the **Agent Content** section of the Agent Information page. You can also select or the **Watchdog Information** node in the Outline View.

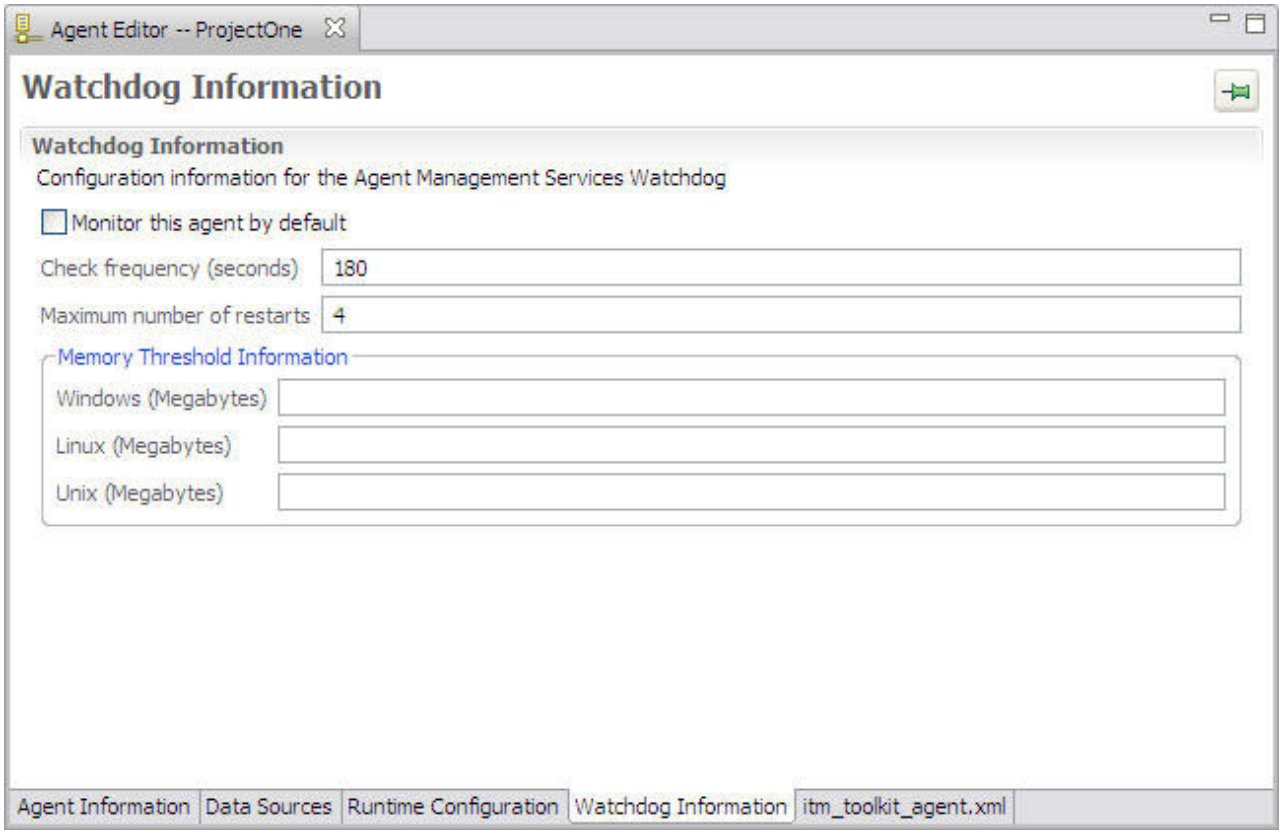


Figure 23. Watchdog Information page

You can specify the following configuration information for the Agent Watchdog:

- **Monitor this agent by default**

  Select this check box to put the agent under management by Agent Management Services when the agent is installed. The agent is monitored for unhealthy behavior or abnormal termination and is restarted by a watchdog.
- **Check frequency (seconds)**

How often the watchdog checks the agent process for unhealthy behavior or abnormal termination. The default is every 180 seconds.

- **Maximum number of restarts**

  Number of times the Watchdog restarts the agent because of unhealthy behavior or abnormal termination in a 24-hour period before it alerts the administrator of the problem. The period starts at midnight each day. So, the first period from when the agent is started might be "short."

  A restart occurs if the agent goes down for any reason or if the Watchdog must stop the agent because it is unresponsive or unhealthy. For example, the Watchdog stops and then starts the agent if the memory threshold is crossed. The default is four restarts in a 24-hour period, where the period is measured from midnight to 11:59 p.m. At midnight, the daily restart count for the agent returns to 0 automatically.
- **Memory Threshold Information**

  Size of the agent process (in megabytes) to which the agent can grow before its watchdog deems it unhealthy. There is a separate value for Windows, Linux, and UNIX. If the agent process grows beyond the threshold, the watchdog stops the process and restarts it. There are no defaults for these properties. If no value is specified, the Watchdog does not monitor the process size. The metric uses the working set size on Windows, and the user memory on UNIX and Linux.

If the Watchdog stops the agent, and the maximum number of restarts is reached, the Watchdog sends an alert that the agent exceeded its restart count, and stops doing auto-restarts. The Watchdog still reports whether the agent is up or down assuming it is started in another manner such as through the Tivoli Enterprise Portal.

You must manually restart the agent by using the AMS Start Agent Take Action command so the restart count does not get reset.

The count gets reset in one of the following ways (the Watchdog continues to work and report status, but does not do auto-restarts):

- The clock strikes midnight.
- The user uses the AMS Start Agent Take Action command, which has an input parameter called **resetRestartCount**. If you enter a value of 1 (meaning “true”or “yes”), the daily restart count resets back to 0.

For more information, see the following sections in the *IBM Tivoli Monitoring Administrator's Guide*:

- For Tivoli System Monitor Agents

  Configuring Agent Management Services on Tivoli System Monitor Agents
- For Tivoli Enterprise Monitoring Agents

  Installing and configuring Tivoli Agent Management Services

Cognos information

Use the Cognos Information page to specify the information that is used when a Cognos data model is generated for your agent.

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35.

Procedure

1. To open the Cognos Information page, click **Cognos Information** in the **Agent Content** section of the Agent Information page or the **Cognos Information** node in the Outline View.

2. In the **Data Source** field, enter the name of the data source that connects Tivoli Common Reporting to the IBM Tivoli Data Warehouse. The default value is `TDW`.
3. In the **Schema** field, enter the name of the database schema that is used for the Tivoli Data Warehouse, which is used to fully qualify table names in Cognos reports. The default value is `ITMUSER`. This value can be changed in Framework Manager when the generated Cognos model is loaded into Framework Manager.

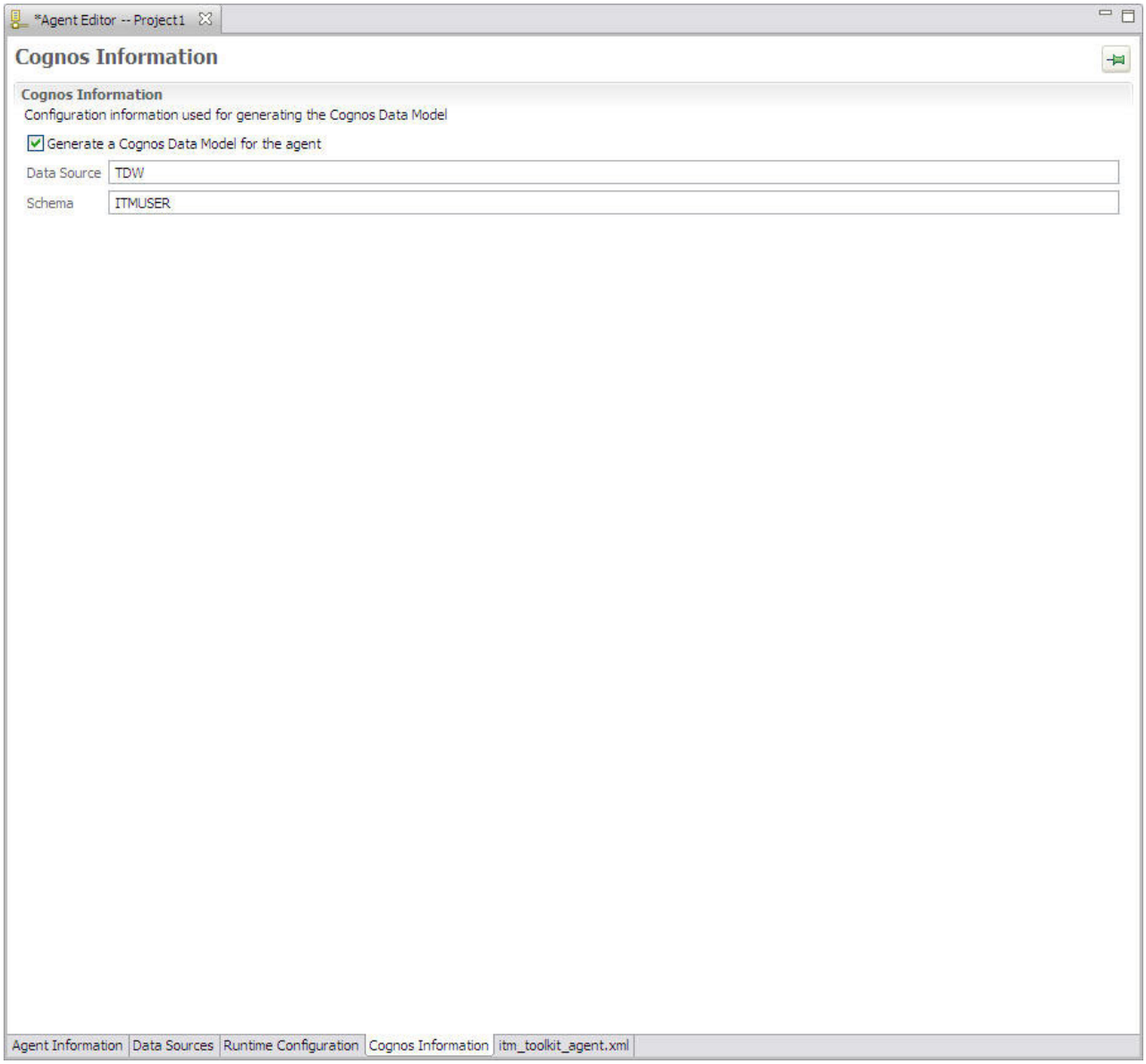


Figure 24. Cognos Information page

The **Add this attribute group to a reporting category** check box in the Data Source Definition page Figure 25 on page 44 determines where in the Cognos model the attribute group is placed. If not selected, the attribute group is placed in the extended attributes folder in the Cognos model. If selected, the attribute group is placed in the selected subfolder (availability or performance) in the Key Metrics folder.

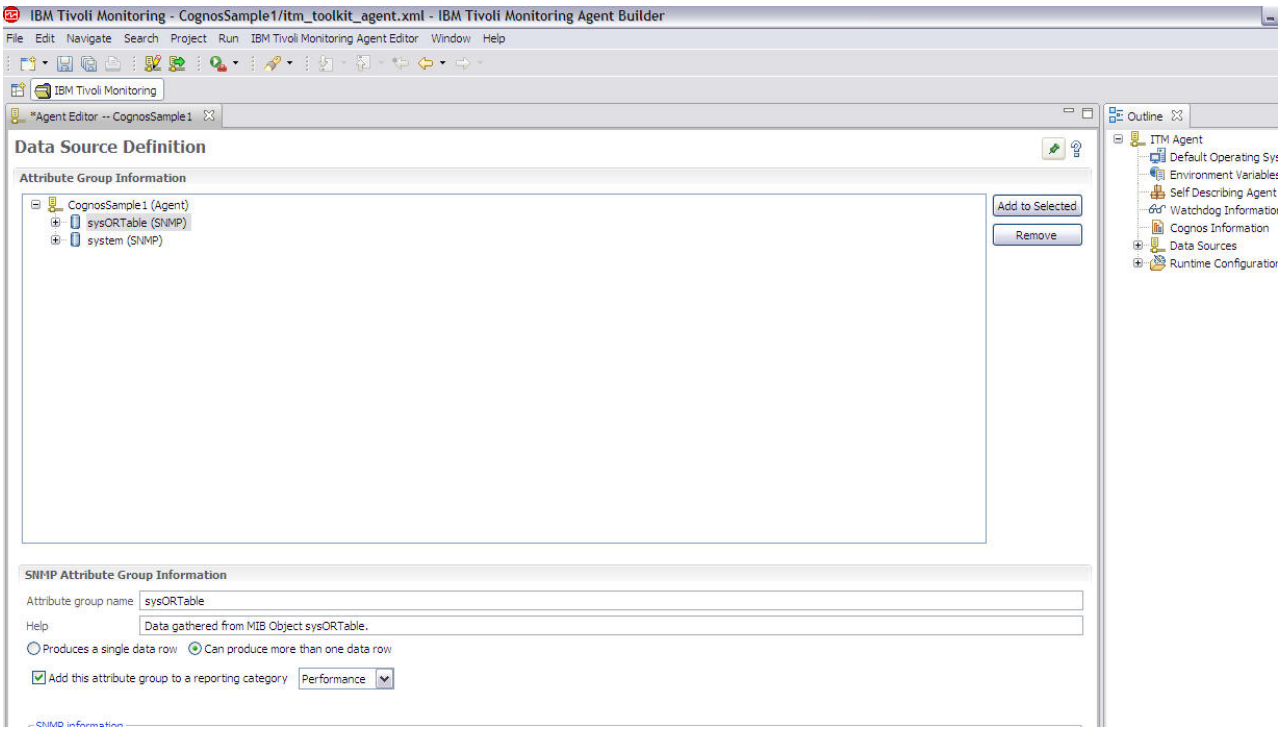


Figure 25. Data Source Definition page for Cognos

For more information about the data source fields, see Table 6 on page 55.

What to do next

You can use the Cognos data model to create Tivoli Common Reporting reports for your agent, see Appendix E, "Cognos data model generation," on page 609.

Generate Agent wizard link

When you finish creating or editing the new agent, use the Generate Agent wizard to prepare the installation.

Before you begin

To create an agent, see Chapter 5, "Creating a basic agent," on page 13.

For more information about the Agent Editor and Agent Information page, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.

Procedure

When you finish creating or editing the new agent, on the Agent Editor Agent Information page, click the **Generate Agent Wizard** link.
With the Generate Agent wizard, you can:

- Generate the agent files with a Tivoli Monitoring installation on the local system. See ("Installing the agent locally" on page 387).
- Create a package so the agent can be installed on other systems. See ("Creating the agent image" on page 389).

- Generate a Tivoli Monitoring 5.x mapping file. See (Appendix F, "Upgrading custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6.2 agents," on page 625).

Committing a version of the agent

Commit your agent when you are certain you are finished developing this version of the agent and you are ready to deliver it.

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.

About this task

Tivoli Monitoring systems require that new versions of an agent include all of the information that is contained in the previous versions of that agent. Including all information from previous versions is necessary so workspaces, situations, and queries continue to work with a mixture of old and new agents. After you complete developing and testing an agent, you must commit the agent.

There is a limit of 1024 versions.

Procedure

1. Open the Agent Editor window, Agent Information page.
2. In the **Commit Agent Version** area, click **commit this level**.
3. Back up the committed agent or check it into your version control system.

What to do next

After you commit an agent, any additional changes to the agent are part of a new version. You must enter the new version number before the additional changes can be saved. Any changes to the new version must not break compatibility with previous versions of the agent.

After you commit the agent, you cannot complete these actions on objects that existed before the agent was committed:

- Delete attributes from an attribute group.
- Delete attribute groups.
- Reorder existing attributes in an attribute group.
- Reorganize existing attribute groups (by using Navigator items).
- Move attribute groups or navigator groups into or out of subnodes.
- Rename attribute groups.
- Rename attributes.
- Change data types of existing attributes.
- Change a subnode name or type if it contains an attribute group that existed before the agent was committed.
- Change a company identifier or agent identifier for the agent.
- Change the product code of the agent. For more information, see ("Changing the product code" on page 52).

You can complete the following actions after you commit the agent:

- Add new attributes to existing attribute groups.
- Add new attribute groups.

- Reorder new attributes.
- Organize new attribute groups by using navigator items.
- Create new subnode types.
- Add new queries.
- Add new situations.
- Add new workspaces.

The Data Source Definition page

Use the Data Source Definition page to manipulate data sources.

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.

About this task

The Data Source Definition page (Figure 26 on page 47) lists the data sources that are configured for the IBM Tivoli Monitoring Agent. When you select a data source or attribute in the tree, the page is updated to display the properties for the selected object. Use the fields to modify the properties for the data source or attribute selected.

Note: For detailed information and procedures that concern monitoring with data sources, see the topics in this documentation about the individual data sources.

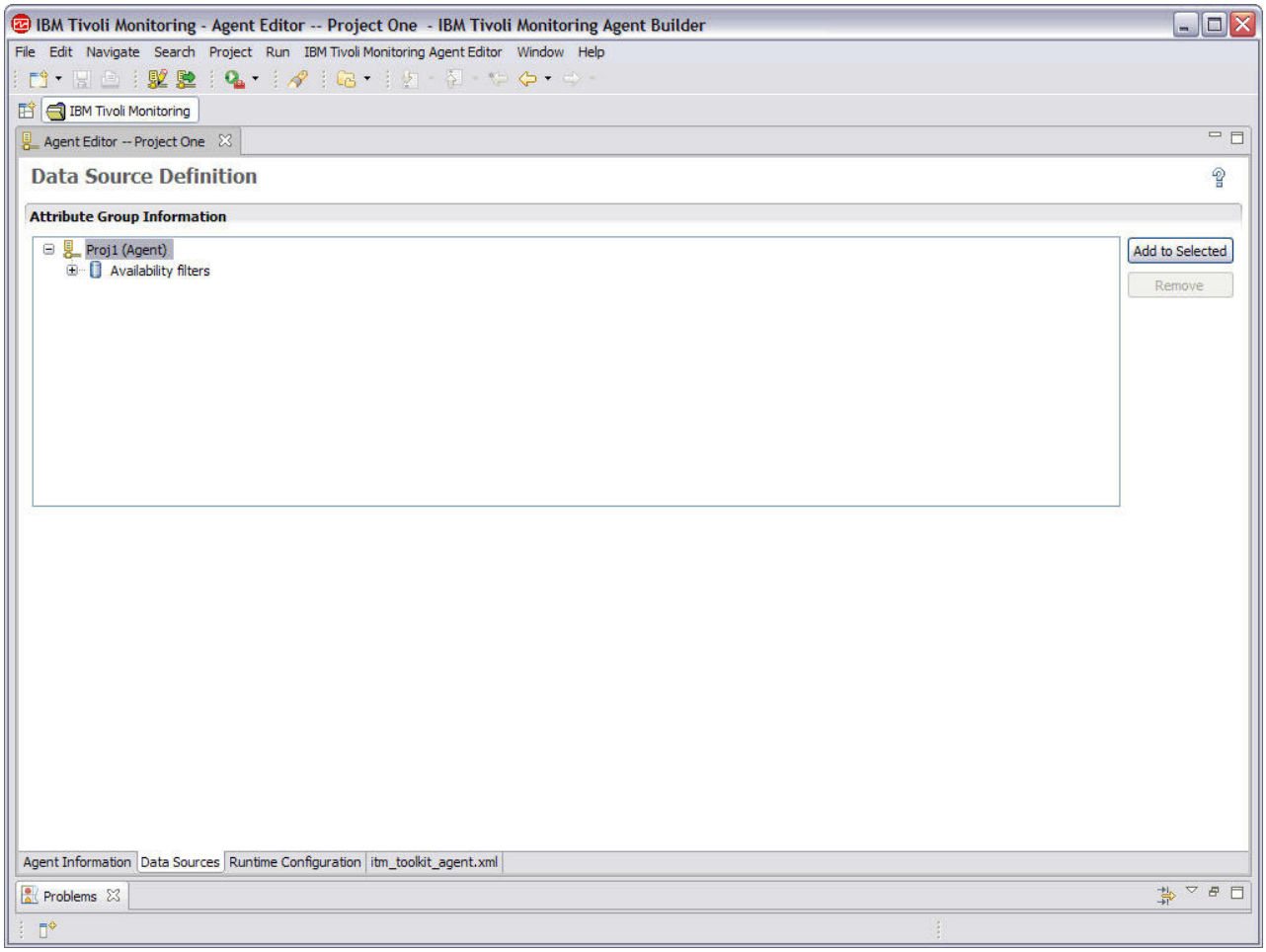


Figure 26. Data Source Definition page

Procedure

- To open the Data Source Definition page, click **Data Sources** in the Agent Content section of the Agent Information page or the **Data Sources** node in the Outline view.
- You can add more data sources by clicking **Add to Selected** or right-clicking in the navigation tree and selecting one of the options.
- You can remove data sources and attributes by right-clicking on them and selecting **Remove**.

Copying data sources by using the Data Source Definition page

Use the Data Source Definition page to copy data sources.

Before you begin

Go to the Data Source Definition page. For more information, see "The Data Source Definition page" on page 46

About this task

Data sources that result in Tivoli Monitoring attribute groups can be copied to the clipboard and pasted back to this agent or another agent. Data sources that do not result in attribute groups are Availability and Windows Event Log data sources.

Procedure

1. Select the attribute groups that you want to copy.
2. Cut or copy the attribute group by using one of the following choices:
   - Click **Edit** > **Cut** > **Edit** > **Copy** from the menu bar.
   - Right-click one of the selected items and click **Cut** or **Copy** from the menu.
   - Use one of the operating system or Eclipse key strokes that calls the cut or copy action. For example, on Windows systems, pressing **Ctrl-C** calls the copy action.

   Your choice depends on whether you want to remove the data sources that are placed on the clipboard or leave the data sources in place. To remove data sources, use **Cut** to leave data sources on the clipboard, use **Copy**.
3. Select the parent of an attribute group (the agent, a subnode, or a navigator group) or select an existing attribute group.
4. Paste the selection by using one of the following choices:
   - Select **Edit** > **Paste** from the menu bar.
   - Right-click the node where you want to paste the selection in the tree, and click **Paste** on the menu.
   - Use one of the operating system or Eclipse key strokes that calls the paste action. For example, on Windows systems, pressing **Ctrl-V** calls the paste action.

Results

The attribute groups from the clipboard are placed in the selected parent. Alternatively, if an attribute group is selected, the attribute groups are placed or in the parent of the selected attribute group.

If there is a name conflict with another attribute group while pasting, the pasted attribute group name is changed slightly to avoid the conflict.

What to do next

You can specify the name and help text for a navigator group by selecting an Agent, right-clicking and selecting **New Navigator Group**. A navigator group is a way of grouping data sources. A navigator group must contain at least two data sources. You can drag data sources into navigator groups as wanted, or drag them out the navigation tree.

Runtime Configuration Information page

The Runtime Configuration Information page displays the values that are defined during configuration.

These values are made available to command return codes and scripts through the environment. To open the Runtime Configuration Information page, click **Runtime Configuration** in the Agent Content section of the Agent Information page or the **Runtime Configuration** node in the Outline View. The Agent Builder automatically constructs the name of the environment variable from the product code and the label. You can add and change the configuration properties by using the Runtime Configuration Information page, see (“Changing configuration properties by using the Agent Editor” on page 369).

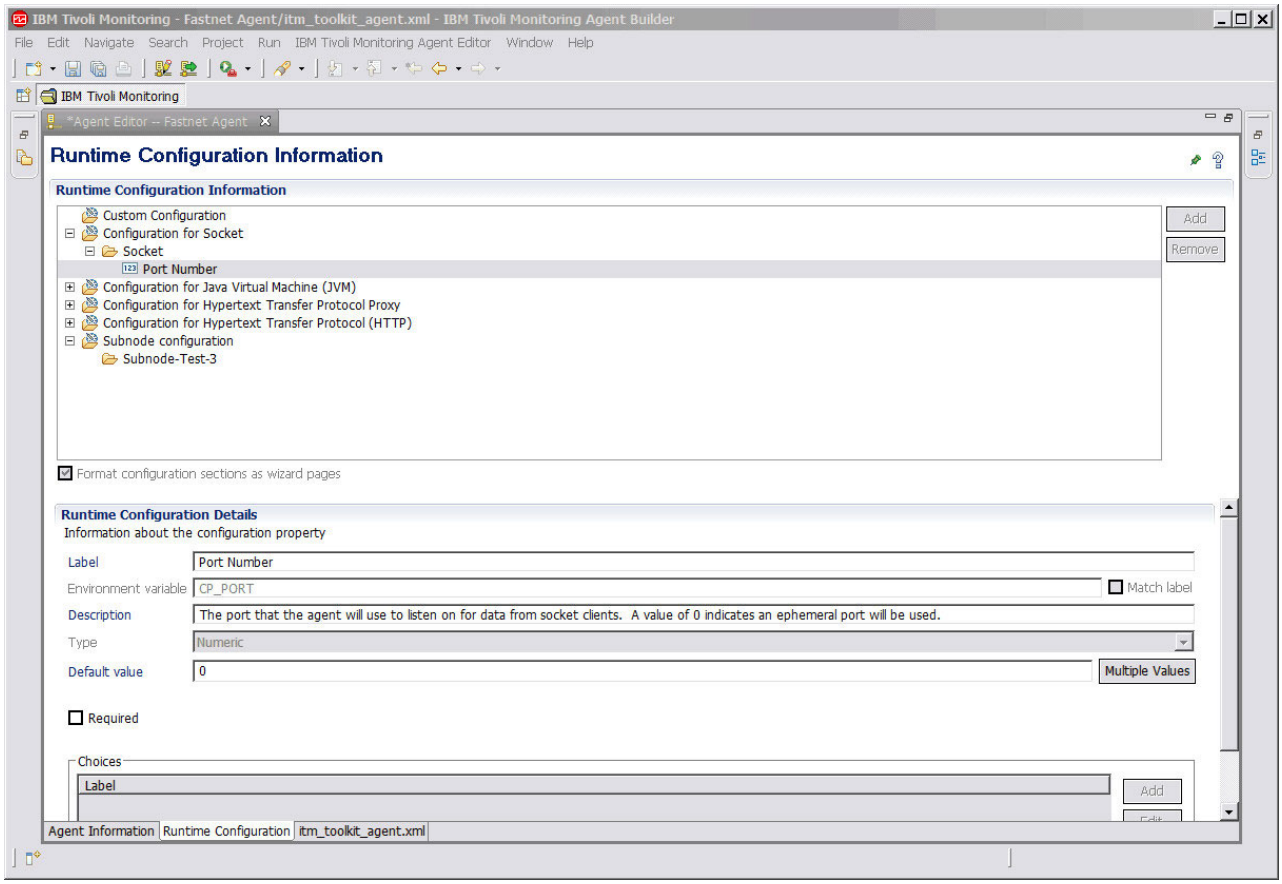


Figure 27. Runtime Configuration Information page

Related concepts:

Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35
Use the Tivoli Monitoring Agent Editor to change, save, and commit a version of your agent.

Agent XML Editor page

The Agent XML Editor page displays the XML for the IBM Tivoli Monitoring Agent definition.

Changes that are made in the XML are reflected in the graphical user interface (GUI) pages. The changes are reflected in the GUI when you save the file or switch to a different page in the editor. In addition, changes made in the GUI pages are reflected in the XML page when you switch to it.

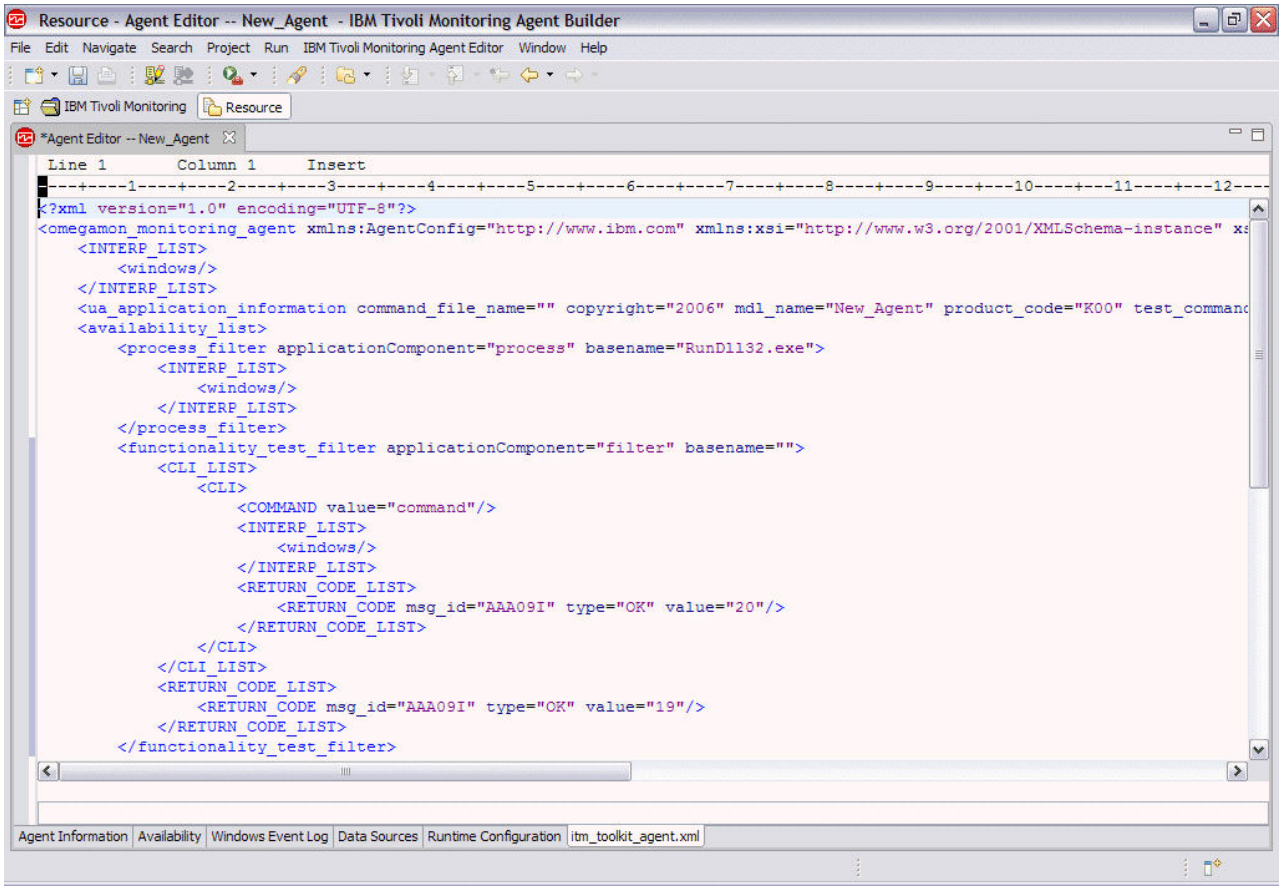


Figure 28. Agent XML Editor page

Attention: To completely document the Agent Builder, information about the XML page is included here. The inclusion of this information must not in any way be interpreted as a recommendation to use the XML page to alter the XML. If you use the Agent XML Editor to alter any part of the XML, your changes can cause errors that prevent you from generating your agent.

Related concepts:

Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35
Use the Tivoli Monitoring Agent Editor to change, save, and commit a version of your agent.

Saving your edits and changes

Changes that you make with the editor are not stored until you save them.

Procedure

Perform a save in one of the following ways:

- Select **File** > **Save**, selecting the save (diskette) icon.
- Press **Ctrl+S**

When you save, a validation occurs to ensure that the information is complete. If problems occur, information about the error is displayed in the Eclipse Problems view. If this view is not visible, select **Window** > **Show View** > **Problems**. If you attempt to generate an agent that has errors, an error message is displayed.

Note: You must correct all errors and save before you can generate and install the agent.

Committing a version of the agent

Commit your agent when you are certain you are finished developing this version of the agent and you are ready to deliver it.

Before you begin

For more information about the Agent Editor and Agent Information page, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.

About this task

Tivoli Monitoring systems require that new versions of an agent include all of the information that is contained in the previous versions of that agent. Including all information from previous versions is necessary so workspaces, situations, and queries continue to work with a mixture of old and new agents. After you complete developing and testing an agent, you must commit the agent.

There is a limit of 1024 versions.

Procedure

1. Open the Agent Editor window, Agent Information page.
2. In the **Commit Agent Version** area, click **commit this level**.
3. Back up the committed agent or check it into your version control system.

What to do next

After you commit an agent, any additional changes to the agent are part of a new version. You must enter the new version number before the additional changes can be saved. Any changes to the new version must not break compatibility with previous versions of the agent.

After you commit the agent, you cannot complete these actions on objects that existed before the agent was committed:

- Delete attributes from an attribute group.
- Delete attribute groups.
- Reorder existing attributes in an attribute group.
- Reorganize existing attribute groups (by using Navigator items).
- Move attribute groups or navigator groups into or out of subnodes.
- Rename attribute groups.
- Rename attributes.
- Change data types of existing attributes.
- Change a subnode name or type if it contains an attribute group that existed before the agent was committed.
- Change a company identifier or agent identifier for the agent.
- Change the product code of the agent. For more information, see ("Changing the product code" on page 52).

You can complete the following actions after you commit the agent:

- Add new attributes to existing attribute groups.
- Add new attribute groups.
- Reorder new attributes.
- Organize new attribute groups by using navigator items.

- Create new subnode types.
- Add new queries.
- Add new situations.
- Add new workspaces.

Changing a subsequent version of your agent

To save changes to a committed agent, you must first enter a new version number.

Procedure

1. Open the Agent Editor window, Agent Information page.
2. Enter a new version or patch level after the Version prompt.
3. Make the edits your agent. If you forget to change the agent version, you are prompted for the new version when you save any of your changes.

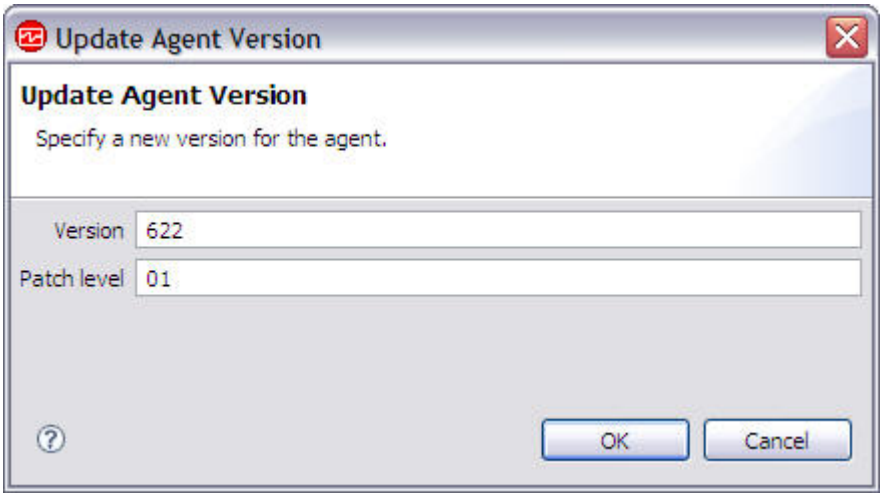


Figure 29. Updating the agent version after commit

Changing the product code

If you change the product code, you have an agent that is incompatible with any previous version of the agent. Any records of previous commit actions are lost and you are developing a new agent.

Any files, situations, Take Action commands, or workspaces that you exported from IBM Tivoli Monitoring and imported into the agent are deleted from the agent. If you try to change the product code, you are reminded of this deletion and asked if you want to continue as shown in Figure 30 on page 53.

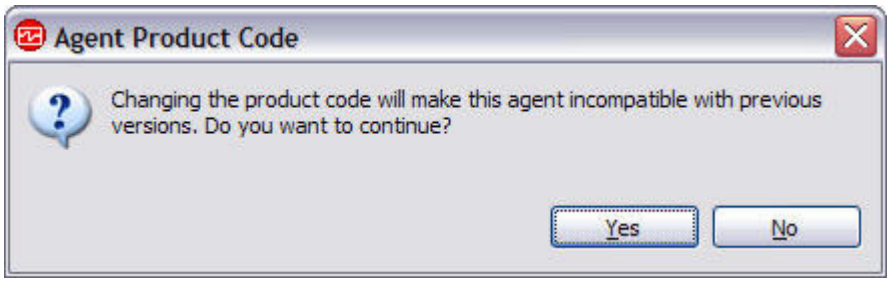


Figure 30. Agent Product Code change

When you click **Yes** in the Agent Product Code window you are warned that the contents of the agent support files are no longer valid. You are also warned that the files will be removed next time the agent is saved.

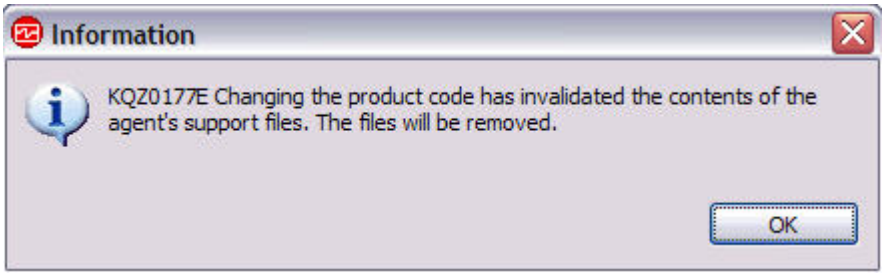


Figure 31. Agent support files invalidated

Chapter 7. Editing data source and attribute properties

There are a number of ways to begin editing a data source

About this task

This documentation covers the following topics that relate to editing data source and attribute properties:

- "Creating attributes" on page 56
- "Copying attributes" on page 62
- "Editing attributes" on page 62
- "Removing attributes" on page 63
- "Creating derived attributes" on page 63
- "Editing derived attributes" on page 67
- "Filtering attribute groups" on page 67
- "Formula Editor" on page 68
- "Formula operators and functions" on page 82
- "Specifying operating systems" on page 88

Procedure

To edit or remove information from a data source (attribute group):

1. In the Agent Content area of the Agent Information page click **Data Sources**. The Data Source Definition page opens.
2. Select the data source (attribute group).

   The attribute group information area of the page is updated to display the properties for the selected data source.

   Note: Alternatively, if you are on the last page of the New Agent wizard, you can double-click the data source to open the Attribute Group Information window. This window has the same information as the attribute group information area of the Data Source Definition page.

   (Table 6) describes the field information that is applicable to all of the data sources. Use the fields to modify the properties for the data source or attribute selected.

Table 6. Fields for editing data sources

Field name	Description	Acceptable values and examples
Attribute group name	Name of the data source as it is displayed in the Tivoli Enterprise Portal	Acceptable values: Descriptive string less than or equal to 32 characters long. It must be unique within the agent. The first character must be a letter and remaining characters can be letters, numbers, or underscores. An underscore is displayed as a space in the Tivoli Enterprise Portal. Do not use spaces or special characters.
Help text	Help text for the data source	Acceptable values: String up 256 characters long.

© Copyright IBM Corp. 2007, 2013

Table 6. Fields for editing data sources (continued)

Field name	Description	Acceptable values and examples
Produces a single data row	The data source returns 1 row of data. Editable in all sampled data sources.	Example: If you are monitoring physical system memory, choose a single row. A system typically manages all of its memory in a single pool; so only one row of data can be returned.
Can produce more than one data row	The data source can return any number of rows of data. Editable in all sampled data sources.	Example: If you are monitoring disk drives, choose multiple rows because there can be more than one disk in a system. For keys, choose the attributes that distinguish a disk from another. For a disk, the key attribute is disk number, drive letter, volume label, or whatever is appropriate in your environment.
Produces Events	The data source returns event-based data, 1 row of data per event.	Example: An SNMP event-based data source sends notifications (traps) as performance thresholds are crossed. **Note:** Not all data sources can produce events.
Add this attribute group to a reporting category	The category in the generated Cognos model to which the attributes in this attribute group are assigned.	Select the check box to place the attribute group in the selected subfolder (Availability or Performance) in the Key Metrics folder. If the check box is not selected, the attribute group is placed in the **Extended Metrics** folder in the Cognos data model.
Metric Category	The category to which the attributes in this attribute group are assigned.	Select either **Performance** or **Availability**.

Note:

a. The **Produce a single data row** and **Can produce more than one data row** fields do not affect data for an event data source.

b. For more about sampled and event data types, see ("Data types" on page 430).

c. For information about the fields for a specific data source, see the relevant data source chapter in this documentation ("Data Sources" on page 25).

Creating attributes

Adding, modifying, and removing attributes.

Before you begin

Open the Data Source Definition page. For more information, see "The Data Source Definition page" on page 46.

About this task

Attributes in your attribute group can be modified or removed, or other attributes can be added as needed.

Add or edit attributes by using the following procedure:

Procedure

1. Right-click the data source and select **Add Attribute** on the menu. The Attribute Information page is displayed (Figure 32).

   Note: The page that is displayed depends on the data source for the attribute.
2. Specify your choices for the new attribute on the Attribute Information page.

   See "Fields and options for defining attributes" on page 58 for information about the fields and options.

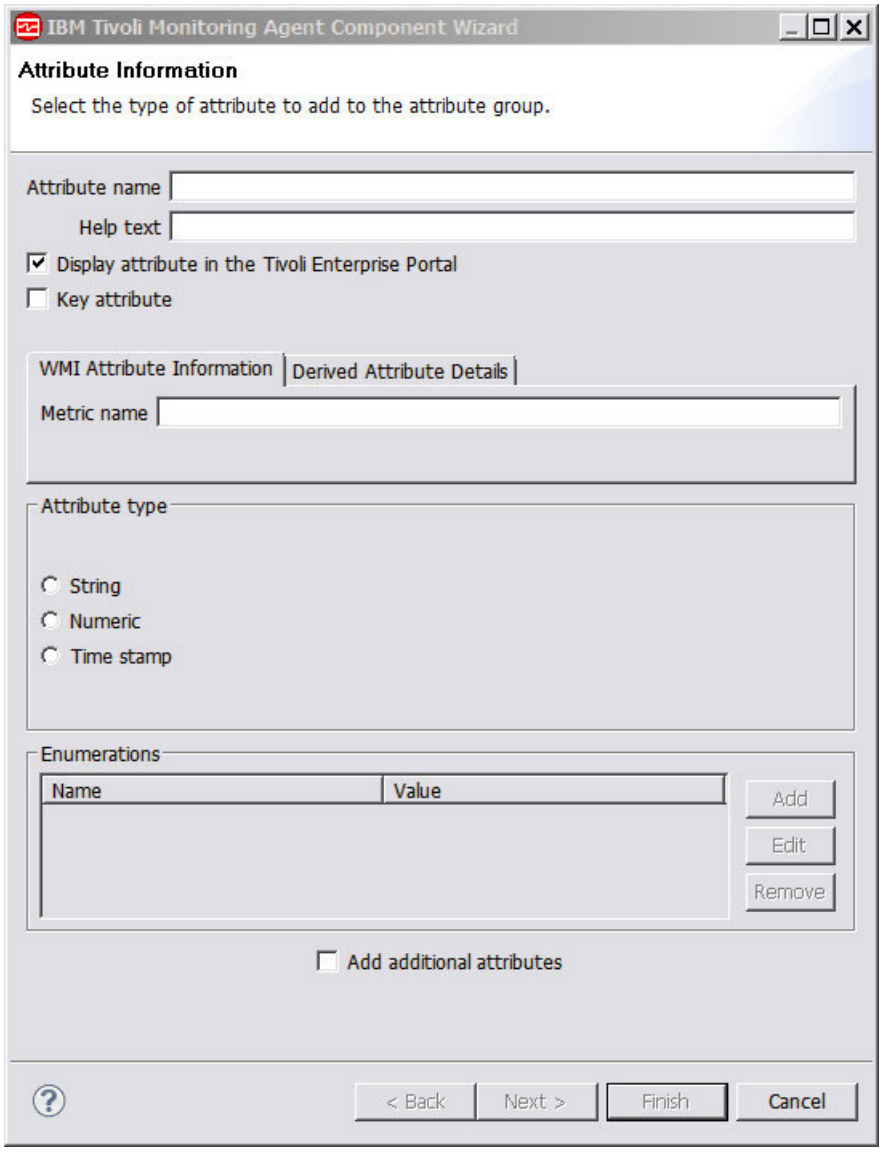


Figure 32. Attribute Information page

3. To add more attributes, select **Add additional attributes** and click **Next**.
4. When finished adding attributes, click **Finish**.

Fields and options for defining attributes

Description of the field information and options for the **Attribute Information** page that are applicable to all of the data sources

For information about the specific field information for each of the data sources, see the relevant documentation for each data source.

Table 7. Fields and options for defining attributes

Field names/options	Description	Acceptable values
Attribute name	Name of the attribute as it is displayed in the Tivoli Enterprise Portal	String with the following characters: • A-Z • _ • a-z • 0-9 **Note:** The name must start with A-Z or a-z. The attribute name has a limit of 63 characters and the attribute group name has a limit of 63 characters
Help text	Help text for the attribute	String
Display Attribute in the Tivoli Enterprise Portal	Attribute is displayed in the Tivoli Enterprise Portal (see note in last row)	Not applicable
Key Attribute	Attribute is a key in the table. Check whether this attribute helps to uniquely define the object that is being reported on. If the data is warehoused and summarized, the key attributes are used to roll up data in the summary tables.	This option is not available for Perfmon attributes.
Attribute Information tab	The contents of this tab depend on the type of data source to which this attribute belongs. See information in the chapter for the data source you want to monitor for more details. If you use this tab, do not use the **Derived Attribute Details** tab.	
Derived Attribute Details tab **Formula** **Insert Attribute**	Use the **Derived Attribute Details** tab only if you want a derived attribute, and not an attribute directly from the data source. If you use this tab, do not use the **Attribute Information** tab. In the **Formula** field, enter a formula to calculate the value of the attribute that is based on other attributes or constants. You can type the formula in the **Formula** field or click **Edit** to use the graphical formula editor. See (“Formula Editor” on page 68).	

Table 7. Fields and options for defining attributes (continued)

Field names/options	Description	Acceptable values
Attribute type	Describes how the attribute is displayed in the Tivoli Enterprise Portal. There are 3 types: • String • Numeric • Time stamp "Attribute types" contains more information about the attribute types.	Table 8 on page 60 contains descriptions of the numeric attribute type values.
Enumerations	Can be a numeric with scale zero or string value.	Add your enumerations to the table by using the procedure in ("Specifying an enumeration for an attribute" on page 62). The enumeration name is displayed in the Tivoli Enterprise Portal when the corresponding Value is received in the attribute from the agent.This attribute is used for a set of specific values with identified meanings (for example, 1=UP, 2=DOWN).
Note: In cases where the attribute is used in calculations with other attributes, there are reasons not to display the base value in the Tivoli Enterprise Portal. For instance, a number that represents a byte count wraps so quickly that it is of little use.		

Attribute types

There are three attribute types

The three types of attributes are:

- String
- Numeric
- Time stamp

String attributes

When you select **String**, you specify the maximum length of the string in bytes as shown in Figure 33. The default size is 64 bytes.

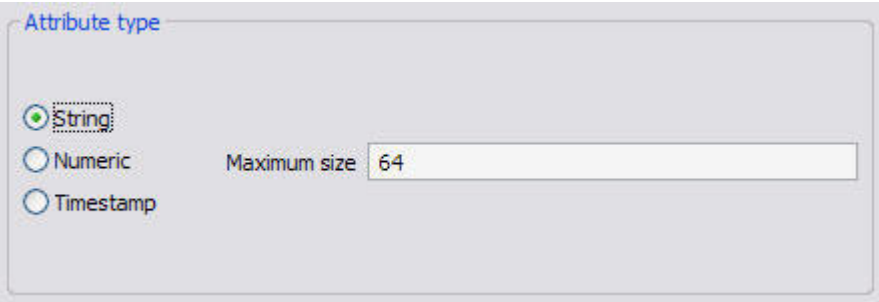


Figure 33. String attribute type

A string value can contain any UTF-8 character. The maximum size is the total length of the buffer that is allocated to contain the string in bytes. Some non-ASCII UTF-8 characters take more than 1 byte, so you must account for this space when you select a maximum size. Data aggregation in the warehouse displays the latest value that is collected during the period.

Numeric

When you specify **Numeric**, you have several variations to choose from as shown in Figure 34.

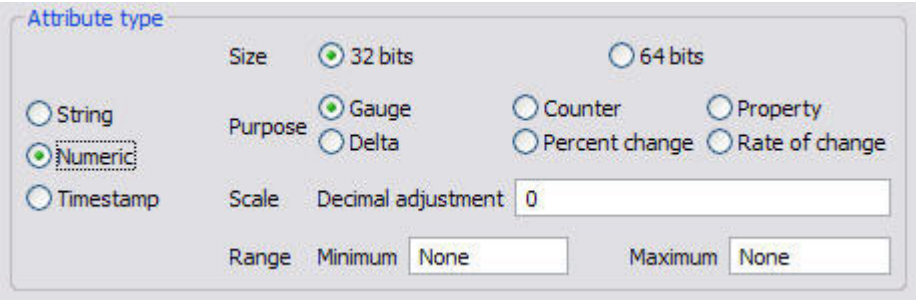


Figure 34. Numeric attribute type

See Table 8 for information about the options for each numeric type of attribute.

Time stamp

A Time stamp attribute is a string attribute with a format that conforms to the `CYYMMDDHHMMSSmmm` format (where C=1 for the 21st century). All 16 characters must be used for scripts or socket clients. When displayed in the Tivoli Enterprise Portal, a time stamp attribute type is displayed in the correct format for the locale.

When you use the browse feature for WMI, the Agent Builder automatically marks attributes whose CIM type is `CIM_DATETIME` as time stamps. The data provider automatically converts WMI attributes to this format.

Numeric aspects of attributes

Descriptions of the size, purpose, scale, and range aspects of attributes.

When you specify a numeric attribute, you must specify the size, purpose, scale, and range of the attribute. For more information, see (Table 8).

Table 8. Numeric attribute options

Numeric aspects	Options and fields	Description
Size	**32 bits** **64 bits**	The value of 32-bit numbers can range from `-2147483648` to `2147483647` (roughly `-2,000,000,000` to `2,000,000,000`). The value of 64-bit numbers can range from `-9223372036854775808` to `9223372036854775807` (roughly $-9x10^{18}$ to $9x10^{18}$)

Table 8. Numeric attribute options (continued)

Numeric aspects	Options and fields	Description
Purpose	**Gauge**	Integer values where the raw values returned are larger or smaller than previous values. Negative values are supported. This type is the default type for integers. Data aggregation in the warehouse produces minimum, maximum, and average values.
	Counter	A positive integer value that contains raw values that generally increase over time. Data aggregation in the warehouse displays the total, high, low, and latest delta values. In the following example of Delta-based calculations, detailed data values in one hour are 9, 15, 12, 20, 22, and delta-based processing has the following rules: • If the current value is greater than or equal to the previous value, the output equals the previous value minus the current value • If the current value is less than the previous value, the output equals the current value • Because 15 is greater than 9, the output equals 6 • Because 12 is less than 15, the output equals 12 • Because 20 is greater than 12, the output equals 8 • Because 22 is greater than 20, the output equals 2 • The TOT_ value is 28, which is the total of outputs • The LOW_ value is 2, which is the lowest of outputs • The HI_ value is 12, which is the highest of outputs
	Property	A property of the object that does not frequently change. Data aggregation in the warehouse displays the latest value that is collected during the period.
	Delta	An integer value that represents the difference between the current value and the previous value for this attribute. Because this attribute is represented as a gauge in the warehouse, data aggregation in the warehouse produces minimum, maximum, and average values.
	Percent change	An integer value that represents the percent change between the current value and the previous value. This type is calculated as: `((new -old)*100)/old`. Because this type is represented as a gauge in the warehouse, data aggregation in the warehouse produces minimum, maximum, and average values.
	Rate of change	An integer value that represents the difference between the current value and the previous value, which is divided by the number of seconds between the samples. It converts a value (such as bytes) to the value per second (bytes per second). Because this type is represented as a gauge in the warehouse, data aggregation in the warehouse produces minimum, maximum, and average values.
Scale	**Decimal adjustment**	Scale determines how many decimal places are in the number. Each decimal place reduces the range that is mentioned earlier by a factor of 10. For example, a decimal adjustment of 2 shows two decimal places, and in a 32-bit number the allowable range becomes `-21474836.48` to `21474836.47`. When a non-zero decimal adjustment is specified, the number is manipulated internally as a floating point number. Therefore, the precision of large 64-bit numbers might be reduced.
Range	**Minimum** **Maximum**	Range gives the expected range of the value. If no minimum or maximum ranges are given, the maximum values that are described earlier are used. The range is used to produce a more useful initial view in some graphical Tivoli Monitoring workspace views.

Specifying an enumeration for an attribute

Specify a value enumeration by using the Attribute Information page.

About this task

Specifying an enumeration for an attribute involves a short procedure. When a value is encountered that has a defined enumeration, the enumeration name is displayed in the Tivoli Enterprise Portal instead of the value.

Procedure

1. In the Attribute Information page **Attribute type** area, click **Numeric**.
2. In the **Enumerations** area, click an enumeration, and click **Add**. The Enumeration Definition window is displayed.

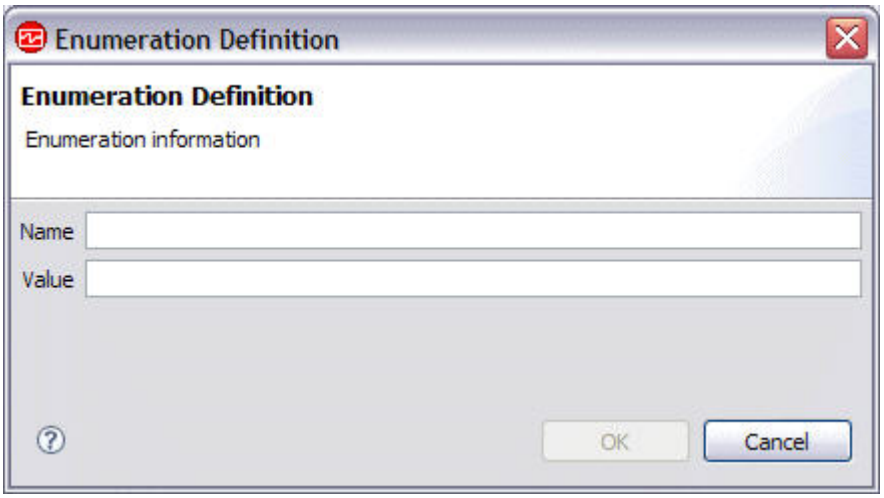


Figure 35. Defining enumeration for an attribute

3. Type the name and value of the enumeration in the fields in the window.
4. Click **OK**.

Copying attributes

You can copy attributes from the Data Source Definition page.

Procedure

1. In the Agent Editor, Data Source Definition page, right-click the attribute that you want to copy, and click **Copy Attribute**.
2. In the Copy Attribute window, type the name of the new attribute in the **Name** field, and click **OK**.

Editing attributes

You can edit and change attribute information by using the Agent Editor.

Before you begin

Open the Data Source Definition Page. For more information, see “The Data Source Definition page” on page 46.

Procedure

1. Select the attribute that you want to edit. The Attribute Information pane of the page is updated to show the properties for the selected attribute.
2. Specify your choices for the new attribute information.

   Note: On the last page of the New Agent wizard (the Data Source Definition page), you can double-click the attribute to open the Attribute Information window. That window contains the same information as the Attribute Information pane of the Data Source Definition page.

Removing attributes

Remove an attribute or attributes.

Before you begin

Open the Data Source Definition Page. For more information, see "The Data Source Definition page" on page 46.

Procedure

To remove an attribute or attributes, right-click the attribute or attributes and select **Remove** from the menu that is displayed.

Note: You cannot remove an attribute that is used by a derived attribute. You must first remove the reference by the derived attribute to the attribute you are removing.

Creating derived attributes

You can create an attribute that derives its value from other attributes instead of directly from the data source.

Before you begin

Open the Data Source Definition Page. For more information, see "The Data Source Definition page" on page 46.

About this task

In the derived attribute, you can perform operations on the values of the source attributes. For example, you can perform basic arithmetic operations on numeric attributes or string concatenation on string attributes.

The basic expression syntax that is used for derived expressions contains functions. These functions provide a more complicated manipulation of data that includes short-term aggregation, conversion from string to integer, and accessing configuration properties and environment variables. In addition, an editor helps you visualize the expression as it is being built.

Procedure

1. On the Data Source Definition page, right-click the data source and click **Add Attribute**.
2. On the Attribute Information page, type an Attribute name and Help text.

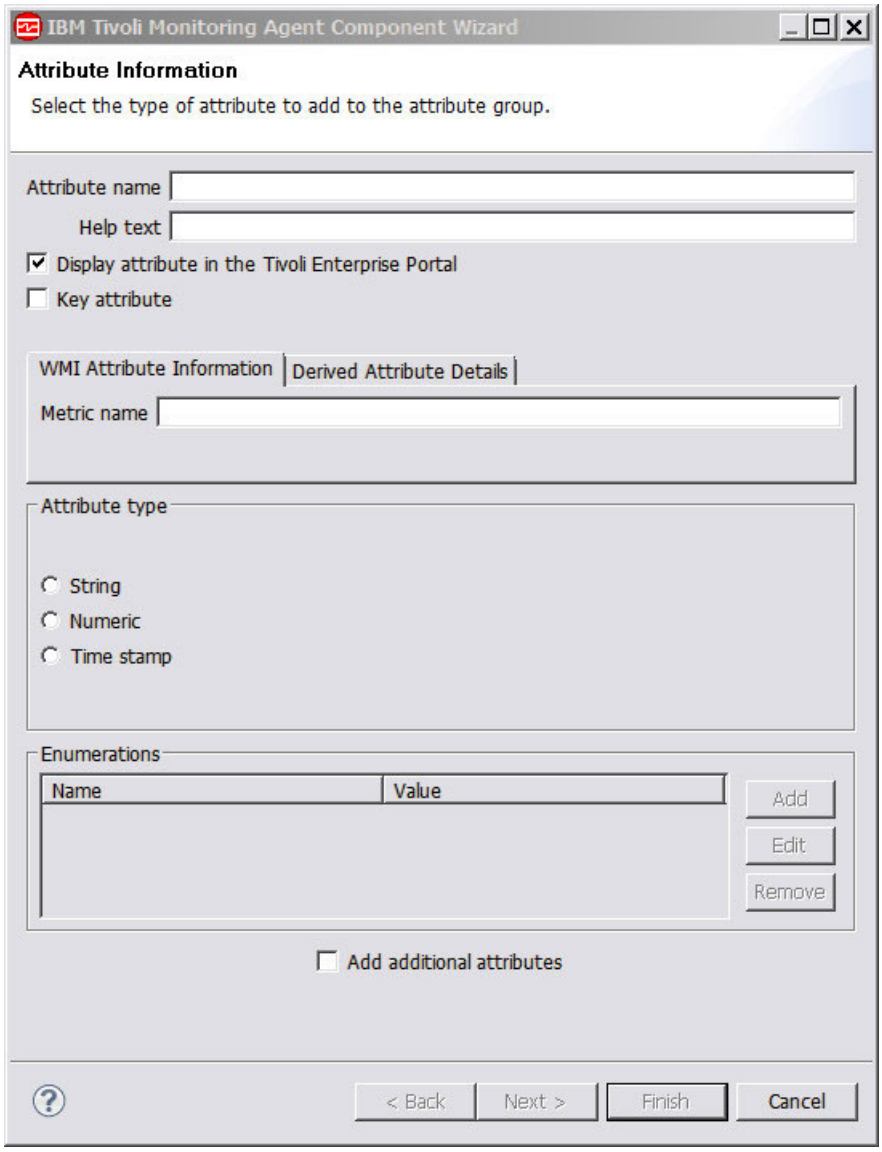


Figure 36. Attribute Information page

3. Click the **Derived Attribute Details** tab to create an attribute that derives its value from other attributes instead of directly from the data source.

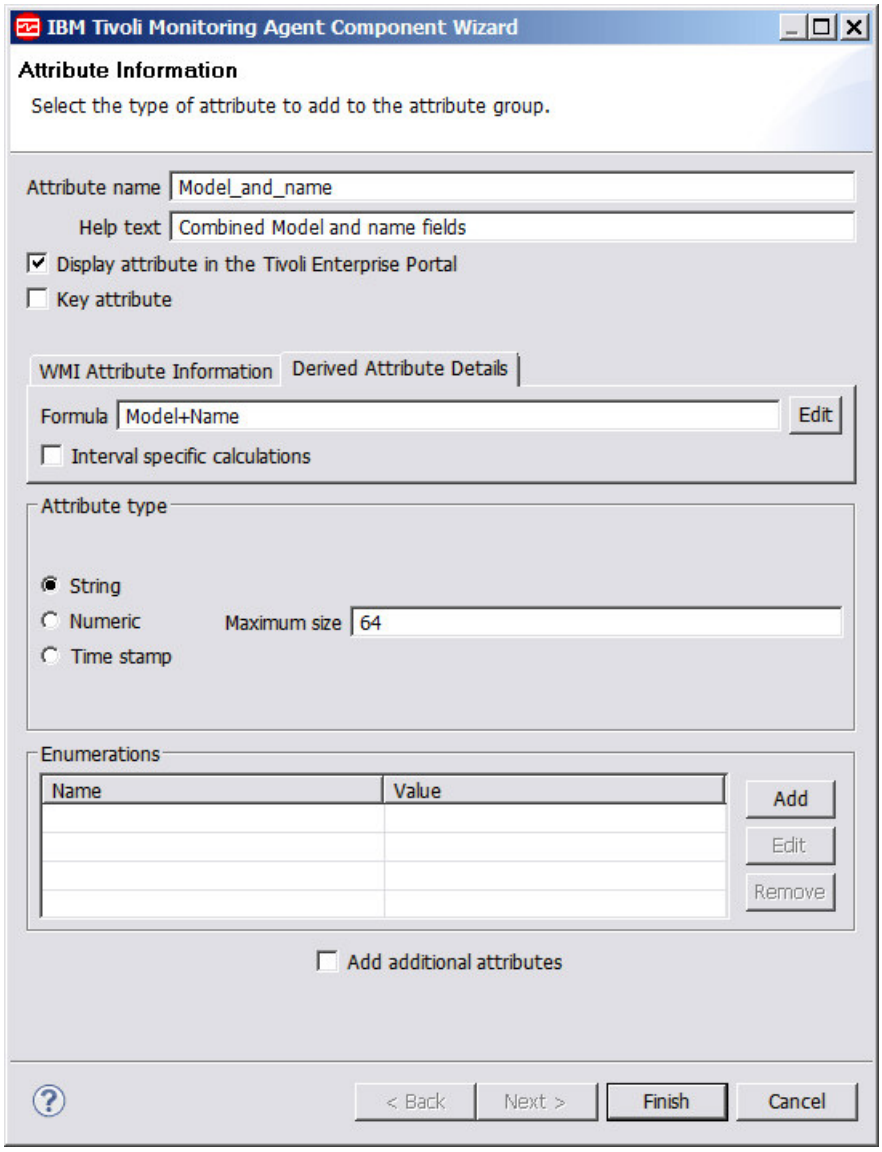


Figure 37. Attribute Information page with Derived Attribute Details tab open

4. In the **Formula** field, type the formula text or click **Edit** to enter the formula with a graphical editor. See "Formula operators and functions" on page 82 for information about the operators and functions that can be used in the formula.

   Note: When you click **Edit**, the Formula Editor opens. See "Editing derived attributes" on page 67 for information about editing derived attributes.
5. Optional: Select or clear the **Interval specific calculations** check box to determine which two attribute sample values are used when the function is calculated. Use this option when your formula uses the `rate` or `delta` functions. For more information about **Interval specific calculations**, see "**Interval specific calculations**" on page 66. For more information about `rate` and `delta` functions, see "Formula operators and functions" on page 82.
6. In the **Attribute type** area, click the type of attribute.
7. Click **OK**. The Data Source Definition page is displayed again with the data source listed in it as before.
8. Click **Finish**.

Interval specific calculations

You can choose **Interval specific calculations** when you define a derived attribute that is based on the `rate` or `delta` functions.

You select **Interval specific calculations** on the Derived Attribute Details tab of the Attribute Information page. For more information, see "Creating derived attributes" on page 63.

When you use the **Interval specific calculations** selection, it is important to understand the concept of a delta or difference between attribute values. The delta is the difference between the most recent value of the attribute and a previous value of the attribute. The delta is returned directly by the `delta` function and is used by the `rate` function to calculate a result.

The `delta` or `rate` function must always have the `last` function as its only argument. The `last` function specifies which values of an attribute are used to determine the delta. If **Interval specific calculations** is not selected, the previous value that is used is always the second-most-recent value. If **Interval specific calculations** is selected, the previous value that is used is the value whose age (relative to the most recent value) is equal to the collection interval of the requester.

For example, suppose `CDP_DP_REFRESH_INTERVAL` is set to 120 seconds and `attribute A` has the following sampled values:

Time	Sampled value
current	`2800`
2 minutes (120 seconds) ago	`2600`
4 minutes (240 seconds) ago	`2499`
6 minutes (360 seconds) ago	`1500`
8 minutes (480 seconds) ago	`1200`
10 minutes (600 seconds) ago	`1000`

When **Interval specific calculations** is not selected, the `delta` function always returns `200`, the difference between the two most recent values, `2800 - 2600`. The same value is returned whether the value is displayed on the Tivoli Enterprise Portal, used in a situation, or a historical collection.

When **Interval specific calculations** is selected, the `delta` function returns a value that depends on the collection interval of the requester.

If a derived attribute with the `delta` function is used in a situation with a 4-minute collection interval, the value that is returned by the `delta` function is `301`, the difference between the most recent value and the value obtained 4 minutes before that, `2800 - 2499`.

If a derived attribute with the `rate` function is used in a situation with a 10-minute (600-second) collection interval, the value that is returned by the `rate` function is `3`, the difference between the most recent value and the value obtained 10 minutes before that, divided by the number of seconds in the interval `(2800 - 1000) / 600`.

Note: The Tivoli Enterprise Portal has no inherent collection interval, so delta and rate calculations for Tivoli Enterprise Portal requests always use the most recent and second most recent attribute values, the same result whether **Interval specific calculations** is selected or not.

For `delta` or `rate` to work correctly with **Interval specific calculations**,

- The agent must collect data periodically in the background, and not on demand (`CDP_DP_THREAD_POOL_SIZE` must be greater than 0).

- Every situation or historical collection interval in which the attribute is used must be a multiple of the background refresh interval (`CDP_DP_REFRESH_INTERVAL`).
- The count (the second argument of the last function) must be large enough to accommodate the largest collection interval from a situation or historical collection. For example, if the agent must support 10-minute (600 second) historical collection and `CDP_DP_REFRESH_INTERVAL` is 120 seconds, the count must be at least `6, 1+(600 / 120)`. A count value of `6` ensures that the `last` function returns the newest sample and samples up to 600 seconds old.

Note: If these conditions are not met, input values are likely invalid and a result of `0` is returned.

Related tasks:

"Creating derived attributes" on page 63
You can create an attribute that derives its value from other attributes instead of directly from the data source.

Related reference:

"Environment variables" on page 435
Use environment variables to control the behavior of the agent at run time.

"Formula operators and functions" on page 82
A reference (including examples) of formula operators and functions that are used in the formula editor.

Editing derived attributes

Use the Formula Editor to edit derived attributes.

The Formula Editor is available from the **Derived Attribute Details** tab on the Attribute Information page as described in Step 3 on page 64 of ("Creating derived attributes" on page 63). For more information about the Formula Editor, see "Formula Editor" on page 68

Filtering attribute groups

You can create a filter to limit the data that is returned from an attribute group that returns sampled data.

Before you begin

If the attribute group exists, open the Data Source Definition page. For more information, see "The Data Source Definition page" on page 46.

If you want to create an attribute group, follow the steps in ("Defining initial data source" on page 23) and click **Advanced** in the initial data source information page.

Procedure

1. Use one of the following steps to begin creating the filter:
   - If you are creating an attribute group, click **Advanced** in the initial data source information page.
   - If the attribute group exists, select the attribute group in the Data Source Definition page and click **Advanced** in the Data Source Definition page.
2. In the Advanced Data Source Properties page, enter a selection formula. The selection formula that you enter must evaluate to a Boolean result, true, or false. In the Advanced Data Source Properties page, you can click **Edit** to enter or modify the formula by using the Formula Editor. For more information about the Formula Editor, see "Formula Editor" on page 68
3. When you finish entering the filter selection formula, click **OK** until you return to the Data Source Definition page. When the filter is created, the agent uses the filter to evaluate each row of data. When the filter evaluates to *true* for a row of data, the data is sent to Tivoli Monitoring. When the filter evaluates to *false*, the row of data is not sent and is discarded.

What to do next

You can validate that the filter is working as intended by using the test function for the attribute group. For more information about attribute group testing, see “Attribute group testing” on page 377

Formula Editor

The Formula Editor is a graphical editor that aids in the creation of formulae in the Agent Builder. The Formula Editor is displayed when you do one of the following tasks:

1. Creating or editing derived attributes, see “Creating derived attributes” on page 63 and “Editing derived attributes” on page 67
2. Creating Filtered Attribute groups, see “Creating a Filtered Attribute Group” on page 324
3. Filtering data from attribute groups, see “Filtering attribute groups” on page 67

Attention:

- When you create derived attributes, the formula that you create must result in a data type that matches the type of the attribute. For example, if the derived attribute type is a number, the formula you create must evaluate to a numeric result.
- When you create filtered attribute groups or filter data from attribute groups, the formula that you create must result in a Boolean value, "true" or "false".

Note: In the following views, the Formula Editor is shown creating formulae for derived attributes. The views are identical when you use the Formula Editor with filtered attribute groups or to filter data from attribute groups. The views show the heading Derived Formula Editor or Filter Formula Editor depending on use.

When the Formula Editor (Figure 38 on page 69) is displayed, the current formula is loaded into the editor. If a formula does not exist, you can enter one by typing directly into the formula space in the Formula Editor window. Alternatively you can click **Insert** to begin entering a formula by using the editor menu options. The editor contains two views of the formula in the default window, and an option for a third view:

Component view (default)
The components of the edited formula are shown in the operand areas and **Operator** field. In Figure 38 on page 69, the (+) operator is the current operator. The operator and its two operands can be edited by using the selection menus.

Formula view (default)
The complete formula is in the formula field in the window. You can edit the formula by typing in this box.

Formula hierarchy tree view (option)
The formula hierarchy tree is displayed by selecting the **Show formula hierarchy** check box. See Figure 39 on page 70. The state of the check box is remembered in subsequent invocations of the Formula Editor.

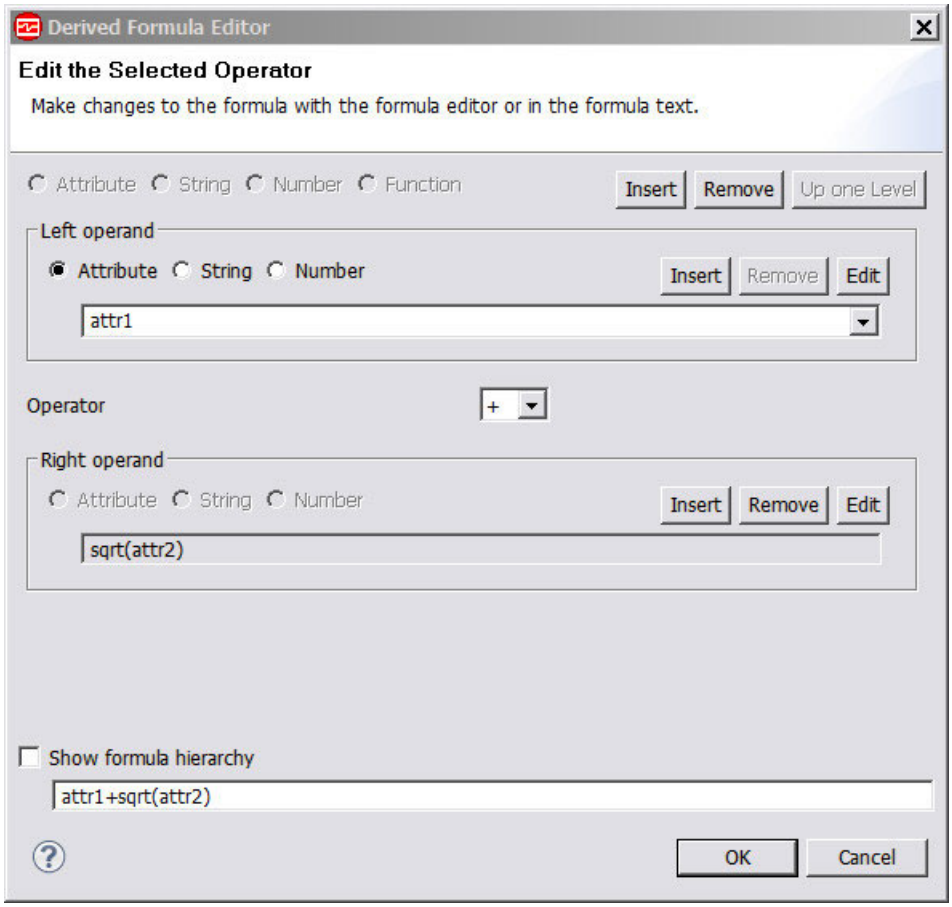


Figure 38. Derived Formula Editor window (default)

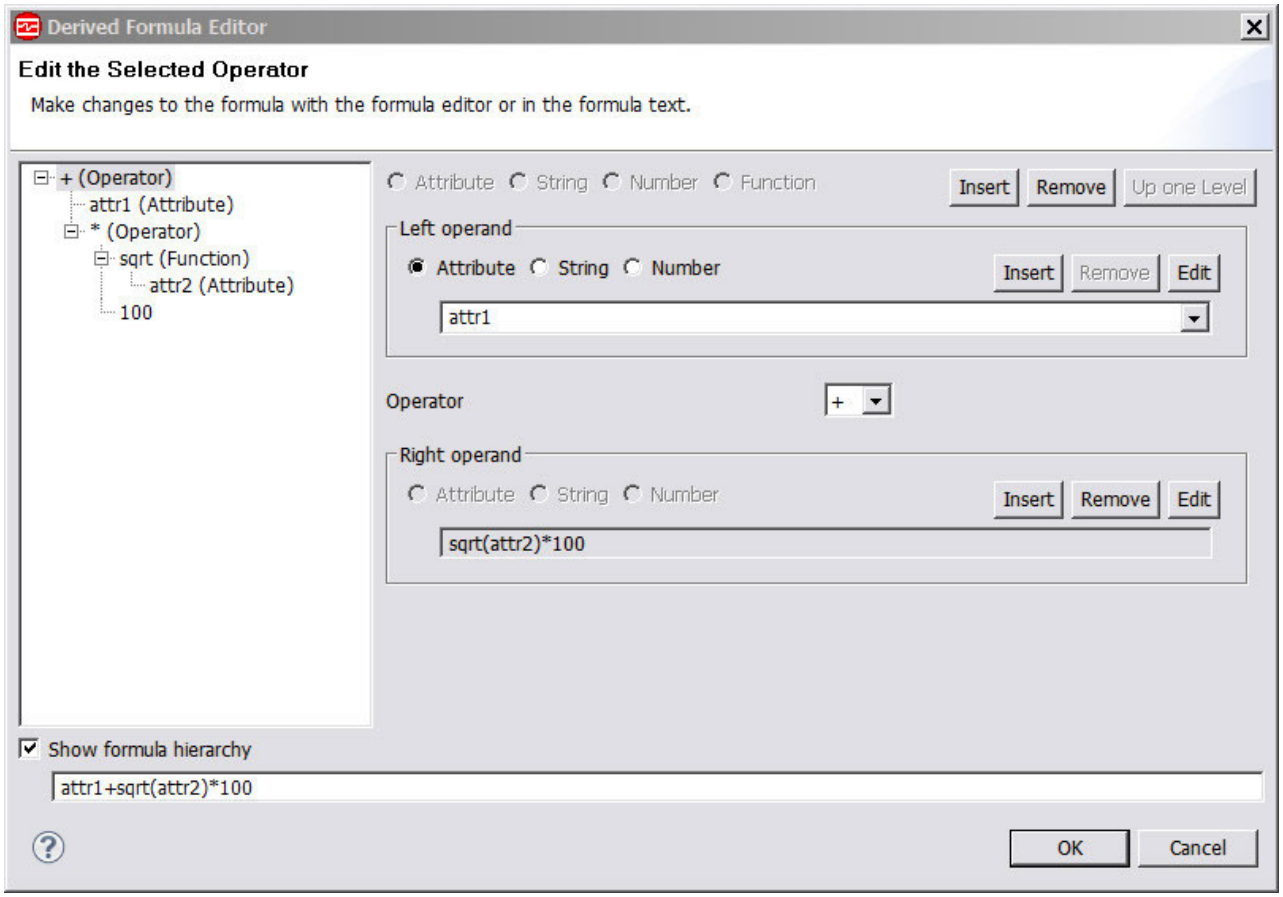


Figure 39. Derived Formula Editor view with formula hierarchy tree view

Changing the Formula Editor component view

Change the component view in the Formula Editor.

Before you begin

For details about the Formula Editor and its component view, see ("Formula Editor" on page 68).

About this task

The component that is shown in the component view can be changed in the following ways:

Procedure

- Move the cursor in the formula text.
- Select a different node in the formula hierarchy tree.
- Select **Up one Level** or one of the Edit buttons.

Component types

In the component view, the current component and any operands or function arguments of that component can be edited.

The following types of components appear differently in the Formula Editor when selected:

- Attribute component
- Literal components
- Operator component
- Conditional expression component
- Function component

Formula Editor Attribute component

Use the attribute component in the Formula Editor to select and manipulate attributes in formulae.

Before you begin

For details about the Formula Editor and its component view, see ("Formula Editor" on page 68).

About this task

You can select an attribute from a list of attributes for the attribute group in the component view of the Formula Editor.

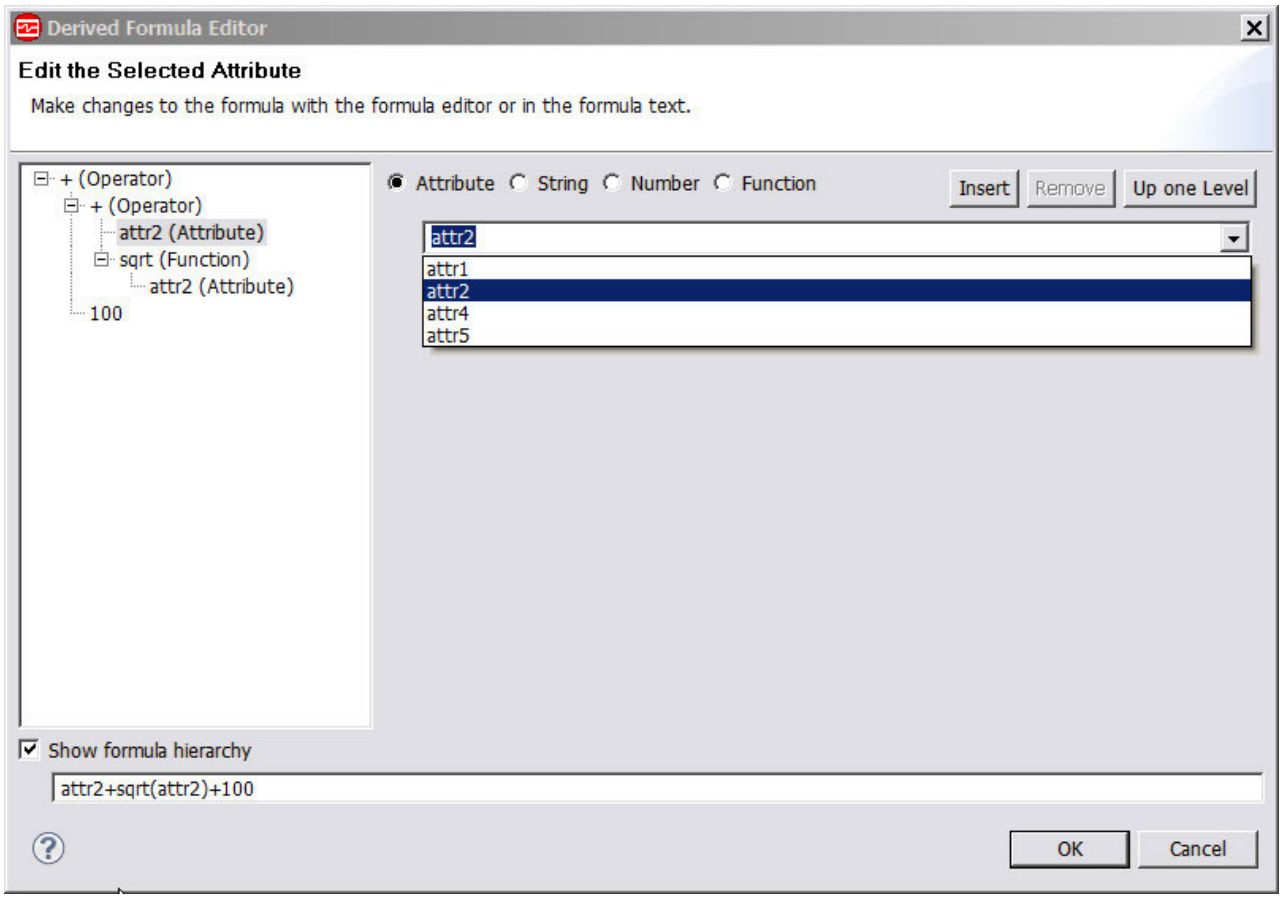


Figure 40. Edit the Selected Attribute window

Procedure

1. To work with a specific attribute, select that attribute from the list and click **Edit** The Edit the Selected Attribute window (Figure 40) is displayed.
2. You can manipulate the selected attribute in the following ways:

- You can replace the attribute with a string or number by selecting **String** or **Number**. The attribute list is replaced by an entry field and the contents are no longer compared to the list of valid attribute names.
- You can replace the attribute with a function by clicking **Function**. Parentheses are added after the name and the list now contains valid function names to choose from.
- You can type an attribute name instead of selecting one. Typing a name is useful if you did not yet define all of the attributes in this attribute group.
  - A warning is displayed if there is no attribute with the name that was entered.
  - An error is displayed if characters are entered that cannot be part of an attribute name, as shown in (Figure 41).
    OK is disabled until the error is corrected.

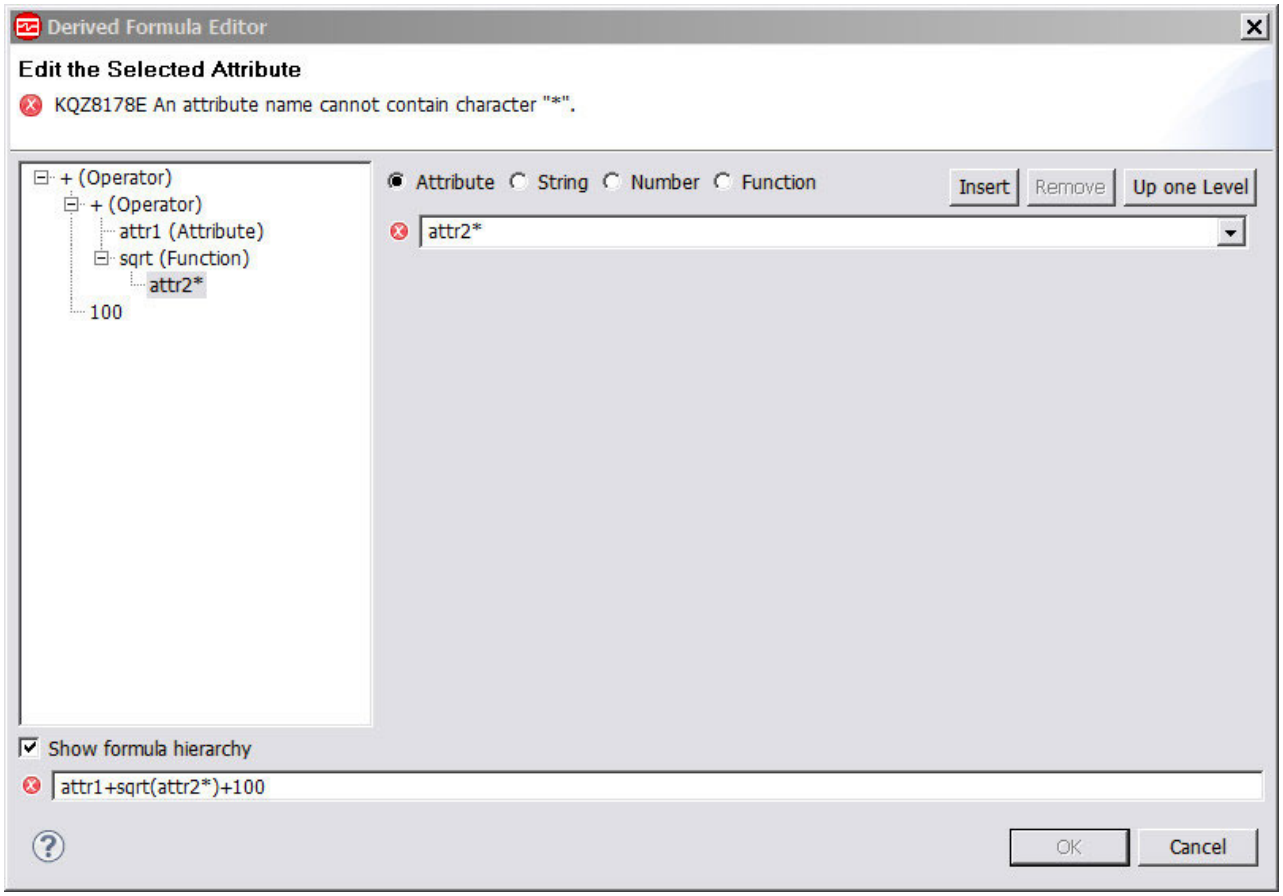


Figure 41. Edit the Selected Attribute window with error

- Attributes are not filtered based on type. If an attribute (or any value) of the wrong type is selected or entered, a warning message like the one shown in Figure 42 on page 73 is displayed.

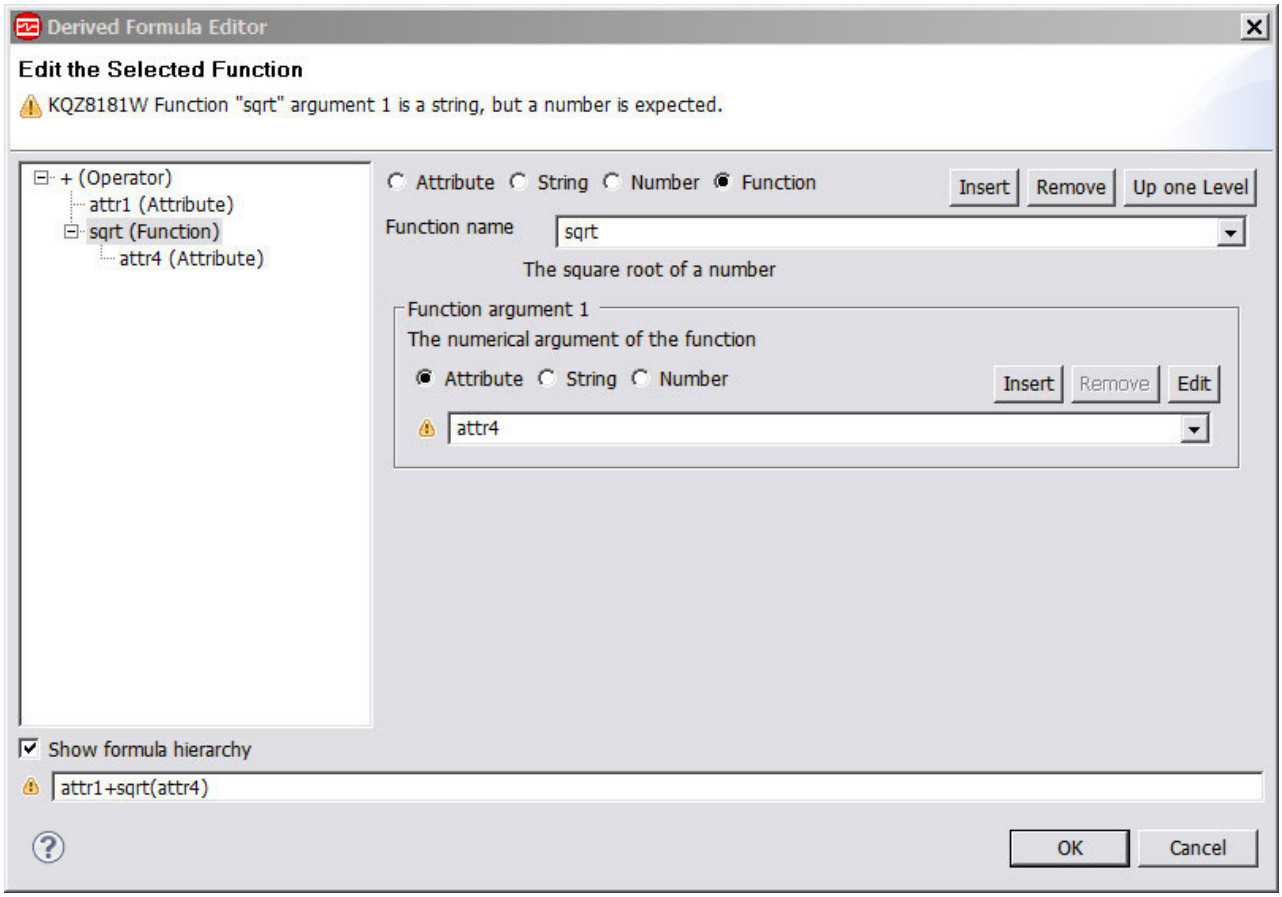


Figure 42. Edit the Selected Function page with warning

What to do next

See "Formula Editor common options" on page 78 for information about using the following options: **Insert**, **Remove**, and **Up one Level**.

Formula Editor Literal components

Use the string and number components in the Formula Editor to manipulate literals in formulae.

Before you begin

For details about the Formula Editor and its component view, see ("Formula Editor" on page 68).

About this task

A literal is any value that is entered directly in the formula that does not come from an attribute value or from a function. A literal value can be either a string or a number.

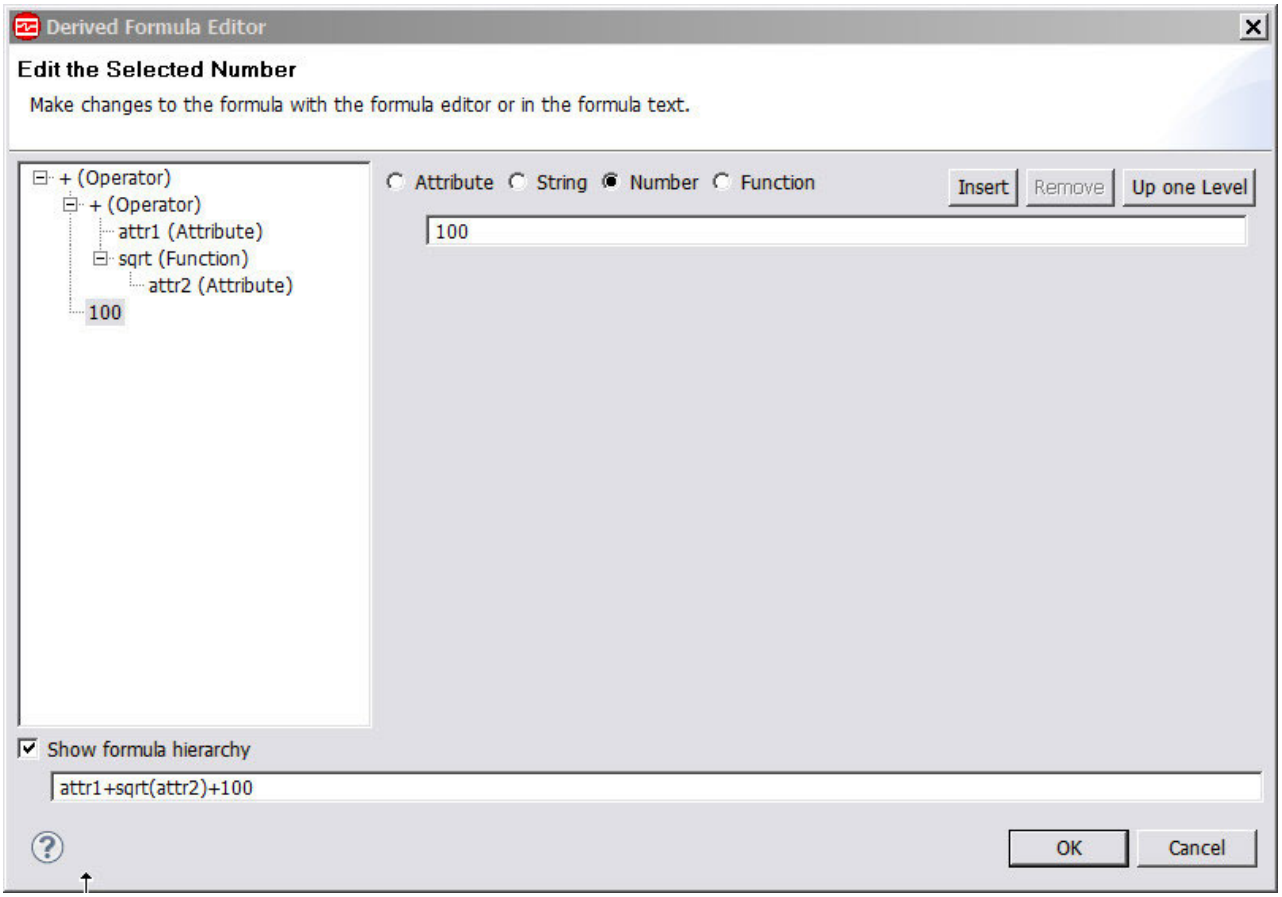


Figure 43. Edit the Selected Number page with a number selected.

Procedure

- You can replace a literal string or number with an attribute by clicking **Attribute**. A valid attribute name must be selected or entered without quotation marks.
- You can replace a literal string or number with a function by clicking **Function**. Parentheses are added after the name and the selection list contains valid function names to choose from.
  - A warning is displayed if a number is entered where a string is expected or vice versa.
  - If **Number** is selected, an error is displayed if the field is not a number. **OK** is disabled until the error is corrected.

What to do next

See "Formula Editor common options" on page 78 for information about using the following options: **Insert**, **Remove**, and **Up one Level**.

Formula Editor Operator component

Use the operator component in the Formula Editor manipulate operators in formulae.

Before you begin

For details about the Formula Editor and its component view, see ("Formula Editor" on page 68).

About this task

An operator component shows an operator and its operands.

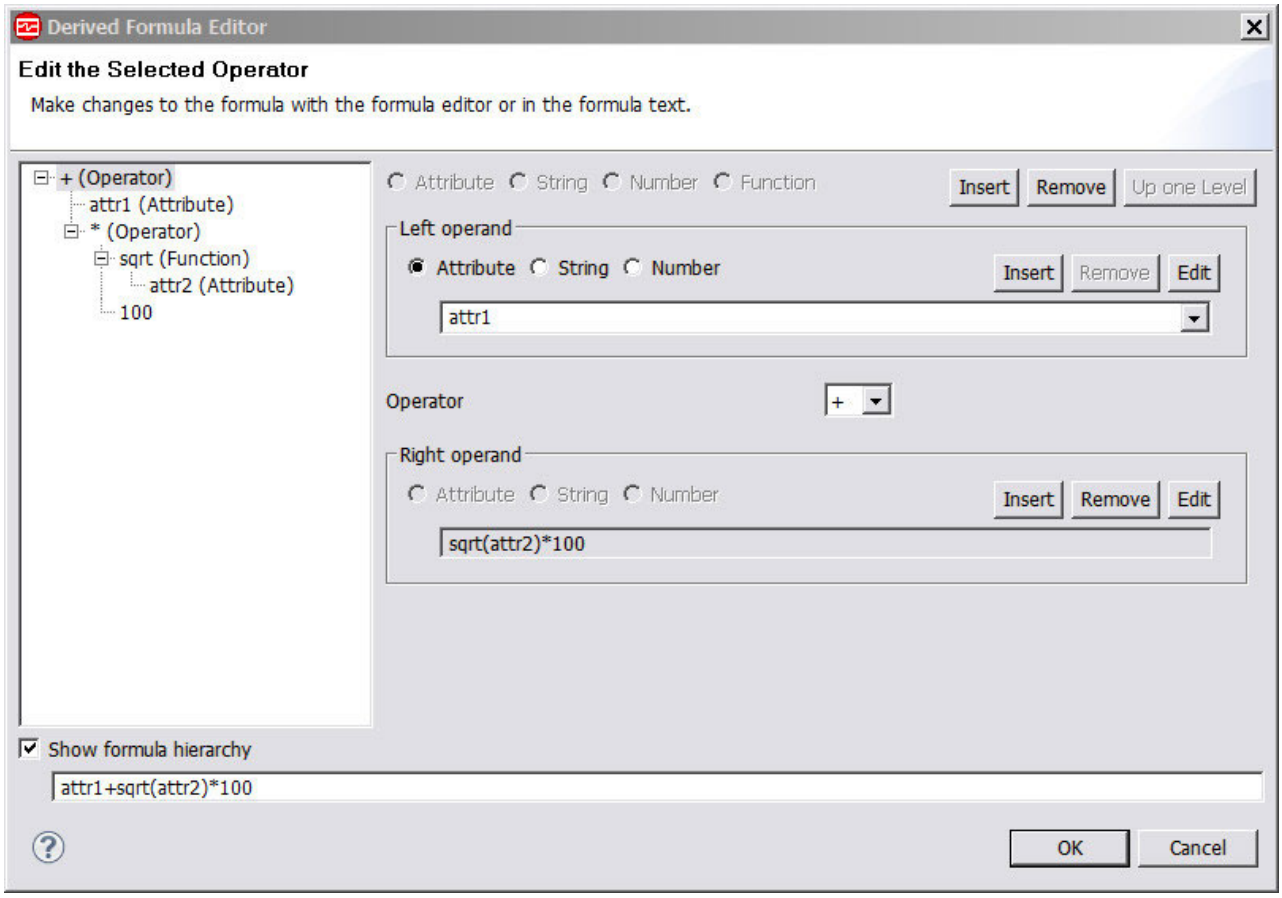


Figure 44. Edit the Selected Operator page

Procedure

- In the Formula Editor component view select the operator from the **Operator** list, between the two operands. The (%) operator multiplies the first operand by 100, and then divides by the second operand.
- Select the operator (`+ - * / or %`).
  - The **Left operand** section of the page is before the operator.
  - The **Right operand** section is after the operator.
  - Simple operands (attributes and literals) can be edited without having to change the selected component to the operand as described in ("Formula Editor Attribute component" on page 71) and ("Formula Editor Literal components" on page 73).
  - Complex operands, which consist of other operators or functions, can be edited by clicking **Edit**. This action highlights the operand component instead of the entire operator.

What to do next

See "Formula Editor common options" on page 78 for information about using the following options: **Insert**, **Remove**, **Up one Level**, and **Edit**.

Formula Editor Conditional expression component

The conditional expression component shows a condition, a value to return if the condition is true, and a value to return if the condition is false.

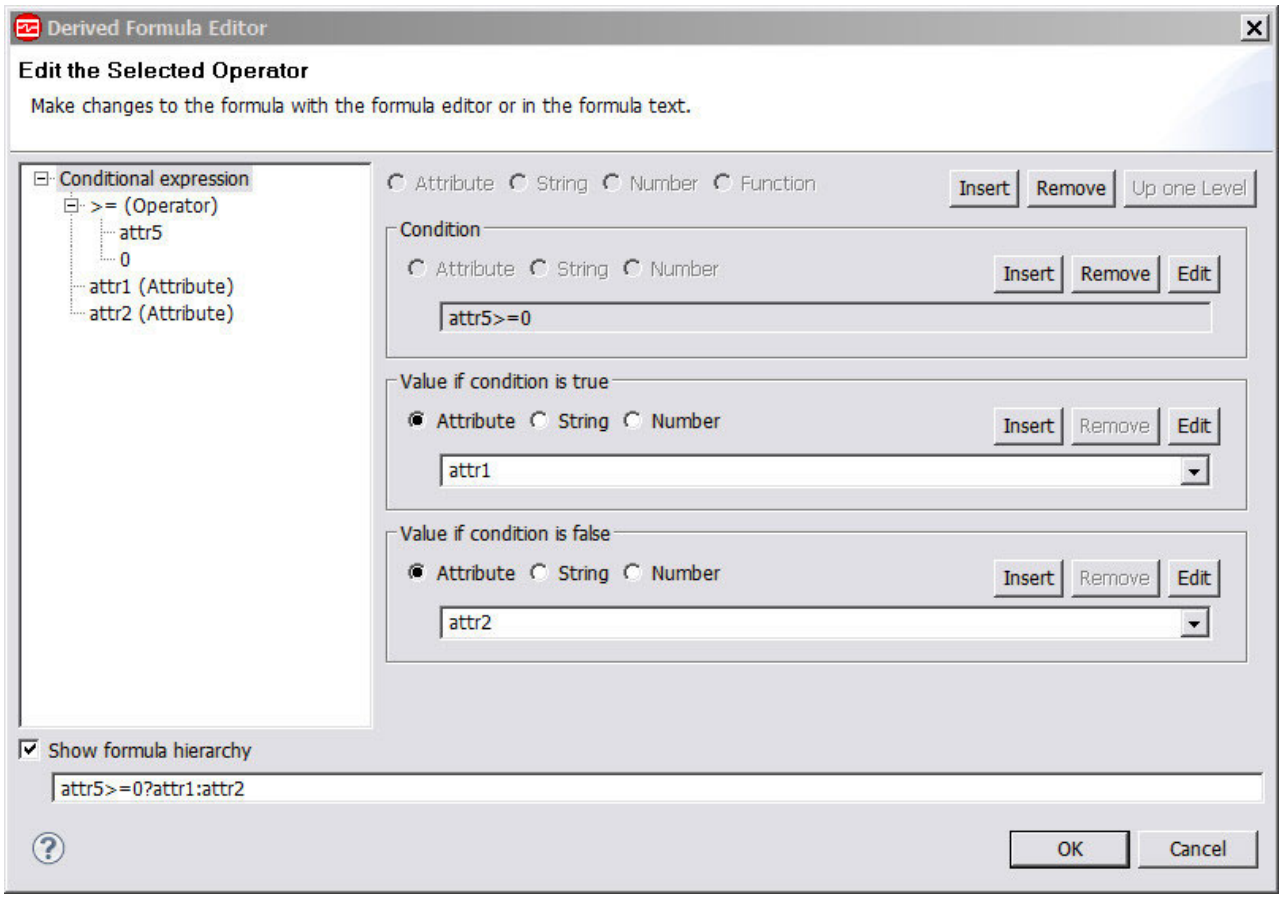


Figure 45. Edit the Selected Operator page (for a conditional operator)

- The expression in the Condition section must evaluate to true or false. Operators `(==)`, `(!=)`, `(<)`, `(<=)`, `(>)`, `(>=)`, `(&&)`, `(||)`, `(!)` are available to form expressions that return true or false.
- Simple operands (attributes and literals) can be edited without having to change the selected component to the operand as described in "Formula Editor Attribute component" on page 71 and "Formula Editor Literal components" on page 73.
- Complex operands, which consist of other operators or functions, can be edited by clicking Edit. This action highlights the operand component instead of the entire conditional expression.
- See "Formula Editor common options" on page 78 for information about using the following options: **Insert**, **Remove**, **Up one Level**, and **Edit**.

Related concepts:

"Formula Editor" on page 68

Formula Editor Function component

Use the function component in the Formula Editor to select and manipulate function components in formulae.

Before you begin

For details about the Formula Editor and its component view, see ("Formula Editor" on page 68).

About this task

The function component shows the function and its arguments.

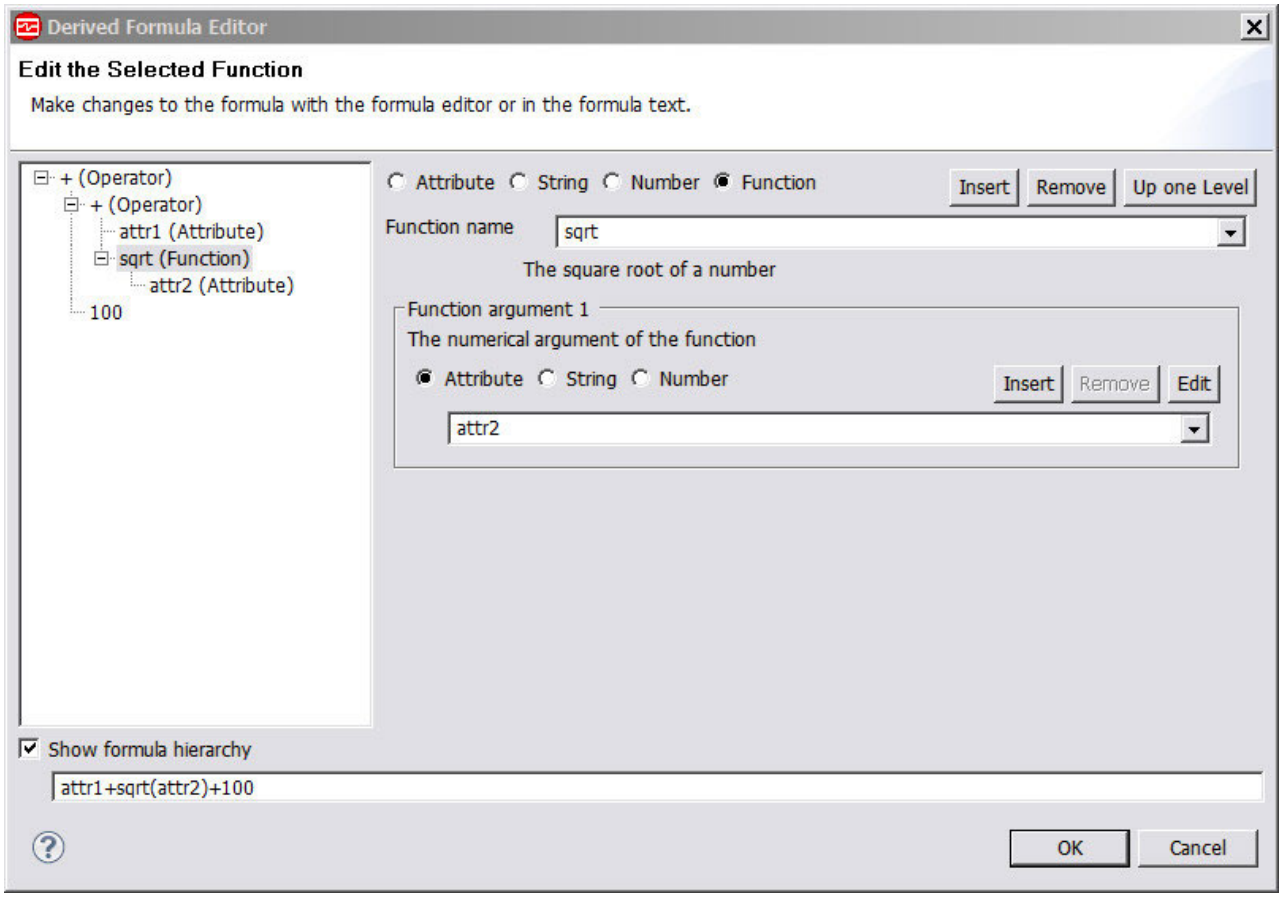


Figure 46. Edit the Selected Function page

Procedure

- To work with the functions Select the **Function name** from the list in the Formula Editor.
  - The description of the selected function is shown after the function.
  - **Function argument** sections are shown after the function name. The appropriate number of arguments for the selected function are shown. A description specific to the function selected is shown.
  - Simple arguments (attributes and literals) can be edited without having to change the selected component to the operand as described in "Formula Editor Attribute component" on page 71 and ("Formula Editor Literal components" on page 73).
  - Complex arguments, which consist of operators or other functions, can be edited by clicking **Edit**. This action highlights the argument component instead of the entire function.
- For functions that take a variable number of arguments, add arguments by clicking **Insert** or remove arguments by clicking **Remove** in addition to the actions described in ("Formula Editor common options" on page 78).
- For the `getenv` function, a configuration property can be chosen by clicking **Insert** as shown in (Figure 48 on page 79). If you select the Configuration property choice, the Configuration Properties window is displayed.

What to do next

See “Formula Editor common options” for information about using the following options: **Insert**, **Remove**, **Up one Level** and **Edit**.

Formula Editor common options

Options that are common to the views in the Formula Editor

The Formula Editor common options are:

- **Insert**
- **Remove**
- **Up one Level**
- **Edit**

Insert

Insert inserts an operator or a function before the component. The component is demoted to one of the operator operands or one of the function arguments. For example, if you click **Insert** before the `sqrt` function, you are asked what you want to insert and the following choices are displayed (Figure 47):

- **An operator with sqrt(attr2) as one of the operator’s operands**
- **A function with sqrt(attr2) as the function's first argument**
- **A conditional expression with sqrt(attr2) as the true or false values**

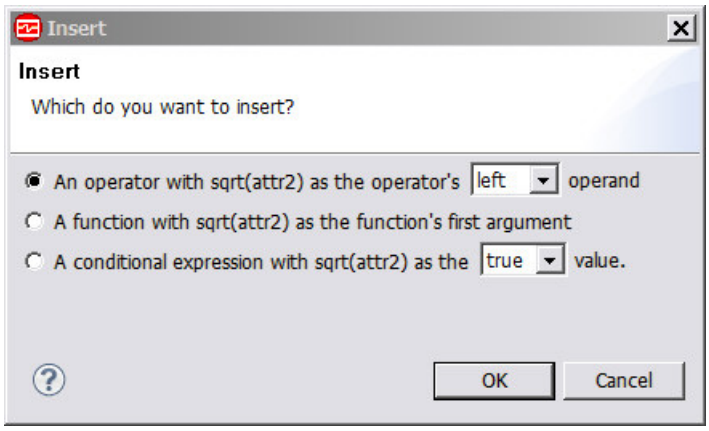


Figure 47. Insert window example

If you click **Insert** before the `getenv` function, you are asked what you want to insert and the following choices are displayed (Figure 48 on page 79):

- **Configuration property**
- **An operator with attr2 as one of the operator’s operands**
- **A function with attr2 as the function's first argument**
- **A conditional expression attr2 as the true or false values**

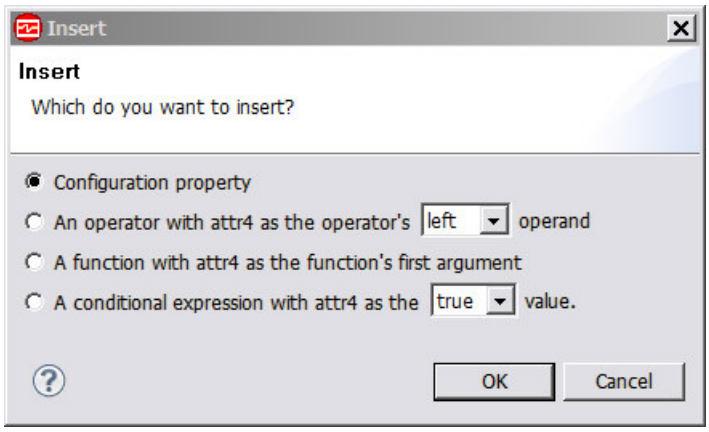


Figure 48. Insert window example for `getenv` function

Remove

Remove is available only for operators and functions, and is the inverse of **Insert**. When you click **Remove**, you are asked what is to replace the removed operator or function. For example, **Remove** before the `sqrt` function shows the following choices (Figure 49):

- **An attribute or literal value**
- **The current argument 1, attr2**

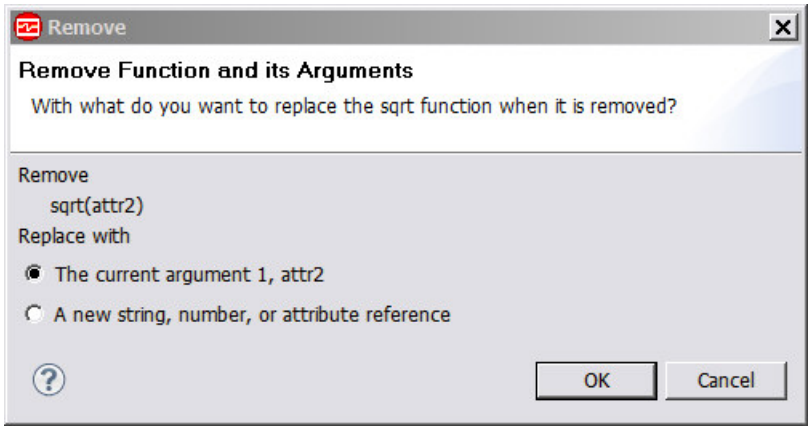


Figure 49. Remove function and its Arguments page

Click **An attribute or literal value** to discard the entire tree after the point that is being removed and replace it with a new attribute or literal value.

Click the second choice to promote the selected operand or argument to replace the removed operator or function. You can click subsequent choices if there are more arguments or operands. Any other operands or arguments are discarded.

Up one Level

Click **Up one Level** to move up in the tree.

Edit

Click **Edit**, before a complex operand or argument, to make it the component to be edited.

Click **Up One Level** after you click **Edit** to restore the current component to what it was before you clicked **Edit**.

Formula Editor - Formula errors

Correcting formula errors in the Formula Editor

The component view is different when there is no formula or the entered formula cannot be parsed. For example:

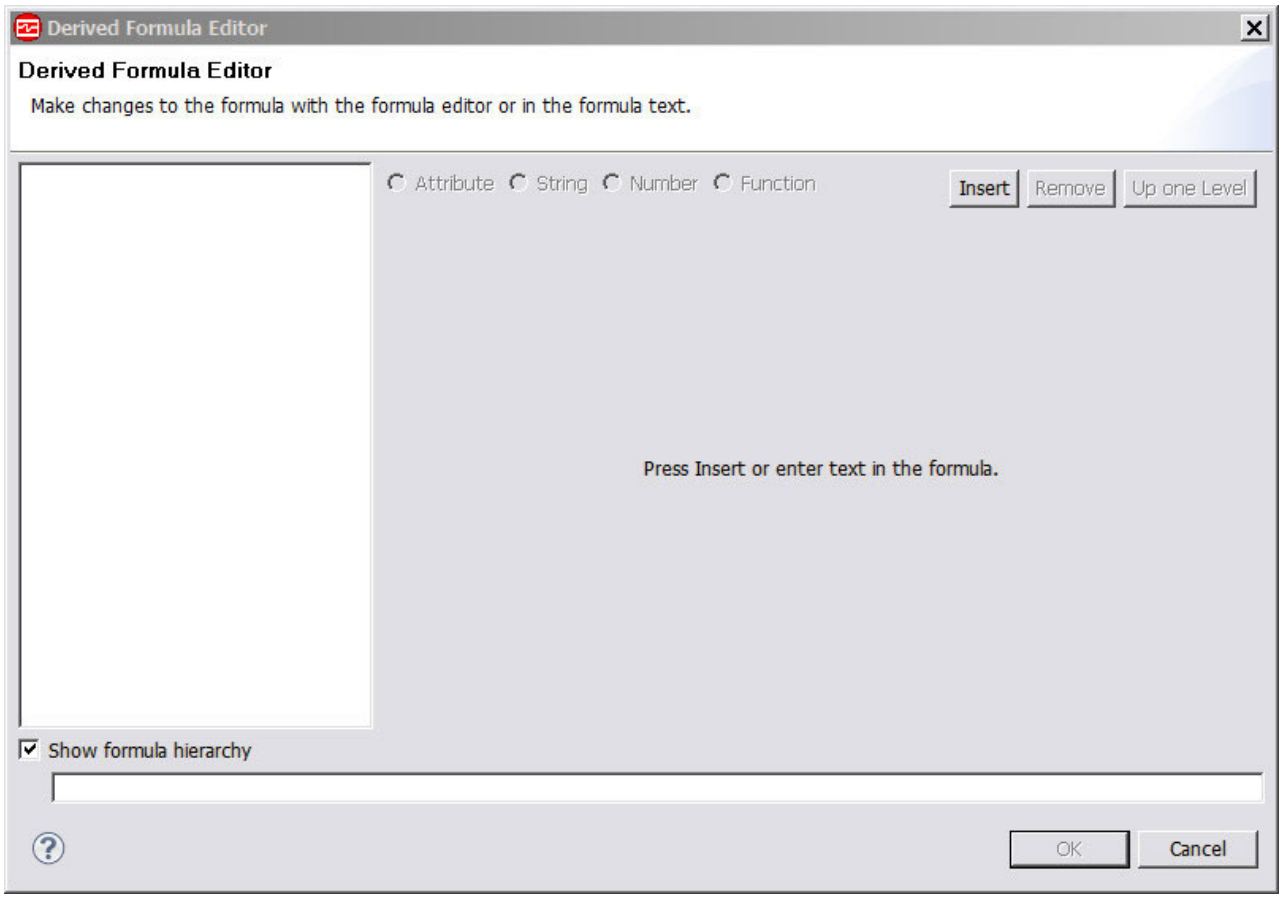


Figure 50. Empty formula example

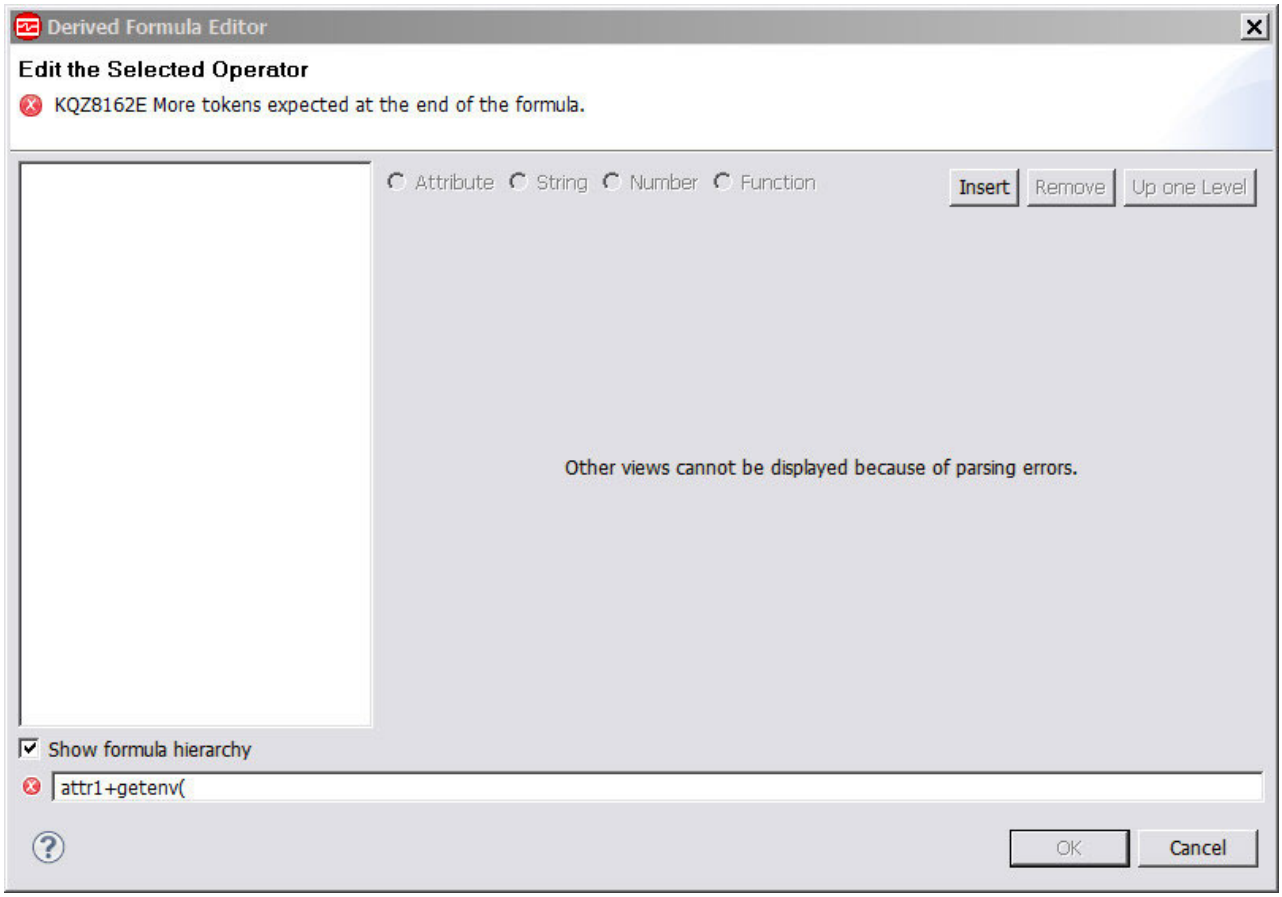


Figure 51. Parsing error example

You can correct a formula with parsing errors by typing directly in the formula field, or by replacing it with a new formula by clicking **Insert**. In this case, **Insert** presents the following choices (Figure 52 on page 82):

- **An attribute**
- **A string**
- **A number**
- **An operator**
- **A conditional expression**
- **A function**



Figure 52. Insert window for fixing a formula manually

Related concepts:
"Formula Editor" on page 68

Formula operators and functions

A reference (including examples) of formula operators and functions that are used in the formula editor.

A derived attribute value is the result of evaluating an expression that is based on constants and other attribute values in the same data source. The expression grammar is the normal mathematical expression - operand operator operand with parentheses used for grouping. Numeric attributes can be combined with other numeric attributes or constants by using the normal mathematical operators: `+ - * /, and %`, which multiplies the **Left operand** by 100 and divides by the **Right operand**. String attributes can be combined with other string attributes or constants with `+`. You can also use the following described functions. Functions are entered in the format: `function_name(argument_1, argument_2, argument_3)`.

An attribute is represented by its name (the same name you see in the Data Sources Information tree). Integer constants are specified as numbers. String constants are surrounded by quotation marks.

You can use the following functions in a formula:

abs Returns the absolute value of a number

atof Converts a string to a floating point value

atoi Converts a string to an integer value. It operates in the same way the normal **C atoi** works: it stops at the first non-decimal character.

average
Returns a single value that is the average of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the average of the most recent values of an attribute).

Examples of this function in use are:

```
average (Attr_A, AttrB, Attr_C)
average (last (Attr_A, 10))
```

ceiling
Returns the least integer that is not less than the argument.

For example, where `attribute_a = 12.4`, `ceiling(attribute_a)` returns the value 13. And, where `attribute_a = -12.4`, `ceiling(attribute_a)` returns the value -12.

delta The difference between the most recent value of an attribute and a previously collected value of that attribute. The single argument to delta must be the `last` function, which obtains the current and previous values of an attribute. A normal use might look like:

```
delta (last(OtherAttribute, 2))
```

For more information about which attribute values from the last function are used to calculate the delta, see "**Interval specific calculations**" on page 66. This function is applicable only for derived attributes, not for attribute group filters.

floor Returns the greatest integer that is not greater than the argument.

For example, where `attribute_a = 12.4`, `floor(attribute_a)` returns the value 12. And, where `attribute_a = -12.4`, `floor(attribute_a)` returns the value -13.

getenv
Returns the value of the provided environment or "configuration variable".

ipAddressToName
Converts an IP address to a host name. This function requires one argument, an IP address string in dotted decimal notation. If the address cannot be resolved, then the IP address is returned.

itoa Converts an integer into a string. This function is most useful when you want to concatenate a numeric value onto a string. The derived string + function takes only two string arguments.

last Returns a list of values for use by the `min`, `max`, `average`, `stddev`, `rate` and `delta` functions. It takes two arguments: the attribute to collect and the number of values to use in the calculation. If the required attribute is an integral value in a string attribute, the first argument can contain the `atoi` function, such as `atoi(numericalStringAttribute)`. The second argument must be a number. It can either be hardcoded as a constant or it can be the result of an `atoi(getenv("ENV_VAR"))` expression. It cannot reference an attribute value.

Examples of this function in use are:

```
average (last (Attr_A, 10))
last (Attribute_A, ${K01_NUM_COLLECTIONS}))
```

matches
Returns a Boolean, true, or false, indicating whether a regular expression matches a value. It takes two arguments, string source and a regular expression whose result the string is compared to. This function is useful for filtering attribute groups.

max Returns a single value that is the maximum of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the maximum of the most recent values of an attribute).

min Returns a single value that is the minimum of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the minimum of the most recent values of an attribute).

NetWareTimeToTivoliTimestamp
Converts a Novell NetWare hexadecimal time value to a Tivoli Monitoring time stamp. This function requires one argument, a special NetWare hexadecimal time value. The attribute type is timestamp.

rate The rate of change (per second) between the most recent value of an attribute and a previously collected value of that attribute. The single argument to rate must be the `last` function, which obtains the current and previous values of an attribute. A normal use might look like:

```
rate (last(OtherAttribute, 2))
```

For more information about which attribute values from the last function are used to calculate the rate, see “**Interval specific calculations**” on page 66. This function is applicable only for derived attributes, not for attribute group filters.

replaceFirst
Replaces the first occurrence of a substring that matches a regular expression with a replacement string. This function takes three arguments. First: the input string. Second: the regular expression which is used to match a substring in the input string. Third: the replacement string. See (Appendix G, “ICU regular expressions,” on page 627) for details on the regular expressions and substitution values that are allowed in the replacement string.

replaceAll
Replaces all occurrences of substrings that match a regular expression with a replacement string. This function takes three arguments. First: the input string. Second: the regular expression which is used to match a substring in the input string. Third: the replacement string. See (Appendix G, “ICU regular expressions,” on page 627) for details on the regular expressions and substitution values that are allowed in the replacement string.

round Mathematically Rounds the number to the nearest whole number.

sqrt Returns the square-root of a number

stddev
Returns a single value that is the standard deviation of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the standard deviation of the most recent values of an attribute).

StringToTivoliTimestamp
Converts a date and time string to a Tivoli Monitoring time stamp. This function requires two arguments. The first argument is a free-form string representation of the time stamp. The second argument is a format string that identifies how to parse the free-form string representation of a time stamp. (Table 9) describes the valid format parameters. The attribute type is timestamp.

Table 9. Valid format parameters for StringToTivoliTimestamp

Symbol	Meaning	Format	Example
`y`	Year	`yy` `yyyy`	*96* *1996*

`M`	Month
	Note: Only English month strings are supported.
		`M` or `MM`
		`MMM`
		`MMMM`
			09
			Sept
			September

`d`	`day`	`d` `dd`	*2* *02*

`E`	Day of week
	Note: Only English day-of-week strings are supported.
		`EE`
		`EEE`
		`EEEE`
			Sa
			Sat
			Saturday

Table 9. Valid format parameters for StringToTivoliTimestamp (continued)

Symbol	Meaning	Format	Example
h	Hour in AM or PM (1-12)	hh	*07*
H	Hour in day (0-23)	HH	*00*
m	Minute in hour	mm	*04*
s	Second in minute	ss	*05*
S	Millisecond	S SS SSS	*2* *24* *245*
a	AM or PM marker	a or aa	*am*
Any other ASCII character	skip this character	- (hyphen) (space) / (forward slash) : (colon) * (asterisk) , (comma)	

(Table 10) provides examples of string representations of time stamps and the format strings that are used to parse them.

Table 10. StringToTivoliTimestamp examples

String representation of the time stamp	Format string
96.07.10 at 15:08:56	yy.MM.dd ** HH:mm:ss
Wed, August 10, 2010 12:08 pm	EEE, MMMM dd, yyyy hh:mm a
Thu 21/01/2010 14:10:33.17	EEE dd/MM/yyyy HH:mm:ss.SS

sum Returns a single value that is the sum of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the sum of the most recent values of an attribute).

TivoliLogTimeToTivoliTimestamp
Converts a Tivoli log file time stamp to a Tivoli Monitoring time stamp. This function requires one argument, the string time stamp from a Tivoli log file. The attribute type is timestamp.

tokenize
One token of a tokenized string. This function requires three arguments. The first argument is a string to be split into tokens. The second argument gives one or more characters in the string that separate one token from another. Any occurrence of any of the characters from this argument is used to identify and separate tokens in the first argument. The third argument is the index of the token to return as a result of this function. The first token is index 0, the second token is index 1, and so on. This argument can also be the string LAST to return the last token.

UTCtoGMT
Converts Coordinated Universal Time to a GMT Tivoli Monitoring time stamp. This function requires one argument, the integer time_t value. The attribute type is timestamp.

UTCtoLocalTime
Converts Coordinated Universal Time to a local Tivoli Monitoring time stamp. This function requires one argument, the integer time_t value. The attribute type is timestamp.

The following functions take no arguments and return a number.

count Keeps a counter that starts at 1 the first time it is called, and increments by 1 each subsequent time it is called. If you use it in an expression that also uses `last`, it matches the number of elements that are stored by `last()`, but only until `last()` reaches its maximum. At that point, `last()` starts deleting the oldest value for each new one, thus staying at the same number of total values, while `count()` keeps increasing forever.

cumulativeSum
Returns the sum of argument values of duplicate events that are represented by a flow control summary event. Or returns the argument if it is a single event from a data source. It takes a single numeric argument. This function applies only to event attribute groups with event filtering and summarization turned on.

eventThreshold
Returns the threshold value that is configured for the attribute group which generated the event. A number, with three enumerations:

- `SEND_ALL (-3)`
- `SEND_FIRST (-2)`
- `SEND_NONE (-1)`

The number in parentheses is the raw value. However, the Agent Builder defines the enumerations so by default the text version is visible on the Tivoli Enterprise Portal. If you specify an actual numeric threshold and not one of the three pre-defined choices, that number is returned by this function. The value is an integer > 0. This function applies only to event attribute groups with event filtering and summarization turned on.

isSummaryEvent
Returns 0 if it is a single event from a data source, or 1 if the event is a flow control summary event. The displayed values are Event and Summary Event if you use the default attribute for the function. If you create the attribute manually, the displayed values are 0 and 1, unless you define the names as enumerations. This function applies only to event attribute groups with event filtering and summarization turned on.

occurrenceCount
The number of matching events that are represented by a flow control summary event, or 1 if it is a single event from a data source. (A flow control summary event includes the first event). This function applies only to event attribute groups with event filtering and summarization turned on.

summaryInterval
Returns the summary interval that is configured for the attribute group which generated the event, in seconds. This function applies only to event attribute groups with event filtering and summarization turned on.

Examples

Examples of the use of formula operators and functions to created derived and filtered attributes

Example 1 - Derived Attributes

If you have a data source that defines the following attribute type:

Name	String
xBytes	Numeric

Name	String
yBytes	Numeric
Virtual_Size	Numeric

You can define:

- An attribute totalBytes to be the sum of xBytes and yBytes. You enter the formula xBytes + yBytes.
- An attribute yPercent to be a percentage of the total bytes, which is yBytes, can be defined as yBytes % (xBytes + yBytes) or yBytes % totalBytes.

Example 2 - Derived Attributes

This formula returns the maximum of the recently collected values for the Virtual_Sizeattribute. The number of samples that are collected is the value of the configuration variable, *K4P_COLLECTIONS_PER_HISTORY_INTERVAL* (accessed through getenv), converted to a number (through atoi):

```
max(last(Virtual_Size,atoi(getenv("K4P_COLLECTIONS_PER_HISTORY_INTERVAL"))))
```

Example 3 - Derived Attributes

This formula returns the square-root of the sum of the squares of the xBytes and yBytes attribute values:

```
sqrt(xBytes * xBytes + yBtyes * yBytes)
```

Example 4 - Derived Attributes

This formula returns the average of the xBytes attribute from the 20 most recent samples of the attribute group. If fewer than 20 samples are collected since the agent was started, it returns the average of the xBytes attribute from all samples:

```
average(last(xBytes,20))
```

Example 5 - Filtered Attributes

You have a data source that returns:

```
Name    Type  Size  Used  Free
Memory  MEM      8     4     4
Disk1   DISK   300   200   100
Disk2   DISK   500   100   400
```

You are only interested in the disk usage. The solution is to create a filter to limit the data that is returned. To limit the returned data, you create a simple filter that returns a Boolean, true, or false value, as follows

Disk Filter:

```
Type=="DISK"
```

Now when the filter Type=="DISK" is true, the attribute group returns only disk usage data, for example:

```
Name    Type   Size  Used  Free
Disk1   DISK   300   200   100
Disk2   DISK   500   100   400
```

Example 6 - Filtered Attributes

You have a data source that returns:

```
Name   Size  Used  Free
Memory    8     4     4
Disk1   300   200   100
Disk2   500   100   400
```

The data that is returned is similar to the previous example, however, there is not a `Type` attribute present this time. Here you can use the matches function to find any data rows with a name attribute value that matches `"Disk"` followed by a number.

Disk Filter:

```
matches(Name, "Disk[0-9]*")
```

Now when the filter matches the string `"Disk"` followed by a number in attribute `Name`, only the disk usage data rows are returned:

```
Name   Size  Used  Free
Disk1   300   200   100
Disk2   500   100   400
```

Specifying operating systems

When you define data sources that are not specific to the Windows operating system, you must specify the operating systems where the data source runs.

About this task

By default, the data source provides data on all of the operating systems that are defined at the agent level, see (“Operating system requirements” on page 13) and (“Default operating systems” on page 37). You can change the operating systems for each data source.

Procedure

1. To open the Operating Systems section, click **Operating Systems** in the data source Information page when you add a data source.
2. Select the operating systems on which the data source is to operate. Select individual operating systems, all operating systems, all operating systems of a specific type, or the agent default operating systems, as shown in the example (Figure 53).

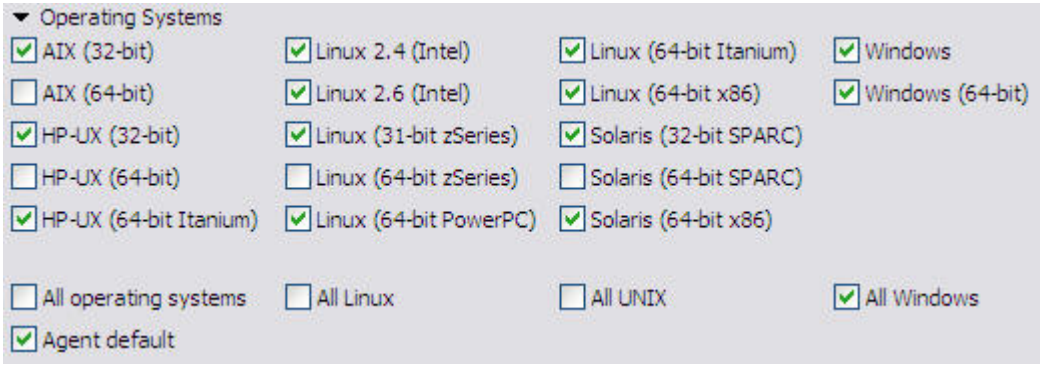


Figure 53. Operating systems where the agent is to run

Chapter 8. Monitoring a process

Define a data source to monitor a server process.

Procedure

1. On the Agent Initial Data Source page (Figure 54) or the Data Source Location page, click **A process** in the **Monitoring Data Categories** area.

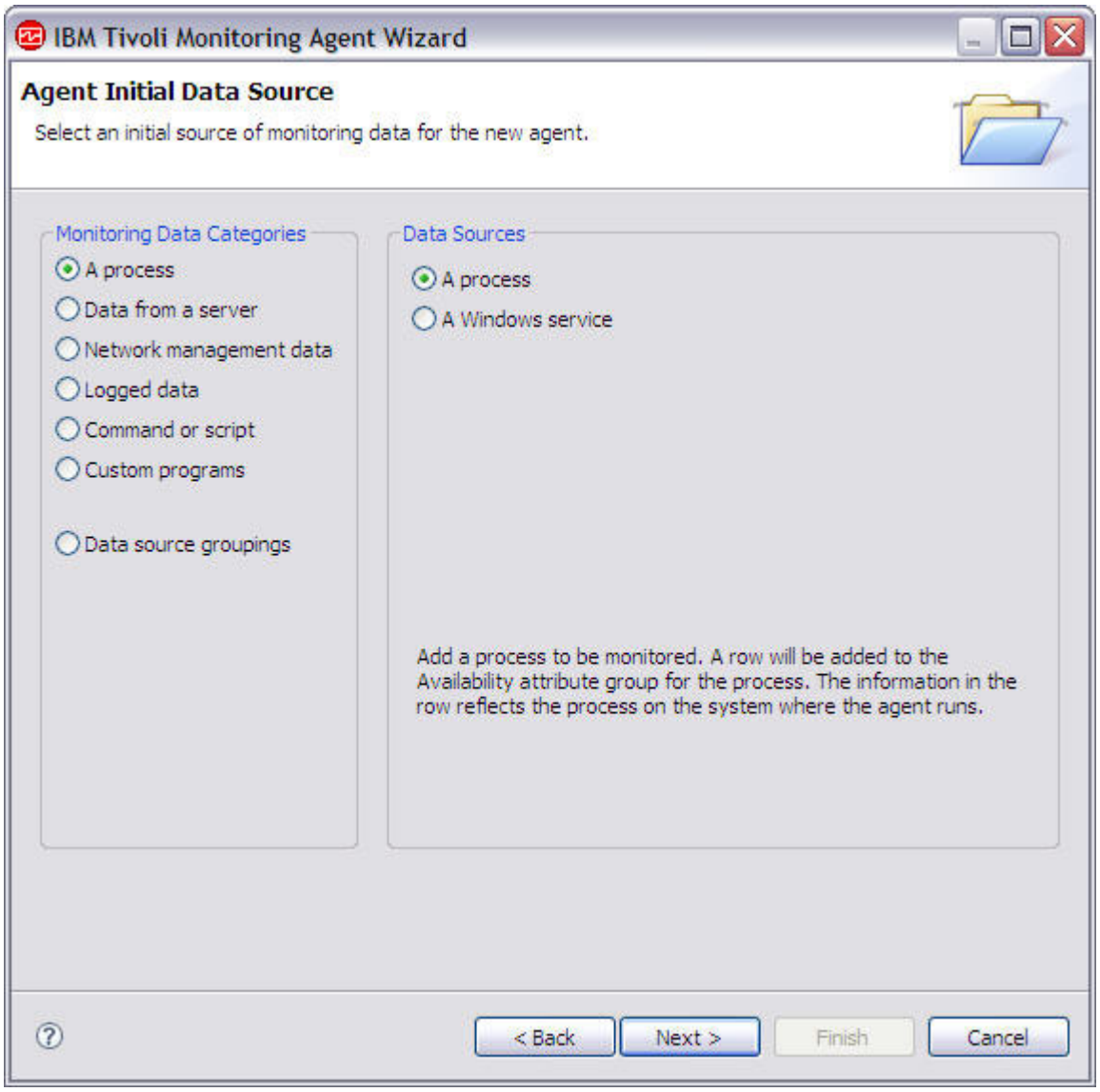


Figure 54. Adding a server process

2. In the **Data Sources** area, click **A process**.
3. Click **Next**.

© Copyright IBM Corp. 2007, 2013

4. On the Process Monitor page (Figure 55), Process information area, provide the display name and process name. You can type the process name manually or obtain it by clicking **Browse**. Clicking **Browse** shows a list of processes that are currently running on the local system or on a remote system.

   You can further discriminate processes, by selecting the **Use argument match** and **Match full command line** options. For example, if multiple instances of the same processes are running on the system, one instance can be distinguished from another by using these options.

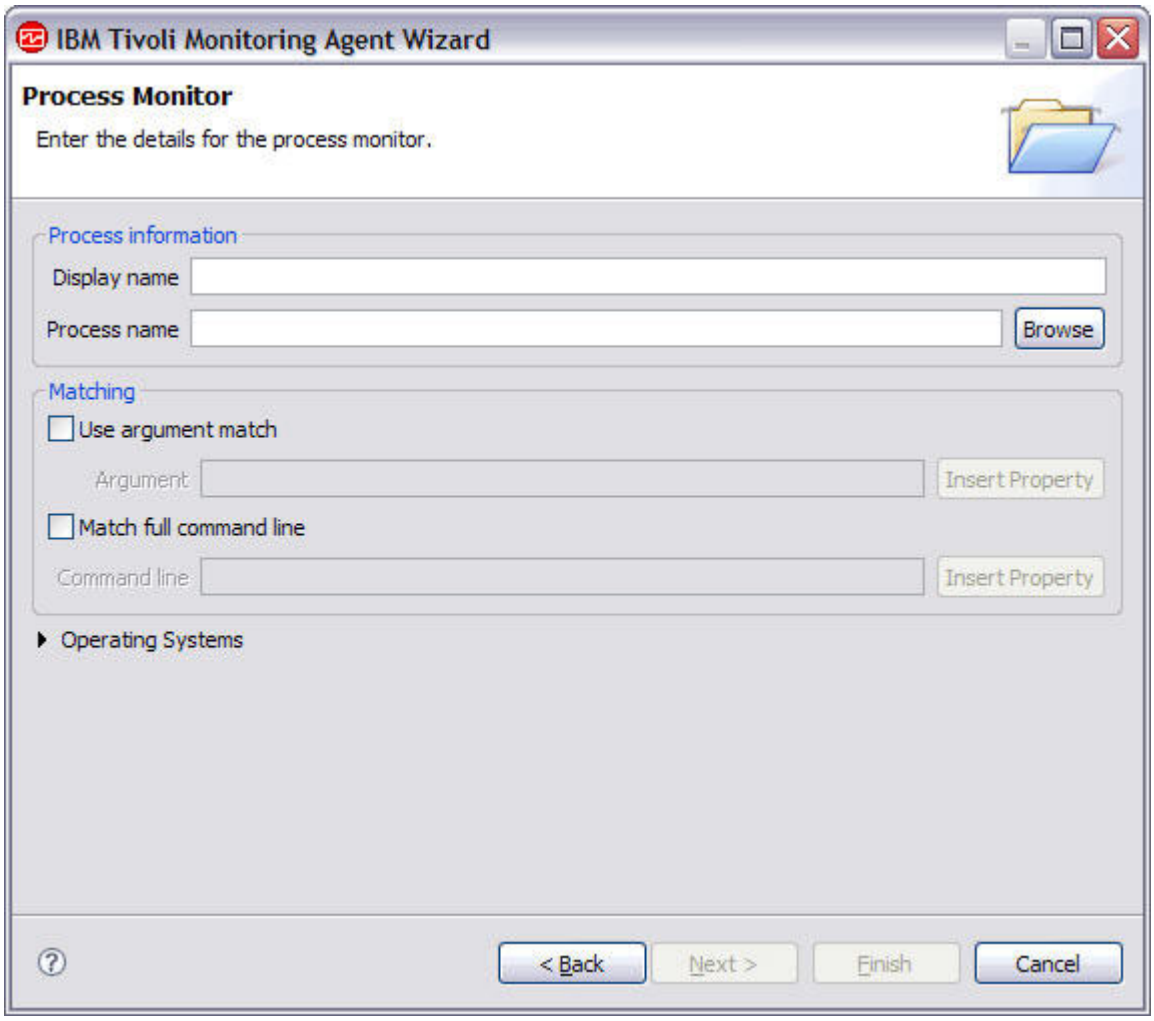


Figure 55. Process Monitor page

Table 11. Fields on the Process Monitor page

Field name	Description	Acceptable values
Display name	Descriptive name for the component of the application that is implemented by the process as it is shown in the Tivoli Enterprise Portal	Descriptive string
Process name	Name of the process that is being monitored	Valid executable file name
Use argument match	Select if you want to match on the process arguments.	On or Off

Table 11. Fields on the Process Monitor page (continued)

Field name	Description	Acceptable values
Argument	Argument string on which to match. Argument matching looks for the provided string as a substring of the arguments. Matching is successful if you provide any part of the arguments as the input string.	String
Match full command line	Specify the entire name of the executable file that might include the path	On or Off
Command line	Matches the provided string against the fully qualified command name that is used to start the process. Command arguments are not included. Fully qualified means the path to the command must be included.	String
Operating systems	Select the operating systems on which this process runs	Any selection

5. If you clicked **Browse**, the Process Browser window opens (Figure 56). This window initially contains detailed information about each process on the Agent Builder system. The information includes the ID, the process name, and the full command line for the process. Select one or more processes or work with the list in the Process Browser window by using one or more of the following actions:

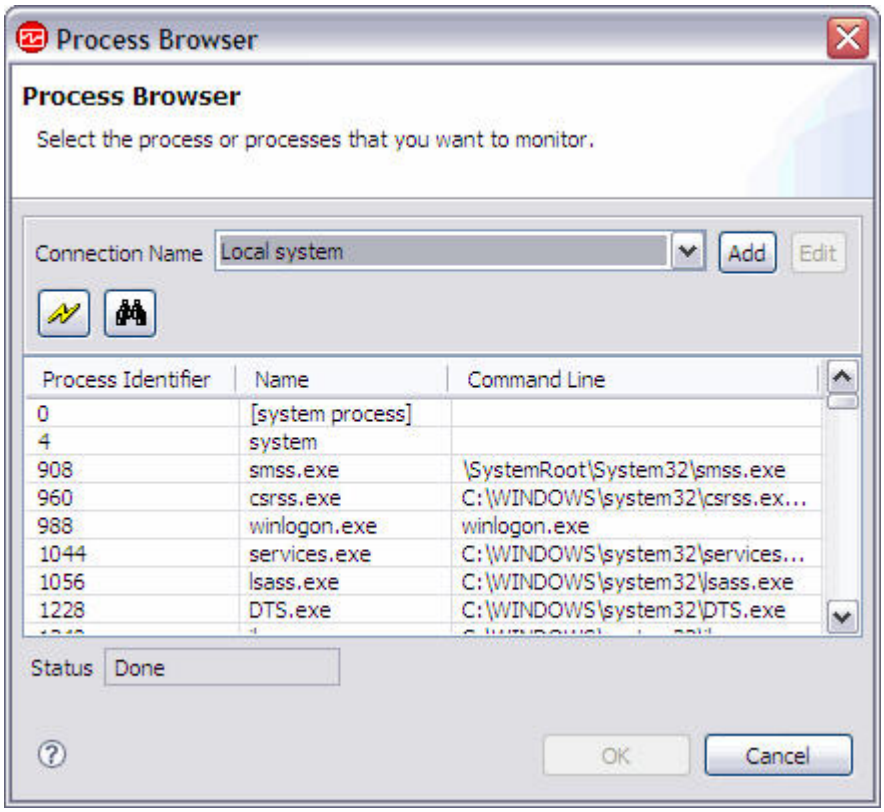


Figure 56. Browse the list of currently running processes

Note: When you browse remote systems, the command-line details are available only when you browse through a Tivoli Enterprise Portal Server.

a. To sort the list of processes, click the column heading.

b. To refresh the information in the window, click the **Refresh** (lightening bolt) icon.

c. To search for specific processes, click the **Search** (binoculars) icon.

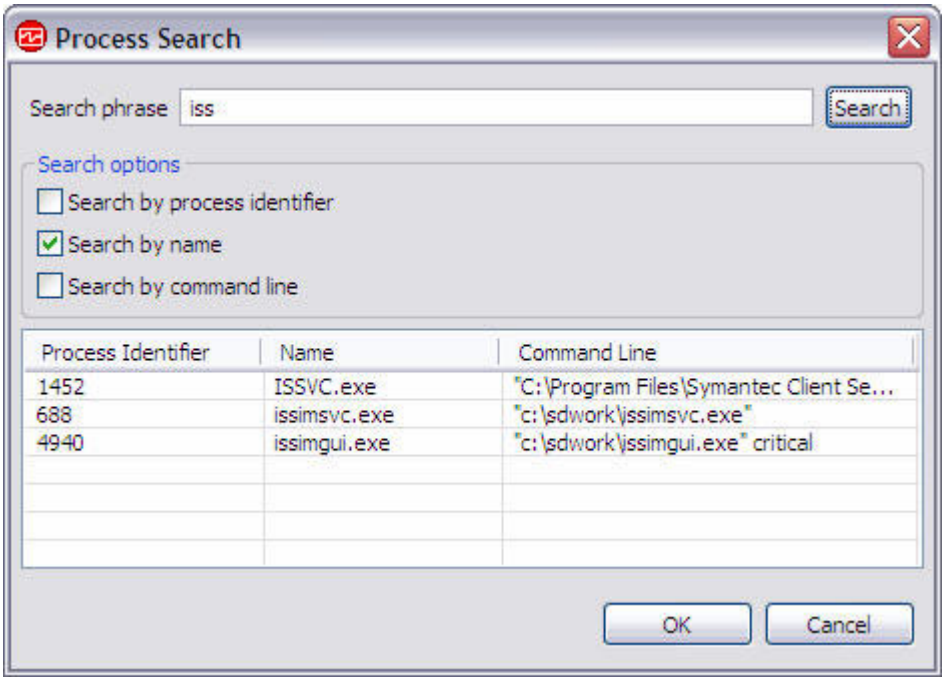


Figure 57. Search for a process

d. To view processes on a different system, select a previously defined system from the **Connection Name** list. Or click **Add** to enter the system information for a new system. For more information, see ("Defining connections for process browsing" on page 93). You can load processes from more than one system at a time, and switch between connections while processes are loading for one or more connections.

In the following example, after you select `svchost.exe`, it is shown in the **Process name** field on the Process Monitor page (Figure 58 on page 93).

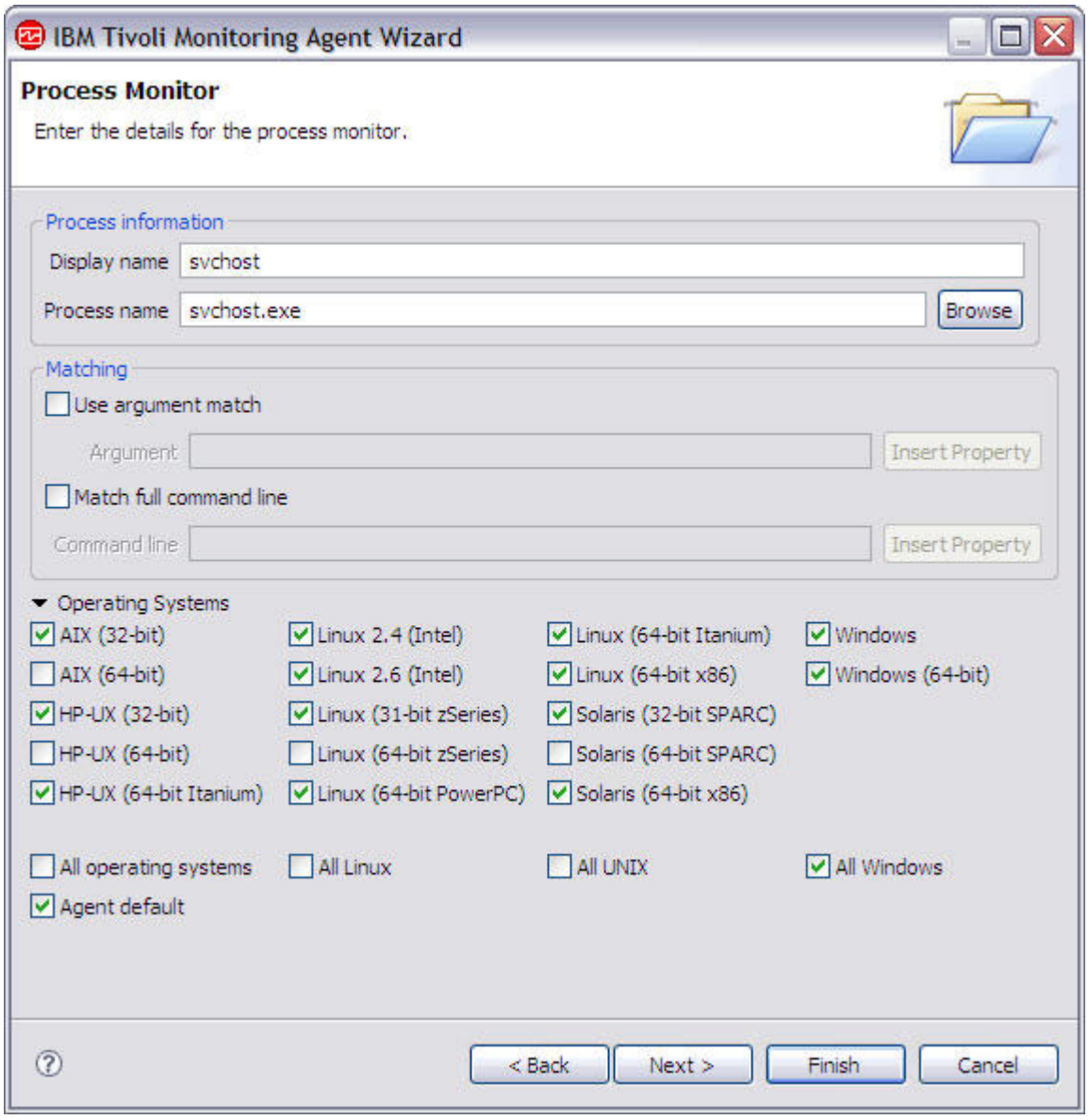


Figure 58. Process Monitor page example

6. Complete the Process Monitor page by using the information in (Table 11 on page 90).

   Note: If the process you described in this monitor is applicable to only some of the operating systems that your application runs on, you might want to create one or more process monitors with the same display name to cover the other operating systems. Add the process monitors one at a time. Ensure that the display name is the same for each monitor, but that the process name can be found on the operating systems that are selected.
7. Do one of the following steps:
   - If you are using the New Agent wizard, click **Next**.
   - Click **Finish** to save the data source and open the Agent Editor.

Defining connections for process browsing

When you define a process data source, you can select processes from other systems.

About this task

You must have credentials for the other systems or they must be monitored by a Tivoli Monitoring operating system agent.

Procedure

1. To define a connection, click **Add** in the Process Browser window (Figure 56 on page 91). You can select either a connection type (Secure Shell (SSH), Windows, or Tivoli Enterprise Portal Server Managed System) or select an existing connection to use as a template.

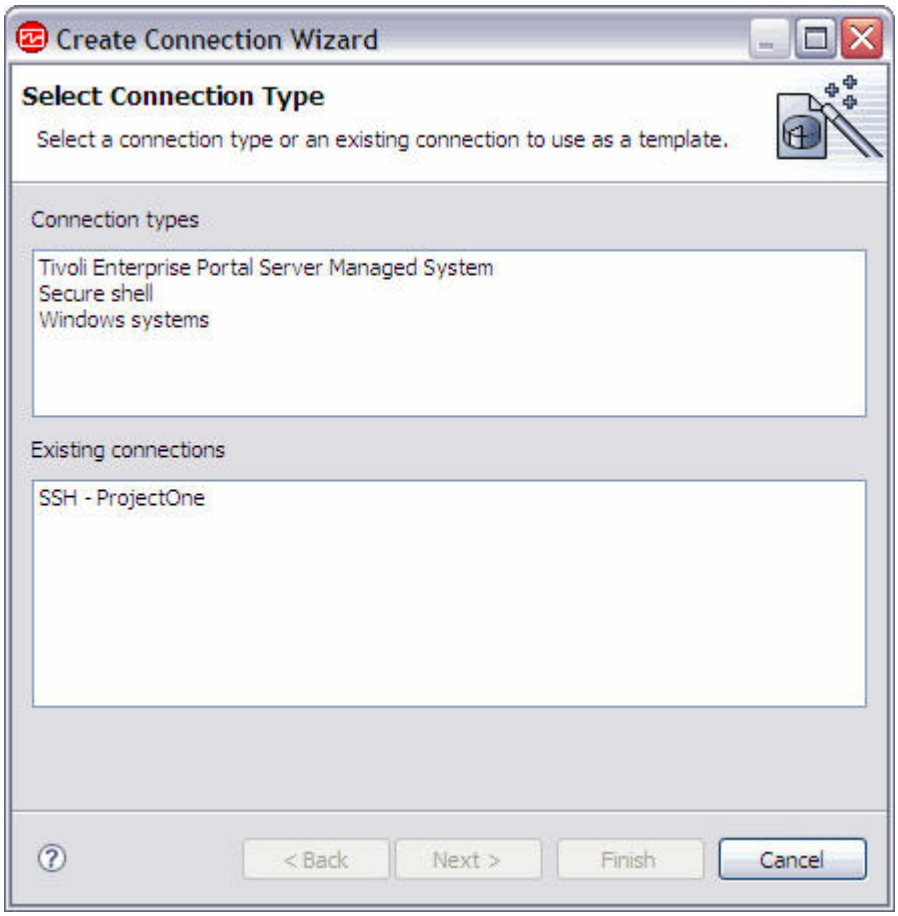


Figure 59. Select a connection type or connection template for process browsing

   To add a Managed System connection, you require a Tivoli Enterprise Server host name, Tivoli Monitoring user name, and password. You also require the managed system name of the remote connection. When a managed system is selected, the table lists the process on the remote system.

   Note: The OS agent must be running on the system you are attempting to browse. The agent must also be connected to a running Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server.

   To add Secure Shell (SSH) or Windows connections, you require a host name, user name, and password.

2. When you add a connection, you can select the connection from the **Connection Name** list in the Process Browser window.

   If all the fields required to make the connection are not saved (for example, the password), the **Connection Properties** window for that connection opens. Enter the missing information. For Tivoli

Enterprise Portal Server Managed System connections, you must connect to the Tivoli Enterprise Portal Server before you can enter a managed system.

3. Enter your user name and password, and then click the **Refresh** (lightening bolt) icon to connect before you select the managed system.

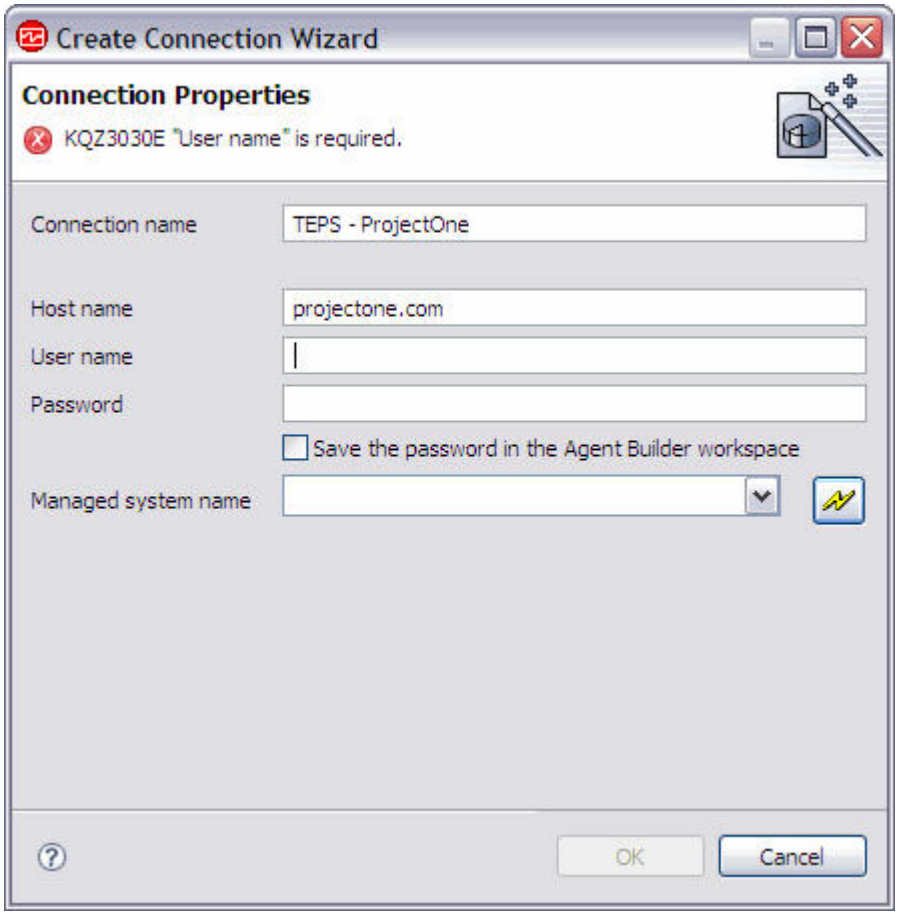


Figure 60. Edit connection properties for Tivoli Enterprise Portal Server Managed System

What to do next

To delete a connection, select the connection and click **Edit** to open the Connection Properties window. Select the **Remove this connection** check box and click **OK**.

Chapter 9. Monitoring a Windows service

Define a data source to collect data from a Windows service.

Procedure

1. On the Agent Initial Data Source page (Figure 61) or the Data Source Location page, click **A process** in the **Monitoring Data Categories** area.

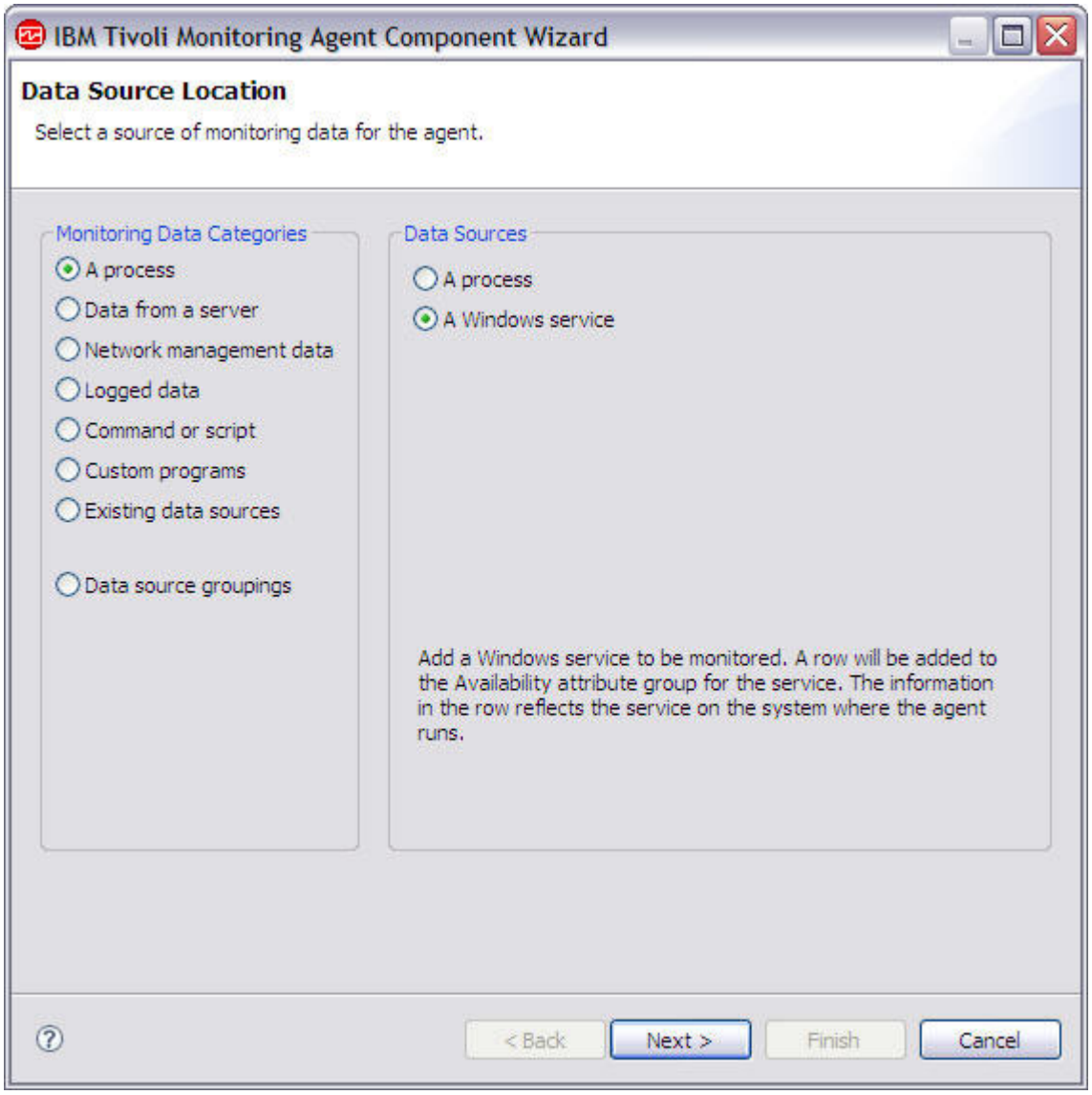


Figure 61. Adding a Windows service monitor

2. In the **Data Sources** area, click **A Windows service**.
3. Click **Next**.
4. On the Service Monitor page, in the **Display name** field, provide a description for the name of the service application to be shown in the Tivoli Enterprise Portal. You can type it manually or click

© Copyright IBM Corp. 2007, 2013

Browse to view a list of services that are currently running on the local system or on a remote system. If you click **Browse**, the Service Browser window opens (Figure 63 on page 99). This window initially

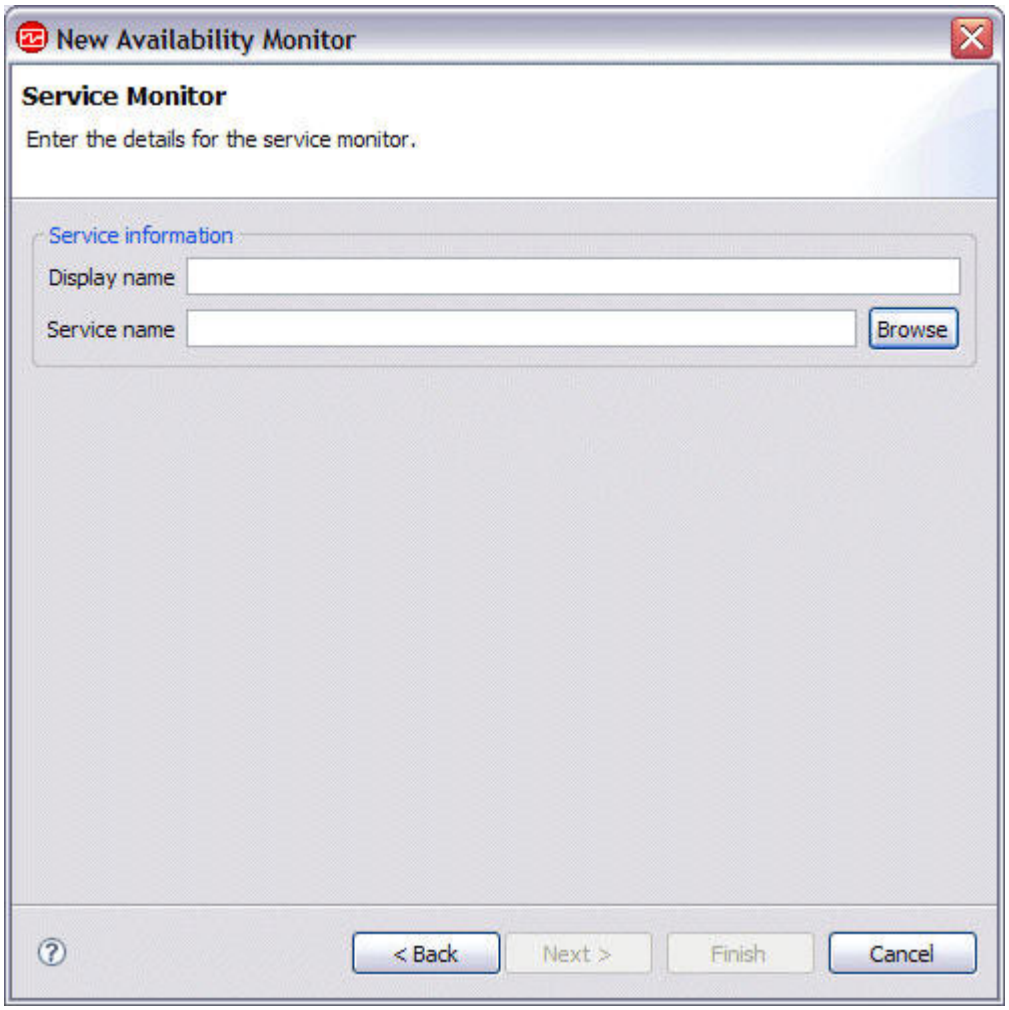


Figure 62. Service Monitor page

contains detailed information about each service on the Agent Builder system. The information includes the service name, the display name, the state, and the description for the service.

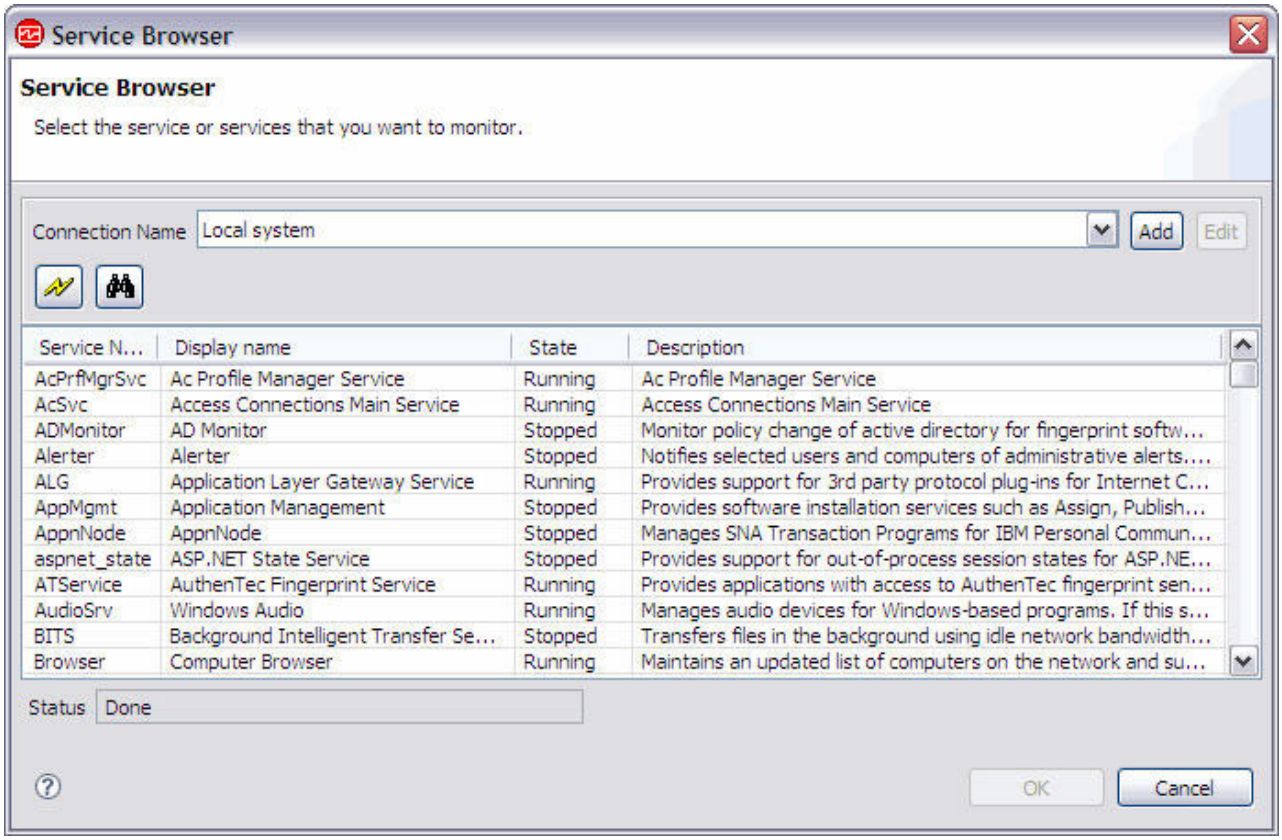


Figure 63. Browse the list of all services defined in the system

Note: Local services are not shown when Agent Builder is not running on a Windows system. A remote system Windows must be defined or selected, see ("Defining connections for service browsing" on page 101).

Note: The service description is not available when you are browsing through the Tivoli Enterprise Portal Server or from a UNIX or Linux system.

5. Select one or more services or do one or more of the following steps to work with the list in the Service Browser window:
   - To sort the list of services, click the column heading.
   - To refresh the information in the window, click the **Refresh** (lightening bolt) icon.
   - To search for a service, click the **Search** (binoculars) icon to open the Service Search window.

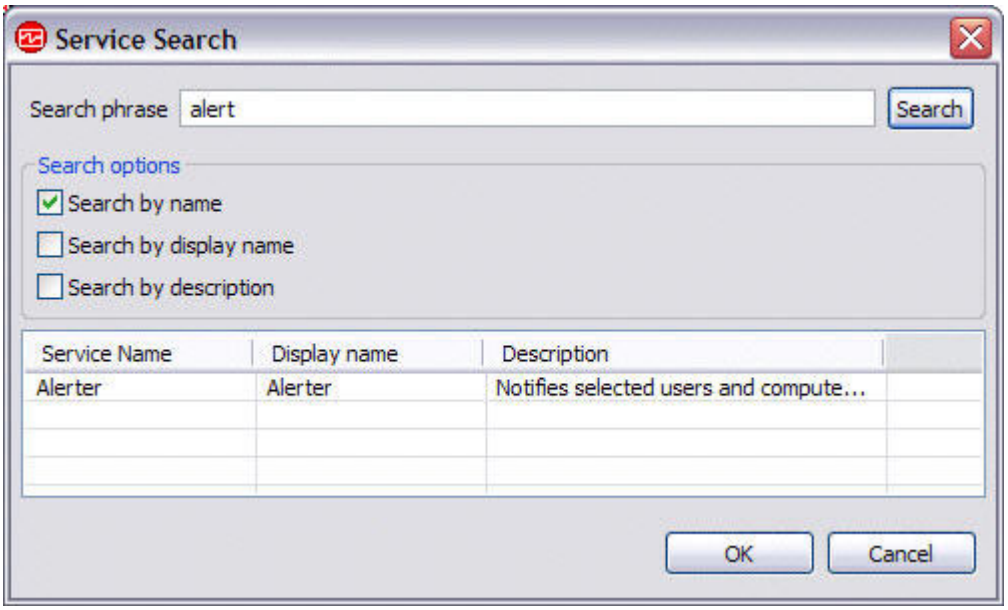


Figure 64. Search for a service

- To view services on a different system, select a previously defined system from the **Connection Name** list or click **Add** to enter the system information. For more information, see ("Defining connections for service browsing" on page 101). You can load services from more than one system at a time, and switch between connections while services are loading for one or more connections.

In the following example, after you select `ccEvtMgr`, it is shown in the **Service name** field on the Service Monitor page (Figure 65 on page 101).

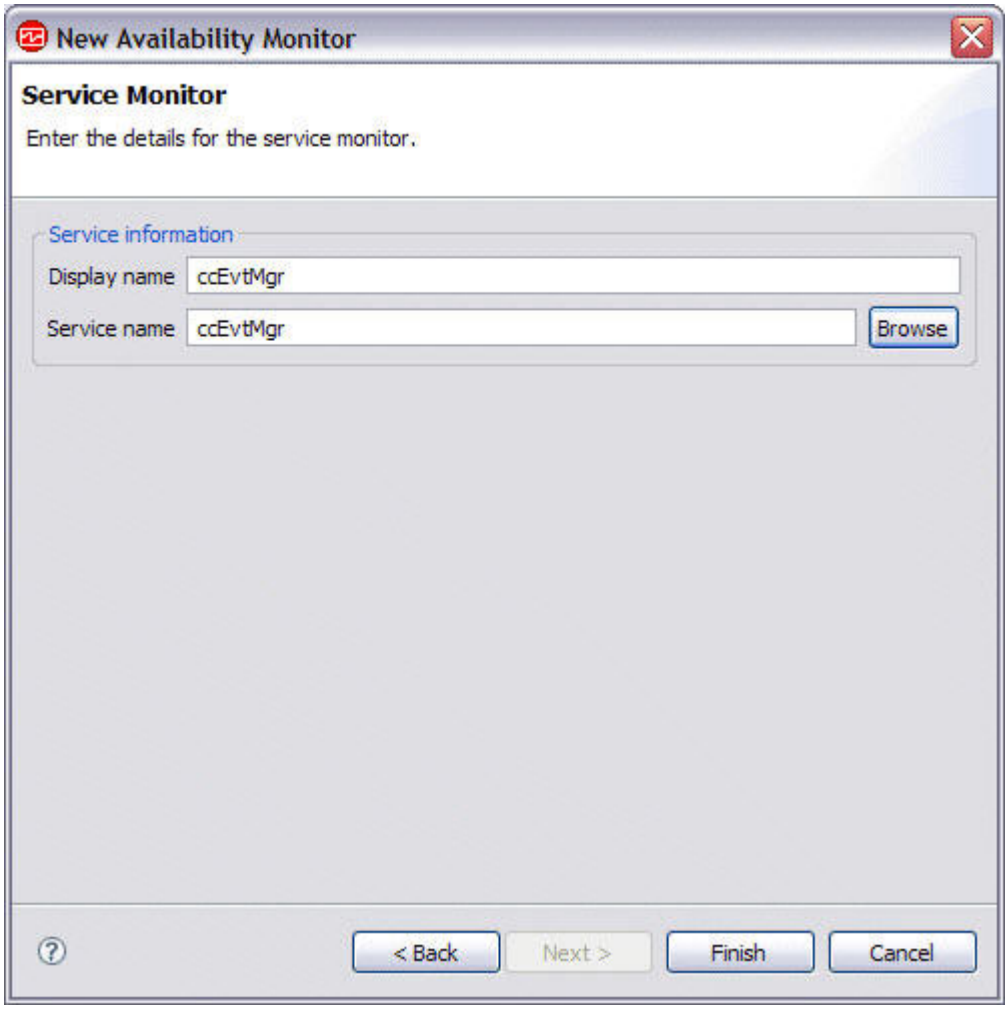


Figure 65. Service information

6. Do one of the following steps:
   - If you are using the New Agent wizard, click **Next**.
   - Click **Finish** to save the data source and open the Agent Editor.

Defining connections for service browsing

In addition to selecting services from the system where Agent Builder is running, you can select services from other Windows systems.

About this task

To select services from other Windows systems, you define a connection to the remote system. You must have credentials for the systems or they must be monitored by a Tivoli Monitoring operating system agent.

Procedure

1. To define a connection, click **Add** in the Service Browser window (Figure 63 on page 99).
   The Select Connection Type window opens.

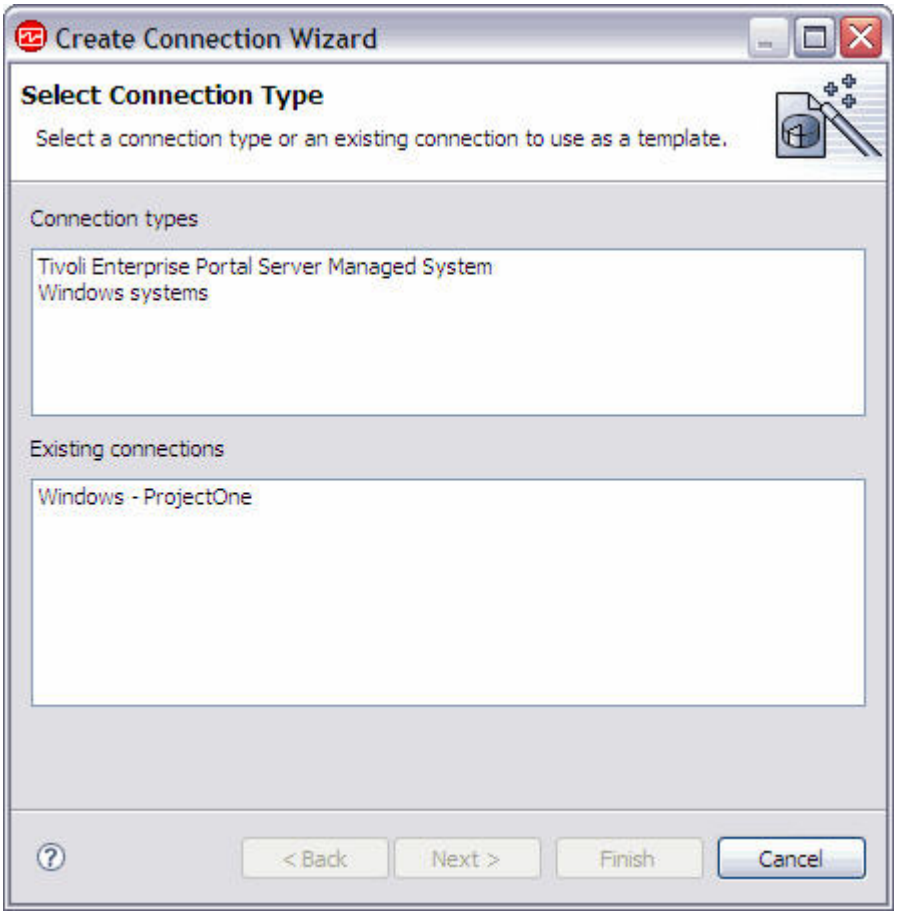


Figure 66. Select a connection type or connection template for service browsing

To add a Managed System connection, you require a Tivoli Enterprise Server host name, Tivoli Monitoring user name and password, and the managed system name. When a managed system is selected, the table lists the service on the remote system.

Note: The OS agent must be running on the system you are attempting to browse and also connected to a running Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server.

You require a host name, user name, and password to add a Windows connection.

2. Select a connection type (Windows, or Tivoli Enterprise Portal Server Managed System) or select an existing connection to use as a template (Figure 66). The Connection Properties window opens.
3. Complete the Connection Properties.
4. Click **Finish**
5. When you add a connection, you can select the connection from the **Connection Name** list in the Service Browser window. If the fields necessary to make the connection are not saved (for example, the password), the**Connection Properties** window opens and you can enter the missing information.
   a. For Tivoli Enterprise Portal Server Managed System connections, you must connect to the Tivoli Enterprise Portal Server before you can enter a managed system. Enter your user name and password, and then click the **Refresh** (lightening bolt) icon to connect before you select the managed system.

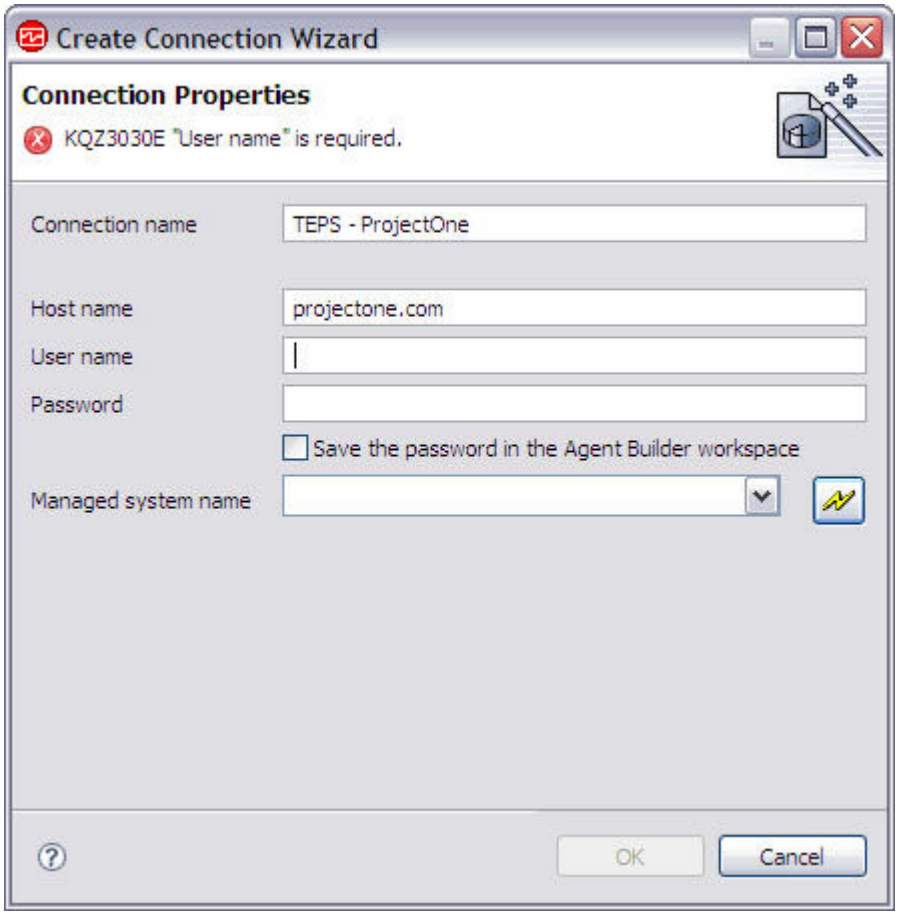


Figure 67. Edit connection properties for Tivoli Enterprise Portal Server Managed System

6. To delete a connection, follow these steps:
   a. Select the connection in the Service Browser window.
   b. Click **Edit** to open the Connection Properties window.
   c. Select the **Remove this connection** check box.
   d. Click **OK**.

Chapter 10. Monitoring data from Windows Management Instrumentation (WMI)

Define a data source to collect data from Windows Management Instrumentation (WMI).

Before you begin

If your agent collects data from a remote system by using Windows Management Instrumentation (WMI), it requires permissions to access WMI data on the remote system. The agent can access WMI data on a remote system when you provide credentials of an account with permissions to access WMI data on the system. The Administrator account has the required permissions. In the procedure that follows you can either provide the Administrator credentials or the credentials of another user with the required permissions. For more information about creating a user account with permissions to browse WMI data, see ("Creating a user with Windows Management Instrumentation (WMI) permissions" on page 373).

About this task

To collect metrics through the Windows APIs, the agent must be hosted on a Windows operating system. Remote registry administration must be enabled on the remote systems.

Procedure

1. On the Agent Initial Data Source page (Figure 68 on page 106) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

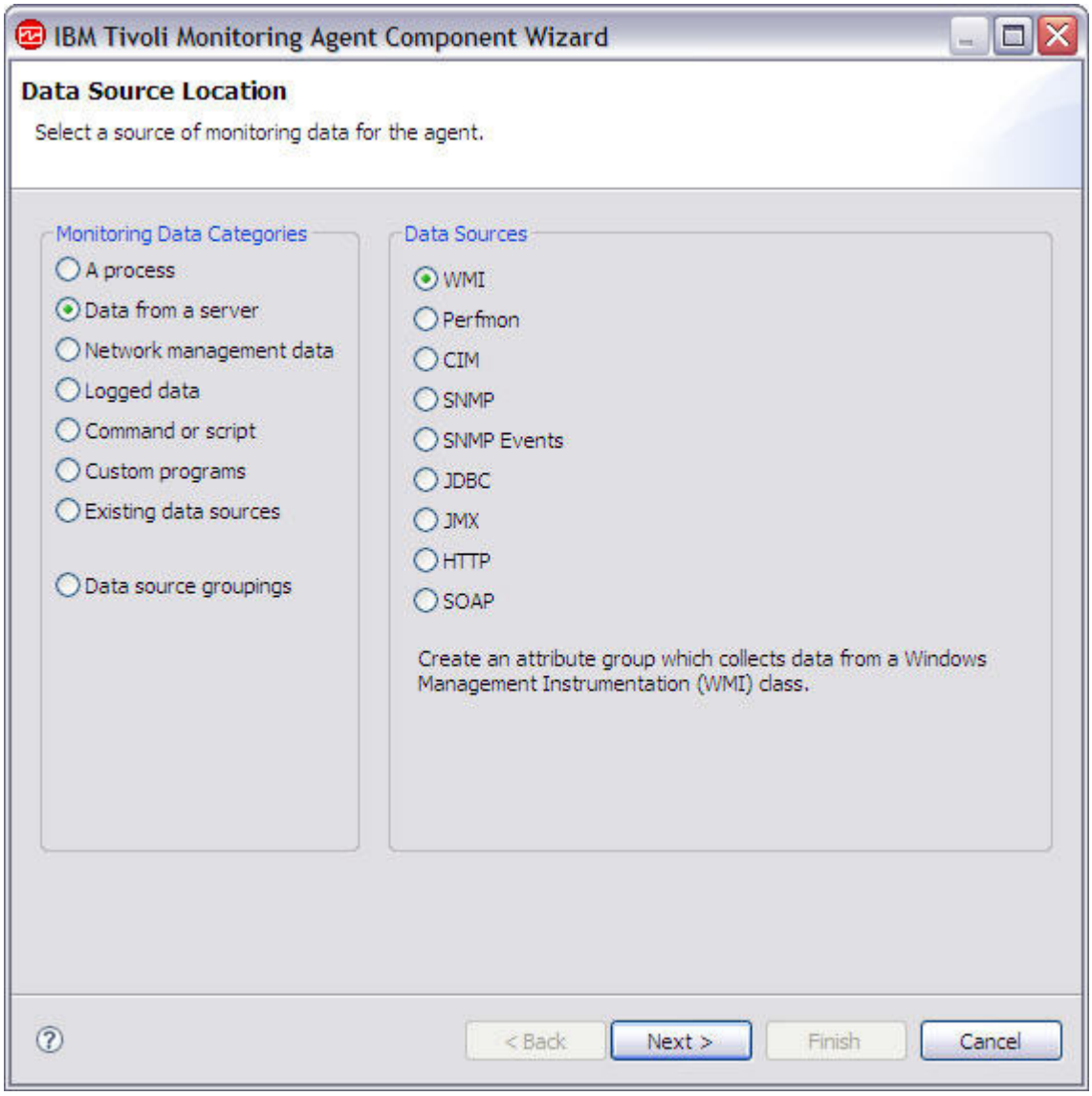


Figure 68. Adding WMI data

2. In the **Data Sources** area, click **WMI**.
3. Click **Next**.
4. On the Windows Management Instrumentation (WMI) Information page (Figure 69 on page 107), do one of the following steps:

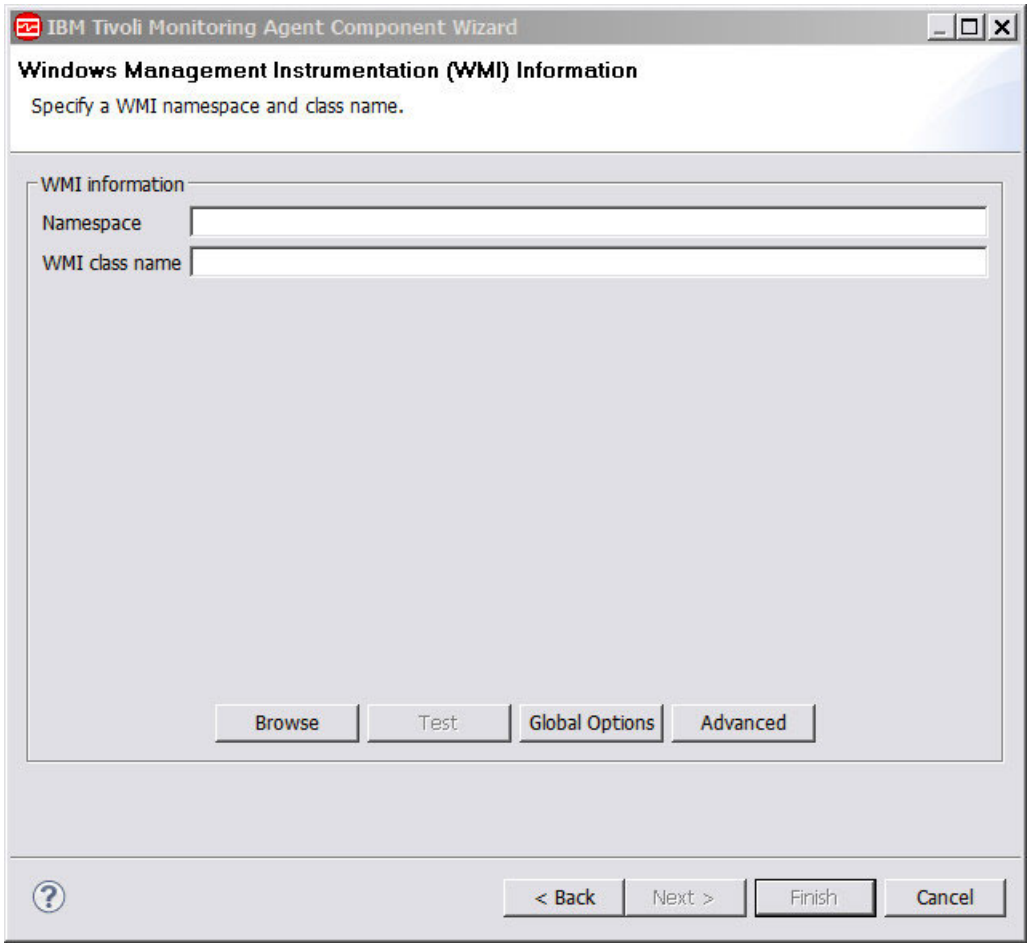


Figure 69. WMI Information page

- Type a name for the WMI namespace and a name for the WMI class name in the fields. Now go to step 9 on page 109
- Click **Browse** to see all of the WMI classes on the system (Figure 70 on page 108).

  To browse a remote system, select a system from the list (if one is defined). Alternatively click **Add** to add the host name of a Windows system. Provide the credentials of a user account with permissions to access WMI data on the remote system, or provide Administrator credentials for the remote system. The page is updated with the information for the remote system. Browsing is only available when the Agent Builder is run on a Windows system, and can browse Windows systems only. For instance, you cannot add the host name of a Linux or Solaris system to do a remote browse.

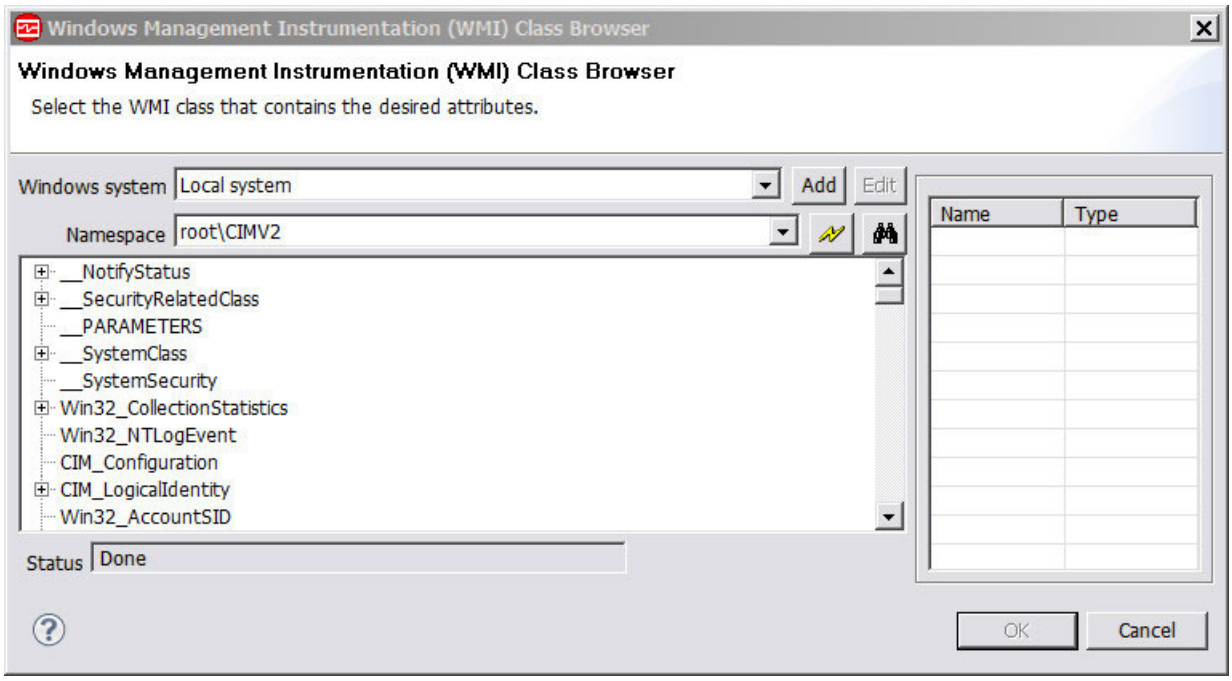


Figure 70. List of classes with their associated attributes

5. Click the plus sign (+) next to a class to expand the class and show the attributes.
6. From the list, select the class with its associated attributes that you want to specify, and click **OK**.

   Note: You can click the **Search** (binoculars) icon to find your selection in the list. Type a phrase in the **Search phrase** field; specify your preference by clicking either the **Search by name**, **Search by class description**, or **Search by class properties** fields; and click **OK** (Figure 71). If you find the item for which you are searching, select it and click **OK**.
   The WMI Information page of the wizard opens again, showing the selected WMI class information

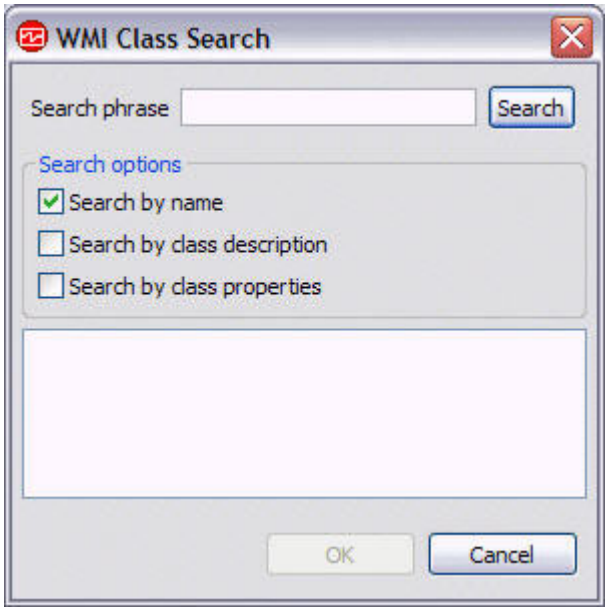


Figure 71. Optionally specifying WMI search options

(Figure 72 on page 109).

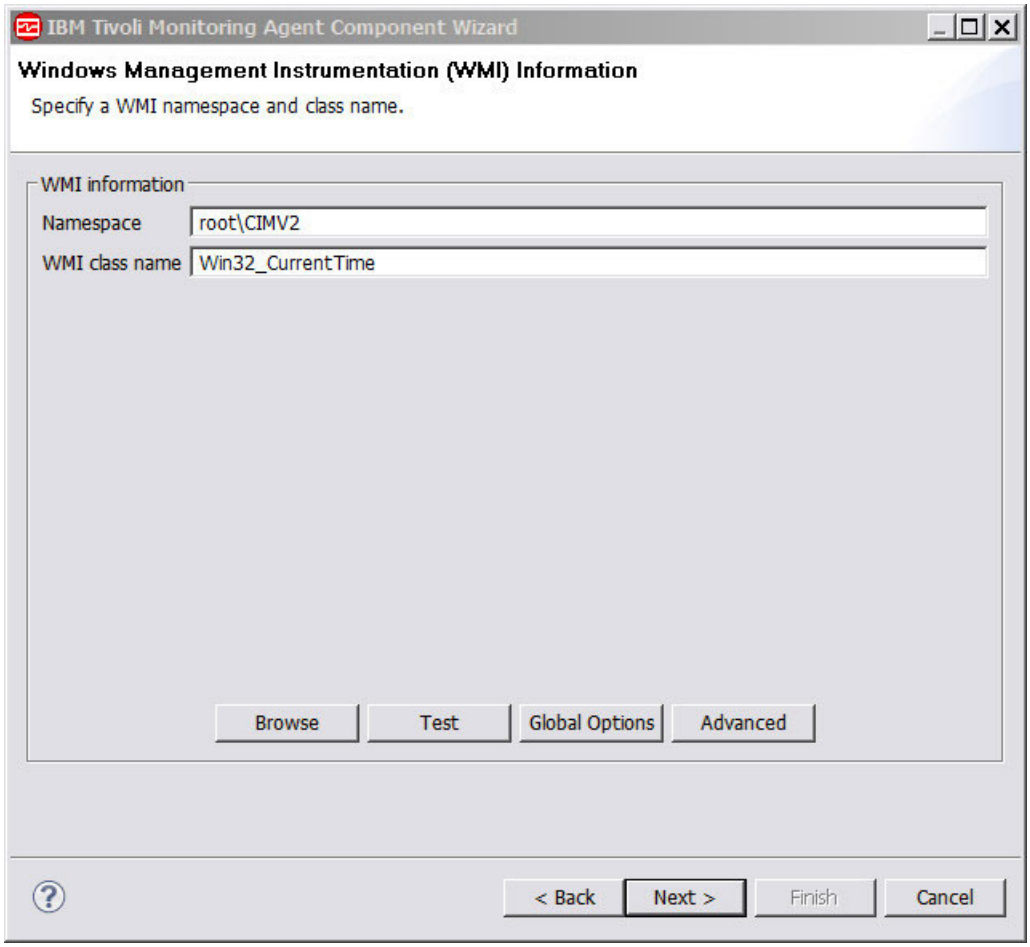


Figure 72. WMI Information page completed

7. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing WMI attribute groups" on page 111
8. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 67
9. Click **Next**.

   Note: If you typed the WMI Namespace and WMI class name manually you are brought to the Attribute Information page, where you can complete attribute information. On the Attribute Information page, you can select **Add additional attributes** if you want to add more attributes. Click **Finish** to complete.
10. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 26).
11. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

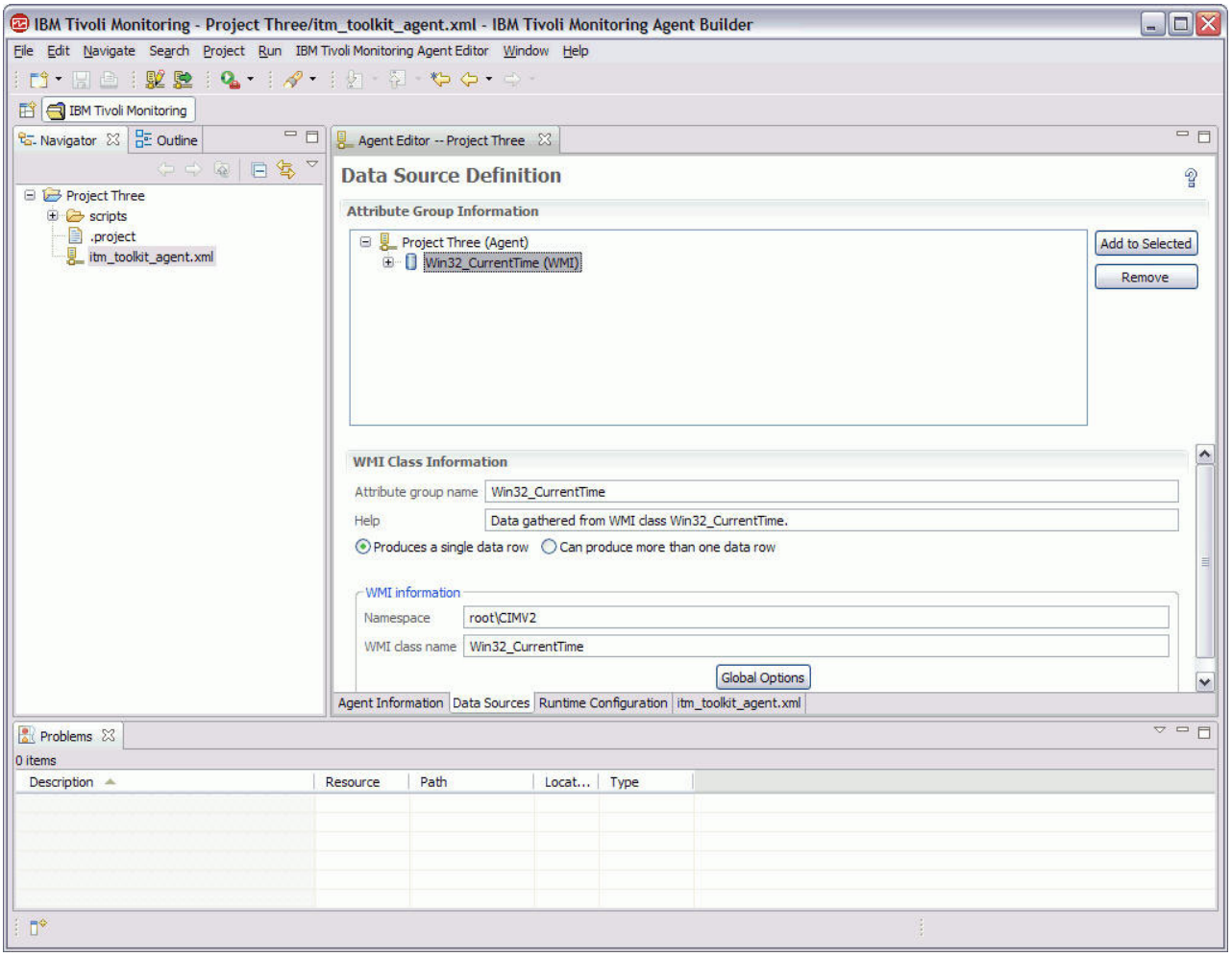


Figure 73. WMI shown on Data Source Definition page in the Agent Editor

12. You can add attributes and supply the information for them. For more information, see ("Creating attributes" on page 56).

    In addition to fields that are applicable to all data sources (Table 7 on page 58), the Attribute Information page for the WMI data source has the following field:

    Metric name
    Property name from the class you want to collect

13. If you want to set global options for the data source, click **Global Options** (Figure 74 on page 111).

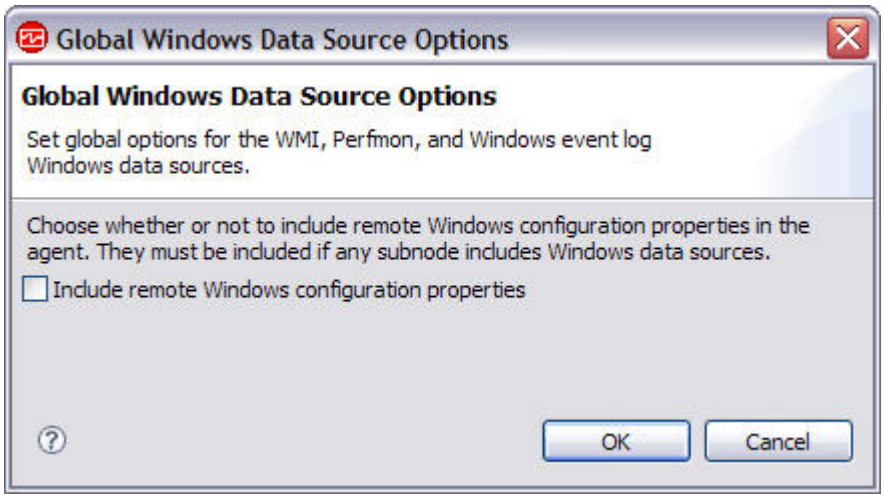


Figure 74. Global Windows Data Source Options window

Select the **Include remote Windows configuration properties** check box if you want to include this option, and click **OK**.

For information about Windows remote connection configuration for Windows data sources, see ("Configuring a Windows remote connection" on page 372).

Testing WMI attribute groups

If you are running the Agent Builder on a Windows system, You can test the WMI attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the WMI Information page (Figure 72 on page 109).
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see (Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35).

   After you click **Test** in one of the previous two steps, the WMI Test window Figure 75 on page 112 is displayed.

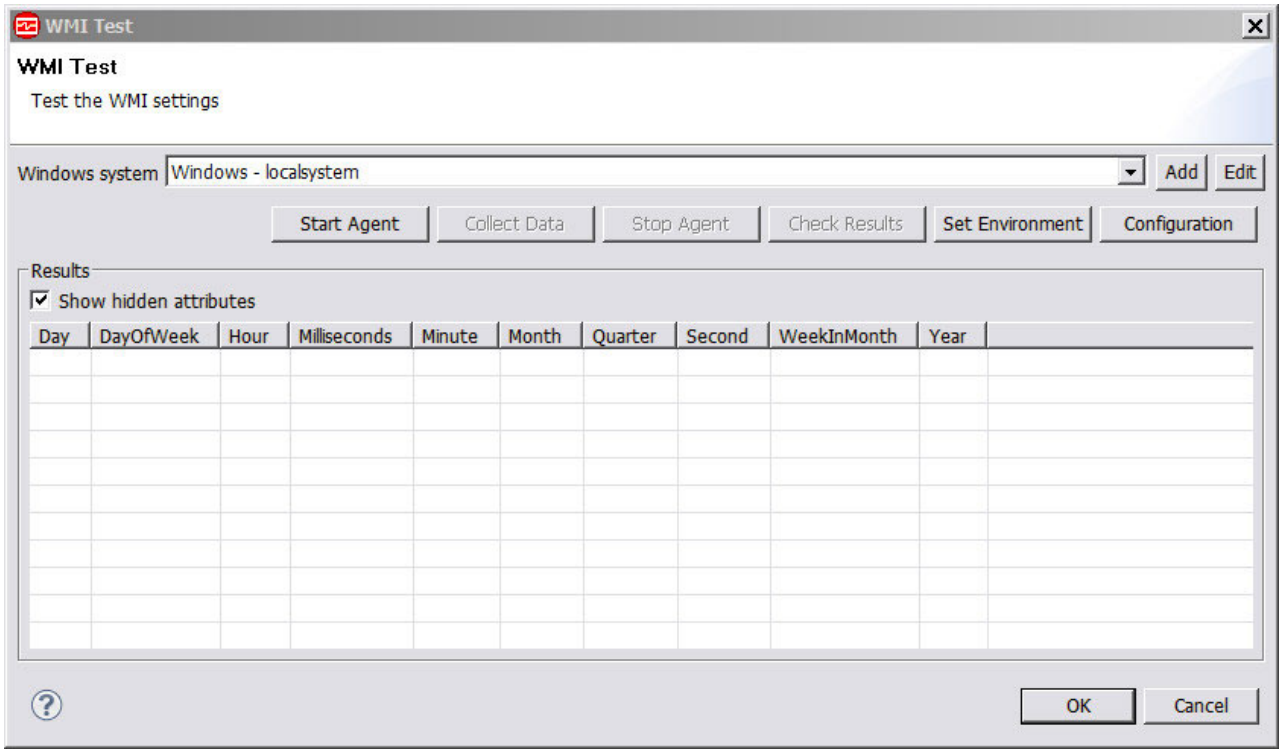


Figure 75. WMI Test window

2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see (“Attribute group testing” on page 377).
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries WMI for data. The WMI Test window collects and displays any data in the agent's cache since it was last started.
5. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in (“Performance Object Status node” on page 545).
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the WMI Test window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, “Testing and debugging your agent,” on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 11. Monitoring a Windows Performance Monitor (Perfmon)

Define a data source to collect data from Windows Performance Monitor.

Procedure

1. On the Agent Initial Data Source page (Figure 76) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

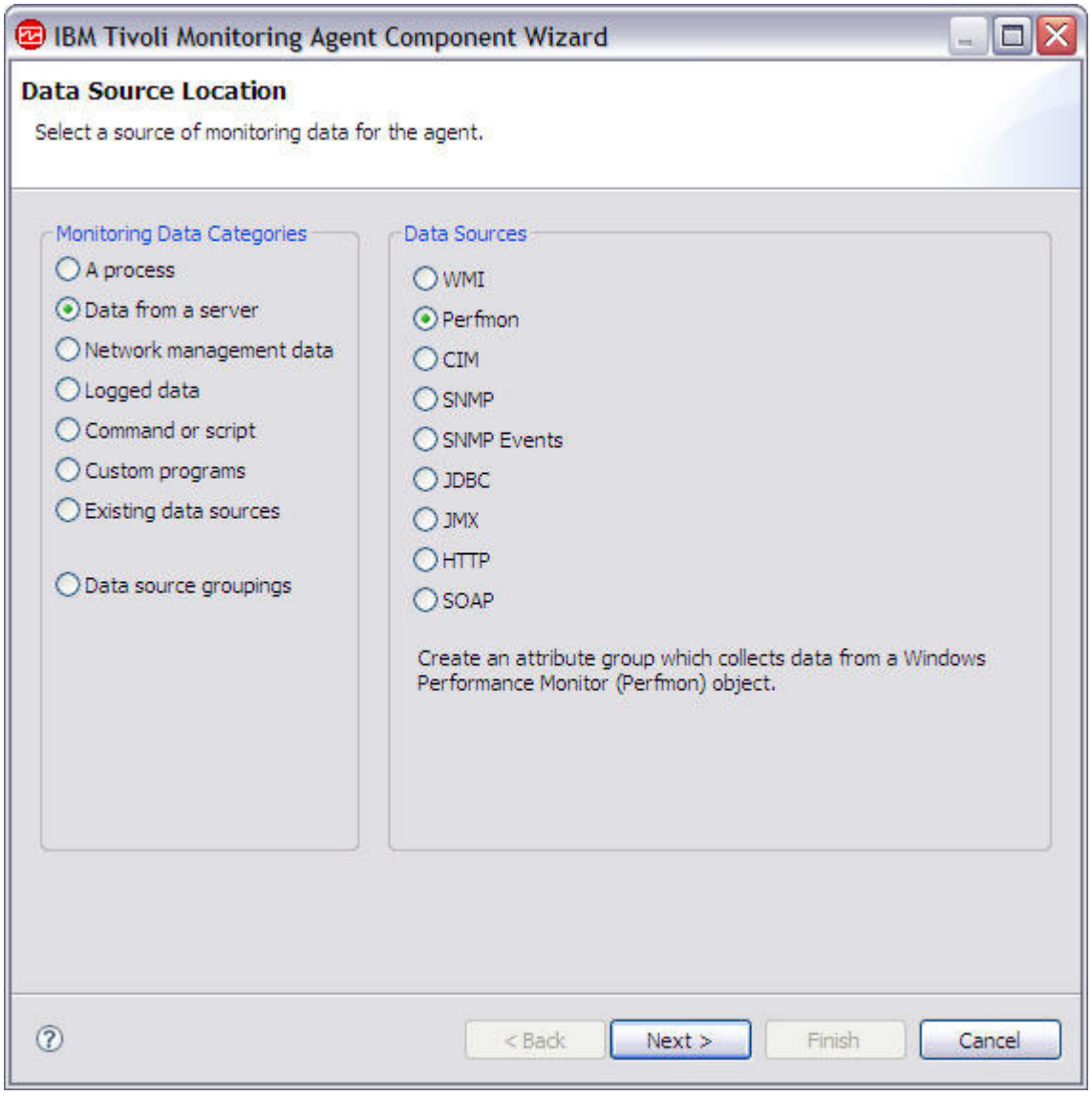


Figure 76. Adding Perfmon data

2. In the **Data Sources** area, click **Perfmon**.
3. Click **Next**.
4. On the Perfmon Information page, do one of the following steps:

© Copyright IBM Corp. 2007, 2013

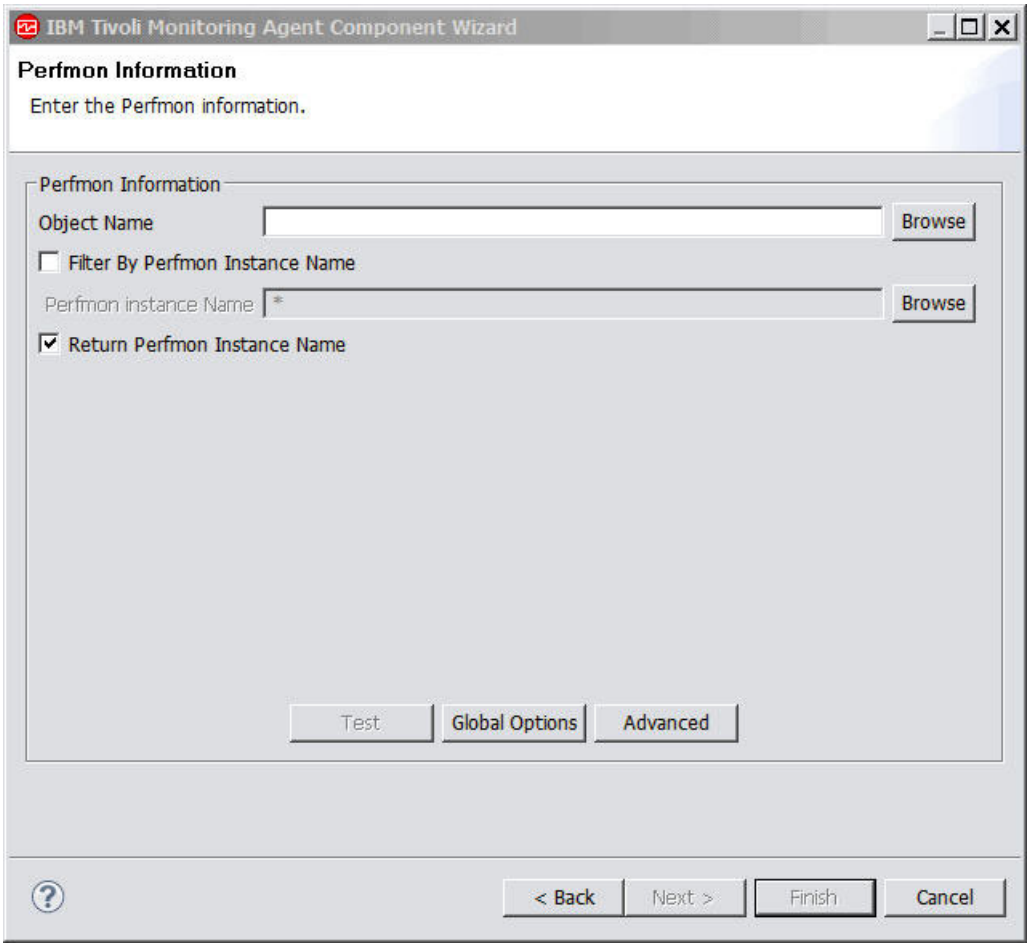


Figure 77. Perfmon Information page

- Type the name of the object in the **Object Name** field, and click **Next** to define the first attribute in the attribute group.

  Note: If you type the name for the Windows Performance Monitor object, it must be the English name.
- Click **Browse** to view the list of Perfmon objects.

  When the Performance Monitor (Perfmon) Object Browser (Figure 78 on page 115) window initially opens, the window populates with the information from the local system. To browse a remote system, select a system from the list (if one is defined), or click **Add** to add the host name of a Windows system. Provide an Administrator ID and password. The window updates with the information for the remote system. Browsing is available only when the Agent Builder is running on a Windows system, and can browse only Windows systems. For instance, you cannot add the host name of a Linux or Solaris system to do a remote browse.
  - When you click an object name, the available counters in that object are shown in the opposite window.

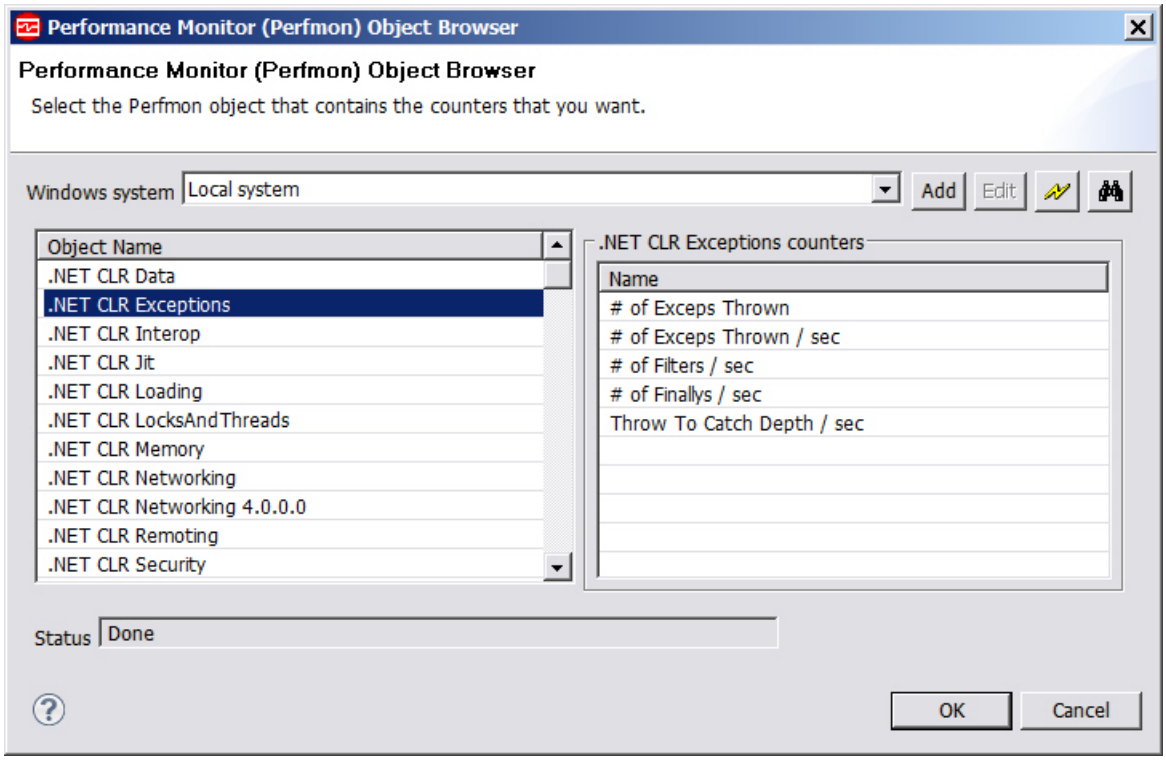


Figure 78. Browsing for Windows Performance Monitor objects

- To sort the Windows Performance Monitor objects or counters, click the column heading.
- To refresh the information in the window, click **Refresh**.
- To search for specific objects or counters click the **Search** (binoculars) icon to open the Performance Monitor Search window (Figure 79). You can search object names, counter names, or both. The search operation does a substring match and is not case-sensitive.

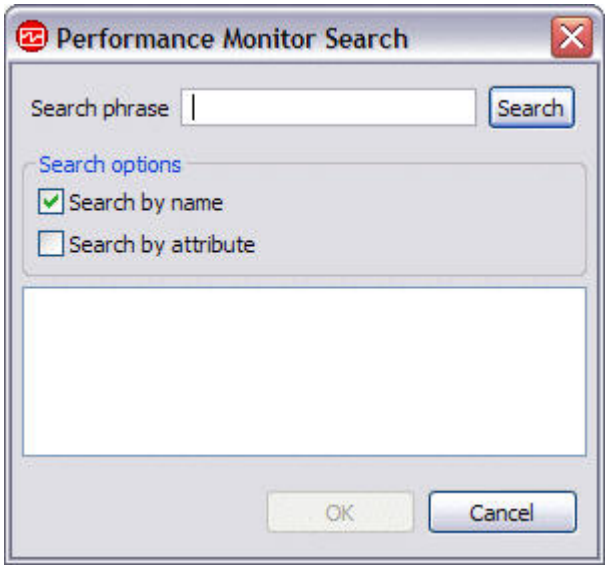


Figure 79. Performance Monitor Search window

- Select an object and click **OK**.

- The Perfmon Information page opens with the name of the selected object in the **Object Name** field.

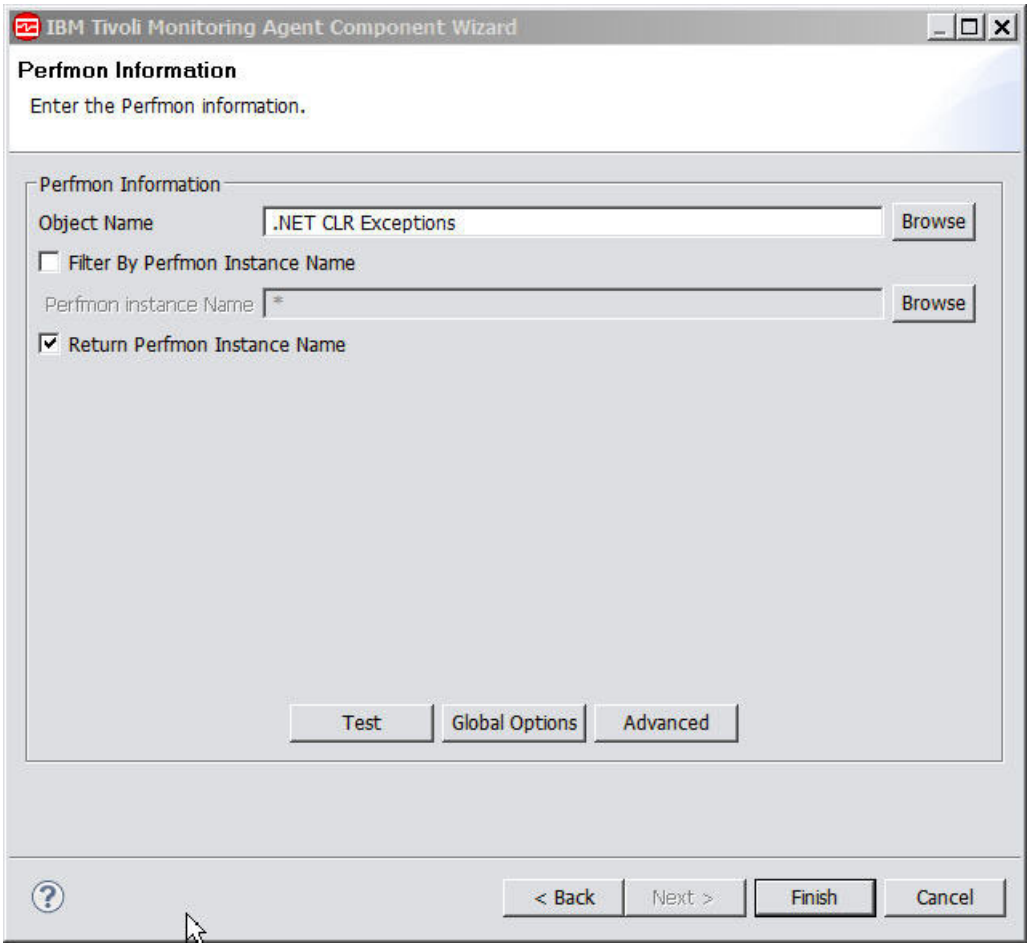


Figure 80. Perfmon Information page with Object Name

- If you want to set global options for the data source, click **Global Options**



Figure 81. Global Windows Data Source Options window

Select the **Include remote Windows configuration properties** check box if you want to include this option, and click **OK**.

For information about Windows remote connection configuration for Windows data sources, see ("Configuring a Windows remote connection" on page 372).

5. If the Windows Performance Monitor object selected returns multiple instances and you want to filter the results that are based on the instance name:
   a. Select the **Filter by Perfmon Instance Name** check box on the Perfmon Information page.
   b. In the **Perfmon Instance Name** field, type the name of the instance to be filtered, or click **Browse** to list the instances available.
   c. To browse a remote system, either select one from the list, or click **Add** to add the host name of a Windows system. After you select a host, provide an Administrator ID and password. The table is updated with the list of instances on the remote system.

   Note: You can also filter by attribute group, see step 9
6. If the selected Windows Performance Monitor Object is to return multiple instances, and you want the instance name to be returned, select **Return Instance Name** on the Perfmon Information page. Checking this option adds an attribute to the data source that is not shown in the list of attributes. This attribute contains the instance name.

   Note: If you browsed for the selected object, and that object is defined as having multiple instances, this check box is selected automatically.
7. If you did not check the option to return the instance name, the Select key attributes page opens. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 26).
8. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing Perfmon attribute groups" on page 118
9. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see step"Filtering attribute groups" on page 67

   Note: You can also filter by instance name, see 5
10. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

    The Agent Editor Data Source Definition page shows a list that contains the object (Figure 82 on page 118) and information about the object.

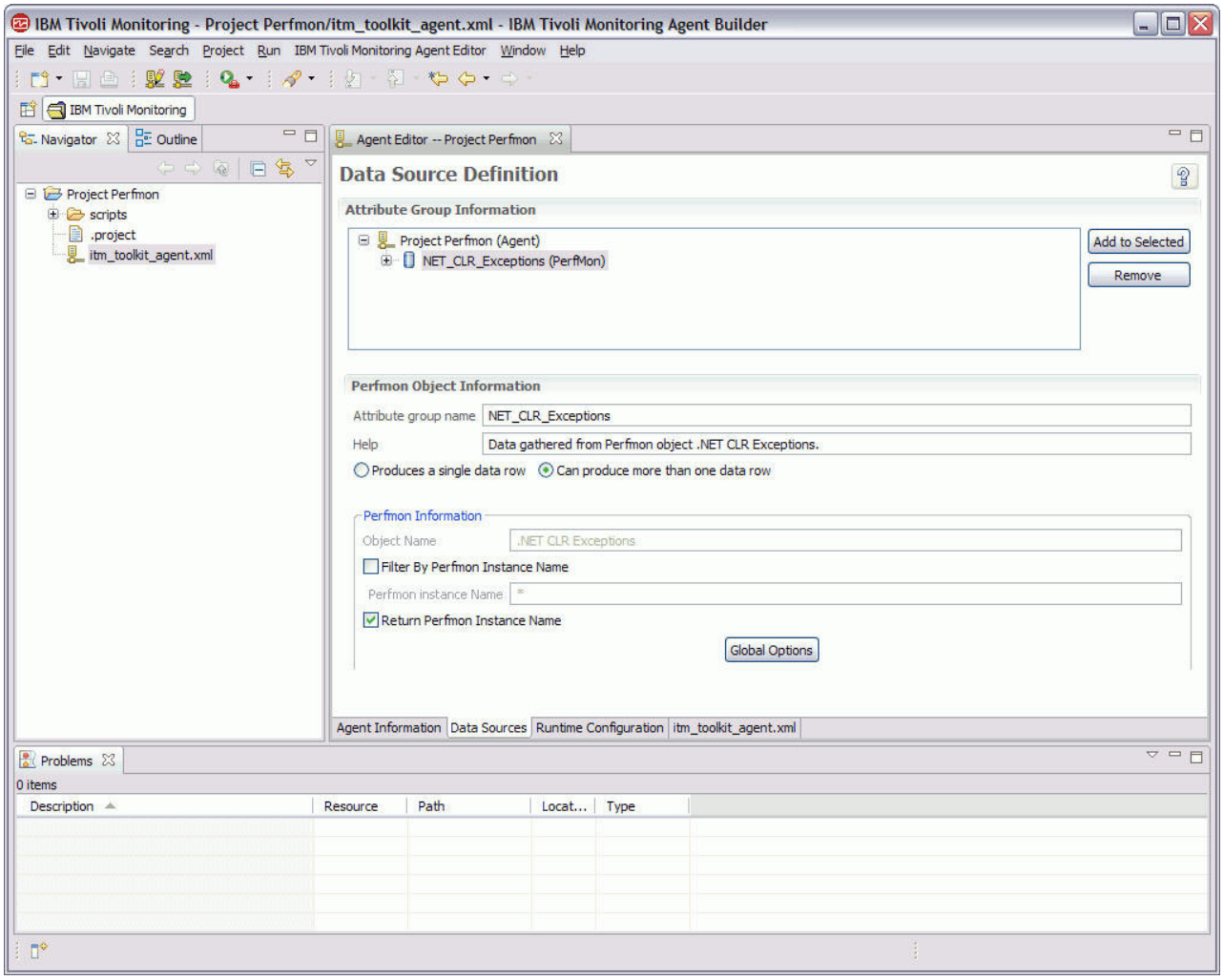


Figure 82. Perfmon Data Source Definition

11. You can add attributes and supply the information for them. For more information, see ("Creating attributes" on page 56).

    In addition to the fields applicable to all data sources, the Perfmon Attribute Information page for the data source has the following field:

    Metric name
    Name of the counter for the specific object.

What to do next

For information about Windows remote connection configuration for Perfmon data sources, see ("Configuring a Windows remote connection" on page 372).

Testing Perfmon attribute groups

If you are running the Agent Builder on a Windows system, You can test the Perfmon attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the Perfmon Information page (Figure 80 on page 116).

- After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see (Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35).

After you click **Test** in one of the previous two steps, the Perfmon Test window Figure 83 is displayed.

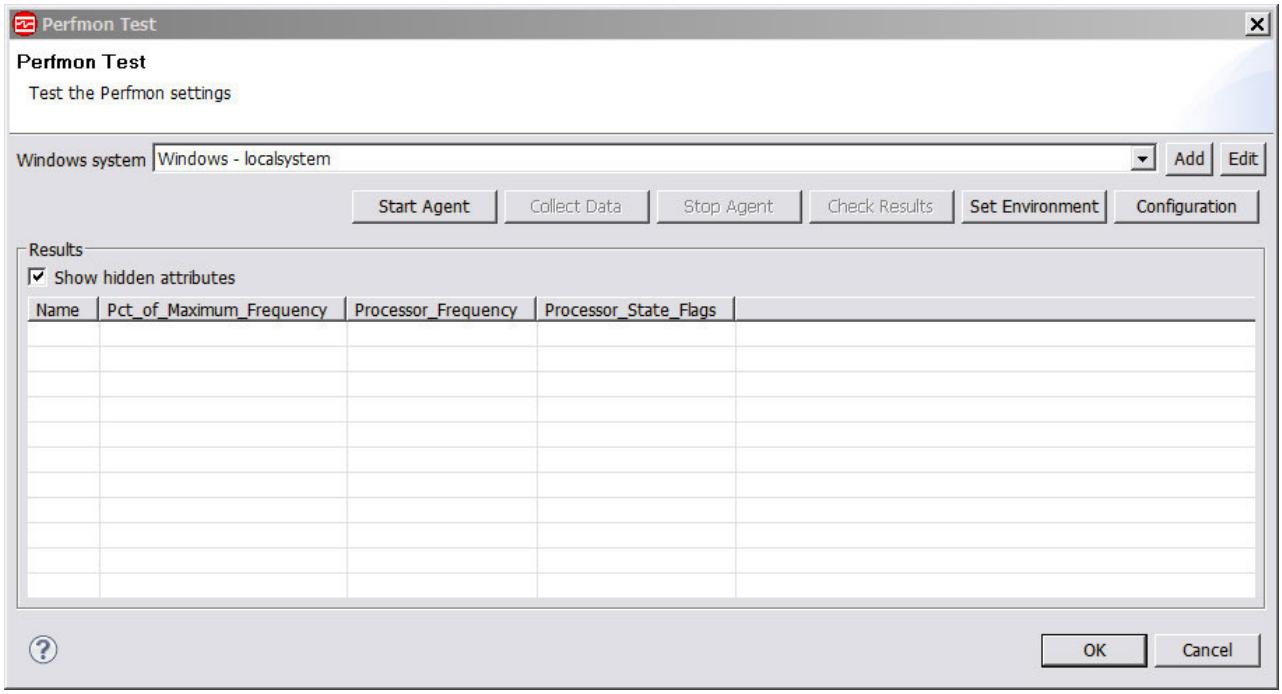


Figure 83. Perfmon Test window

2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see (“Attribute group testing” on page 377).
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries Performance Monitor for data. The Perfmon Test window collects and shows any data in the agent's cache since it was last started.

   Note: You might not see useful data for all attributes until you click **Collect Data** a second time. The reason is that some Performance Monitor attributes return delta values, and a previous value is required to calculate a delta value.
5. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and shown by the Data Collection Status window is described in (“Performance Object Status node” on page 545).
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the Perfmon Test window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, “Testing and debugging your agent,” on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 12. Monitoring data from a Simple Network Management Protocol (SNMP)

Define a data source to collect SNMP data.

About this task

Simple Network Management Protocol V1, V2C (note that the version is V2C and not just V2), and V3 are supported by agents.

Procedure

1. On the Agent Initial Data Source page (Figure 84 on page 122) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **SNMP**.

© Copyright IBM Corp. 2007, 2013

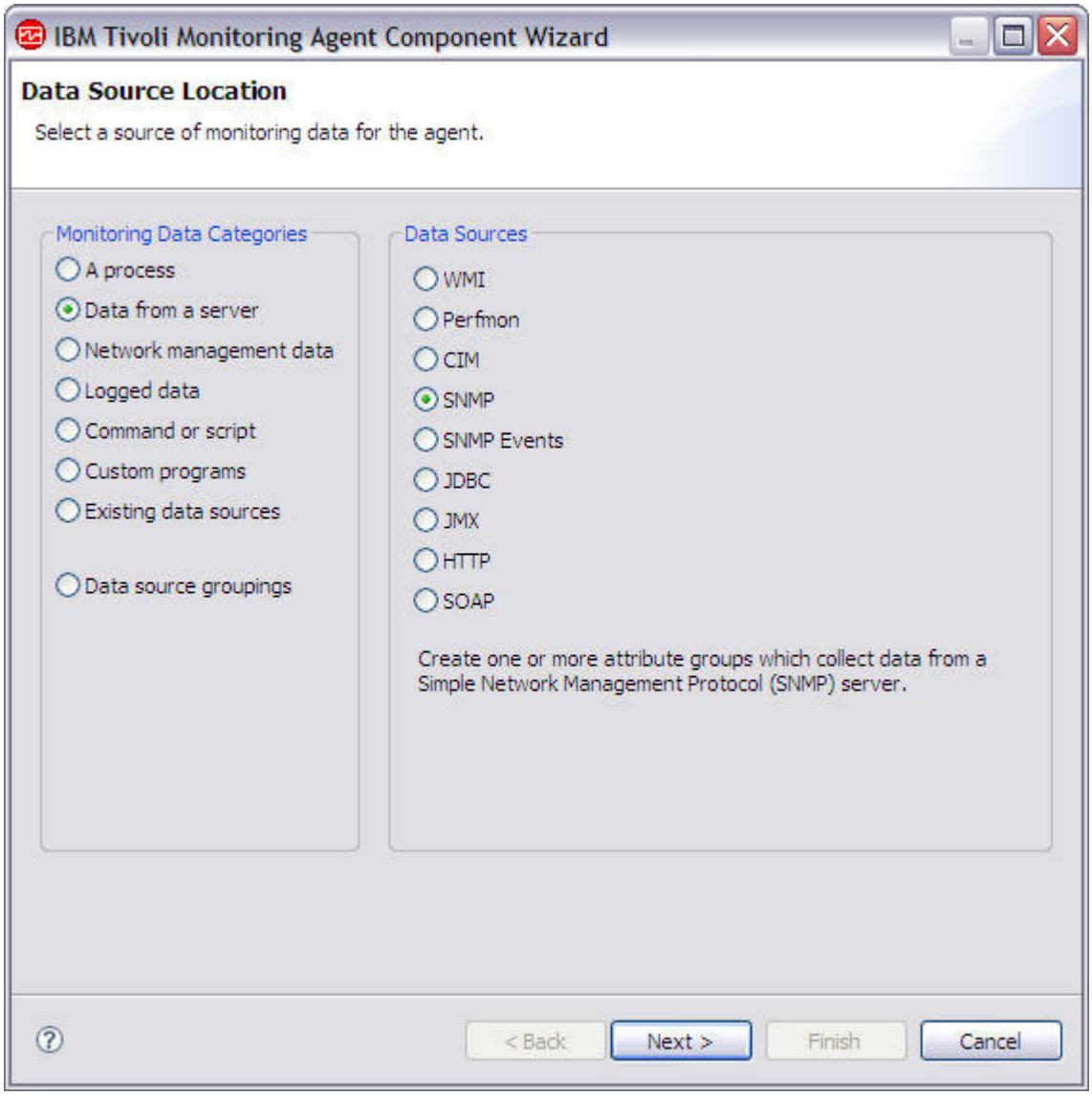


Figure 84. Adding SNMP data

3. Click **Next**.

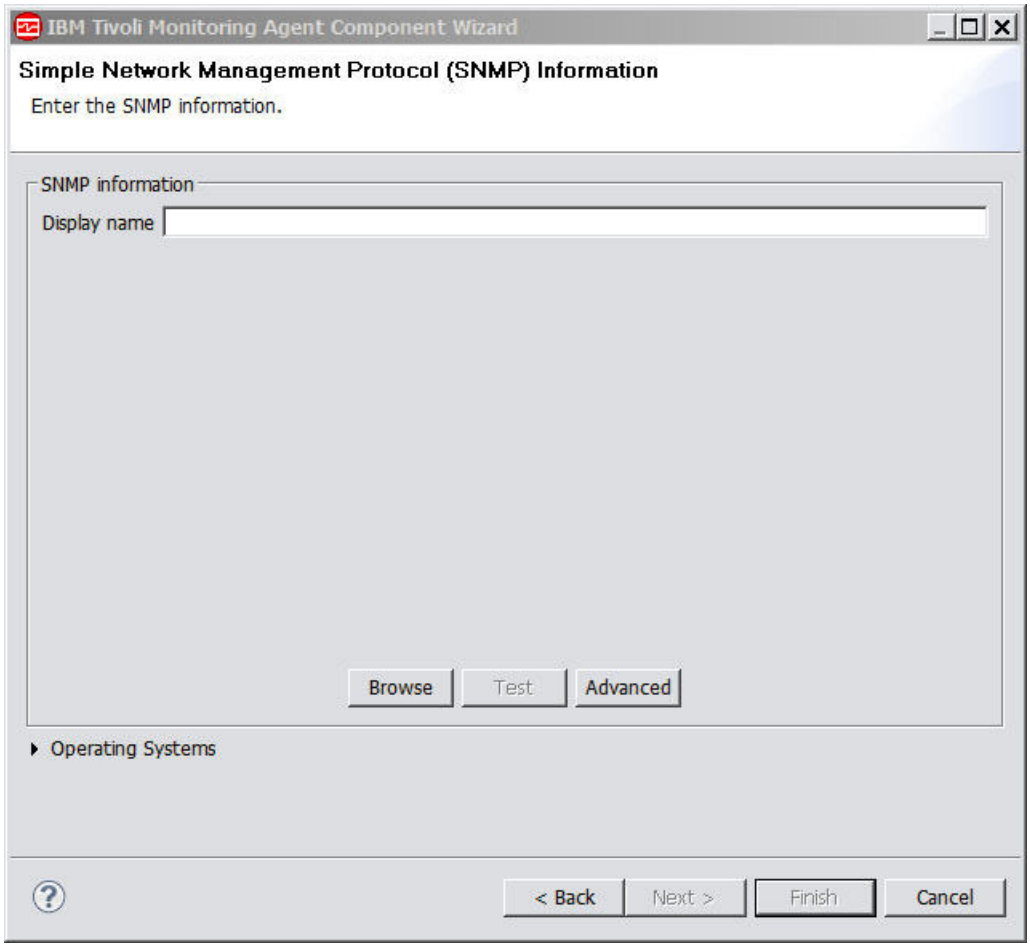


Figure 85. Simple Network Management Protocol Information page

4. On the Simple Network Management Protocol (SNMP) Information page (Figure 85), type the display name or click **Browse** to see all of the objects on the system (Figure 86 on page 124). After you define the data source, you can add an attribute. The OIDs for these attributes can be long and difficult to type correctly. Using the Browse option is an easy way to input the correct OID.

   Note: The browser does not browse the live system, it reads definitions, Management Information Bases (MIBs).

   Note: Clicking the **Refresh** icon clears the in-memory version of the parsed MIB files and reparses the files in the workspace cache. The cache is in the following location: *workspace_directory*\ .metadata\.plugins\ com.ibm.tivoli.monitoring.agentkit\mibs

   Where:

   workspace_directory
   Identifies the workspace directory that you specified when you initially ran the Agent Builder, see ("Starting the Agent Builder" on page 11).

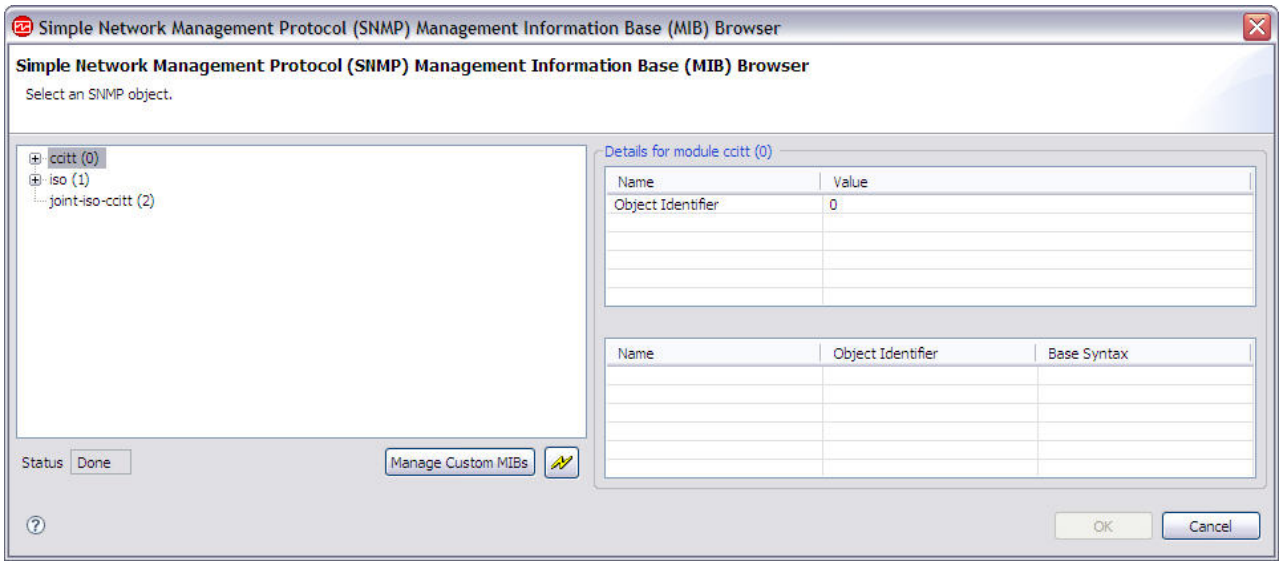


Figure 86. List of objects

a. If the MIB that defines the wanted object is not loaded, click **Manage Custom MIBs** to open the Manage Custom MIBs dialog.

b. Click **Add** to browse to the MIB file to add. To delete a MIB from the cache, select it and click **Remove**.

c. Click **OK** to update the cache. If there are any errors when the MIBs are parsed, the Manage Custom MIBs dialog remains open. This dialog gives you the opportunity to add or remove MIBs to eliminate the errors.

   Clicking **Cancel** returns the MIB cache to the state it was in when the dialog was opened.

   Agent Builder includes a set of MIBs:

   - `hostmib.mib`
   - `rfc1213.mib`
   - `rfc1243.mib`
   - `rfc1253.mib`
   - `rfc1271.mib`
   - `rfc1286.mib`
   - `rfc1289.mib`
   - `rfc1315.mib`
   - `rfc1316.mib`
   - `rfc1381.mib`
   - `rfc1382.mib`
   - `rfc1443.mib`
   - `rfc1461.mib`
   - `rfc1471.mib`
   - `rfc1493.mib`
   - `rfc1512.mib`
   - `rfc1513.mib`
   - `rfc1516.mib`
   - `rfc1525.mib`
   - `rfc1573a.mib`
   - `rfc1595.mib`
   - `rfc1650.mib`
   - `rfc1657.mib`
   - `rfc1659.mib`
   - `rfc1666.mib`

- rfc1695.mib
- rfc1747.mib
- rfc1748.mib
- rfc1757.mib
- rfc1903.mib
- rfc1907.mib
- rfc2011.mib
- rfc2021.mib
- rfc2024.mib
- rfc2051.mib
- rfc2127.mib
- rfc2128.mib
- rfc2155.mib
- rfc2206.mib
- rfc2213.mib
- rfc2232.mib
- rfc2233.mib
- rfc2238.mib
- rfc2239.mib
- rfc2320.mib
- rfc3411.mib

All of these MIBs are standard, IETF defined MIBs. The MIBs are included because they represent common definitions that can be useful in monitoring. Also, many of the MIBs are necessary so that custom MIBs can resolve the symbols that they import.

d. Select an object from the list. (Figure 87)

Click the plus sign (+) next to an object to expand and show the levels.

e. From the list, select the object that you want to specify and click **OK**.

The new data source is then listed on the Data Source Definition page (Figure 89 on page 127).

Note: If you select an object that defines other objects (objects that are nested underneath the first object), all of these objects are turned into data sources. If you select a high-level object, many data sources are added.

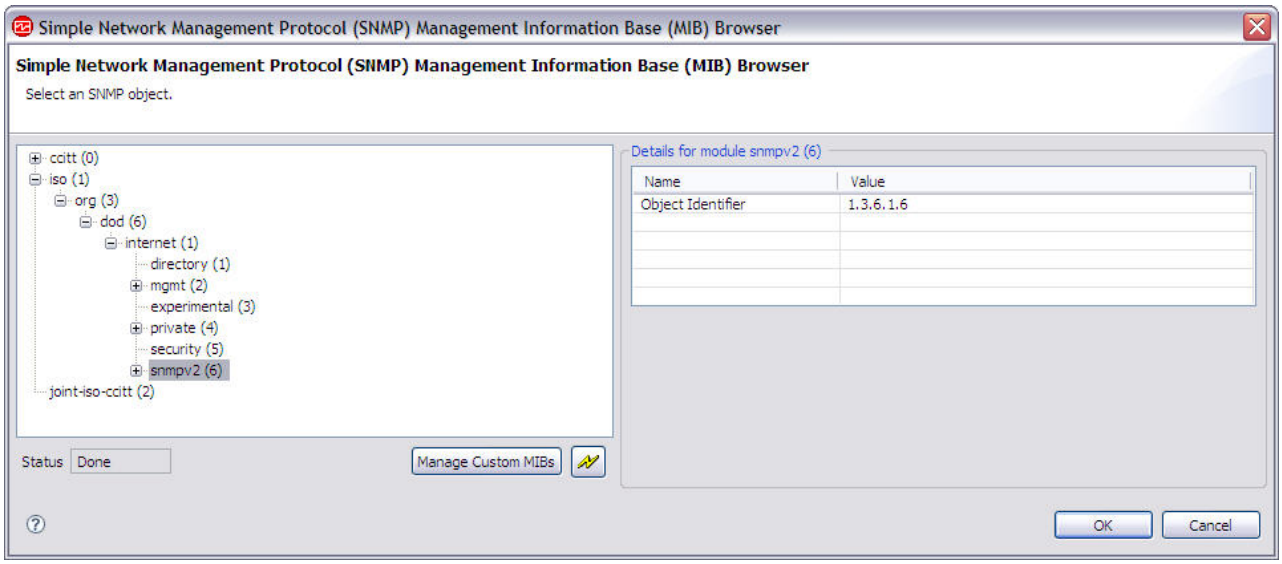


Figure 87. Selecting an object

5. On the Simple Network Management Protocol (SNMP) Information page (Figure 85 on page 123), select the operating systems.
6. Optional: You can test the data source or sources by clicking **Test** on the Simple Network Management Protocol (SNMP) Information page (Figure 88). For more information about testing, see "Testing SNMP attribute groups" on page 130

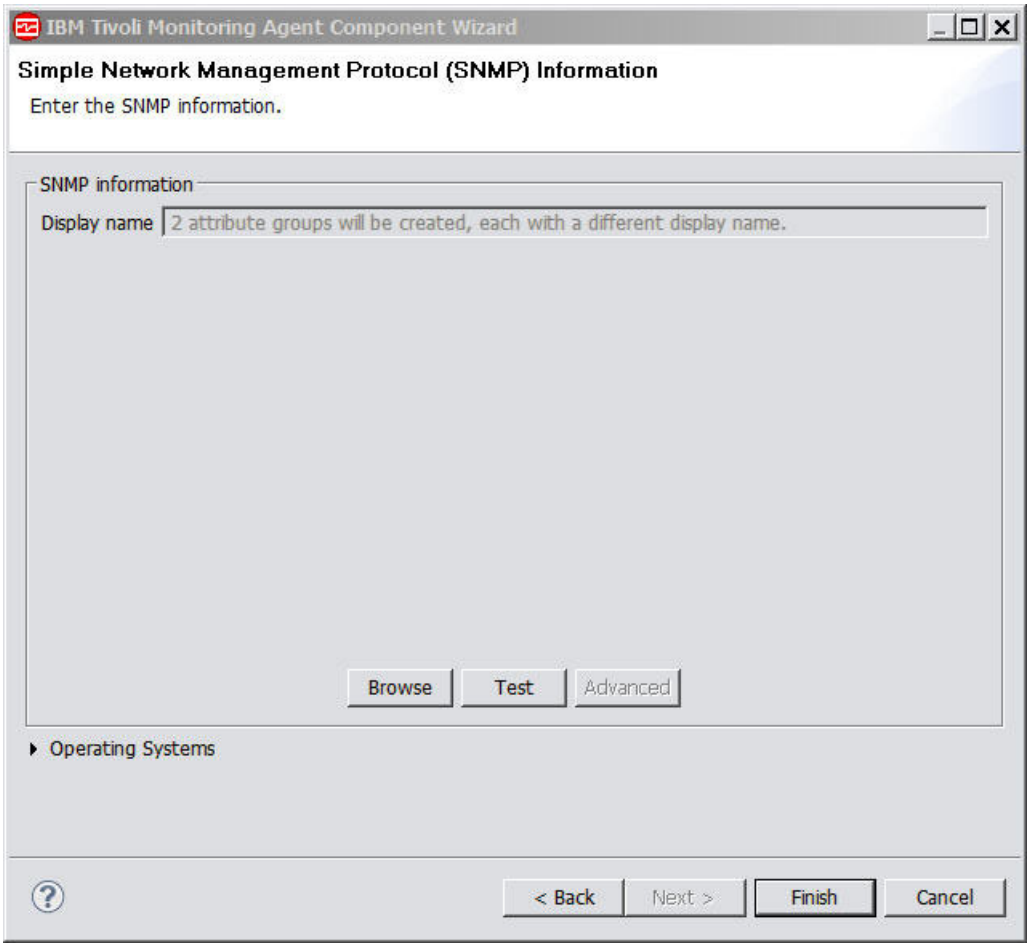


Figure 88. Simple Network Management Protocol Information page

7. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 67
8. Click **Next**.
9. On the Attribute Information page, specify the information for the attribute.
10. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

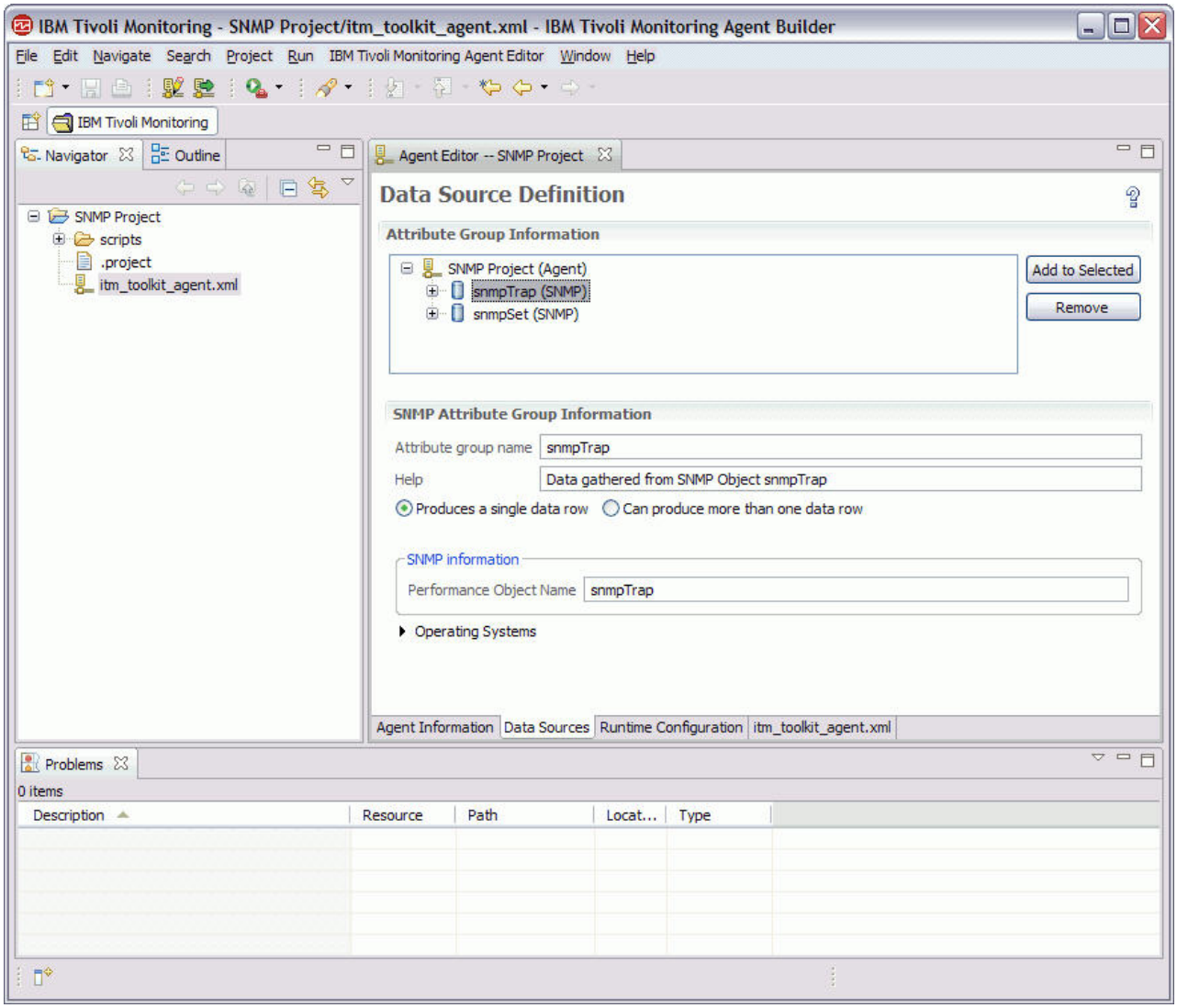


Figure 89. SNMP Data Source Definition

11. For more information about adding attributes and supplying the information for them, see ("Creating attributes" on page 56).

    In addition to fields that are applicable to all data sources, the Attribute Information page for the SNMP data source has the following fields:

    Metric name
    Arbitrary string

    Object identifier
    Full OID that is registered to the object, not including index values

What to do next

To enable the Agent Builder to generate 64-bit data types and to handle the maximum value for 32-bit unsigned MIB properties, see ("SNMP MIB Parsing options" on page 128).

SNMP MIB errors

Dealing with errors in SNMP MIBs.

It is not unusual to find errors when you are adding SNMP MIBs. Click **Details>>** in the Agent Builder Error window to see what the MIB error is.

One of the most common errors is missing definitions that are defined in other MIBs. You can import several MIBs simultaneously to resolve this problem, or you can incrementally add MIBs until all of the missing definitions are resolved. Agent Builder can use any definitions that are resolved. So you can choose to ignore an error that affects only that part of the MIB that you do not plan to use. The order of the MIBs does not matter because they are all loaded, and then the references are resolved.

As many of these errors as possible are fixed by the Agent Builder. However, some errors are so severe that they cannot be corrected. In many cases, it is possible to edit the MIB, correct the file, and then attempt to import the MIB into the Agent Builder. As MIBs and errors are discovered, IBM, customers, and IBM Business Partners add information to the A&BSM Technical Exchange Wiki. The information indicates the MIBs on which errors were discovered and the corrections that can be imported into the Agent Builder. For more information, see IBM Tivoli AA&BSM Technical Exchange

SNMP MIB Parsing options

Set your preferences for SNMP MIB parsing

Procedure

1. In the Agent Builder, select **Window** > **Preferences** to open the Preferences window.
2. In the navigation pane, expand **IBM Tivoli Monitoring Agent Builder**.
3. Click **MIB Parsing** to open the MIB Parsing window.

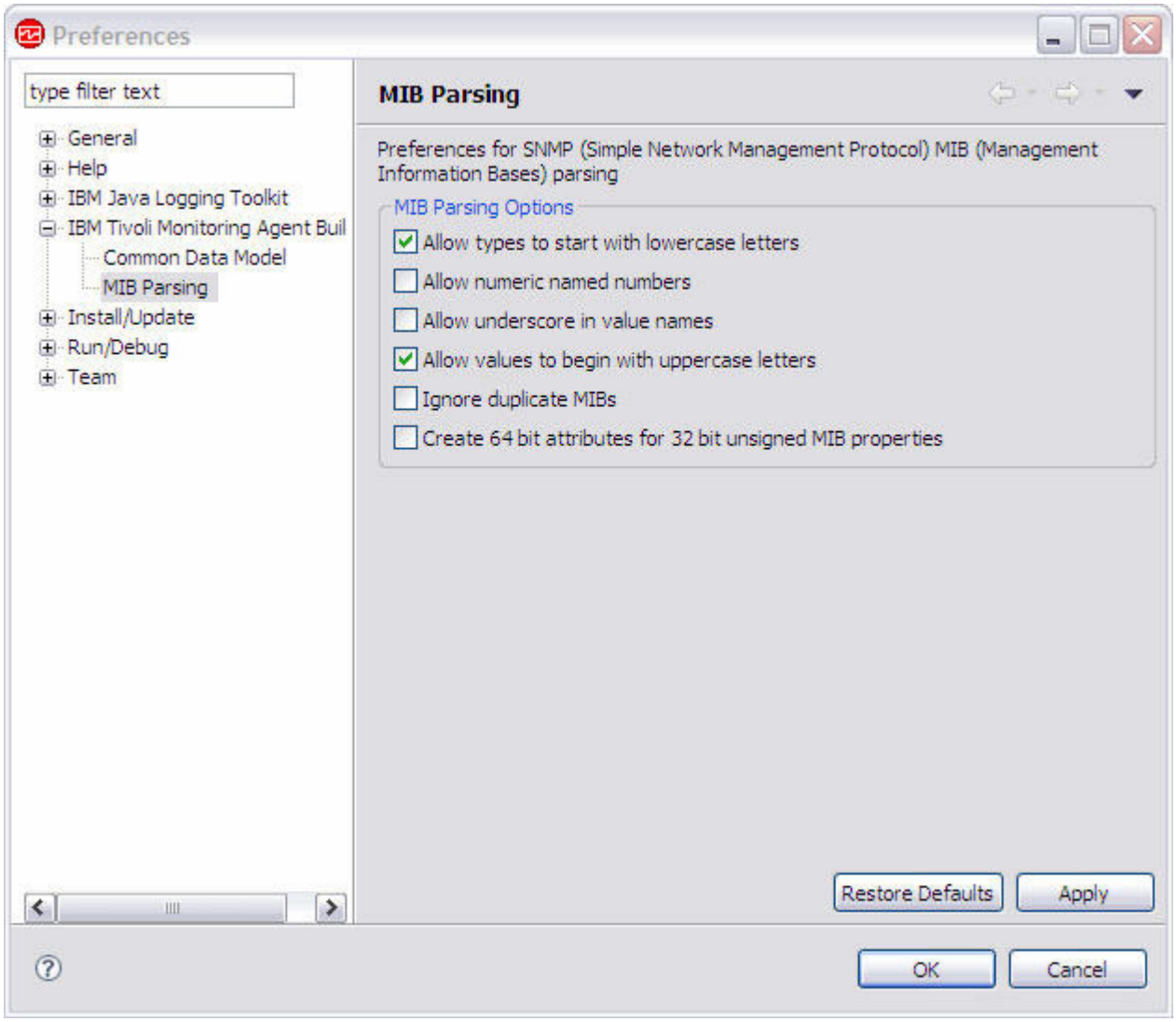


Figure 90. MIB Parsing window

The MIB parser that is used by the Agent Builder uses the grammar that is defined by ASN.1 to parse the MIBs. Some MIBs do not follow the grammar correctly. The parser can relax certain rules to accommodate the most common errors. By relaxing these rules, you can parse non-conforming MIBs.

Allow types to start with lowercase letters
Allows types that people write in MIBs, such as values

Allow numeric named numbers
Allows numbers that start with uppercase letters

Allow underscore in value name
Allows underscore characters

Allow values to begin with uppercase letters
Allows values that start with uppercase letters

Ignore duplicate MIBs
Turns off warning for duplicate MIB modules

4. Optional: Selecting the **Create 64-bit attributes for 32 bit unsigned MIB properties** check box, enables the Agent Builder to generate 64-bit data types to handle the maximum value for 32-bit unsigned MIB

properties. Selecting this option does not change any existing agent field definitions. You must browse to the MIB file to create new data sources for these properties.

5. When you are finished editing the preferences, click **OK**.

Testing SNMP attribute groups

You can test the SNMP attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the Simple Network Management Protocol Information page (Figure 88 on page 126).

     Note:

     If the SNMP object selected contains more than one attribute group, you are prompted to select the attribute group to test, on the Select Group window.

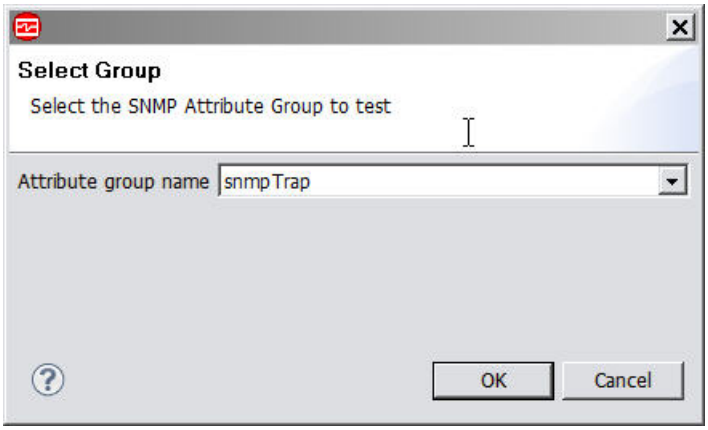


Figure 91. SNMP Attribute Group Test selection

   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35

   After you click **Test** in one of the previous two steps, the SNMP Test settings window Figure 92 on page 131 opens.

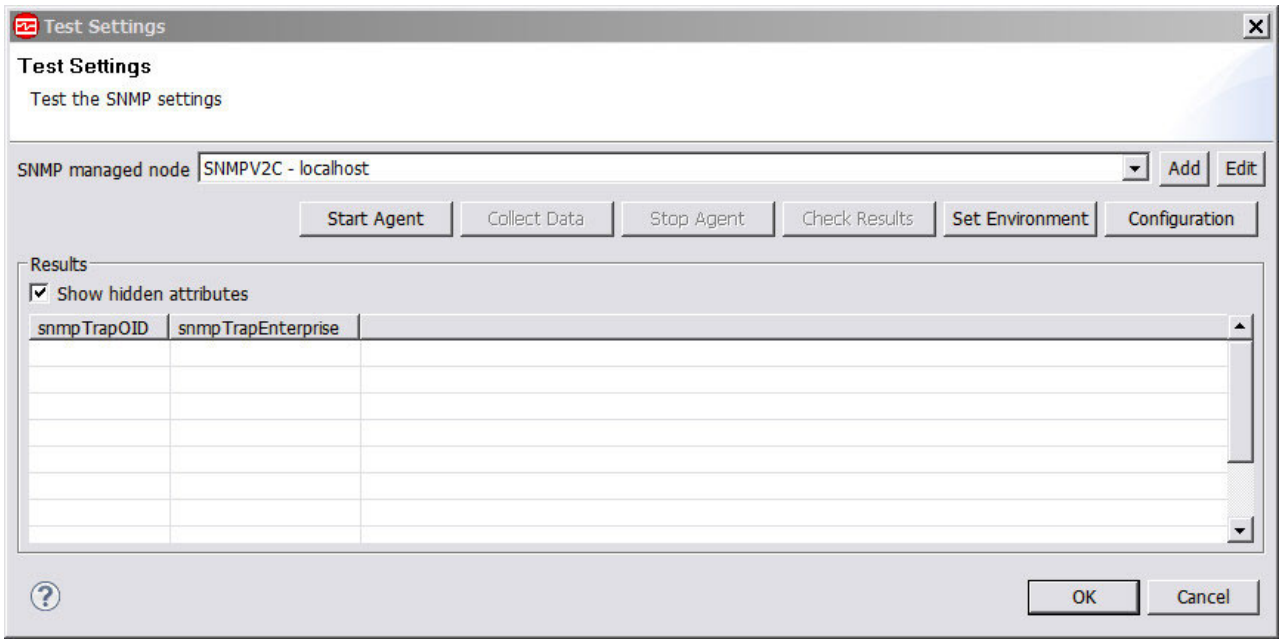


Figure 92. SNMP Test settings window

2. Select an existing connection from **Connection name** or click **Add** and you are prompted to select a connection type. Alternatively select an existing connection to use as a template, by using the Create Connection Wizard, Figure 93

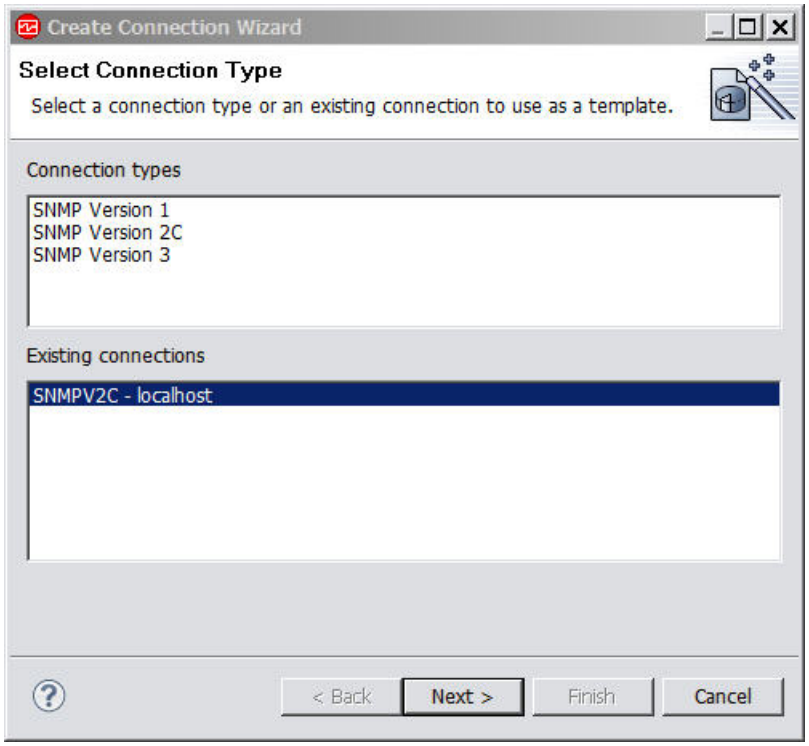


Figure 93. Create Connection Wizard

3. After you select a connection type or an existing connection, click **Next** to complete the SNMP connection properties. When complete click **Finish** to return to the SNMP Test settings window, (Figure 92).

4. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see (“Attribute group testing” on page 377).
5. Click **Start Agent**. A window indicates that the Agent is starting.
6. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries the configured SNMP connection for data.
7. The Test Settings window collects and shows any data in the agent's cache since it was last started.
8. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and shown by the Data Collection Status window is described in “Performance Object Status node” on page 545
9. Stop the agent by clicking **Stop Agent**.
10. Click **OK** or **Cancel** to exit the Test Settings window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, “Testing and debugging your agent,” on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 13. Monitoring events from a Simple Network Management Protocol event sender

Define a data source to collect SNMP event data.

About this task

Simple Network Management Protocol (SNMP) V1, V2C (note that this version name is V2C and not just V2), and V3 are supported by IBM Tivoli Monitoring agents. SNMP Traps and Informs can be received and processed by the agent. Data that is received by this provider is passed to Tivoli Monitoring as events.

For more information about the attribute groups for SNMP events, see ("SNMP Event attribute groups" on page 570).

Procedure

1. On the Agent Initial Data Source page or the Data Source Location page (Figure 94 on page 134), click **Data from a server** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

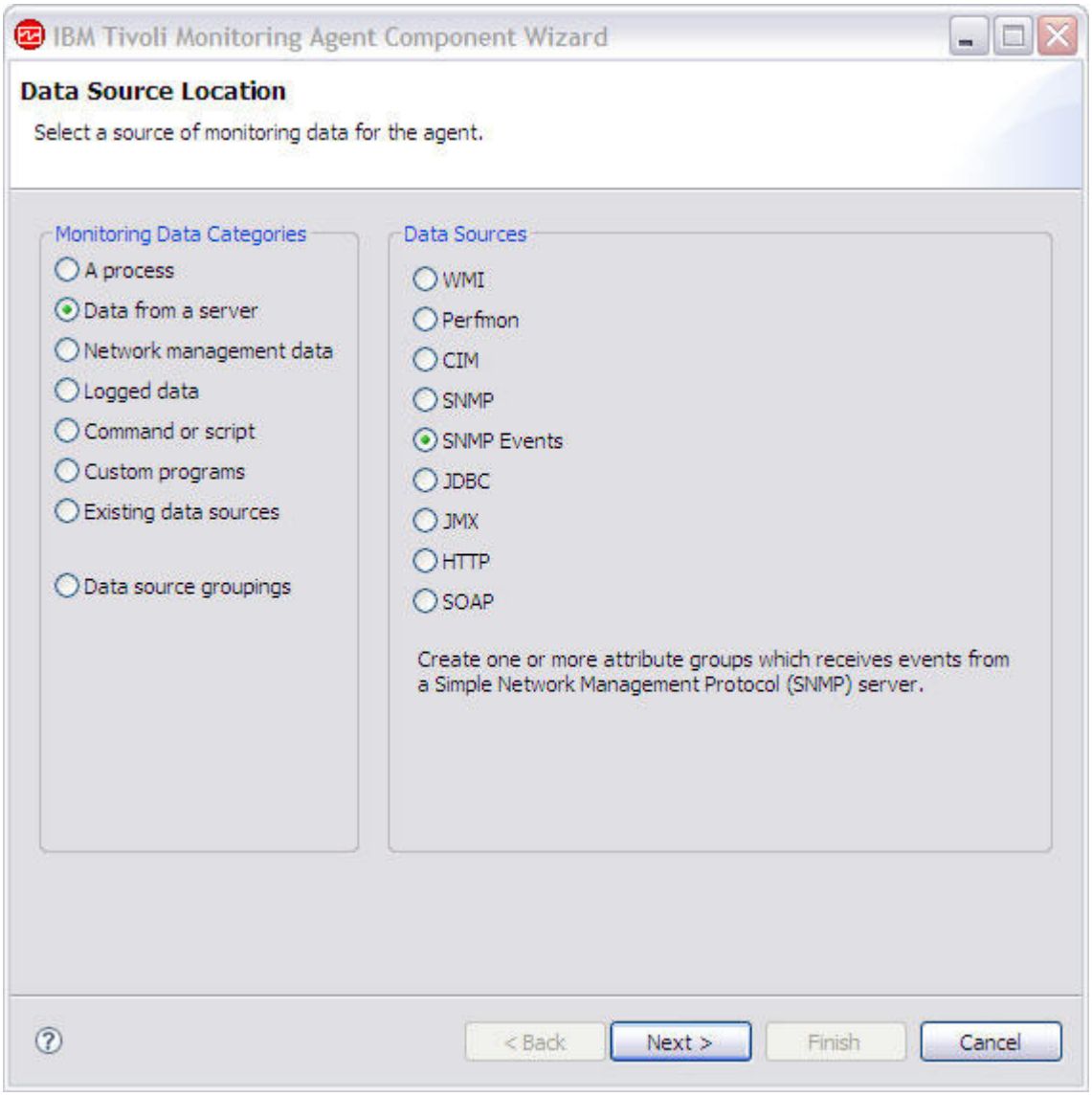


Figure 94. Adding SNMP events

2. In the **Data Sources** area, click **SNMP Events**.
3. Click **Next**.
4. In the Simple Network Management Protocol Event Information window (Figure 95 on page 135), do one of the following steps:

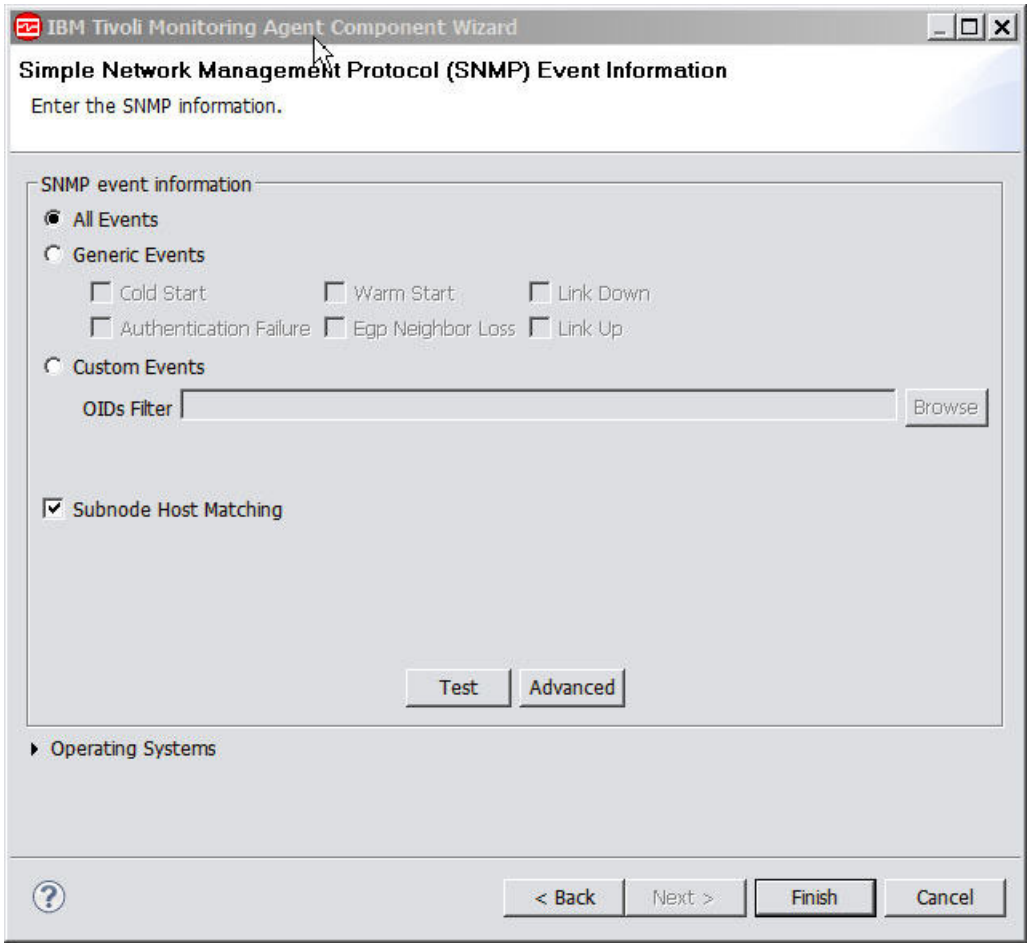


Figure 95. Simple Network Management Protocol Event Information window

- Click **All Events** to create an attribute group that sends an event for any received SNMP event.
- Click **Generic Events** to create an attribute group that sends an event for any received generic SNMP event that matches any of the selected generic event types.
- Click **Custom Events** to create one or more attribute groups that send events for enterprise-specific SNMP events. Click **Browse** to choose the events to be monitored.

  In the Simple Network Management Protocol (SNMP) Management Information Base (MIB) Browser window, the events in the selection pane are organized by the MIB module in which they were defined. Expand an SNMP object to show the events in that MIB module. In the list, click the object that you want to specify and click **OK**.

  Select the **Include attributes that show information defined in the trap configuration file** check box if you have a trap configuration file that contains static data for your traps. For more information about the SNMP trap configuration file, see (Appendix J, "SNMP trap configuration," on page 647).

  Select the **Include variable binding (VarBind) data attribute** check box if you want to include an attribute with all of the variable binding (VarBind) data that is received in the trap protocol data unit (PDU). For more information about this attribute, see the attribute definition ("SNMP Event attribute groups" on page 570).

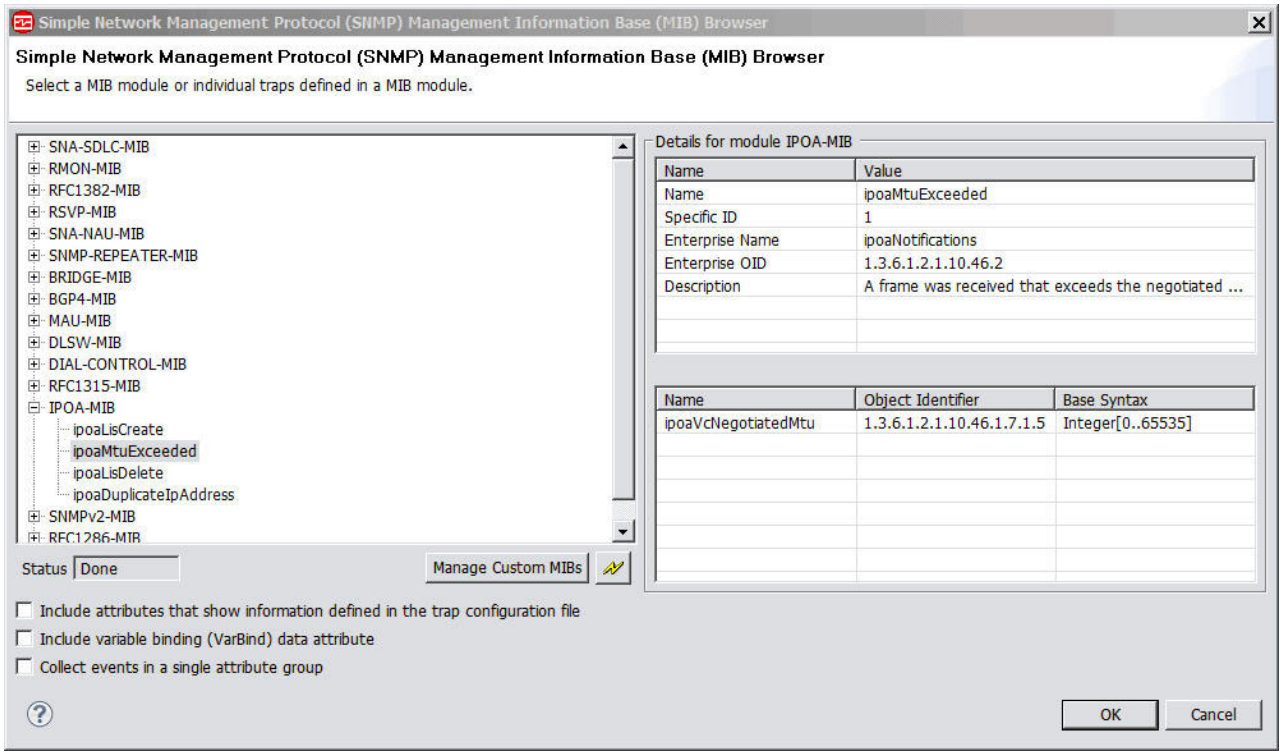


Figure 96. SNMP MIB Browser window

Note:

a. The browser does not browse the live system; it reads definitions and, Management Information Bases (MIBs). The list of MIBs included with Agent Builder is defined in (Chapter 12, “Monitoring data from a Simple Network Management Protocol (SNMP),” on page 121). MIBs loaded by either SNMP data provider are available in both.

b. If you select a MIB module or individual events, all the events in that module are converted to separate data sources. One attribute is added for each of the variables that are defined in the event. If you want all the events for the selected modules or traps to arrive in a single event source, select the **Collect events in a single attribute group** check box. If you select individual traps and the **Collect events in a single attribute group** flag is selected, one attribute is added for each of the variables that are defined in each of the events (duplicate variables are ignored). If you select a module, variable attributes are not added.

c. If you want to type your own filter, use the following syntax:

   The value of the OID (object identifier) element is used to determine which traps to process for this attribute group.

   – **Trap matching:** The OID attribute of the global_snmp_event_settings_for_group element can be a comma-delimited list of tokens. A single token has the following syntax:

     `[enterpriseOID][-specificType]`

   – **Example:** "1.2.3.5.1.4,1.2.3.4.5.6.7.8.9-0" The first token matches any trap with an enterprise OID of 1.2.3.5.1.4. The second token matches any trap with an enterprise of 1.2.3.4.5.6.7.8.9 and specific of 0. Because the tokens are listed together in one attribute group, an event received that matches either is processed by that attribute group.

d. Every event that is received is processed only by the first attribute group that matches the received event. Subnode attribute groups are processed first, and then the base attribute groups are processed. The agent developer must ensure that the groups are defined in a way so that events are received in the expected attribute group.

5. In the SNMP Event Information window (Figure 95 on page 135), select the **Subnode Host matching** check box to match events to subnodes. If the SNMP event attribute group is part of a subnode, you can select the **Subnode Host Matching** check box to control whether the event must come from the SNMP agent that is monitored.

   For example: You have an agent to monitor routers, where each subnode instance represents a specific router. You develop an agent to collect data from a router with the SNMP data collector. You also define an attribute group to receive SNMP events sent by that router. Each router instance includes the same data that is defined for the event filter. Therefore, you need another way to make sure that events from your router are shown in the attribute group for that router.

   When subnode host-matching is selected, an event that is sent by the router is compared to the host defined for the SNMP data collector. If the host in use by the SNMP data collector is the same host that sent the received event, the subnode instance processes the SNMP event. Otherwise, the event is passed to the next subnode instance. Address-matching applies only to subnodes. No address-matching is done by the SNMP event attribute groups in the base agent. For the address-matching to work, the subnode definition must contain at least one SNMP attribute group. The SNMP host that is used by SNMP for that subnode instance is the host that is used for matching.

   If the **Subnode Host Matching** check box is clear, your subnode instances do not do this extra comparison. You must allow the user to configure a different OID filter for each subnode in this case. Otherwise, you do not need to include SNMP event attribute groups in the subnode definition.
6. In the SNMP Event Information window (Figure 95 on page 135), select the operating systems.
7. Optional: You can click **Test** in the SNMP Event Information window Figure 95 on page 135 to start and test your agent. For more information, see “Testing SNMP event attribute groups” on page 140
8. Optional: In the SNMP Event Information window, click **Advanced** to select **Event Filtering and Summarization Options**. For more information, see (Chapter 34, “Event filtering and summarization,” on page 417).
   a. When you finish selecting **Event Filtering and Summarization Options**, return to the SNMP Event Information window (Figure 95 on page 135). If you previously selected **Custom Events** in the SNMP Event Information window, click **Next**, to select key attributes, otherwise skip the next step.
   b. On the Select key attributes page (Figure 97 on page 138), click one or more key attributes for the attribute group, or click **Produces a single data row**.

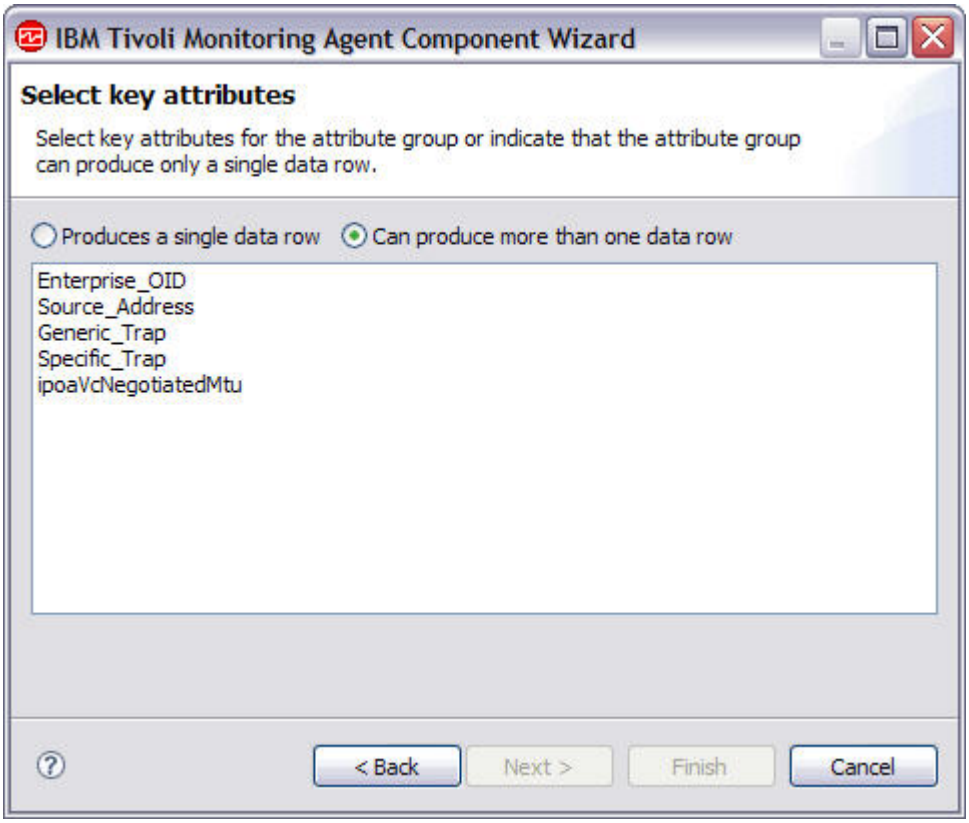


Figure 97. Select key attributes page

9. Click **Next**, or click **Finish** if you are using the new agent wizard to save the agent and open the Agent Editor.
10.

What to do next

For information about adding further attributes, see ("Creating attributes" on page 56).

SNMP Event Configuration properties

Certain configuration properties are automatically created when an SNMP Event attribute group is added to the agent

After a data source is added, the configuration is displayed on the Runtime Configuration Information page of the Agent Editor. For example, Figure 98 on page 139 shows the configuration sections and some configuration properties that are automatically created when an SNMP Event attribute group is added to the agent.

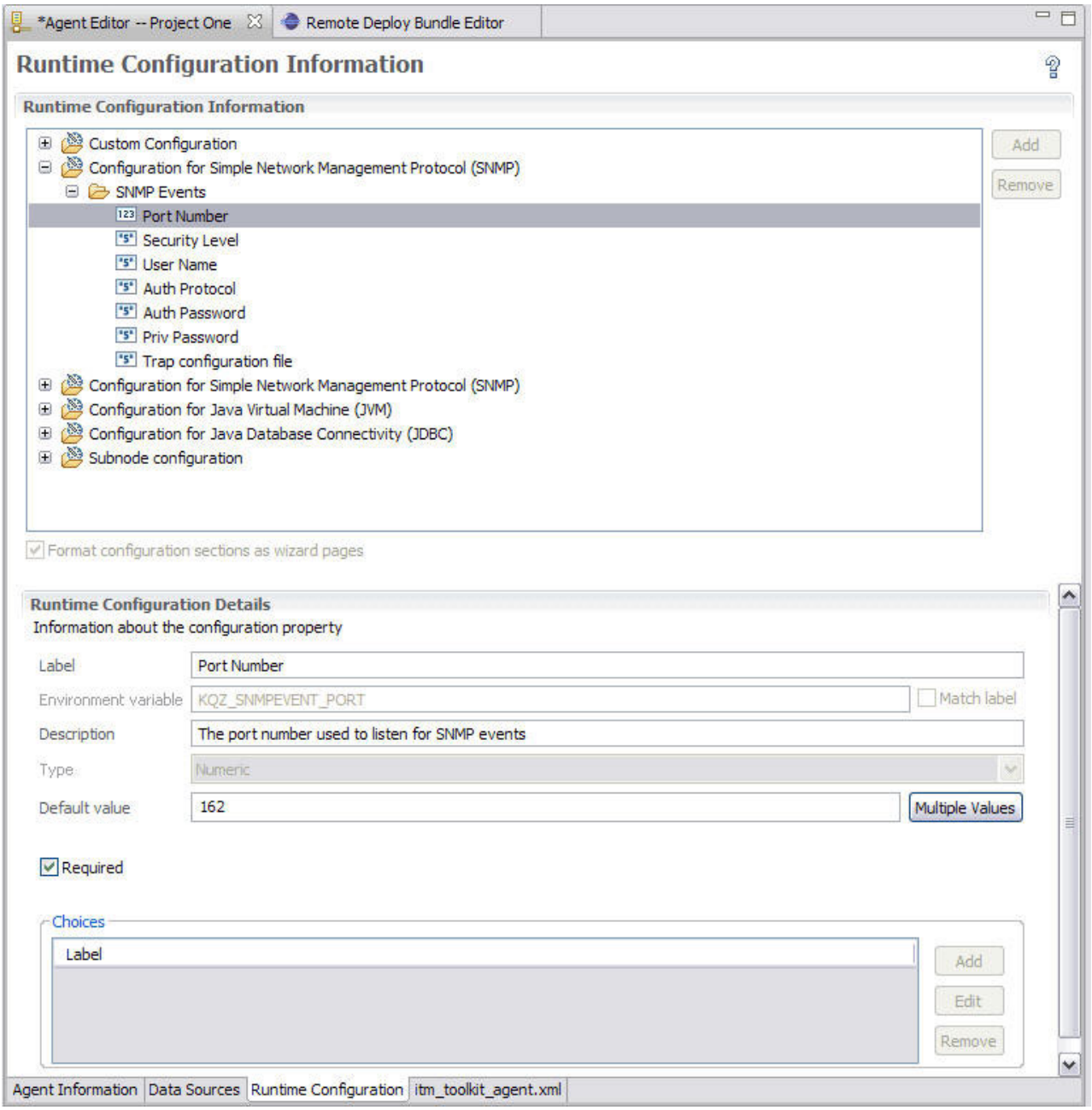


Figure 98. Runtime Configuration page

The labels, descriptions, and default values of predefined configuration properties can be changed, but variable names and types cannot be changed. The SNMP Events configuration section contains the following properties:

Table 12. SNMP Events configuration properties

Name	Valid values	Required	Description
Port Number	positive integer	Yes	Required port number that is used for listening to events
Security Level	noAuthNoPriv, authNoPriv, authPriv	No	SNMP V3 security level

Table 12. SNMP Events configuration properties (continued)

Name	Valid values	Required	Description
User Name	String	No	SNMP V3 user name
Auth Protocol	MD5 or SHA	No	SNMP V3 authentication protocol
Auth Password	String	No	SNMP V3 authentication password
Priv Password	String	No	SNMP V3 privacy password
Trap configuration file	File name that includes the path	No	Location of the trap configuration file. If the file is not located by using this configuration property, an attempt is made to find a `trapcnfg` file in the agent bin directory.

No configuration is required for V1 or V2C events. All V1 or V2C events are processed regardless of the source or community name specified. The only supported privacy protocol is DES, so there is no option to specify the privacy protocol. The SNMP V3 configuration options are not required (each can be optionally specified). If you want to specify them, you must specify the appropriate values for the security level you select.

Testing SNMP event attribute groups

You can test the SNMP event attribute group that you created, within Agent Builder.

Before you begin

To test the SNMP event attribute group, use a test program, or application to generate SNMP events.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** in the SNMP Event Information window(Figure 95 on page 135).
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35

   After you click **Test** in one of the previous two steps, the Test Event Setting window Figure 99 on page 141 opens.

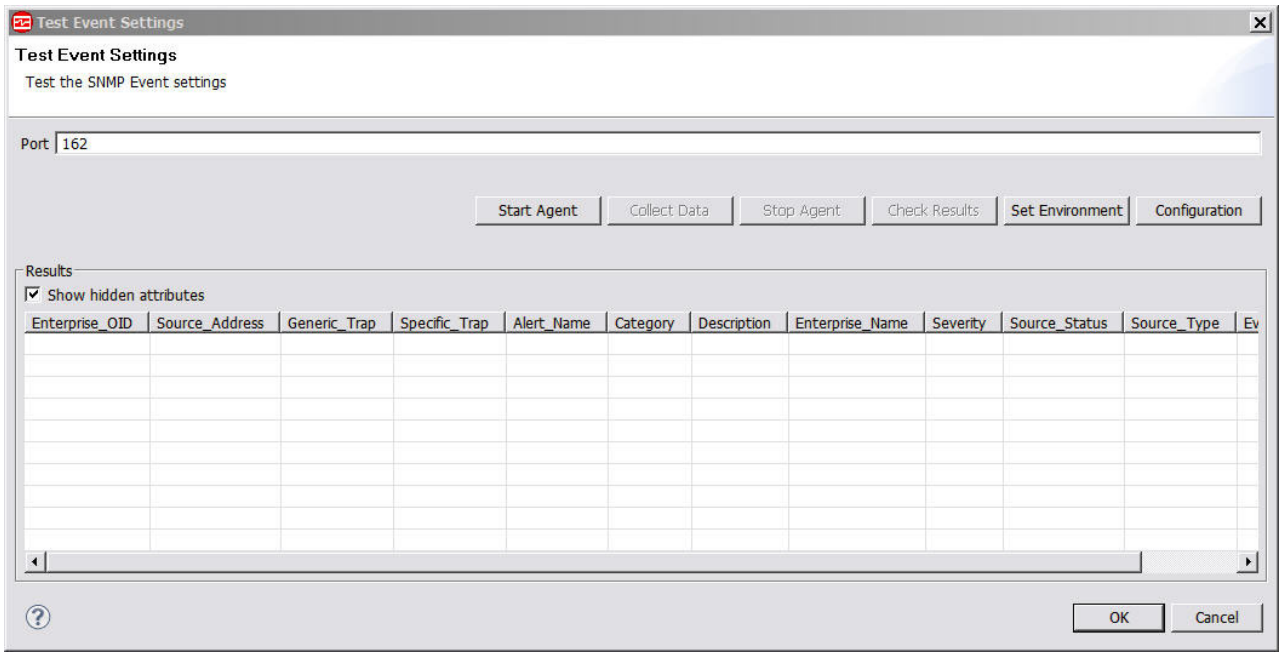


Figure 99. Test Event Settings window

2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see ("Attribute group testing" on page 377). For more about SNMP Event configuration properties, see ("SNMP Event Configuration properties" on page 138).
3. Click **Start Agent**. A window indicates that the Agent is starting. When the agent starts, it listens for SNMP events according to its configuration.

   Note: The agent that starts is a simplified version that includes the one attribute group you are testing.
4. To test your agent's data collection you generate SNMP events that match the agents configuration. You can do this using an application or an event generator. When the agent receives SNMP events that match its configuration, it adds the events to its internal cache.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The Test Event Settings window collects and shows any events in the agent's cache since it was last started. An example data collection is shown in Figure 100 on page 142

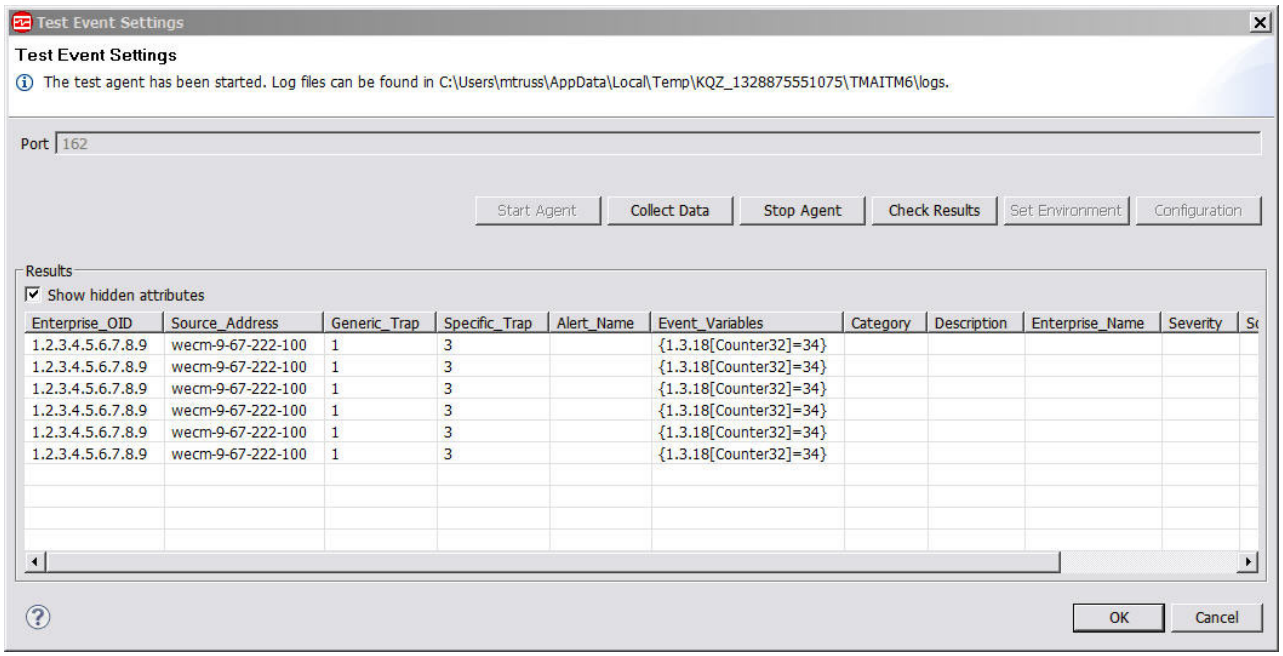


Figure 100. Test Event Settings window that shows collected SNMP event data

6. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. An example is shown in (Figure 101). The data that is collected and shown by the Data Collection Status window is described in "Performance Object Status node" on page 545

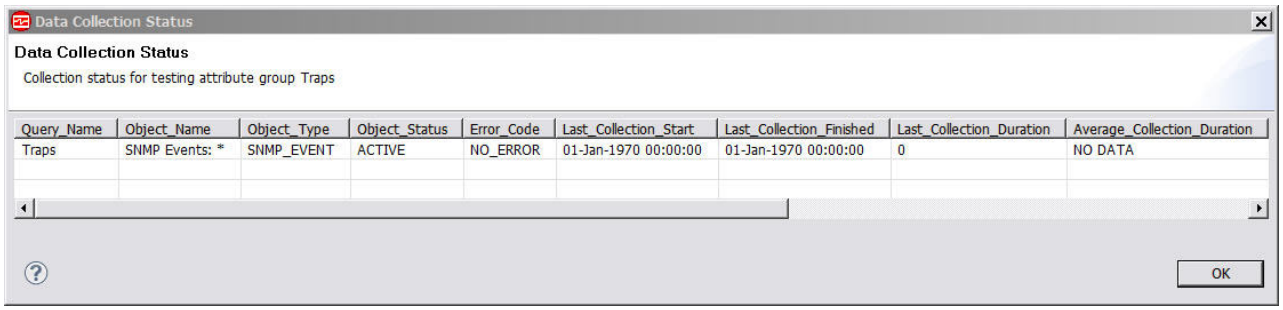


Figure 101. Data Collection Status window

7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the Test Event Settings window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, "Testing and debugging your agent," on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 14. Monitoring Java Management Extensions (JMX) MBeans

Define a data source to collect data from JMX MBeans.

About this task

Each JMX data source that you define must identify either a single MBean (single instance) or a certain type of MBean (multiple instances). You must know the Object Name of the MBean or an Object Name pattern for a MBean type that contains the data you want to collect. Use an Object Name pattern to identify only a set of similar MBeans. The set of MBeans that matches the pattern must all provide the data that you want to see in the monitoring table. A typical Object Name pattern looks like `*:j2eeType=Servlet,*`. This Object Name Pattern matches all MBeans that have a j2eeType of servlet. You can expect any MBean matching that pattern to have a similar set of exposed attributes and operations that can be added to your data source. A data source that uses that pattern collects data from each MBean matching that pattern. The attributes that you define for this data source must be available for any MBean matching the Object Name pattern of the data source.

Java Version 5 or later is supported.

Procedure

1. On the Agent Initial Data Source page (Figure 102 on page 144) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

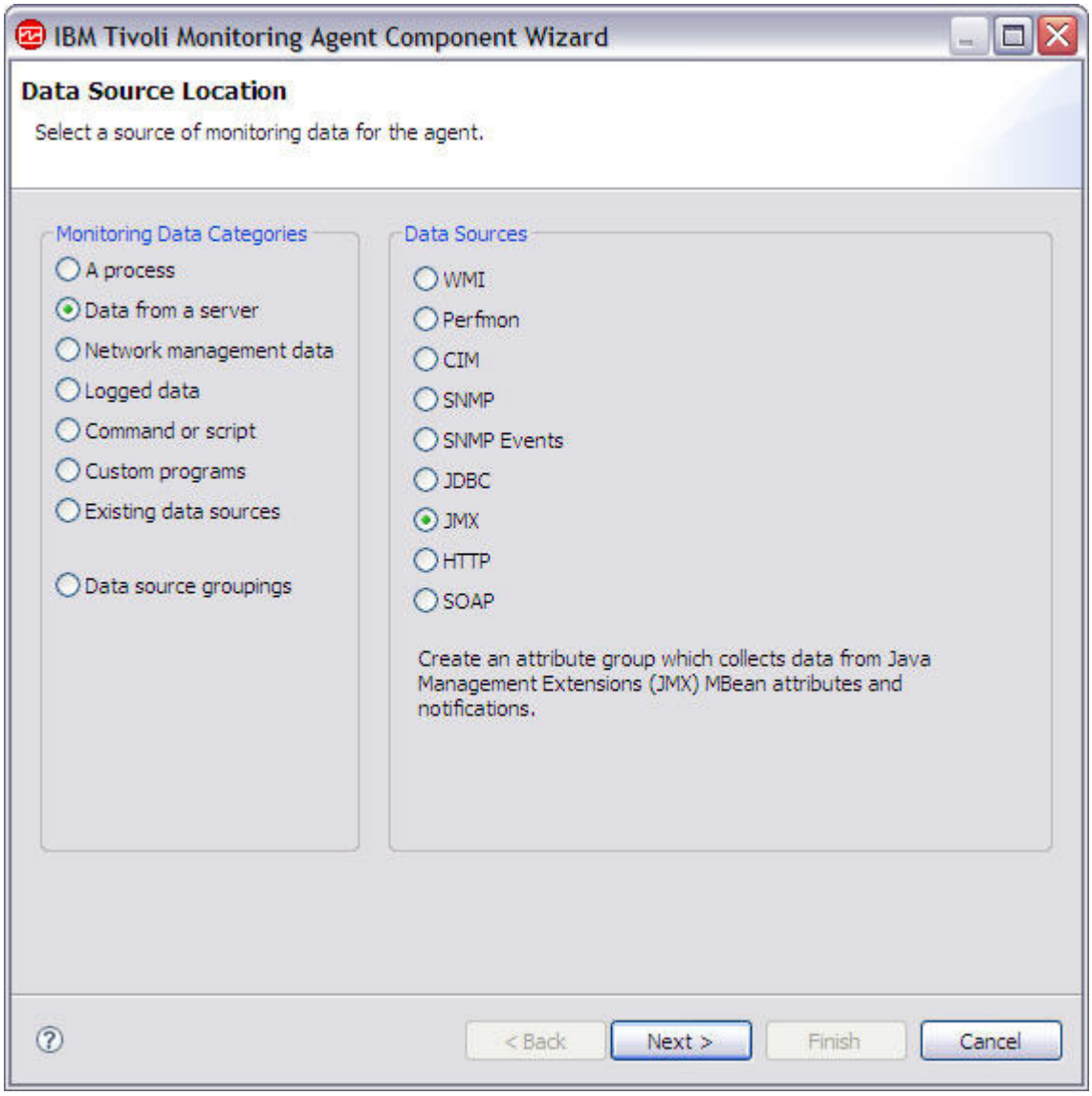


Figure 102. Adding JMX data

2. In the **Data Sources** area, click **JMX**.
3. Click **Next**.
4. On the JMX Information page (Figure 103 on page 145), click **Browse** to see all of the JMX MBeans on the MBean server.

   After you define the data source, you can use the browse function to pre-populate your attribute list. You can then add to, remove from, or modify the attributes that the browser inserted. The names for these attributes can be long and difficult to type correctly. Using the Browse option is an easy way to input the correct name.

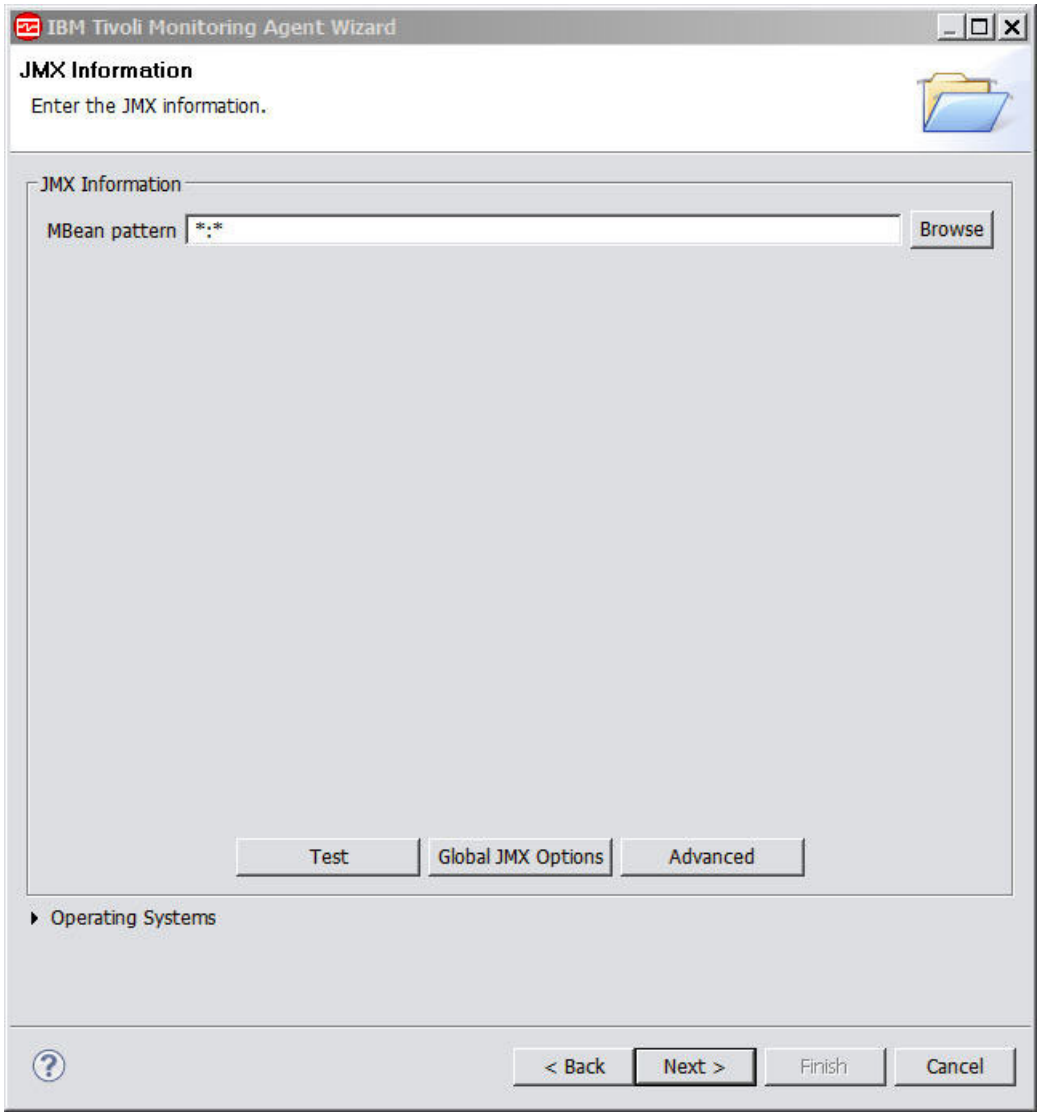


Figure 103. JMX Information page

Note: You can manually create JMX data sources by specifying an Object Name and clicking **Next** without using the browser. Manually creating JMX data sources creates two data sources. An event data source that contains predefined attributes for JMX notifications is created. Also, a collection data source is defined containing one attribute that you must specify in the wizard.

MBean pattern
Shows the MBean pattern.

Global JMX Options
Shows the level of support (Figure 104 on page 146).

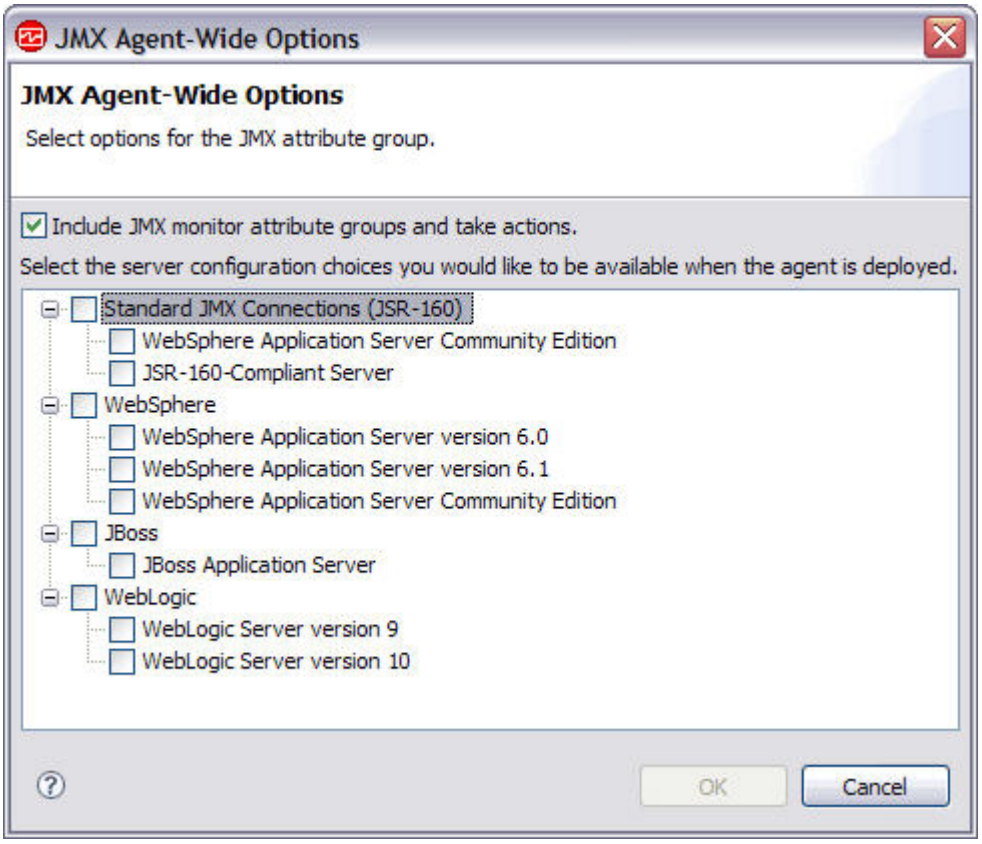


Figure 104. JMX Agent-Wide options window

Support is provided for the following JMX servers:

- Java 5 operating system MBean Server. Connection is made by using the JSR-160 connector. Notifications and monitors are supported.
- WebSphere® Application Server, version 6 and later. Connectors are provided for both SOAP and RMI protocols. JMX Monitors are not supported because MBeans cannot be created by a remote agent.
- WebSphere Community Edition and other Apache Geronimo-based application servers. Connection is made through standard JSR-160 connectors. JMX notifications and monitors are supported in versions 1.1 and later.
- JBoss Application Server, version 4.0 and later. See the following link to an item in the troubleshooting appendix if you need support for JMX Monitors.
  - JMX monitors
- WebLogic Server, version 9 and newer. Connector is provided for T3 protocol.

5. The first time that you run the JMX browser, there are no items in the **Connection Name** scroll down menu. To add connections, click the plus (+) icon. You can also use the plus (+) icon to modify or delete connections that you already defined. The connection definitions are stored in the workspace, so that, when you create a connection, it is remembered. Complete the following steps to create a connection. If you already have a connection, skip to the next step.
   a. To create a connection to an MBean Server, click **Add** to add a connection or to edit an existing connection. The Java Management Extensions (JMX) Browser window is shown when no connections are defined:

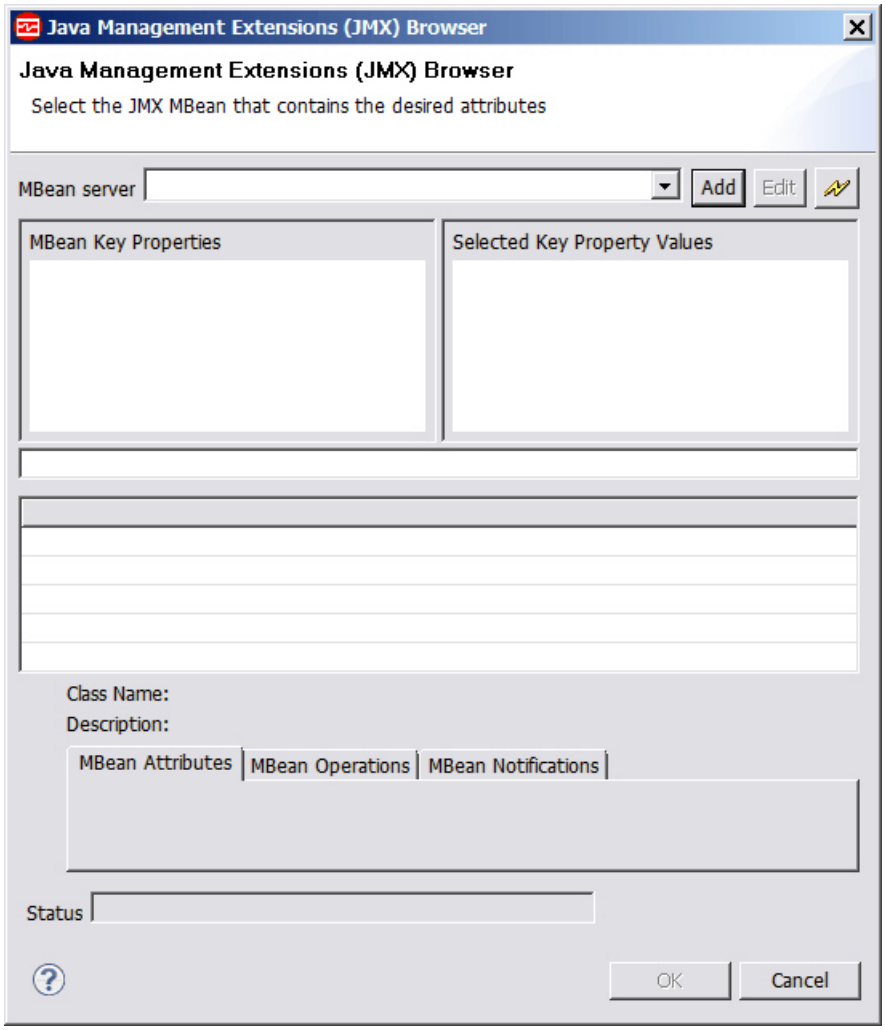


Figure 105. JMX Browser with no connection selected

b. After you click **Add**) to add a connection, the Select Connection Type page opens.

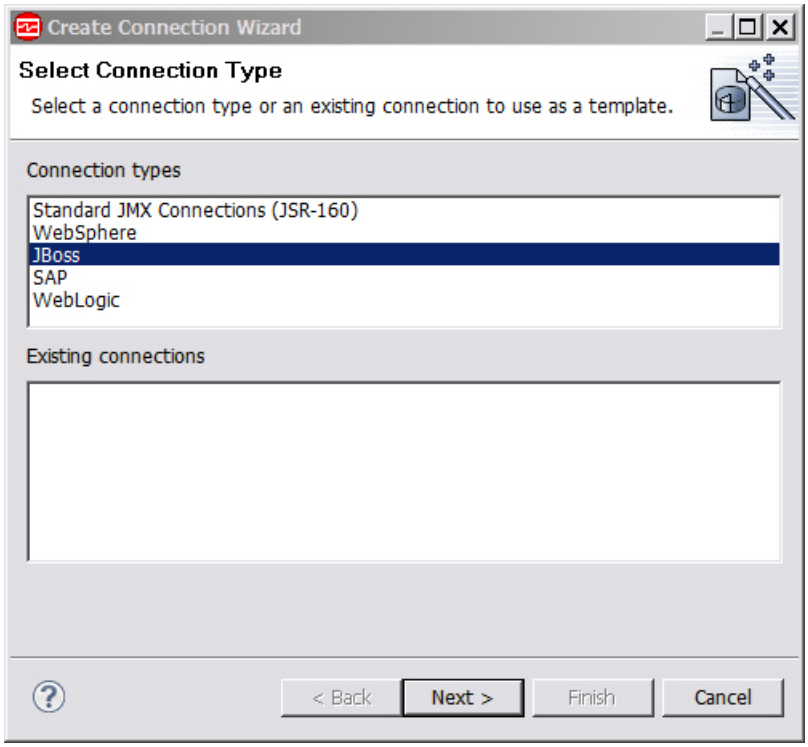


Figure 106. Connection Type page

c. Use the MBean Server Connection wizard to connect to an MBean server. The new connections that are listed on the page are selections that you can make to create connection. You can use the list of existing connections to reconfigure an existing connection. Select one of the new connection types and click **Next** to begin creating a connection.

d. After you select a connection type, you might be asked to select a more specific type of connection. Two templates that are based on the **Standard JMX Connections (JSR-160)** connection type are shown. Select the template that is most appropriate for your MBean server and click **Next**.

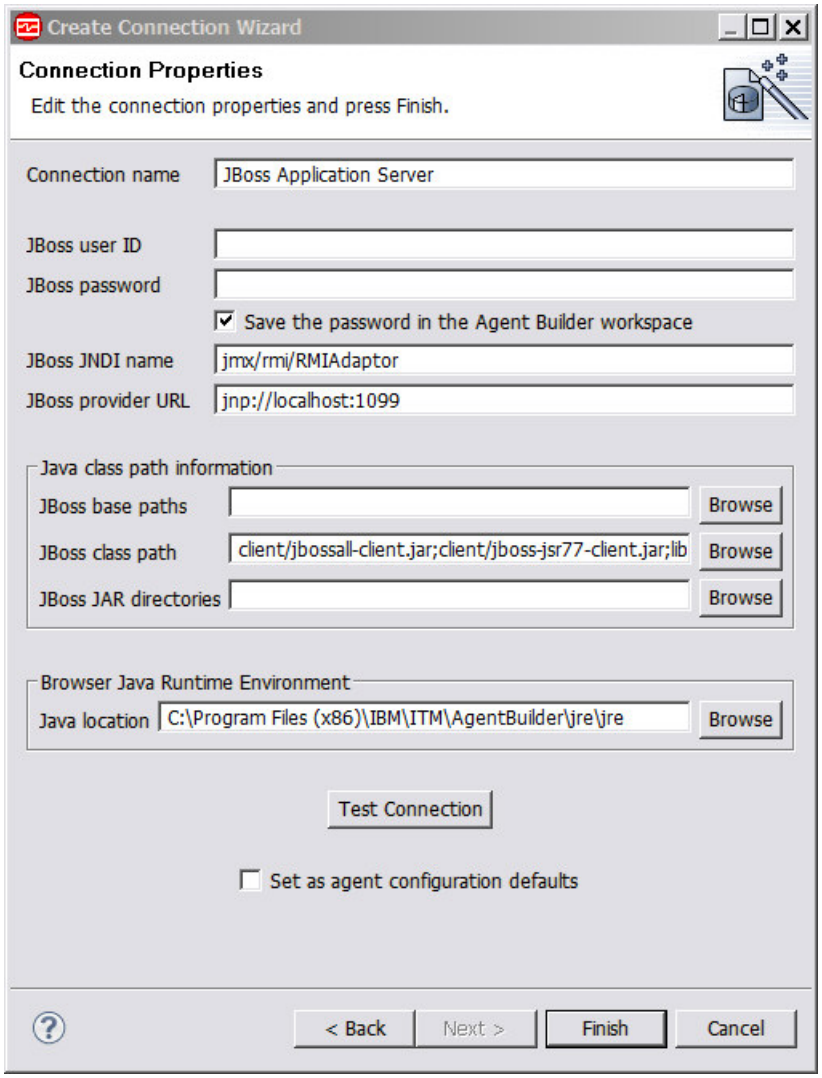


Figure 107. JMX connection properties

The Connection Properties page (Figure 107) contains the details on how to connect to an MBean server. You must complete the page with details about your MBean server.

e. Select the **Save the password in the Agent Builder workspace** check box if you want to save the password for this connection.

f. Optional: Select **Set as agent configuration defaults** if you want the defaults for JMX to be copied from these connection properties. For example, the default JBoss base path is `C:\jboss-4.0.2`.

   1) After you specify the properties that are required to connect, click **Test Connection** to ensure that the connection can be established. If the connection is not successful, correct the necessary properties.

   2) When the connection is successful, click **Finish** to return to the browser that uses the connection you configured.

The Java class path information in the Connection Properties page contains three fields. These fields must be completed as necessary to connect to an MBean server that requires Java classes that are not included in the Java runtime environment. Normally, the MBean server you want to connect to must be installed on the same system as the Agent Builder. In this case, specify the directory where the application that contains the MBean server was installed as the **Jboss base paths** field. The **JBoss Jar Directories** field then lists the directories relative to the Base Paths directory that contain the JAR files that are required to connect to the MBean server. The **JBoss class path** field can be used to

include specific JAR files. The JAR files that are listed in the **JBoss Jar Directories** field are not required to be listed separately in the **JBoss class path** field.

Any of the fields can contain more than one reference; separate the entries by a semicolon. These values are the same values that are required when you configure the agent. For more information, see ("JMX configuration" on page 156).

6. After you select a connection, the JMX Browser downloads information about the MBeans from the JMX server. This information is shown in the following four areas of the JMX Browser window (Figure 108):

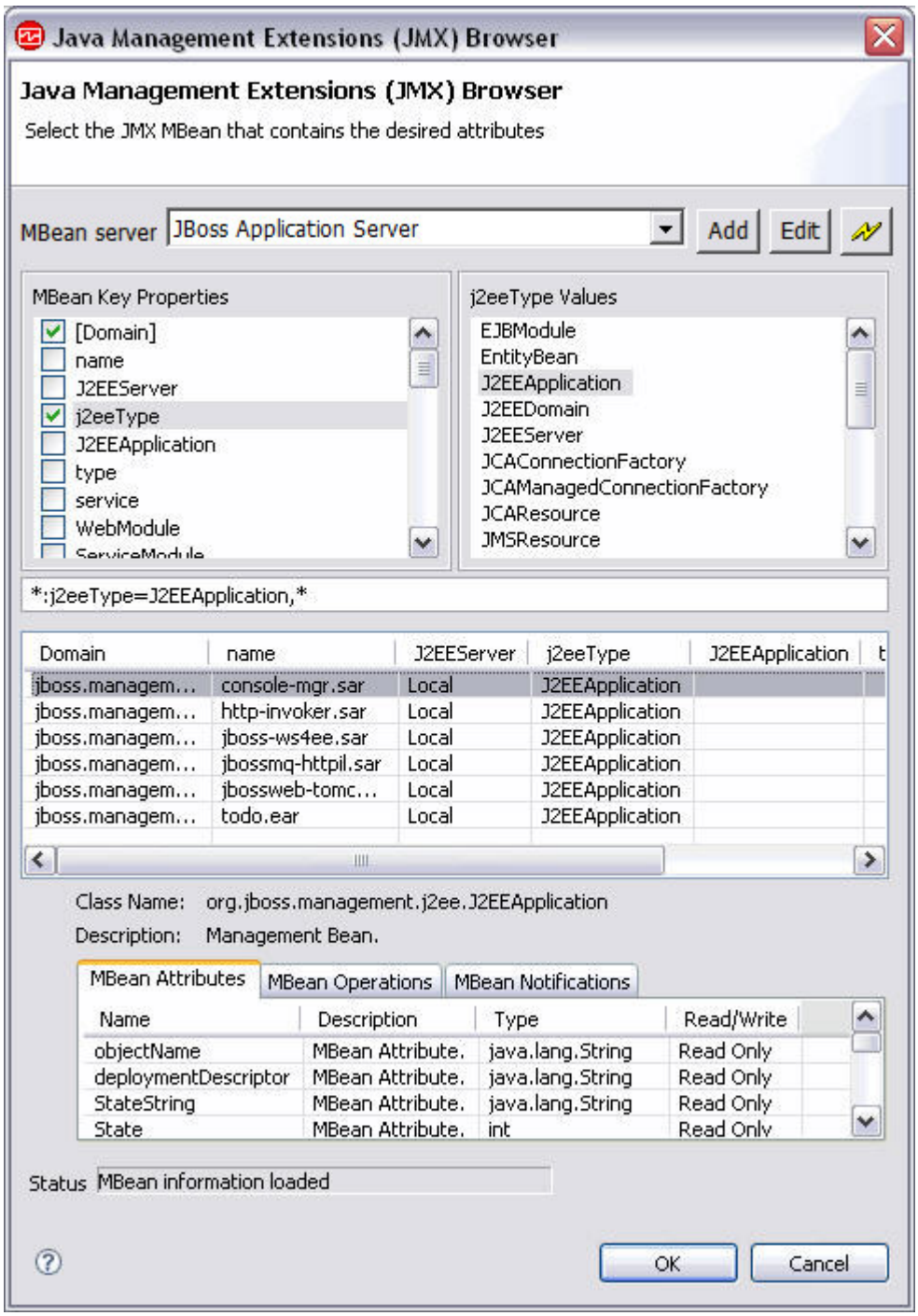


Figure 108. Java Management Extensions (JMX) Browser window

- **MBean Key Properties** area: This area is a collection of every unique Object Name key that is found from all the MBeans on the server. The **[Domain]** entry is special because it is not really a key. However, the **[Domain]** entry is treated as an implied key for the value of the MBean domain. Select an item from this list, and the MBeans that contain that key property are found. The list of values of the key property are shown in the **Selected Key Property Values** list. When you check a key property, it is included in the Object Name pattern for the data source.
- **Selected Key Property Values** area: This area shows the values of the currently selected MBean Key Property from all MBeans. Selecting one of these values checks the MBean key property. The selection also updates the Object Name Pattern shown in the message field with the MBean key property name and value.
- A table lists all MBeans matching the Object Name Pattern: As you select Key Properties and Values from the MBean Key Properties and Selected Key Property Values lists, you see the Object Name Pattern update. You also see the list of MBeans in this table change to reflect the list of MBeans that match the pattern you selected. If you have a pattern that is not matching any MBeans, you can clear entries in the MBean Key Properties list. You clear entries by clicking the check box next to a key that is being used by your pattern and removing the check mark. Also, you can manually edit the pattern to find the MBeans you are looking for. The pattern *:* selects all MBeans.

  You can use this table to browse the MBeans from the server and decide which ones contain the data you want to monitor. To help browse a potentially large number of MBeans, you can sort by any key attribute (from the menu or by clicking a column header). You can also show any key attribute in any column by selecting **Show Key Property** from the menu. When you see a key property value in the table that identifies MBeans you want to monitor, right-click on that value and choose **Select only MBeans with Key Property** from the menu.
- A table that contains details for a selected MBean: The JMX Browser shows you information about a single MBean. To see details for an MBean, you select the MBean from the table that shows the list of MBeans matching the current filter. The key information about the MBean is the list of Attributes, Operations, and Notifications it defines.

To create a data source from the JMX Browser, use the four panels that were described earlier to build an Object Name Pattern. Build the Object Name Pattern to match a set of MBeans that each contains the monitoring data you want to collect. For instance, if you wanted to monitor data from all of the ThreadPool MBeans, use the following steps:

a. Select **type** from the MBean Key Properties panel. Selecting **type** causes the values in the Selected Key Properties Values to be updated to list all unique values from the type key of any MBean.
b. Select **ThreadPool** from the list of values for the type key. After you select **ThreadPool**, the type key property name is selected in the MBean Key Properties panel and the Object Name Pattern is updated to *:type=ThreadPool,*. The list of MBeans is also updated to show only the MBeans that match this pattern.
c. Select one of the MBeans from the MBean list to see the attributes, operations, and notifications available for the MBean. If your MBean list contains more MBeans than you want to monitor, you must continue this procedure of selecting key properties and values. Continue until you have the Object Name Pattern that identifies the set of MBeans you want to monitor. You can also open a menu in the MBean list to update the Object Pattern with key property values shown in the table.

7. When the object name pattern is correct, select an MBean from the table. All attributes of the selected MBean are the initial attributes in the new JMX data source. Some attributes might not contain data. After the JMX data source is created, review the attributes and remove any that are not significant. If the selected MBean has no attributes, you are warned that the data source is created with no attributes. If the selected MBean contains notifications, an Event data source is also created to receive notifications from the MBeans.
8. Click **Finish** in the filled on JMX Information page. Data sources are created based on the MBean that was selected in the previous step. If no MBean was selected, an attribute group with no

attributes is created. A warning is shown, giving you a chance to select an MBean. The notification data source has the word, **Event**, at the beginning of the data source name to distinguish it from the data source that shows attributes.

9. To change other JMX options for the agent, click **Global JMX Options**. With these options, you can:
   a. Choose whether JMX monitors are supported by this agent. If you want JMX monitor attribute groups and Take Action commands to be created, select **Include JMX monitor attribute groups and take actions** See the next section for a description of JMX monitors.
   b. Select the types of MBean servers your agent connects to when it is deployed. There are several vendor-specific types of servers that are listed, along with a generic JSR-160-Compliant Server for standards-based servers. You can select as many as needed, but you must select only server types that support the MBeans that are being monitored. You must select at least one. If you select more than one, at agent configuration time you are prompted to specify which type of server you want to connect to.
10. Click **OK** after you select the wanted option.

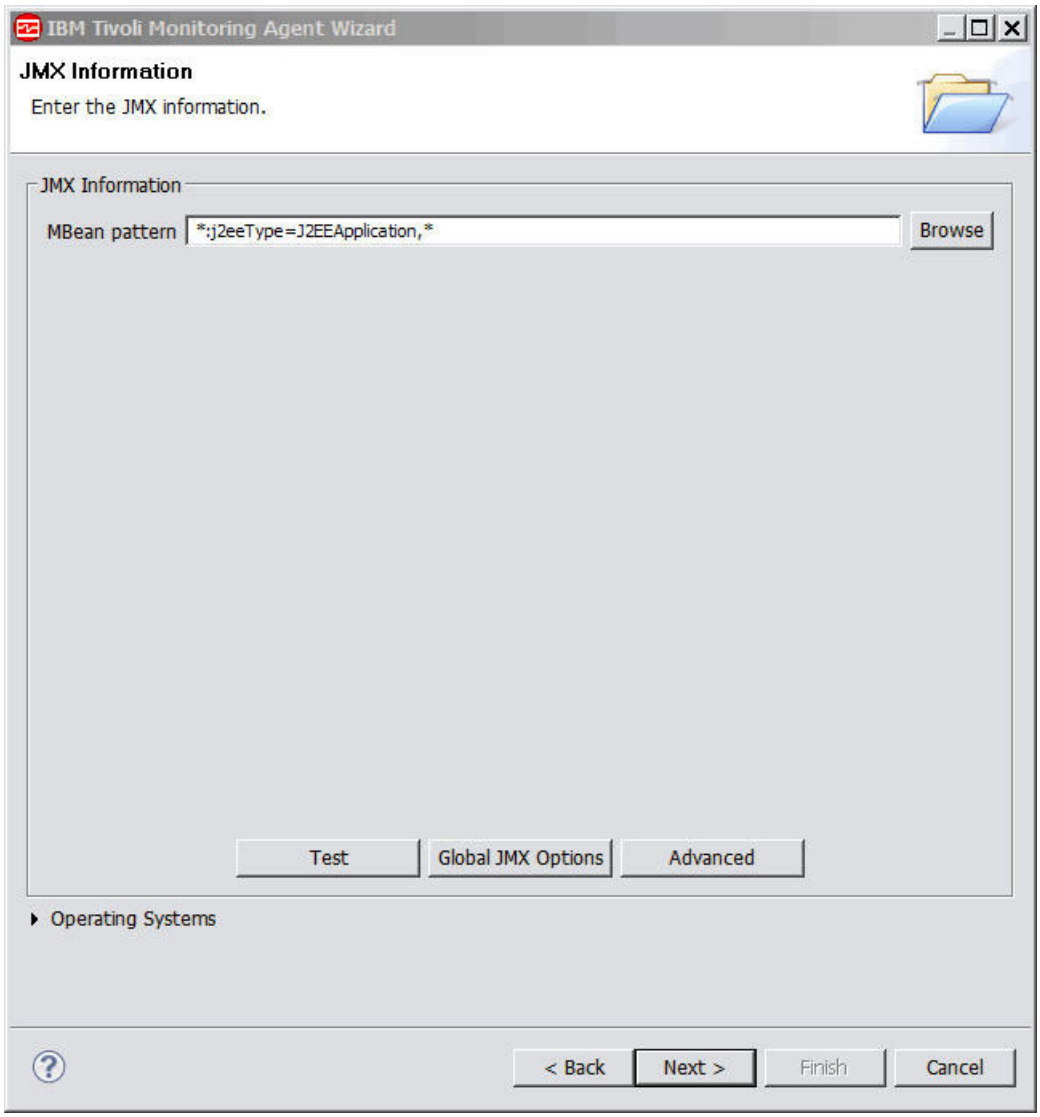


Figure 109. JMX Information page

11. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see ("Testing JMX attribute groups" on page 169)

12. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see “Filtering attribute groups” on page 67
13. Click **Next**.
14. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see (“Selecting key attributes” on page 26).
15. Click **Next**.

    The JMX Agent-Wide Options window shows the types of application servers that the Agent Builder supports. If you previously selected **Set as agent configuration defaults** on the Connection Properties page, the type of application server that you browsed to is automatically selected.
16. In the **JMX Agent-Wide Options** window (Figure 110), select any other types of application servers to which you want your agent to be able to connect.

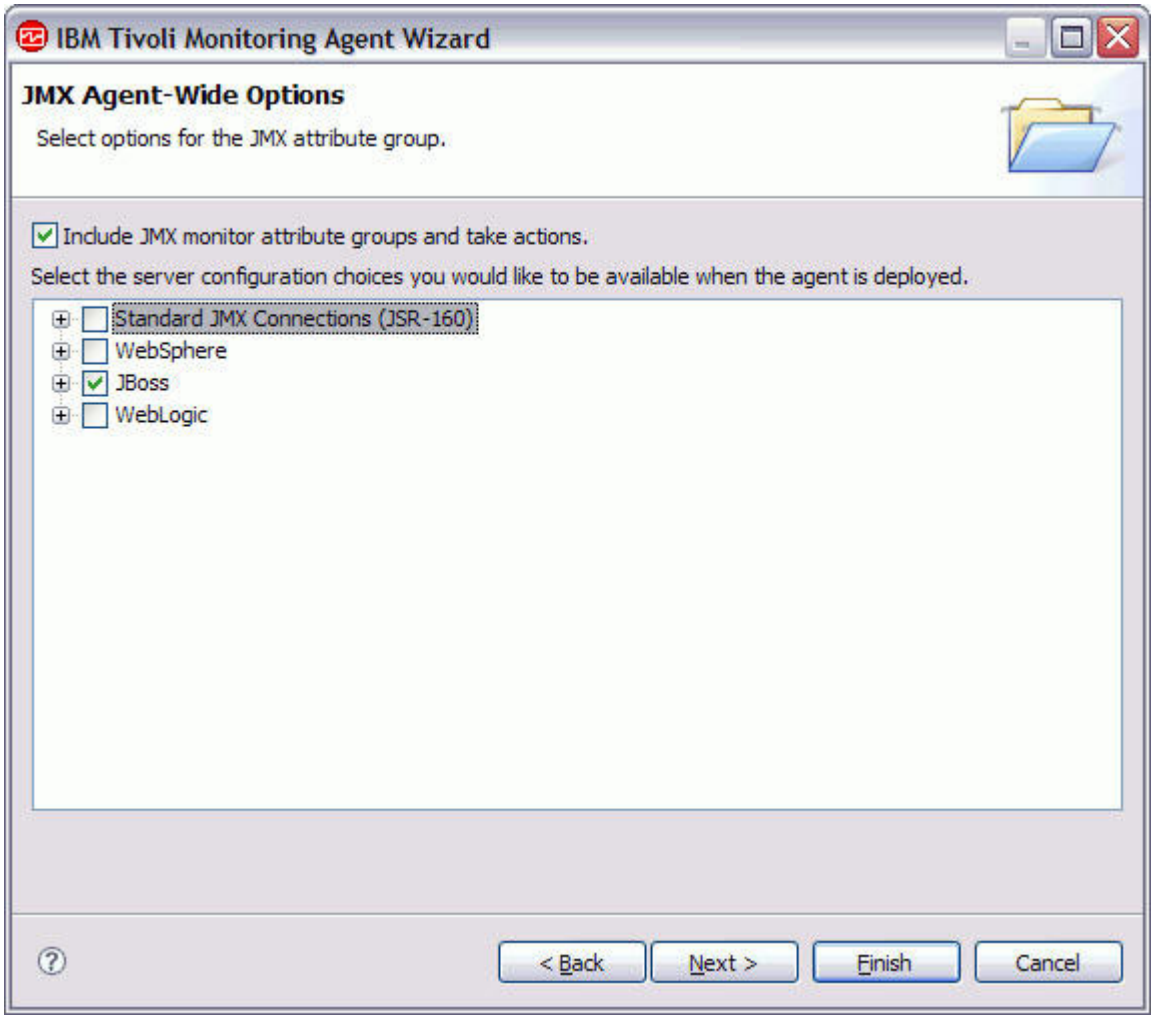


Figure 110. JMX Agent-Wide Options window

17. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

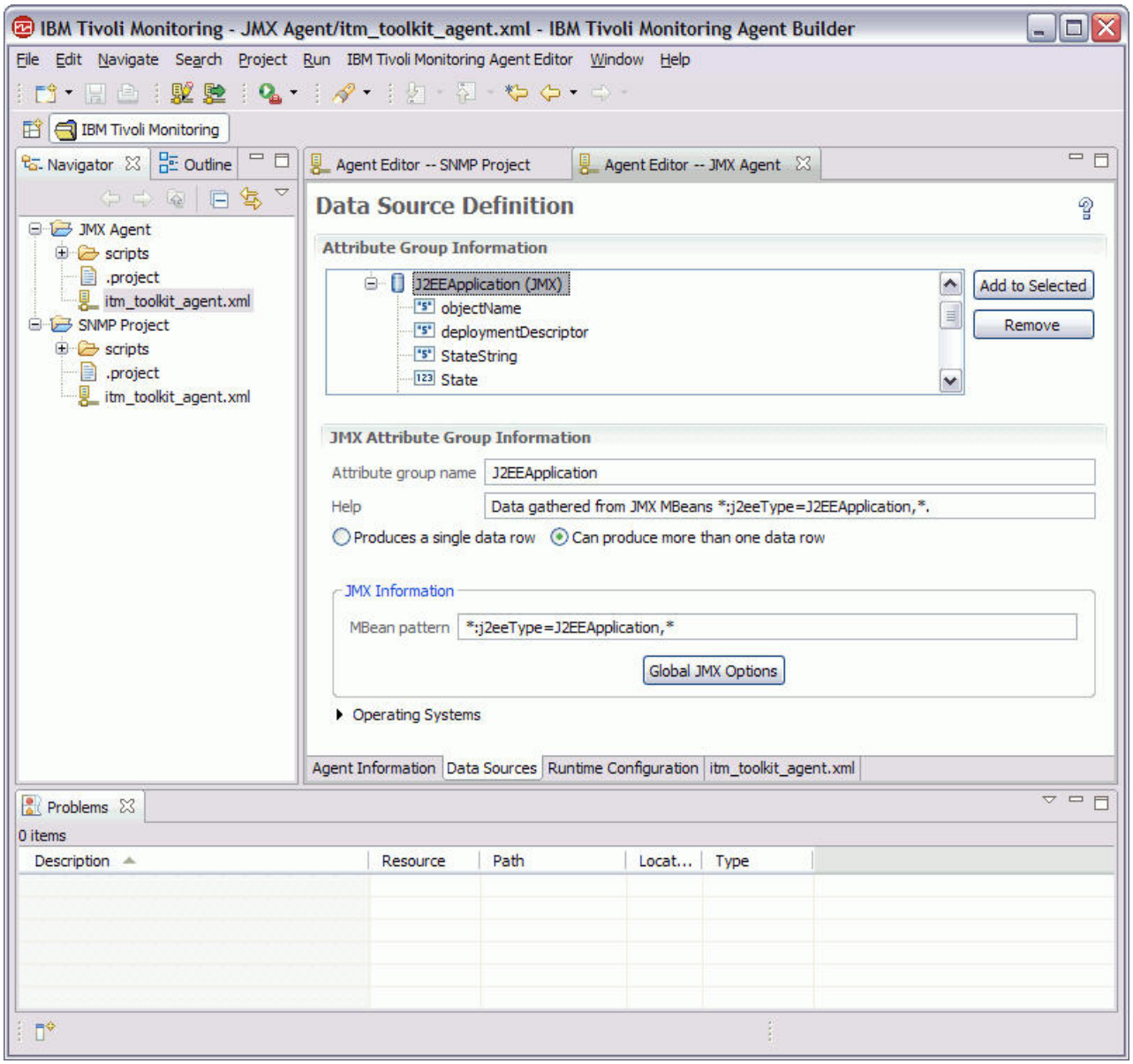


Figure 111. New attribute group in Agent Editor

18. If you want to change the types of application servers to which you can connect after the agent is created, click **Global JMX Options** in the **JMX Data Source Information** area (Figure 111).
19. In the JMX Agent-Wide Options window (Figure 112 on page 155), change any selections that you want.

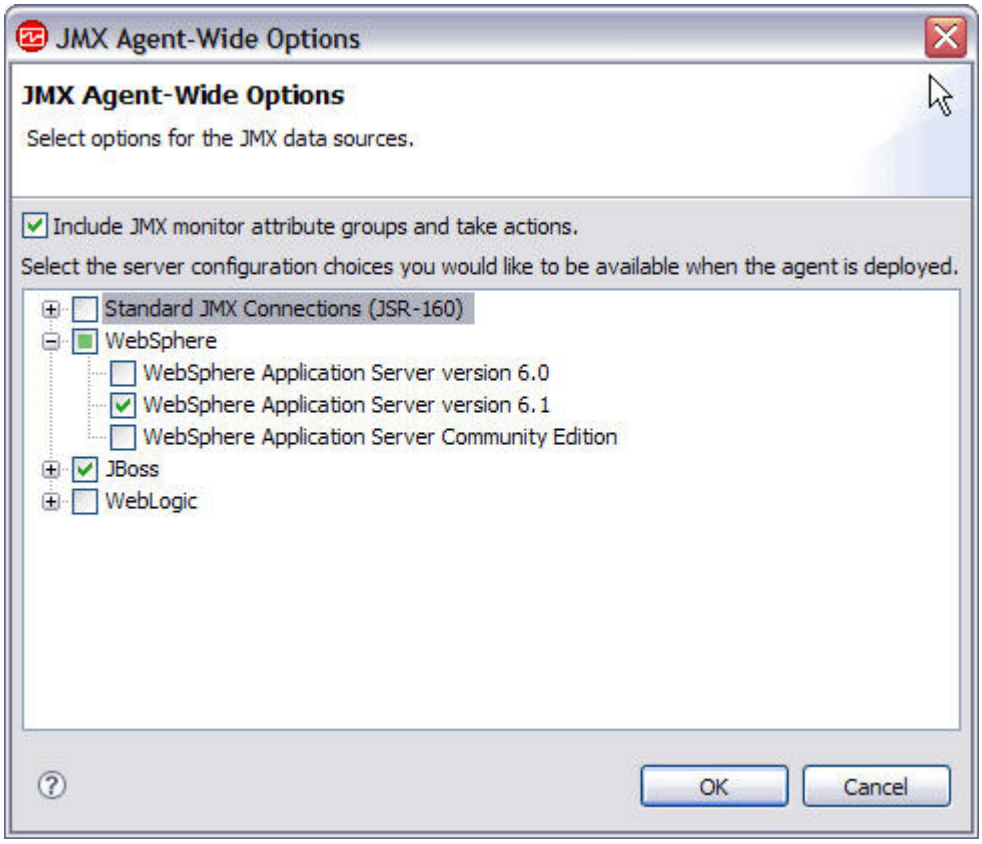


Figure 112. JMX Agent-Wide Options window

20. Click **OK**.
21. To view the configuration sections and properties that were automatically generated, click the **Runtime Configuration** tab of the Agent Editor (Figure 113 on page 156).

    The default value of the property, JBoss base paths, has the same value that was entered in the JMX browser in (Figure 108 on page 150).

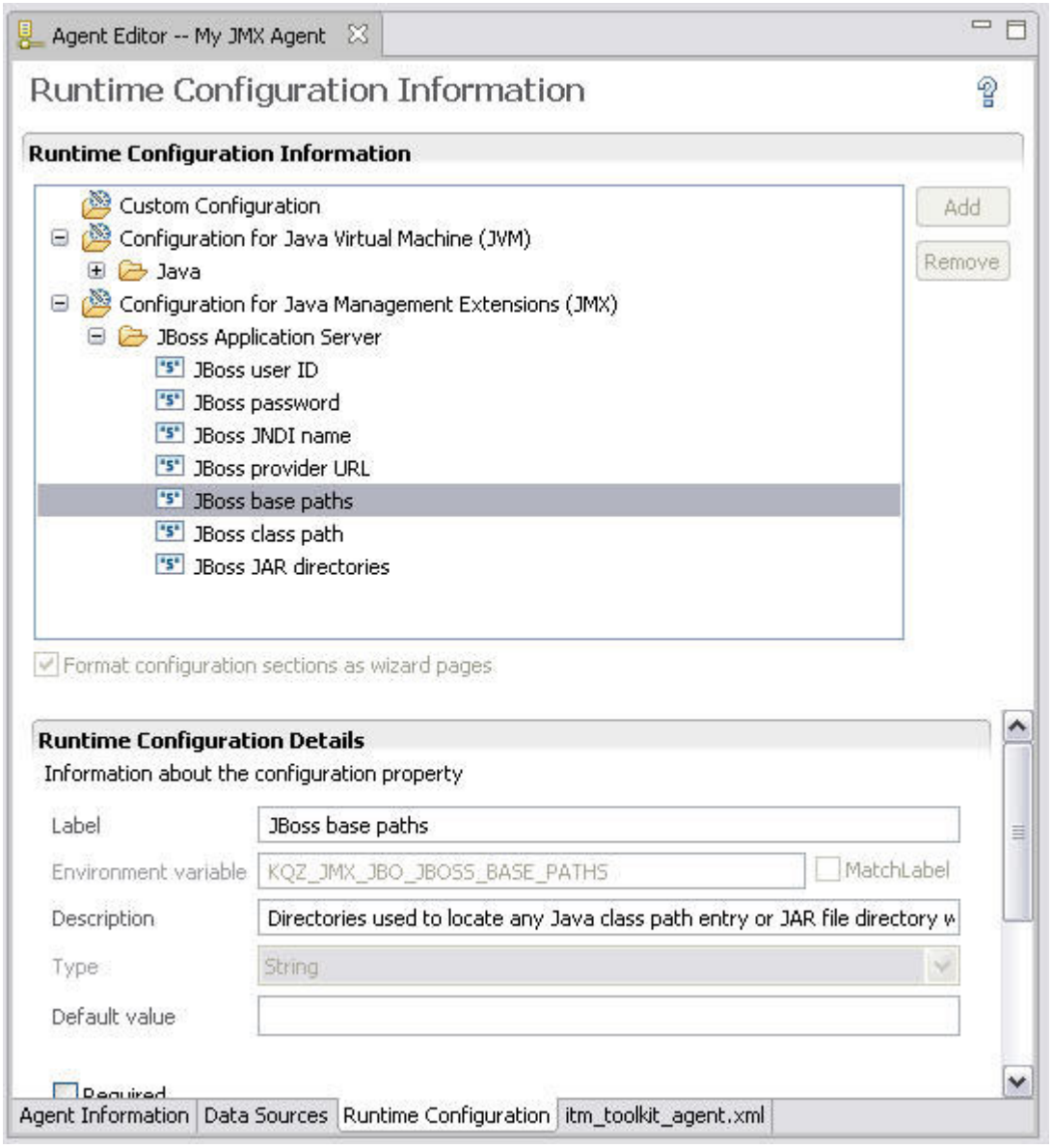


Figure 113. Runtime Configuration tab of the Agent Editor

What to do next

For more information about the attribute groups for JMX events, see ("JMX Event attribute groups" on page 571),

JMX configuration

When you define a JMX data source in your agent, some configuration properties are created for you.

JMX runtime configuration is unique because it provides you with some control over how much configuration is displayed. The JMX client for the agent can connect to several different types of application servers. However, it is not necessary to support all of those types of application servers in any one agent. You can determine which types of application servers to support, and unnecessary configuration sections are not included in the agent.

In most cases, an agent is designed to monitor one JMX application server type. When you create the JMX data source, you can use the JMX Browser. When you use the JMX Browser, the JMX server configuration options that are used to browse the MBean server are added to your agent automatically. To change the types of application servers to which you can connect after the agent is created, click **Global JMX Options** in the JMX Information area, (Figure 111 on page 154). In the JMX Agent-Wide Options page (Figure 112 on page 155), change any selections that you want.

You can design a generic agent that monitors more than one type of JMX application server. In this case, more than one JMX server configuration choice can be selected on the JMX Agent-Wide Options page (Figure 112 on page 155). When more than one type of JMX connection is supported, the runtime configuration prompts you for the connection type that are used for that agent instance.

Note: An instance of an agent can connect only to one type of JMX application server. Subnodes can be used to connect to different JMX application servers of the same type within an agent instance. To connect to more than one type of JMX application server, you must configure at least one agent instance for each JMX application server type.

You can view, add, and change the configuration properties by using the Agent Editor. See (“Changing configuration properties by using the Agent Editor” on page 369). If a JMX data source is defined in a subnode, you are also able to specify Subnode Configuration Overrides. See (“Subnode configuration” on page 344).

If you define a JMX data source in your agent, the agent must use Java to connect to the JMX application server. Java configuration properties are added to the agent automatically.

The following Java configuration properties are specific to the agent runtime configuration:

Java Home
Fully qualified path that points to the Java installation directory

Configure the agent to use the same JVM that the application you are monitoring uses, particularly for the WebLogic Server and WebSphere Application Server.

JVM Arguments
Specifies an optional list of arguments to the Java virtual machine.

Trace Level
Defines the amount of information to write to the Java trace file. The default is to write-only Error data to the log file. For more information, see (“Trace log format” on page 426).

Note: The Agent Builder does not require these properties because it uses its own JVM and logging, which is configured through the JLog plug-in.

If you define a JMX data source in your agent, the following required, common configuration fields are added to the agent automatically:

Connection
The type of connection to the MBean server

User ID
User ID that is used to authenticate with the MBean server.

Password
Password for the user ID.

Base paths
Directories that are searched for JAR files that are named in **Class path**, or directories that are named in **JAR directories**, that are not fully qualified. Directory names are separated by a semi-colon (`;`) on Windows, and by a semi-colon (`;`) or colon (`:`) on UNIX systems.

Class path
Explicitly named JAR files to be searched by the agent. Any that are not fully qualified are appended to each of the Base Paths until the JAR file is found.

JAR directories
Directories that are searched for JAR files. Directory names are separated by a semi-colon (`;`) on Windows, and by a semi-colon (`;`) or colon (`:`) on UNIX systems. The JAR files in these directories are not required to be explicitly identified; they are found because they are in one of these directories. Subdirectories of these directories are not searched. Any directory names that are not fully qualified are appended to each of the Base Paths until the directory is found.

Note: For remote monitoring, the JAR files and all of their dependent JAR files must be installed locally on the computer where the agent is running. These JAR files are the files that are required to connect to the application that is being monitored. These JAR files must be configured in **JAR directories**, and in **Base paths** and **Class path**. In addition, locally install a supported JVM for the application you are monitoring and specify the path in the **Java Home configuration** field.

Examples:

- For WebLogic 10, the class path is `server/lib/wlclient.jar;server/lib/wljmxclient.jar`. The base path points to the WebLogic application server directory where the `server/lib` directory is located.
- For WebSphere, the base path points to the location where the WebSphere Application Server is installed. Multiple base paths are listed in this example to provide a default for Windows and UNIX. The class path lists the JAR files relative to the base path. The relative value `lib` for the **JAR directories** field causes all JAR files in this directory under the base path to be loaded.
  - **Base paths**: `C:\Program Files\IBM\WebSphere\AppServer;/opt/IBM/WebSphere/AppServer`
  - **Class path**: `runtimes/com.ibm.ws.admin.client_6.1.0.jar;plugins/com.ibm.ws.security.crypto_6.1.0.jar`
  - **JAR directories**: `lib`

Depending on which JMX server types are selected in the JMX Agent-Wide Options page, some or all of the following configuration properties are added. Default values are provided by the Agent Builder, and can be modified:

JSR-160 Compliant Server and WebSphere Community Edition connection-specific configuration properties:

MX Service URLJ
JMX Services URL to connect to for monitoring.

WebSphere Application Server 6.0, 6.1, and 7.0 connection-specific configuration properties:

Host name
Host name of the system where the application server you are monitoring is located. For local monitoring, the name is the local system name. For remote monitoring, the name is the host name of the system where the application server is located.

Port Port number to use on the host name to be monitored.

Connector protocol
Connector protocol to be used by the monitoring connection. RMI and SOAP are supported.

Profile name
Name of the profile to use for configuring the connection.

JBoss Application Server connection-specific configuration properties:

JNDI Name
JNDI Name that is used to look up the MBean server.

Provider URL
JMX Services provider URL to connect to for monitoring.

WebLogic Server version 9 and 10 connection-specific configuration properties:

Service URL
JMX Services provider URL to connect to for monitoring that includes the JNDI name.

Note: If WebSphere administrative security is enabled, you must make sure that client login prompts are disabled in the appropriate client connection properties files. For RMI connections, to prevent clients from prompting the user, you must modify the *com.ibm.CORBA.loginSource* property in the `sas.client.props` file in the profile properties directory of your WebSphere Application Server. For a SOAP connection, you must modify the *com.ibm.SOAP.loginSource property*in the `soap.client.props` file in the same directory. In both cases, the *loginSource* property must be set to not contain a value.

You can view, add, and change the configuration properties by using the Agent Editor. See ("Changing configuration properties by using the Agent Editor" on page 369). If a Windows data source is defined in a subnode, you can also specify Subnode Configuration Overrides. See ("Subnode configuration" on page 344).

JMX notifications

In addition to providing monitoring data when requested, some MBeans also provide notifications.

A notification is an object that is generated by an MBean that is passed to registered listeners when an event occurs.

Agents that are built by the Agent Builder can define attribute groups that contain values from notifications rather than MBeans.

When the agent is started, a notification listener is registered with each MBean that matches the MBean pattern of the attribute group. The attribute group then displays one row per notification received. Each column contains one item of data from the notification. The data wanted from the notification is defined by a column value similar to the way column data is defined for MBeans.

For non-event based attribute groups, data is collected when needed. For event-based attribute groups, the agent maintains a cache of the last 100 events received. These events are used to respond to requests from the Tivoli Enterprise Portal. The events are forwarded immediately for analysis by situations and warehousing.

JMX monitors

In addition to providing monitoring data when requested, some MBeans also provide monitors.

The JMX Provider supports the ability for an agent to create JMX Monitors. A JMX Monitor is an MBean that the JMX agent creates on the JMX Server. It monitors the value of an attribute of another MBean and sends a notification when that value meets some criteria. Thresholds are defined that enable the Monitor to report on specific attribute values.

Not all application servers support the creation of monitors from a JMX client, which is true for current releases of WebSphere Application Server. JMX Monitors and Take Action commands can be included in your agent by selecting **Include JMX monitor attribute groups and take actions** under Global JMX Options.

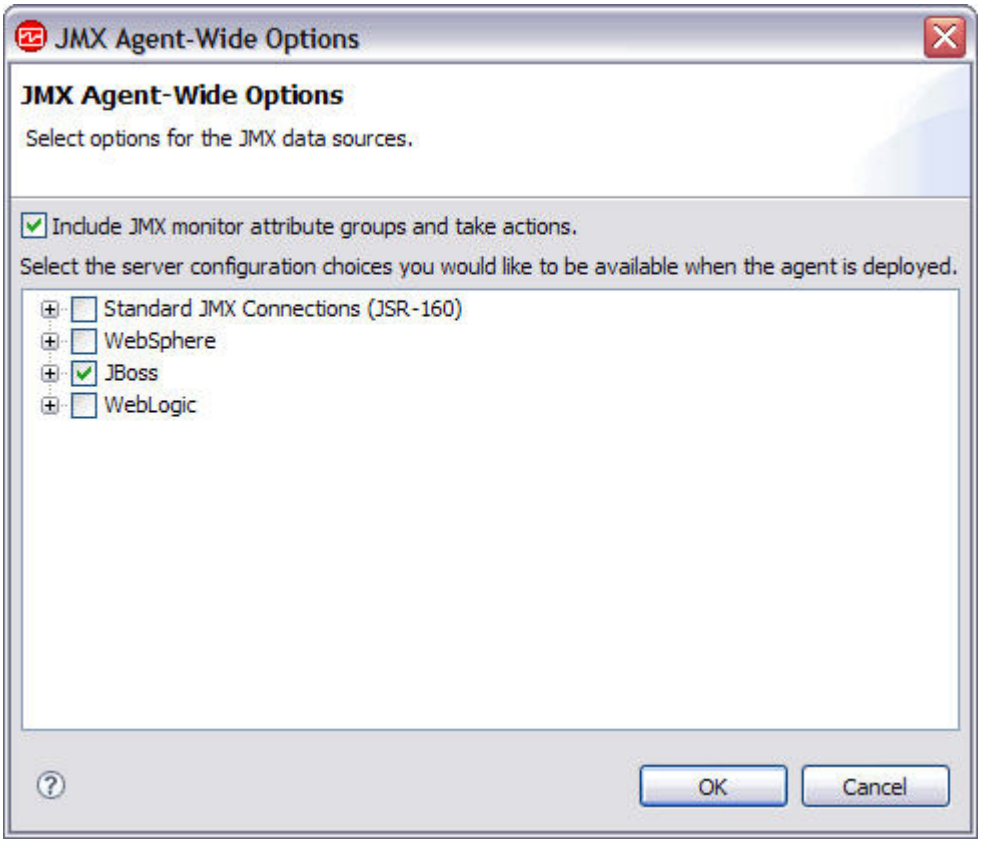


Figure 114. JMX Agent-Wide Options page

Any MBean that reports on an attribute of another MBean can be considered a monitor. In practice, JMX defines three concrete monitor classes, which are the types of monitors that are created. The following concrete monitor types are created:

- String monitor – watches a string attribute, reports equality, or inequality of that string.
- Gauge monitor – watches a variable numeric attribute, reports up or down movement beyond threshold values.
- Counter monitor – watches an increasing numeric attribute, reports when it reaches a threshold value or increases by a certain amount.

The following attribute groups might be automatically added to the agent to collect or represent JMX Monitor notifications:

- Registered Monitors

  This attribute group displays all of the JMX Monitors that are added by the user.
- Counter Notifications

  This attribute group reports all notifications that are received from Counter Monitors.
- Gauge Notifications

  This attribute group reports all notification received from Gauge Monitors.
- String Notifications

  This attribute group reports all notifications that are received from String Monitors.

Take Action commands for JMX monitors

A monitor is created by running a Take Action command.

Three Take Action commands are defined, one to create each type of monitor, and a fourth Take Action is defined to delete an existing monitor. A 256-character limit applies to Take Action commands.

The monitor attribute groups are a part of every JMX agent that is built, including all agents that are built by the Agent Builder. The four Take Action commands are available to all agents, though they cannot be used unless it is a JMX agent.

For more information, see the following topics:

- "JMX Add String Metric Watcher"
- "JMX Add Gauge Metric Watcher"
- "JMX Add Counter Metric Watcher" on page 162
- "JMX Delete Metric Watcher" on page 163

JMX Add String Metric Watcher

Use this Take Action command to create a monitor to watch a string attribute.

Parameters

MBean pattern
All MBeans matching this pattern are monitored by this monitor.

Observed attribute
Name of the MBean string attribute that is being watched.

Notify match
True if a notification is to be sent when the monitored string matches a reference value, false if not (defaults to false).

Notify differ
True if a notification is to be sent when the monitored string does not match the reference value, false if not (defaults to true)

Reference value
String to compare with the observed attribute.

A default means that the argument is not specified.

Example: Request a notification when a service is stopped

```
STRING_METRIC_WATCHER [*:type=Service,*] [StateString] [true] [false] [Stopped]
```

Where:

:type=Service,
MBean pattern: Monitors any MBean with a key property named type whose value is Service.

StateString
Observed attribute: A string attribute that is common to all MBeans of `type=Service`.

true
Notify match: You want a notification to be sent to your agent when the `StateString` attribute matches your reference value of Stopped.

false
Notify differ: You do not want to be notified when the Service attribute does not match Stopped.

Stopped
Reference value: When the `StateString` attribute changes to the value Stopped, a notification is sent.

JMX Add Gauge Metric Watcher

Use this Take Action command to create a monitor to watch a gauge attribute.

Parameters

MBean pattern
All MBeans matching this pattern are monitored by this monitor.

Observed attribute
Name of the MBean string attribute that is being watched.

Difference mode
True if the value monitored is the difference between the actual current and previous values of the attribute. False if the value monitored is the actual current value of the attribute (defaults to false).

Notify high
True if a notification is to be sent when an increasing monitored value crosses the high threshold, false if not (defaults to true).

Notify low
True if a notification is to be sent when a decreasing monitored value crosses the low threshold, false if not (defaults to true).

High threshold
Value that the observed attribute is expected to stay under.

Low threshold
Value that the observed attribute is expected to stay over.

Example: Request a notification when free memory goes under 10 Mb

```
GAUGE_METRIC_WATCHER [ServerInfo] [FreeMemory] [false] [false] [true] [30000000] [10000000]
```

Where:

***:type=ServerInfo**
MBean pattern: Monitors any MBean whose name has a single key property named type whose value is ServerInfo.

FreeMemory
Observed attribute: Numeric attribute that fluctuates up or down, this one indicating the amount of free memory in the application server.

false
Difference mode: Monitors the actual attribute value, not the difference between one observation and another.

false
Notify high: Notification is not sent when free memory goes up.

true
Notify low: Notification is not sent when the free memory becomes too low.

30000000
High threshold: Even though you are not concerned with passing a high threshold, you need a reasonable high threshold value. A second low threshold notification does not occur until the attribute value hits or passes the high threshold.

10000000
Low threshold: Low threshold value that you want to be notified about.

JMX Add Counter Metric Watcher

Use this Take Action command to create a monitor to watch a counter attribute.

Parameters

MBean pattern
All MBeans matching this pattern are monitored by this monitor.

Observed attribute
Name of the MBean string attribute that is being watched.

Initial threshold
Value that the observed attribute is compared.

Offset
Value added to the threshold after the threshold is exceeded to create a changed threshold.

Modulus
Maximum value of counter after which it rolls over to 0.

Difference mode
True if the value monitored is the difference between the actual current and previous values of the attribute. False if the value monitored is the actual current value of the attribute (defaults to false). This mode effectively turns on rate-of-change monitoring.

Granularity period
Frequency with which measurements are taken (defaults to 20 seconds). Most important if difference mode is true

Example: Request a notification when any server has three or more errors

```
COUNTER_METRIC_WATCHER [*:j2eeType=Servlet,*] [errorCount] [3] [4] [] [diff] [gran]
```

Where:

:j2eeType=Servlet,
MBean pattern: Monitors any J2EE servlet MBean whose name has a single key property named type whose value is `ServerInfo`

errorCount
Observed attribute: Increasing numeric attribute, this one indicating the number of errors of the servlet.

3 Initial threshold: You want to be notified when `errorCount` meets or exceeds 3.

4 Offset: When you get a notification for three errors, 4 is to the previous threshold of 3 to make a new threshold of 7. A second notification will be sent after `errorCount` reaches 7; a third at 11; a fourth at 15, and so on. Zero or none is not valid because it expects the counter to always increase and not increasing the offset would not make sense for a counter.

Modulus:
errorCount has no architected maximum value, so use an unreasonably high value.

false
Difference mode: You are concerned with absolute error counts. Difference is true if you are interested in the rate that errorCount was increasing.

Granularity period: Not set, so take the default granularity period of 20 seconds. Granularity period is available for all monitor types. However, it is shown with a counter monitor so that a meaningful rate of change (with difference mode=true) can be determined.

JMX Delete Metric Watcher

Use this Take Action to delete a monitor.

Parameter

Number
Monitor number as shown in the REGISTERED_MONITORS table

Example: Delete monitor number 2

```
DELETE_WATCHER [2]
```

Where:

2= Number of monitor to be deleted.

JMX operations

In addition to providing monitoring data when requested, some MBeans also provide operations.

Agents that have JMX data sources include the JMX_INVOKETake Action command that you can use to run JMX operations against the server you are monitoring.

Take Action command syntax

The action has the following syntax:

```
JMX_INVOKE [MBean pattern] [Operation name] [Argument 1] [Argument 2]
[Argument 3] [Argument 4]
```

Where:

MBean pattern
MBean query that selects the MBeans on which the operation runs. If the pattern matches more than one MBean, the operation runs on each of the matched MBeans.

Operation name
Name of the MBean operation to run.

Argument 1, Argument 2, Argument 3, Argument 4
Optional arguments that can be provided to the MBean operation. Arguments must be a simple data type such as a string or an integer.

The JMX invoke Take Action command returns success if the operation is successfully run. If the operation returns a value, that value is written to the JMX data provider log file.

Example: Start an operation to reset a counter

This action runs the resetPeakThreadCount operation on the Threading MBeans:

```
JMX_INVOKE [*:type=Threading,*] [resetPeakThreadCount][] [] [] []
```

Where:

:type=Threading,
MBean Pattern: This pattern matches all MBeans that have a type of Threading.

resetPeakThreadCount
Operation name: The operation that is run on every MBean that matches the pattern.

[] [] [] []
Argument 1, 2, 3, 4: The arguments are not needed for this operation. They are specified only to comply with the syntax of the action.

Example: Start an action with an argument

This action runs the getThreadCpuTime operation on the Threading MBeans. The result is logged to the JMX data provider trace file.

```
JMX_INVOKE [*:type=Threading,*] [getThreadCpuTime] [1] [] [] []
```

Where:

:type=Threading,
: MBean Pattern: This pattern matches all MBeans that have a type of Threading

getThreadCpuTime
: Operation name: The operation that is run on every MBean that matches the pattern.

1
: Argument 1: The thread id that is being queried.

[] [] []
: Argument 2, 3, 4: These arguments are not needed for this operation. They are specified as empty arguments to comply with the Take Action command syntax.

Running the JMX_INVOKE Take Action command

The agent developer cannot expect the user to run the JMX_INVOKE Take Action command. Instead, more actions must be developed that run the JMX_INVOKE Take Action. If possible in these actions, hide the details such as the operation name and the MBean pattern from the user.

Starting and stopping JMX monitors

JMX monitors are persistent across starts and stops of the agent and the JMX server.

If the agent detects that the JMX server was recycled, it reregisters the monitors. If the agent is recycled, monitors are reregistered. The monitor definitions are stored in a file that is named `default_`*`instanceName`*`.monitors` where *instanceName* is the agent instance name or default if it is a single instance agent. This file is in the following directory (note that *xx* denotes the two character product code):

- Windows systems: `TMAITM6/k`*`xx`*`/config`
- UNIX and Linux systems: *`architecture`*`/`*`xx`*`/config` (see "New files on your system" on page 392 for information about determining the architecture value)

If the agent is restarted, it uses the monitor definitions file to restore the monitors.

Specific fields for Java Management Extensions (JMX) MBeans

The syntax of the metric name for a JMX Attribute group must follow certain rules when specified on the Attribute Information window.

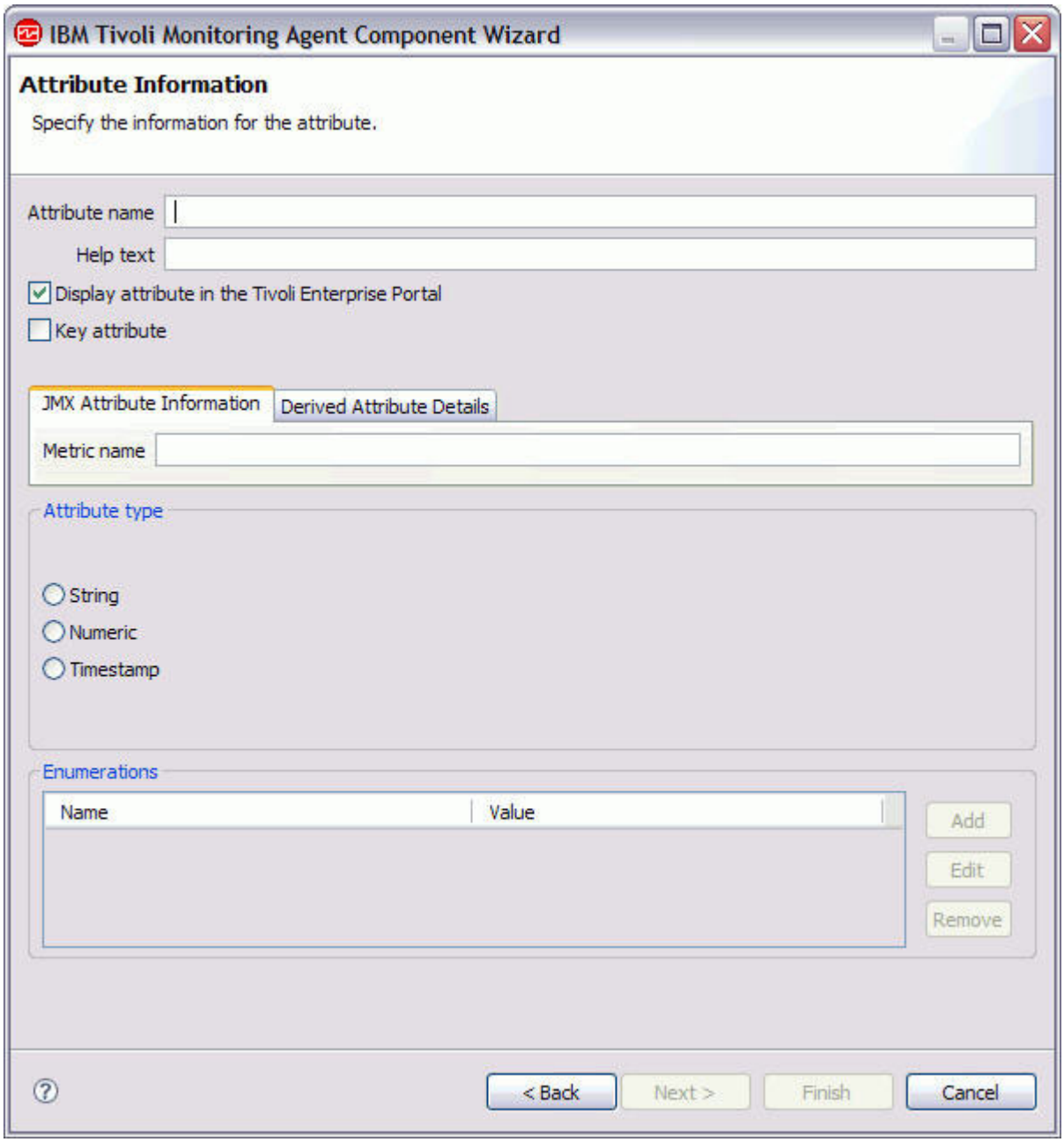


Figure 115. Attribute Information page

The syntax of the metric name for a JMX Attribute group consists of tokens that are separated by a period. The tokens form primary values and optionally secondary values:

- **Primary value**: a value that is obtained directly from the MBean or Notification in a specific row of the table. Primary values from an MBean are obtained either from an MBean attribute or from the invocation of an MBean operation (method call). Primary values from a Notification are obtained from a field or invocation of a method on the Notification object. Primary values can be primitive types, or can be Java objects.
- **Secondary value**: a value that is obtained by further processing a primary value or other secondary value. Secondary values are processed internally to the engine and do not involve calls to the JMX server. If the primary (or other secondary value) is a Java object, a secondary value is the result of fetching a public field from that object. A secondary value can also be the result of a method call on that object. Such secondary values are obtained by using Java introspection of the primary (or other

secondary) Java object. If the primary (or other secondary) value is a Java String in the form of an MBean name, the secondary value can be the domain. The secondary value can also be any of the properties that make up the MBean name.

The following syntax describes the format for the **Metric name** field:

```
Metric Name = PrimaryValue [ .SecondaryValue ]
PrimaryValue = Attribute.attributeName |
  Method.methodName |
  Domain |
  Property.propertyName |
  Field.fieldName |
  Name
SecondaryValue = Field.fieldName |
  Method.methodName |
  Domain |
  Property.propertyName |
  Explode |
  ElementCount

propertyName = the name of a key property in an MBean ObjectName
attributeName = the name of an MBean attribute
methodName = a zero-argument operation of an MBean or a zero-argument method
of a Notification or other Java object.
methodName(argument) = A single-argument operation of an MBean or a
single-argument method of a Notification or other Java object. The
argument will be passed to the method as a string.
fieldName = the name of a public instance variable in a Notification or
other Java object
notificationMethod = the name of a public zero-argument method of a
Notification object
```

By including only a primary value in the metric name definition, the data that is collected can be any of the following items:

- MBean domain
- MBean string value
- Key property from the MBean name
- Numeric or string attribute value on an MBean attribute (including the full name of another MBean). A numeric or string return value from an operation of a MBean.
- Value of a numeric or string public instance variable in a Notification object
- Numeric or string return value from an operation of a Notification.

By adding a secondary value to the definition of a metric, you can drill down into the primary value of a Java object. Also, you can start a public method or fetch a public instance variable.

By adding a secondary value to another secondary value in the definition of the metric, you can drill down into a secondary value object. You can continue as deeply as objects are nested inside an MBean or a Notification.

Tokens that make up primary and secondary values are either keywords or names. In most cases, a keyword token is followed by a name token. The following table shows some examples:

Metric name sample	Attribute group type	Description of the data returned
Domain	MBean	The domain portion of the MBean (the part before the colon).
Name	MBean	The full string representation of the MBean.
Attribute.serverVendor	MBean	MBean attribute serverVendor.

Metric name sample	Attribute group type	Description of the data returned
Method.getHeapSize	MBean	The value that is returned by the getHeapSize() on the MBean.
Property.j2eeType	MBean	The value of j2eeType is extracted from the MBean name.
Field.Message	Event (Notification)	The **Message** field in a notification.

The keywords `Attribute`, `Method`, and `Field` can return Java objects which contain other data. You can run operations on those objects by appending secondary value definitions. More examples:

Metric name sample	Attribute group type	Description of the data returned
Attribute.deployedObject.Method.getName	MBean	Takes the deployedObject attribute from the MBean and gets the result of the getName() method.
Attribute.eventProvider.Method. getException.Method.getDescription	MBean	Goes 3 deep: an attribute named eventProvider is presumed to be an object which has a getException() method. This method returns an object with a getDescription() method. That method is called and the return value is put in the column.
Attribute.HeapMemoryUsage.Method. get(used)	MBean	Takes the HeapMemoryUsage attribute from the MBean and gets the result of the get(String value) method. The string that is used is passed to the method as the argument. Only 1 argument can be provided and it must be a literal string value. Shows how you can collect data from an open MBean composite data structure.

Domain and Property can be used as keywords in secondary values if the previous value returned a String in the format of an MBean name. For example:

Metric name sample	Attribute group type	Description of the data returned
Attribute.jdbcDriver.Property.name	MBean	The attribute jdbcDriver returns an MBean name, and the key property, name, is extracted from the MBean name.
Attribute.jdbcDriver.Domain	MBean	The attribute jdbcDriver returns an MBean name, and the domain is extracted from the MBean name.

The ElementCount and Explode keywords run operations on arrays or collections of data.

- `ElementCount` – returns the number of elements in an array.
- `Explode` – explodes a row into several rows, one new row for each element of an array.

Examples of each of the keywords:

Metric name sample	Attribute group type	Description of the data returned
Attribute.deployedObjects.ElementCount	MBean	The MBean attribute deployedObjects is an array, and this column contains the number of elements in the array.
Attribute.deployedObjects.Explode. MBean.Property.j2eeType	MBean	Causes the table to have 1 row for each element in deployed objects. This column contains the j2eeType of the deployed Object.
Attribute.SystemProperties.Method. values.Explode.Method.get(key)	MBean	Causes you to get one new row for each entry in an open MBean tabular data structure. Each tabular data structure contains a composite data structure with an item named `key`, which is returned.

Testing JMX attribute groups

You can test the JMX attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the JMX Information page (Figure 109 on page 152).
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test** . For more information about the Agent Editor, see (Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35).

   After you click **Test** in one of the previous two steps, the JMX Test window Figure 116 is displayed

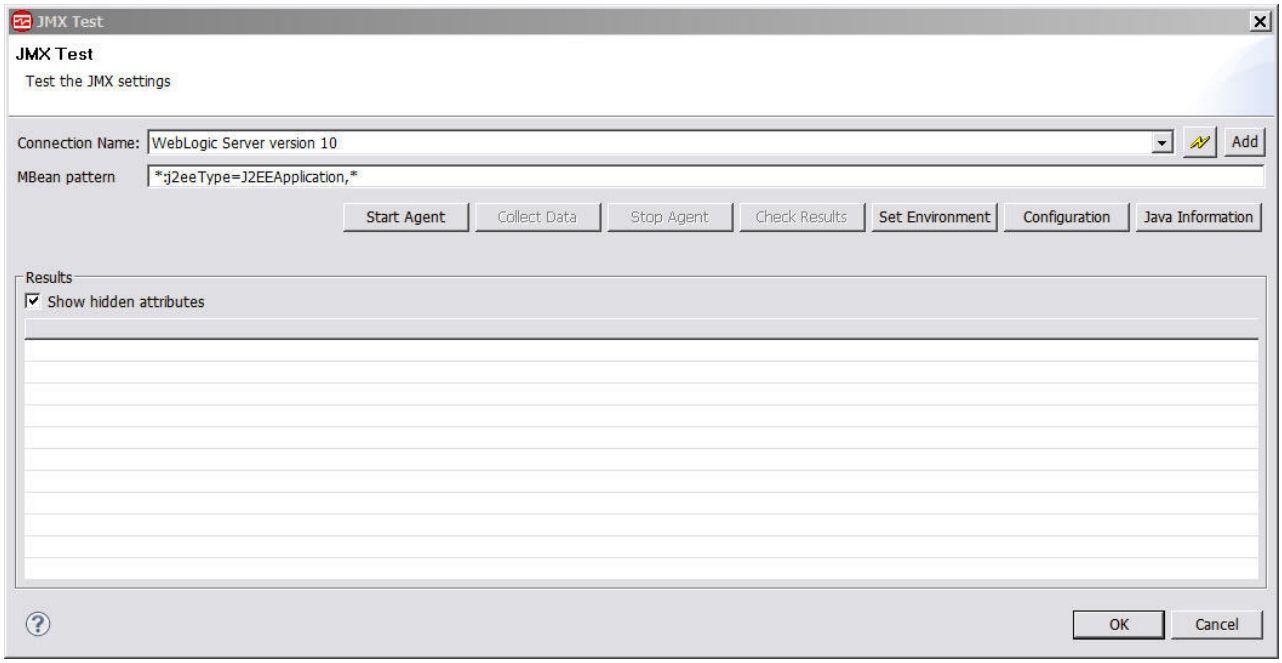


Figure 116. JMX Test window

2. Select a connection from the list available under **Connection Name** or alternatively click **Add** to add a connection and follow the procedure that is detailed underChapter 14, "Monitoring Java Management Extensions (JMX) MBeans," on page 143.

3. Optional: Before you start testing, you can set environment variables, configuration properties, and Java information. For more information, see (“Attribute group testing” on page 377). For more about JMX configuration, see (“JMX configuration” on page 156).
4. Click **Start Agent**. A window indicates that the Agent is starting.
5. Click **Collect Data** to simulate a request from Tivoli Enterprise Portal or SOAP for agent data. The agent monitors the JMX Server for data. The JMX Test window collects and shows any data in the agent's cache since it was last started.
6. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in “Performance Object Status node” on page 545
7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the JMX Test window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, “Testing and debugging your agent,” on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 15. Monitoring data from a Common Information Model (CIM)

Configure your Agent to receive data from a Common Information Model (CIM) data source.

About this task

This task describes the steps to configure a Common Information Model (CIM) data source.

Procedure

1. On the Agent Initial Data Source page (Figure 117) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

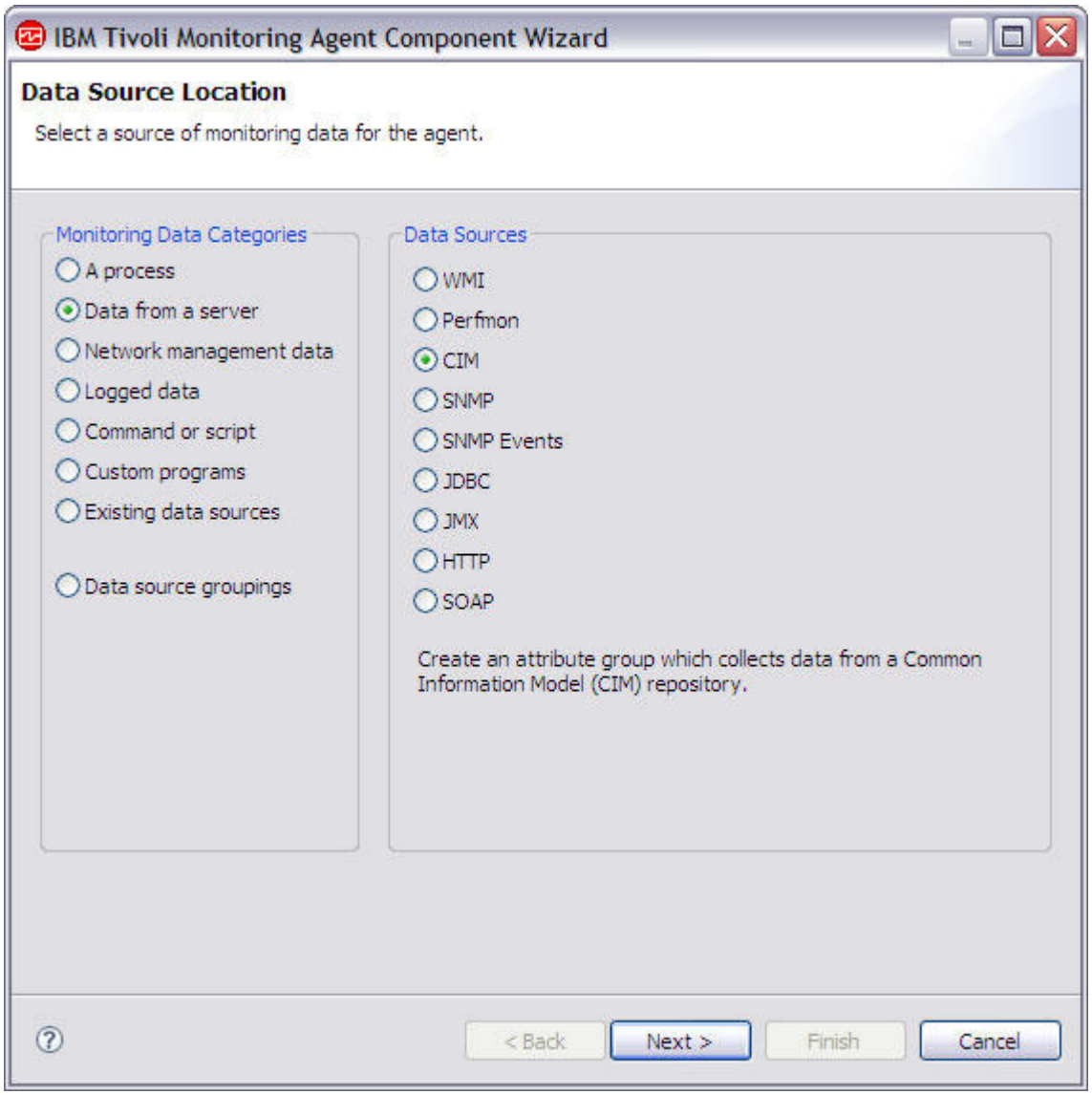


Figure 117. Adding CIM data

2. In the **Data Sources** area, click **CIM**.

© Copyright IBM Corp. 2007, 2013

3. Click **Next**.
4. On the Common Information Model (CIM) Information page (Figure 118) **CIM information** area, make one of the following choices:

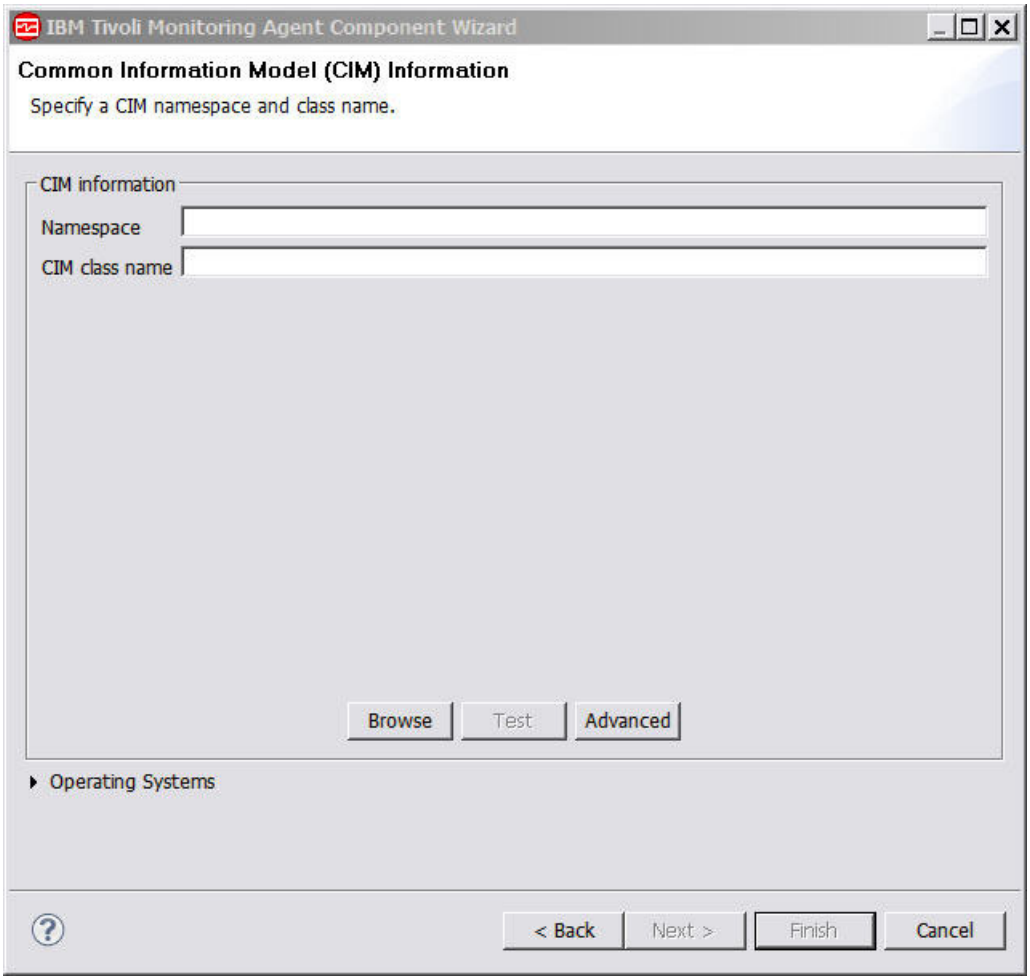


Figure 118. Common Information Model (CIM) Information page

- Complete the **Namespace** and **CIM class name** fields for the data that you want to collect.
- Click **Browse** to browse a CIM repository on a specific system.

  The Common Information Model (CIM) Class Browser window is displayed. This browser connects to a CIM server and provides you with information about the classes that exist on that server (Figure 119 on page 173).

  To browse a remote system, select a system from the **Hostname** list (if one is defined). Alternatively click **Add** to add the host name of the system where the CIM server is located.

  The syntax for specifying the host name is `http[s]://hostname:port`. If you provide the host name only, the Common Information Model (CIM) Class Browser connects by using a default URL of `http://hostname:5988`. If you provide a protocol without specifying a port, `5988` is used as the default for `http` or `5989` as the default for `https`.

  If you provide a port without specifying a protocol, `http` is used with the port provided.

  Provide a user ID and password for an account with read permission to the objects in the namespace that you want to browse. The window is updated with the information for the remote system.

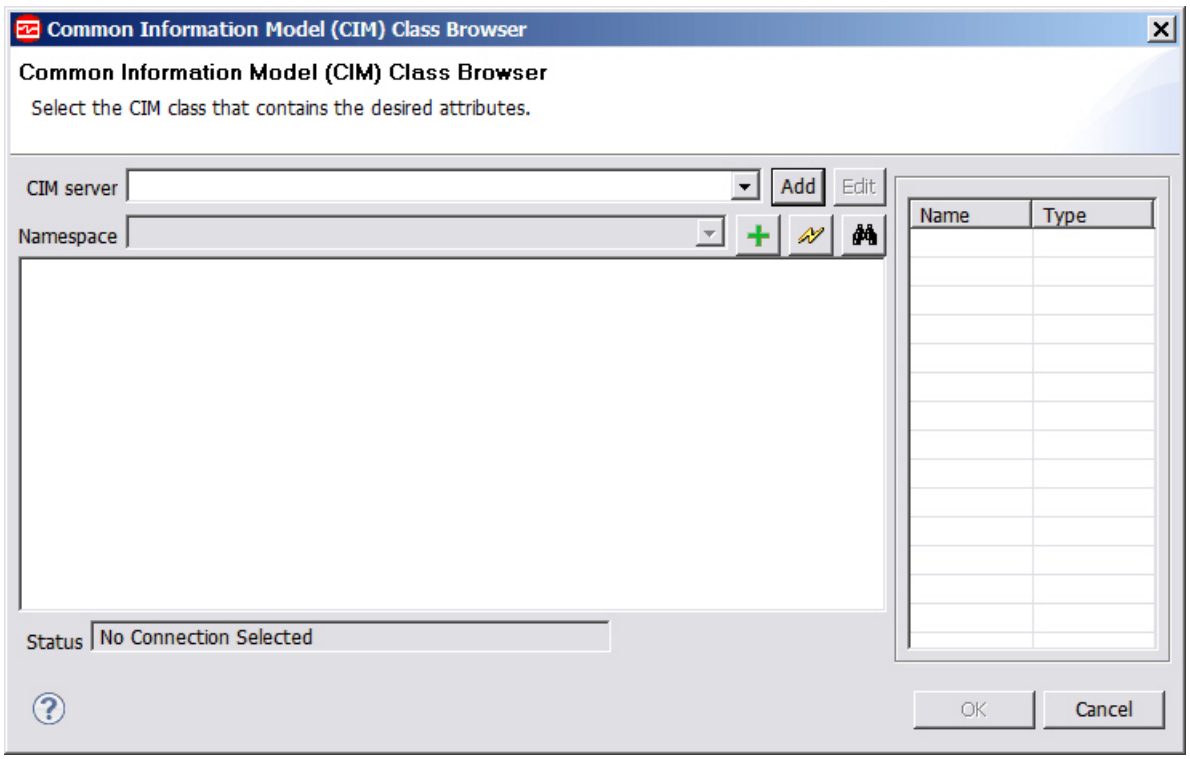


Figure 119. Common Information Model (CIM) Class Browser window

The Agent Builder attempts to discover the namespaces available on the CIM Server. The discovered namespaces are displayed in the **Namespace** list. However, the Agent Builder might not be able to discover all namespaces that are available on the server. If you want to browse a namespace that is not listed in the **Namespace** list, click the plus (+) icon next to the **Namespace** list. Enter the name of the namespace in the field and click **OK**. If the namespace is present on the CIM server, the classes that are defined in the namespace are listed. The namespaces you type are saved and put into the **Namespace** list the next time you browse that particular CIM server.

When you select a namespace from the **Namespace** list, the Agent Builder collects all of the class information for that particular namespace. Then, the Agent Builder caches this information so you can switch between namespaces quickly. If you want to force the Agent Builder to recollect the class information for a particular namespace, select the namespace and click **Connect**. Clicking **Connect** deletes any cached information, and causes the Agent Builder to recollect the class information.

You can click the **Search** (binoculars) icon to find your selection in the list. Type a phrase in the **Search phrase** field; specify your preference by clicking either the **Search by name** or **Search by class properties** fields; and click **OK**. If you find the item for which you are searching, select it and click **OK**.

5. On the Common Information Model (CIM) Information page, **Operating systems** area, select the operating systems on which the collection is to take place.
6. If you typed the Namespace and CIM class name in the **CIM information** area, do the following steps:
   a. Click **Next** to display the Attribute Information page and define the first attribute in the attribute group.
   b. Specify the information about the Attribute Information page, and click **Finish**.
7. If you browsed the CIM information, the Select key attributes page is displayed. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 26).

8. If you browsed to the CIM information, click **Finish**.
9. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing CIM attribute groups"
10. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 67
11. Do one of the following steps:
    a. If you are using the New Agent wizard, click **Next**.
    b. Click **Finish** to save the data source and open the Agent Editor.

CIM configuration

Details about CIM configuration properties.

If you define a CIM data source in your agent, CIM configuration properties are added to the agent automatically. You can view, add, and change the configuration properties by using the Agent Editor. See ("Changing configuration properties by using the Agent Editor" on page 369). If a CIM data source is defined in a subnode, specify Subnode Configuration Overrides. See ("Subnode configuration" on page 344).

The following connection-specific configuration properties are on the CIM configuration page:

CIM Local or Remote
Local or remote authentication to the CIM server. Local/Remote Default value is Remote

CIM user ID
The user ID used to access the CIM server

CIM password
The password to access the CIM server

CIM host name
The host name to be accessed for CIM data

CIM over SSL
Use SSL for communication with the CIM server. The options are Yes and No. The default value is No.

CIM port number
The port number that is used for communication that is not secure.

CIM SSL port number
The port number that is used for secure communication. The default value is 5989. (The default value for Solaris 8 is normally different.)

Testing CIM attribute groups

You can test the CIM attribute group that you created, within Agent Builder.

Procedure

1. Start the Testing procedure in the following ways:
   - During agent creation click **Test** on the CIM Information page (Figure 118 on page 172).
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35

After you click **Test** in one of the previous two steps, the Test Settings window Figure 120 is displayed

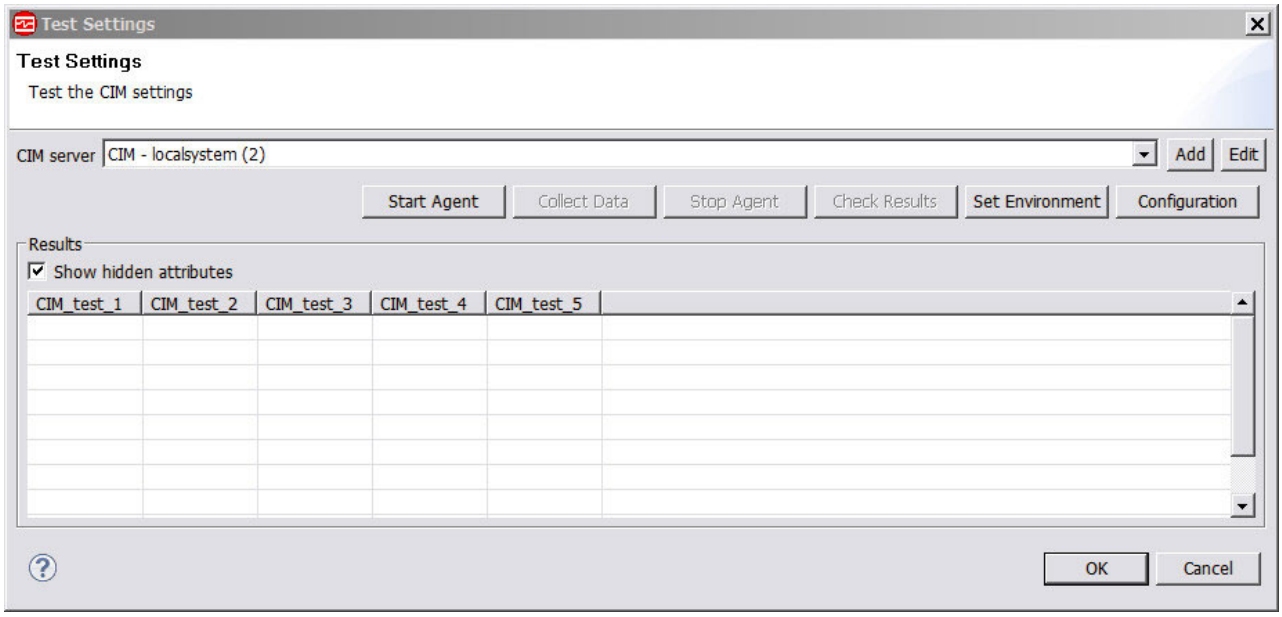


Figure 120. CIM Test Settings window

2. Optional: Set environment variables and configuration properties before you start testing. For more information, see (“Attribute group testing” on page 377).
3. Select or add a **Host name**. For more about adding a **Host name**, see Chapter 15, “Monitoring data from a Common Information Model (CIM),” on page 171
4. Click **Start Agent**. A window opens indicating that the Agent is starting.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries the CIM Server for data. The Test Settings window collects and shows any data in the agent's cache since it was last started.
6. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in “Performance Object Status node” on page 545
7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the Test Settings window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, “Testing and debugging your agent,” on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 16. Monitoring a log file

Configure your agent to receive data from a log file data source.

Before you begin

Note: The agent monitors log files that are in the same locale and code page that the agent runs in.

Procedure

1. On the Agent Initial Data Source page (Figure 121) or the Data Source Location page, click **Logged Data** in the **Monitoring Data Categories** area.

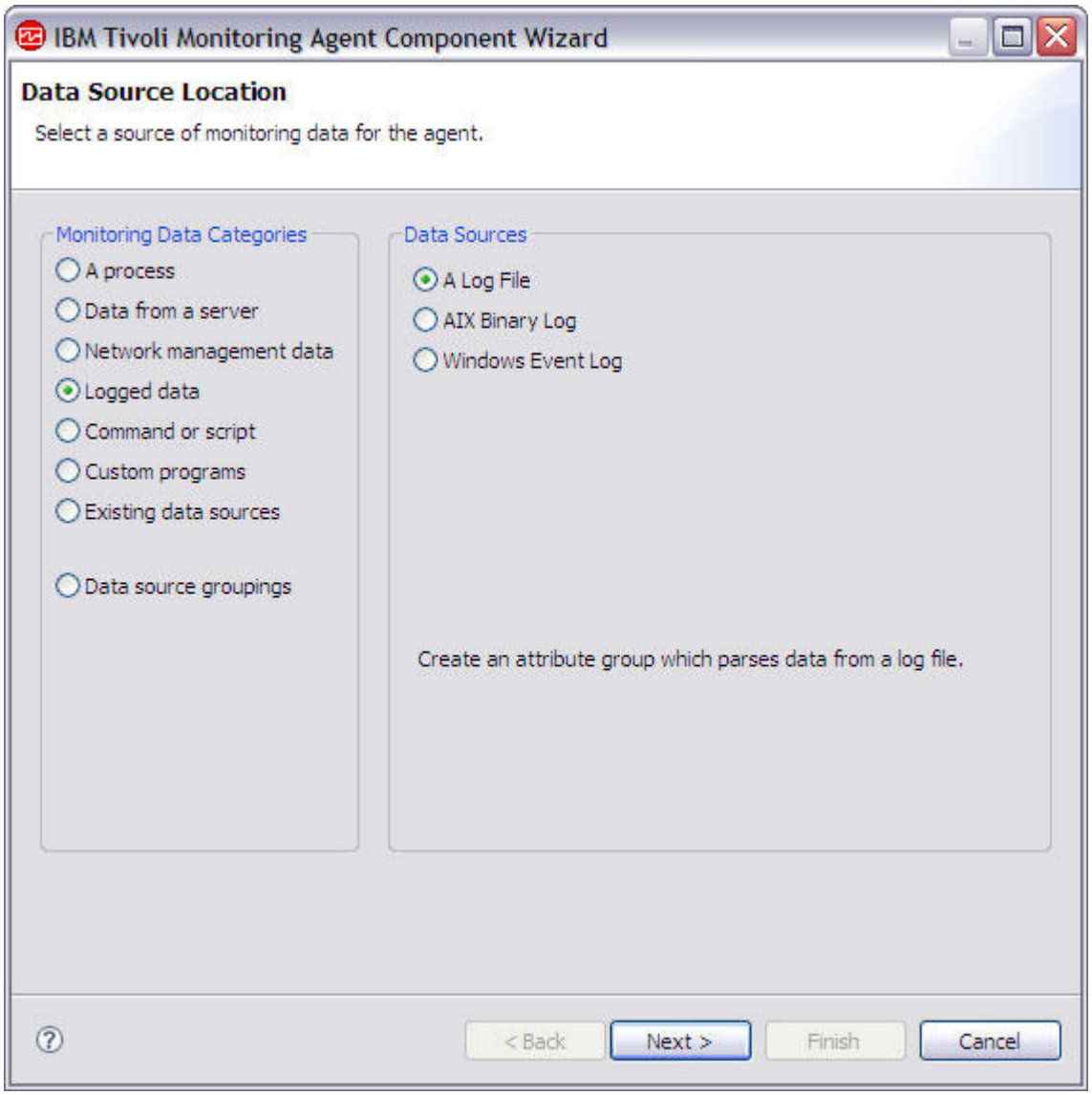


Figure 121. Adding a log file

2. In the **Data Sources** area, click **A Log File**.
3. Click **Next**.

© Copyright IBM Corp. 2007, 2013

4. On the Log File Information page (Figure 122), type the name of the log file you want to monitor in the **Log File Information** area. The file name must be fully qualified.

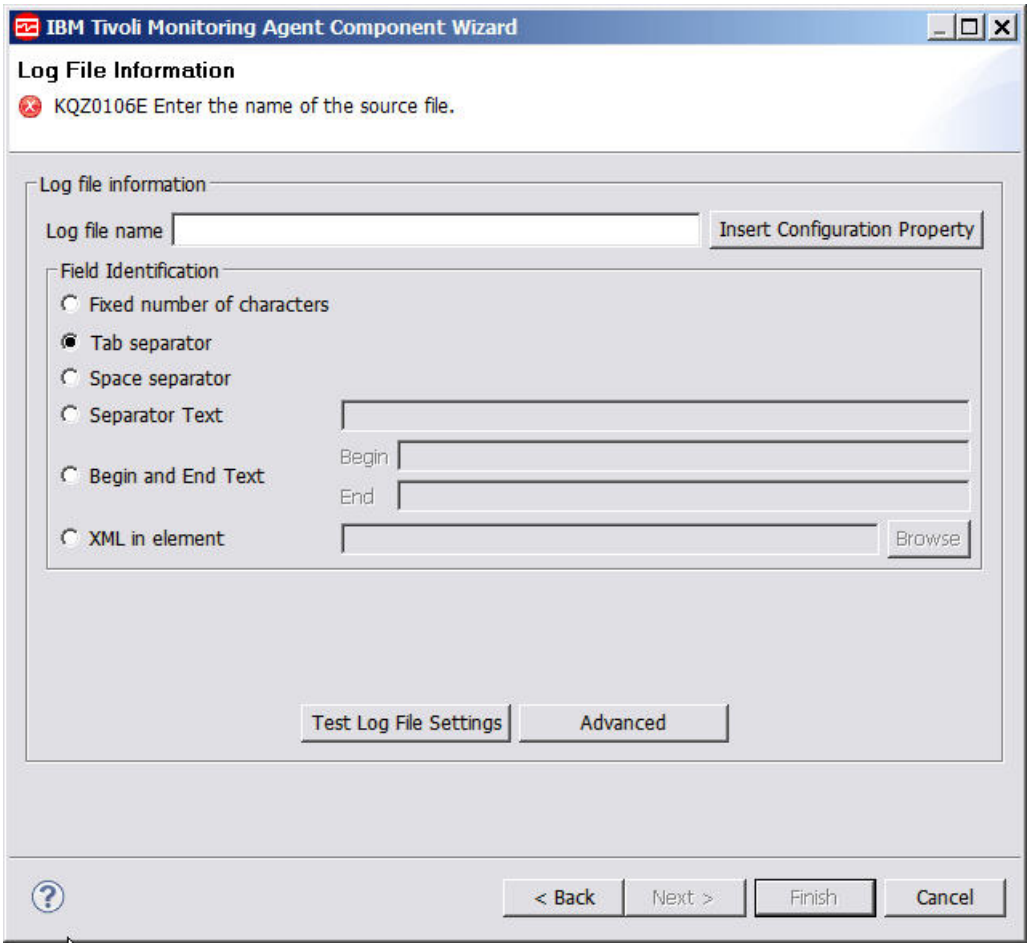


Figure 122. Log File Information page

a. Optional: Part of the log file name can come from a runtime configuration property. To create a log file name, click **Insert Configuration Property** and select a configuration property (Figure 244 on page 362).
b. Optional: The file can also be a dynamic file name. For more information, see (Appendix I, "Dynamic file name support," on page 643).

5. In the **Field Identification** area, click one of the following options:

Fixed number of characters
When selected, limits the number of characters.

With this option, each attribute is assigned the maximum number of characters it can hold from the log file. For example, if there are three attributes A, B, and C (in that order), and each attribute is a String of maximum length 20. Then, the first 20 bytes of the log record go into A, the second 20 into B, and the next 20 into C.

Tab separator
When selected, you can use tab separators.

Space separator
When selected, multiple concurrent spaces can be used as a single separator.

Separator Text
When selected, type in separator text.

Begin and End Text
When selected, type in both Begin and End text.

XML in element
When selected, type the name of the XML element to use as the record, or click **Browse** to define the element.

If you clicked **Browse**, the XML Browser window is displayed (Figure 123). If you use the browse function, the Agent Builder identifies all possible attributes of the record by looking at the child tags and their attributes.

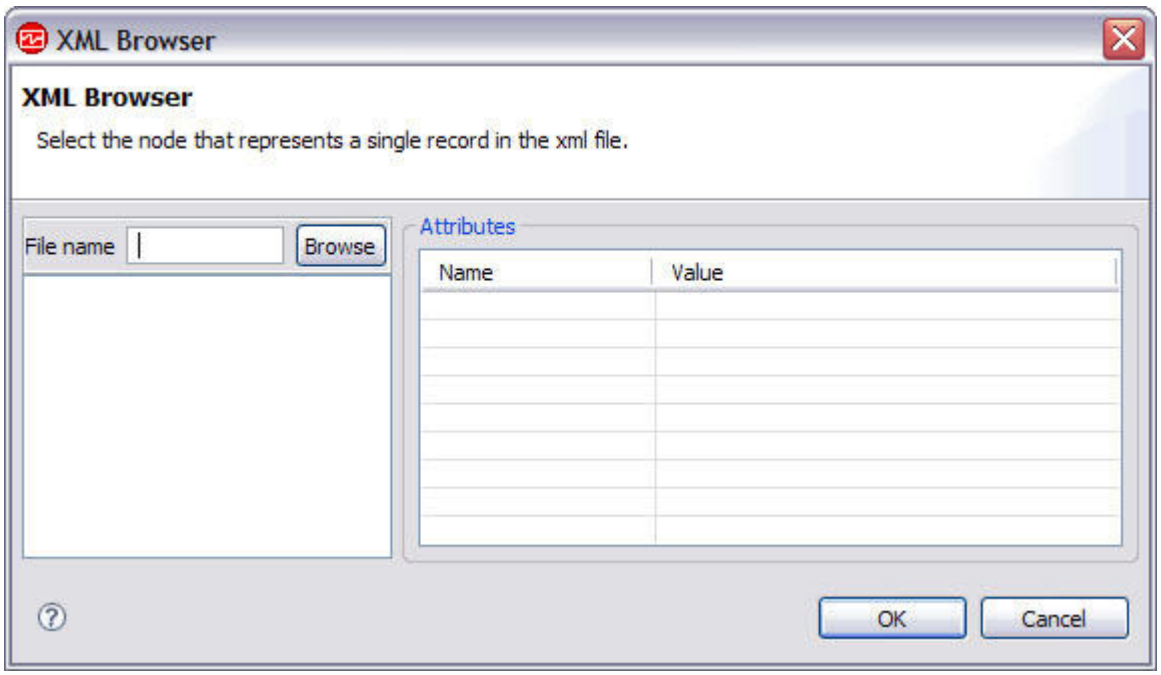


Figure 123. XML Browser window

Note: Unless you click **Advanced** and fill out the information in that window, the following assumptions are made about information that you complete:

- Only a log file at a time is monitored.
- Each line of the log file contains all the fields necessary to fill the attributes to be defined.

For more information about log file parsing and separators, see ("Log file parsing and separators" on page 192).

6. Optional: Click **Advanced** on the Log File Information page to do the following by using the Advanced Data Source Properties page (Figure 124 on page 180):
   - Monitor more than one file, or monitor files with different names on different operating systems or monitor files with names that match regular expressions.
   - Draw a set of fields from more than one line in the log file.
   - Choose **Event Filtering and Summarization Options**.
   - Produce output summary information. This summary produces an additional attribute group at each interval. For more information about this attribute group, see ("Log File Summary" on page 556). This function is deprecated by the options available in the Event Information tab.

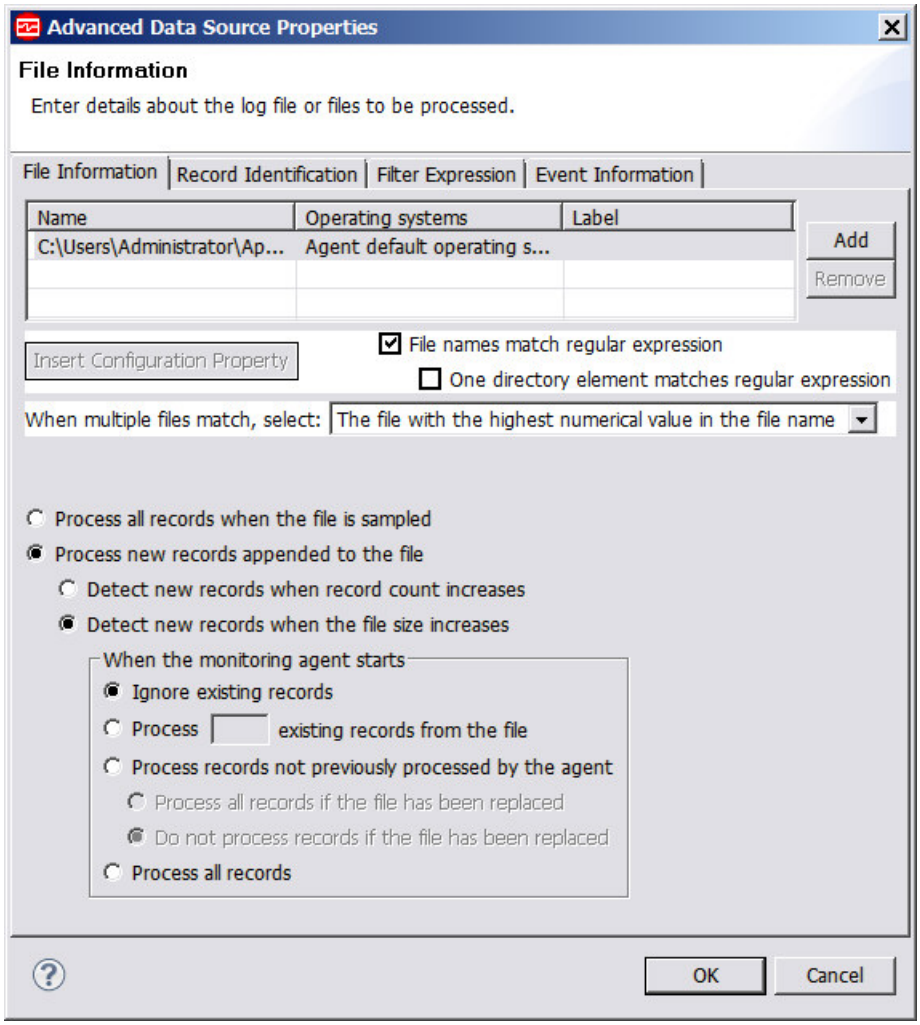


Figure 124. Advanced Data Source Properties page, File Information

a. To monitor more than 1-log file, click **Add** and type the name. If more than one file is listed, a unique label must be entered for each file. The label can be displayed as an attribute to indicate which file generated the record. It must not contain spaces.

b. Optional: To select the operating systems on which each log file is to be monitored, follow these steps:

   1) Click in the **Operating systems** column for the log file.

   2) Click **Edit**.

   3) In the Operating Systems window, select the operating systems.

   4) Click **OK** to save your changes and return to the Advanced Data Source Properties page.

c. Optional: Select **File names match regular expression** if the file name you are providing is a regular expression that is used to find the file instead of being a file name. For more information, see (Appendix G, "ICU regular expressions," on page 627). If you do not check this box, the name must be an actual file name. Alternatively it must be a pattern that follows the rules for file name patterns that are described in ("Dynamic file name syntax" on page 643).

d. Optional: Select **One directory element matches regular expression** to match one subdirectory of the file name path with a regular expression. You can select this option only if you also selected **File names match regular expression** in the previous step.

   If regular expression meta characters are used in the path name, the meta characters can be used in only one subdirectory of the path. For example, you can specify `/var/log/[0-9\.]*/mylog.*` to

have meta characters in one subdirectory. The `[0-9\.]*` results in matching any subdirectory of `/var/log` that consists solely of numbers and dots (`.`). The `mylog.*` results in matching any file names in those`/var/log` subdirectories that begin with `mylog` and are followed by zero or more characters.

Because some operating systems use the backslash (`\`) as a directory separator it can be confused with a regular expression escape meta character. Because of this confusion forward slashes must always be used to indicate directories. For example, Windows files that are specified as `C:\temp\mylog.*` might mean the `\t` is a shorthand tab character. Therefore, always use forward slashes (`/`) on all operating systems for directory separators. The `C:/temp/mylog.*` example represents all files in the `C:/temp` directory that start with `mylog`.

e. In the **When multiple files match** list, select one of the following options:
   - **The file with the highest numerical value in the file name**
   - **The biggest file**
   - **The most recently-updated file**
   - **The most recently-created file**
   - **All files that match**

   Note: When you select **All files that match**, the agent identifies all files in the directory that match the dynamic file name pattern. The agent monitors updates to all of the files in parallel. Data from all files is intermingled during the data collection process. Its best to add an attribute by selecting **Log file name** in **Record Field Information** to correlate log messages to the log files that contain the log messages. Ensure that all files that match the dynamic file name pattern can be split into attributes in a consistent manner. If the log files selected cannot be coherently parsed, then its best to select **Entire record** in **Record Field Information** to define a single attribute. For more information about specifying **Record Field Information** for attributes, see step (8 on page 185).

f. Choose how the file is processed. With **Process all records when the file is sampled**, you can process all records in the entire file every time the defined sampling interval for the log monitor expires. The default interval is 60 seconds. This interval can be modified by using the *KUMP_DP_COPY_MODE_SAMPLE_INTERVAL* environment variable (specifying a value in seconds). The same records are reported every time unless they are removed from the file. With this selection, event data is not produced when new records are written to the file. With **Process new records appended to the file**, you can process new records that are appended to the file while the agent is running. An event record is produced for every record added to the file. If the file is replaced (first record changes in any way), the file is processed and an event is produced for each record in the file.

   Note: If appending records to an XML log file, the append records must contain a complete set of elements that are defined within the XML element you selected as **Field Identification**.

g. If you chose to process new records that are appended to the file, you can also choose how new records are detected. With **Detect new records when record count increases**, new records can be detected when the number of records in the file increases, whether the size of the file changes. This feature is useful when an entire log file is pre-allocated before any records are written to the file. This option can be selected for files that are not pre-allocated, but it is less efficient than monitoring the size of the file. With **Detect new records when the file size increases**, you can determine when a new entry is appended to a file in the typical way. There might be a brief delay in recognizing that a monitored file is replaced.

h. If you selected **Detect new records when the file size increases**, you can also choose how to process a file that exists when the monitoring agent starts. **Ignore existing records** disables event production for any record in the file at the time agent starts. **Process ___ existing records from the file** specifies production of an event for a fixed number of records from the end of the file at the time the agent starts. **Process records not previously processed by the agent**: Specifies for restart data to be maintained by the monitoring agent so the agent knows which records were

processed the last time that it ran. Events are produced for any records that are appended to the file since the last time the agent was running. This option involves a little extra processing each time a record is added to the file.

i. If you selected **Process records not previously processed by the agent**, you can choose what to do when the agent starts and apparently the existing file was replaced. **Process all records if the file has been replaced**: If information about the monitored file and the restart data information do not match, events are produced for all records in the file. Examples of mismatches include: The file name is different, the file creation-time is different, the file-size decreased, the file last modification time is earlier than before. **Do not process records if the file has been replaced**: If the information about the monitored file and the restart data information do not match, disables processing of existing records in the file.

j. Click the **Record Identification** tab (Figure 125 on page 183) to interpret multiple lines in the log file as a single logical record.

   Note: If you select **XML in element** as the field identification on the Log File Information page, the **Record Identification** tab does not display.

   - **Single line** interprets each line as a single logical record.
   - **Separator line** you can enter a sequence of characters that identifies a line that separates one record from another.

     Note: The separator line is not part of the previous or the next record.
   - **Rule** identifies a maximum number of lines that make up a record and optionally a sequence of characters that indicate the beginning or end of a record. With **Rule**, you can specify the following properties:
     - **Maximum non-blank line** defines the maximum number of non-blank lines that can be processed by a rule.
     - **Type of rule**: Can be one of:
       - **No text comparison** (The Maximum lines per record indicates a single logical record).
       - **Identify the beginning of record** (Marks the start of the single logical record).
       - **Identify the end of record** (Marks the end of the single logical record).
     - **Offset**: Specifies the location within a line where the Comparison String must occur.
     - **Comparison Test**: Can either be **Equals**, requiring a character sequence match at the specific offset, or **Does not equal**, indicating a particular character sequence does not occur at the specific offset.
     - **Comparison String** defines the character sequence to be compared.
   - **Regular Expression** identify a pattern that is used to indicate the beginning or end of a record. By using **Regular Expression**, you can specify the following properties:
     - **Comparison String** defines the character sequence to be matched.

       OR
     - Beginning or end of record:
       - **Identify the beginning of record** marks the start of the single logical record.
       - **Identify the end of record** marks the end of the single logical record.

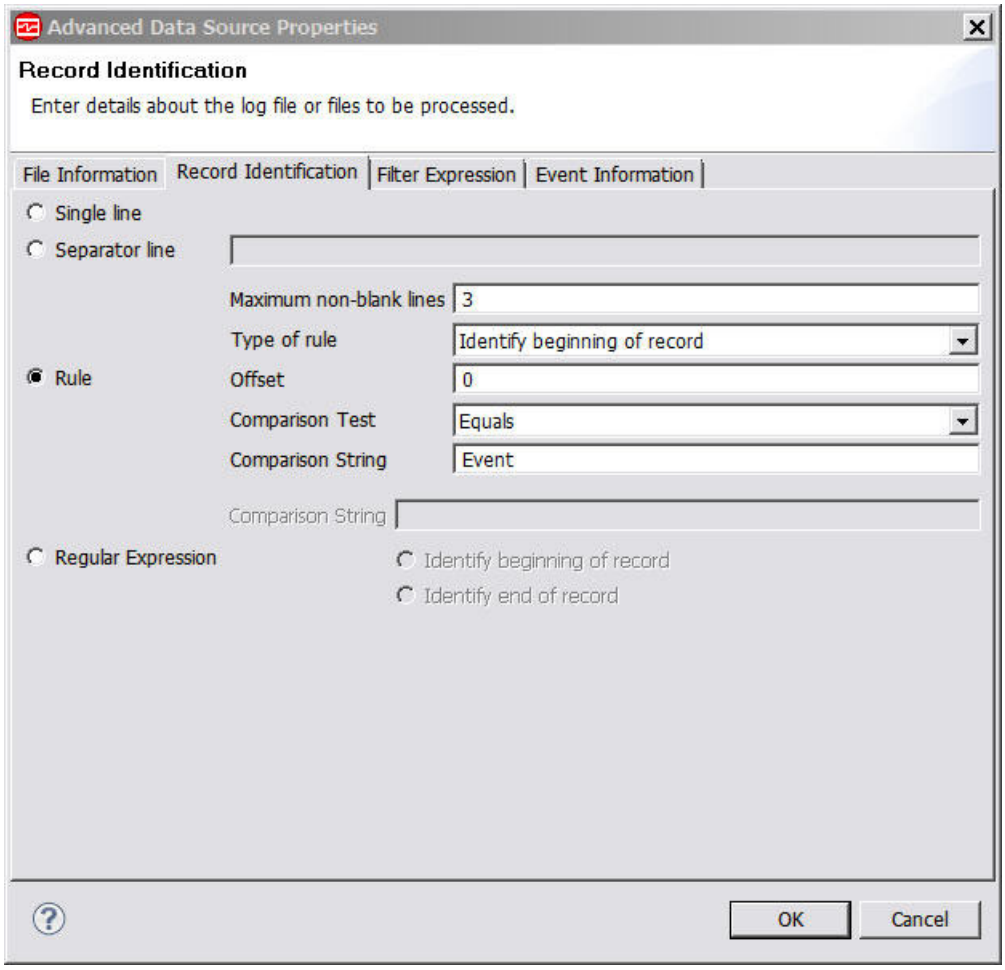


Figure 125. Advanced Data Source Properties page, Record Identification

k. If you did select **Process all records when the file is sampled** earlier, click the **Filter Expression** tab. By clicking **Filter Expression** you can filter the data that is returned as rows based on the values of one or more attributes, configuration variables or both. If you selected **Process new records appended to the file** earlier you cannot create a filter expression. For more information about filtering data from an attribute group, see ("Filtering attribute groups" on page 67).

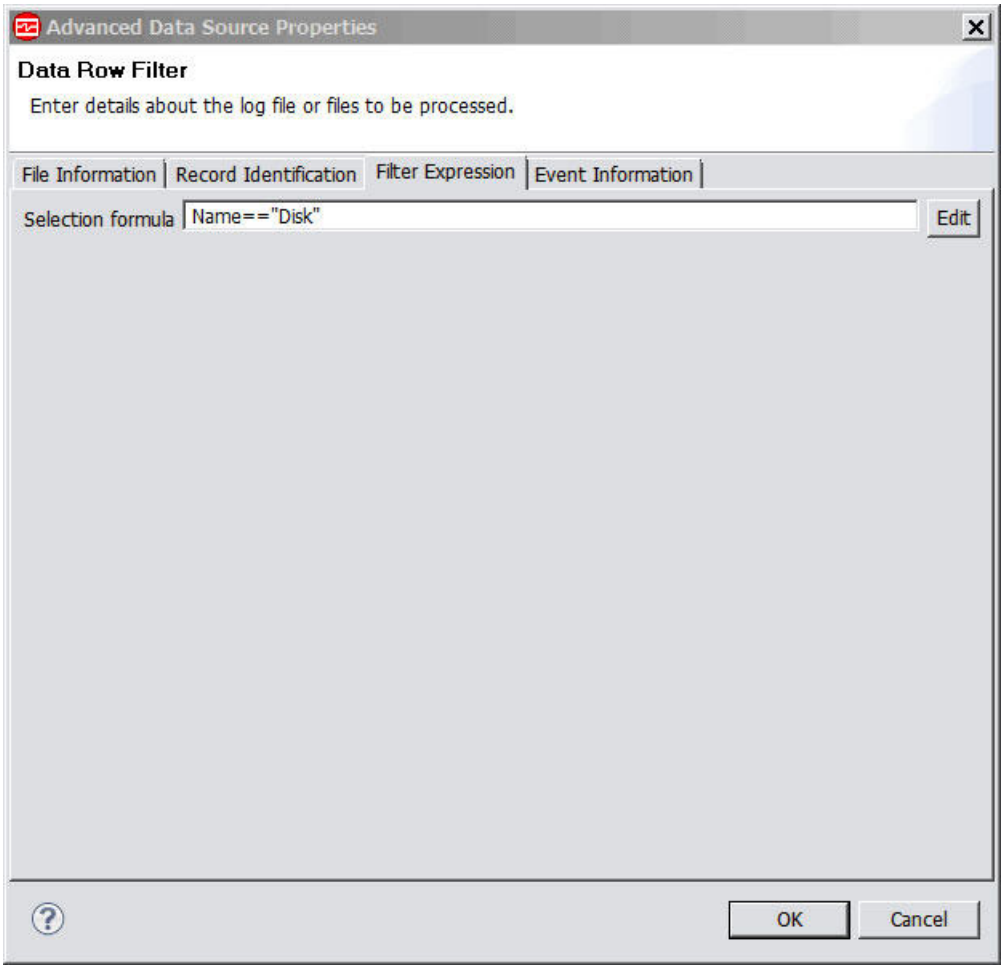


Figure 126. Advanced Data Source Properties page, Filter Expression tab

l. If you selected **Process new records appended to the file** earlier, click the **Event Information** tab Figure 127 on page 185 to select **Event Filtering and Summarization Options**. For more information, see (Chapter 34, "Event filtering and summarization," on page 417).

Note: The Summary tab can be present if the agent was created with an earlier version of Agent Builder. The summary tab is now deprecated by the Event Information tab

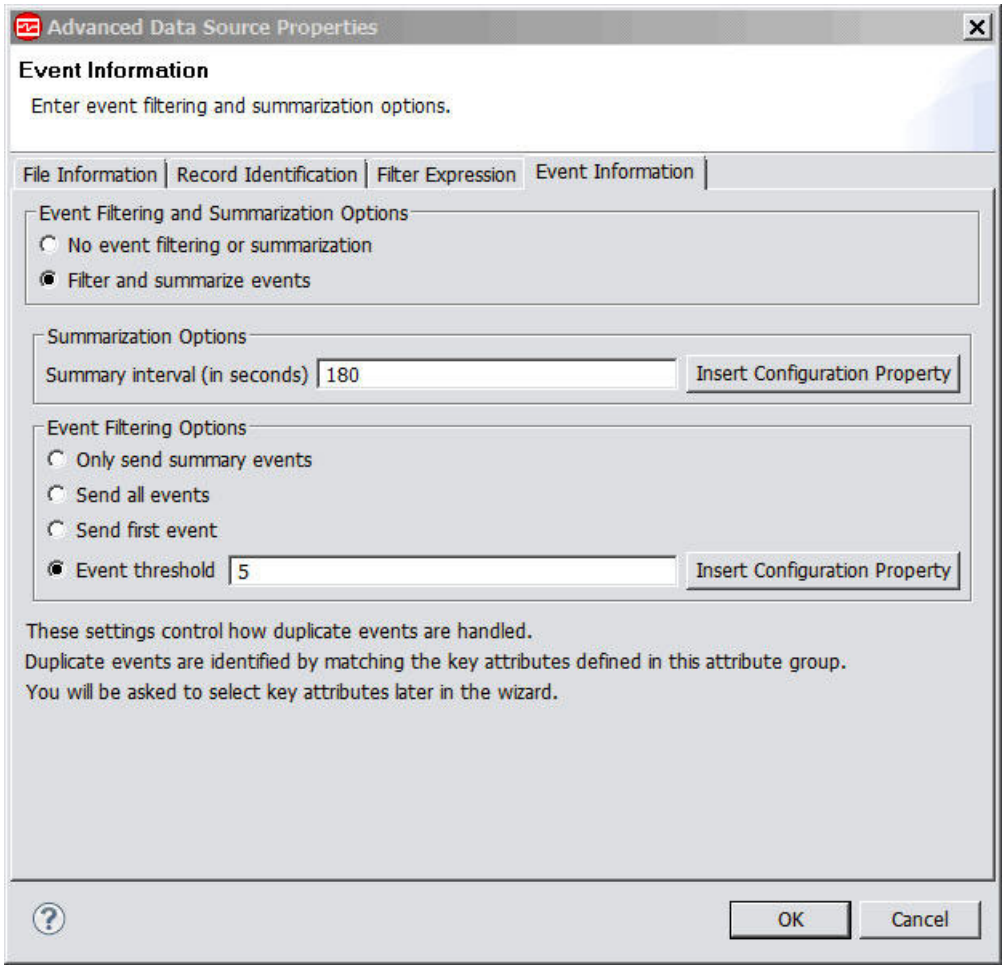


Figure 127. The Event Information tab of the Advanced Data Source Properties page,

7. Optional: Click **Test Log File Settings** on the Log File Information pageto start and test the agent (Figure 122 on page 178). Click **Test Log File Settings** after you select the options for the log source. For more information about testing, see (“Testing log file attribute groups” on page 193).
8. Use the following steps if you did not use the test function earlier and you typed the log file name in the **Log File Information** area of the Log File Information page:
   a. Click **Next** to display the Attribute Information page and define the first attribute in the attribute group.
   b. Specify the information, on the Attribute Information page, and click **Finish**.

   Note: When a log file attribute group is added to an agent at the default minimum Tivoli Monitoring version (6.2.1) or later, a `Log File Status` attribute group is included. For more information about the `Log File Status` attribute group, see (“Log File Status attribute group” on page 585).

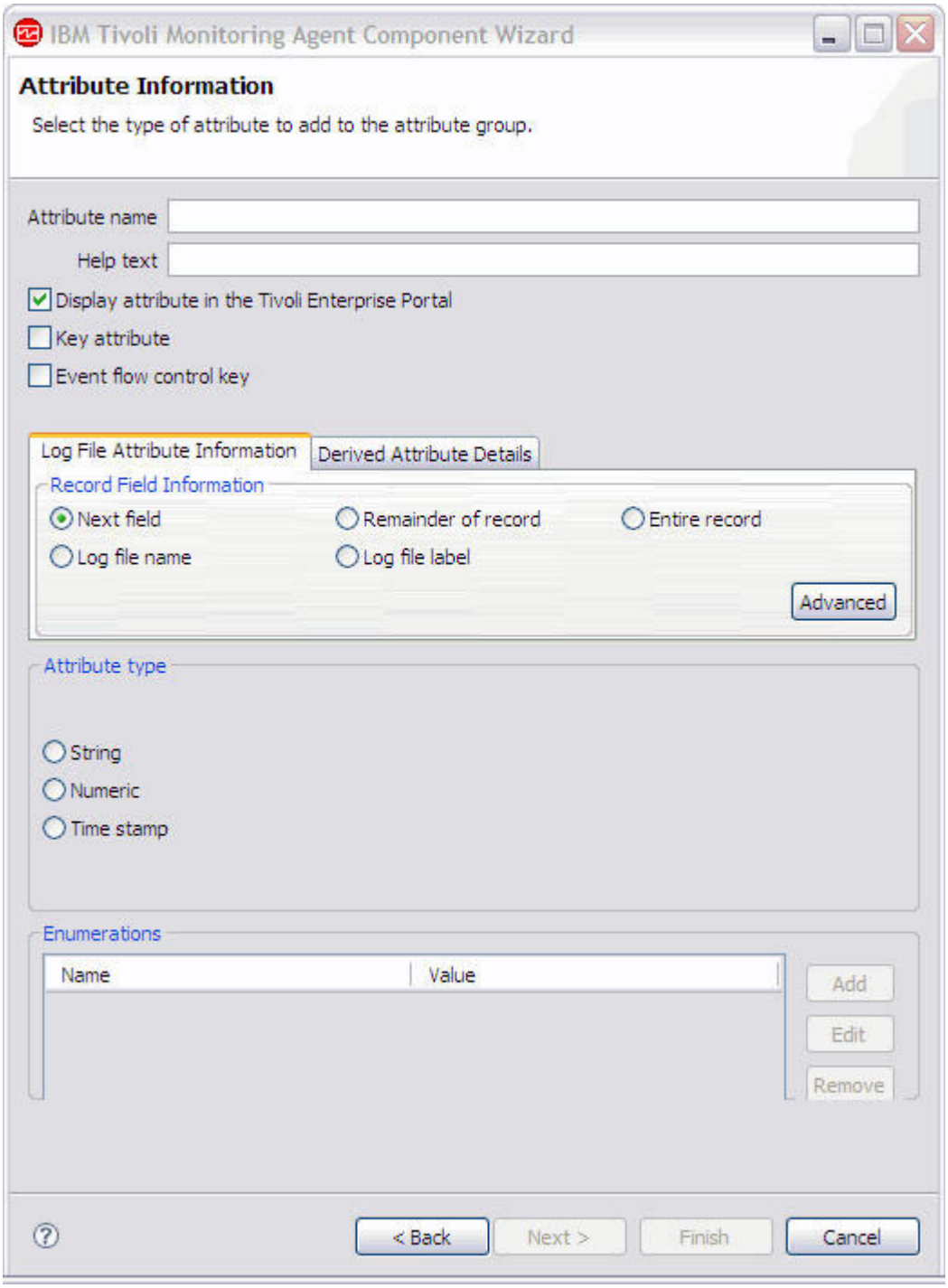


Figure 128. Attribute Information page

Along with the fields applicable to all data sources, the Attribute Information page for the log file data source has some additional fields in the **Record Field Information** area.

The **Record Field Information** fields are:

Next field
Shows the next field after parsing, by using the delimiters from the attribute group (or special delimiters for this attribute from the Advanced dialog).

Remainder of record
Shows the rest of the record after previous attributes are parsed. This attribute is the last attribute, except for possibly the log file name or log file label.

Entire record
Shows the entire record, which can be the only attribute, except for possibly the log file name or log file label.

Log file name
Shows the name of the log file.

Log file label
Shows the label that is assigned to the file on the advanced panel.

Note: Use the **Derived Attribute Details** tab only if you want a derived attribute, and not an attribute directly from the log file.

9. Click **Advanced** in the **Record Field Information** area to display the Advanced Log File Attribute Information page (Figure 129).

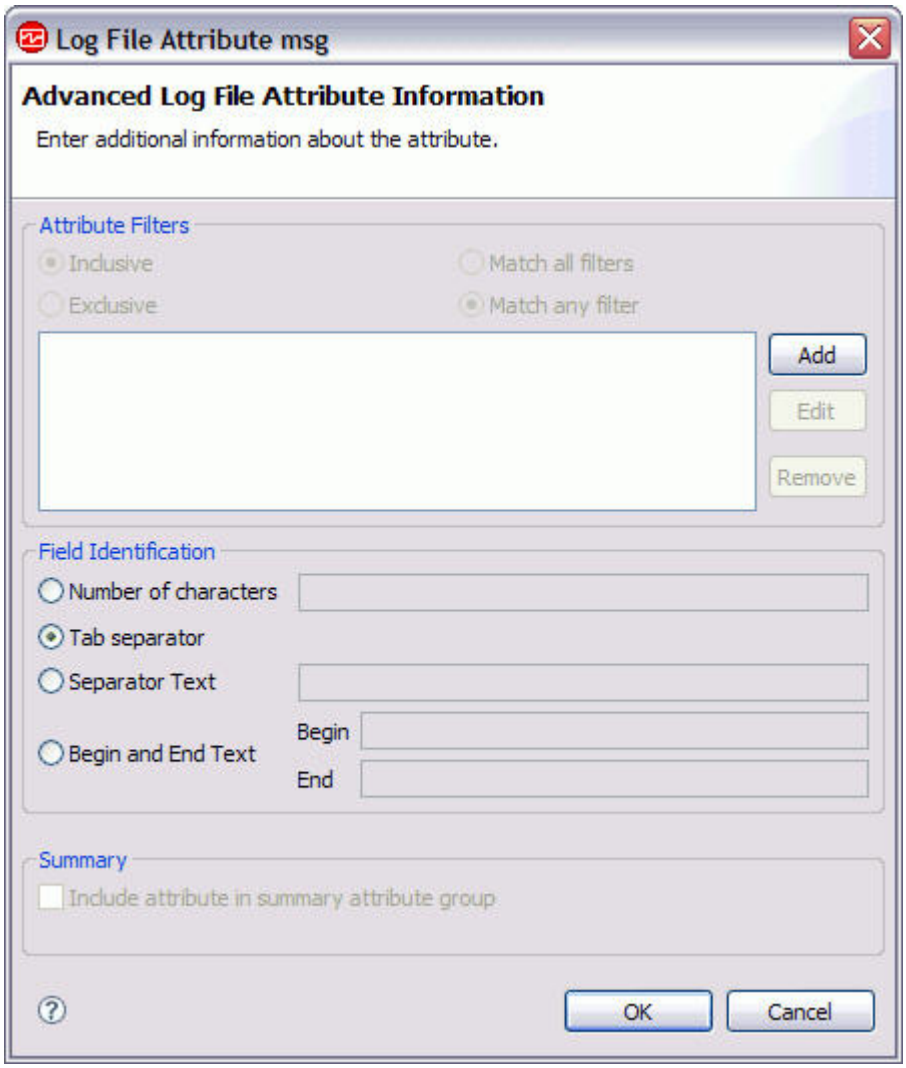


Figure 129. Advanced Log File Attribute Information page

a. In the **Attribute Filters** section, specify the criteria for data to be included or excluded. Filtering attributes can enhance the performance of your solution by reducing the amount of data processed. Click one or more of the attribute filters:
   - **Inclusive** indicates that the attribute filter set is an acceptance filter, meaning that if the filter succeeds, the record passes the filter, and is output.
   - **Exclusive** indicates that the attribute filter set is a rejection filter, meaning that if the attribute filter succeeds, the record is rejected, and is not output.
   - **Match all filters** indicates that all filters defined to the filter must match the attribute record in order for the filter to succeed.
   - **Match any Filter** indicates that if any of the filters that are defined to the filter match the attribute record, the filter succeeds.

b. Use **Add**, **Edit**, and **Remove** to define the individual filters for an attribute filter set.

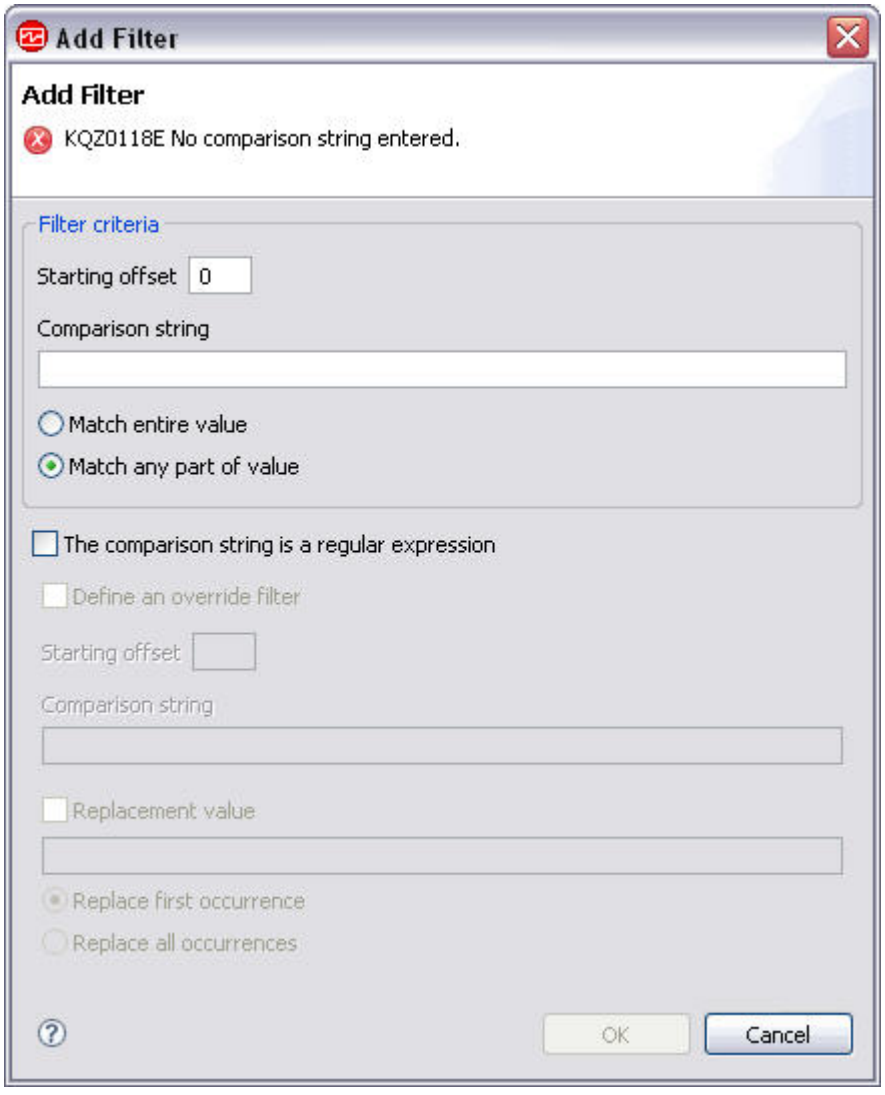


Figure 130. Add Filter window

c. To add a filter, follow these steps:
   1) Click **Add**, and complete the options in the Add Filter window (Figure 130) as follows:
      a) The **Filter criteria** section defines the base characteristics of the filter, including the following properties:

- **Starting offset** defines the position in the attribute string where the comparison is to begin.
- **Comparison string** defines the pattern string against which the attribute is defined.

  Type a string, pattern, or regular expression that is used by the agent to filter the data read from the file. The records that match the filter pattern are eliminated from the records that are returned to the Tivoli Enterprise Portal, or are the only records returned. The result depends on whether you choose for the filter to be inclusive or exclusive.
- **Match entire value** checks for an exact occurrence of the comparison string in the attribute string. Checking starts from the starting offset position.
- **Match any part of value** checks for the comparison string anywhere in the attribute string. Checking starts from the starting offset position.

b) **The comparison string is a regular expression** indicates that the comparison string is a regular expression pattern that can be applied against the attribute string.

   Regular expression-filtering support is provided by using the International Components for Unicode (ICU) libraries to check whether the attribute value examined matches the specified pattern.

   To effectively use regular expression support, you must be familiar with the specifics of how ICU implements regular-expressions. This implementation is not identical to how regular expression support is implemented in `Perl`, `grep`, `sed`, Java regular expressions, and other implementations. See Appendix G, "ICU regular expressions," on page 627 for guidance on creating regular expression filters.

c) **Define an override filter** indicates that you want to provide a more specific filter comparison that overrides the base characteristics previously defined. This additional comparison string is used to reverse the filter result. When the filter is Inclusive, the override acts as an exclusion qualifier for the filter expression. When the filter is Exclusive, the override acts as an inclusion qualifier for the filter expression. (For more about Inclusive and Exclusive, see step 9 on page 187, and the examples that follow). The override filter has the following properties:
   - **Starting offset** defines the position in the attribute string where the comparison is to begin.
   - **Comparison string** defines the pattern string against which the attribute is matched.

     Type a regular expression that is used by the agent to filter the data read from the file. The records that match the filter pattern are eliminated from the records that are returned to the Tivoli Enterprise Portal, or are the only records returned. The result depends on whether you choose for the filter to be inclusive or exclusive.

d) **Replacement value** can be used to alter the raw attribute string with a new value. See Appendix G, "ICU regular expressions," on page 627 for more details about special characters that can be used.

e) **Replace first occurrence** replaces the first occurrence that is matched by the comparison string with new text.

f) **Replace all occurrences** replaces all occurrences that are matched by the comparison string with new text.

2) Click **OK**.

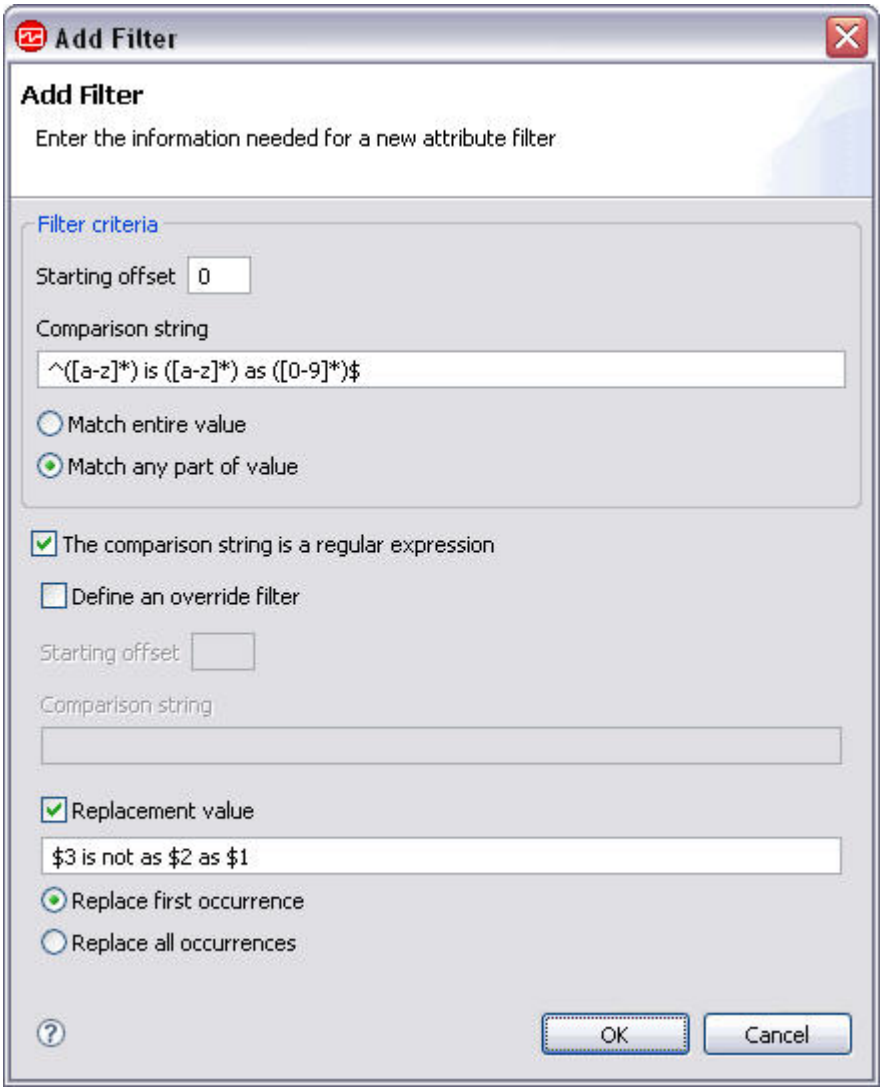


Figure 131. Add Filter example 1

If the attribute string is `abc is easy as 123`, then the replaced string that is displayed in the Tivoli Enterprise Portal as `123 is not as easy as abc`.

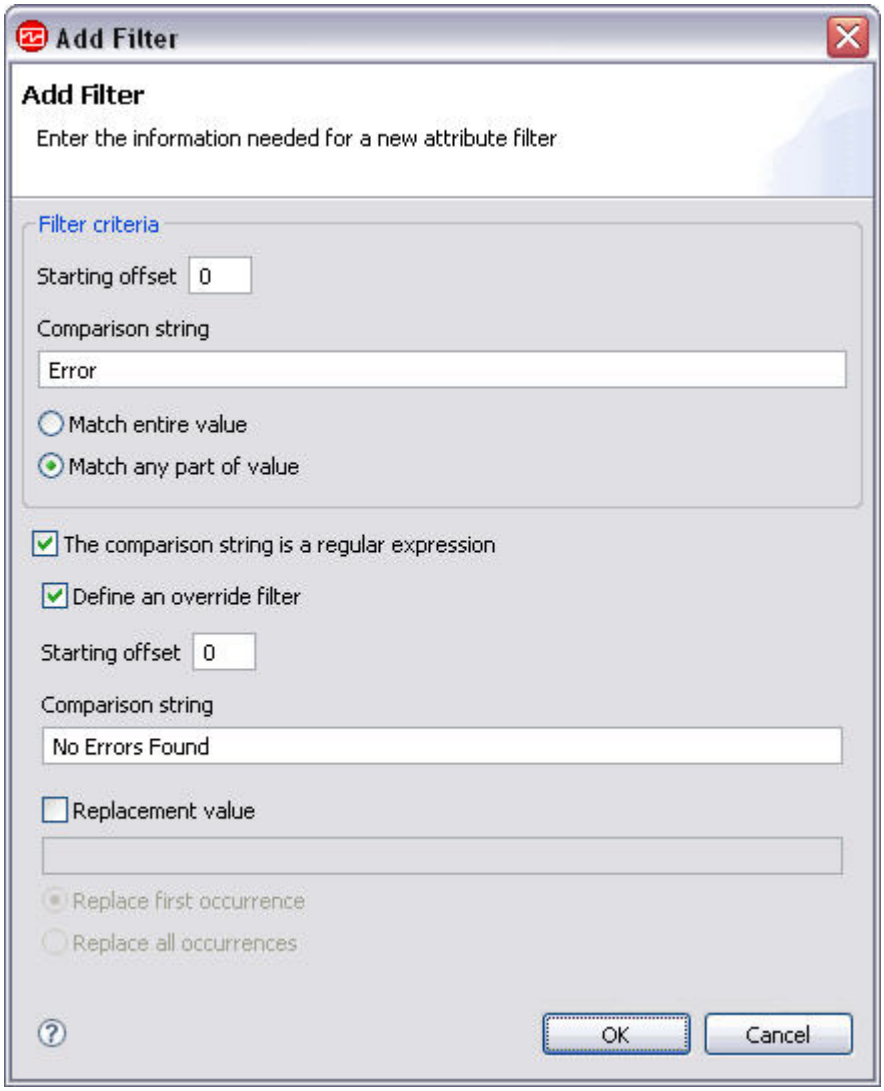


Figure 132. Add Filter example 2

If the attribute string is `Unrecoverable Error reading from disk`, and the filter is Inclusive, then the attribute is displayed in the Tivoli Enterprise Portal. If the attribute string is `No Errors Found during weekly backup` and the filter is Inclusive, then the attribute is not displayed in the Tivoli Enterprise Portal.

d. In the **Field Identification** section of the Advanced Log File Attribute Information page (Figure 129 on page 187), specify how to override the attribute group field delimiters for this one attribute only. Click one of the attribute filters, and complete the required fields for the option:
   - **Number of characters**: Enter the limit for the number of characters.
   - **Tab separator** specifies the use of tab separators.
   - **Separator Text**: Enter the separator text that you want to use.
   - **Begin and End Text** Enter both **Begin** text and **End** text.

e. In the **Summary** section of the Advanced Log File Attribute Information page (Figure 129 on page 187), click the **Include attribute in summary attribute group** check box to add the attribute to the summary attribute group. This attribute group is produced when a user turns on log attribute summarization.

f. Click **OK**.

10. If you used the test function in step (7 on page 185), the Select key attributes page is displayed. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see (“Selecting key attributes” on page 26).
11. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

    Note: When a log file attribute group is added to an agent with the default minimum Tivoli Monitoring version (6.2.1) or later, a `Log File Status` attribute group is included. For more information about the `Log File Status` attribute group, see (“Log File Status attribute group” on page 585).

Log file parsing and separators

You can change the default separator that is used to separate one or more attributes in a log file record.

When you create a log file attribute group, a separator is by default assigned. The default separator is a tab. The separator is used by the agent to parse and delimit the data for each attribute in the data row. You can change the default attribute separator to be:

- A fixed number of characters
- A space
- A different character or characters
- A specific beginning and end text
- An XML element.

You change the default separator that is used for all attributes in the group in the following ways:

1. When you are creating the attribute group, on the Log File Information page.
2. After you create the attribute group, by opening the **Agent Editor** > **Data Sources** tab, selecting the attribute group and choosing a separator in the Field Identification area.

You can also optionally assign specific separators to one or more individual attributes. You can assign specific separators for individual attributes to use:

- A fixed number of characters.
- A tab separator
- A space separator
- A different character or characters
- A specific beginning and end text.

You change the separator that is used for individual attributes in the following ways:

1. By selecting **Advanced** on the Attribute Information page when you are creating an attribute.
2. By opening the **Agent Editor** > **Data Sources** tab, selecting the attribute and selecting **Advanced** on the Log File Attribute Information tab.

Example 1 - Simple log file output

Some log file records have clear and regular separators, for example:

```
one,two,three
```

Here the `","` character is a clear and regular separator between the three pieces of data on the row. In this case, select **Separator Text** and specify `","` as the default separator for the attribute group. There is no need to change or define other separators.

Defining this separator for a log file that contains the data row that is shown earlier in this example is shown in the following output:

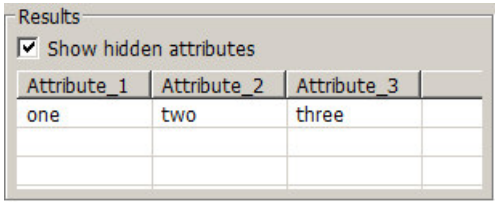


Figure 133. Example attribute value output when Agent parses a simple log file data row.

Example 2 - Complex log file output

Some log files can contain data rows that have irregular or changing separators, for example:

```
one,two,three,[four]12:42,five
```

In this example an assignment of separators to attribute definitions that you can use is:

1. In the previous example you set the default separator to `","`. This separator is used for all attributes unless you over-ride it with a specific separator. In this example the default separator of `","` is correct to use again for the first three attributes in the row.
2. For the fourth attribute, assume the string between the`"["` and `"]"` is a value that you want to extract. In this case when you define the fourth attribute, you assign a separator type **Begin and End Text** with begin and end text values of `"["` and `"]"`.
3. For the fifth attribute, assume that you want to extract the values between the `"]"` and `":"` characters. In this case when you define the fifth attribute, you assign separator type **Separator Text** set to `":"`.
4. For the sixth attribute, your default attribute group separator `","` is fine again.
5. For the seventh attribute, you do not need to specify a separator as it is the last attribute.

Defining these separators on a log file that contains the data row that is shown earlier in this example is shown in the following output:

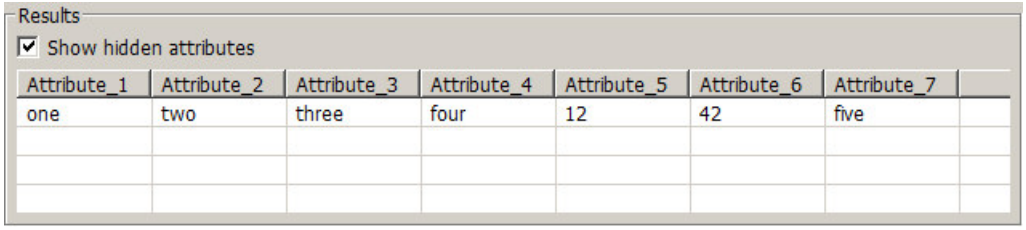


Figure 134. Example attribute value output when Agent parses a complex log file data row.

The procedure to define the attribute separators is described under step 5 on page 178 of (Chapter 16, “Monitoring a log file,” on page 177).

Testing log file attribute groups

You can test the log file attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test Log File Settings** on the Log File Information page (Figure 122 on page 178).

- After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test Log File Settings**. For more information about the Agent Editor, see (Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35).

After you click **Test Log File Settings** in one of the previous two steps, the Parse Log window opens.

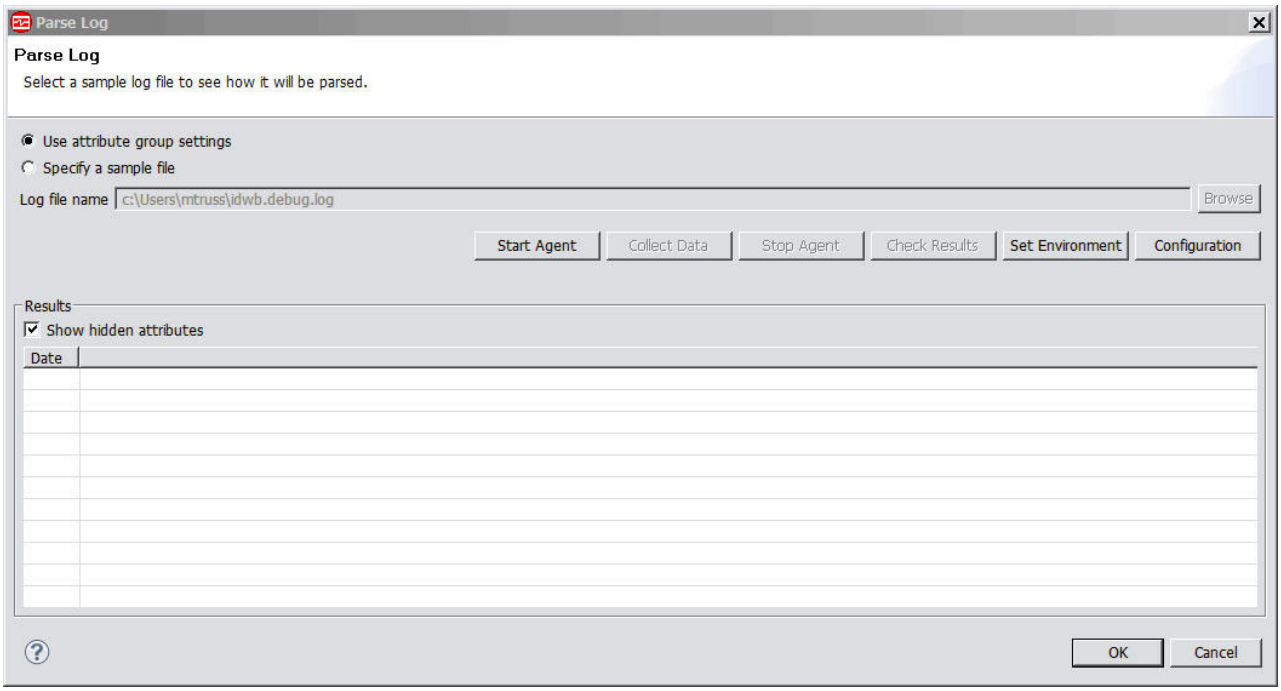


Figure 135. Parse Log window

2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see ("Attribute group testing" on page 377).
3. Click **Start Agent**. A window opens indicating that the Agent is starting. When the agent starts, it monitors the configured log file for new records
4. To test your agent's data collection, generate new records in the monitored log file. When new records are added to the log file, the agent parses them according to its configuration and updates the corresponding attribute values in its cache.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The Parse Log window collects and shows any new attribute values in the agent's cache since it was last started. An example data collection is shown in Figure 136 on page 195

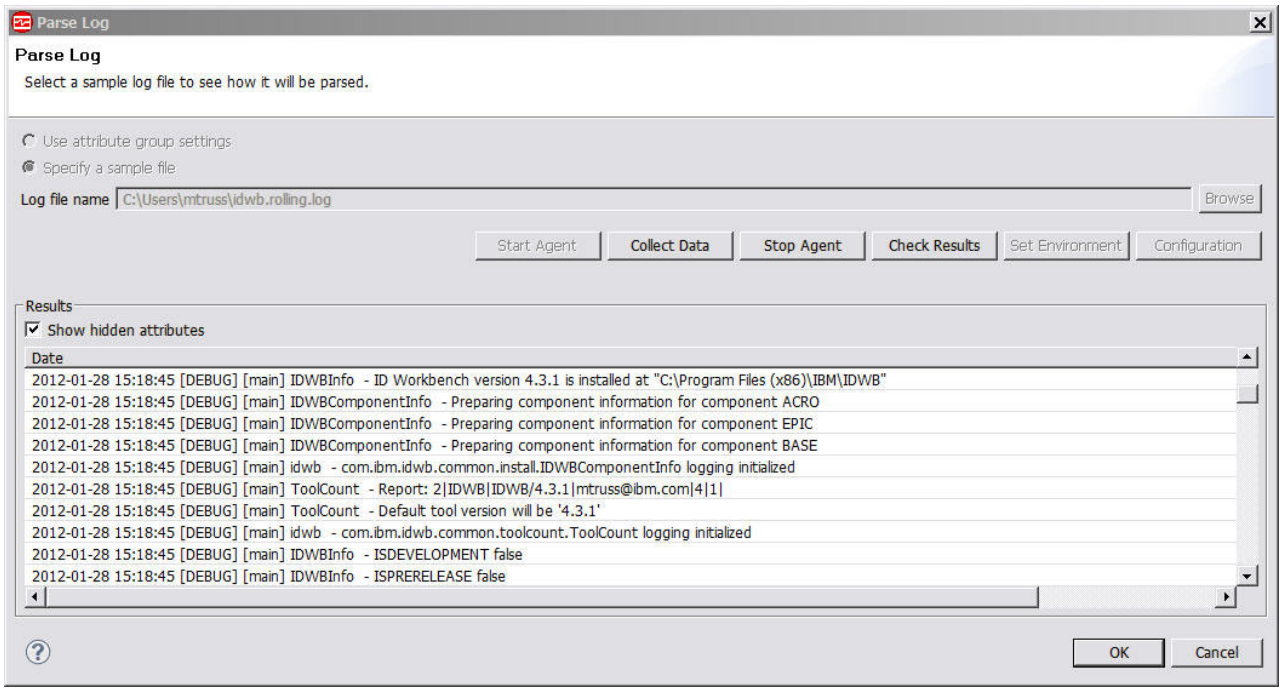


Figure 136. Parse Log window that shows parsed log file attribute values

6. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. An example is shown in the (Figure 137). The data that is collected and shown by the Data collection Status window is described in "Performance Object Status node" on page 545

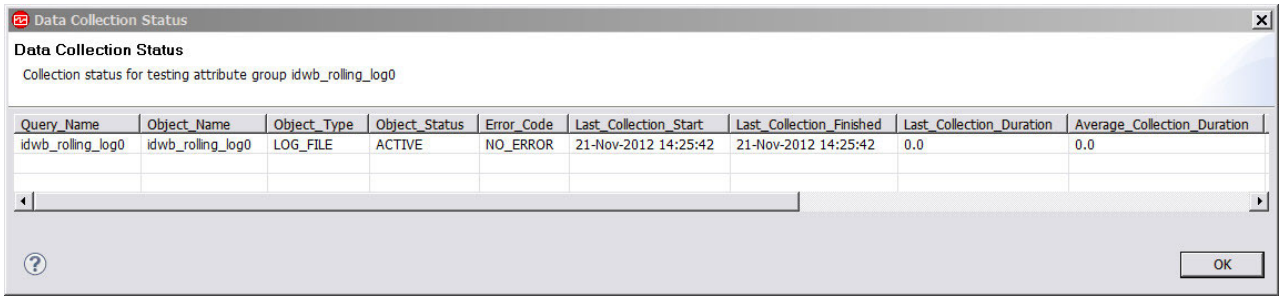


Figure 137. Data Collection Status window

7. The agent can be stopped by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the Parse Log window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, "Testing and debugging your agent," on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 17. Monitoring an AIX Binary Log

You can monitor AIX binary error logs through the `errpt` command.

About this task

Log Monitoring supports the monitoring of AIX binary error logs through the `errpt` command. The `errpt` command generates an error report from entries in an error log. It includes flags for selecting errors that match specific criteria. This support for the monitoring of AIX binary error logs through the `errpt` command is modeled on the support for the same function in the Tivoli Monitoring UNIX Logs Agent (product code `kul` or `ul`).

When you supply the Agent Builder with an **`errpt`** command string, it processes the events that result from running this command. Agent Builder enforces the same constraints on this command that the Monitoring Agent for UNIX Logs does. In particular, you must use the **`-c`** (concurrent mode) option so that the command runs continuously, and you cannot use the **`-t`** option or the following options that result in detailed output: **`-a`**, **`-A`**, or **`-g`**.

An Agent Builder agent that monitors the AIX **`errpt`** command automatically includes the same information as a Monitoring Agent for UNIX Logs does. For more information about the attribute groups for AIX binary error logs, see "AIX Binary Log attribute group" on page 558.

Procedure

1. On the Agent Initial Data Source page (Figure 150 on page 214) or the Data Source Location page, click **Logged data** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

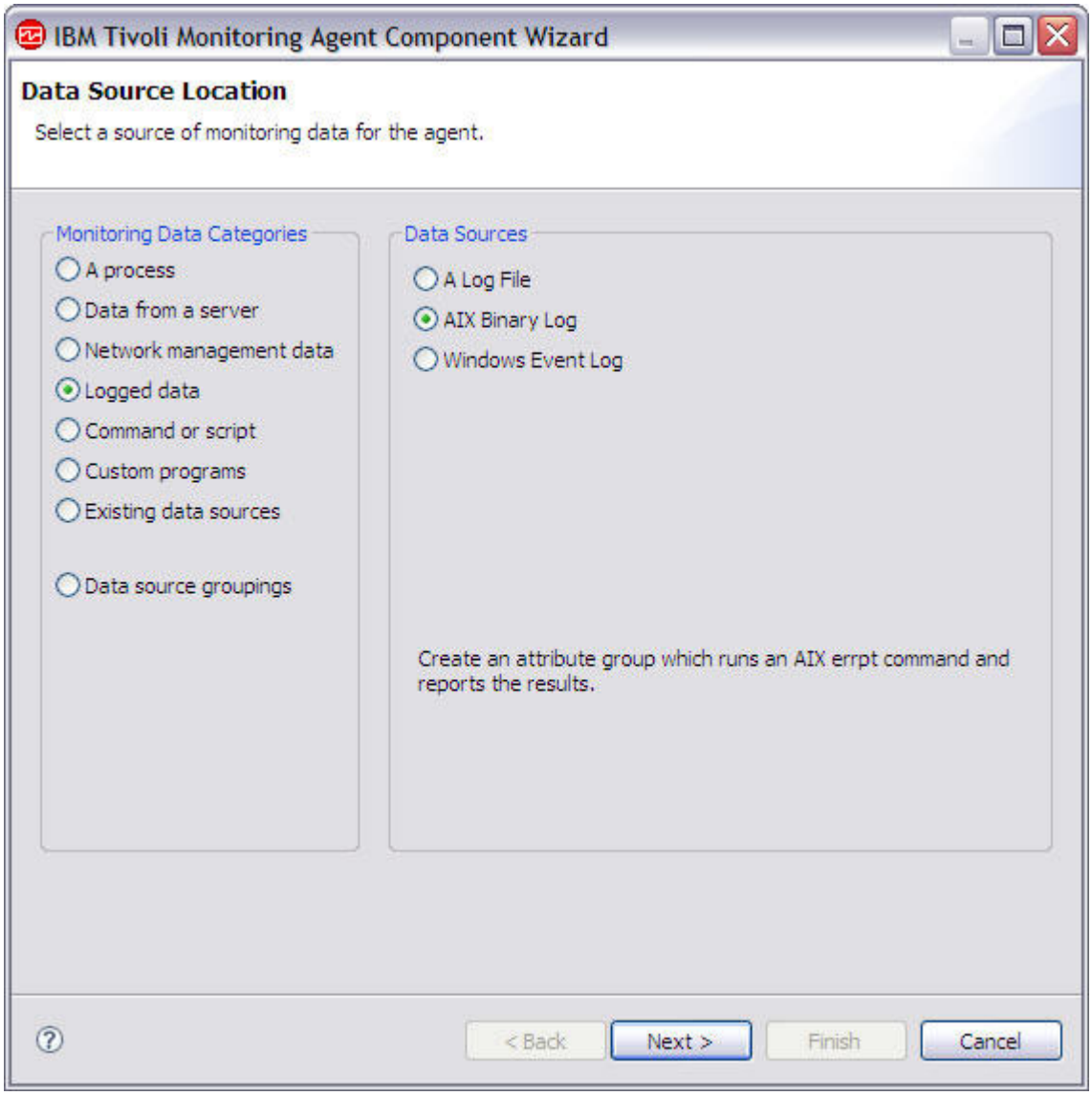


Figure 138. Adding an AIX Binary Log

2. In the **Data Sources** area, click **AIX Binary Log**.
3. Click **Next**.
4. On the Binary Log Information page Figure 139 on page 199, enter an `errpt` command. The default value is:

   ```
   errpt -c -smmddhhmmyy
   ```

   The agent searches for the 'mmddhhmmyy' string and replaces it with the actual date and time on startup. Only the first occurrence of the string is replaced.

   You can supply your own `errpt` command but Agent Builder enforces the same constraints on this command that the Monitoring Agent for UNIX Logs does. In particular, you must use the **-c** (concurrent mode) option so that the command runs continuously, and you cannot use the **-t** option or the following options that result in detailed output: **-a**, **-A**, or **-g**.

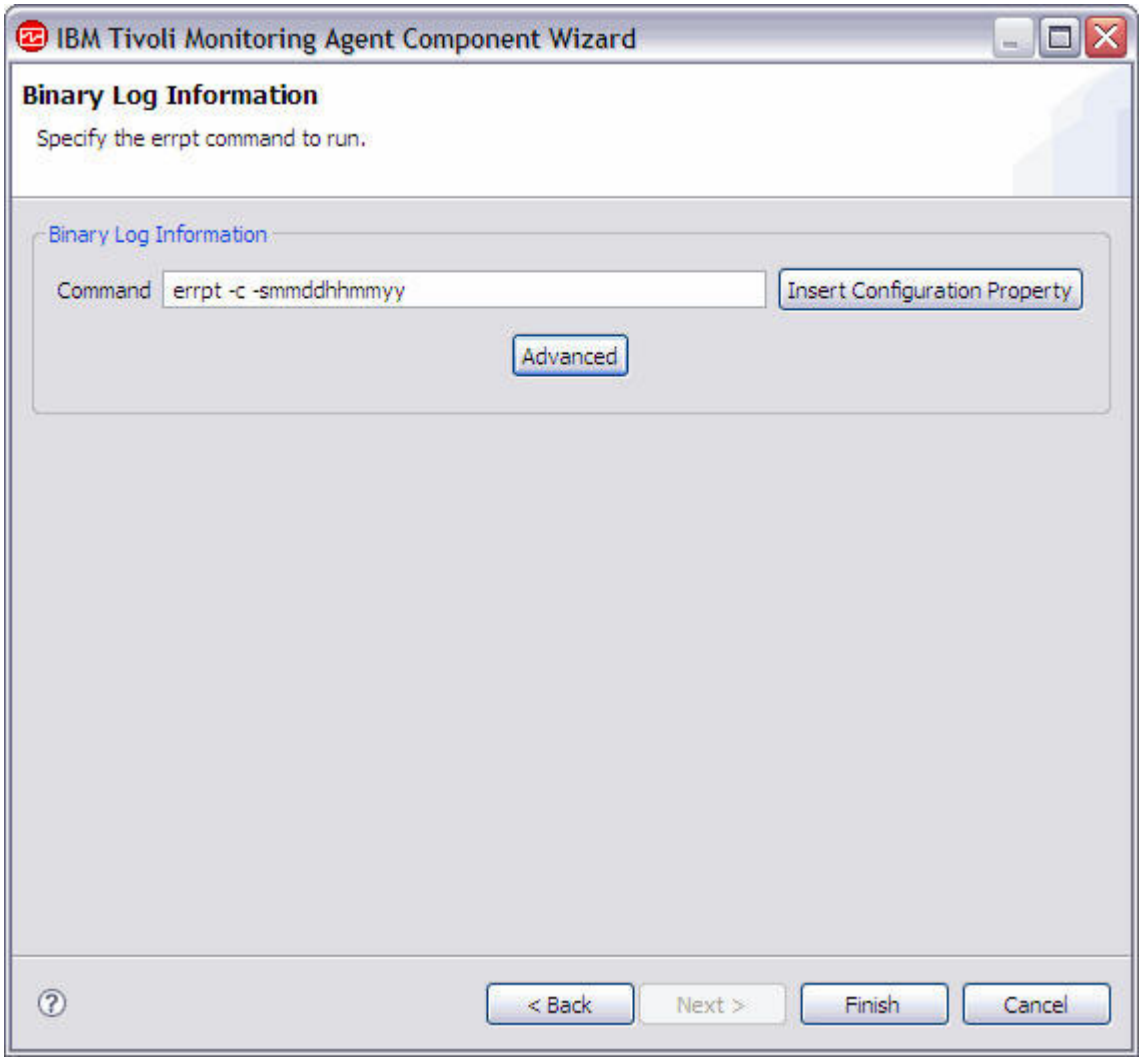


Figure 139. Binary Log Information page

5. (Optional) Click **Advanced** to select filtering and summarization options for events. For more information, see "Controlling duplicate events" on page 417.
6. Do one of the following steps:
   - If you are using the New Agent wizard, click **Next**.
   - Click **Finish** to save the data source and open the Agent Editor.

Related reference:

"AIX Binary Log attribute group" on page 558
The AIX Binary Log attribute group displays events from the AIX Binary Log as selected by the provided `errpt` command string.

Chapter 18. Monitoring a Windows Event Log

Define a data source to collect data from a Windows event log.

About this task

You can collect data from the Windows event log by using the type, source, or ID of events. You use these parameters to filter the log events that the Windows system gathered. The agent compares each new event in the monitored event log against the specified filter. If the event matches one of the event types, event sources, and event IDs specified in the filter, it passes.

For example, if the Event log filter is for the Application log, specify **Error** as the event type. This choice matches all events that are logged to the Application log with an event type value of `error`. If you add the **Diskeeper** and **Symantec AntiVirus** event sources, the agent matches all error events from either of these sources. You can add specific event IDs to refine the filter further. No direct association exists between the event type, event source, and event ID. If one of the values for each matches an event, the event matches.

By default, only events that are generated after the agent starts are processed. However, you can enable the agent when it restarts to process log events that are generated while the agent is shut down. For more information about enabling the agent to process events generated while the agent is shut down, see step (6 on page 203).

Procedure

1. On the Agent Initial Data Source page (Figure 140 on page 202) or the Data Source Location page, click **Logged Data** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

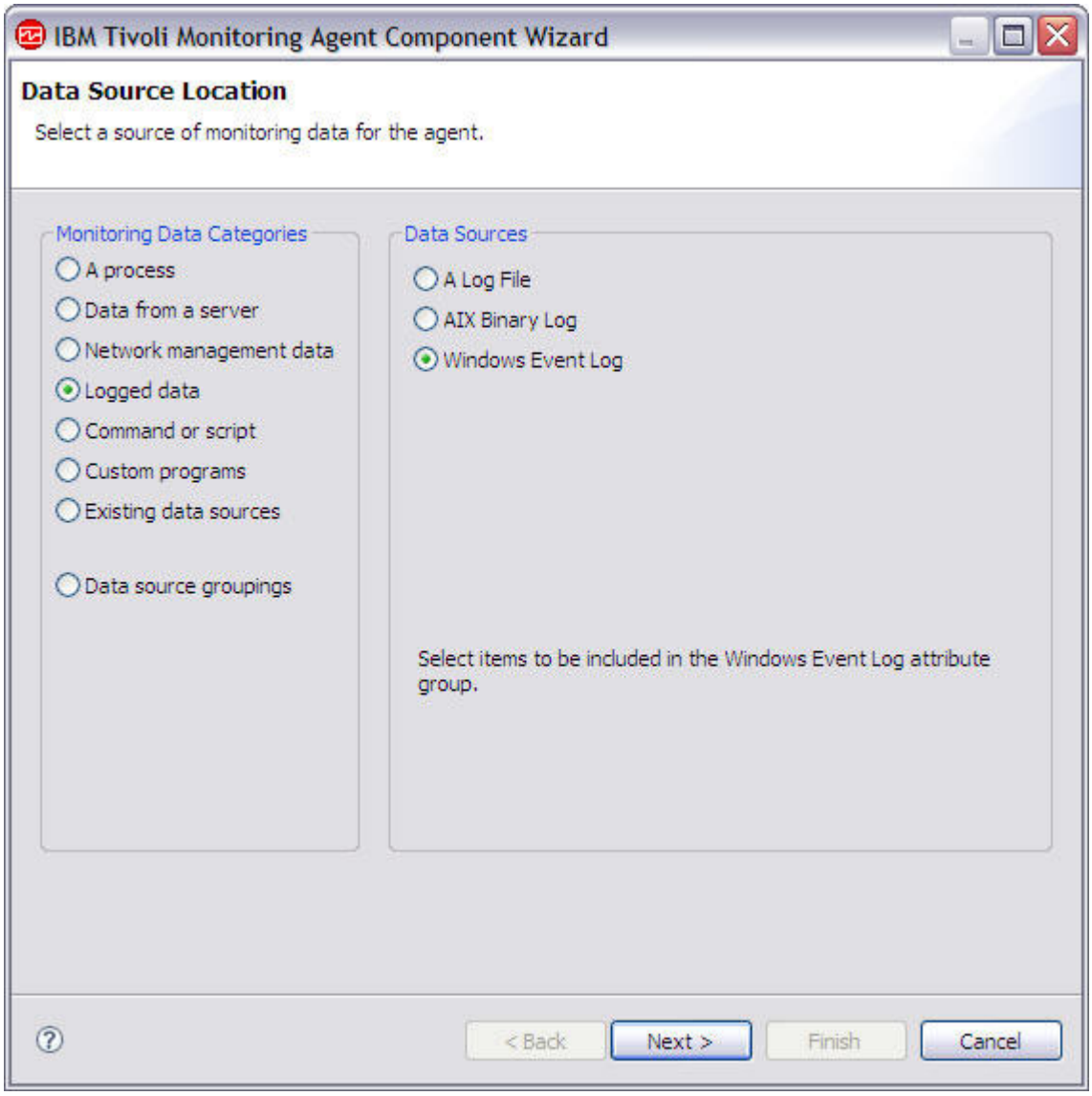


Figure 140. Adding a Windows Event log

2. In the **Data Sources** area, click **Windows Event Log**.
3. Click **Next**.
4. On the Windows Event Log page (Figure 141 on page 203), select the name from one of the logs in the **Windows Event Log name** list, or type a name for the event log.

   The list is constructed from the set of logs on the current system, for example:

   ```
   Application
   Security
   System
   ```

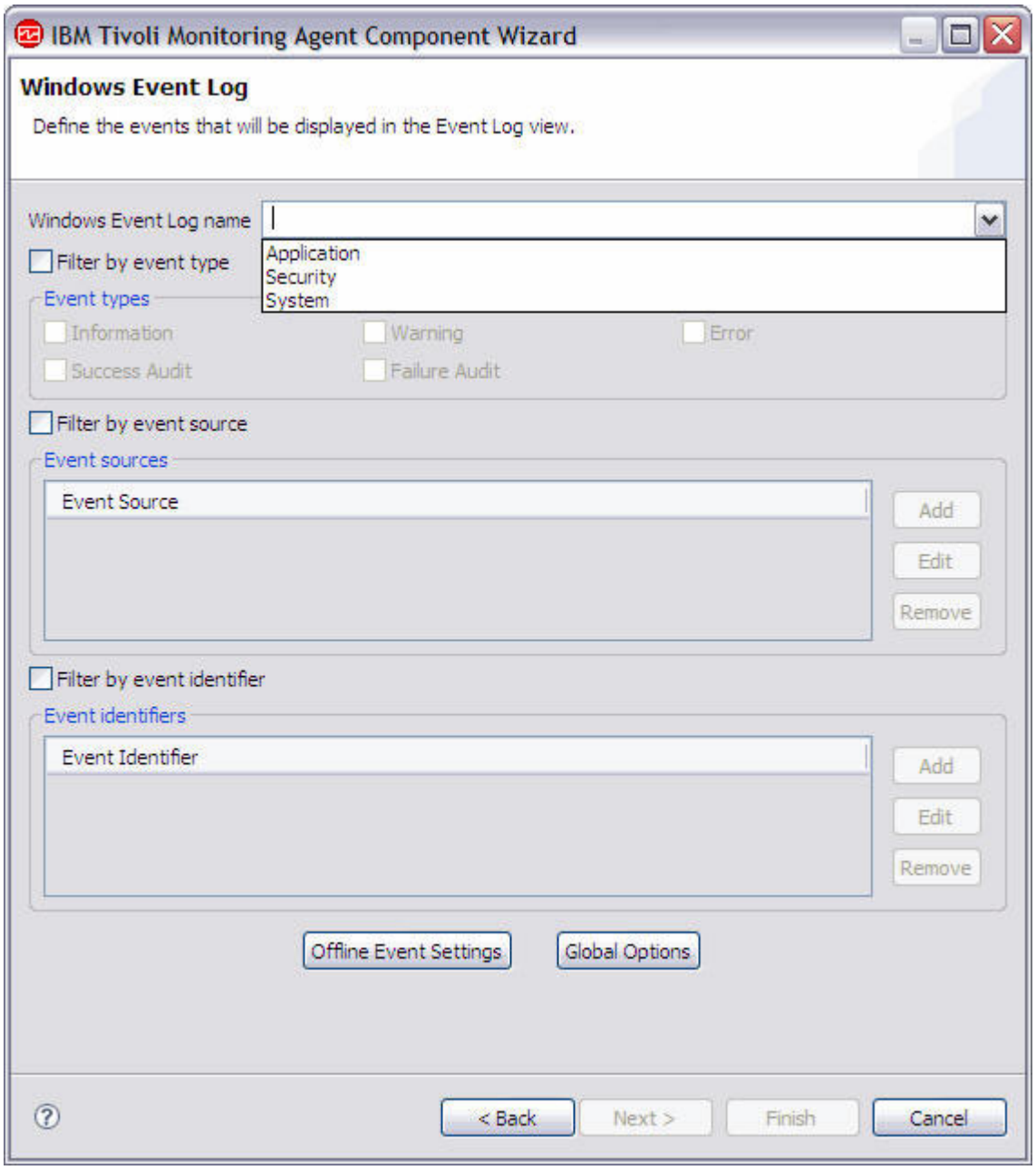


Figure 141. Windows Event Log page

5. In the Windows Event Log page, specify whether you want to filter the results by using one or more of the following mechanisms:
   - "Filtering by event type" on page 206
   - "Filtering by event source" on page 207
   - "Filtering by event identifier" on page 209

   Note: You must select at least one of these filter criteria.
6. To process log events that are generated while the agent is shut down, on an agent restart, click **Offline Event Settings** on the Windows Event Log page. The Windows Event Log Bookmark Settings window opens.

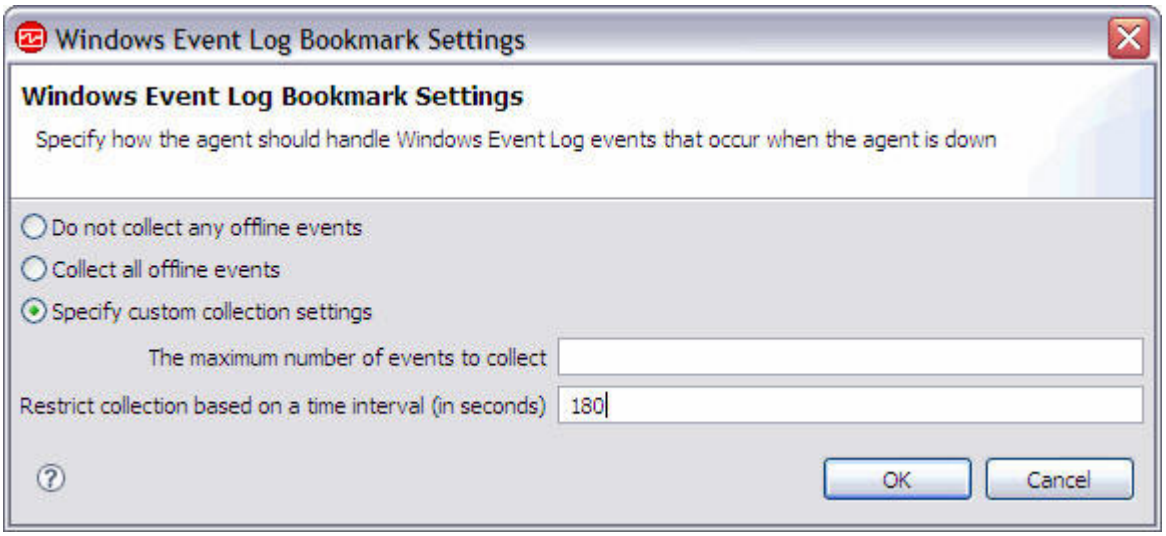


Figure 142. Windows Event Log Bookmark Settings window

7. Select one of the following bookmarking options:

   Note: These options apply to all Windows event logs being monitored.

   - **Do not collect any offline events**: Events that are generated while the agent is shut down are not processed. This option is the default option.
   - **Collect all offline objects**: All events that are generated while the agent is shut down are processed.
   - **Specify custom collection settings**: You can enter a value to throttle the processing of old events that are based on a time value, or number of events, or both. By using this option, you ensure that the Tivoli Enterprise Monitoring Server is not overloaded with events when the agent starts.

     For example, if 100 is entered in **The maximum number of events to collect** field and 30 is entered in the **Restrict collection based on a time interval (in seconds)** field. The number of events that are processed is either the last 100 events that are generated before the agent starts, or any event that is generated within 30 seconds of agent start. Which result depends on the variable that is matched first.

     When you enter a value for the maximum number of events to collect, the *CDP_DP_EVENT_LOG_MAX_ BACKLOG_EVENTS* environment variable is added. When you enter a value to restrict collection that is based on a time interval, the *CDP_DP_EVENT_LOG_MAX_BACKLOG_TIME* environment variable is added. When either or both of these variables are added, the *eventlogname_productcode_instancename_subnodename*.rst file is created containing the last event record that is processed for the event log. This file is in the *%CANDLE_HOME%*\tmaitm6\logs directory and is used when the agent is restarted to process old events that are generated while the agent was shut down.
8. If you want to set global options for the data source, click **Global Options** on the Windows Event Log page The Global Windows Data Source Options window opens.

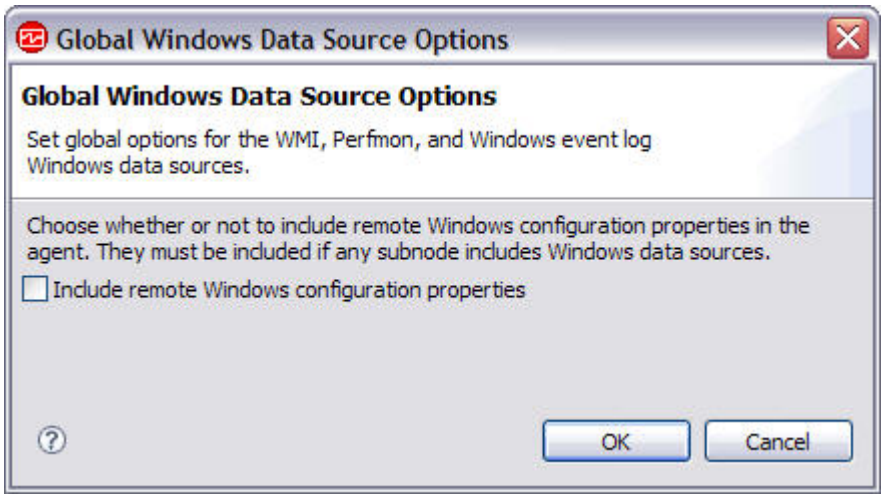


Figure 143. Global Windows Data Source Options window

9. Select the **Include remote Windows configuration properties** check box if you want to include this option, and click **OK**.

   For information about Windows remote connection configuration for Windows data sources, see ("Configuring a Windows remote connection" on page 372).

10. After you specify the filter and click **OK**, on the Windows Event Log page, do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor. The name of the new Windows Event Log is shown on the Agent Editor Data Source Definition page.

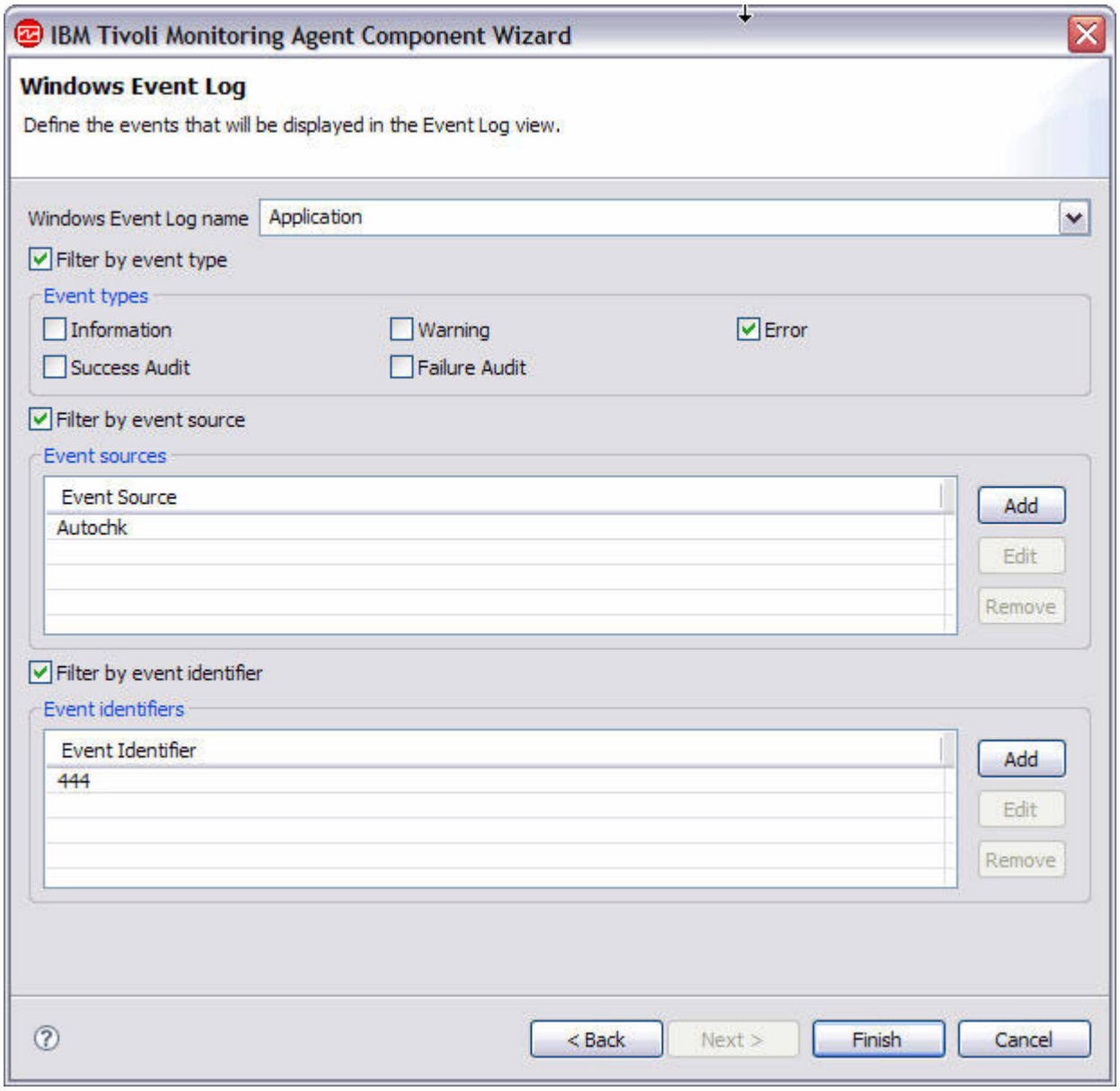


Figure 144. Windows Event Log page

What to do next

For information about Windows remote connection configuration for Windows Event Log data sources, see ("Configuring a Windows remote connection" on page 372).

Filtering by event type

Filter Windows Event Log results by event type

Procedure

1. In the Windows Event Log page (Figure 144), select **Filter by event type**.
2. Select one or more of the following Event types:
   - **Information**
   - **Warning**

- **Error**
- **Success Audit**
- **Failure Audit**

3. Click **Finish** to complete.

Filtering by event source

Filter Windows Event Log results by event source

Procedure

1. Select **Filter by event source** and click **Add** in the **Event sources** area of the Windows Event Log page (Figure 144 on page 206). The Event Source window opens.

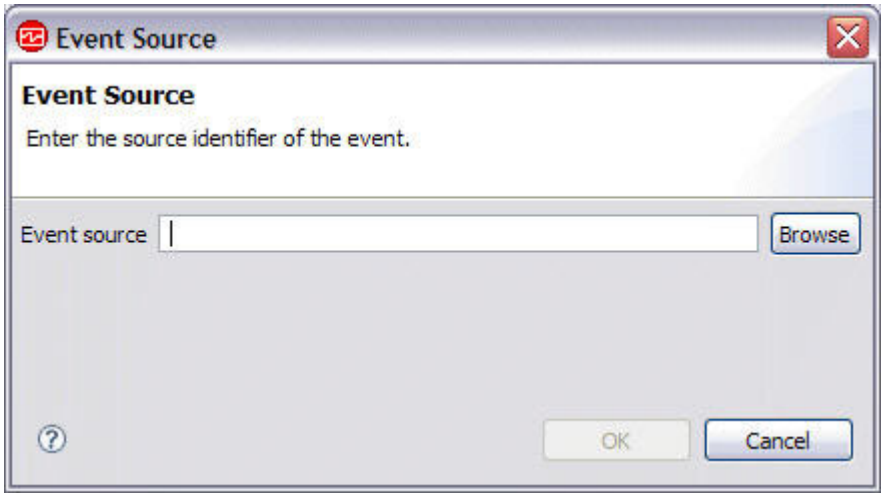


Figure 145. Event Source window

2. Make one of the following choices.
   - Type the event source name and click **OK**.
   - Click Browse **Browse** to find and select an event source from a list (Figure 146 on page 208) and click **OK**.

     The name that you selected is shown in the Event Source window.

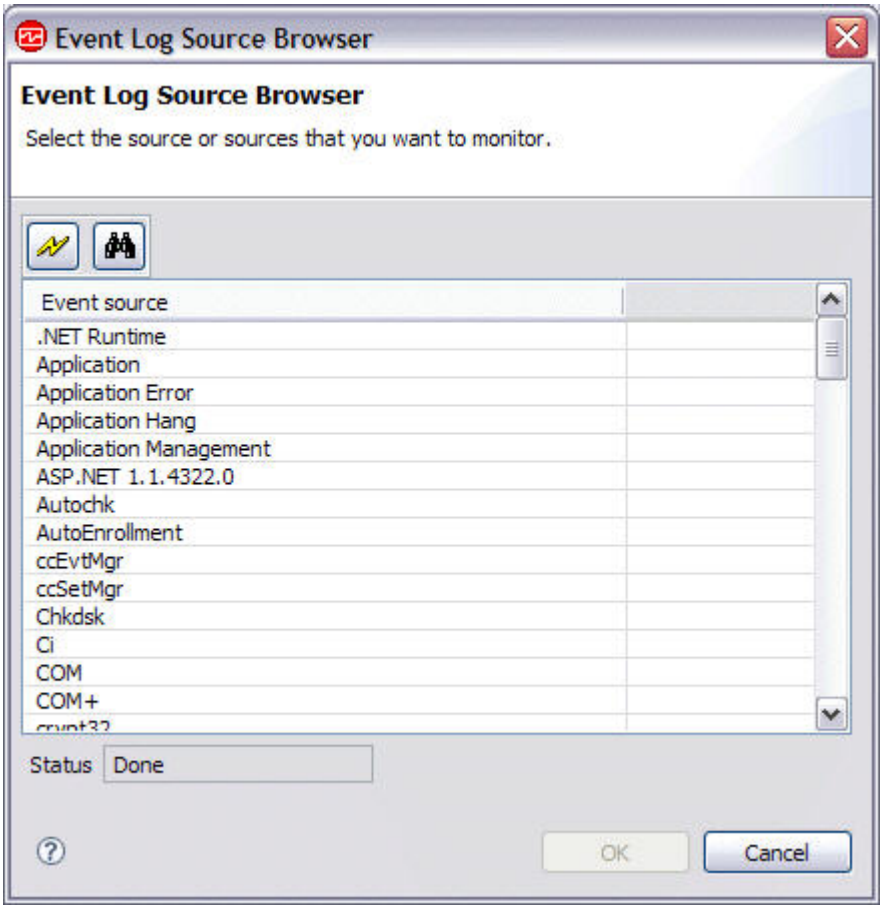


Figure 146. Event Log Source Browser window

Note:

a. To sort the list of event sources, click the column heading.

b. To refresh the information in the window, click the **Refresh** icon.

c. To search for specific event sources, click the **Search** (binoculars) icon.

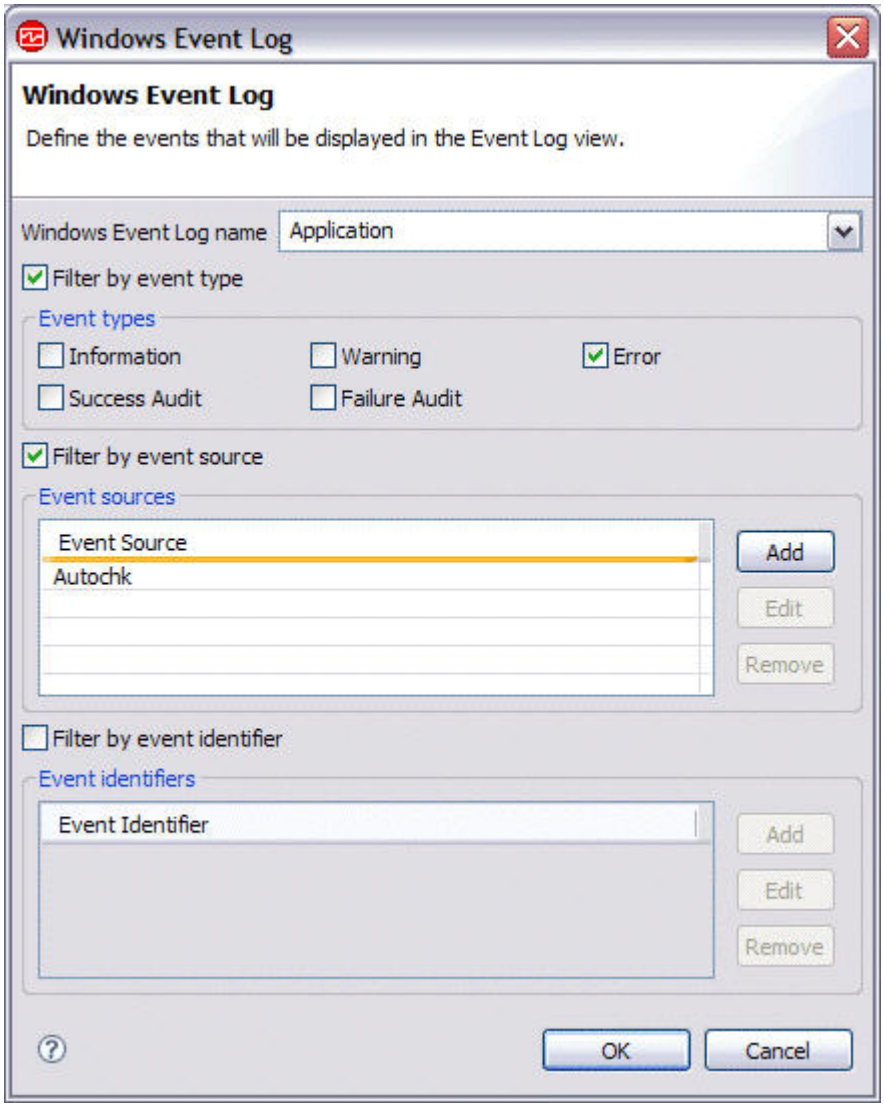


Figure 147. Windows Event Log window

3. Click **OK** to see the new event source filter in the Event sources list in the Windows Event Log window (Figure 147).

Filtering by event identifier

For the Windows Event Log data source, you can filter events by event identifier.

About this task

To filter by event identifier, use the following procedure:

Procedure

1. Select **Filter by event identifier** and click **Add** in the **Event identifiers** area of the Windows Event Log window. The Event Identifier window is displayed.

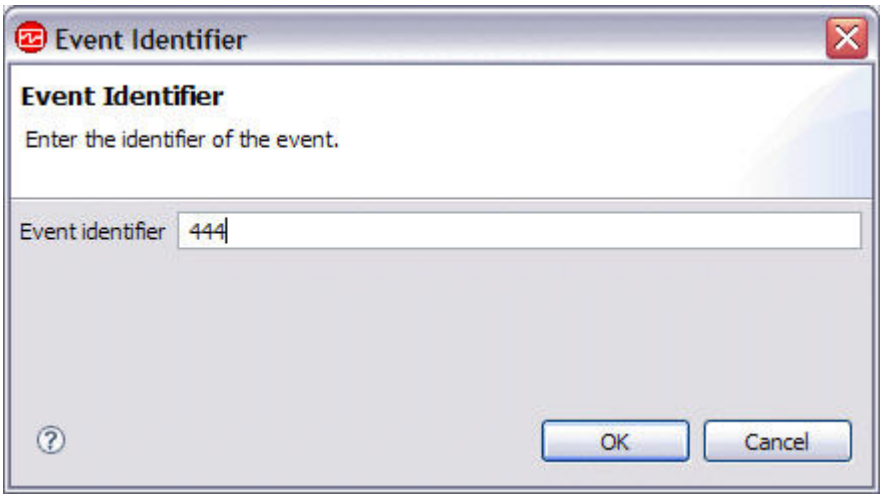


Figure 148. Event Identifier window

2. If you know that you want to monitor specific events from an application, specify the numbers of the event as the application defines it. Type an integer as the event identifier and click **OK**. The new numeric event identifier filter is displayed in the Event identifiers list in the Windows Event Log.

   Note: Each event identifier must be defined individually.

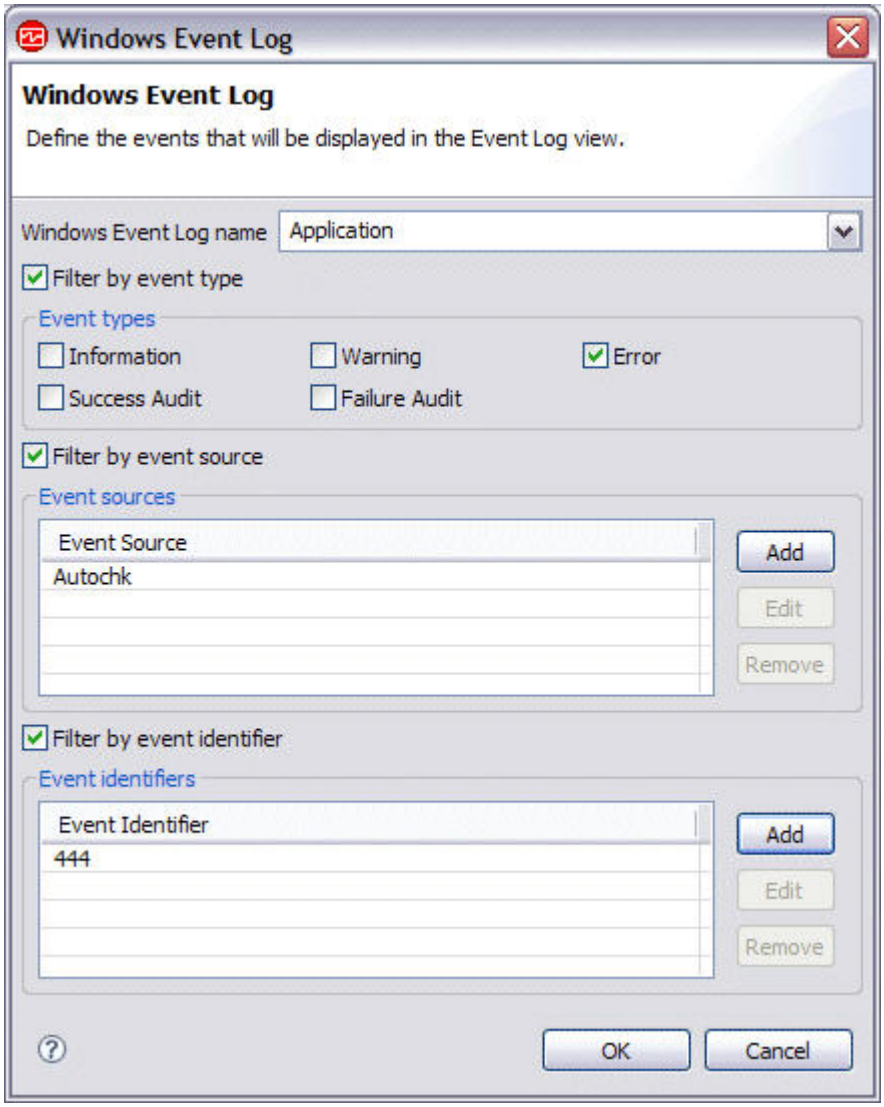


Figure 149. Windows Event Log window

3. If you want to modify a Windows event log, select it and click **Edit**.
4. If you want to delete a Windows event log, select it and click **Remove**.
5. You can add more event logs to the list, or click **Finish**.

Chapter 19. Monitoring a command return code

Define a data source to monitor an application or system by using a *command return code*.

About this task

A command return code is a user created script, executable file, query, or system command. A command return code is an application-specific mechanism for determining whether the application or monitored system is available. The agent runs the specified command and determines the state of the application or monitored system by examining the return code.

The command must present a unique return code for each descriptive state. The command must also define a message to be used by the agent for each of these return codes. The command can use environment and configuration variables within the user created script, executable file, query, or system command. The command must not use environment or configuration variables on the command-line invocation of the command, with only the following exceptions available: *AGENT_BIN_DIR*, *AGENT_ETC_DIR*, *AGENT_LIB_DIR*, *CANDLE_HOME*, and *CANDLEHOME*.

Procedure

1. On the Agent Initial Data Source page or the Data Source Location page, select **Command or script** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

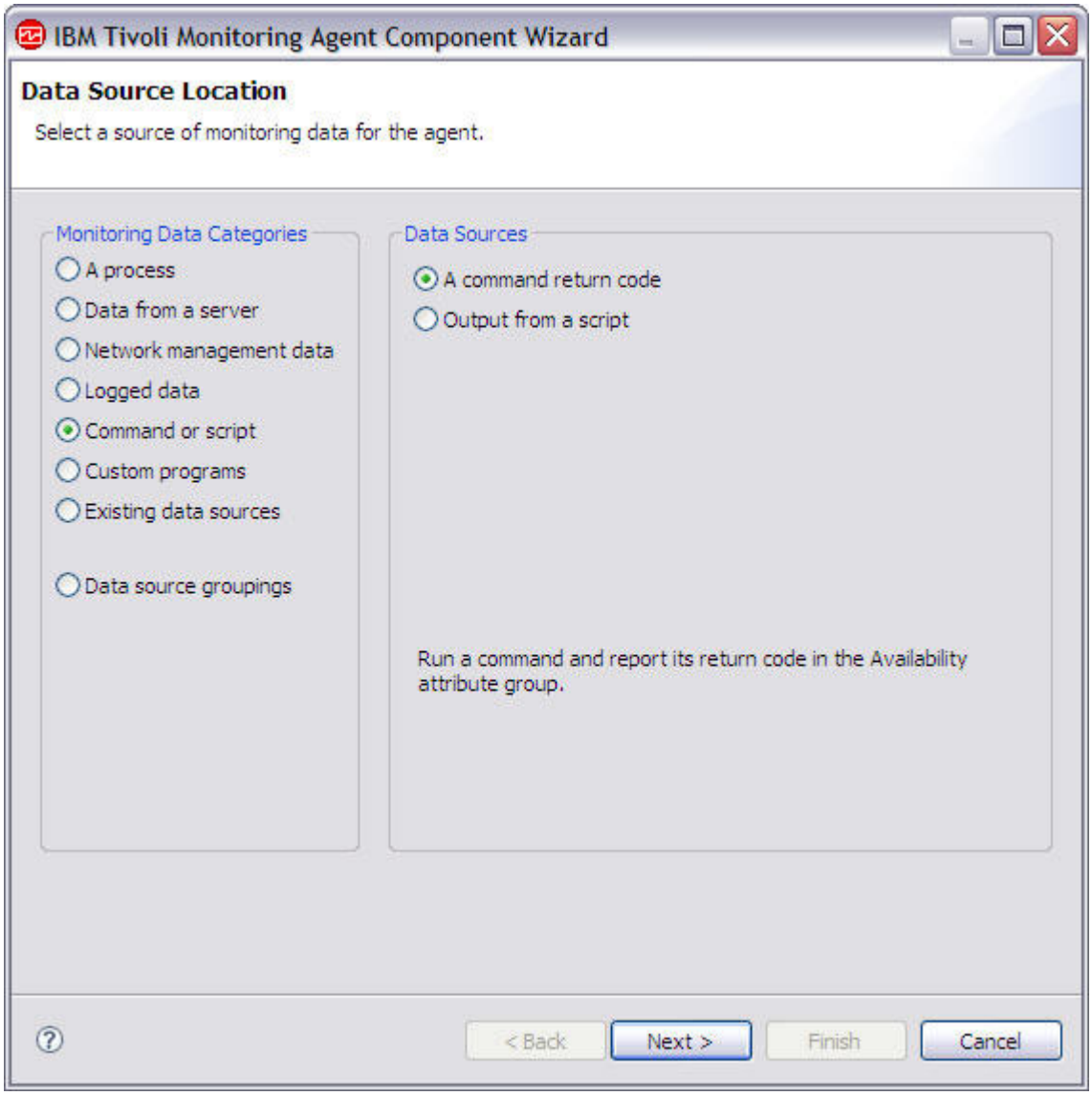


Figure 150. Adding data from a command

2. In the **Data Sources** area, click **A command return code**.
3. Click **Next**.
4. On the Command Return Code page, **Command return code information** area, type the display name.

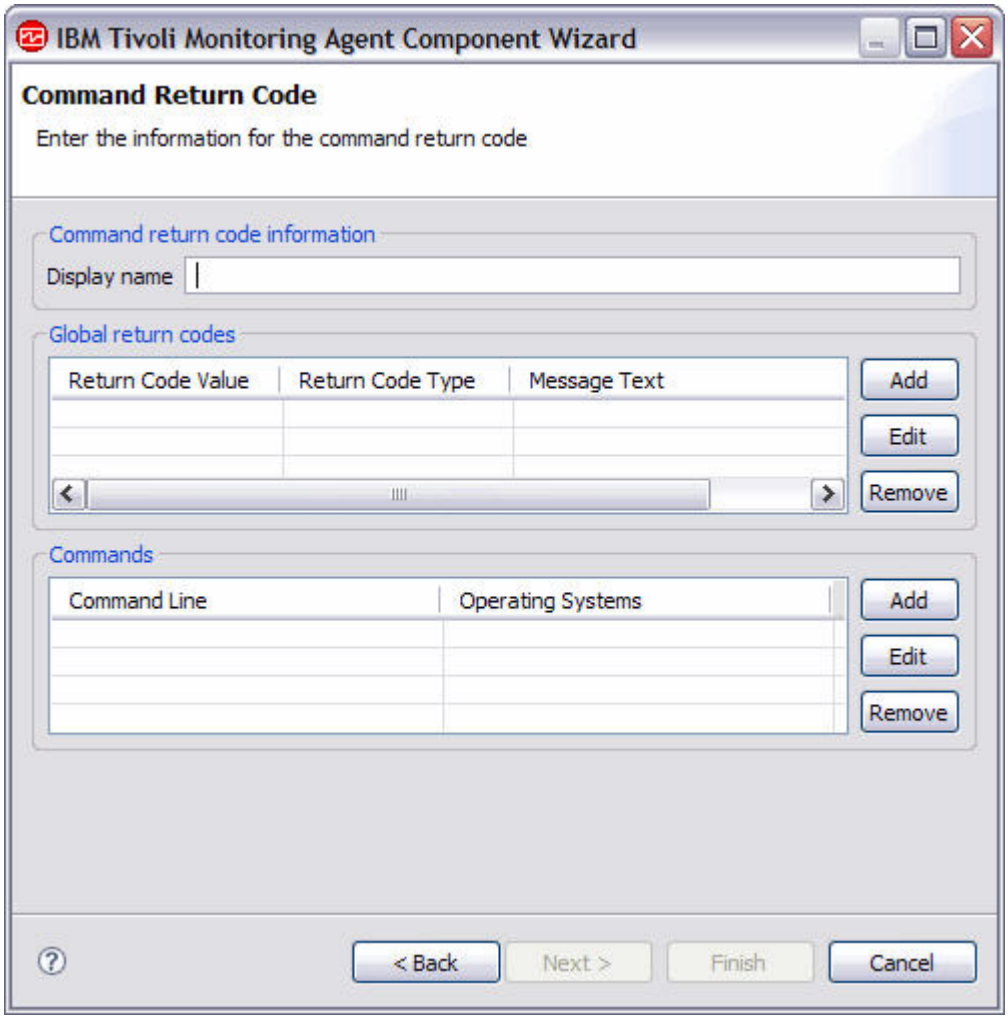


Figure 151. Command Return Code page

5. Use the following substeps to define and describe command lines that you want your command return code to use.

   Note: Define a command for each operating system that is supported by the agent. Commands can be shared, but the total set of operating systems for all of the commands must equal the set of agent supported operating systems.

   a. Click **Add** in the **Commands** area of the Command Return Code window to open the Command Information window.

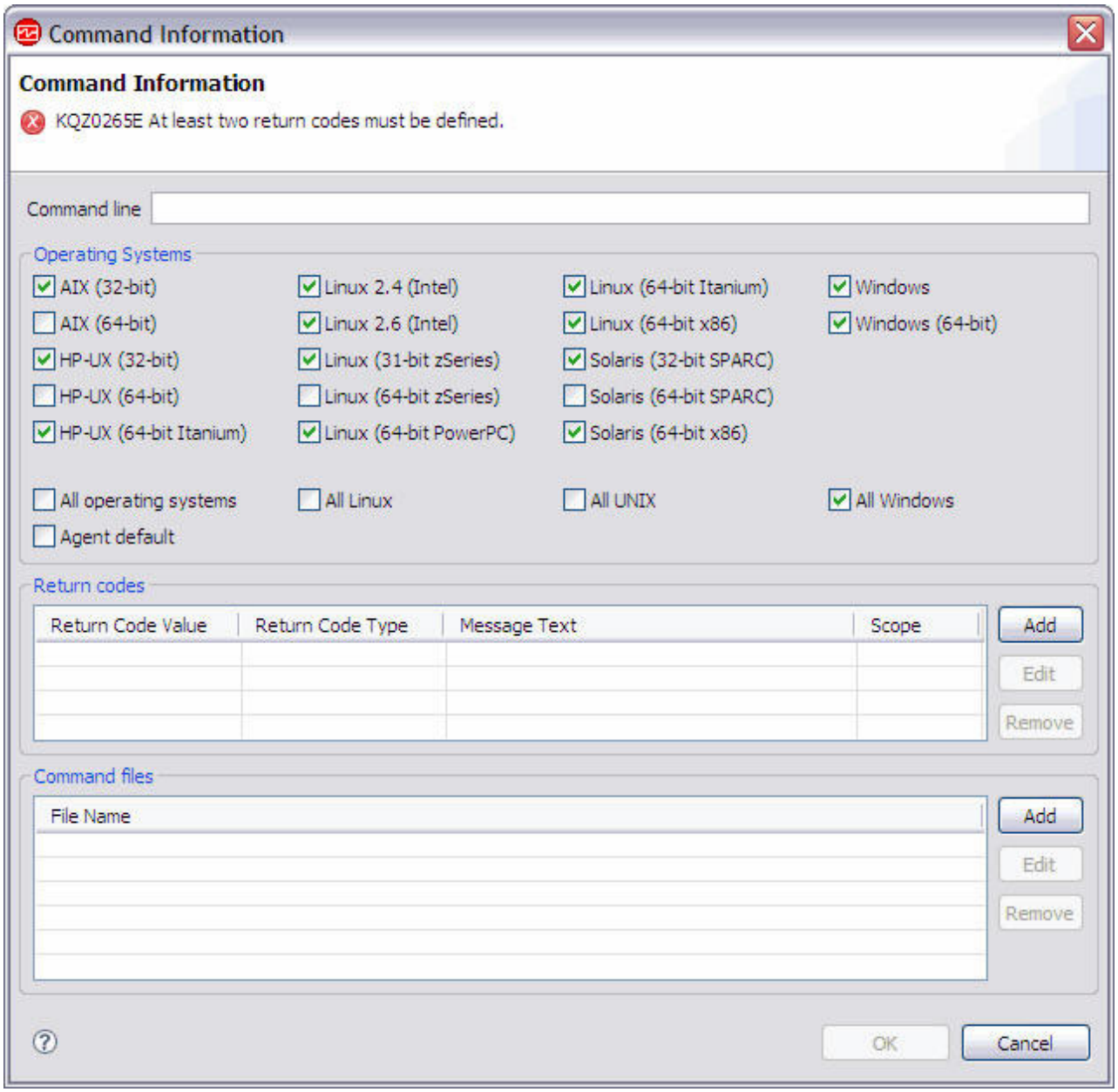


Figure 152. Command Information window

b. Type a command line and select an operating system from the list in the Operating Systems area of the Command Information window.

   Note:

   1) For a Windows command, you must type the full name of the command. For example, `command_to_run.bat` and not just `command_to_run`.
   2) Place quotation marks around the name so that it is not parsed by the command interpreter. For example, type `"this is a test.bat"argument` and not `this is a test.bat argument`.
   3) You can click a command and click **Edit** to modify it, or click **Remove** to delete it.

c. Click **Add** in the **Return Codes** area of the Command Information window.

d. Select a return code type from the list that is shown in the Return Code Definition window

   You can assign the following states to the test return codes:

- ALREADY_RUNNING
- DEPENDENT_NOT_RUNNING
- GENERAL_ERROR
- NOT_RUNNING
- OK
- PREREQ_NOT_RUNNING
- WARNING

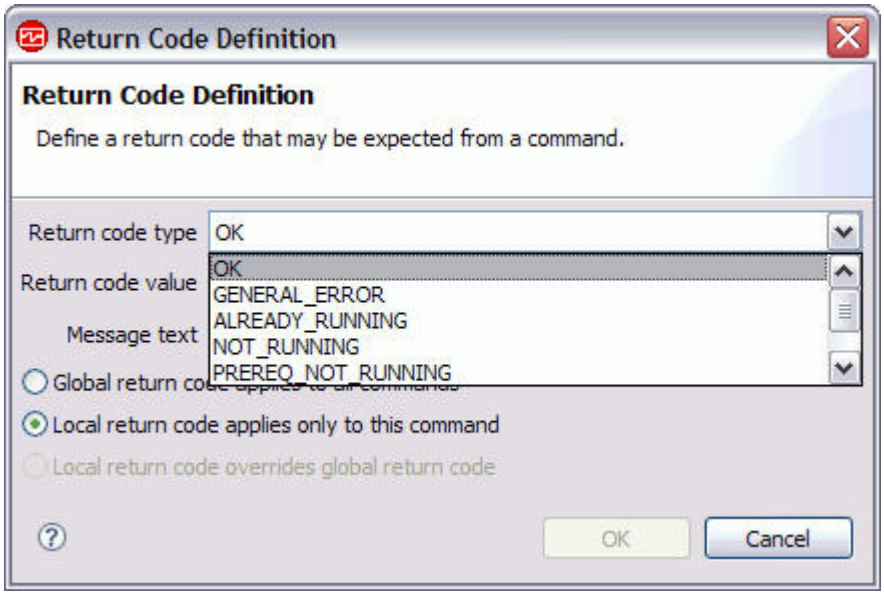


Figure 153. Return Code Definition window: Return code type

e. Type a numeric value for the return code type that you selected. The return code value is an integer that specifies a defined return code for the command return code. For portability between operating systems, use a return code value of `0 - 255`. For a command that runs only on Windows, the return code value can be `-2147483648 - 2147483647`.

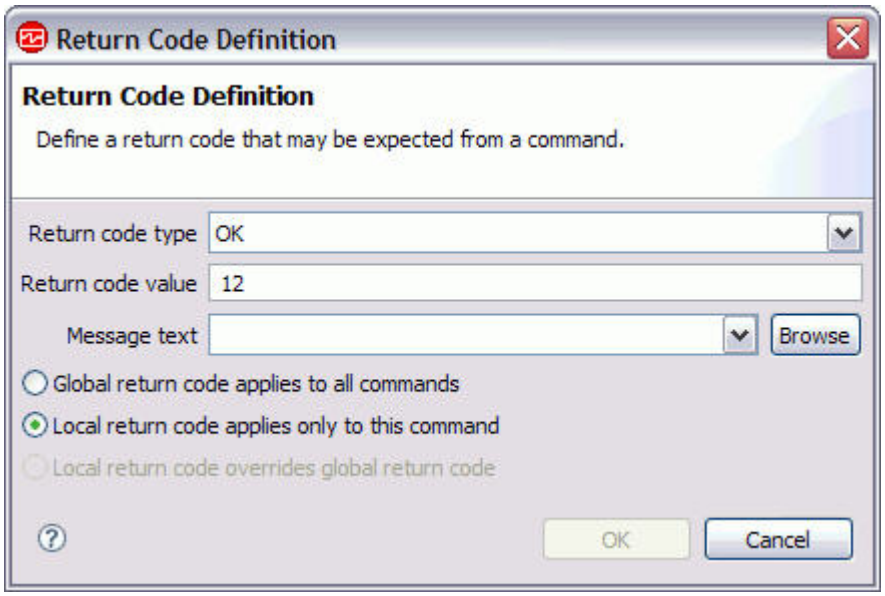


Figure 154. Return Code Definition window: Return code value

f. Define a message for each return code so that the message and code can be shown together. Click **Browse** to set up the message text.

   The message window lists messages that are defined in the agent. The Messages (list) window opens.

   Note:

   1) You can select text that was entered previously by selecting it in the Message text menu instead of clicking **Browse**. Then, continue to Step 5k.
   2) Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you defined.

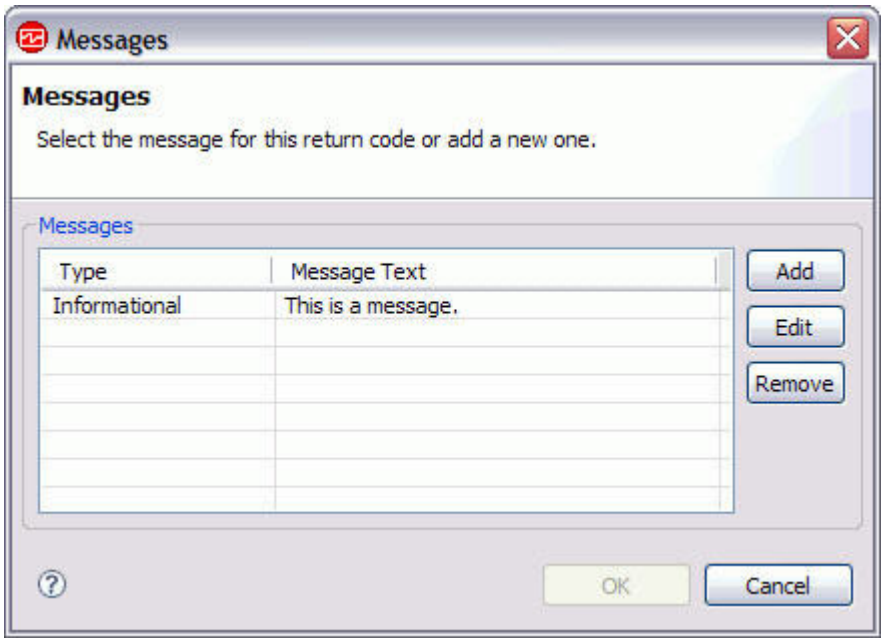


Figure 155. Messages window

g. In the Messages (list) window, click **Add** The Message Definition window opens.

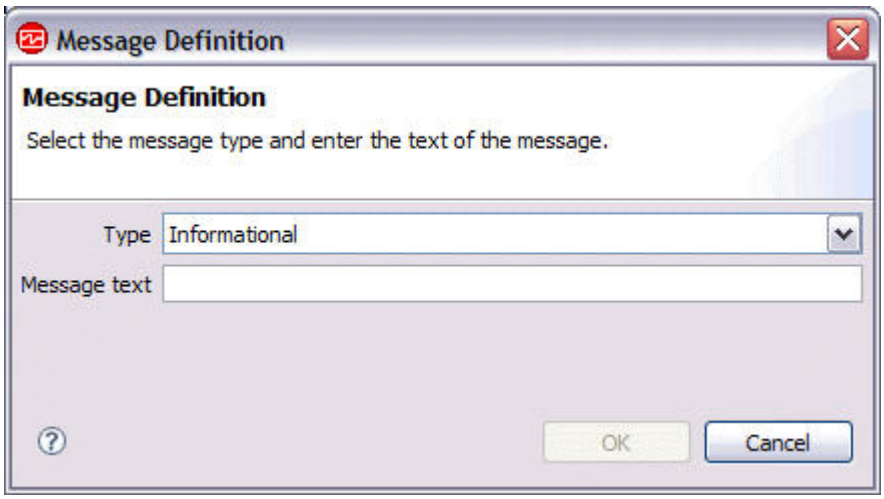


Figure 156. Message Definition window

Note: The message identifier is automatically generated for you.

h. Enter some text that describes the meaning of the new message in the **Message text** field.

i. Click **OK**. The Messages (list) window opens showing the new message (Figure 160 on page 223).

j. To verify the message and make it permanent, select it in the list and click **OK**. The new return code type, value, and text are shown in the Return Code Definition window.

k. If you want this return code to be available to other commands on other operating systems for this command return code, select **Global return code applies to all commands**. If you want this return code to be available only to this command, leave **Local return code applies only to this command** selected.

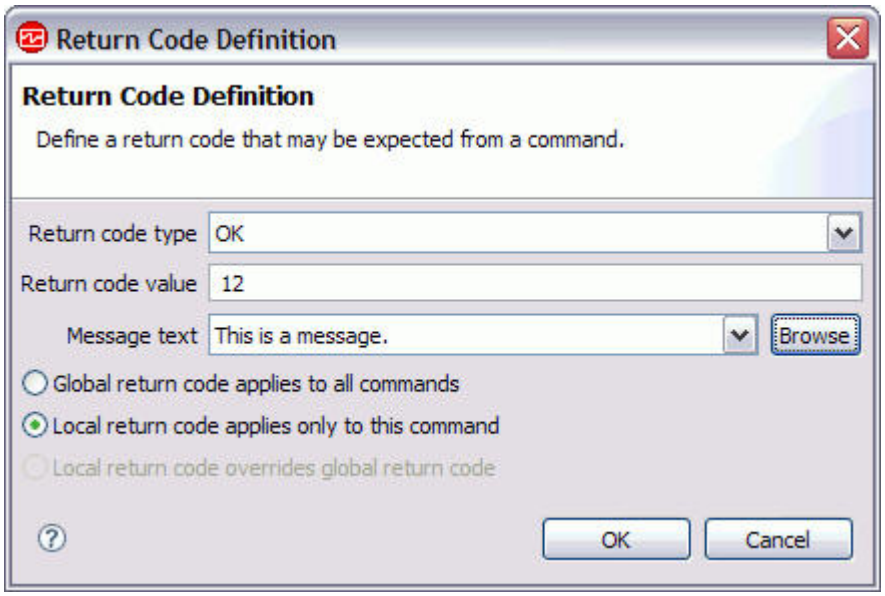


Figure 157. Return Code Definition window completed

l. Click **OK** in the Return Code Definition window.

m. Define at least two return codes before you leave the Command Information window. One return code to indicate no problems with the availability, another to indicate whether a problem occurred. If you want to add another return code, return to step c.

n. Optional: In the Command Information window (Figure 152 on page 216), **Command files** area, click **Add** if you want to select one or more scripts or executable files for the agent to run. The file or files are copied into the project folder of the agent under `scripts/`*`operating system`*, where *operating system* is a variable that depends on what you selected in (Figure 152 on page 216). These files are also packaged and distributed with the agent. To edit the definition of an existing command file, or the original command file since copied into the project, select the file and click **Edit**. See ("Editing a command file definition" on page 224).

o. Click **OK** in the Command Information window.

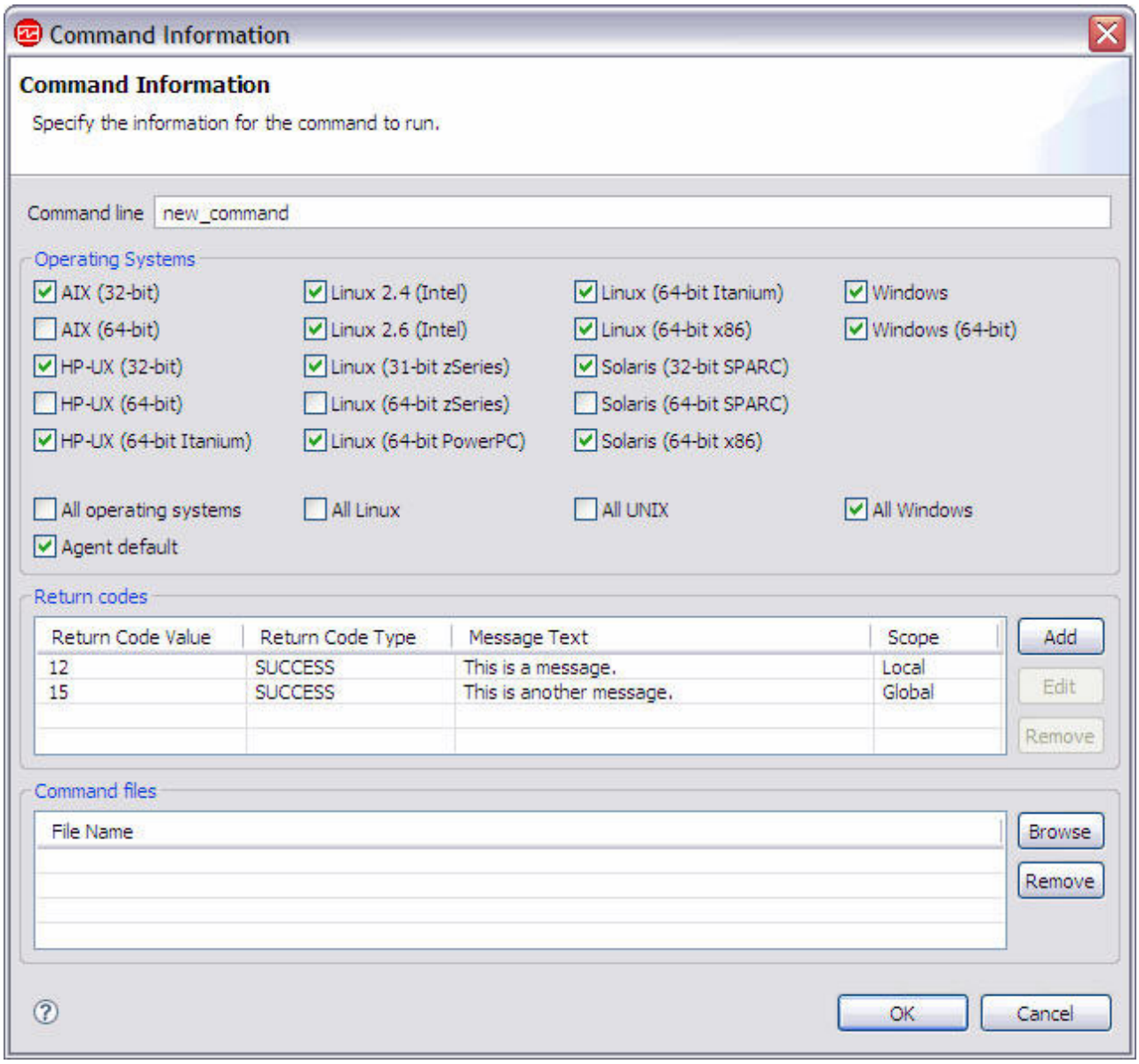


Figure 158. Command Information window completed

Note: The command files table is where you define any external files that you want to include in the agent package. These files are copied into the project directory and packaged with the agent for distribution.

6. If you have other return codes that are not already defined, define and describe global return codes that your command return code can use.
   a. Click **Add** in the Global return codes area of the Command Return Code page (Figure 151 on page 215).

   Note: The return codes that are defined here are global. This means that the return codes are appropriate for all of the commands that are defined for the command return code. (They are not shared between command return codes). In addition, you can define return codes when you enter the command information (Figure 152 on page 216). The return codes that are defined here can be global or local. Local return codes are only appropriate for this specific command. This hierarchy is useful if you have a return code that is the same across all operating systems. (For instance, a return code of 0 means that everything is functioning correctly. You can define it at the global

level, and then all defined commands interpret 0 in this way.) If none of the other operating systems return a 5, you can define the return code of 5 only for the Windows command. If you define a return code at the local command level that is already defined at the global level, the command level is used. You can use this method to override return codes on specific operating systems. For instance, if on all UNIX operating systems, a return code of 2 means one thing, but, on Windows, it means something different. You can define a return code of 2 at the global level as expected by the UNIX operating systems. Then, in the command for Windows, you can redefine return code 2 for the meaning on Windows.

b. Select a return code type from the list that is shown in the Return Code Definition window (Figure 153 on page 217).

   You can assign the following states to the test return codes:

   - `ALREADY_RUNNING`
   - `DEPENDENT_NOT_RUNNING`
   - `GENERAL_ERROR`
   - `NOT_RUNNING`
   - `OK`
   - `PREREQ_NOT_RUNNING`
   - `WARNING`

c. Type a numeric value for the return code type that you selected (Figure 159). The return code value is an integer that specifies a defined return code for the command return code.

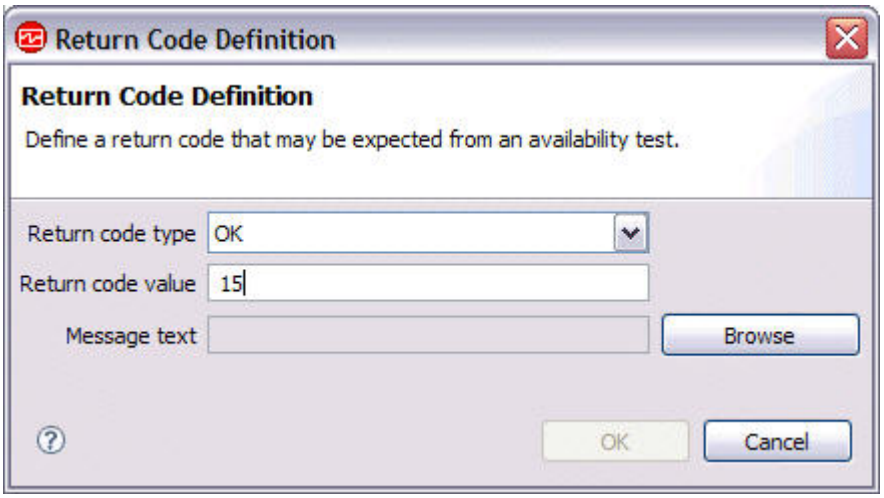


Figure 159. Return Code Definition window: Message text

d. Click **Browse** to set up the message text and its associated meaning. You must define a message for each return code so that the message and code are shown together.

   The Messages window lists messages that are defined in the agent. The Messages (list) window opens(Figure 160 on page 223).

   Note:

   1) Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages you defined.
   2) You can select text that was entered previously by selecting it in the **Message text** list instead of clicking **Browse**. Then, continue to Step 6h.

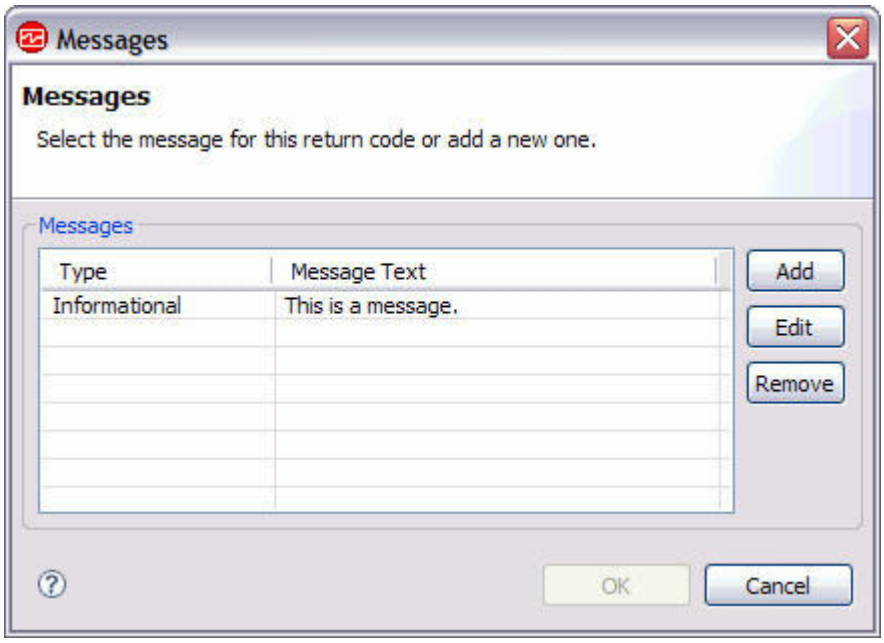


Figure 160. Messages window

e. In the Messages (list) window, click **Add** to see a Message Definition window, where you can type text that describes the meaning of the new message.

f. Click **OK**.

g. The Messages (list) window opens with the new message (Figure 155 on page 219). To verify the message and make it permanent, select it in the list and click **OK**.

h. When the new text, type, and value are shown in the Return Code Definition window, (Figure 161), click **OK**.

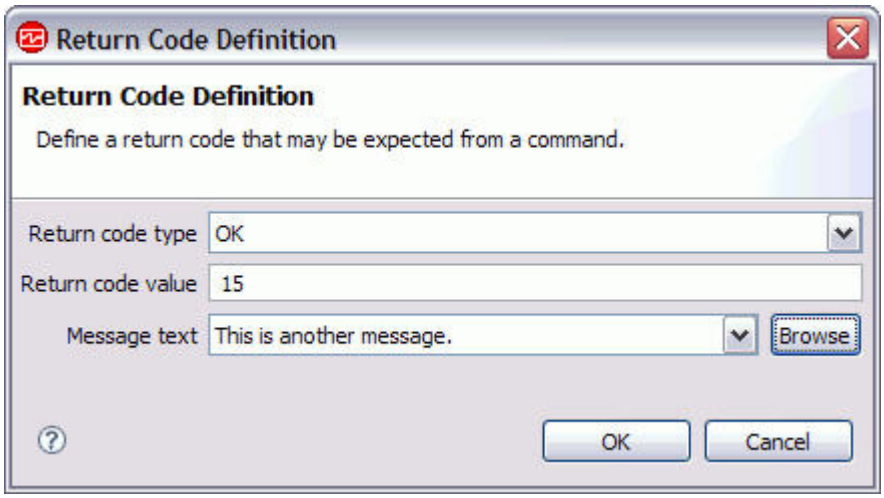


Figure 161. Return Code Definition window

i. On the Command Return code page, when you finish defining the return codes and commands for all supported operating systems, do one of the following steps:

- If you are using the New Agent wizard, click **Next** or click **Finish** to save the data source and open the Agent Editor.
- If you are using the New Agent Component wizard, click **Finish** to return to the Agent Editor.

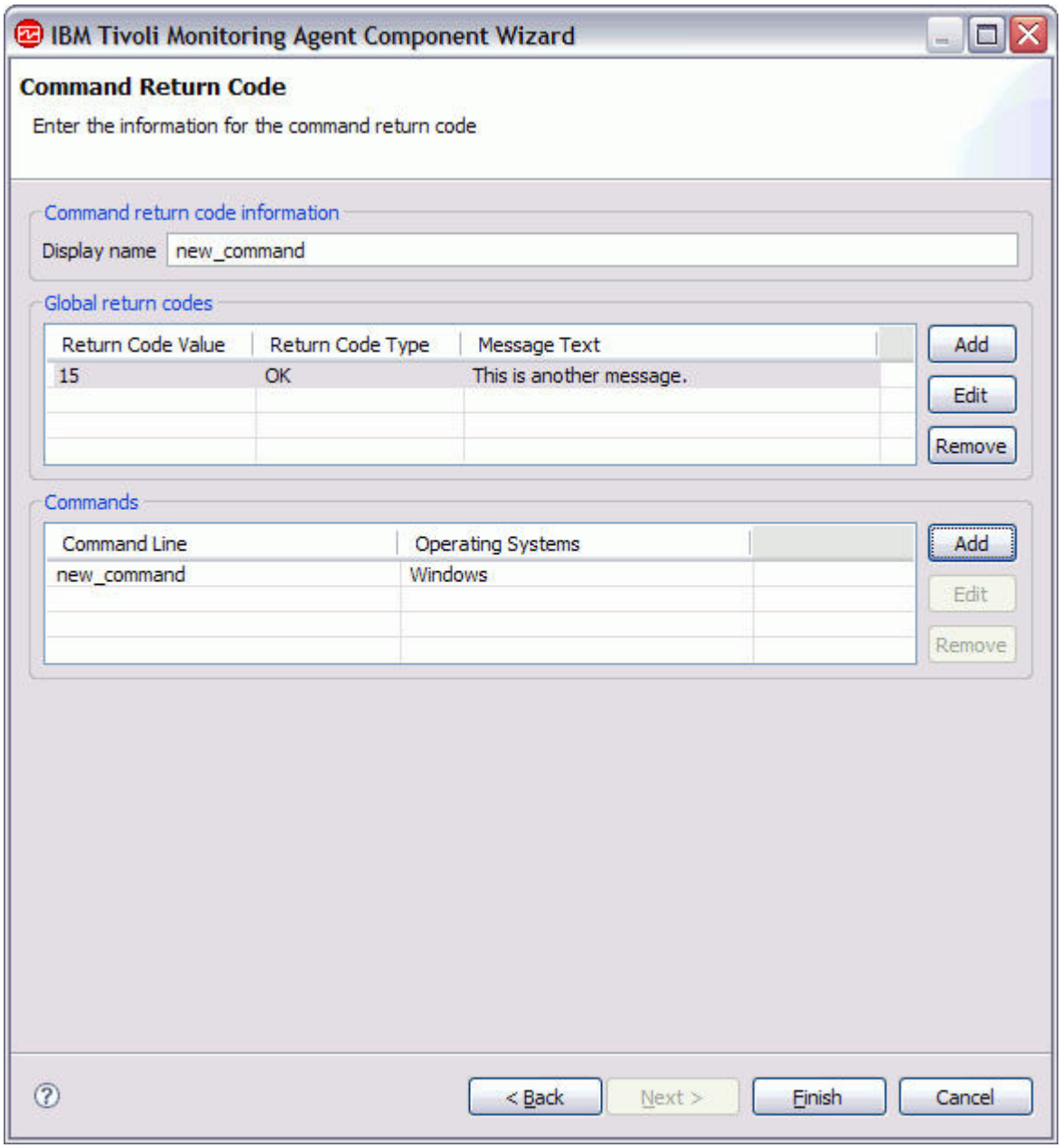


Figure 162. Command Return Code page completed

Editing a command file definition

You can change the command file that is imported into the project, or import changes to the existing command file into the project.

Procedure

1. Select the file in the Command files area of the Command Information window.
2. Click **Edit** to open the Import Command File window Figure 163 on page 225 From the Import Command File window, you can get the status of the command file. You can also change the location of the original source file, and recopy the source file into the agent.
3. Choose one of the following steps:

- Click **OK** to schedule a copy of the file to occur the next time that the agent is saved.
- Click **Copy Immediately** to copy the file without first saving the agent.

Note: The **Copy Immediately** option is not available when you access the Import Command File window from the New Agent wizard.

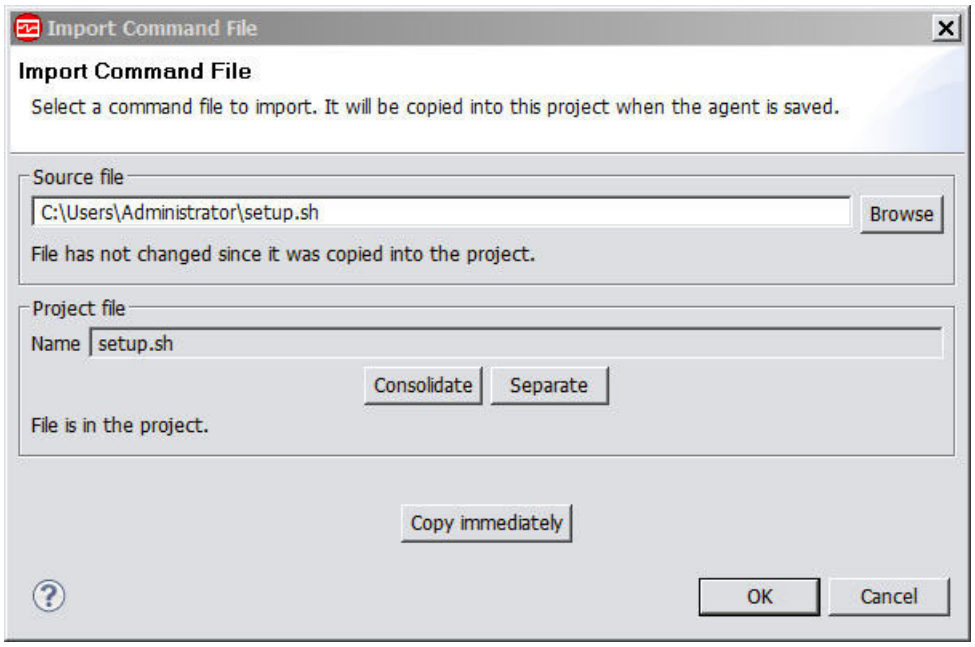


Figure 163. Import Command File window

File Separation & Consolidation

You can use the Separate and Consolidate functions to move files in and out of operating system-specific folders in the agent.

When a file is first added to the agent, a single copy is added in the `scripts/all_windows` folder, the `scripts/all_unix` folder, or the `scripts/common` folder. The `scripts/common` folder is used if the file is used on both Windows and UNIX.

To place different copies of the file on different operating systems (for example, a binary executable file), click Edit and click **Separate**. The file is removed from the common folder and copied into operating system-specific folders. Then, you can replace individual copies of the file with ones appropriate for specific operating systems.

Note: Java resource files must remain in the `scripts/common` folder. You cannot click **Separate** to make separate copies of Java resource files for individual operating systems.

If you separated the files into operating-system-folders, you can use **Consolidate** to move them back into a common folder. If you created the agent in an Agent Builder version that did not support common folders, use **Consolidate** to move them back into a common folder. If any of the copies of the file differ from one another, you are prompted to select the file to use as the common file. All other copies are discarded.

Chapter 20. Monitor output from a script

When application data is not available through a standard management interface, a script or external program can be used to collect data for an attribute group.

Data can be collected from either a local or remote system. The output of the script or program must contain only values for each attribute within the attribute group. To return multiple rows of data, the data for each row must be separated by a line break. The attributes in each row of data are separated by the separators you define. For more information about separators, see "Script parsing and separators"

The command can use environment and configuration variables within the user-created script, executable file, query, or system command. The command cannot use environment or configuration variables on the command-line invocation of the command, with only the following exceptions available: `AGENT_BIN_DIR`, `AGENT_ETC_DIR`, `AGENT_LIB_DIR`, `CANDLE_HOME`, and `CANDLEHOME`.

The agent monitors script output that is written by using the same locale and code page that the agent runs in.

Collecting script data from a remote system

To collect script or program data from a remote system, Agent Builder uses a Secure Shell (SSH)

To collect data from a remote system, Agent Builder creates a Secure Shell (SSH) session and starts the script or external program on the remote system. The agent establishes and logs on to an SSH session. The agent then uploads the scripts to the remote system, starts the script or external program, and retrieves the output. The agent can be configured to keep the session open or reestablish the session for each invocation. If the session is kept open, the script can be reused or uploaded for each invocation. By default, a single SSH session is used and the scripts are reused for each invocation.

Agent Builder supports use of only SSH Protocol Version 2 with Rivest, Shamir, and Adleman (RSA) or Digital Signature Algorithm (DSA) keys. The agent is either authenticated by user name and password, or by public key authentication. The generation and distribution of the public keys is an administrative task that must be done outside of the agent and Agent Builder.

To run a Take Action command that is written against a Secure Shell (SSH) enabled script data provider on the remote system, see "SSHEXEC action" on page 651.

Restriction: If your agent was built with an Agent Builder version before 6.3 and it has a script data provider that uses SSH, the provider fails when run with IBM Tivoli Monitoring version 6.3 or later. For more information about the failure, see the problem titled Script data providers that use SSH are failing in "Troubleshooting: Agents" on page 508.

The restriction is because IBM Tivoli Monitoring version 6.3 uses a newer version of the Global Secure ToolKit (GSKit) API. You must rebuild the agent with Agent Builder 6.3 or later to run it with IBM Tivoli Monitoring version 6.3 or later. If you build the agent with Agent Builder 6.3, it can also run with earlier versions of IBM Tivoli Monitoring.

Script parsing and separators

You can change and assign specific script separators to one or more attributes.

© Copyright IBM Corp. 2007, 2013

When you create a script attribute group, a single character text separator is by default assigned. The default separator is ";". The separator is used by the agent to parse and delimit the data for each attribute in the data row. You can change the default separator to use a different character. You can also assign specific separators to one or more individual attributes.

You can assign specific separators for individual attributes that:

- Take a fixed number of bytes from the output.
- Separate one attribute from the next with a custom separator, which can be more than one character.
- Delimit an attribute value with a string at the beginning and end of the value.
- Return the rest of the text as the attribute value (whether it contains embedded separators or not).

You can use one or more of these separators to extract attribute values from the data rows.

Example 1 - Simple script output

Some scripts can output data rows with clear and regular separators, for example:

```
Row One;1;2
Row Two;3;4
Row Three;5;6
```

Here the ";" character is a clear and regular separator between the three pieces of data on each row. In this case, the default separator is fine, so there is no need to change or define other separators. It is not difficult to imagine a similar script output where the separator is a different character, as in the following example.

```
Row One-1-2
Row Two-3-4
Row Three-5-6
```

In this example the separator is changed from a ";" character to a "-" character. In this case when you define the attributes, change the default separator to use the "-" character.

Example 2 - Complex script output

Some scripts can output data rows that have irregular or changing separators, for example:

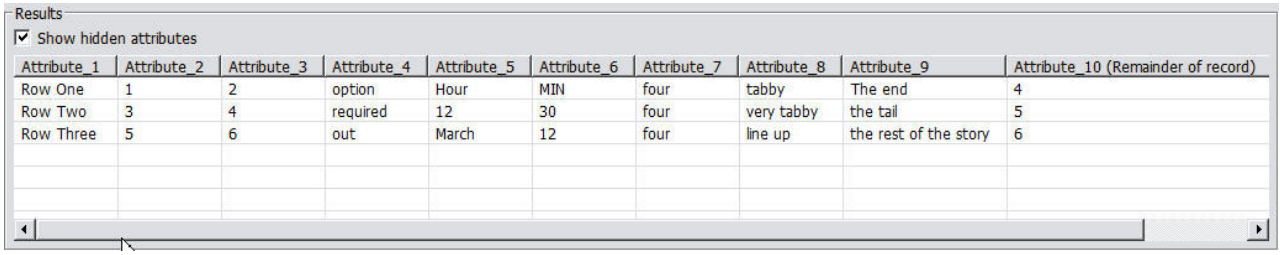
```
Row One;1;2;[option]Hour:MIN;fourtabby     The end;4
Row Two;3;4;[required]12:30;fourvery tabby the tail;5
Row Three;5;6;[out]March:12;fourline up    the rest of the story;6
```

In this example an assignment of separators to attribute definitions that you can use is:

1. Initially the default separator ";" is fine for the first three attributes in each data row. In this case, you assign the separator type **Separator Text** set to ";" when you define each attribute, this setting is the default one.
2. For the fourth attribute, assume the string between the"[" and "]" is a value that you want to extract. In this case when you define the fourth attribute, you assign a separator type **Begin and End Text** with begin and end text values of "[" and "]".
3. For the fifth attribute, assume that you want to extract the values between the "]" and ":" characters. In this case when you define the fifth attribute, you assign separator type **Separator Text** set to ":".
4. For the sixth attribute, the default separator ";" is fine again, accept the default.
5. For the seventh attribute, you would like to extract the string in the next four characters "four". There is not a clear separator at the end of this string. You can assign a number of characters to define the separation from the next attribute. You assign a separator type **Number of characters**, and specify four characters as the length.

6. For the eighth attribute you would like to extract the strings `tabby`, `very tabby` and `line up`. In this case, you can assume that all of these strings are followed by a tab character. In this case, you assign a separator of type **Tab separator**.
7. For the ninth attribute, you revert again to the default separator type to extract the remaining text to this attribute.
8. For the 10th attribute, you specify **Remainder of record** to assign the remainder of the data row to this attribute

Defining these separators on a script that outputs the data rows that are shown earlier in this example is shown in the following output:


Results

Show hidden attributes

Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	Attribute_6	Attribute_7	Attribute_8	Attribute_9	Attribute_10 (Remainder of record)
Row One	1	2	option	Hour	MIN	four	tabby	The end	4
Row Two	3	4	required	12	30	four	very tabby	the tail	5
Row Three	5	6	out	March	12	four	line up	the rest of the story	6

Figure 164. Example attribute value output when Agent parses complex script output.

The procedure to define the attribute separators is described under step 10 on page 235 of (“Steps for monitoring output from a script”).

Steps for monitoring output from a script

Configure your agent to receive data from a script data source.

Before you begin

See Chapter 20, “Monitor output from a script,” on page 227

About this task

Use the following procedure to monitor output from a script:

Procedure

1. On the Agent Initial Data Source page (Figure 165 on page 230) or the Data Source Location page, select the option **Command or script** in the **Monitoring Data Categories** area.

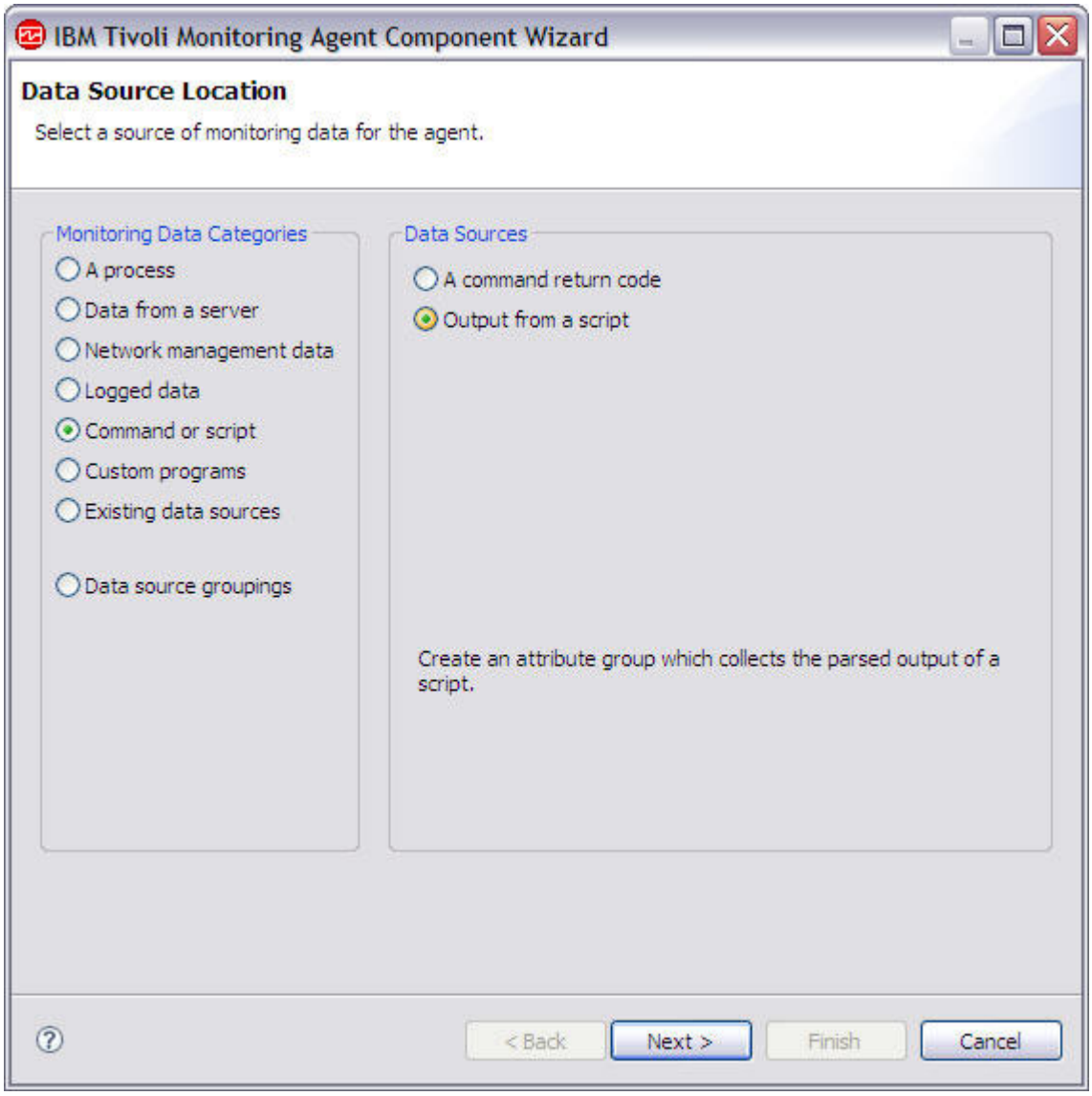


Figure 165. Adding data from a script

2. In the **Data Sources** area, click **Output from a script**.
3. Click **Next**.
4. On the Command List page (Figure 166 on page 231), click **Add** to display a Command Information window.

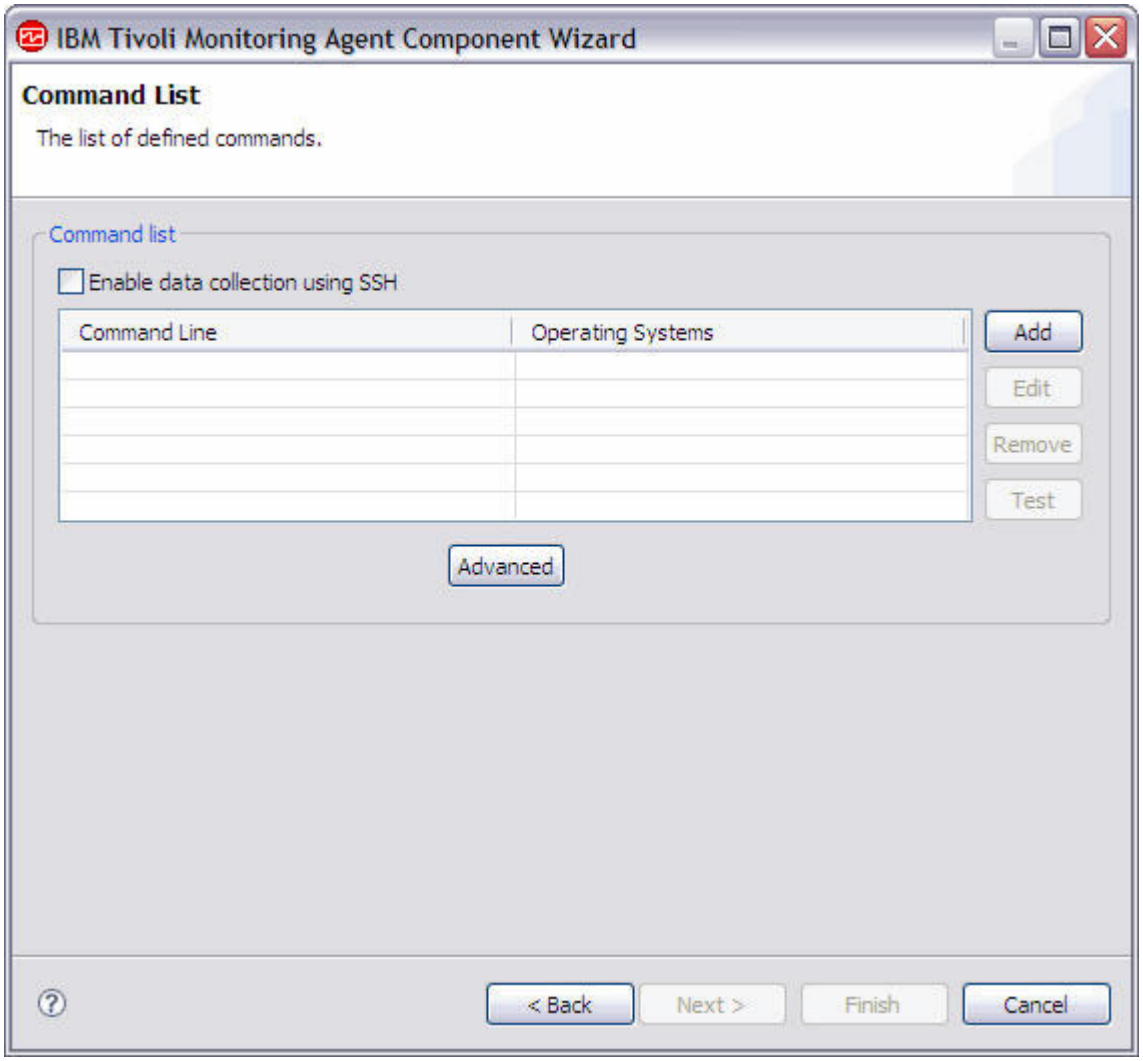


Figure 166. Command List page

Note: Selecting the **Enable data collection using SSH** check box enables SSH for this attribute group. If this check box is not selected, the attribute group runs locally.

Note: If a command exists that can be run on the operating system on which the Agent Builder is running, the **Test** option is enabled. You can use **Test** to test a command that you defined.

5. In the **Command Information** area in the Command Information window (Figure 167 on page 233), type a command name with the necessary arguments in the **Command** field, and a separator in the **Separator** field.

   Note:

   a. Scripts in Windows are frequently started without specifying the `.bat` or `.cmd` extension on the command line. For remote execution, a shell environment must be installed and you must specify the `.bat` or `.cmd` in the script data source command for the script to run. `Cygwin` is an example of a shell environment that is available for Windows. Linux, Red Hat, and AIX. To verify that a shell environment exists, SSH or log on to the remote host and enter the command:

      `PATH=$PATH:. <command>`

      If the command runs, then a shell environment exists.

b. Use quotation marks around the name so that it is not parsed by the command interpreter. For example, `this is a test.bat argument` becomes:

   ```
   "this is a test.bat" argument
   ```

c. Environment variables and configuration variables can be used in the user-provided script, but cannot be part of the command line that starts the script. The following variables are exceptions to this rule:

   AGENT_BIN_DIR
   The directory where the agent places binary files or scripts

   AGENT_ETC_DIR
   The directory where the agent places configuration files

   AGENT_LIB_DIR
   The directory where the agent places shared libraries or dynamic-link libraries

   CANDLEHOME
   The Linux Tivoli Monitoring installation directory

   CANDLE_HOME
   The Windows Tivoli Monitoring installation directory

d. If the SSH data collection option is being used, the command line is run relative to the user's home directory on the remote system. If you are uploading scripts or executables to the remote system, they are copied to the location specified in the agent's environment variable *CDP_SSH_TEMP_DIRECTORY*. The location defaults to the user's home directory on the remote system. On some systems, you might need to define the command line with a relative path, such as `./Script.sh`.

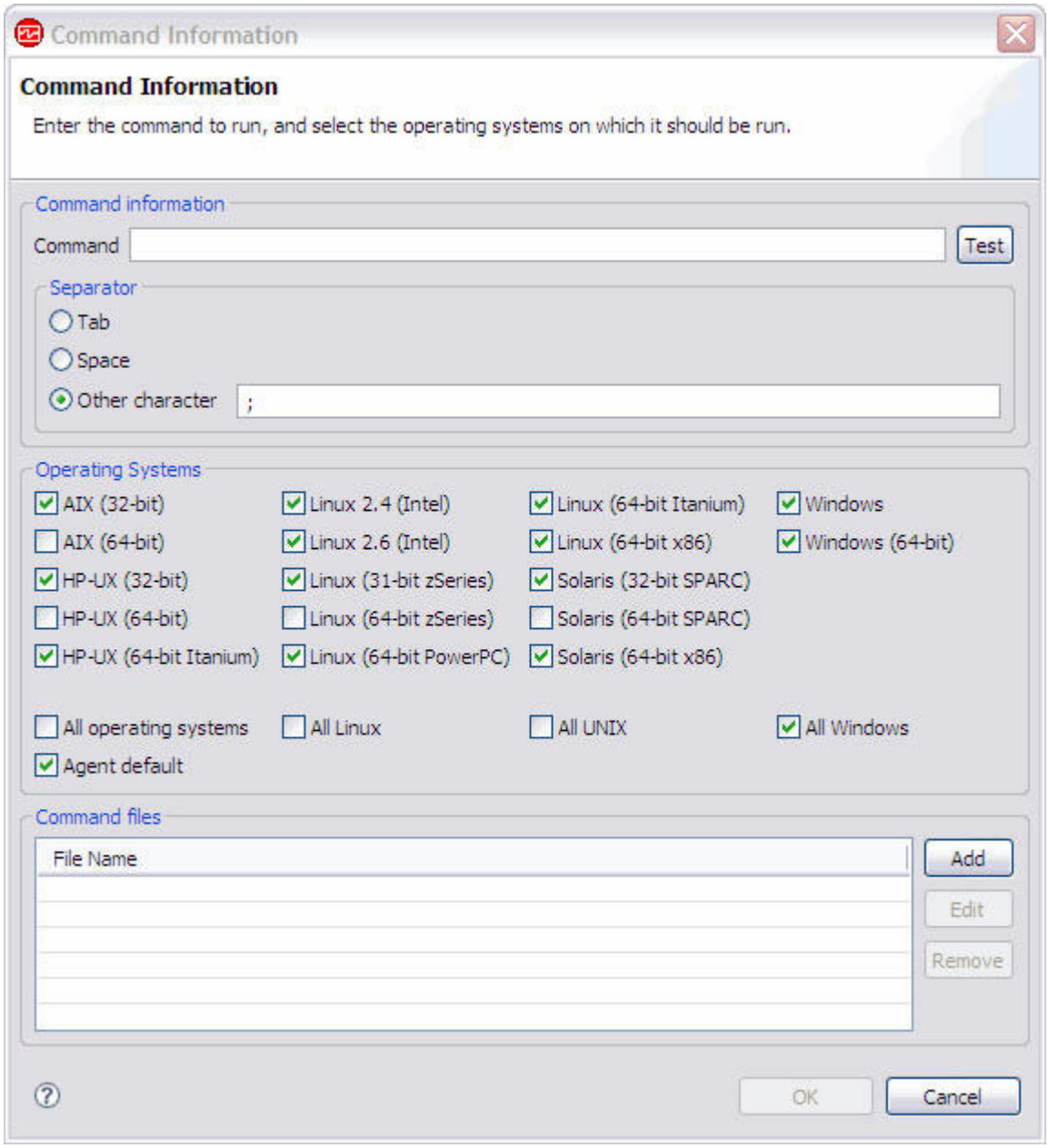


Figure 167. Command Information window for a script

6. In the **Operating Systems** area, select one or more operating systems. When you collect data from a remote system by using SSH, Operating Systems is a property of the system on which the agent is installed. It is not the Operating System of the remote system. It is advised that you select the **All operating systems** check box when you use the SSH data collection features.
7. Optional: If one or more user-defined files are necessary to run the command, click **Add** in the Command files area to specify the files from your system. The files are copied into the project folder of the agent under `scripts/`*`operating system`*, where *operating system* is a variable that depends on what you selected in (Figure 152 on page 216. These files are also packaged and distributed with the agent. If you want to edit the definition of a command file you already added, or changed the contents of, select the file and click **Edit**. See (“Editing a command file definition” on page 224).
8. Click **OK**. The Command List page is displayed.
9. To test the command, use the following steps:

a. Click **Test** to open the command information and display the Test Command window (Figure 168). To test the script on a remote system, select a system from the **Connection name** list or click **Add** to add the host name of a system.

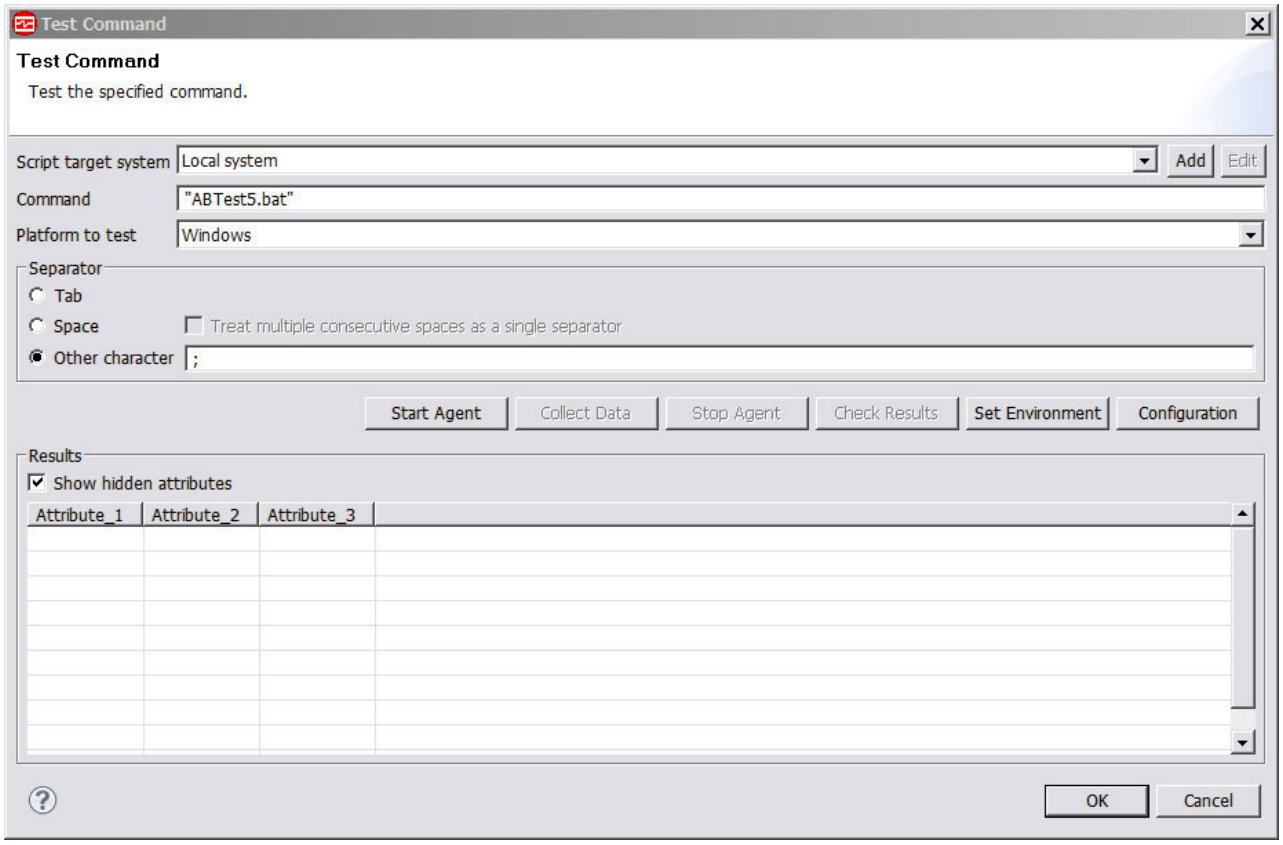


Figure 168. Test Command window

b. Use the Test Command window to change the command, default separator, and attribute separators, and to view how these changes affect the data that is returned.

1) Type the command and separator in the fields if they are not already entered.

   Note: You can specify other separators by using the Attribute Information window Figure 169 on page 236 at attribute creation time or by using the Agent Editor to modify an existing attribute. For more information about the Agent Editor, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35 and for more information about manipulating data source and attributes, see Chapter 7, "Editing data source and attribute properties," on page 55

2) Before you start testing, you can set environment variables and configuration properties. For more information, see ("Attribute group testing" on page 377).

3) Click **OK** to return to the Test Settings window.

4) Click **Start Agent**. A window indicates that the Agent is starting.

5) To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The Agent Builder runs your command. If you specified a remote system, provide a user ID and password. Even if the return code is not 0, the Agent Builder parses the results of the command in the same way the agent does.

6) The Test Settings window collects and displays any data in the agent's cache since it was last started. The initial names of the attributes are **Attribute_1, Attribute_2**, and so on; however, you can modify the properties of the attributes by clicking the appropriate column heading.

7) Click **Check Results** to view the return code from the command, the unparsed data, and any error messages that were returned.
8) The agent can be stopped by clicking **Stop Agent**.
9) Click **OK** to return to the Command Information window.

   If you change the command or the separator, the appropriate command is updated to reflect those changes.

   If this window was opened when you created the script data source, the attributes were added to the new script data source.

   If this window was opened from an existing script data source, then any changes to the attributes are made to the script data source. Any additional attributes are added, but any extra attributes are not removed. These options affect only the attributes that are parsed from the script output. Any derived attributes are not affected. If any of these attributes become invalid based on the attributes they reference, you can update or remove derived attributes manually. The derived attribute formula is displayed and not the actual result value.

Note: If the attribute group exists, to start a test, complete the following procedure

a. Select the attribute group on the Agent Editor Data Sources Definition page.
b. Select the script to be tested from the Command List
c. Click **Test** and follow the procedure at step 9 on page 233

10. If you skipped testing the command in step (9 on page 233), use the following steps:
    a. On the Command List page with the completed command information, click **Next**. See (Figure 166 on page 231).
    b. On the Attribute Information page (Figure 169 on page 236), complete the attribute name and type information by using (Table 7 on page 58). Select **Add additional attributes** to add further attributes
    c. On the Attribute Information page (Figure 169 on page 236), use the Script Attribute Information tab to choose a specific data separator for this attribute. The standard separator **;** is selected by default. You can choose a number of other separators such as, a string, a number of characters, a tab, or a space. You can also choose to use a different string separator for the beginning and end of the data. Finally, you can also choose **Remainder of record** to assign the remainder of the record to the attribute. For more information about script parsing and separators, see “Script parsing and separators” on page 227.

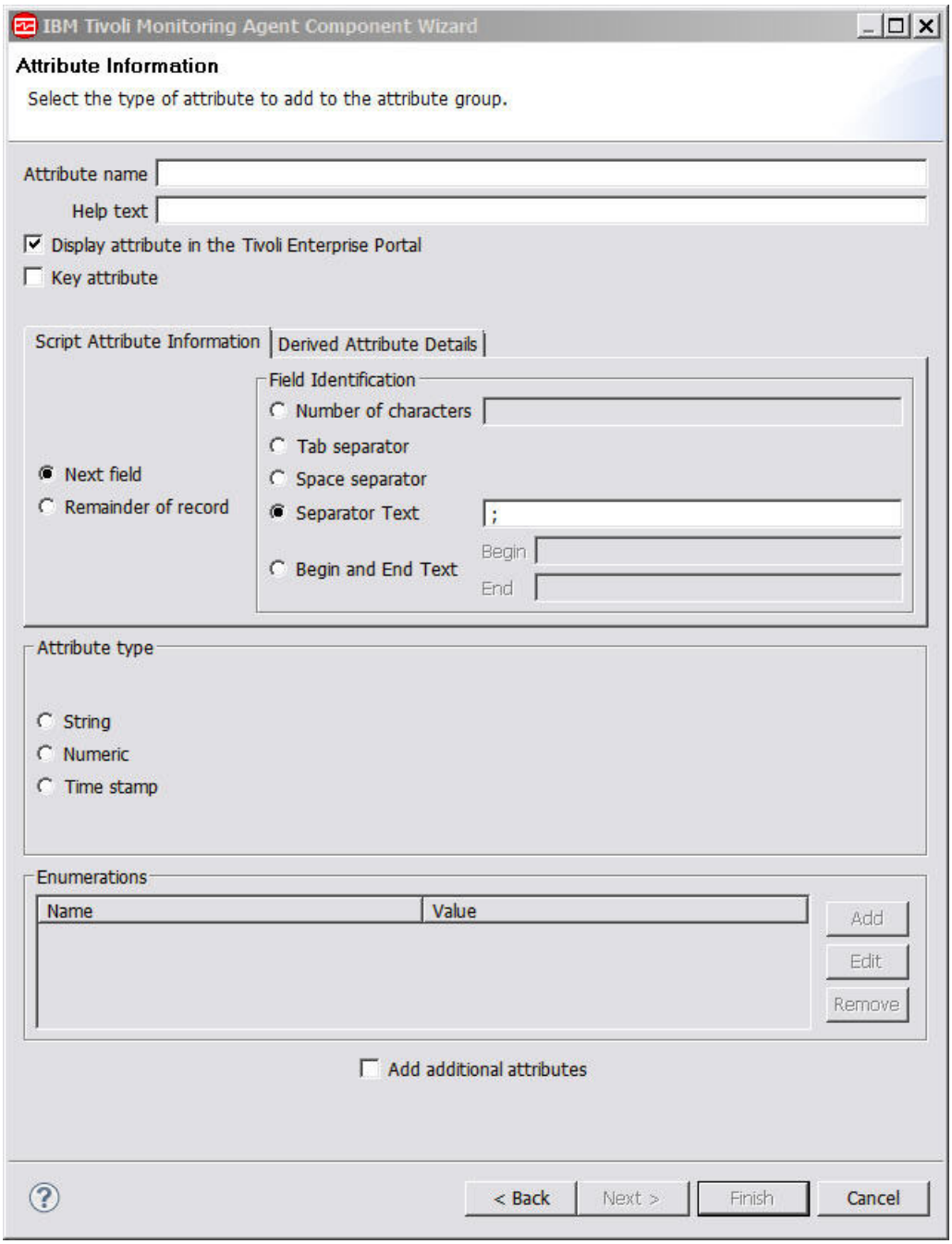


Figure 169. Attribute Information page

11. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

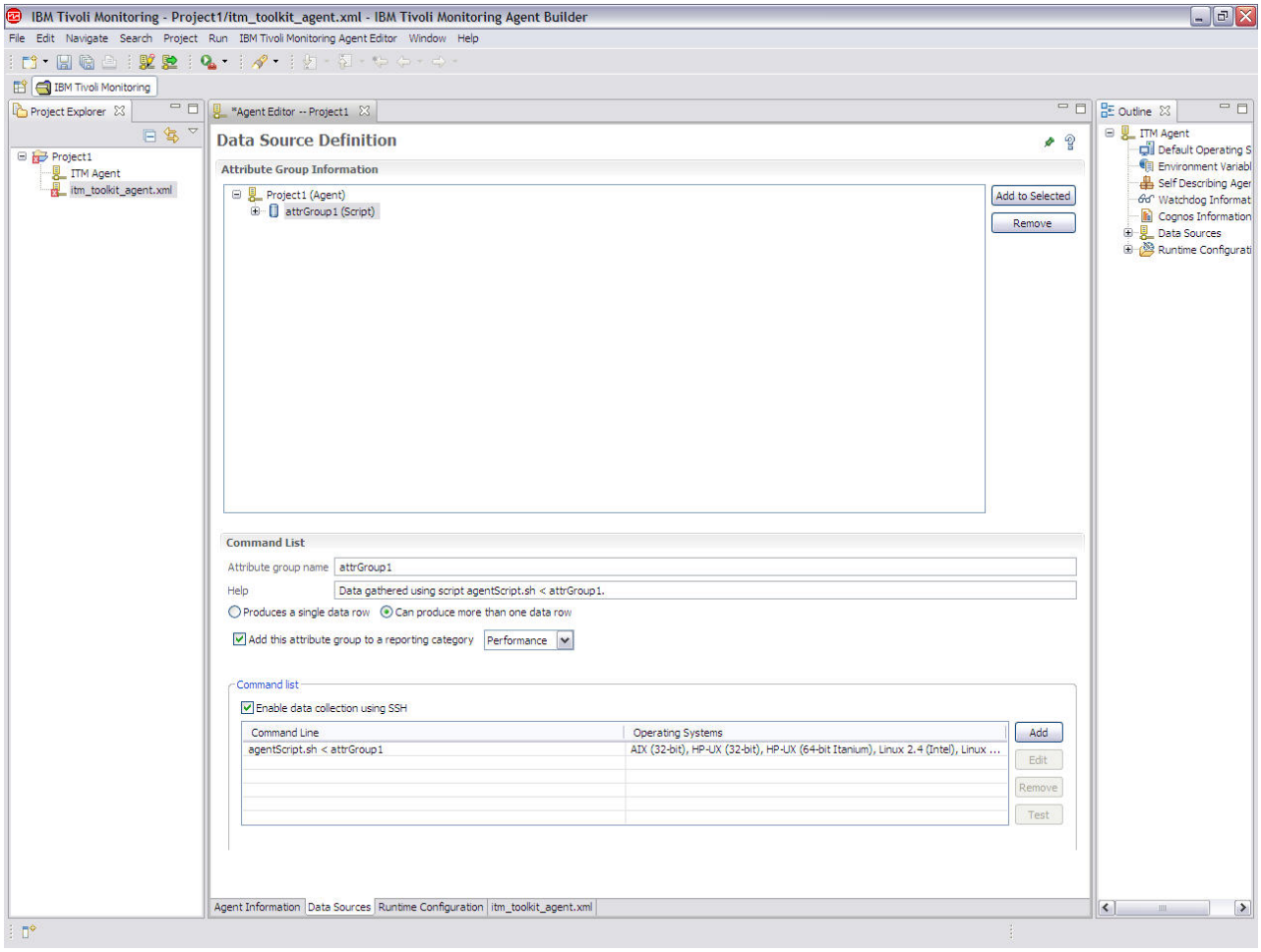


Figure 170. Agent Editor Data Sources Definition page

12. You can add attributes and supply the information for them. For more information, see ("Creating attributes" on page 56). In addition to the fields applicable to all data sources (Table 7 on page 58), the Data Sources Definition page for the Script data source Figure 170 has the following options:

    Command List
    Provides access to the commands and scripts to start during data collection.

    Add Allows the user to add a command to be started by this attribute group.

    Edit Allows the user to edit an existing command entry.

    Remove
    Allows the user to delete an existing command entry.

    Test Allows the user to access the test environment for this attribute group.

    Enable data collection using SSH
    Selecting this check box enables SSH for this attribute group. If this check box is not selected, the attribute group runs locally.

    For information about SSH remote connection configuration for script data sources, see ("Configuring a Secure Shell (SSH) remote connection" on page 375).

Chapter 21. Monitoring data from Java Database Connectivity (JDBC)

Configure your agent to receive data from a JDBC data source.

About this task

Use the JDBC data source to collect data from Agent Builder databases. Each JDBC data source that you define runs an SQL query to collect data from the database. Each column that is returned by the query is a Tivoli Monitoring attribute in the JDBC data source. The JDBC data provider supports the following Database servers:

- IBM DB2 9.x and 8.x
- Microsoft SQL Server 2008, 2005, and 2000
- Oracle database `11g` and `10g`

Agent Builder does not include the JDBC drivers for these databases. The JDBC drivers are a set of JAR files that are provided by the vendor that are necessary to establish a JDBC connection to the database. For convenience, here are links to where those drivers can be downloaded:

- IBM DB2: JDBC drivers are included with the database server installation in a subdirectory named `java` located under the main DB2 installation directory.
- Microsoft SQL Server website at www.microsoft.com
- Oracle database: Oracle Database JDBC

Note: An important thing to remember is that the JDBC data provider can remotely monitor your Database servers. A Java runtime environment and JDBC driver JAR files for the database server you are connecting to must be on the system where the agent runs.

The following versions of Java are supported:

- Oracle Corporation Java Version 5 or later
- IBM Corporation Java Version 5 or later

Procedure

1. On the Agent Initial Data Source page (Figure 171 on page 240) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

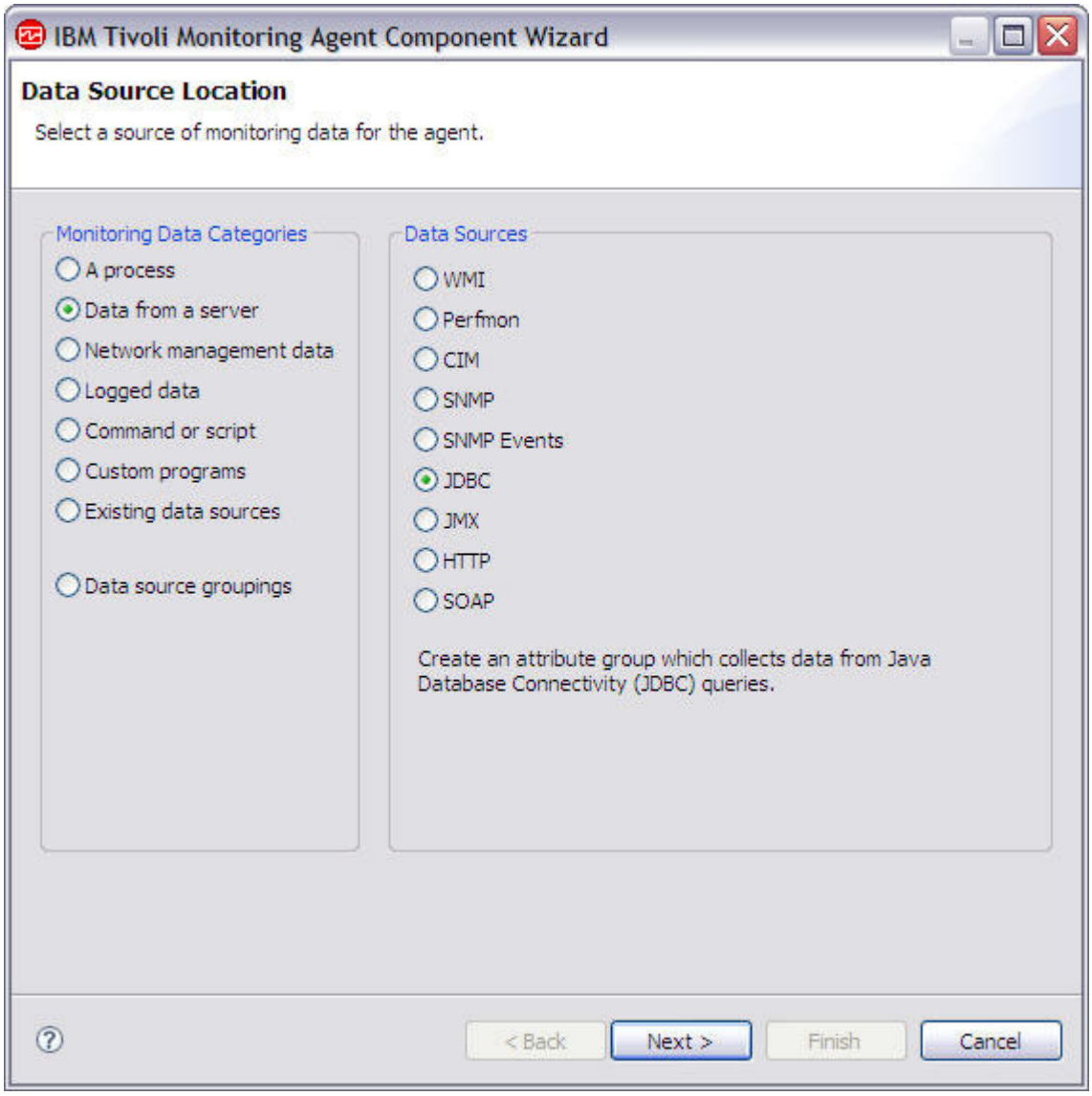


Figure 171. Adding JDBC data

2. In the **Data Sources** area, click **JDBC**.
3. Click **Next**.
4. On the **JDBC Information** area in the JDBC Information page (Figure 172 on page 241), click **Browse** to connect to a database and build your SQL Query.

   Use the JDBC Browser to connect to a database and view its tables so you can build an SQL query that collects the data you need. When you select a table and columns, a query is generated for you. You can modify and test the query that is generated to make sure the data that is returned is what you need.

   Note: You can also manually create the JDBC data source without clicking **Browse**. If you want to manually create the data source, specify the query and click **Next**. You must define the first attribute for this data source in the wizard. Other attributes can be added to the data source later.

   With the JDBC data provider, you can run SQL queries and stored procedures against a database to collect monitoring data. When you specify an SQL query to collect data, you can include a where clause in your SQL statement to filter the data that is returned. The SQL statement can also join data from multiple tables. In addition to SQL select statements, the JDBC data provider can run stored

procedures. For information about running stored procedures, see (“Stored procedures” on page 251).

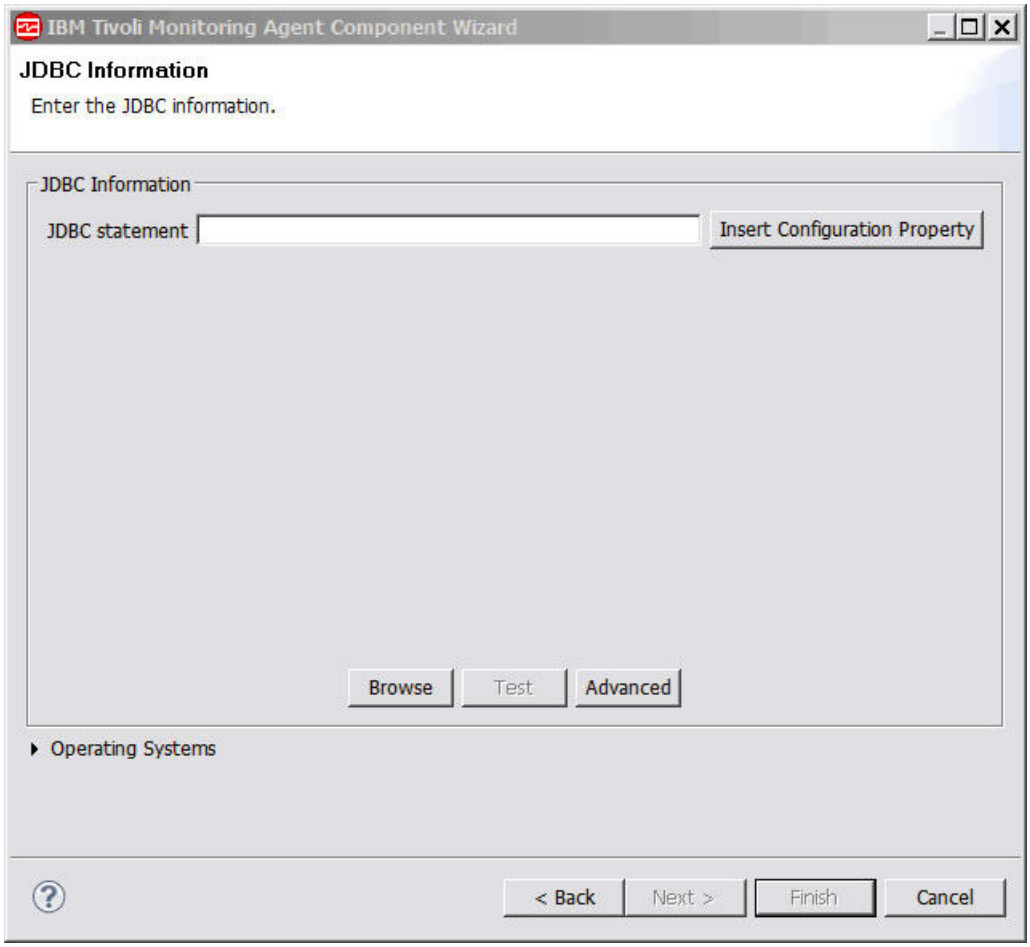


Figure 172. JDBC Information page

5.

Note: The first time the Browser opens, the Java Database Connectivity (JDBC) Browser window (Figure 173 on page 242) indicates that no connections are selected. You must add a connection. Click **Add** and follow the Steps to add a connection. If you already defined a connection, that connection is used and you can proceed to Step (6 on page 244).

Note: The **Status** field shows the status of the current connection.

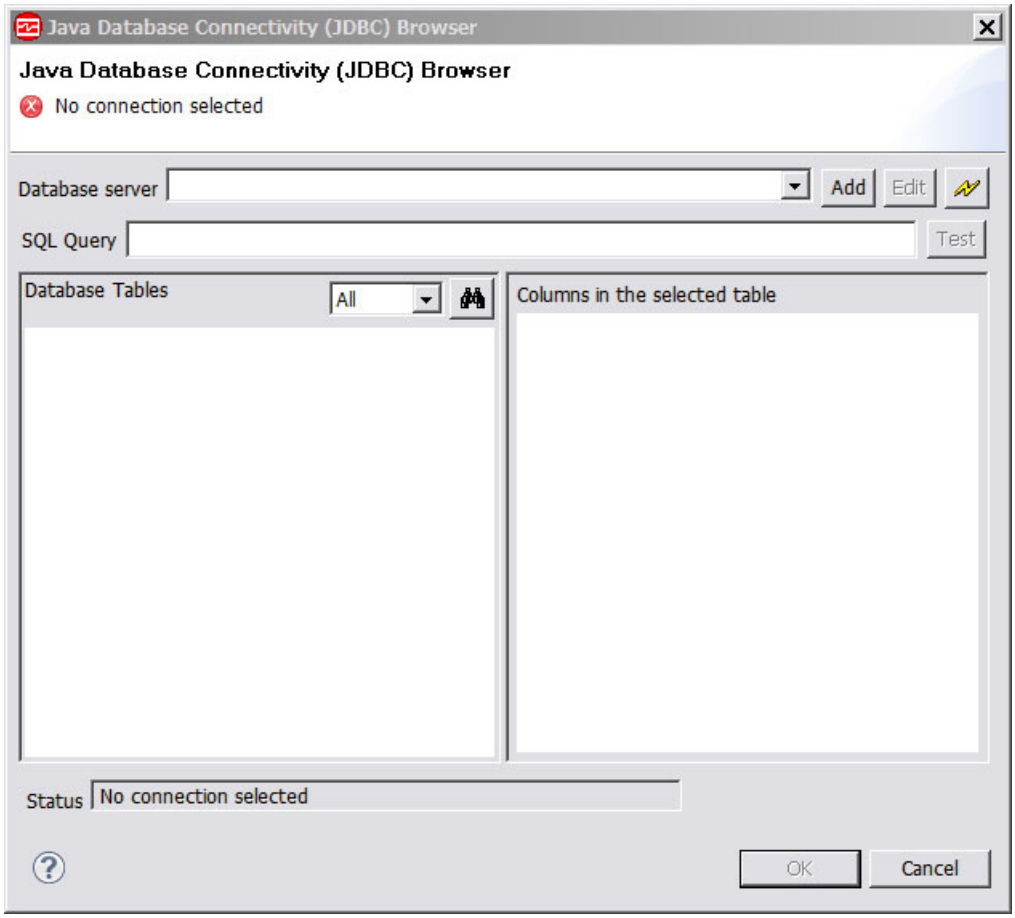


Figure 173. JDBC Browser window

Use the following steps to add a connection:

a. On the JDBC Connections page (Figure 174 on page 243), click **JDBC Connection**, and click **Next**.

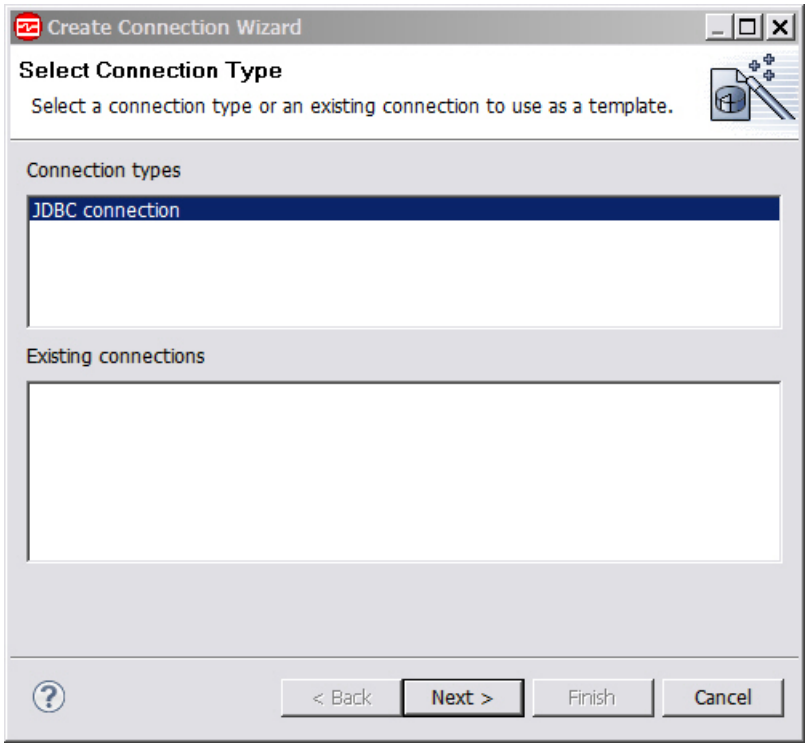


Figure 174. Database Connection Wizard: JDBC Connectionspage

b. On the Connection Properties page (Figure 175 on page 244), complete the fields as follows:

Connection Name
Name of the JDBC connection. Type a unique name for this connection. You use this name to reference the connection in the browser.

Database Type
Type of database. Select the database product to which you are connecting. For example, to connect to the IBM DB2 database, select **DB2**.

User Name
Must be defined with at least read access to the database, but does not have to be the database administrator

Password
Must be defined with at least read access to the database, but does not have to be the database administrator

Host name
Host name on which the database server is running. With JDBC, you can monitor remote databases so you are not restricted to monitoring databases on the local system.

Port Port on the host name on which the database server is listening.

Database
Name of the database to which to connect.

Jar Directory
Directory containing the JDBC JAR files used to connect to the database. Type the path name, or click **Browse** to locate the directory.

c. Optional: Select the **Save the password in the Agent Builder workspace** check box if you want to save the password for this connection.

d. Optional: Select the **Set as agent configuration defaults** check box if you want the defaults for this application server type to be copied from these properties. If you are building the agent on a system that is similar to your monitored systems, it is advisable to check this box. If you do not check this box, the user who configures the agent sees an empty field. The user must then determine the values for all of the information without default values.

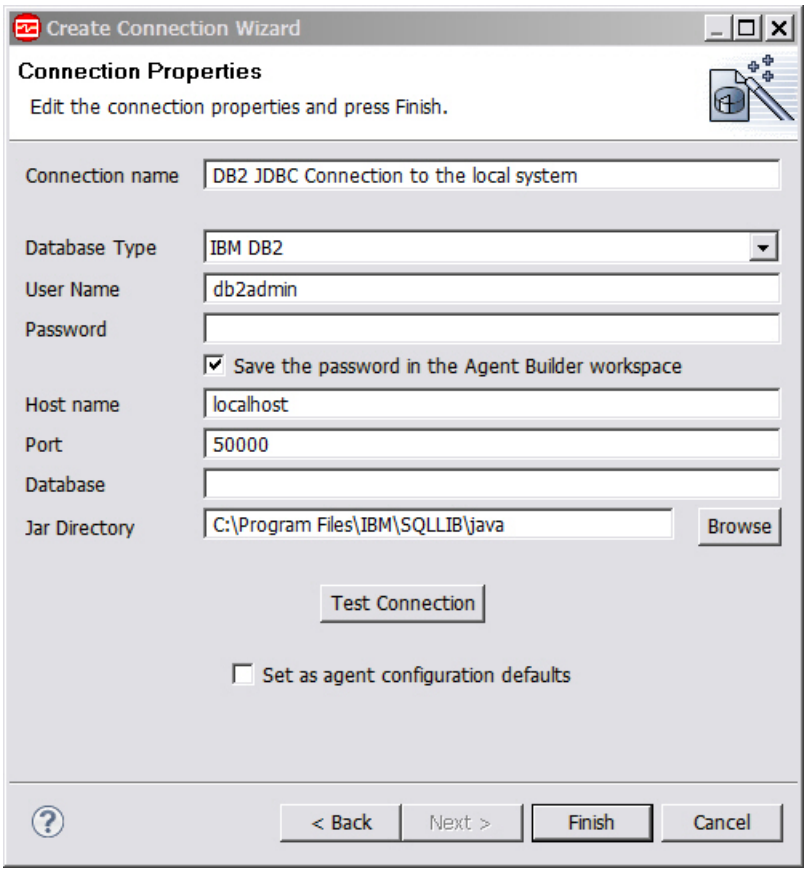


Figure 175. Database Connections Wizard: Connection Properties page

e. Click **Test Connection** to create a connection to the database that uses the configuration parameters you specified. A message on the Connection Properties page indicates whether the connection succeeds.

f. When you have a working connection, click **Finish**.

6. In the Java Database Connectivity (JDBC) Browser window (Figure 176 on page 245), a connection is made to the configured database. The tables that are contained in the database are shown in the **Database Tables** area. Select a database table to see the columns that are contained in that table in the **Columns in the selected table** area.

   Note:

   a. Click the binoculars icon to search for a table in the **Database Tables** list.

   b. All tables are shown by default. You can filter the tables that are shown by selecting a different filter option. The available filter options are shown in (Table 13).

Table 13. Filter options

Filter option	Description
All	Show all tables
User	Show only user tables

Table 13. Filter options (continued)

Filter option	Description
System	Show only system tables
View	Show only database views

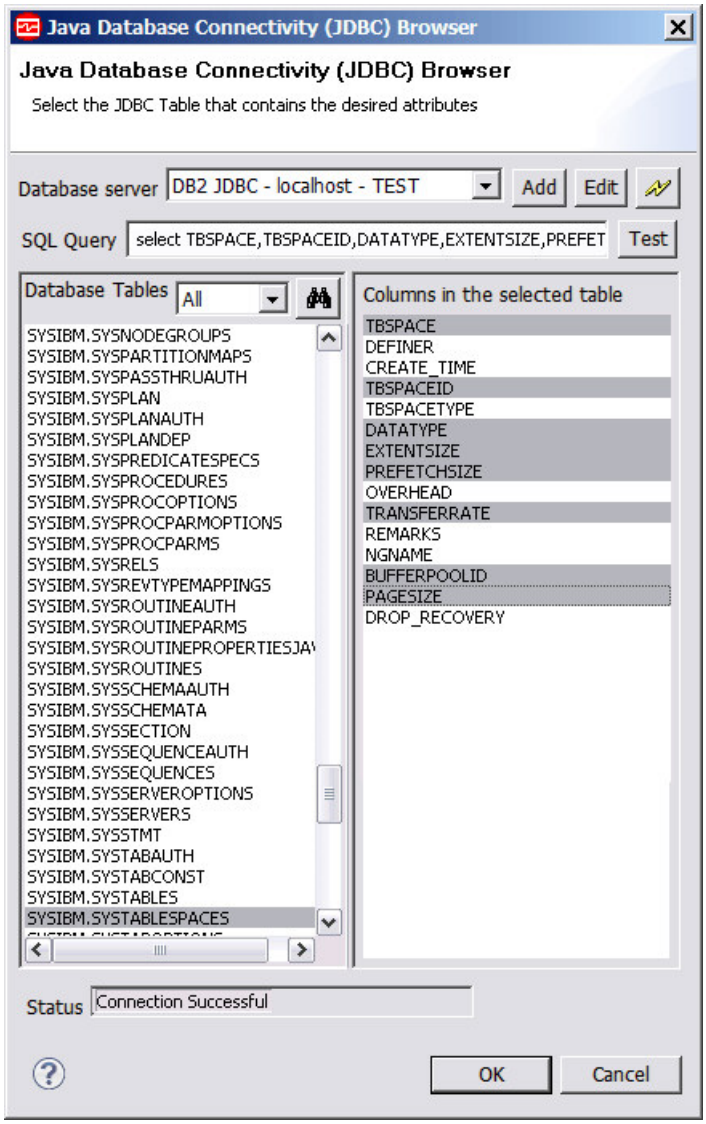


Figure 176. Java Database Connectivity (JDBC) Browser window

Note: Select only columns if you want to retrieve specific columns. If you select the table, it automatically builds a query that gathers all of the columns from the table.

You can select columns in the following ways:

- Select the table and get the default query for all columns.
- Select columns to get only those columns.

7. Optional: Click **Test** on the Java Database Connectivity (JDBC) Browser window to test and modify the SQL statement. The Run the SQL statement window opens.
   a. Enter or modify the SQL statement in the **SQL statement** field.

b. Click **Run** to run the SQL statement. The results are displayed in the Results area. Continue to modify and test the statement until you are satisfied with the data that is returned.
c. Click **OK** to save the statement and return to the JDBC Information window.

8. Optional: Click **Test** on the JDBC Information window, (Figure 177) to test the attribute group in a more realistic agent environment. For more information about testing JDBC attribute groups, see "Testing JDBC attribute groups" on page 253
9. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 67
10. On the JDBC Information page (Figure 177), **Operating Systems** section, select the operating systems, and click **Next**. See "Specifying operating systems" on page 88 for information about which operating systems to select.

    Note: Click **Insert Configuration Property** to select a property to insert. For more information, see (Chapter 30, "Customizing agent configuration," on page 361).

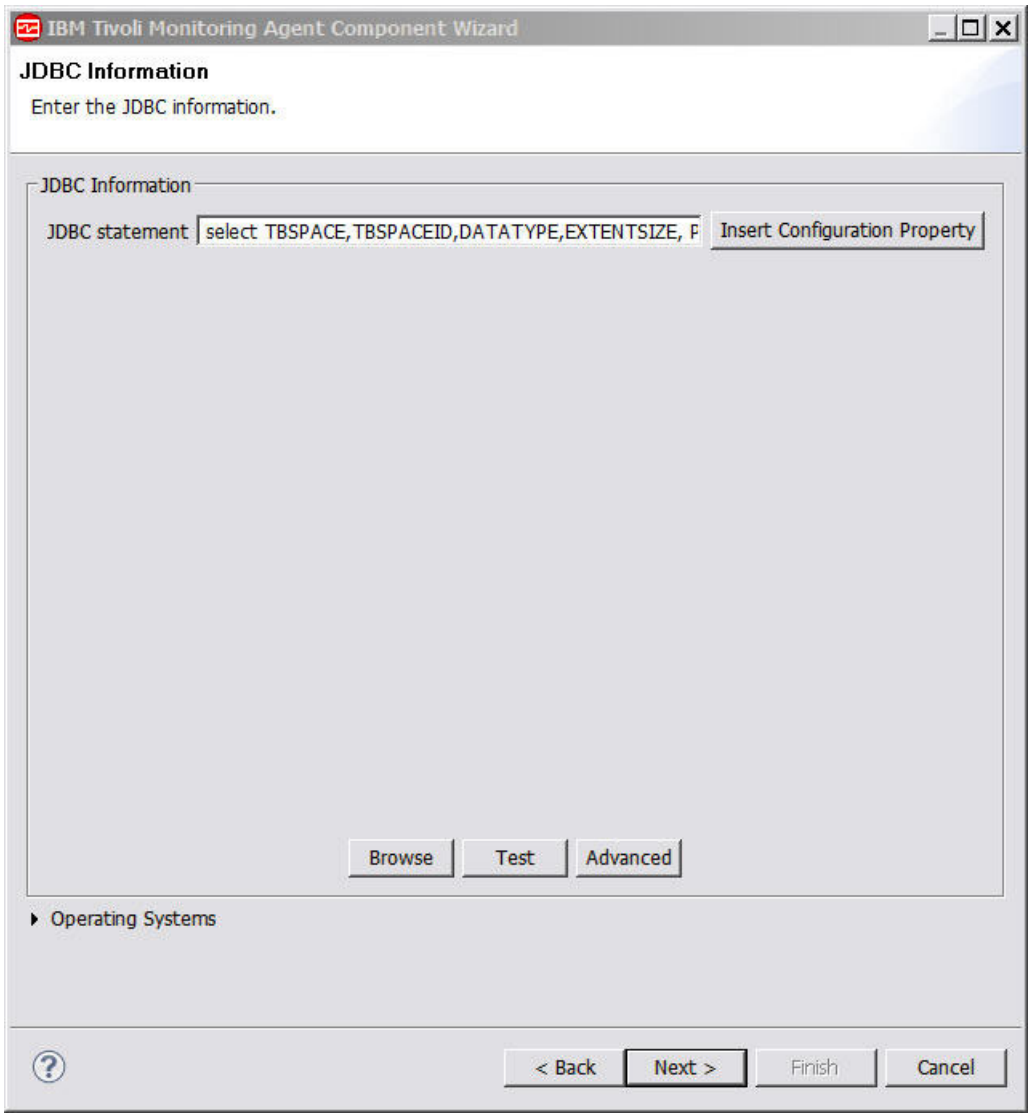


Figure 177. JDBC Information page

11. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 26).
12. If you want to test a data source that you previously defined, in the Agent Editor window, select the **Data Sources** tab and select a JDBC data source. In the JDBC Attribute Group Information area, click **Test**. For more information about testing, see ("Testing JDBC attribute groups" on page 253).

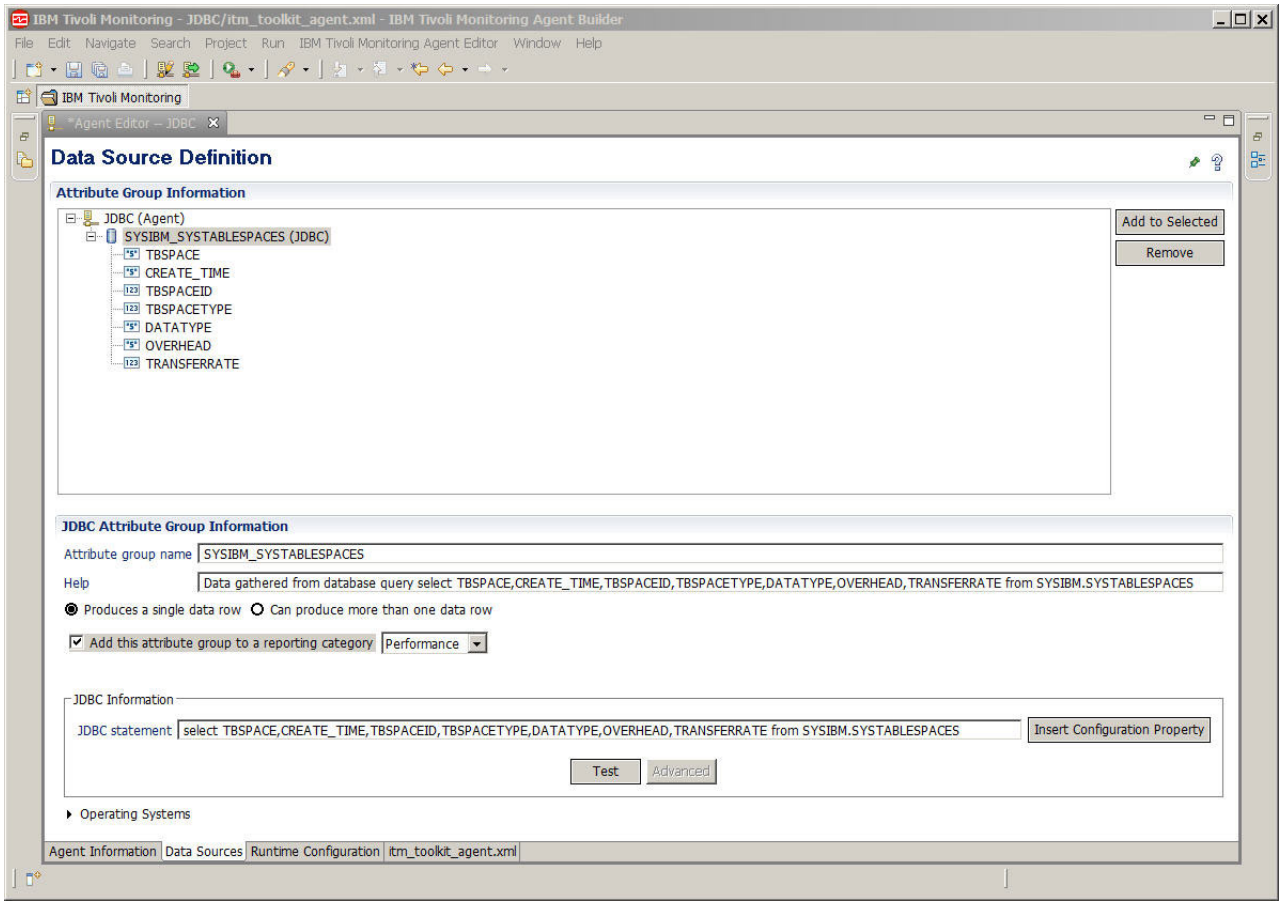


Figure 178. Data Source Definition page Attribute Group Information

13. If you want to view the configuration sections that were automatically generated, click the **Insert Configuration Property** tab of the Agent Editor (Figure 179 on page 248). You can change the labels or default values for these properties to match the defaults that the user sees when they initially configure the agent.

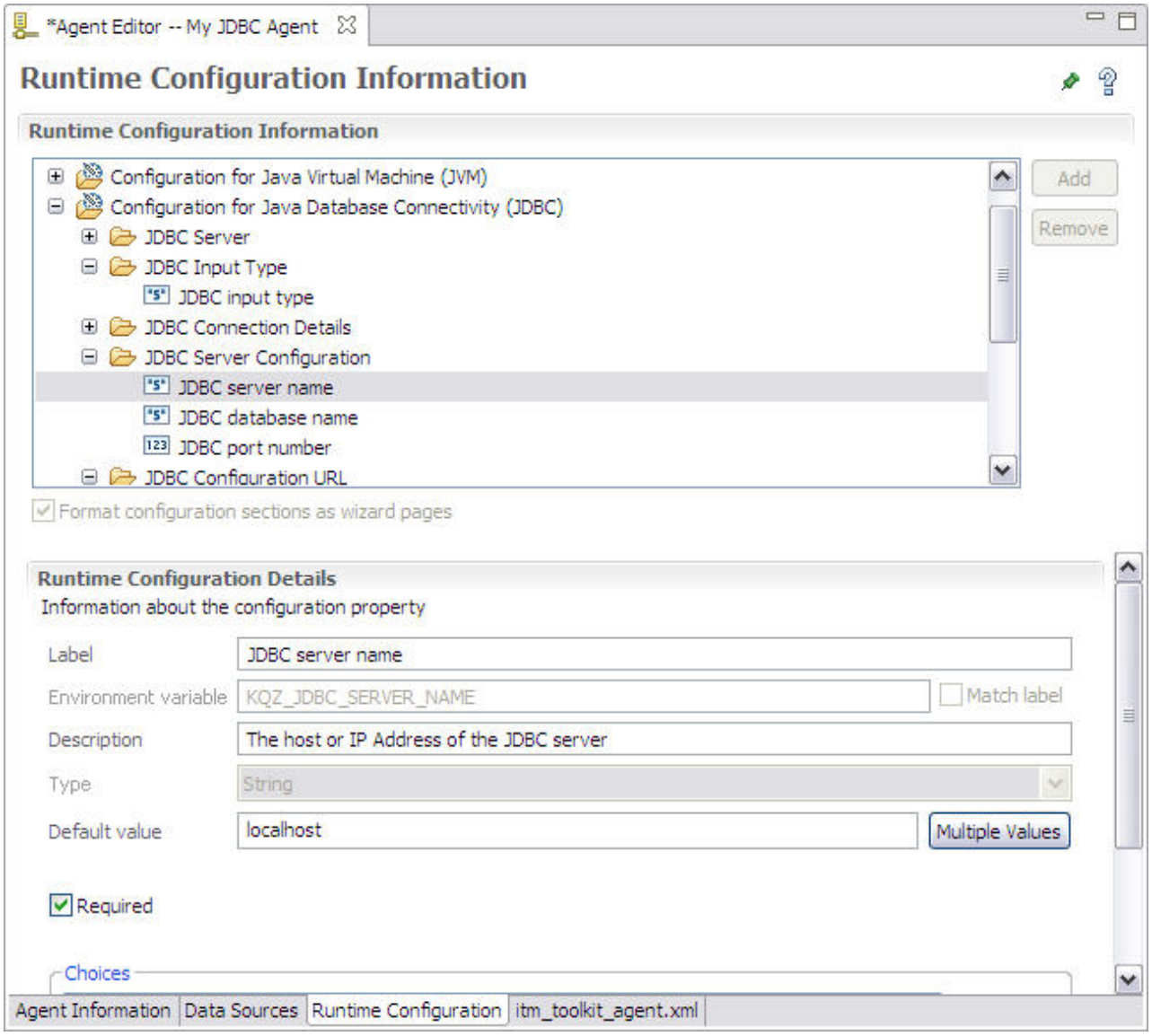


Figure 179. Runtime Configuration Information page, Configuration for Java virtual machine (JVM)

14. Optional: Complete the Attribute Information page by using (Table 7 on page 58). Do this step if you chose to manually create the JDBC data source without clicking Browse in step (4 on page 240).

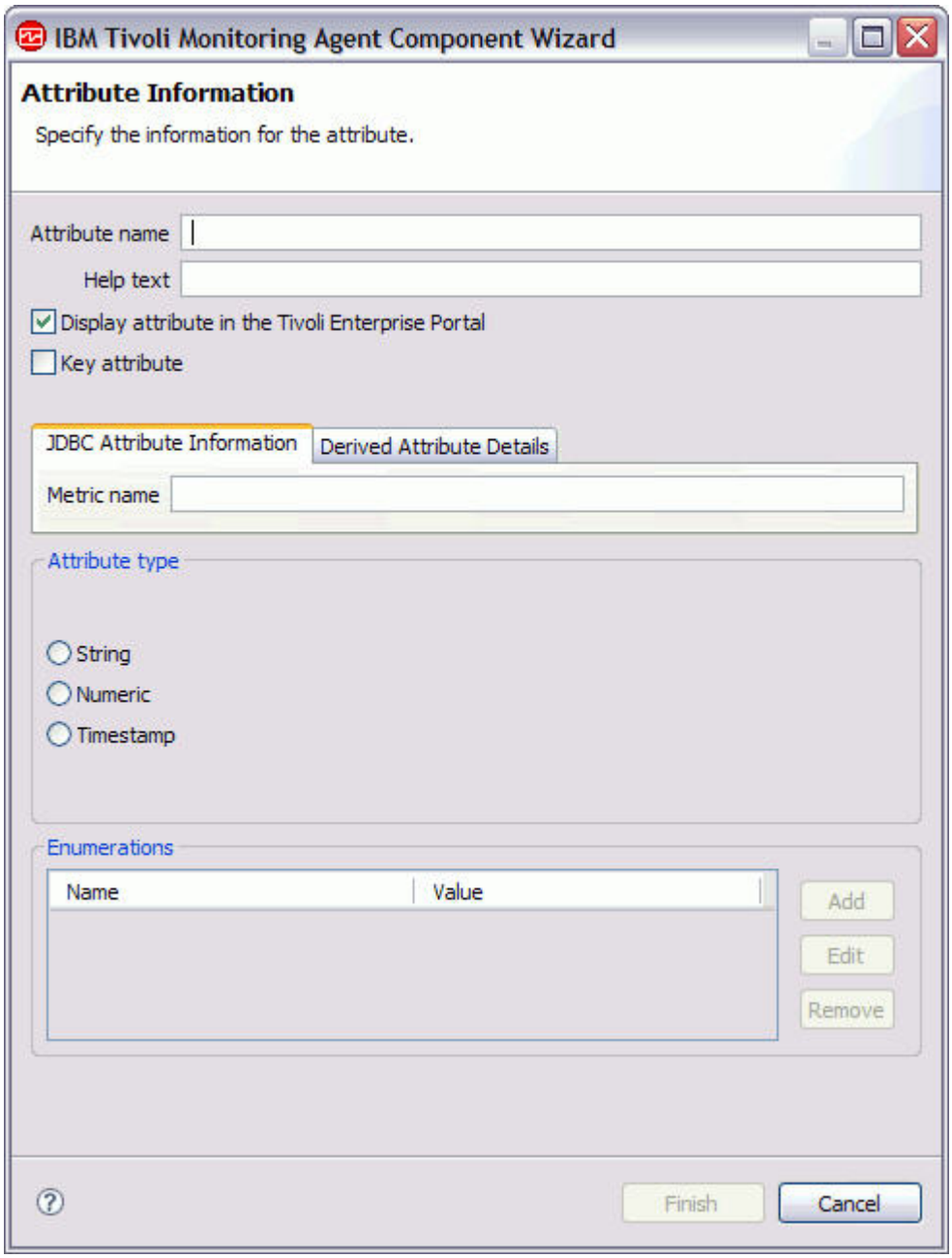


Figure 180. Attribute Information page

The Agent Builder JDBC data source supports collecting data from most SQL types. The information in (Table 14) describes the type of attribute that is created by the JDBC Browser when it detects a column of one of these types. These data types are the supported types for use with a monitoring agent.

Table 14. Supported SQL data types for use with a monitoring agent

SQL data type	IBM Tivoli Monitoring attribute that is created
BIGINT	This data type is a 64-bit gauge value in IBM Tivoli Monitoring. If you select IBM Tivoli Monitoring V6.2 compatibility, it is a 32-bit gauge.

Table 14. Supported SQL data types for use with a monitoring agent (continued)

SQL data type	IBM Tivoli Monitoring attribute that is created
DECIMALDOUBLEFLOATNUMERICREAL	These SQL Types are created as 64-bit gauge attributes in IBM Tivoli Monitoring. If the database metadata contains a scale value, that value is used; otherwise, the scale is set to 1. If you select IBM Tivoli Monitoring V6.2 compatibility, the attribute is a 32-bit gauge.
BITINTEGERSMALLINTTINYINT	The following SQL types are created as 32-bit gauge attributes in IBM Tivoli Monitoring.
BOOLEAN	This value is a 32-bit gauge in IBM Tivoli Monitoring with enumerations for TRUE and FALSE.
TIMESTAMP	Data in columns of this type are converted to a 16-byte IBM Tivoli Monitoring time stamp attribute.
TIMEDATECHARLONGVARCHARVARCHAR	These SQL types are all treated as string attributes by the browser. The column size is used as the attribute size up to 256, which is the default string attribute size for the JDBC browser.

Note: If you collect data from a data type that is not listed, a string attribute is used by default. The agent also tries to collect the data from the database as a string.

JDBC configuration

When you define a JDBC data source in your agent, some configuration properties are created for you.

If you define a JDBC data source in your agent, the agent must use Java to connect to the JDBC database server. Java configuration properties are added to the agent automatically. The following Java configuration properties are specific to the agent runtime configuration:

- Java Home: A fully qualified path that points to the Java installation directory
- JVM Arguments: Use this parameter to specify an optional list of arguments to the Java virtual machine.
- Trace Level: This parameter defines the amount of information to write to the Java trace file. The default is to write-only Error data to the log file. For more information, see ("Trace log format" on page 426).

Note: The Agent Builder does not require the Java properties because it uses its own JVM and logging, which are configured through the `JLog` plug-in.

If you define a JDBC data source in your agent, the following required, common configuration fields are added to the agent automatically:

- *JDBC database type*: Type of database to which you are connecting, IBM DB2, Microsoft SQL Server, or Oracle Database Server.
- *JDBC user name*: User name that is used to authenticate with the database server.
- *JDBC password*: Password that is used to authenticate with the database server.
- *Base paths*: List of directories that are searched for JAR files that are named in the *Class Path* field, or directories that are named in the *JAR directories* field, that are not fully qualified. Directory names are separated by a semi-colon `(;)` on Windows, and by a semi-colon `(;)` or colon `(:)` on UNIX systems.
- *Class path*: Explicitly named JAR files to be searched by the agent. Any files that are not fully qualified are appended to each of the Base Paths until the JAR file is found.
- *JAR directories*: List of directories that are searched for JAR files. Directory names are separated by a semi-colon `(;)` on Windows, and by a semi-colon `(;)` or colon `(:)` on UNIX systems. The JAR files in

these directories do not have to be explicitly identified; they are found because they are in one of these directories. Subdirectories of these directories are not searched. Any directories that are not fully qualified are appended to each of the Base Paths until the directory is found.

The runtime configuration also requires that you specify some additional details to connect to the database. You can choose how to specify the remaining configuration items, either as a JDBC URL or as basic configuration properties (the default):

- URL configuration option
  - JDBC connection URL: Vendor-specific connection URL that provides details on which host the database is located and the port number to which to connect. The URL format typically looks as follows:

    `jdbc:identifier://server:port/database`

    see the JDBC driver vendor documentation for the different URL formats.
- JDBC Basic Properties option (default)

  JDBC server name: Host name that the database server is running on.

  JDBC database name: Name of the database on the host where the connection is made.

  JDBC port number: Port number on which the database server is listening.

Note: With the JDBC data provider, you can monitor multiple database types in the same agent by using subnodes. To monitor in this way, you must carefully define the Subnode Configuration Overrides. If you monitor multiple database types, the following configuration settings are likely to be different:

- JDBC database type
- JDBC user name
- JDBC password

If you are using the basic configuration option, you must also define overrides for the following properties on the Subnode Configuration Overrides page:

- JDBC server name
- JDBC port number
- JDBC database name

To define the configuration overrides for your subnode, see Chapter 29, "Use of subnodes," on page 331 for more details about accessing the Subnode Configuration Overrides page. When you configure the agent at run time, all of these properties must be configured for each new subnode instance that is created.

In addition to configuration overrides, your agent must also point to JDBC drivers for each database type that you plan to connect to from your subnodes. The *JAR directories* parameter is the most convenient way to point to your JDBC drivers. List the directories that contain the JDBC drivers by using a semicolon to separate each directory. For example, if you are connecting to DB2 and Oracle databases with the agent, you must specify a *JAR directories* value similar to this example: `C:\Program Files\IBM\SQLLIB\java;C:\oracle\jdbc`.

Stored procedures

Example SQL and DB2 stored procedures that you can use with the JDBC data provider.

The JDBC data provider can process the result sets returned by a stored procedure. String or integer input parameters can be passed to the stored procedure. The following syntax runs a stored procedure:

`call[:index] procedureName [argument] ...`

Where:

index
An optional integer that specifies which result set is to be used by the data provider. This parameter is useful when the stored procedure returns multiple result sets and you want to collect only the values from one of the result sets. If an index is not specified, data from each result set is collected and returned.

procedureName
The name of the stored procedure that is to be run by the JDBC data provider.

argument
An input argument to the stored procedure. Multiple arguments must be separated by a space. If the argument contains a space character, enclose the entire argument in double quotation marks. If the argument can be parsed as an integer, it is passed to the stored procedure as an integer argument. Any argument that is enclosed in double quotation marks is passed as a string argument.

SQL Server Samples

call sp_helpdb

Runs the procedure `call sp_helpdb` which requires no arguments. Data from all returned result sets are included in the data that is returned by the data provider.

call:2 sp_helpdb master

Runs the procedure `sp_helpdb` with the master argument. This argument is a string input argument. Only data from the second result set that is returned by the stored procedure is included in the data that is returned by the data provider.

When the index is not specified, data from all returned results sets is collected. You must ensure that the data returned in these cases is compatible with the attributes you define. Agent Builder creates attributes from the first returned result set, and any further result sets are expected to be compatible with the first one.

DB2 stored procedure

Here is a sample DB2 function that is written in SQL. This function demonstrates how to return results that can be processed by the Agent Builder JDBC data provider:

```
-- Run this script as follows:
-- db2 -td# -vf db2sample.sql

-- Procedure to demonstrate how to return a query from
-- a DB2 stored procedure, which can then be used by
-- an Agent Builder JDBC provider. The stored procedure
-- returns the following columns:
-- Name               Description              Data Type
-- current_timestamp  The current system time  timestamp
-- lock_timeout       The lock timeout         numeric scale 0
-- user               The user for the session String 128 characters long
DROP procedure db2sample#

CREATE PROCEDURE db2sample()
  RESULT SETS 1
  LANGUAGE SQL
BEGIN ATOMIC

  -- Define the SQL for the query
  DECLARE c1 CURSOR WITH HOLD WITH RETURN FOR
  SELECT CURRENT TIMESTAMP as current_timestamp,
CURRENT LOCK TIMEOUT as lock_timeout, CURRENT USER as user
  FROM sysibm.sysdummy1;
```

```
  -- Issue the query and return the data
  OPEN c1;
END#
```

This function can be called from Agent Builder by using the same syntax that is defined for other stored procedures. In this case, you define `call db2sample` as your JDBC statement to run this stored procedure.

Oracle stored procedures

Oracle stored procedures do not return result sets. Instead, you must write a function that returns an Oracle reference cursor. Here is a sample Oracle function that is written in `PL/SQL` that demonstrates how to return results that can be processed by the Agent Builder JDBC data provider:

```
CREATE OR REPLACE FUNCTION ITMTEST
RETURN SYS_REFCURSOR
IS
        v_rc SYS_REFCURSOR;
BEGIN
  OPEN v_rc FOR SELECT * FROM ALL_CLUSTERS;
  RETURN v_rc;
END;
```

This function can be called from Agent Builder by using the same syntax that is defined for other stored procedures. In this case, you define `call ITMTEST` as your JDBC statement to run this stored procedure. Because the Oracle function must return a cursor reference, only one result set can be processed by Oracle functions. This means that the index option is not supported for Oracle because there is no way to return multiple result sets.

Testing JDBC attribute groups

You can test the JDBC attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the JDBC Information page (Figure 177 on page 246).
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see (Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35).

   After you click **Test** in one of the previous two steps, the Test JDBC Statement window Figure 181 on page 254 is displayed

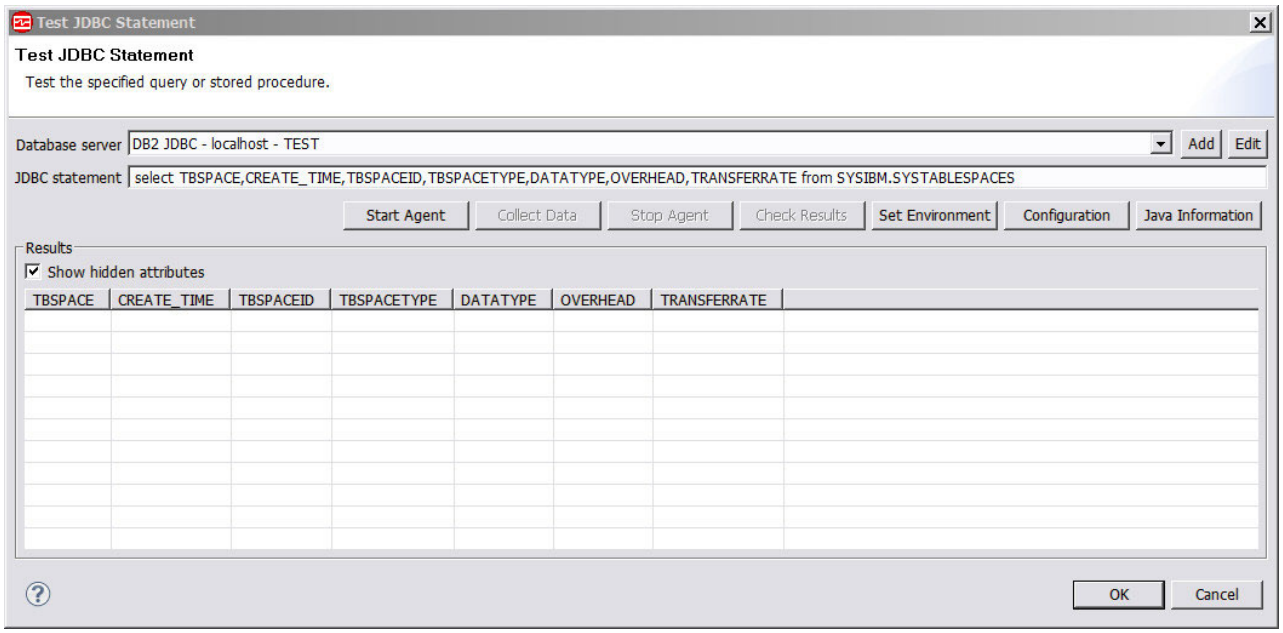


Figure 181. Test JDBC statement window

2. Optional: Before you start testing, you can set environment variables, configuration properties, and Java information. For more information, see ("Attribute group testing" on page 377). For more about JDBC configuration properties, see ("JDBC configuration" on page 250).
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries the database with the specified SQL query. The Test JDBC Statement window collects and shows any data in the agent's cache since it was last started.

   Note: The order of the returned data is significant; for example, the data value in the first returned column is always assigned to the first attribute. If you reorder the attributes, you must rebuild your query to have a matching order.
5. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 545
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the Test JDBC Statement window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, "Testing and debugging your agent," on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 22. Monitoring system availability by using Ping

Configure your agent to receive data from a list of systems by using the Internet Control Message Protocol (ICMP).

Before you begin

Create a ping configuration file, and a device list file. For more information about ping configuration files, see ("Ping configuration files" on page 257).

About this task

Part of network management involves the ability to determine whether systems respond to an Internet Control Message Protocol (ICMP) ping. Use this data source to monitor basic online or offline status for a set of servers or other critical devices in your environment. Monitoring with ping is simple and low-overhead. To monitor a list of systems, add the Ping data collector to your agent.

Procedure

1. On the Agent Initial Data Source page (Figure 182 on page 256) or the Data Source Location page, click **Network management data** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

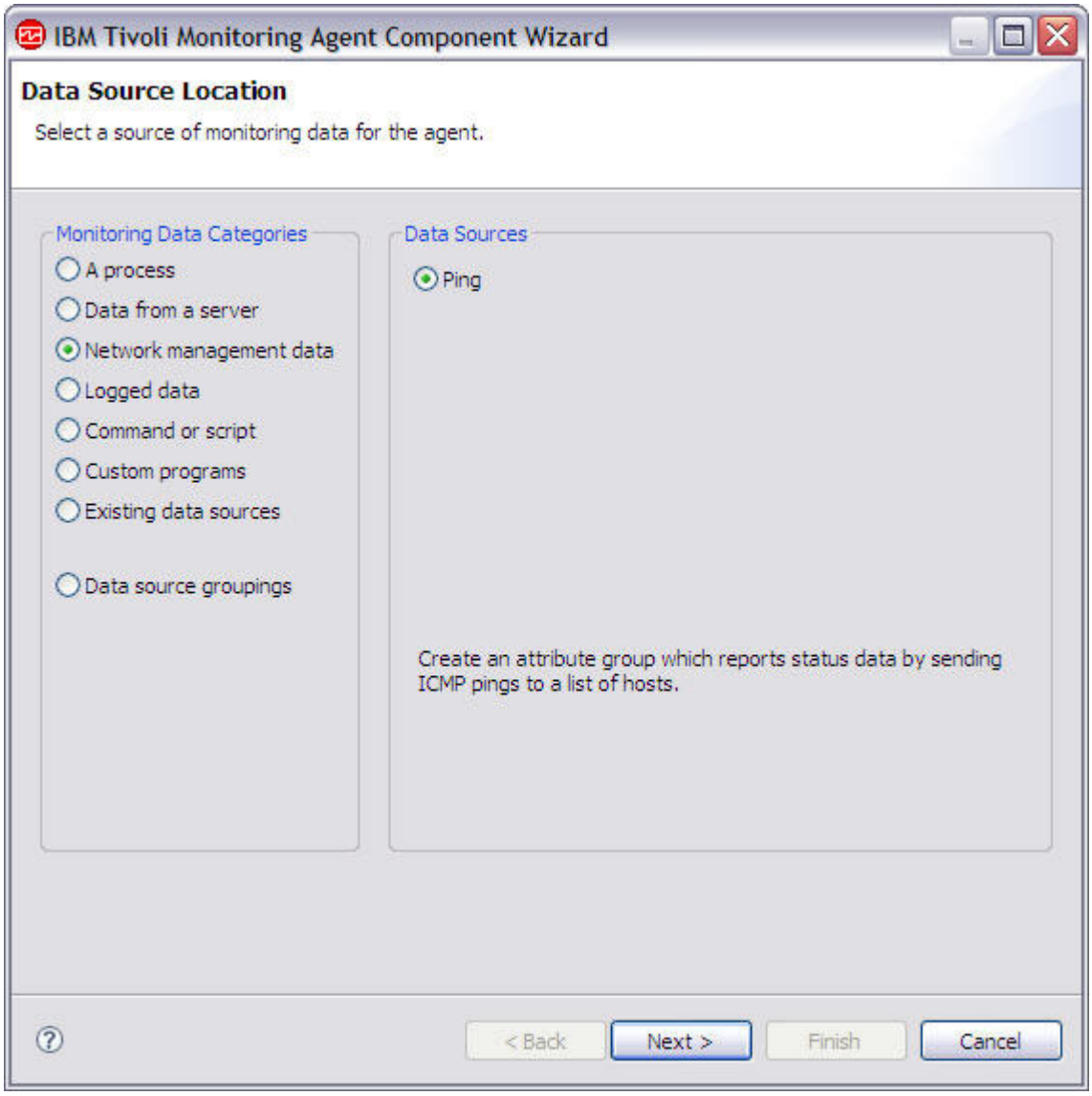


Figure 182. Adding Ping data

2. In the **Data Sources** area, click **Ping**.
3. Click **Next**.
4. In the **Operating Systems** area in the Ping Information window (Figure 183 on page 257), select the operating systems.

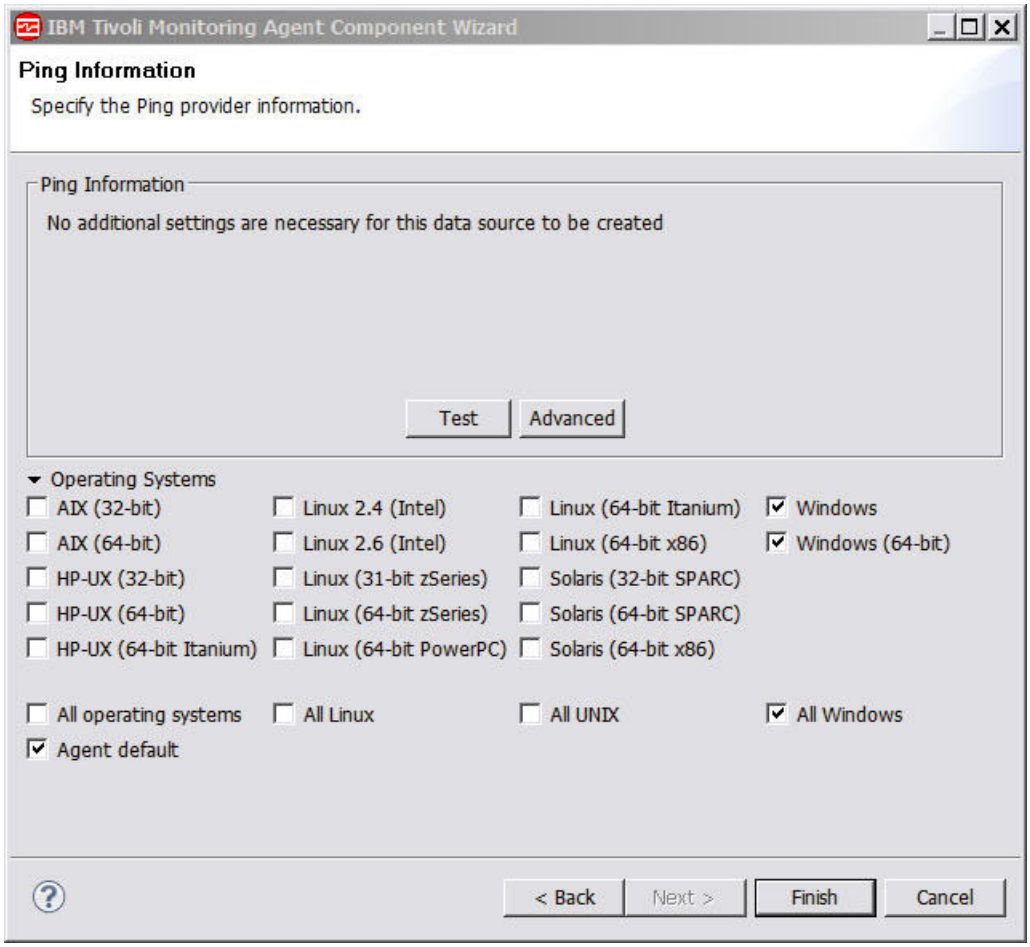


Figure 183. Ping Information window

5. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing ping attribute groups." on page 259
6. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 67
7. Do one of the following steps:
   a. If you are using the New Agent wizard, click **Next**.
   b. Click **Finish** to save the data source and open the Agent Editor.
8. For more information about adding attributes, see ("Creating attributes" on page 56).

Results

For more information about the attribute group for Ping, see ("Ping attribute group" on page 573).

Ping configuration files

The agent must be provided with the list of systems to be monitored by using ping configuration files.

The files that are used are not managed or updated by the agent; the user must make all required updates to the files. Both a ping configuration file and a device list file must be created.

The ping configuration file is a text file that must be specified as a runtime configuration parameter to the agent. The contents of each line in this file must be the location of a device list file. Each entry in the ping configuration file causes the agent to start a separate thread for monitoring the devices in the provided device list file. This thread loops through all the devices in the device list file, sending each a ping request. These threads run at intervals of 60 seconds or at intervals of 30 seconds plus the time it takes to ping the list, whichever is longer.

Note:

- Each time before the agent monitors the devices in the device list file, it checks the last modification time of the device list file. If the last modification time of the file is more recent than the last time the agent read the file, the agent rereads the file without requiring a restart.
- Each entry to a device list file in the ping configuration file must contain either a fully qualified path or a path relative to the location of the ping configuration file.

A device list file is a text file that contains a `LISTNAME` definition (optional) and a list of device and host names. For example:

```
LISTNAME=TivoliWeb
www.ibm.com
mercury
us07 ishtar
```

The list of device and host names can be separated by spaces and line breaks. In the example, `us07` and `ishtar` are separate devices.

`LISTNAME definition`. The defined list name becomes the description for the devices in that file. If no list name is defined, the name of the device list file is used. If the list name is defined, the list name definition must take the following form: `LISTNAME=listname` The list name specification does not need to be the first record of the managed node list definition. However, if the list name is defined more than one time in the description, the last specification is used as the definition for the systems.

`Device and host names`. There is no limit to the number of network devices and hosts that you can include in a device list file. However, including too many entries defeats the purpose of having a targeted list of critical devices and increases the overall workload. There might also be difficulties in obtaining a status on each device within the 60-second monitoring interval. You can specify one or more network devices in each record.

Duplicate entries in the device list file are filtered only if you enter the same name more than one time. If multiple names resolve to the same IP address, each entry is listed, but only one ping is sent to that device by the monitoring thread. If you specify the same devices in multiple device list files, those duplicate entries are not detected. As a result a device can be pinged by more than one of the device monitoring threads.

Ping configuration properties

After a ping data source is added, the configuration is displayed on the Runtime Configuration Information page of the Agent Editor.

The **Network Management** configuration section of the Runtime Configuration Information page contains the following property:

Table 15. Network Management configuration properties

Name	Valid values	Required	Description
Ping configuration file	Path to a file	No. If this file is not provided, the KUMSLIST file is used from the agent bin directory.	The path to the file that contains a list of files, each containing a list of hosts to monitor by using ICMP pings.

Testing ping attribute groups.

You can test the Ping attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the Ping Information page (Figure 183 on page 257).
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test** . For more information about the Agent Editor, see (Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35).

   After you click **Test** in one of the previous two steps, the Test Settings window opens.

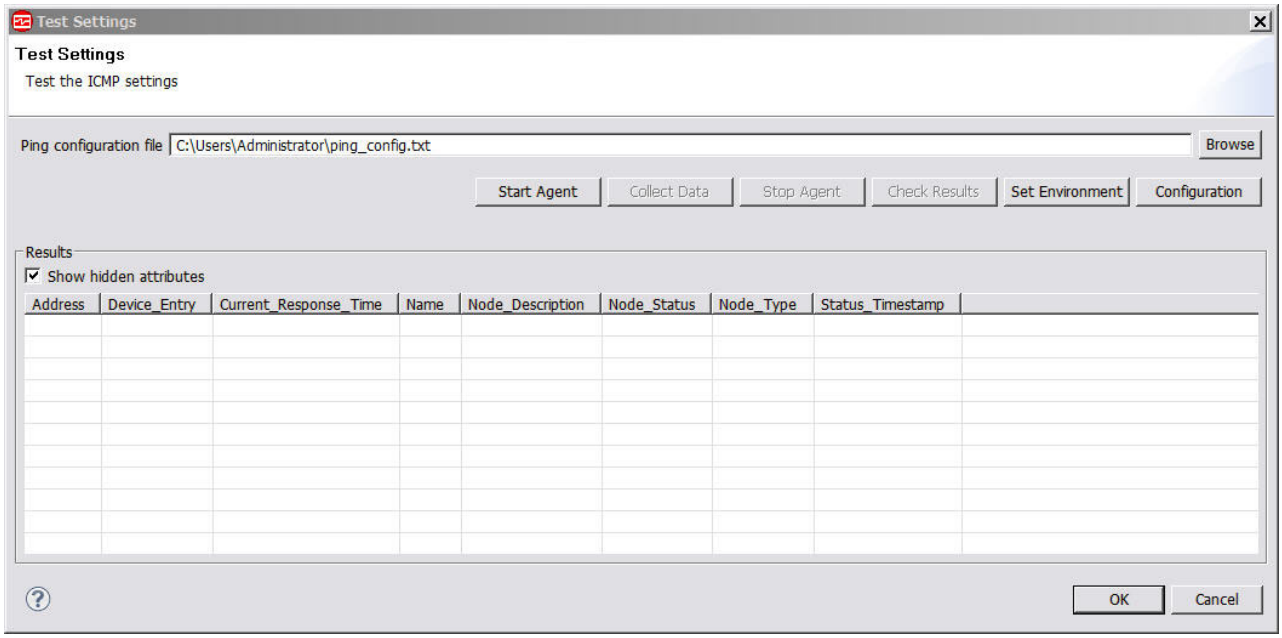


Figure 184. Ping Test Settings window

2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see ("Attribute group testing" on page 377).
3. Click **Browse** to select a Ping configuration file. For more about Ping configuration files, see "Ping configuration files" on page 257
4. Click **Start Agent**. A window indicates that the Agent is starting.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent pings the devices that are specified in the device list file which is referenced from the Ping configuration file.
6. The Test Settings window collects and shows any data in the agent's cache since it was last started.

7. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and shown by the Data collection Status window is described in ("Performance Object Status node" on page 545).
8. Stop the agent by clicking **Stop Agent**.
9. Click **OK** or **Cancel** to exit the Test Settings window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, "Testing and debugging your agent," on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 23. Monitoring HTTP availability and response time

You can monitor the availability and response time of selected URLs by using an HTTP data source.

About this task

For each URL you monitor, the results provide general information about the HTTP response to the HTTP request. The results include whether it can be retrieved, how long it takes to retrieve, and the size of the response. If the response content is HTML, information is also provided about the page objects within the URL.

You can monitor URLs that use the HTTP, HTTPS, FTP, and file protocols. You specify the URLs to monitor in the HTTP URLs file, or through Take Action options.

This data source requires a Java runtime environment. The following versions of Java are supported:

- Oracle Corporation Java Version 5 or later
- IBM Corporation Java Version 5 or later

Use the following procedure to create an attribute group to monitor a list of URLs:

Procedure

1. On the Agent Initial Data Source page (Figure 185 on page 262) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

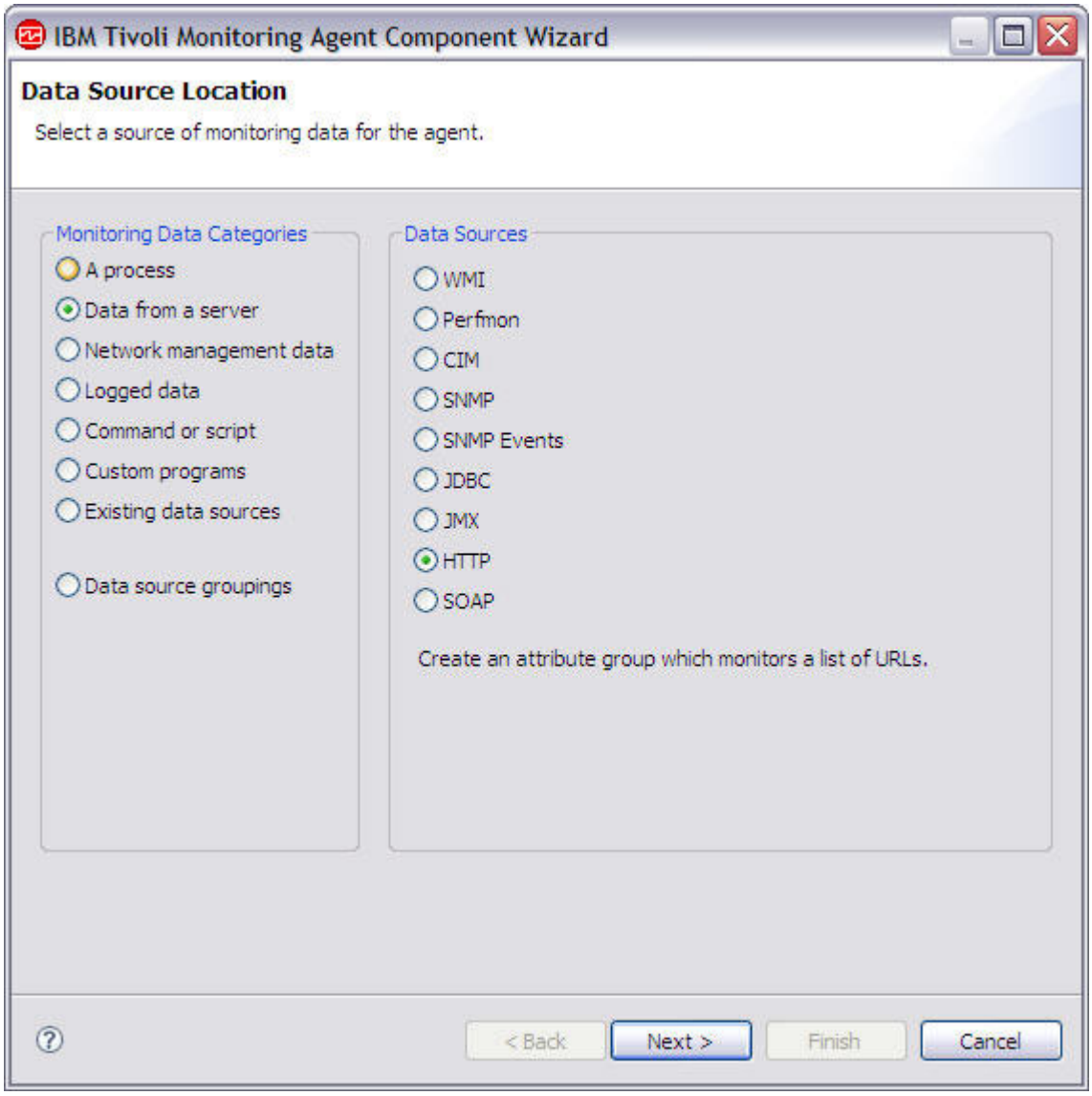


Figure 185. Adding HTTP data

2. In the **Data Sources** area, click **HTTP**.
3. Click **Next**.
4. On the HTTP Information page (Figure 186 on page 263), select one or more operating systems in the **Operating Systems** area.

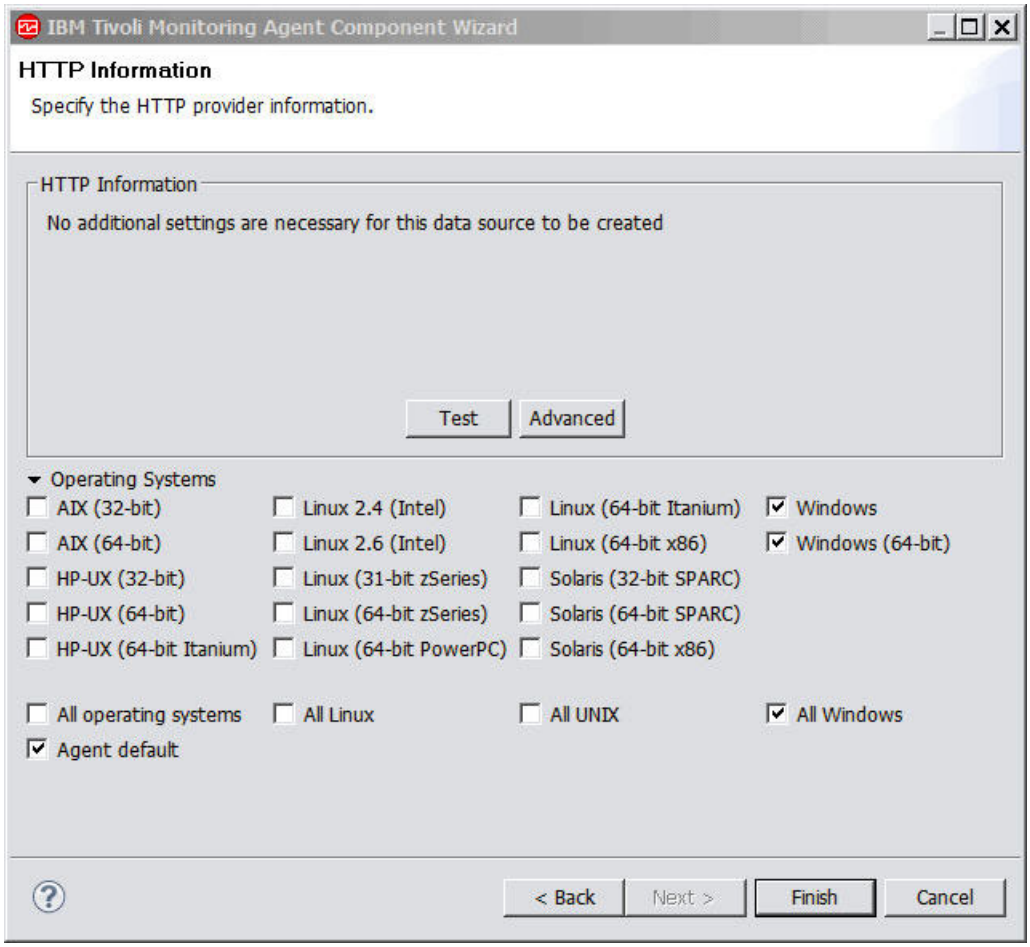


Figure 186. HTTP Information window

5. Optional: Click **Test** to test this attribute group. For more information about testing, see "Testing HTTP attribute groups" on page 270
6. Optional: Click **Advanced** to create a filter to limit the data that is returned by this attribute group. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 67
7. Do one of the following steps:
   a. If you are using the New Agent wizard, click **Next**.
   b. Click **Finish** to save the data source and open the Agent Editor.

Results

The HTTP data source creates two attribute groups: Managed URLs and URL Objects. You can add, modify, or delete attributes.

Related tasks:

"Creating attributes" on page 56
Adding, modifying, and removing attributes.

Related reference:

"HTTP attribute groups" on page 576
The two HTTP attribute groups, `Managed URLs` and `URL Objects`, are used to receive information from URLs and the objects within theses URLs.

HTTP tables

Reference information about the default HTTP attribute groups.

The two attribute groups that are created by the HTTP data source are:

`Managed URLs`
: The Managed URLs table provides availability and response time data about each URL being monitored.

`URL Objects`
: The URL Objects table contains a separate URL entry for each embedded object. For example, the `.gif` and `.jpg` files that might be used in the website that is listed in the Managed URL report.

For information about the syntax that is used in the Managed URLs and URL Objects tables, see ("Specific fields for HTTP attributes").

When you want to monitor the response time and availability of specific objects within a website, review the contents of the URL Objects table. The URL Objects table monitors a specific list of objects that are detected in downloaded HTML files. The following table lists the HTML elements that are searched for objects to monitor and the attributes within these elements that reference the objects:

Table 16. HTML elements searched for objects to monitor

HTML element	Attribute containing object to be monitored
`img`	`src`
`script`	`src`
`embed`	`src`
`object`	`codebase` or `data`
`body`	`background`
`input`	`src`

In the following example HTML extract, the object that is monitored is the image that is referenced by the `src` attribute of the `img`element.

```
<img src="/v8/images/id-w3-sitemark-simple.gif" alt="" width="54" height="33" />
```

A full URL to the image is calculated based on the URL to the source document.

Note: If you do not want to monitor objects that are found in a web page, in the URL Monitoring configuration section, set the **Page object collection** property to **No**.

Specific fields for HTTP attributes

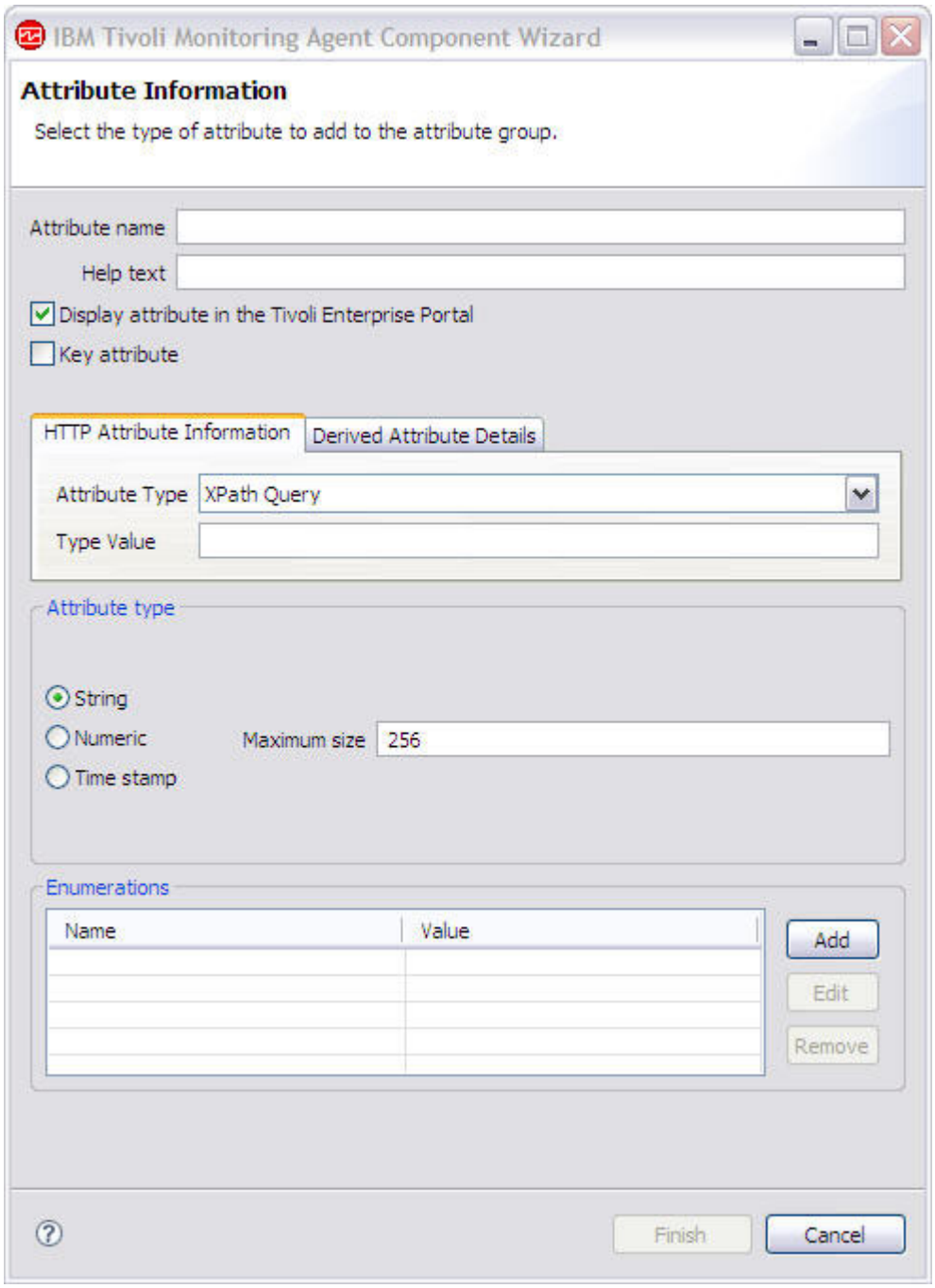


Figure 187. Attribute Information page

In the Attribute information page (Figure 187), there are two fields for HTTP attributes that define how data is collected from the URL. The **Attribute Type** field can be any value from a list that controls the information about the URL that is returned. Some attribute types require a value in the **Type Value** field.

The following table describes all of the attribute types for the Managed URLs attribute group, and the type value when one is required:

Table 17. HTTP Attribute Information - Managed URLs

Attribute type	Description	Type value	Data type that is returned	Differences with FTP and file protocols
XPath Query	Runs an XPath query on the content that is returned from a URL connection. The query must be written to return data useful for an attribute, not a list of nodes.	The XPath query to run against the content that is obtained from a URL connection.	The data that is returned can be a string, a numeric, or a timestamp value. If the data is in the XML DateTime format, you can specify timestamp as the attribute type. The agent converts the value to a Candle Timestamp.	None
Response Time	The amount of time in milliseconds that it took to download the content from the requested URL.	None	Integer (number of milliseconds)	None
Response Message	The HTTP response message that is returned by the server.	None	String	The response message applies only if the URL uses the HTTP or HTTPS protocols.
Response Code	The HTTP response code that is returned by the server.	None	Integer	The response code applies only if the URL uses the HTTP or HTTPS protocols. It is always 0 for file or FTP URLs.
Response Length	The size of the content in bytes that is downloaded from the requested URL	None	Integer (size in bytes)	None
Response Header	The response header can be used to retrieve a value from one of the URL response header fields. The argument specifies which field is requested.	The response header to collect.	String	Generally FTP and file protocols do not have any headers that can be collected.
Request URL	The connection is made to this URL. All of the response keywords provide information about the connection to this URL. The XPath Query can be used to obtain information that is obtained from the content that is returned by accessing this URL.	None	String	None

Table 17. HTTP Attribute Information - Managed URLs (continued)

Attribute type	Description	Type value	Data type that is returned	Differences with FTP and file protocols
Page Objects	The number of objects that are discovered on the monitored HTML page that are monitored by the URL Objects attribute group.	None	Integer	None
Total Object Size	The total size of the objects that is monitored in the URL Objects attribute group for this web page.	None	Integer (in bytes)	None
Alias	The user specified alias for this URL.	None	String	None
User	The user specified data for this URL.	None	String	None

The following table describes the attribute types for the URL Objects attribute group:

Table 18. HTTP Attribute Information - URL Objects

Attribute type	Description	Type value	Data type that is returned	Differences with FTP and file protocols
URL	The URL that is monitored in the Managed URLs table.	None	String	None
Object Name	The URL for the object that is monitored within the HTML page.	None	String	None
Object Size	The size in bytes of the content that is downloaded from the Object Name URL.	None	Numeric	None
Object Response Time	The time in milliseconds it took to download the page object.	None	Numeric	None

Monitoring a URL

You can start monitoring any URL by including it in the URLs file or by using the URL Add Take Action option.

URLs file

The URLs file specified in configuration can be in any directory. If this file does not exist or is empty, then you can start URL monitoring by using Take Actions. For more information, see (“Take Action

option”). If you already have a Tivoli Universal Agent that uses the Tivoli Universal Agent HTTP Data Provider, you can reuse the KUMPURLS file. When you are configuring the agent, point to your KUMPURLS file.

The following table provides examples of how URLs are entered in the URLs file, depending on the method by which they were added.

Table 19. URLs file entries

URLs	Added by
www.bbc.co.uk http://weather.com www.ibm.com	Manually adding entries to the file. If no protocol is specified, as in the www.ibm.com example, http is assumed.
ftp://userid:password@ftpserver/index.html	Manually added by using File Transfer Protocol (FTP)
http://www.ibm.com USER=ibm ALIAS=ibm	Using the URL Add Take Action
file:/tmp/samples.html USER=samples \ ALIAS=samples	Using a URL Add Take Action that uses FTP
http://google.com INTERVAL=60 CACHE=50 \ USER=google ALIAS=search	Example from the Tivoli Universal Agent KUMPURLS file

When you directly edit the URLs file, your changes are implemented when the agent does its next data collection.

Take Action option

You can also specify URLs to monitor through a Take Action option that is called URL Add. When this option is selected, a window is displayed where you can specify the following parameters:

URL A required parameter that represents the URL itself. You can type this parameter with or without the http://or https://prefix.

Alias An optional parameter that you can specify to associate a more meaningful name to a URL. No spaces are allowed in this parameter. If this parameter is not completed, the Alias Name defaults to blank.

User_Data
An optional parameter that you can specify to enter data about the URL. If this parameter is not completed, the User_Datadefaults to INITCNFG.

After you complete the information and close the window, assign the HTTP URL Add action to the destination managed system that is associated with the agent. Monitoring begins immediately for the new URL. The URL is also added to the URLs file so that it continues to be monitored across agent restarts.

A corresponding Take Action option is named HTTP URL Remove. Use the HTTP URL Remove action to immediately stop monitoring for a particular URL. The removed URL is also deleted from the URLs file. The URL Remove window requests only the URL and User_Data values. The URL and User_Data values must match the values that are seen in the Tivoli Enterprise Portal or the Remove action fails. For example, if you omitted the http:// from the URL field of the Add action, you must include it in the URL field of the Remove action. If you did not specify User_Data, you must specify INITCNFG as seen in the Tivoli Enterprise Portal.

If a URL is added manually to the URLs file, you can delete it with the Take Action. If you delete with the Take Action, you must specify the values as seen in the Tivoli Enterprise Portal. For example, if you

added www.ibm.com to your URLs file, the Tivoli Enterprise Portal displays http://www.ibm.com as the URL and INITCNFG as the User_Data. To remove the URL with the Take Action, you must use the values that are seen in the Tivoli Enterprise Portal.

After you complete the information and close the window, assign the URL Remove action to the destination managed system that is associated with the agent.

Monitor https:// URLs

The HTTP data source can monitor only secure https:// URLs that do not require scripted access or interactive prompting.

If the https:// URL can be retrieved with a standard HTTP Get call, then it can be monitored.

Proxy server

If the system where the agent is running requires a proxy to access the SOAP data provider, you must specify proxy server configuration properties.

For more information, see ("Proxy Server configuration").

HTTP configuration

Reference information about HTTP configuration.

After an HTTP data source is added, the configuration is displayed on the Runtime Configuration page of the Agent Editor. Configuration sections are added for URL Monitoring, for Proxy Server authentication, and for Java.

URL Monitoring configuration

The URL Monitoring configuration section contains the following properties:

Table 20. URL Monitoring configuration properties

Name	Valid values	Required	Description
HTTP URLs file	Path to a file	Yes	The path to the file that contains a list of URLs.
Page Object Collection	Yes, No The default value is Yes.	No	Whether to download objects that are found in a web page and collect data from them.

Proxy Server configuration

The Proxy Server configuration section contains the following properties:

Table 21. Proxy Server configuration properties

Name	Valid values	Required	Description
Proxy Hostname	String	No	The proxy host name to be used for HTTP connections.
Proxy User Name	String	No	The user name for the proxy server.

Table 21. Proxy Server configuration properties (continued)

Name	Valid values	Required	Description
Proxy Port	Positive integer The default value is 80.	No	The HTTP port number of the proxy server.
Proxy Password	Password	No	The password for the proxy server.

Note: If the **Proxy Hostname** property is blank, no proxy is used.

Java configuration

If you define an HTTP data source in your agent, the agent must use Java to connect to the HTTP server. Java configuration properties are added to the agent automatically. The following Java configuration properties are specific to the agent runtime configuration. The Agent Builder does not require the Java properties because it uses its own JVM and logging, which are configured through the JLog plug-in):

Table 22. Java configuration properties

Name	Valid values	Required	Description
Java Home	Fully qualified path to a directory	No	A fully qualified path that points to the Java installation directory.
Trace Level	Choice (The default value is Error)	Yes	Use this property to specify the trace level that is used by the Java providers. For more information, see ("Trace log format" on page 426).
JVM Arguments	String	No	Use this property to specify an optional list of arguments to the Java virtual machine.

Testing HTTP attribute groups

You can test the HTTP attribute group that you created, within Agent Builder.

Procedure

1. Start the Testing procedure in the following ways:
   - During agent creation click **Test** on the HTTP Information page.
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35

   After you click **Test** in one of the previous two steps, the HTTP Test window Figure 188 on page 271 is displayed.

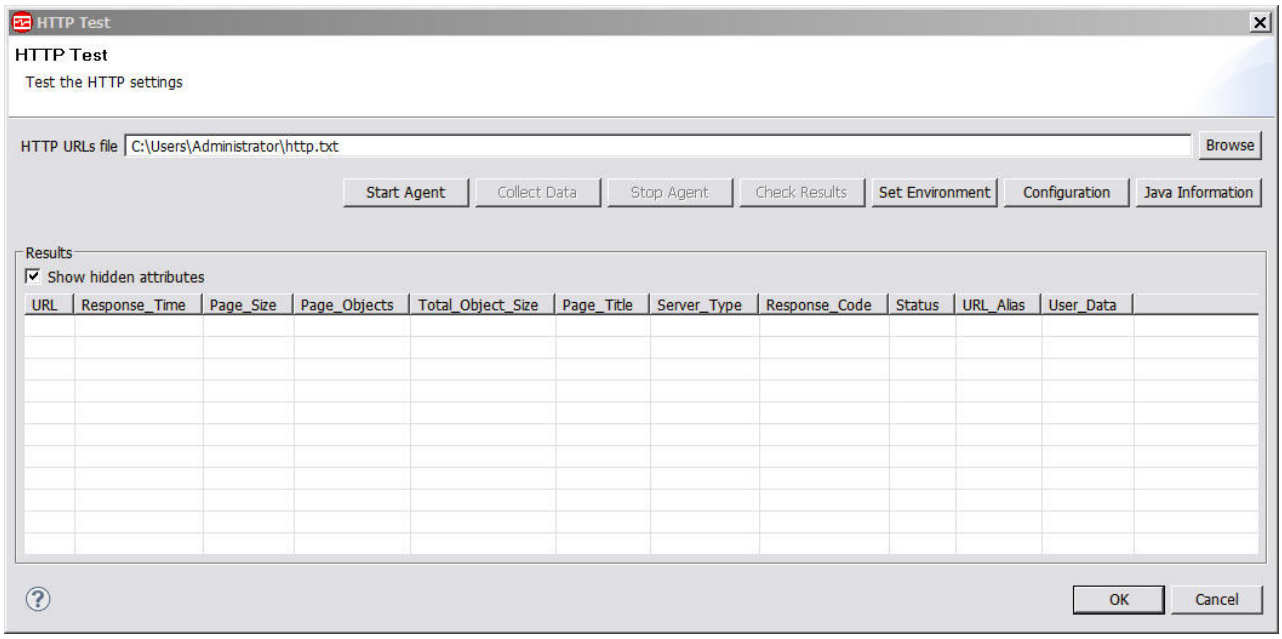


Figure 188. HTTP Test window

2. Click **Browse** to select the HTTP URLs file. For more information about URLs files, see ("URLs file" on page 267).
3. Optional: Set environment variables, configuration properties, and Java information before you start testing. For more information, see ("Attribute group testing" on page 377). For more about HTTP configuration, see ("HTTP configuration" on page 269).
4. Click **Start Agent**. A window indicates that the Agent is starting.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent monitors the URLs defined in the HTTP URLs file. The HTTP Test window displays any data that is returned.
6. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 545
7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the HTTP Test window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, "Testing and debugging your agent," on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 24. Monitoring data from a SOAP data source.

Configure your agent to receive data from a SOAP data source.

About this task

By using the SOAP data source, you can specify an HTTP URL and send a GET, POST, or PUT request. For POST or PUT requests, you can specify the associated POST data. An XML response is retrieved and parsed, and the data is exposed to Tivoli Monitoring in attributes. You can define the attributes as all of the values within a particular element. Or you can define custom `XPaths` to specify how to populate individual attributes. You can also combine the two mechanisms.

Use the following procedure to collect XML responses from a URL:

Procedure

1. On the Agent Initial Data Source page (Figure 189 on page 274) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

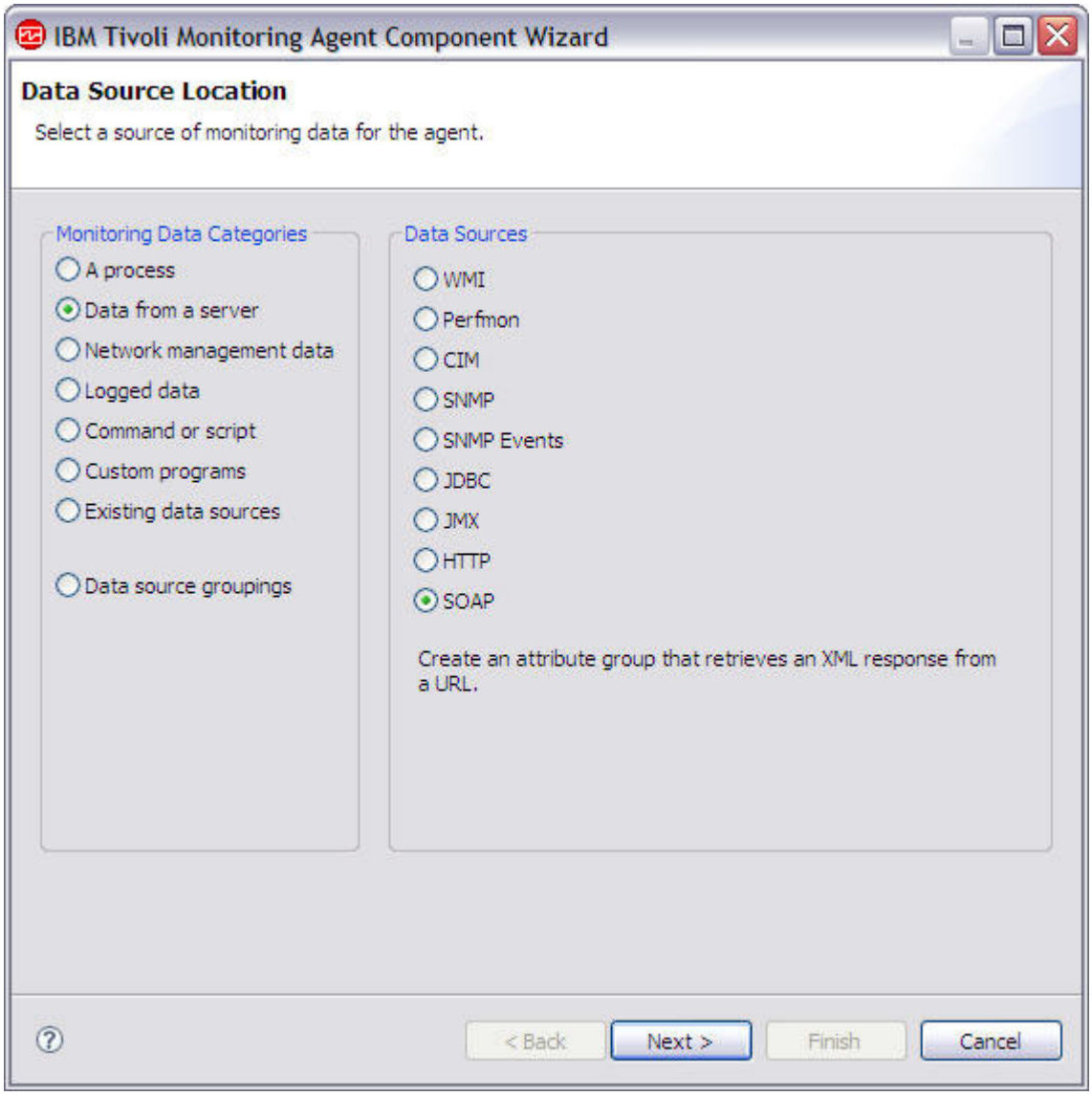


Figure 189. Adding HTTP data

2. In the **Data Sources** area, click **SOAP**.
3. Click **Next**.
4. On the SOAP Information page (Figure 190 on page 275), enter a URL. The default value is:
   `http://${KQZ_HTTP_SERVER_NAME}:${KQZ_HTTP_PORT_NUMBER}`

   Note: You can use a configuration variable or multiple configuration variables that resolve to a URL. Click **Insert Configuration Property** to select a property to insert. For more information, see (Chapter 30, “Customizing agent configuration,” on page 361).

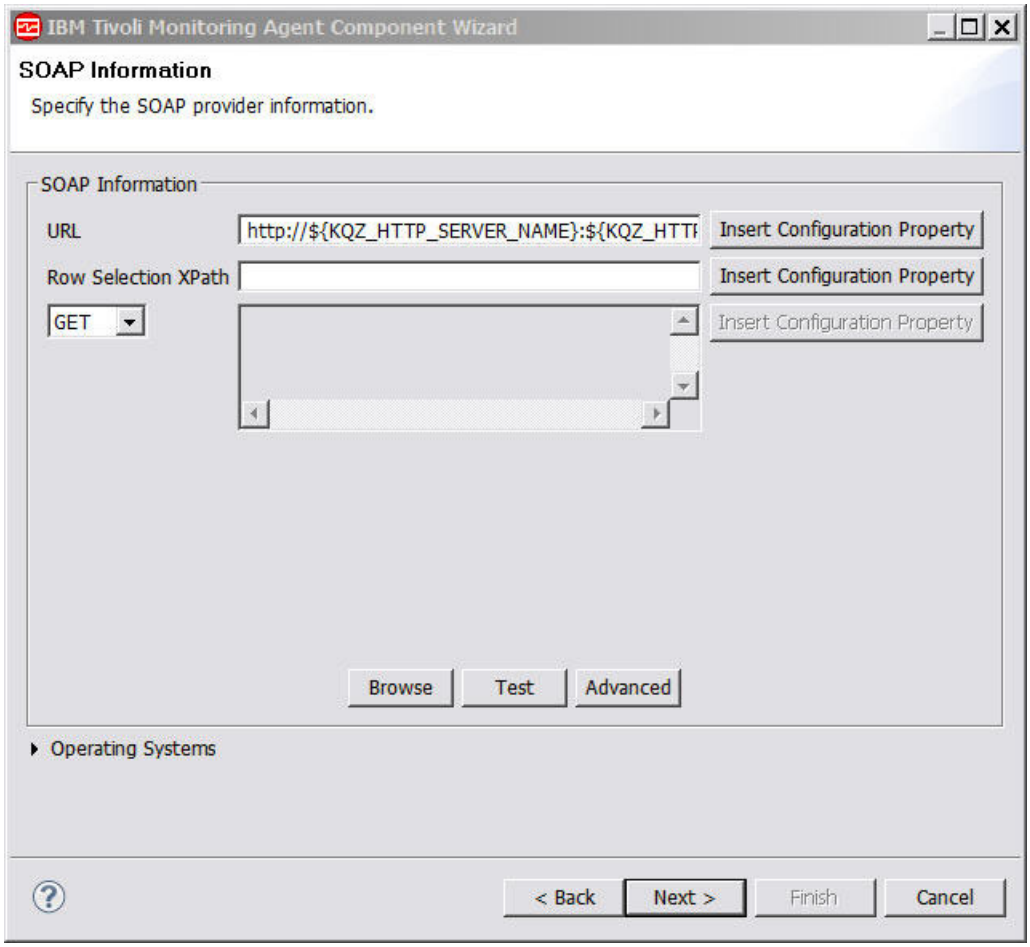


Figure 190. SOAP Information page

5. Select a request type. The default request type is Get. For Post and Put requests, enter the data to be processed.

   Note: For Post and Put requests, the **Insert Configuration Property** is enabled. Click **Insert Configuration Property** to include a configuration variable in the data to be processed. For more information, see (Chapter 30, "Customizing agent configuration," on page 361).

6. Click **Browse**

   Note: If after you enter a URL and select a request type, you do not want to use the SOAP browser to build the definition, enter a **Row Selection XPath**. You enter the **Row Selection XPath** in the SOAP Information window. Next, define all of the attributes for the attribute group.

7. In the SOAP Browser window, do the following steps:
   a. Enter a URL and select a request type if you did not already do so.
   b. Click **Configuration** to set any configuration properties that are referenced in the URL or other fields.
   c. Click **Connect** to obtain data from the SOAP provider.

      When you connect to the URL, a list of XML elements for this URL is shown in a Document Object Model (DOM) tree. In the WebSphere Application Server example in (Figure 191 on page 276), the following URL was entered:

      ```
      http://nc053011.tivlab.raleigh.ibm.com:9080/wasPerfTool/servlet
      /perfservlet?module= \threadPoolModule
      ```

The `PerformanceMonitor` XML element is shown. This element is the top-level XML element in the XML document that is returned by the SOAP provider.

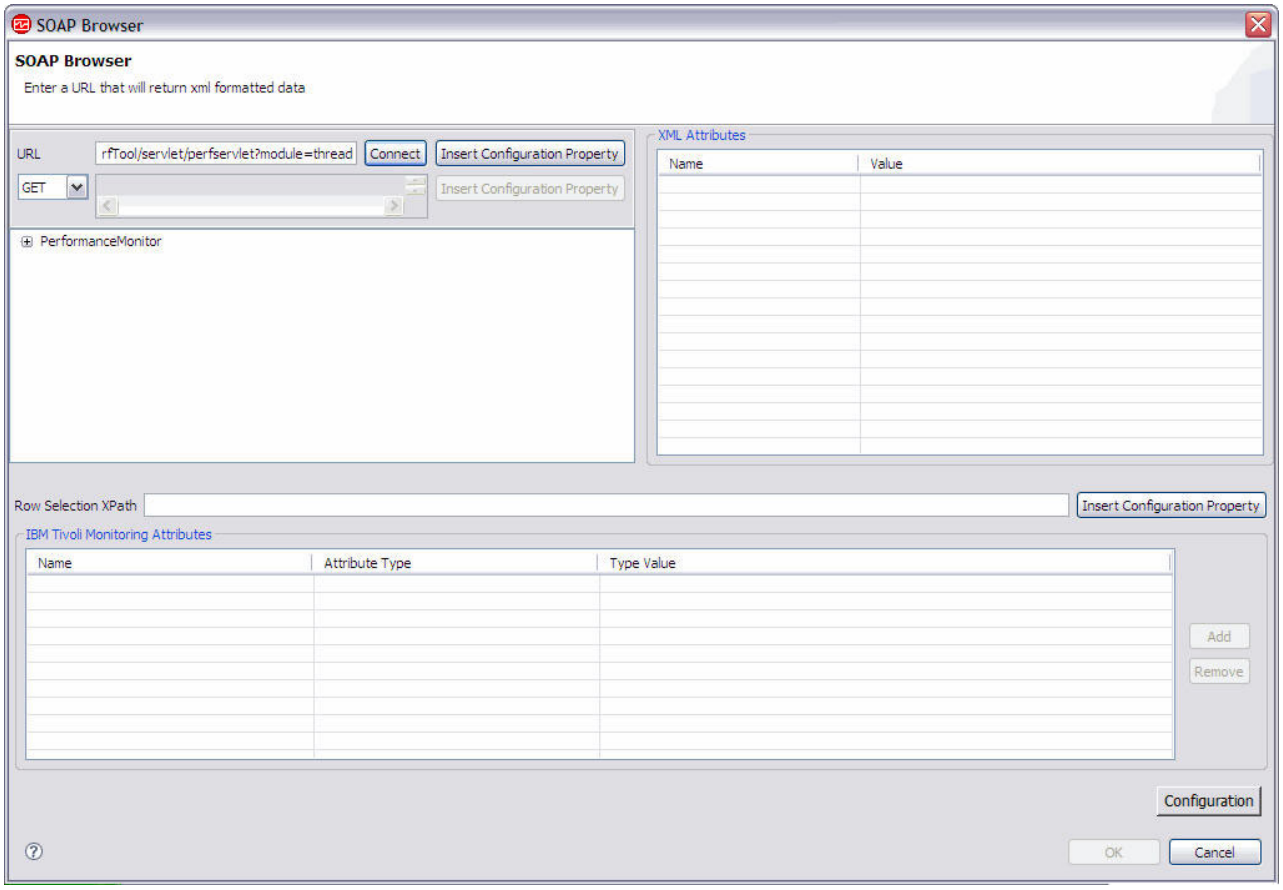


Figure 191. SOAP Browser window

d. In the DOM tree, find and select the XML node that you want to set as the **Row Selection XPath**. In the WebSphere Application Server example in (Figure 192 on page 277), the `PerformanceMonitor/Node/Server/Stat/Stat/Stat` node is selected. This node represents a row of data in the attribute group. When you select a node in the DOM tree and click **Add**, you get all of the attributes and elements defined on that node of the tree. (You click **Add** in the **IBM Tivoli Monitoring Attributes** area).

   When a node is selected, the **XML Attributes** area shows any XML attributes defined for the selected node. Select an XML attribute and click **Add** to include this attribute in the list of IBM Tivoli Monitoring Attributes.

   Note: If more than one row of data is expected, the XPath must map to a node set. Where the Row Selection XPath returns a node that is set with only one item, the attribute group contains only one row.

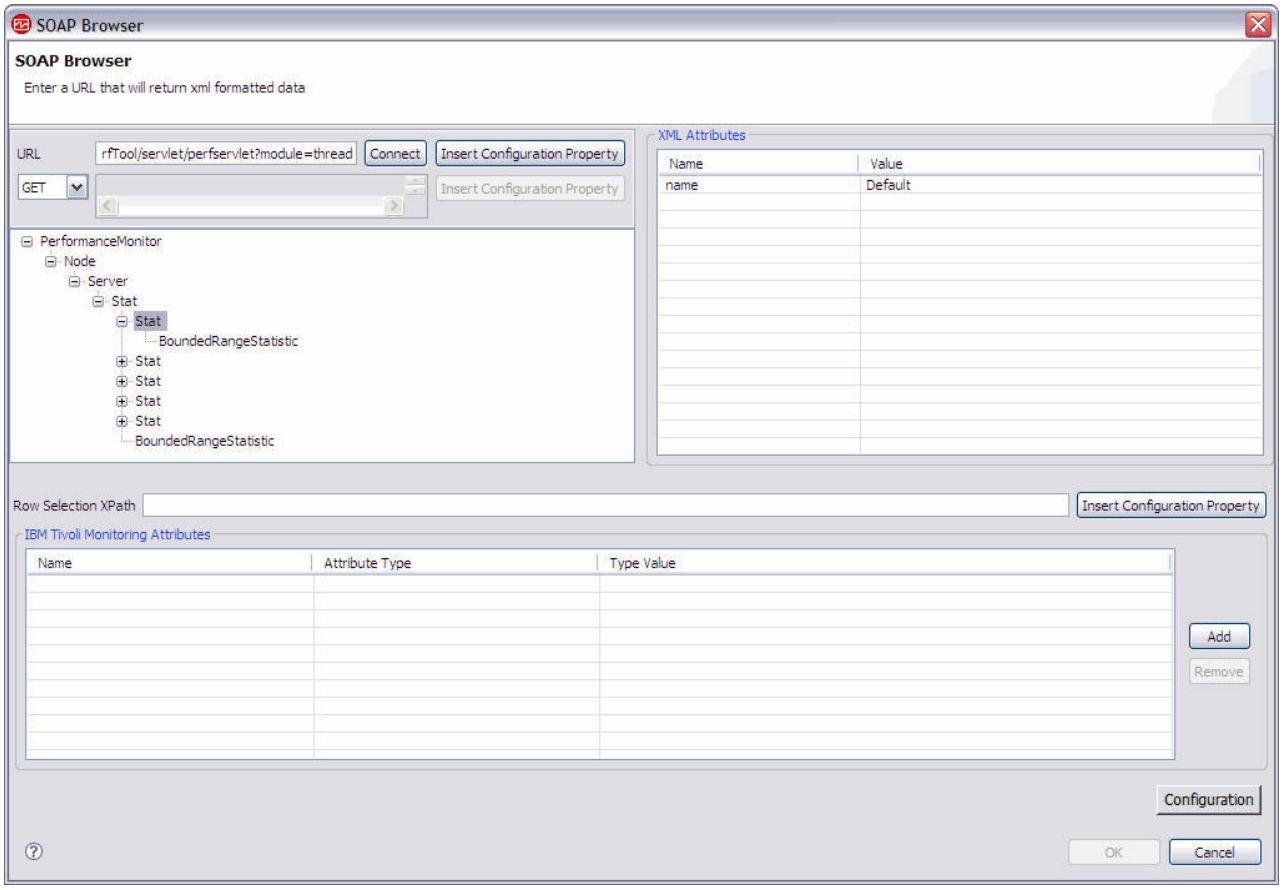


Figure 192. SOAP Browser window

e. Click **Add** in the IBM Tivoli Monitoring Attributes area. The list of Tivoli Monitoring attributes are shown and the **Row Selection XPath** field is filled.

   The XPath for each Tivoli Monitoring attribute is used to map XML nodes or elements to Tivoli Monitoring attributes. In the WebSphere Application Server example in theFigure 193 on page 278, the first attribute in the list of Tivoli Monitoring attributes, `Stat`, is not of use and would be removed.

   You can edit the name and XPath for a Tivoli Monitoring attribute in the **Type Value** field. For more information about using XPaths, see "XPath options" on page 281

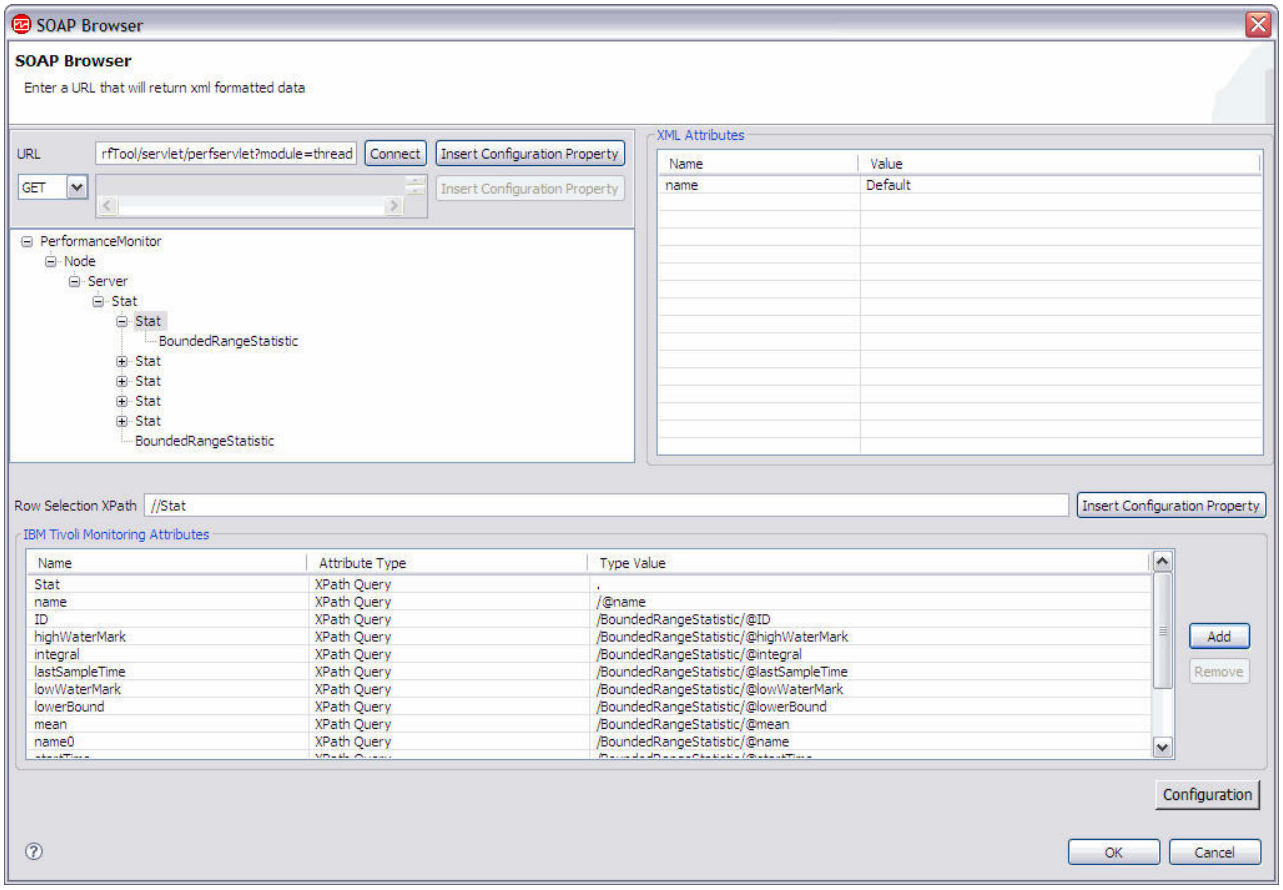


Figure 193. SOAP Browser window

f. In the SOAP Browser window, click **OK** to save your changes and return to the SOAP Information window.

8. In the SOAP Information window, click **Next**.
9. If you did not use **Browse** earlier and you entered the **URL** and **Row Selection XPath** in the SOAP Information window, the Attribute Information page is shown. Specify the information for the first attribute on the Attribute Information page, and click **Finish**. You can then specify more attributes by using the Agent Editor. For more information about creating attributes, see (“Creating attributes” on page 56).
10. If you used the Browse function in step (6 on page 275), the Select key attributes page is shown. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see (“Selecting key attributes” on page 26).
11. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see “Testing SOAP attribute groups” on page 282
12. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see “Filtering attribute groups” on page 67
13. Do one of the following steps:
    a. If you are using the New Agent wizard, click **Next**.
    b. Click **Finish** to save the data source and open the Agent Editor.

Specific fields for SOAP attributes

In the Attribute Information window, there are two fields for SOAP attributes that define how data is collected from the SOAP response.

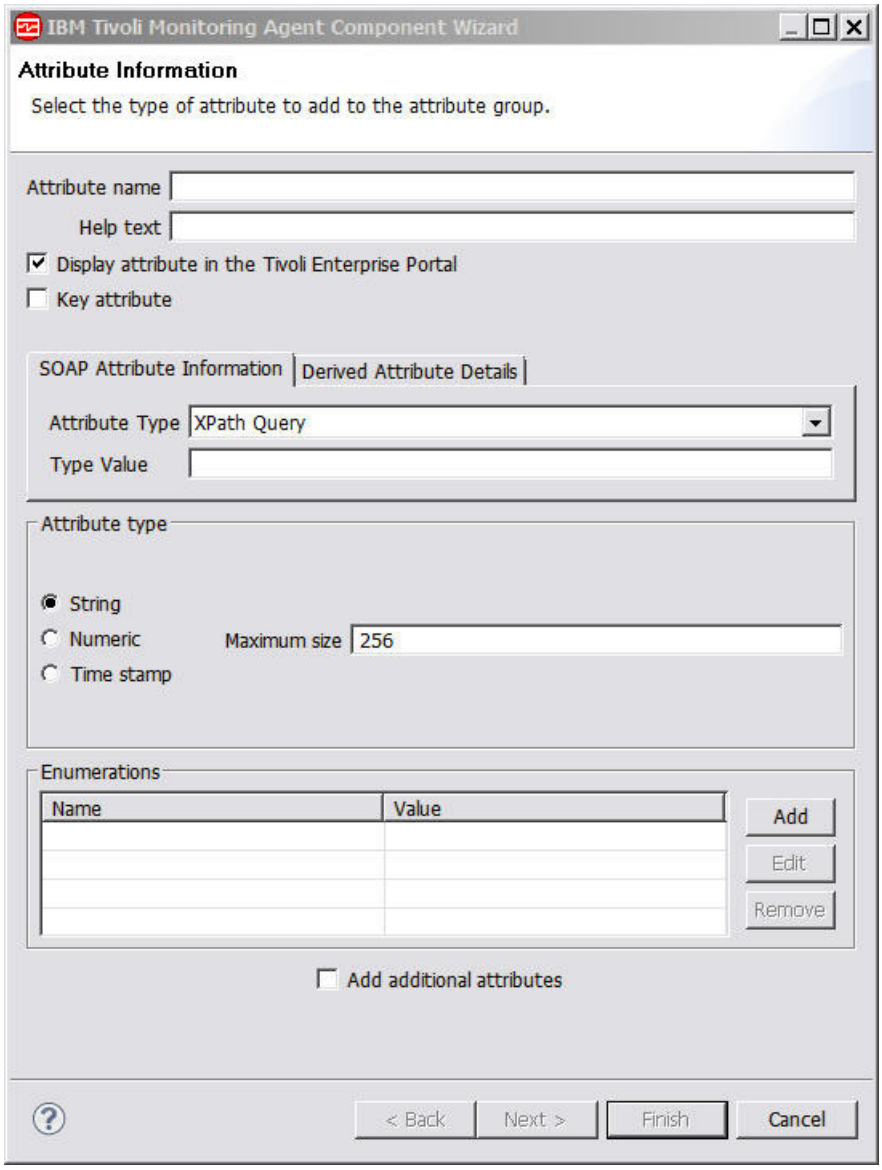


Figure 194. Attribute Information page

The **Attribute Type** field can be any value from a list that controls the information about the response that is returned. Some attribute types require a value in the **Type Value** field. The default attribute type is XPath Query, which runs an XPath query against the SOAP server response content. The type value is the XPath query that is run. The following table describes all of the attribute types and the type value when one is required:

Table 23. SOAP Attribute Information

Attribute type	Description	Type value	Returned data type	Differences with FTP and file protocols
XPath Query	Runs an XPath query on the content that is returned from a URL connection. The query must be written to return data useful for an attribute, not a list of nodes.	The XPath query to run against the content that is obtained from a URL connection. If a row selection query was defined, this XPath query must be relative to the row selection query.	The data that is returned can be a string, a numeric, or a timestamp value. The Agent Builder browser for SOAP generally detects the correct data type for the attribute from the data that is being browsed. If the data is in XML DateTime format, you can specify timestamp as the attribute type and the agent converts the value to a Candle Timestamp.	None
Response Time	The amount of time in milliseconds that it took to download the content from the requested URL.	None	Integer (number of milliseconds)	None
Response Message	The HTTP response message that is returned by the server.	None	String	The response message applies only if the URL uses the HTTP or HTTPS protocols.
Response Code	The HTTP response code that is returned by the server.	None	Integer	The response code applies only if the URL uses the HTTP or HTTPS protocols. It is always 0 for file or FTP URLs.
Response Length	The size of the content in bytes that was downloaded from the requested URL	None	Integer (size in bytes)	None
Response Header	The response header can be used to retrieve a value from one of the URL response header fields. The argument specifies which field is requested.	The response header field to collect.	String	Generally FTP and file protocols do not have any headers that can be collected.

Table 23. SOAP Attribute Information (continued)

Attribute type	Description	Type value	Returned data type	Differences with FTP and file protocols
Request URL	The connection was made to this URL. All of the response keywords provide information about the connection to this URL. The XPath Query can be used to obtain information that is obtained from the content that is returned by accessing this URL.	None	String	None

XPath options

Using XML Path Language, you can select nodes from an XML document. A few of the possible uses of XPaths for the SOAP data sources include:

- Using predicates in the XPath to identify the XML elements that correspond to rows of data in the IBM Tivoli Monitoring attribute group. You can use predicates in the XPath that maps XML elements or attributes to Tivoli Monitoring attributes, as in the following example:

  ```
  Stat[@name="URLs"]/CountStatistic[@name="URIRequestCount"]/@count
  ```

  Where there are multiple location steps in the XPath, each location step can contain one or more predicates. The predicates can be complex and contain boolean values or formula operators. For example:

  ```
  //PerformanceMonitor/Node/Server[@name="server1"]/Stat/Stat/Stat[@name=
  "Servlets"]/Stat
  ```

- Including node set functions in the XPath, if a row contains multiple XML elements of the same type. And if the position of an XML element in the node list determines the Tivoli Monitoring attribute the element maps to. Examples of node set functions are, `position()`, `first()`, `last()`, and `count()`.
- Doing simple data transformation, such as substring. If you specify the following substring:

  ```
  substring(myXMLElement,1,3)
  ```

  the XPath returns the first three characters of the XML element, myXMLElement.

You can specify elements outside the context of the Row Selection XPath by using two periods, `(..`, as in the following example:

```
/../OrganizationDescription/OrganizationIdentifier
```

SOAP configuration

After a SOAP data source is added, the configuration is displayed on the Runtime Configuration page of the Agent Editor.

Configuration sections are added for HTTP Server, for Proxy Server, and for Java. For information about Proxy server configuration, see (“Proxy Server configuration” on page 269). For information about Java configuration, see (“Java configuration” on page 270).

HTTP Server

The HTTP Server configuration section contains the following properties:

Table 24. HTTP Server configuration properties

Name	Valid values	Required	Description
HTTP user name	String	No	The HTTP user.
HTTP password	Password	No	The HTTP server password.
HTTP server name	String (The default value is localhost)	No	The host or IP address of the HTTP server
HTTP port number	Numeric (The default value is 80)	No	The host or IP address of the HTTP server
Certificate validation enabled	True, False (The default value is True)	Yes	Disabling certificate validation is potentially insecure
HTTP trust store file	Path to a file	No	The HTTP trust store file
HTTP trust store password	The HTTP trust store password	No	The HTTP trust store password

Proxy server

If the system where the agent is running requires a proxy to access the SOAP data provider, you must specify proxy server configuration properties. For more information, see (“Proxy Server configuration” on page 269).

Testing SOAP attribute groups

You can test the SOAP attribute group that you created, within Agent Builder

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the SOAP Information page (Figure 190 on page 275).
   - After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test**. For more information about the Agent Editor, see Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35

   After you click **Test** in one of the previous two steps, the Test SOAP Collection window Figure 195 on page 283 is displayed

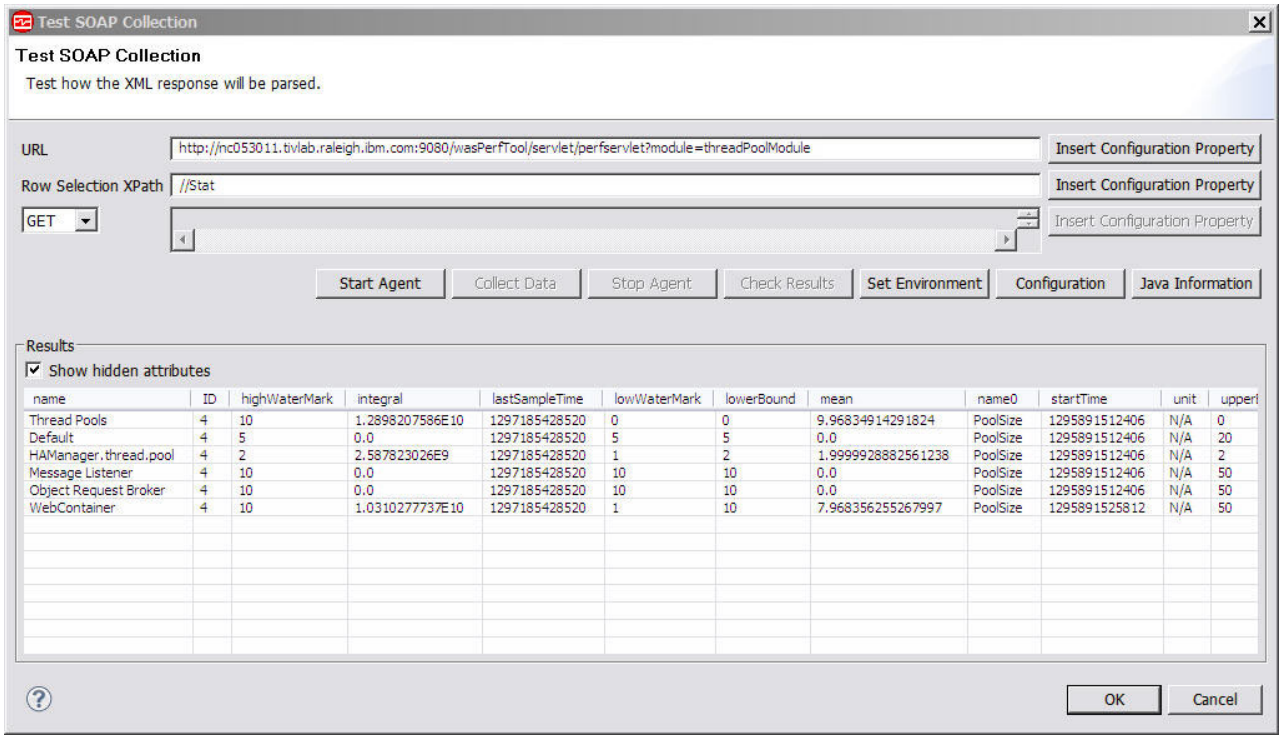


Figure 195. Test SOAP Collection window

2. Optional: Before you start testing, you can set environment variables, configuration properties, and Java information. For more information, see ("Attribute group testing" on page 377). For more about SOAP configuration, see ("SOAP configuration" on page 281).
3. Change the URL, Row Selection XPath, and request type.
4. Click **Start Agent**. A window indicates that the Agent is starting.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. This action populates the Results table and you can preview how the data is parsed and shown in columns in the Tivoli Enterprise Portal.

   In the Results area, you can change the attribute definitions and reload the data to see how your changes affect the attribute group. You can right-click in a column results area to display options to edit the attribute. The attribute edit options are:

   - **Edit Attribute**
   - **Hide Attribute**
   - **Insert Attribute Before**
   - **Insert Attribute After**
   - **Remove**
   - **Remove Subsequent Attributes**
   - **Remove All**
6. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and shown by the Data Collection Status window is described in "Performance Object Status node" on page 545.
7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the Test SOAP Collection window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, “Testing and debugging your agent,” on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 25. Monitoring data by using a socket

Collect data from an external application by using a TCP socket.

About this task

Use the socket data source to provide data to the agent from an external application, running on the same system as the agent. The external application can send data to the agent anytime it wants to. For example, you can develop a command-line interface that allows a user to post data to an attribute group when it is run. Another option is to modify a monitored application to send updates to the agent. The agent does not start or stop the application that is sending data to the socket; this action is controlled by the user.

There are some limitations with the socket data source:

- By default only connections to the local host (127.0.0.1) are possible. For more information about configuring your agent to accept connections from a remote host, see "Remote socket port connection" on page 296.
- There is no mechanism in the socket API for the client to determine what subnodes are available. The client can send data for a specific subnode, but it must already know the subnode name.

Use the following procedure to create an attribute group to collect data by using a Transmission Control Protocol socket (TCP) socket.

Procedure

1. On the Agent Initial Data Source page (Figure 196 on page 286) or the Data Source Location page, click **Custom programs** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

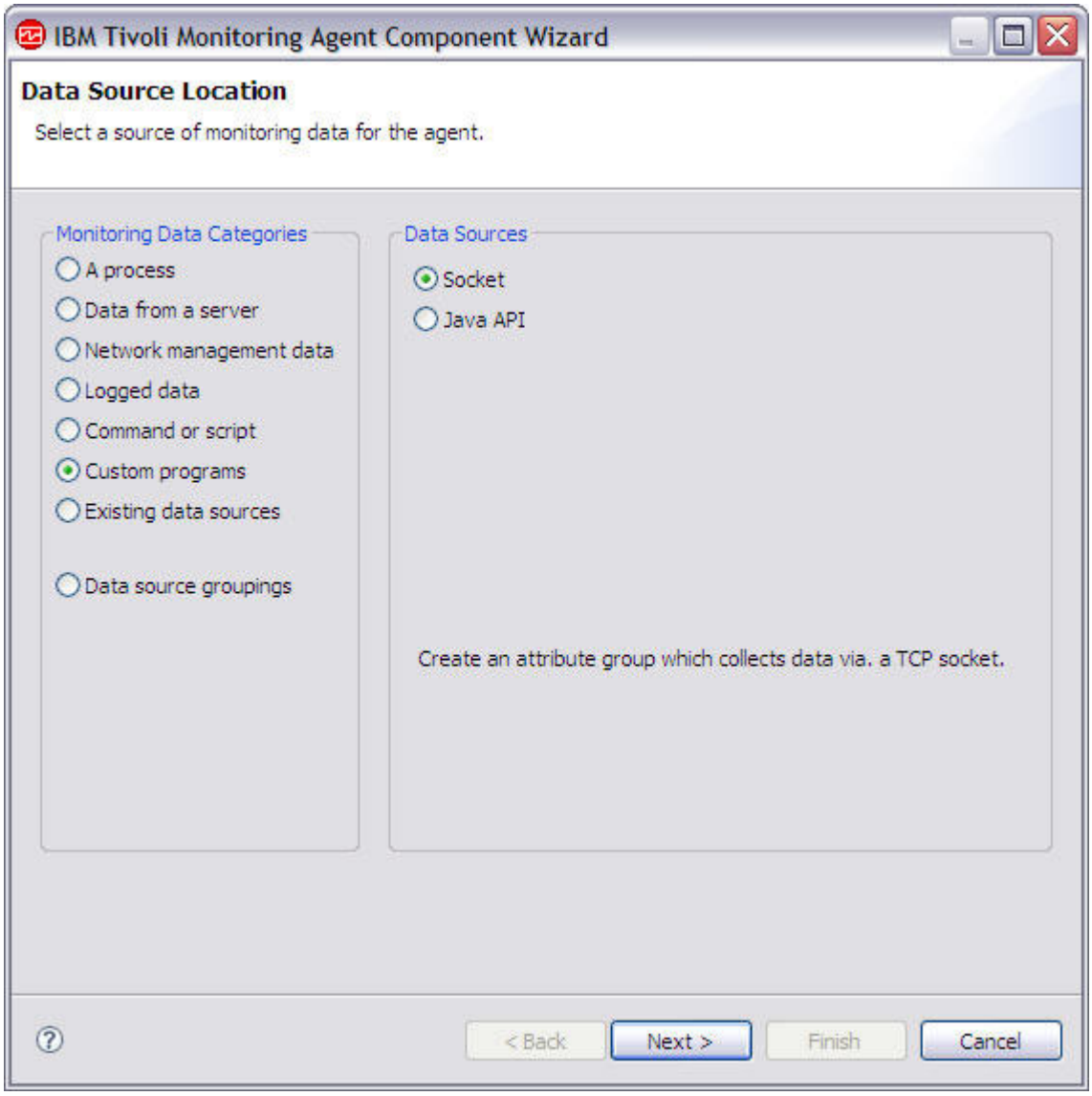


Figure 196. Monitoring data with Socket

2. In the **Data Sources** area, click **Socket**.
3. Click **Next**.
4. On the Socket Information page, enter an Attribute group name.

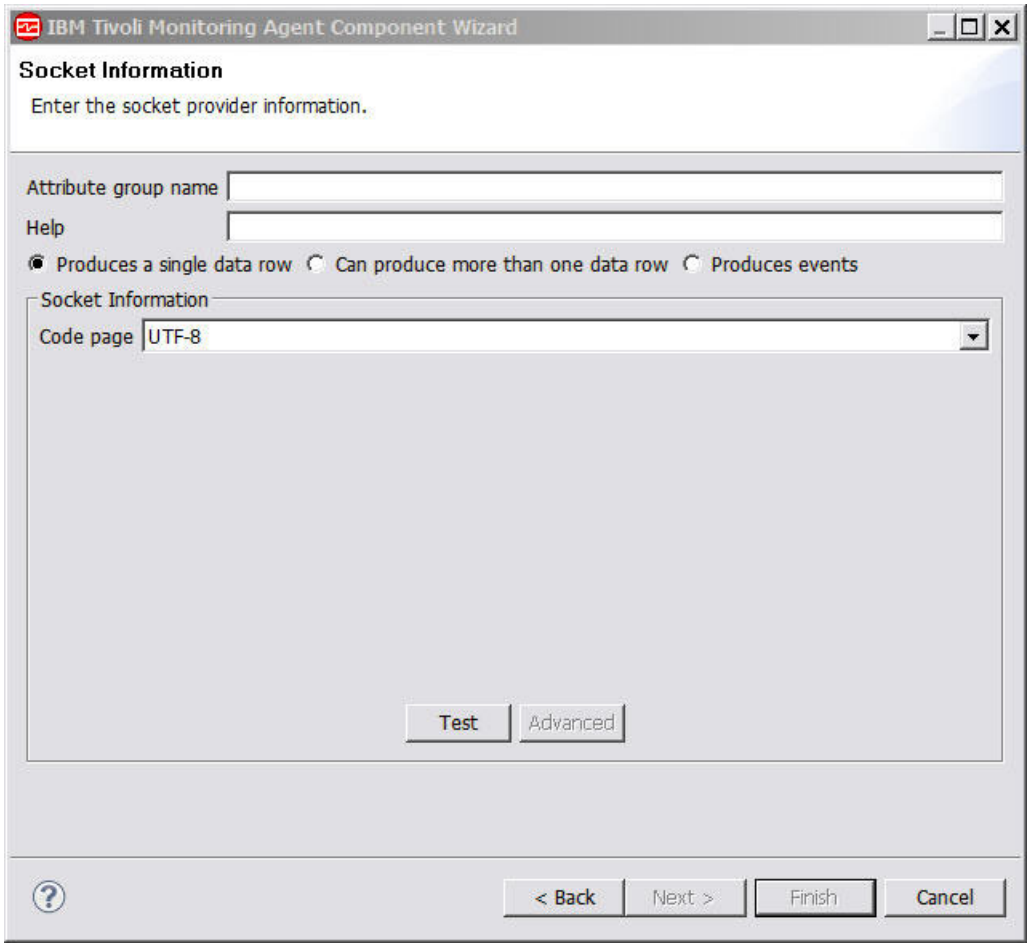


Figure 197. Socket Information window

5. Enter a help text for the attribute group.
6. Select whether the attribute group **Produces a single data row**, **Can produce more than one data row**, or **Produces events**. For more information, see (“Sending data” on page 291).
7. In the Socket Information section, select a **Code page**. For more information, see (“Character sets” on page 294).
8. Optional: Click **Advanced** to modify the advanced properties for the attribute group. The **Advanced** option is active when you select that the attribute group **Can produce more than one data row**, or **Produces events**.
9. Click **Next**.
10. On the Attribute Information page, specify the first attribute for the attribute group. For more information about creating attributes, see (“Creating attributes” on page 56).
11. Click **Next**.
12. Optional: On the Global Socket Data Source Information page, **Error Codes** section (Figure 198 on page 288), you can define the error codes that the socket client can send when it cannot collect data. For more information, see (“Sending errors instead of data” on page 292). To define an error code, use the following steps:

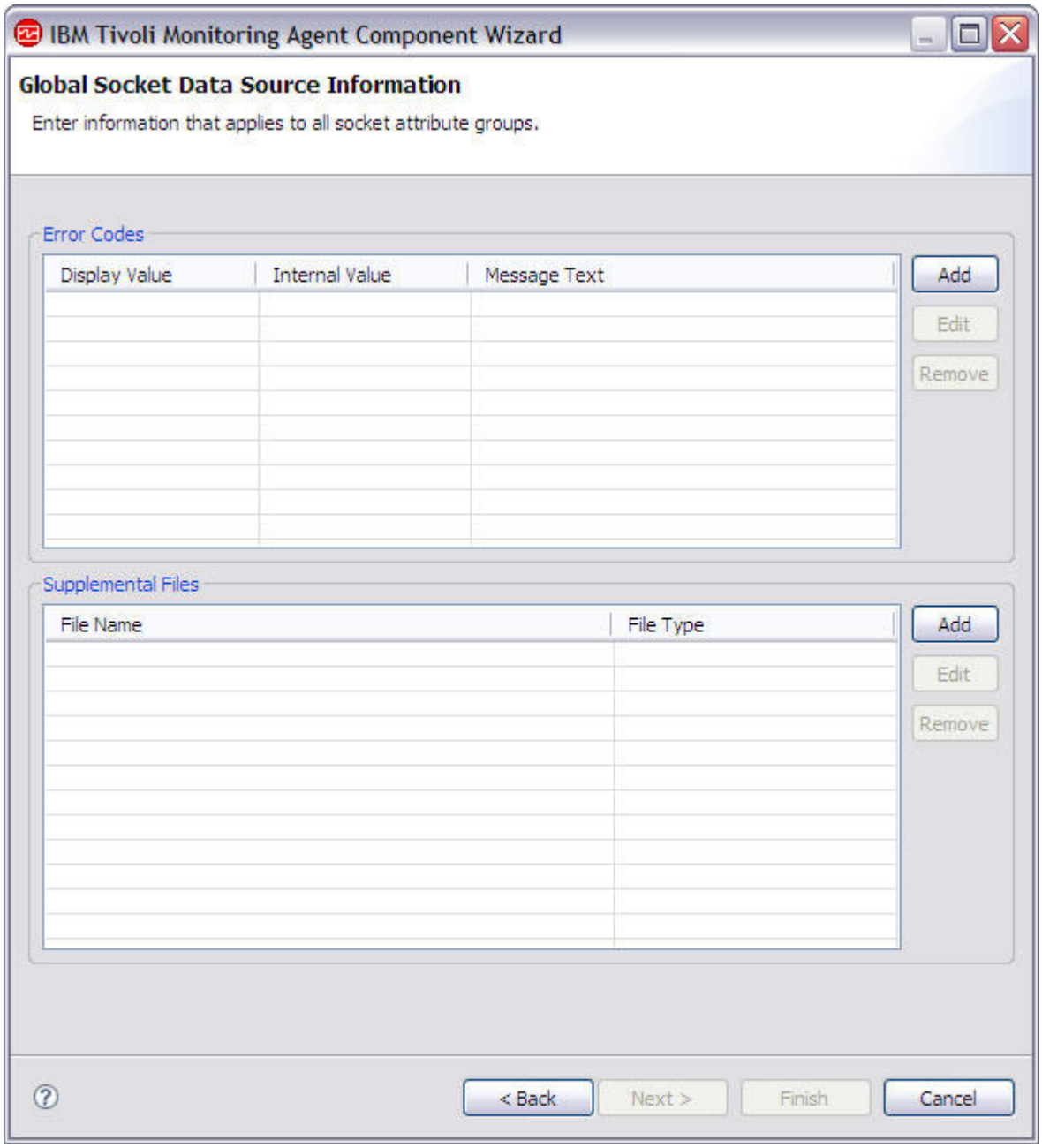


Figure 198. Global Socket Data Source Information page

a. In the Error Codes section, click **Add**. An error code has a limit of 256 characters. Only ASCII letters, digits, and underscores are allowed. No spaces are allowed.
b. In the Socket Error Code Definition window, enter a display value that is shown in the Performance Object Status attribute group.

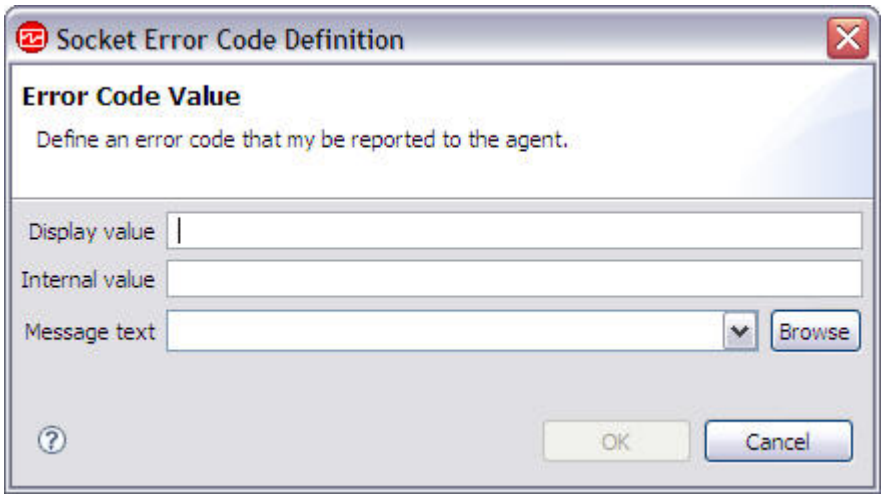


Figure 199. Socket Error Code Definition window,

c. Enter an internal value. The internal value must be an integer from 1,000 to 2,147,483,647.

d. You must define a message text for each error. You can use message text that was entered previously by selecting it from the list. Click **OK** to return to the Global Socket Data Source Information page. The message text is used in the agent log file. If no suitable message text is available, click **Browse** to set up the message text. The Messages (list) window opens (Figure 200). The message window lists messages that are defined in the agent. Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you defined.

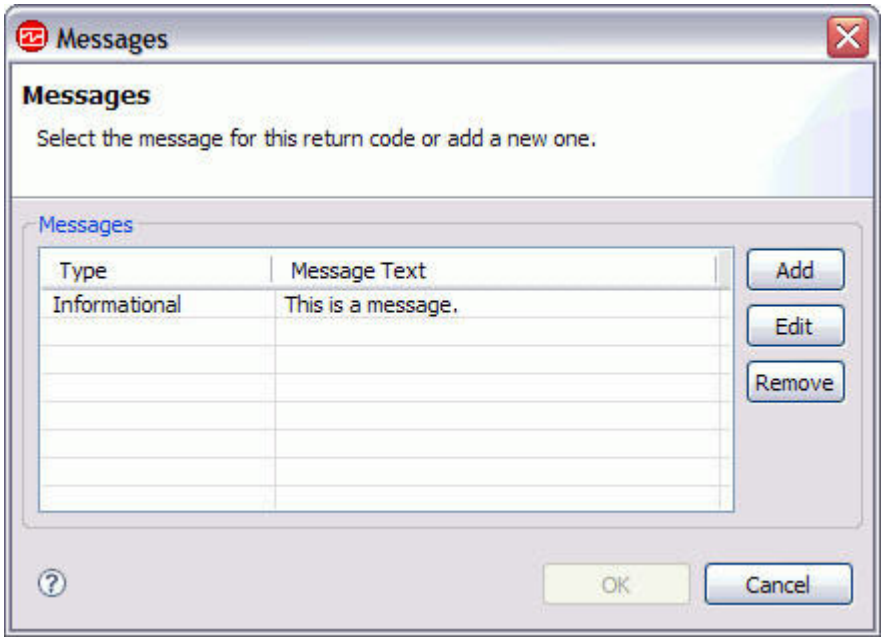


Figure 200. Messages window

e. In the Messages (list) window, click **Add** to see a Message Definition window (Figure 201 on page 290). In the Message Definition window type, the text that describes the meaning of the new message and select the message type.

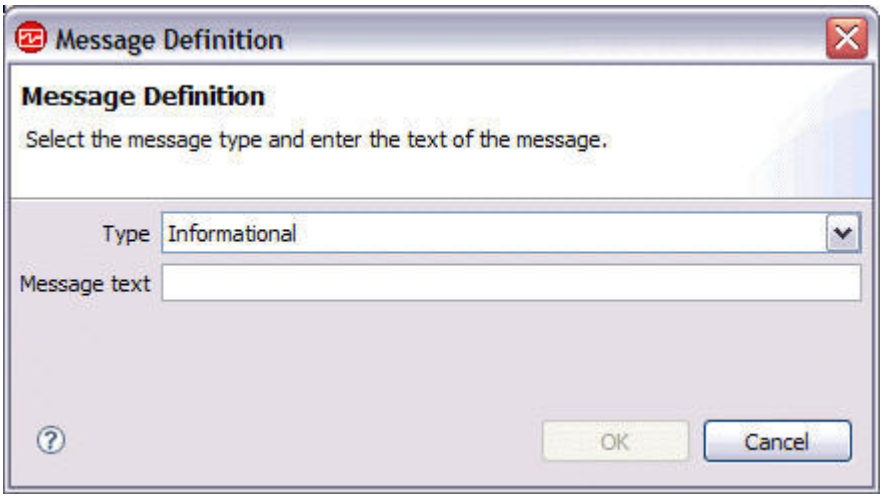


Figure 201. Message Definition window

Note: The message identifier is automatically generated for you.

f. Click **OK**.

g. The Messages (list) window opens, with the new message. To verify the message and return to the Global Socket Data Source Information page, click **OK**.

13. Optional: In the **Supplemental Files** section of the Global Socket Data Source Information page, you can add files that are packaged with the agent. These files are copied to the agent system when the agent is installed. The **File Type** column describes how each file is expected to be used. Three possible uses are described in the following table:

Table 25. File types for supplemental files

File Type	Description
Executable	Select this option if you want to include an executable file with the agent. The agent does not use these files.
Library	Select this option if you to include a library with the agent. The agent does not use these files.
Java resource	Select this option to include Java resources with the agent. The agent does not use these files.

For information about where the Supplemental Files are installed with your agent, see (“New files on your system” on page 392).

Click **Edit** to edit the imported file. For more information, see (“Editing a command file definition” on page 224).

14. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see “Testing socket attribute groups” on page 298

15. Optional: If the data source is sampled, you can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. The data source is sampled when you did not select "Produces events" on the Socket Information page. For more information about filtering data from an attribute group, see “Filtering attribute groups” on page 67

16. Do one of the following steps:

a. If you are using the New Agent wizard, click **Next**.

b. Click **Finish** to save the data source and open the Agent Editor. Select the operating systems on which the agent listens to data from socket clients in the **Operating Systems** section of the Socket Provider Settings page. To open the page, click **Socket Provider Settings** in the outline view or click **Global Settings** in the Agent Editor on any socket attribute group page.

Note: Error codes and supplemental files can be updated in the **Error Codes** and **Supplemental Files** sections of the Socket Provider Settings page.

Sending socket information to the agent

When your agent contains one or more socket attribute groups, the agent opens a socket and listens for data from clients.

The application that sends socket data to the agent connects to a port that is defined in the agent. The port is either the value that is set by an agent configuration property or an ephemeral port that is allocated automatically by TCP/IP. For more information about socket ports and configuration, see "Socket configuration" on page 295.

The data that is received must follow a structured XML format. The following XML information flows are possible by using the socket data source:

- Send one or more rows of data to the agent for a sampled attribute group
- Send a row of data to the agent for an attribute group that Produces events
- Send an error code to the agent instead of data.
- Send a task prefix registration to the agent
- Receive a task request from the agent
- Send a task response to the agent

Sending data

An attribute group is defined to receive sampled data or event data. When you create the attribute group, you specify an option that indicates whether the data to be received:

- Produces a single data row
- Produce more than one data row
- Produces events

If you select **Produces a single data row** or **Can produce more than one data row**, that is a sampled attribute group. If you select **Produces events**, then your attribute group sends an event to Tivoli Monitoring each time that a row is received.

When you view sampled data in the Tivoli Enterprise Portal, you see the latest set of collected rows. The data that is displayed for an event attribute group is the contents of a local cache that is maintained by the agent. For event data, the agent adds the new entry to the cache until the size is reached when the oldest one is deleted. For sampled data, the agent replaces the contents of the cache every time you send data.

If you select **Produces events** or **Produces a single data row**, you must send only one row of data to the agent for that attribute group in each message. You can send as many events as you want, send each event in a separate message.

Normally sampled data is collected by the agent on request, but the socket client provides updated samples on its own schedule. You can update a sampled attribute group (single row or multiple row) as often as you require. When the data is requested by Tivoli Monitoring, the agent provides the latest data.

If there are missing rows of data for the socket attribute group in the Tivoli Enterprise Portal, check the errors in the log file. Also, if the data in the attribute group is not as expected, check the errors in the log file. The socket data source attempts to process whatever it can from the input. For example, if the client sends three well-formed rows and one that is not valid (for example, malformed XML), you see:

- Three rows of data in the attribute group

- An error is logged for the malformed row in the agent's log file
- Since valid rows were returned, the Performance Object Status shows a status of NO_ERROR

For both event and sampled data, the data is sent to the agent as a single XML data flow from the socket client. Data that is sent from a socket client must always be terminated with a newline character: '\n'. The agent reads data until it sees the newline character and then an attempt is made to process what was received. Any data received that cannot be processed is discarded. The following is a sample of how you would send two rows of data to the agent for an attribute group named abc:

```
<socketData><attrGroup name="abc"><in><a v="1"/><a v="no"/><a v="5"/></in><in> \
<a v="3"/><a v="yes"/><a v="5"/></in></attrGroup></socketData>\n
```

This sample sends two rows of data to the agent where each row contains three attributes. The order of the attributes is important and must follow the order that is defined in your attribute group. The only exception to this is that the derived attributes must be skipped, regardless of where they are in your attribute group.

If the attribute group is defined in a subnode, then the subnode instance ID must be identified when data is sent to the agent. The subnode instance ID is identified by using the subnode attribute in the socketData element. A convention must be adopted for configuring subnode instance IDs for use by the socket client since the client cannot query instance IDs or configuration properties. Data sent to a subnode which is not configured is ignored.

Here is a sample:

```
<socketData subnode="app1"><attrGroup name="abc"><in><a v="1"/><a v="no"/><a v="5"/>
</in><in> \<a v="3"/><a v="yes"/><a v="5"/></in></attrGroup></socketData>\n
```

In this sample, the data is sent to the subnode with an instance ID equal to "app1". "app1" is not the managed system name, but the instance identifier that is specified when the subnode instance is configured.

The following XML elements make up the socket data:

socketData
The root element. It has one optional attribute that is called subnode that specifies the subnode instance ID.

attrGroup
This element identifies the attribute group that the socket data is for. The name attribute is required and is used to specify the attribute group name.

in This element is required to identify a new row of data. All of the attribute values for a row of data must be children of the same in element.

a The a element identifies an attribute value. The v attribute is required and is used to specify the attribute value.

Sending errors instead of data

Sometimes the application that posts socket data might not be able to collect the data necessary for an attribute group. In this case, instead of sending data to the agent, an error code can be returned. The error code gives you a way to tell the Tivoli Monitoring environment about your problem. An example error is:

```
<socketData><attrGroup name="abc"/><error rc="1000"/></attrGroup></socketData>\n
```

The error code must be defined in the agent in a list that is common to all of the socket attribute groups. When the agent receives an error code, the defined error message is logged in the agent log file. In addition, the attribute group named Performance Object Status has an Error Code attribute is updated with the Error Code Type. The Error Code Type is defined for the error code you send.

For the previous example, you must define the Error Code Value of 1000 in the agent. See the following sample error code definition:

Table 26. Sample error code

Error Code Value	Error Code Type	Message
1000	APP_NOT_RUNNING	The application is not running

When the error code is sent, a message similar to the following is logged in the agent log file:

```
(4D7FA153.0000-5:customproviderserver.cpp,1799,"processRC") Received error code 1000
 from client. \Message: K1C0001E The application is not running
```

If you select Performance Object Status query from the Tivoli Enterprise Portal, the **Error Code** column for the row **abc** attribute group shows the value `APP_NOT_RUNNING` in that table.

Sending an error to a sampled attribute group clears any data that was previously received for that attribute group. Sending data to the attribute group causes the error code to no longer be displayed in the Performance Object Status attribute group. You can also send an error code of 0 to clear the error code from that table.

Sending an error to an attribute group that produces events does not clear the cache of events that were previously sent.

Handling take action requests

The socket client can register to receive take action requests from the agent when the action command matches a certain prefix. Any action that does not match is handled by the agent. The prefix must not conflict with actions that the agent is expected to handle, so use the agent product code as the prefix. Take actions provided with the Agent Builder are named after the data source that the take action uses. For example, the JMX_INVOKE take action operates on the JMX data source. Another example is the SSHEXEC take action which uses the SSH script data provider. Since these actions do not use the product code, the product code is a safe prefix to use as the take action prefix.

The socket client must be long running and leave the socket open. It must send a registration request for the prefix and listen for requests from the socket. The agent ensures that a timeout does not occur on the socket of a long-running client, even if no data is flowing. The following is a sample registration request:

```
<taskPrefix value="K42"/>\n
```

In this sample, any take action command that is received by the agent that begins with `"K42"` is forwarded to the socket client that initiated the registration. The following shows a sample take action request that the socket client might receive:

```
<taskRequest id="1"><task command="K42 refresh" user="sysadmin"/></taskRequest>\n
```

The `id` is a unique identifier that the agent uses to track requests that are sent to clients. When the socket client responds to the task, it must provide this identifier in the `id` attribute of the `taskResponse` element.

The socket client must process the action and send a response. A sample response is:

```
 <taskResponse id="1" rc="1"/>\n
```

If the action completes successfully, an `rc` attribute value of `0` is returned. The value of `rc` must be an integer, where any value other than 0 is considered a failure. The task return code value is logged to the agent log file and shown in the Take Action Status query that is included with the agent. The dialog that is displayed on the Tivoli Enterprise Portal after an action is run does not show the return code. That dialog indicates whether the take action command returned success or failure. The agent log or Take Action Status query must be viewed to determine the actual return code if a failure occurred.

It is the agent developer's responsibility to document, create, and import any actions that are supported by the socket clients that are used with an agent. If users send unsupported actions to the socket client, the client must be developed to handle those scenarios in an appropriate manner. If users define more actions that start with the registered prefix, they are passed to the client. The client must be developed to handle those scenarios in an appropriate manner.

There is a timeout that controls how long the agent waits for a response from the socket client. The setting is an environment variable that is defined in the agent that called `CDP_DP_ACTION_TIMEOUT` and the default value is 20 seconds.

Note: The error code messages that are defined for socket data source attribute groups are not used for take actions. You can return the same return code values. However, the agent does not log the message that is defined or affect the `Error Code` field in the Performance Object Status attribute group.

Encoding of socket data

The socket client encodes data that is sent to the agent.

It is important to be aware of how your socket client is encoding data that is being sent to the agent.

Special characters

Data sent to the agent must not contain any newline characters except at the end of each event or data sample. Newline characters that occur inside of attribute values must be replaced with a different character or encoded as shown in (Table 27). You must also be careful not to break the XML syntax with your attribute values. The following table shows the characters that occur in the attribute values that you encode:

Table 27. Characters to encode in attribute values

Character	Header
&	`&`
<	`<`
>	`>`
"	`"`
'	`'`
\n	`
`

Note: The agent uses the newline character to separate responses received from a client. Unexpected newline characters prevent data from being parsed correctly.
The agent does not contain a full-featured XML parser so you must not use special encoding for characters not in (Table 27). For example, do not encode `¢` or `&cent` in place of a cent sign ¢.

Character sets

In addition to encoding special characters, the agent must know what code page was used to encode your data. Define each socket attribute group to indicate whether you are sending the data to the agent as **UTF-8** data or as **Local code page**. Be aware of how your client is sending data. If you use a client that is written in Java, specify **UTF-8** as the encoding on the writer you use to send data to the agent. Specify **UTF-8** as the **Code Page** for your attribute group. **Local code page** means the local code page of the agent. If the data is sent over a remote socket, it must conform to the local code page of the agent or use UTF-8.

Numeric Data

Be aware of how you are formatting your numeric attribute values. The numeric values that you send to the agent must not contain any special characters. One example is the thousands separator character. Other examples are currency symbols or characters that describe the units of the value. If the agent encounters a problem when it is parsing numeric data, it logs an error that indicates the issue. The Performance Object Status Error Code is not set when an attribute fails to parse. The following is an example error message from the agent log:

```
(4D3F1FD6.0021-9:utilities.cpp,205,"parseNumericString") Invalid characters :00:04 \
found getting numeric value from 00:00:04, returning 0.000000
```

Note: For information about how a time stamp attribute must be formatted, see ("Time stamp" on page 60).

Socket errors

Errors are written to the agent log file for problems that occur with data received from a socket client.

Other errors that are logged are take actions that return a value other than `0`. Error values that are sent by the socket client are logged along with the message associated with the error code.

The Performance Object Status for the attribute group is set when the socket client sends an error return code to the agent. Some other values can be seen in addition to the ones defined by the agent. The following table describes other "Error Code" values you are likely to encounter with socket attribute groups:

Table 28. Performance Object Status values

Error Code	Description
`NO_ERROR`	No error occurred. Indicates that there are no problems with the attribute group. Problems with a row of sampled data do not cause the state to change from `NO_ERROR`. You must validate the number of rows that are shown and the attribute values even when you see `NO_ERROR` as the error code.
`NO_INSTANCES_RETURNED`	A socket client sent no rows of data for a sampled attribute group. Not an error. It indicates that there are no instances of the resources that are being monitored by this attribute group.
`XML_PARSE_ERROR`	The agent failed to parse data that is received from the client. See the agent log for more details.
`OBJECT_CURRENTLY_UNAVAILABLE`	The client sent the agent an error code that was not defined in the global list of error codes.
`GENERAL_ERROR`	A problem occurred collecting data from the client, usually because the client did not reply to the request within the timeout interval. See the agent trace log for more details. The client can also specify GENERAL_ERROR as an error code, but it is better if a more detailed error code is defined.

Socket configuration

After you add a socket data source to your agent, you can configure the agent to accept data from a specified socket port.

About this task

After you add a Socket data source, the configuration is displayed on the Runtime Configuration page of the Agent Editor. The Socket configuration section contains the following property:

Table 29. Socket configuration property

Name	Valid values	Required	Description
Port number	0 or any positive integer The default value is 0	Yes	The port that the agent uses to listen on for data from socket clients. A value of 0 indicates that an ephemeral port is to be used.

The agent writes the value of the port that is being used to a file. Socket clients that run on the agent computer can later read this file to determine which port to connect to. The file that the port is written to is named *kxx_instanceName*_cps.properties, where: *kxx* is the three character product code of the agent and *instanceName* is the agent instance name for a multiple instance agent. If the agent is not a multiple instance agent, this part of the name is not included so the file name is *kxx*_cp.properties.

In Windows, the file is written to the %CANDLE_HOME%\TMAITM6 directory for 32-bit installations or %CANDLE_HOME%\TMAITM6_x64 for 64-bit installations. In UNIX, the file is written to /tmp.

Procedure

1. Optional: Set the environment variable CDP_DP_HOSTNAME to the host name or IP address of your network interface, if your system has multiple interfaces:
   a. Go to the Agent Editor Agent Information view and select **Environment Variables**.
   b. Click **Add** and select CDP_DP_HOSTNAME from the list of environment variables by using the Name field.
   c. Set the host name or IP address in the Value field.
2. Start your agent.

   When the agent is started, it binds to the interface that is defined by the CDP_DP_HOSTNAME environment variable. If CDP_DP_HOSTNAME is not set, the agent binds to the default host name.

   If you want the agent to bind to a defined port instead of an ephemeral port, you can set the configuration property **Port Number** (CP_PORT).

   To set the port number configuration property, use the following steps:
   a. Go to the Agent Editor Runtime Configuration view.
   b. In the Runtime Configuration Information pane select **Configuration for Socket** > **Socket** > **Port Number**
   c. Enter a port number value in Default value. If you do not enter a value, a value of 0 is used. A value of 0 indicates that an ephemeral port is used.

Remote socket port connection

You can configure your agent to accept data from a remote socket port. The agent must run on a system that has a network interface connection to a remote system.

Procedure

1. Set the value of the environment variable CDP_DP_ALLOW_REMOTE to YES by completing the following steps.
   a. Go to the Agent Editor Agent Information page and select **Environment Variables**.

b. Click **Add** and select CDP_DP_ALLOW_REMOTE from the list of environment variables by using the Name field.
c. Set the Value field to YES.

2. Follow the procedure that is detailed in "Socket configuration" on page 295.

Restriction:

- The data that is sent between the socket application and the agent:
  - Must conform to the XML syntax defined for a socket data provider. For more information, see "Encoding of socket data" on page 294.
  - Must be encoded in UTF-8.
  - Is sent in clear text (unencrypted). If the data contains sensitive information, the communication must be secured through an SSH tunnel or other mechanism outside the agent.
- The agent processes data that is received from any remote hosts, so the environment must be secured with appropriate firewall or network traffic filters.

Results

You can run code that implements a socket data provider on any system which can connect to the system where the agent is running.

Sample script for socket

This samples script demonstrates how a socket client might be written.

Perl sample

The following sample Perl script connects to a socket and sends data. This sample was written for an agent that runs on UNIX, with the product code k00 and an attribute group called SocketData.

```
#!/usr/bin/perl -w
# SocketTest.pl
# A simple Agent Builder Socket client using IO:Socket
#------------------------

use strict;
use IO::Socket;

# Initialize socket connection to the agent
#----------------------
my $host = '127.0.0.1';
my $port = 0;
# This sample is for an agent with the k00 product code.  The product code is
# used in the following line to find the file containing the port number to use.
open PORTFILE, "/tmp/k00_cps.properties" || die "Port file not found $!\n";
while (<PORTFILE>) {
    if (/^CP_PORT=([0-9]+)/) {
        $port = $1;
    }
}

if ($port == 0) {
    die "Could not find port to use to connect to agent.\n";
}

my $sock = new IO::Socket::INET( PeerAddr => $host, PeerPort => $port,
Proto => 'tcp'); $sock or die "no socket :$!";

# The following call sends 2 rows of data to the agent.  Each row contains 1
# String attribute and 3 numeric attributes.
syswrite $sock, "<socketData><attrGroup name=\"SocketData\"><in><a v=\"A message
```

```
from perl\"/>  \<a v=\"1\"/><a v=\"2\"/><a v=\"123\"/></in><in><a v=\"More from
perl\"/><a v=\"456\"/>  \<a v=\"123\"/><a v=\"789\"/></in></attrGroup>
</socketData>\n";

close $sock;
```

Testing socket attribute groups

You can test the socket attribute group that you created, within Agent Builder.

Before you begin

To test the attribute group, you need a socket client to send data. An example socket client that is written with perl script can be seen at “Sample script for socket” on page 297

Restriction: Unlike most other attribute groups, you cannot test the socket attribute group while it is being created. You can test the attribute group when you complete its creation.

Procedure

1. Select an attribute group on the Agent Editor Data Source Definition page after agent creation and click **Test**. For more information about the Agent Editor, see (Chapter 6, “The Tivoli Monitoring Agent Editor,” on page 35).

   After you click **Test** in one of the previous two steps, the Test Socket Client window Figure 202 is displayed

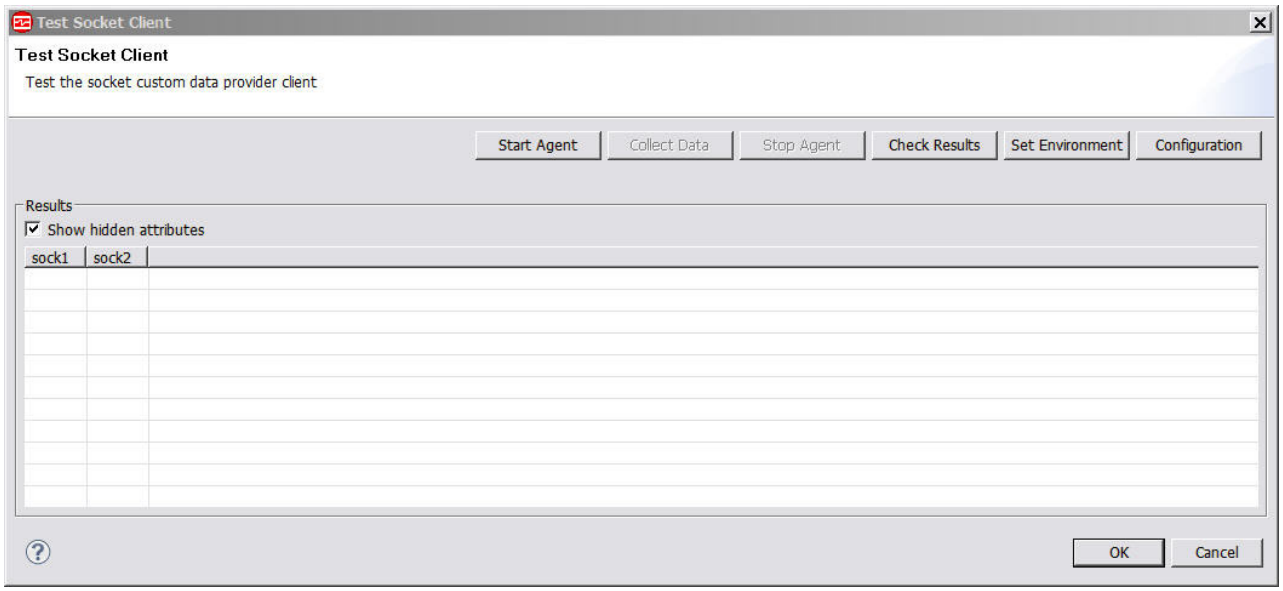


Figure 202. Test Socket Client window

2. Optional: Set environment variables and configuration properties before you start testing. For more information, see (“Attribute group testing” on page 377).
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. When the agent starts, it listens for socket data according to its configuration.
5. To test your agent 's data collection, you now generate socket data that matches the agents configuration. You can generate socket data by using a socket client. When the agent receives socket data that matches its configuration, it adds the data to its internal cache.
6. To simulate a request from Tivoli Enterprise Portal for agent data, click **Collect Data**. The Test Socket Client window collects and displays any data in the agent's cache since it was last started.

7. Click **Check Results** if something does not seem to be working as expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 545
8. Stop the agent by clicking **Stop Agent**.
9. Click **OK** or **Cancel** to exit the Test Socket Client window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, "Testing and debugging your agent," on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 26. Use the Java API to monitor data.

Use the Java API data source and the Java programming language to collect data in a long-running process that is separate from the agent process.

About this task

Use the Java API data source and the Java programming language to collect data that cannot be collected by using other Agent Builder data sources. The agent starts the Java application and sends a shutdown request when it is time to shutdown. The Java application must exit only when it is requested to do so.

An agent that contains Java API attribute groups interfaces with the Java application process. The Java application uses the Java Provider Client API to interface with the agent. For information about the API, see the Javadoc on the Tivoli Monitoring and OMEGAMON® XE Information Center. Using the Java API you can:

- Connect to the agent process and register for attribute groups that are supported by the Java application
- Receive and reply to a request for sampled data
- Send data asynchronously for an attribute group that produces events
- Send an error for an attribute group where data collection is failing
- Support attribute groups in subnodes with configured subnode instances
- Receive and reply to a "Take Action" request

Use the following procedure to create an attribute group which collects data in a Java application and sends it using the Java API. The procedure shows how to create a sample Java application to use as a starting point for your Java application.

Procedure

1. On the Agent Initial Data Source page or the **Data Source Location** page, click **Custom programs** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

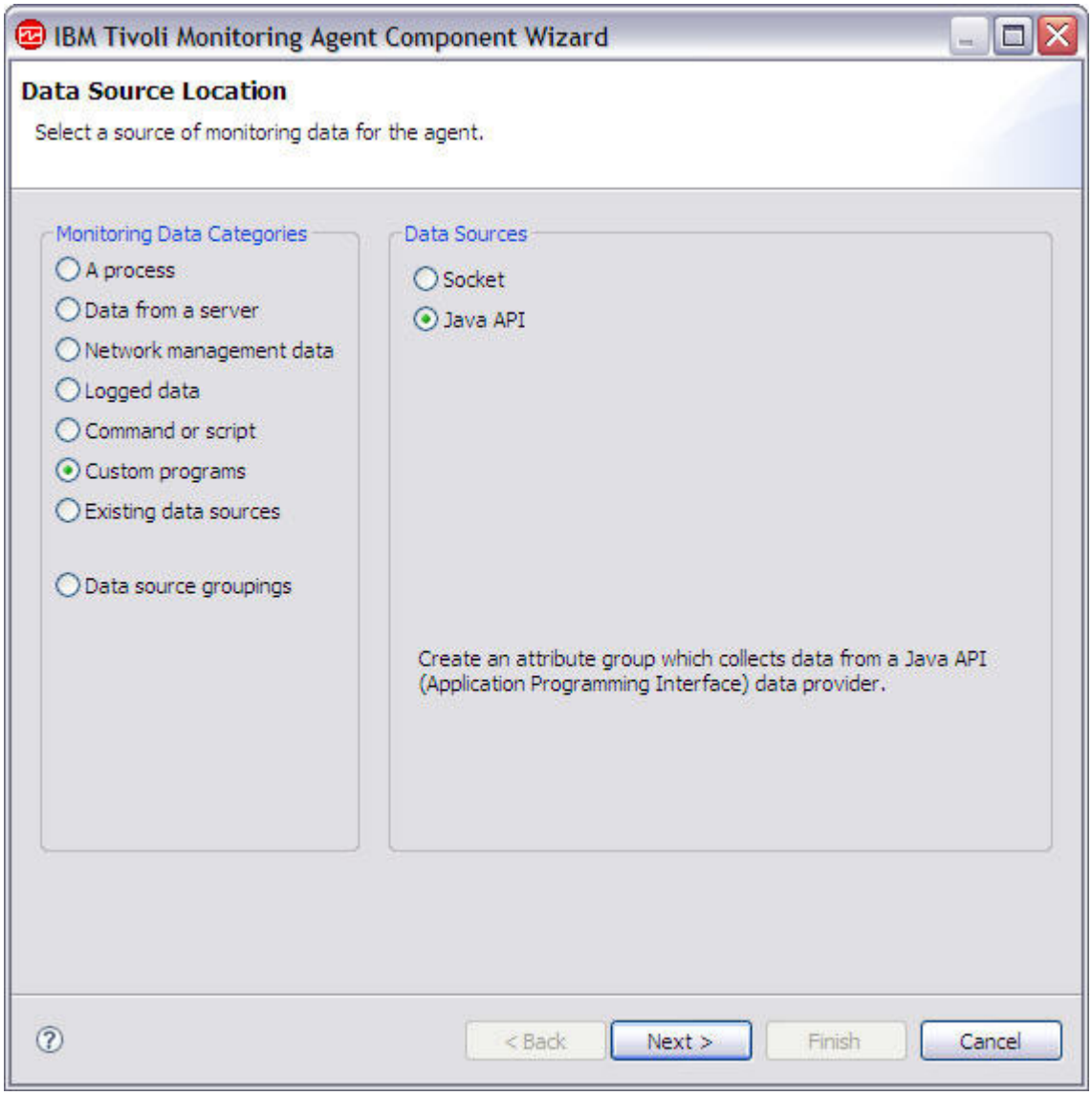


Figure 203. Monitoring data with Java API

2. In the **Data Sources** area, click **Java API**.
3. Click **Next**.
4. On the **Java API Information** page, enter an Attribute group name.

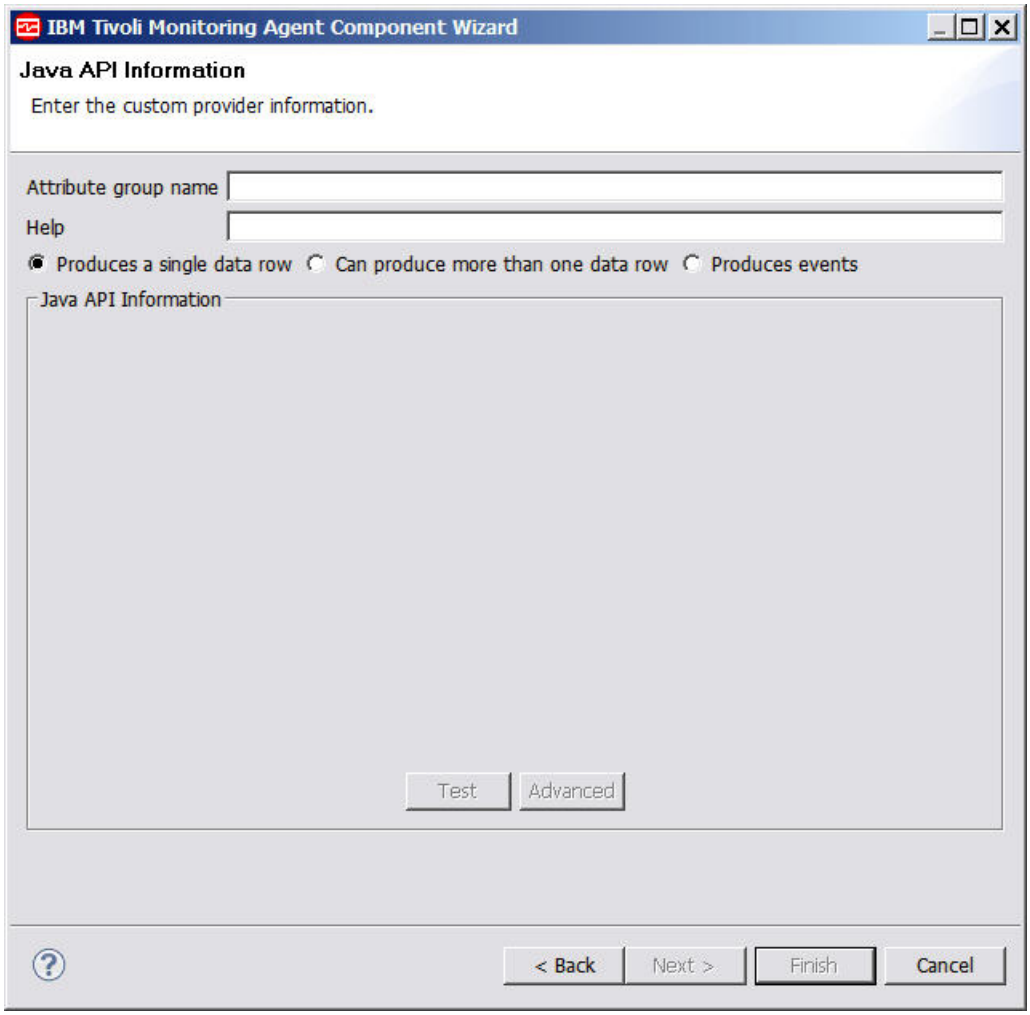


Figure 204. Java API Information window

5. Enter a help text for the attribute group.
6. Select whether the attribute group **Produces a single data row**, **Can produce more than one data row**, or **Produces events**. This choice affects the sample Java application that is created at the end of the wizard. For more information, see ("Sending data" on page 291).
7. Optional: Click **Advanced** to modify the advanced properties for the attribute group. **Advanced** is available when you select that the attribute group **Can produce more than one data row**, or **Produces events**.
8. Click **Next**.
9. On the Attribute Information page, specify the first attribute for the attribute group. For more information about creating attributes, see ("Creating attributes" on page 56).
10. Select **Add additional attributes** and click **Next** to add other attributes to the agent. References to the attributes are incorporated into the sample Java application that is created at the end of the wizard.
11. Click **Next**.
12. On the **Global Java API Data Source Information** page, enter a Class name and a JAR file name.

    The class name is a fully qualified class name whose main method is called when Java is started. The sample Java application is created with the main Java method in this class.

The JAR file is the archive that contains the Java classes that comprise the Java application. The JAR file is packaged and installed with the agent.

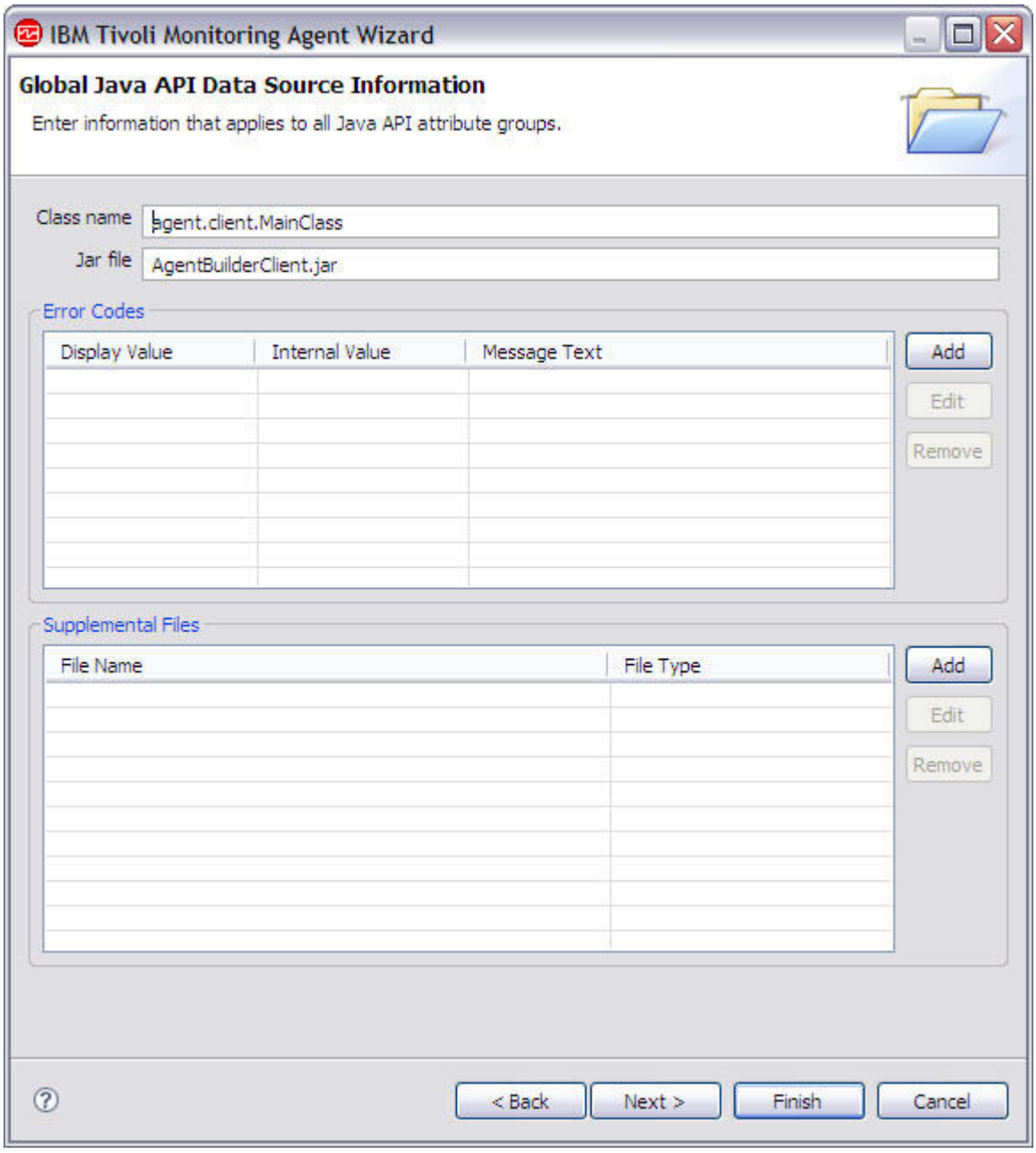


Figure 205. Global Java API Data Source Information page

13. Optional: Define the error codes that the Java application can send, on the Global Java API Data Source Information page, **Error Codes** section. These error codes are sent by the Java application when it cannot collect data.

    Restriction: An error code has a limit of 256 characters. Only ASCII letters, digits, and underscores are allowed. No spaces are allowed.

    a. Click **Add** in the Error Codes section.

    b. In the **Java API Error Code Definition** window, enter a display value.

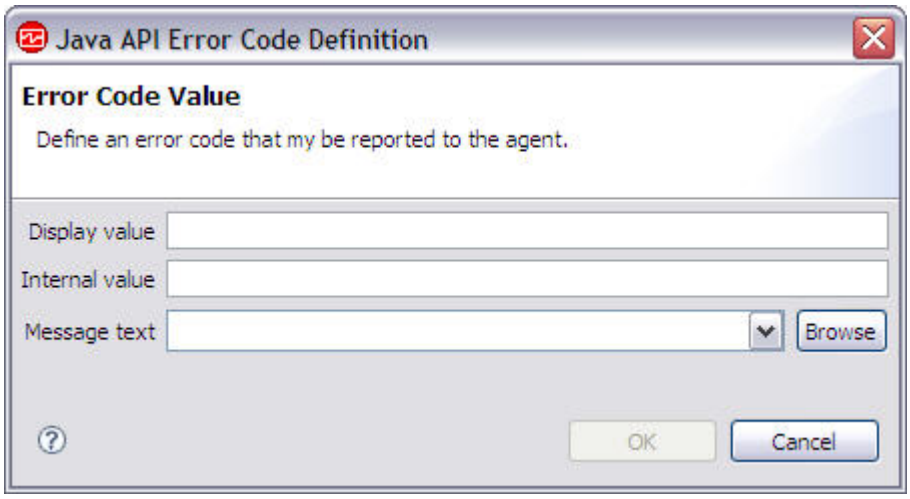


Figure 206. Java API Error Code Definition window,

c. Enter an internal value. The internal value must be an integer from 1,000 to 2,147,483,647.

d. Define a message text for each error. You can use message text that was entered previously by selecting it from the list. Click **OK** to return to the **Global Java API Data Source Information** page. The message is logged in the agent log file.

e. If no suitable message text is available, click **Browse** to set up the message text. The Messages (list) window is displayed (Figure 207). The message window lists messages that are defined in the agent. Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you defined.

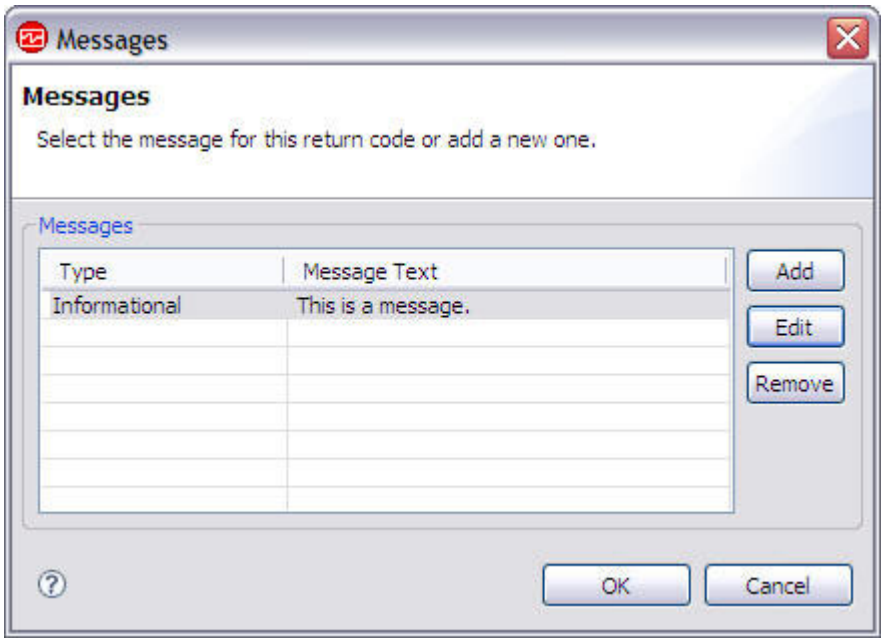


Figure 207. Messages window

f. In the Messages (list) window, click **Add** to see a Message Definition window. In the Message Definition window, you can type the text that describes the meaning of the new message and select the message type.

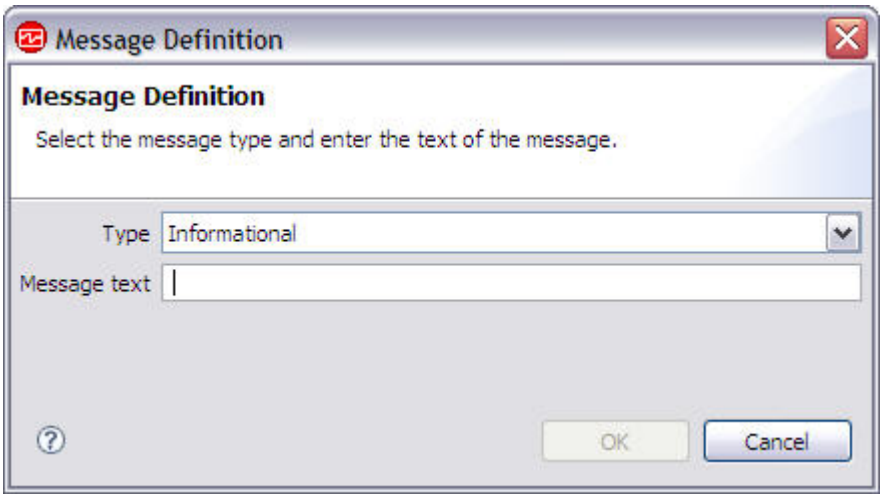


Figure 208. Message Definition window

Note: The message identifier is automatically generated for you.

g. Click **OK**.

h. The Messages (list) window is displayed with the new message. To verify the message and return to the Global Java API Data Source Information page, click **OK**.

14. Optional: In the **Supplemental Files** section of the **Global Java API Data Source Information** page, you can add files that are packaged with the agent and copied to the agent system on agent installation. The Java provider client API JAR file is not listed here; it is automatically copied to the agent system. The **File Type** column describes how each file is expected to be used. Three possible uses are described in the following table (Table 30). Click **Edit** to edit the imported file. For more information, see ("Editing a command file definition" on page 224).

Table 30. File types for supplemental files

File type	Description
Executable	Select this option if you want to include an executable file with the agent. The agent does not use this file, but it is in the path for the Java application to use.
Library	Select this option if you to include a library with the agent. The agent does not use this file, but it is in the library path for the Java application to use.
Java resource	Select this option to include Java resources with the agent. The agent does not use this file, but it is in the class path for the Java application to use.

Note: When a Java resource supplemental file is added to the Agent Builder, the file is automatically added to the project class path. The Java compiler uses the supplemental file to resolve any references that your code has, to classes in the resource.
For information about where the Supplemental Files are installed with your agent, see ("New files on your system" on page 392).

15. Optional: Create a filter to limit the data that is returned by this attribute group, if the data is sampled. Create a filter by clicking **Advanced**.

Note: The data is sampled if you did not select **Produces events** on the Java API Information page. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 67.

16. Optional: Add configuration properties to the subnode. If you are adding this data source to a subnode, the Subnode Configuration Overrides page is shown so you can add configuration properties to the subnode. At least one configuration property is needed under the subnode for the sample Java application to be created. At least one configuration property is needed because the sample uses a configuration property to distinguish one subnode instance from another.
17. Do one of the following steps:
    a. If you are using the New Agent wizard, click **Next**.
    b. Otherwise, click **Finish** to save the data source and open the Agent Editor.

What to do next

Select the correct operating systems on the Java API Settings page. Make this selection if this attribute group and the Java application, run on operating systems different from the operating systems that are defined for the agent. To open the page, click **Java API Settings** in the outline view or click **Global Settings** in the Agent Editor on any Java API attribute group page.

Note: Error codes and supplemental files can be updated later in the **Error Codes** and **Supplemental Files** sections of the **Java API Settings** page.

Running the Java application

Information about the initialization of the Java application and its dependencies

Initializing the Java application

The agent starts the Java application while the agent is starting and initializing. Configuration settings are used to control which Java run time is used to start the process. Java virtual machine arguments and the Java logging level can also be specified in the configuration. For more information about Java API configuration, see ("Java API configuration" on page 316). The Java process inherits the environment variables that are defined for the agent. Runtime configuration settings are also placed in the environment and can be queried by using API calls.

The Java application must be a long-running process. It must not terminate unless it receives a shutdown request from the API. If the Java application does terminate after it is registered with the agent, the agent will attempt to restart the Java application up to three times. If data collection is successfully resumed, this restart count is reset. The agent logs an error when a Java application terminates and when a restart is initiated.

Note: If the Java application terminates before attribute group registration is completed, no restart is attempted.

Dependencies

A Java application must use a Java runtime environment. The following versions of Java are supported:

- Oracle Corporation Java Version 5 or later
- IBM Corporation Java Version 5 or later

Java must already be installed on the agent system when the agent is configured and started. The JAR file that contains the API used to communicate with the agent is included with the agent runtime and included in the classpath of the JVM. Any additional JAR files that are needed by your Java application must be defined as Supplemental Files to the Java API attribute groups. Any supplemental files that have a *File Type* of *Java resource* are automatically added to the base classpath of the Java application, along with the Java API JAR file.

Any JAR files that are necessary for the runtime operation of the Java application that are not included with the agent, must be included in the *Classpath for external jars* configuration setting.

Generated sample Java application

A reference that describes the code the Agent Builder generates and the code you must add or replace for the resources you want to monitor.

When you create an agent with one or more Java API data sources, the Agent Builder generates Java application source code. The code is generated in the agent project and follows the structure of your agent. You must add your own Java code to the generated application. Your code collects data for sampled attribute groups, handles events to be posted to event-based attribute groups, reports errors if problems are encountered, and runs tasks. The generated application supplies the agent with data, but it is sample data, to be replaced with data obtained from the resources you want to monitor.

A sample agent is assumed that has the following characteristics:

- Product code: `K91`
- Java API Main class: agent.client.MainClass
- Agent data source structure as shown in (Figure 209):

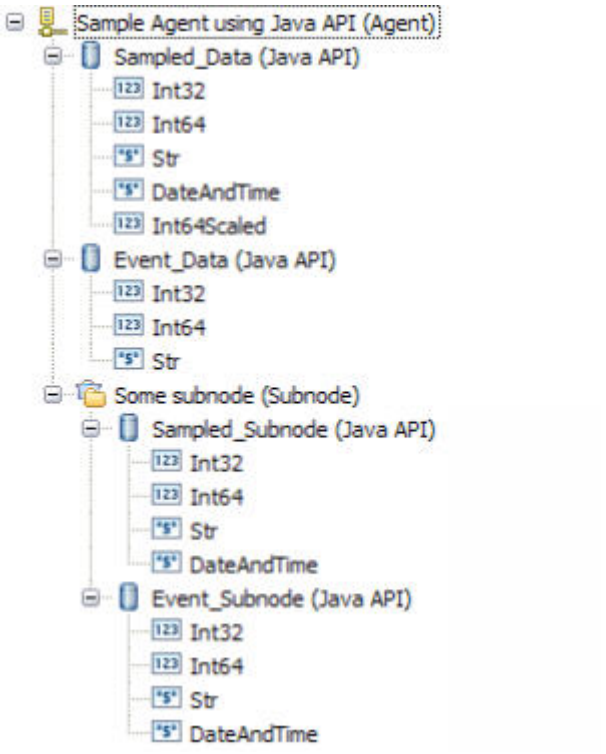


Figure 209. Sample agent structure

- *Some subnode* configuration property: `K91_INSTANCE_KEY`

Class structure

The generated Java application separates, to a great degree, code that interfaces with the agent from code that interfaces with the resources you are monitoring. It contains files that you modify, and files that you do not modify.

The following Java classes are created by the Agent Builder:

MainClass (agent.client package)
The class that you specified on the Global Java API Data Source Information page. This class contains a main method and a method that handles *take action* requests. This class inherits from the helper class described next. You must modify this class to interface with resources you want to monitor and the actions you want to take.

MainClassBase (agent.client package)
A helper class which initializes the connection to the server, registers attribute groups, and waits for requests from the server. Do not modify this class.

Sampled_Data, Sampled_Subnode, Event_Data, and Event_Subnode classes (agent.client.attributeGroups package)
There is one class for each Java API attribute group which handles data collection requests for the attribute group or generates events for the attribute group. These classes each inherit from one of the helper classes described next. You must modify these classes to gather data from the resources you want to monitor.

Sampled_DataBase, Sampled_SubnodeBase, Event_DataBase, and Event_SubnodeBase classes (agent.client.attributeGroups package)
Helper classes, one for each Java API attribute group, which define the structure of the attributes of the group in an internal class. Do not modify these classes.

ICustomAttributeGroup interface (agent.client.attributeGroups package)
An interface that defines public methods in each attribute group class. Do not modify this interface.

The classes which you can modify are never overwritten by the Agent Builder. The Agent Builder creates them only if they do not exist.

The helper classes and the interface are overwritten each time the Agent Builder is saved. As you modify and save the agent, the helper classes are updated to reflect any structural changes to the Java API attribute groups. The interface and helper classes contain a warning in the header that reminds you not to modify the file.

Initialization and cleanup

The main method in MainClass is called when the agent is started. It creates a MainClass instance and then enters the long-running method to receive and handle agent requests.

Most of the initialization and cleanup code must be added to MainClass. In the constructor, add initialization that is needed to create or access your resources. You might want to open connections to remote resources, create handles, or initialize data structures.

Before the agent terminates, the stopDataCollection method is called. If you want to close connections or cleanup before the Java application ends, add that code to the stopDataCollection method.

If initialization is needed only for a particular attribute group, that initialization can be added to the constructor of the attribute group class. Similarly, if any cleanup is needed only for a particular attribute group, that cleanup code can be added to the stopDataCollection method of the attribute group.

Any code in the Java application can use the logger object to write log entries. (The main helper class creates a protected logger object in its constructor. The attribute group helper objects create a protected reference to that logger in their constructors). The logger object uses the Java trace log utility. Errors and detailed trace information can be obtained from the trace log that is created by the logger. The trace information is important for troubleshooting problems with the provider.

When stopDataCollection is called, if you pass the cleanup work to another thread, wait for that thread to finish before you return from the stopDataCollection method. Otherwise, the cleanup work can be abruptly terminated when the process ends because the main thread completed.

One of the agent configuration settings is for the Java trace level. The following table shows the values that you can set in the `JAVA_TRACE_LEVEL` configuration property. If the API created the logger for you, the table shows the Level that is used by the logger.

Table 31. Java trace level options

Configured trace level	Java logging trace level	Description
Off	OFF	No logging is done.
Error	SEVERE	Trace problems that occurred in the Java application.
Warning	WARNING	Trace errors and potential errors.
Information	INFORMATION	Trace important information about the Java application.
Minimum Debug	FINE	Trace high-level details necessary to analyze the behavior of the Java application.
Medium Debug	FINER	Trace details about the program flow of the Java application.
Maximum Debug	FINEST	Trace all details about the Java application.
All	ALL	Trace all messages.

The name of the log file that is created by the Java application in this example is `k91_trace0.log`. If the agent is a multiple instance agent, the instance name is included in the log file name.

Note: Do not write messages to standard error or to standard out. On Windows systems, these messages are lost. On UNIX and Linux systems, this data is written to a file that does not wrap.

Collecting sampled attribute group data

The class for a sampled attribute group (one that collects one or more data rows) contains a collectData method, for example, `Sampled_Data.collectData`. This method is called whenever data is requested by the agent.

The helper class of the attribute group defines an inner class that is called Attributes. This class has one field for each attribute that is defined in your attribute group. Derived attributes are not included since they are calculated by the agent. The data types of attribute fields are Java equivalents to the Tivoli Monitoring attribute types, as shown in (Table 32).

Table 32. The data types of attribute fields and their IBM Tivoli Monitoring attribute type equivalents

Tivoli Monitoring type	Data type of attribute field
String	String
Numeric, 32 bit, no decimal adjustment	int
Numeric, 64 bit, no decimal adjustment	long
Numeric, non-zero decimal adjustment	double
Time stamp	Calendar

The collectData method must:

1. Collect the appropriate data from the resource that is being monitored.
2. Create an Attributes object.
3. Add the data to the fields of the Attributes object.
4. Call the Attributes.setAttributeValues method to copy the data to an internal buffer.
5. Repeat steps 1 - 4 as necessary for each data row. (You can skip steps 1 - 4 altogether and return no rows. In this case, the Error Code column of the Performance Object Status table has a value of NO_INSTANCES_RETURNED. For more information about error codes, see (“Error codes” on page 314).
6. Call AgentConnection.sendDatato send the data to the agent, or call sendError to discard data that is copied from calls to setAttributeValuesand send an error code instead.

You must collect the data from your resource (Step 1), replacing the sample data that is used in the generated application.

To populate the Attributes object, you can pass the data in using the Attributes constructor (as is done in the generated application). Alternatively use the zero-argument constructor to create an Attributes object and then assign the fields of the attributes object to the attribute values you collected. Fields have the same name as the attributes, though they start with a lowercase letter.

Collecting sampled data for a subnode

If a sampled attribute group is in a subnode, there are presumably multiple resources that you are monitoring (a different one for each subnode). You must determine which resource to collect data from. There must be one or more configuration properties that identify which resource is being monitored.

For this example, it is assumed that one configuration property, `K91_INSTANCE_KEY`, contains a value that identifies the resource from which data must be collected.

Use the following steps to find the correct resource:

1. Get the instance ID of all configured subnodes by calling AgentConnection.getConfiguredSubnodeInstanceIDs. Each subnode that is configured has a unique instance ID.
2. For each instance ID, get the `K91_INSTANCE_KEY` configuration property by calling AgentConnection.getSubnodeConfigurationProperty.
3. Find the resource that is represented by the value in `K91_INSTANCE_KEY`.

These steps might be done in the collectData method before the series of steps that are detailed in (“Collecting sampled attribute group data” on page 310).

Alternatively, you might want to do these steps in the attribute group class constructor and establish a direct mapping from instance ID to resource. An example attribute group class constructor is the Sampled_Subnode constructor. This procedure also gives you the opportunity to create handles or open connections that might be used through the life of the agent. Creating handles or open connections can make access to your resources more efficient.

The generated code creates sample resource objects of type MonitoredEntity in the constructor, and adds them to a configurationLookup map. You must remove the MonitoredEntity inner class, and replace the MonitoredEntity objects with objects that access your own resources. If you choose to do the entire lookup procedure in the collectData method, you can remove the configurationLookup map from the class.

If you choose to use the constructor, to map the subnode instance ID to your resource, the steps in the collectData method are:

1. Retrieve the instance ID of the subnode from the request parameter, by calling Request.getSubnodeInstanceID.
2. Retrieve the resource object from the map that is created in the constructor.
3. Perform the series of steps that are detailed in "Collecting sampled attribute group data" on page 310 to send data to the agent.

An arbitrary subnode property is chosen in the Agent Builder example, in this case `K91_INSTANCE_KEY`. If not the correct property, or more than one property is needed to identify the correct resource, you must choose the properties to identify the resource.

Sending events

For attribute groups that generate events, there is no periodic call to a collectDatamethod. Events are sent by your application as your resource posts them.

As an example of producing events, the generated code for an event-based attribute group creates and starts a thread which runs from an internal class named SampleEventClass. The event-based attribute group that is used in the example is the Event_Dataclass. The thread periodically wakes up and sends an event. If you want to periodically poll your resource for events, you can use the structure of the Event_Data class as it was generated:

1. From the Event_Data constructor, create and start a thread.
2. In the run method of the thread, loop until the agent terminates.
3. Sleep for a time before you check for events. You might want to change the polling interval of 5000 milliseconds to a number that makes sense for your agent.
4. Determine whether one or more events occurred. The generated application does not check, but always posts a single event.
5. For each event that must be posted, get the event data to be posted.
6. Create and populate the Attributes object (like the collectData method did for a sampled attribute group).
7. Call the Attributes.sendEventData method. Events consist of a single row, so only a single event can be sent at a time.

Alternatively, if you are working with a Java API that reports events from its own thread, you can initialize that thread in the Event_Data constructor. You can also register your own event-handling object with the event-handling mechanism of your resource. In your event handler, use the following steps:

1. Get the event data to be posted.
2. Create and populate the Attributes object.
3. Call the Attributes.sendEventData method.

In this case, you do not have to create your own thread in the Event_Data class nor would you need the SampleEventClass class.

Sending events in a subnode

When an event is detected for a subnode attribute group, the Java application must post the event to the correct subnode.

For this example, it is assumed that one configuration property, `K91_INSTANCE_KEY`, contains a value that identifies an instance of a resource which can produce events. It is also assumed that the value of the `K91_INSTANCE_KEY` property is retrieved along with data to be posted in the event. To do retrieve the property and data, the Java application does the following steps:

1. Gets the event data to be posted, along with the "instance key".
2. Creates and populates the Attributes object.

3. Gets a list of all configured subnode instance IDs by calling AgentConnection.getConfiguredSubnodeInstanceIDs.
4. For each subnode instance, fetches the value of `K91_INSTANCE_KEY` by calling AgentConnection.getSubnodeConfigurationProperty.
5. When the value of `K91_INSTANCE_KEY` is found which matches the value that is obtained with the event data, remembers the corresponding subnode instance ID.
6. Calls Attributes.sendSubnodeEventData, passing the remembered subnode instance ID.

The generated application does not do the lookup described in steps 4 and 5, but instead posts an event to the attribute group of every subnode. This behavior is probably not the correct one for a production agent.

Take action commands

Take action commands are defined either in the Tivoli Enterprise Portal or by using the `tacmd createaction` command. The actions can be imported into the agent's Agent Builder project so that they are created when the agent is installed. For more information about importing take action commands, see (Chapter 33, “Importing application support files,” on page 409).

The generated Java application registers for any actions that begin with the product code of the agent, for example, `K91Refresh`. This registration is done in the main helper class (MainClassBase) from the registerActionPrefix method. If you want to register other prefixes, or not register for actions at all, override the registerActionPrefix in (MainClassBase).

When the agent wants to run an action which starts with a prefix that your agent registered, the MainClass.takeAction method is called. You add code to call Request.getAction(), do the appropriate action, and then call AgentConnection.sendActionReturnCode to send the return code from your action. A return code of `0` means the action is successful, any other return code means the action failed.

Handling exceptions

The collectData and takeAction methods can throw any Java exception, so you can allow your collection code to throw exceptions without catching them. The handleException method (for collectData) or handleActionExceptionmethod (for takeAction) is called when the helper class gets the exception.

For collectData exceptions, you must call AgentConnection.sendError when an exception occurs or when there is a problem in data collection. The generated application passes an error code of `GENERAL_ERROR`. However, you must replace this error code with one defined by your agent that best describes the problem that was encountered. For more information about adding error codes, see Step (13 on page 304).

For takeAction exceptions, you must call AgentConnection.sendActionReturnCode with a non-zero return code.

Some of the AgentConnection methods throw exceptions that are derived from com.ibm.tivoli.monitoring.agentFactory.customProvider.CpciException. The handleException method is not called if a CpciException is thrown during the collecting of data as the helper class handles the exception.

Note: If you choose to catch your exceptions inside the collectData method rather than using the handleException method, ensure any CpciException is rethrown. You ensure CpciException is rethrown so it can be handled by the base class.

Error codes

A typical response to an exception or other resource error is to send an error code to the agent by calling the AgentConnection.sendError method. An error for an event-based attribute group can be sent at any time. An error for a sampled attribute group can be sent only in response to a collect data request, and in place of a sendData call.

If you send an error to the agent, the following happens:

1. An error message is logged in the agent trace log. This error message includes the error code and the message that is defined for that error code.
2. There is a Performance Object Status query that can be viewed to obtain status information about your attribute groups. The Error Code column is set to the Error Code type defined for the error you sent. The error status clears after data is successfully received by the agent for the attribute group. If you reply to a collect data request with a sendData call but you included no data rows, you get `NO_INSTANCES_RETURNED` in the Error Code column.

The following table describes some error codes that are internal to the agent that you can expect to see in certain situations:

Table 33. Internal error codes for the agent

Error Code	Description
`NO_ERROR`	There are no problems with the attribute group currently.
`NO_INSTANCES_RETURNED`	The Java application responded to a data collection request but provided no data. Not providing data is not an error. It generally indicates that there are no instances of the resource that are being monitored by the attribute group.
`OBJECT_NOT_FOUND`	The agent tried to collect data for an attribute group that is not registered through the client API. This error can mean that the application failed to start or did not initiate the attribute group registration when the agent tried to collect data.
`OBJECT_CURRENTLY_UNAVAILABLE`	The application sent the agent an error code that is not defined in the global list of error codes.
`GENERAL_ERROR`	A problem occurred collecting data from the application, usually because the application did not reply to the request within the timeout interval. See the agent trace log for more details. The application can also specify `GENERAL_ERROR` as an error code, but it is better if a more detailed error code is defined.

Changes to the agent

Certain changes to the agent require you to make corresponding changes to the Java application. If the structural changes are complex, you can delete any or all of the Java source files before you save the agent. You can also delete the files if you want to start over without the customizations you made,

The following table describes required modifications to the Java application source files after certain changes are made in the Agent Builder when the agent is saved.

Table 34. Changes to an agent that require modifications to the Java source

Agent change	What the Agent Builder does	Manual changes that are needed in the Java source
Change of the main class package name	• Generates all classes in the new package structure. • Removes all helper classes from the old package.	• Port main and attribute group class content from the classes in the old package to the classes in the new package. • Remove the classes from the old package after migration is complete.
Change of the main class name	• Creates new main classes. • Removes old main helper class.	• Port main class content to the new class. • Update references to the class name from the attribute group classes.
Addition of a Java API attribute group	• Creates classes for the new attribute group. • Adds registration for the new attribute group in the main helper class.	Overwrite sample code with custom logic in the attribute group class.
Removal of a Java API attribute group	Removes registration from the main helper class.	• Remove the attribute group class or port customized logic to some other class. • Remove the attribute group helper class.
Renaming of a Java API attribute group	• Creates classes for the new name of the attribute group. • Updates registration for the renamed attribute group in the main helper class.	• Port customized logic in the attribute group class with the old name to the attribute group class with the new name. • Remove the attribute group class with the old name. • Remove the attribute group helper class with the old name.
Addition of an attribute to a Java API attribute group	Updates the Attributes inner class in the attribute group helper class.	Collect data for the new attribute in the attribute group class.
Removal of an attribute from a Java API attribute group	Updates the Attributes class in the attribute group helper class.	Remove data collection for the former attribute in the attribute group class.
Renaming of an attribute in a Java API attribute group	Updates the attribute name in the Attributes class in the attribute group helper class.	Update any references to the attribute name in the Attributes class (often there are no references because the Attributes constructor, with positional arguments, is used).
Reordering of attributes in a Java API attribute group	Updates the attribute order in the Attributes class in the attribute group helper class.	Update the argument order in any calls to the Attributes constructor.

Some of the changes that are mentioned in the previous table can be streamlined if you use the Eclipse Refactor - Rename action. Use this action on all the affected names (including helper class names) before you save the changed agent.

Use of the Java API

The Java API is used throughout the generated Java application to communicate with the agent. Often your only direct interaction with the Java API is to modify a parameter of an existing method call. For example, changing a posted error code from `GENERAL_ERROR` to an error code defined in your agent.

If you want to do more extensive coding with the Java API, you can view Javadoc from the Eclipse text editor. You can view Javadoc while you edit the Java code by doing the following steps:

1. Highlight a package, class, or method name from the API.
2. Press **F1** to open the Eclipse Help view.
3. Select the Javadoc link.

You can also see a brief description from the Javadoc by hovering over a class or method name. Javadoc for the API can also be found on the Tivoli Monitoring and OMEGAMON XE Information Center, see Javadoc.

The classes for the Java API are in `cpci.jar`. The `cpci.jar` file is automatically added to the Java Build Path of the project when an agent which contains a Java API attribute group is created. The file is also added when an agent that contains a Java API attribute group is imported. The file is also added when a Java API attribute group is added to an existing agent. The`cpci.jar` is also automatically packaged with each agent that contains a Java API attribute group and added to the `CLASSPATH` of the Java application.

Java API configuration

When you define a Java API data source in your agent, some configuration properties are created for you.

If you define a Java API data source in your agent, the agent must use Java to connect to the Java API server. Java configuration properties are added to the agent automatically. The following Java configuration properties are specific to the agent runtime configuration:

Table 35. Java configuration properties

Name	Valid values	Required	Description
Java home	Fully qualified path to a directory	No	A fully qualified path that points to the Java installation directory.
Java trace level	Choice	Yes	Use this property to specify the trace level that is used by the Java providers.
JVM arguments	String	No	Use this property to specify an optional list of arguments to the Java virtual machine.
Class path for external jars	String	No	Path containing any JAR files that are not included with the agent, but are necessary for the runtime client operation.

These configuration variables are available on the Runtime Configuration Information page of the Agent Editor under **Configuration for Java Virtual Machine (JVM)**, and **Configuration for Java API**.

Testing Java application attribute groups

You can test the Java application attribute group that you created, within Agent Builder.

Before you begin

Restriction: Unlike most other attribute groups, you cannot test the Java application attribute group while it is being created. You can test the attribute group when it is added to the agent and the agent is saved. Saving the agent causes the Java code to be generated for the attribute group.

Procedure

1. Select an attribute group on the Agent Editor Data Source Definition page after agent creation and click **Test** .

   For more information about the Agent Editor, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35 After you click **Test** in one of the previous two steps, the Test Java Client window Figure 210 is displayed.

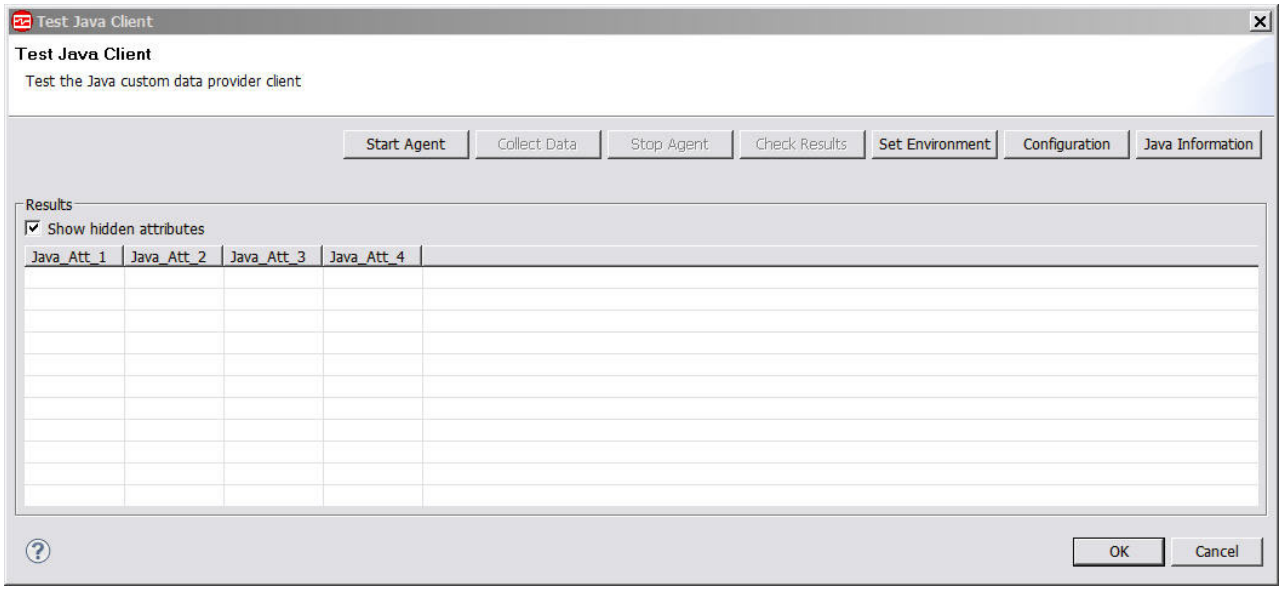


Figure 210. Test Java Client window

2. Optional: Set environment variables, configuration properties, and Java information before you start testing. For more information, see ("Attribute group testing" on page 377). For more about default Java runtime configuration properties, see ("Java API configuration" on page 316).
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent monitors the Java Client for data. The Test Java Client window displays any data that is returned.
5. Optional: Click **Check Results** if the returned data is not as you expected. The Data Collection Status window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 545
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the Test Java Client window. Clicking **OK** saves any changes that you made.

Related concepts:

Chapter 31, "Testing and debugging your agent," on page 377
After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Chapter 27. Creating attribute groups from existing sources

When at least one attribute group exists the option to create an attribute group by using an existing attribute group or groups is available to you.

The option is available on both the Agent Initial Data Source page or the Data Source Location page. You can create an attribute group by using existing data sources in the following ways:

1. Joining data from two existing attribute groups. For more information, see "Joining two attribute groups."
2. Filtering data from an existing attribute group. For more information, see "Creating a Filtered Attribute Group" on page 324.

Note: The option to join two attribute groups is available only after two or more attribute groups are created. You can then join two attribute groups to create a third attribute group that contains all the attributes that were in the original two attribute groups.

Joining two attribute groups

Create an attribute group from two other attribute groups.

About this task

Joining attribute groups is most useful when the agent collects data from two different types of data sources. For example, the agent might collect data WMI and PerfMon, or SNMP and script data sources. Each set of attributes might be more useful when used together in one Tivoli Enterprise Portal view.

For example, assume that your attribute groups are defined as follows:

```
First_Attribute_Group
 index integer
 trafficRate integer
 errorCount integer
Second_Attribute_Group
 index2 integer
 name string
 traffic string
```

One definition provides you with counters (like Perfmon) and the other provides you with identification information. Neither attribute group is useful to you by itself. However, if you can combine both attribute groups by using the index to match the appropriate rows from each, you have a more useful attribute group. You can use the combined attribute group to display the name, type, and metrics together.

This same mechanism can be used to add tags to information collected through normal attribute groups. The information can then be more easily correlated in an event system when a problem is detected. For example, a company wants to manage all its servers by collecting common data and by using common situations to monitor the health of the servers. It also wants to be able to identify the servers with more information that tells it what application is running on a particular server. It wants to have control of the values that are used on each server, but it does not want to create different agents for each application. It can accomplish this control by creating an additional attribute group in its single agent as follows:

```
Application_Information
 application_type  integer
 application_name string
 application_group string
```

© Copyright IBM Corp. 2007, 2013

This attribute group would be defined as a script attribute group that gathers its values from agent configuration. You can specify different values for each agent instance and use one agent to manage all of their systems. This attribute group would then be joined to all the source attribute groups where this application information might be needed. The information is then available in the Tivoli Enterprise Portal, situations, events, and warehoused data.

When you join two attribute groups, a third attribute group is created. This attribute group contains all the attributes that are contained within the source attribute groups.

The results of a join operation vary depending on the number of rows that each source attribute group supports. If both attribute groups are defined to return only a single row of data, then the resulting joined attribute group has one row of data. The single row contains all attributes from both source attribute groups.

Table 36. source attribute group one (single row)

Attribute1	**Attribute2**	**Attribute3**
16	some text	35

Table 37. source attribute group 2 (single row)

Attribute4	**Attribute5**	**Attribute6**	**Attribute7**
5001	more data	56	35

Table 38. Resulting join

Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**	**Attribute6**	**Attribute7**
16	some text	35	5001	more data	56	35

Suppose that one source attribute group is defined to return only one row (single-row) while the other can return more than one row (multi-row). The resulting joined attribute group contains the same number of rows as the multi-row source attribute group. The data from the single-row attribute group is added to each row of the multi-row attribute group.

Table 39. source attribute group one (single row)

Attribute1	**Attribute2**	**Attribute3**
16	some text	35

Table 40. source attribute group two (more than one row)

Attribute4	**Attribute5**	**Attribute6**	**Attribute7**
user1	path1	56	35
user2	path2	27	54
user3	path3	44	32

Table 41. Resulting join

Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**	**Attribute6**	**Attribute7**
16	some text	35	user1	path1	56	35
16	some text	35	user2	path2	27	54
16	some text	35	user3	path3	44	32

Finally, assume that both source attribute groups are defined to return more than one row. You must identify an attribute from each of the source attribute groups on which to join. The resulting attribute group contains data rows where the attribute value in the first attribute group matches the attribute value from the second attribute group.

Table 42. source attribute group one (more than 1 row)

Attribute1	Attribute2	Attribute3
16	some text	35
27	more text	54
39	another string	66

Table 43. source attribute group 2 (more than 1 row)

Attribute4	Attribute5	Attribute6	Attribute7
user1	path1	56	35
user2	path2	27	54
user3	path3	44	32

Table 44. Resulting join (joining on Attribute3 and Attribute7)

Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	Attribute7
16	some text	35	user1	path1	56	35
27	more text	54	user2	path2	27	54

With Agent Builder, you can also join user-defined attribute groups to the Availability attribute group if there are any availability filters defined in your agent. For more information about the data that is contained in the Availability attribute group, see (“Availability node” on page 541).

You can create this type of attribute group by accessing the menu on the data sources tree by right-clicking and then selecting **Join Attribute Groups**.

Procedure

1. On the Data Source Definition page, right-click one of the attribute groups you would like to join and select **Join Attribute Groups**. This option is only visible if there are at least two attribute groups defined. Having an availability filter defined counts as having an attribute group defined. The Attribute Group Information page is displayed.

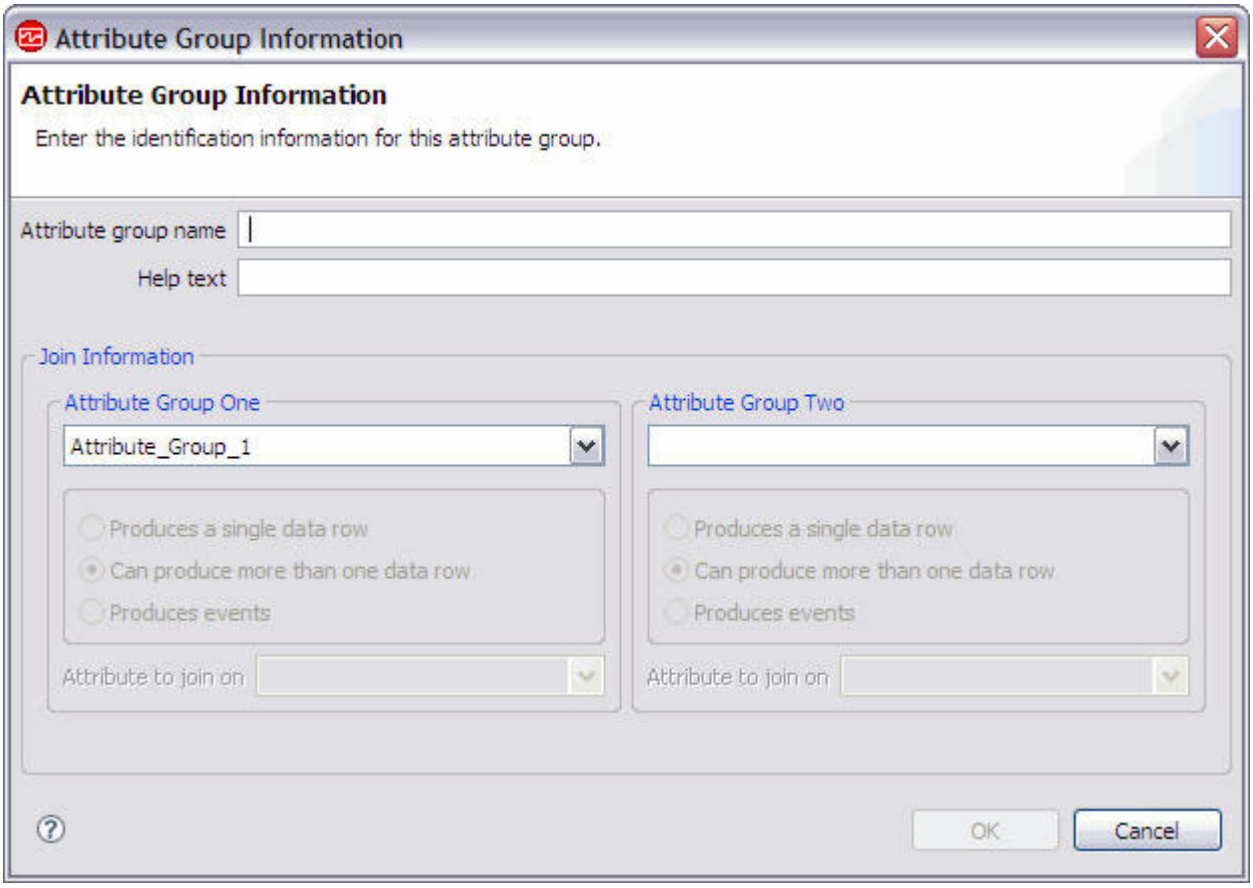


Figure 211. Attribute Group Information pageAttribute Group Information window

2. In the Join Information area, select the two attribute groups you would like to join. Select the attribute groups by choosing from the groups available in the **Attribute Group One** and **Attribute Group Two** lists. For each attribute group, either **Produces a single data row** or **Can produce more than one data row** is selected for you. This selection is locked and depends on how the source attribute groups were originally defined.

   Note: There are restrictions on which attribute groups can be joined:
   - You cannot join an attribute group in one subnode type to an attribute group in another subnode type.
   - You can join only an event attribute group to a single row non-event attribute group.

   a. Select the attribute that you want to join on for each attribute group when both attribute groups show **Can produce more than one data row**, under **Attribute to join on**.

   The **Attribute group name** and **Help** fields are filled for you using information from the chosen attribute groups. If you want to, you can change these entries.
3. Click **OK**.

Results

The joined attribute group that you created is added to the Attribute Group Information area of the Data Source Definition page

Manipulating attributes in joined attribute groups

Using attributes in joined attribute groups can impose rules on how those attributes are manipulated.

Deleting an attribute group

An attribute group cannot be deleted if it is referenced in a joined attribute group unless the joined attribute group is also being deleted.

Deleting an attribute

An attribute cannot be deleted if its parent attribute group is referenced in a joined attribute group and one of these statements is true:

- The attribute is defined as a join attribute in the joined attribute group.
- The attribute is used in any derived attribute in the joined attribute group.

Joined attributes cannot be deleted. Only derived attributes, if any are added, can be deleted from the joined attribute group.

Reordering attributes

The order of the joined attributes is fixed by the order of the source attributes. The joined attribute list cannot be reordered. Only derived attributes, if any, can be reordered.

When the version of an agent, is committed, source, and derived attributes cannot be reordered or removed. Attributes added in a new version of the agent, whether source or derived attributes, will come after all committed attributes. For more information, see "Committing a version of the agent" on page 45.

Adding an attribute

New joined attributes cannot be explicitly added. Only derived attributes can be explicitly created.

Removing availability filters

The last availability filter cannot be removed if the Availability attribute group is referenced in a joined attribute group.

Joined attributes

Manipulate information that relates to joined attributes

Procedure

- Change the attribute name and help text of the joined attribute can be changed so that they are different from the source attribute:
  1. Select the attribute in the joined attribute group in the **Attribute Group Information** pane of the **Data Source Definition** page.
  2. Enter the new name and help text.
- The joined attribute can be shown or not shown on the Tivoli Enterprise Portal by selecting or clearing the **Display attribute in the Tivoli Enterprise Portal** check box. The check box is in the **Joined Attribute Information** section of the **Data Source Definition** page. This choice is irrespective of whether the source attribute is shown on the Tivoli(r) Enterprise Portal.
- Any attribute or combination of attributes (that are shown on the Tivoli Enterprise Portal) can be marked as key attributes by selecting the **Key attribute** check box. This choice is independent of whether the attributes are key attributes in the source attribute groups. The choice is also independent of whether the source attributes are shown on the Tivoli(r) Enterprise Portal.
- Attribute type information for joined attributes is taken from the source attributes and cannot be changed in the joined attribute. In the **Joined Attribute Group Information** section of the agent editor

(Figure 212), click **Locate source attribute** to go to the source attribute.

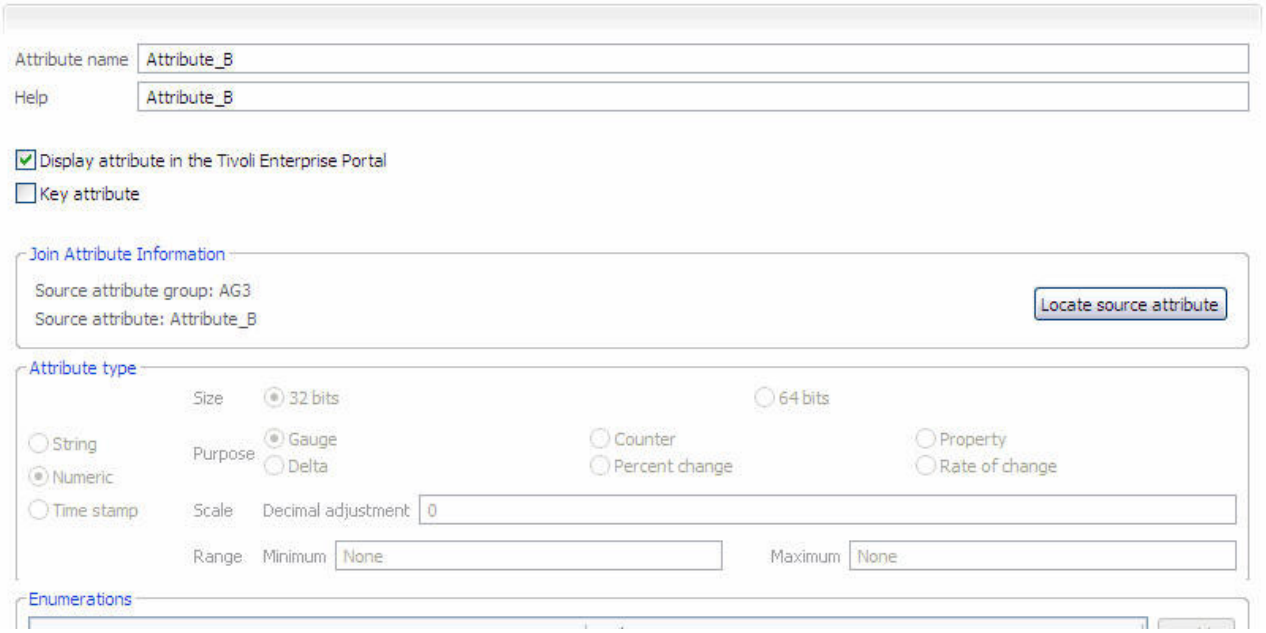


Figure 212. Locating source attribute information

Any changes to the source attribute groups are reflected in the joined attributes. If the source attribute groups change, those attributes are automatically updated under the joined attribute group. This automatic update also occurs if a different attribute group is set as the source attribute group. Changes to a source attribute type are copied to the joined attribute. Changes to a source attribute name or help text are copied to the joined attribute. However, such source attribute changes are not copied after you change the name or help text of a joined attribute.

Creating a Filtered Attribute Group

Create a filtered attribute group by filtering rows of data from an existing attribute group.

About this task

A filtered attribute group is most useful when a base data source query returns data that you prefer to divide into separate groups. Examples of such data sources are Windows Performance Monitor, SNMP and WMI.

For example, assume that a data source can return the following data:

```
Name    Type  Size  Used  Free
Memory  MEM      8     4     4
Disk1   DISK   300   200   100
Disk2   DISK   500   100   400
```

This is a table that reports on the storage that exists on the system and it includes both memory and disk space. You might prefer to break down the table into memory and disk as separate tables. You can break down the table by creating two base attribute groups. Each of these base attribute groups collects the same data and filters out the rows you do not want. However, that is not the most efficient way to do things. Instead, you define one base attribute group that returns both the memory and disk usage data together. Next, define two filtered attribute groups. Each uses the same base table as its source. One includes a filter where `Type=="MEM"` and the other includes a filter where `Type=="DISK"`.

In the example, for the filtered attribute group where Type=="MEM", the returned data is:

```
Name    Type  Size   Used  Free
Memory  MEM      8      4     4
```

and where Type=="DISK", the returned data is:

```
Name    Type  Size  Used  Free
Disk1   DISK  300   200   100
Disk2   DISK  500   100   400
```

Note: Attributes groups whose data is event-based cannot be used to create filtered attribute groups. Only attribute groups whose data is sampled can be used.

Procedure

1. Click **Existing data sources** in the **Monitoring Data Categories** area on the Agent Initial Data Source page or the Data Source Location page

   Note:

   - You reach the Agent Initial Data Source page by using the new agent wizard. For more information, see "Starting the new agent wizard" on page 13.
   - You can reach the Data Source Location page by right-clicking an agent in the Data Source Definition page of the Agent Editor and selecting **Add Data Source**.
2. Select **Filter an attribute group's data rows** in the **Data Sources** area.

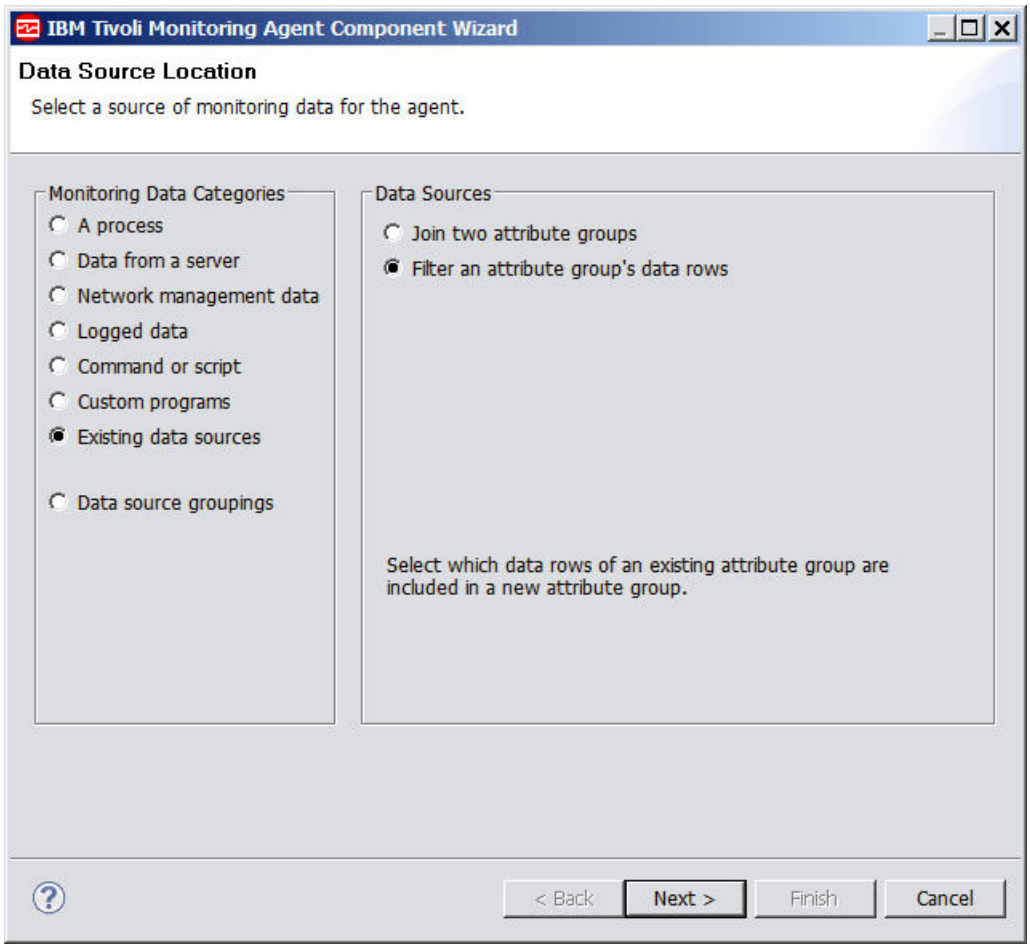


Figure 213. Adding existing data sources

3. Click **Next** The Filter Information page is displayed.

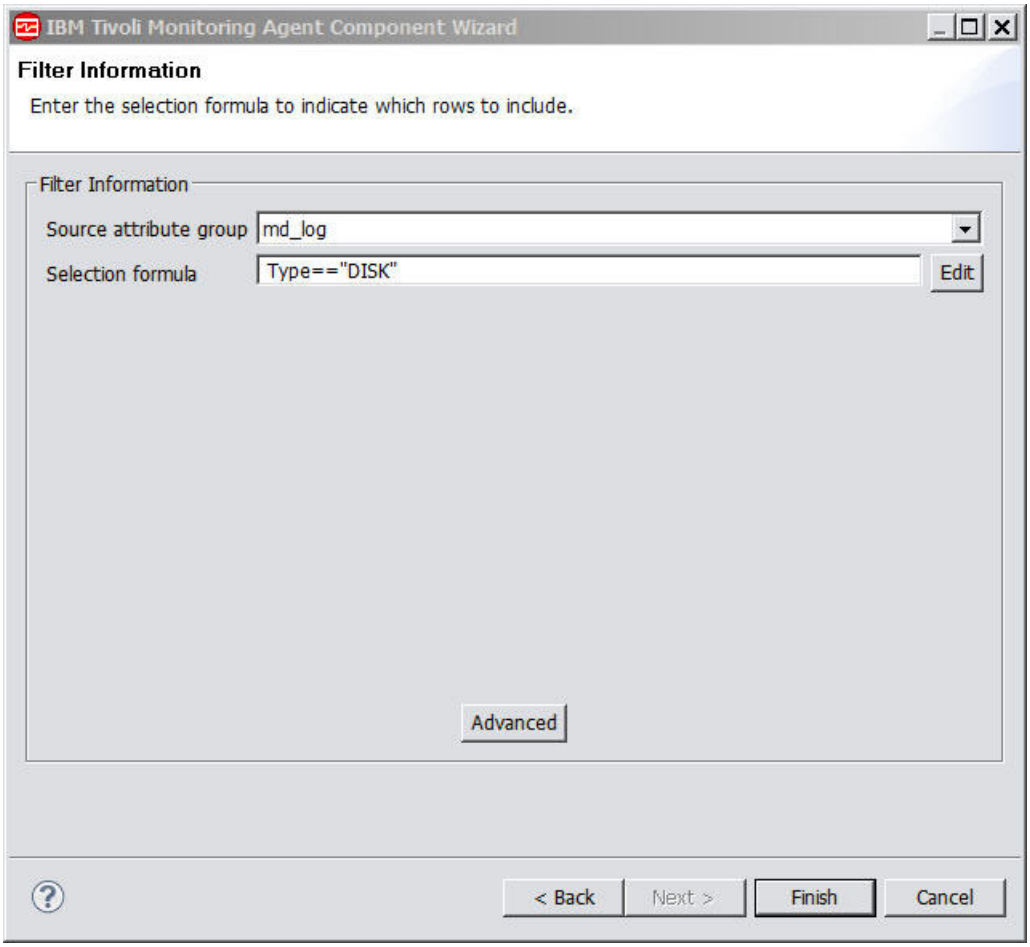


Figure 214. Filter Information window

4. Select a **Source attribute group** from the list.
5. Enter a **Selection formula** to filter the data from the attribute group you selected. For example, in the Filter Information page that is shown earlier, the selection formula filters data rows where the `Type` attribute is equal to `"DISK"`. Data rows whose `Type` attribute does not match `"DISK"` are discarded. The selection formula that you enter must evaluate to a Boolean result, true, or false.

   Note: In the Filter Information page, you can click **Edit** to enter or modify the formula by using the Formula Editor. For more information about the Formula Editor, see ("Formula Editor" on page 68).
6. Click **Next**.
7. Select **Produces a single data row** or **Can produce more than one data row**.
   a. If you selected **Can produce more than one data row**, select a key attribute or attributes from the list.
8. Click **Finish**.

Chapter 28. Creating a navigator group

Create a navigator group while you create an agent by using the IBM Tivoli Monitoring Agent wizard at the base agent level. You can also create a navigator group while you define a subnode by using the IBM Tivoli Monitoring Agent Component wizard.

About this task

Provides information about how to create navigator groups. See "Navigator groups" on page 30 for an explanation of how navigator groups are used.

Procedure

1. On the Agent Initial Data Source page or the Data Source Location page, click **Data source groupings** in the **Monitoring Data Categories** area.

© Copyright IBM Corp. 2007, 2013

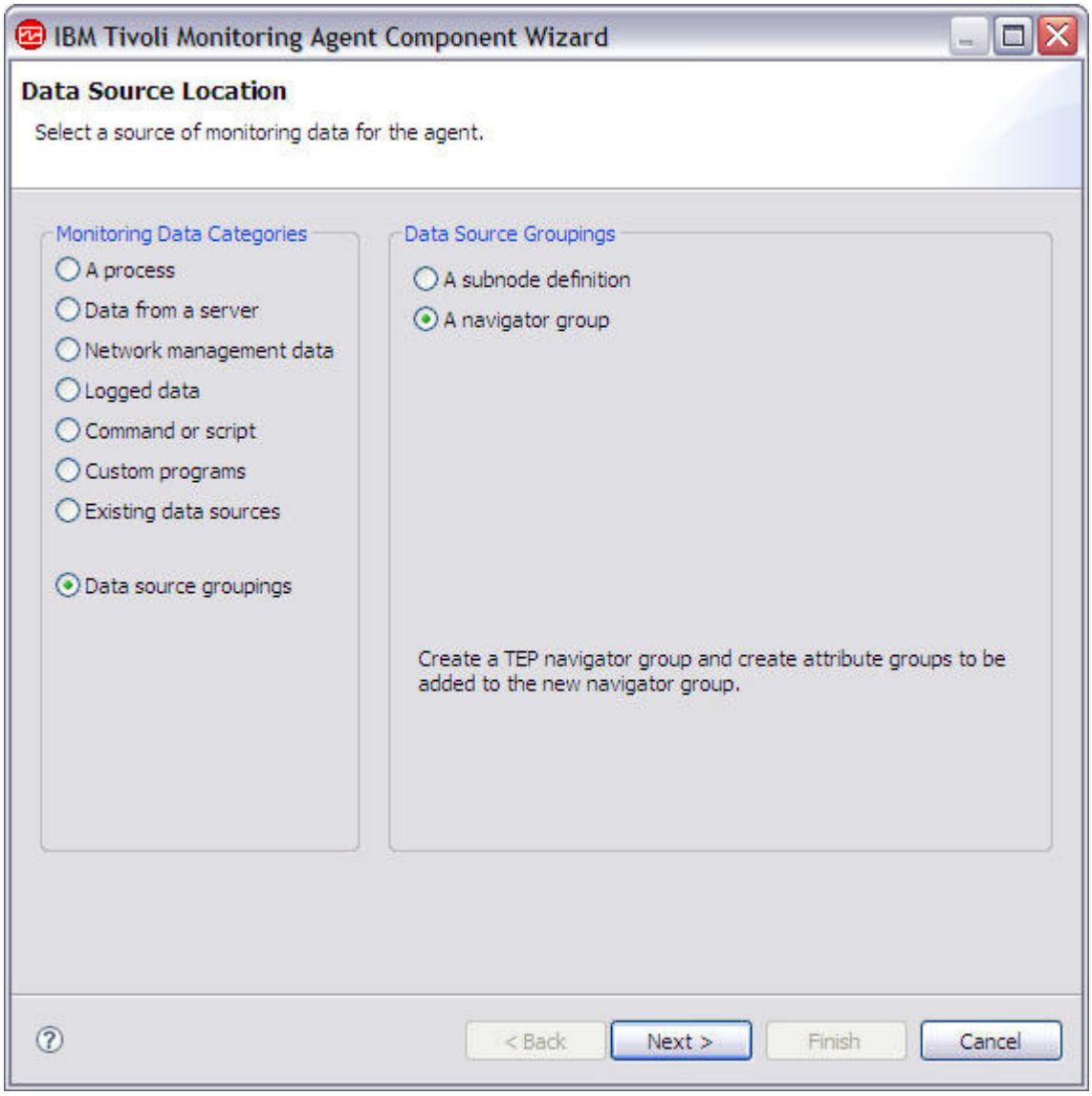


Figure 215. Selecting the initial source of monitoring data for a new agent

2. In the **Data Sources** area, click **A navigator group**.
3. Click **Next**.
4. On the Navigator Group Information page (Figure 216 on page 329), type the navigator group name and the text for the Help you want associated with the name, and click **Next**.

   Note: The Agent Builder automatically creates navigator groups in certain situations. The following navigator group name is reserved:

   - Availability

   .



Figure 216. Providing the navigator group name and help text

5. On the First Navigator Group Data Source page, select the first source of monitoring data for the new navigator group. Click a category in the **Monitoring Data Categories** list and a data source in the **Data Sources** list. Then, click **Next**.

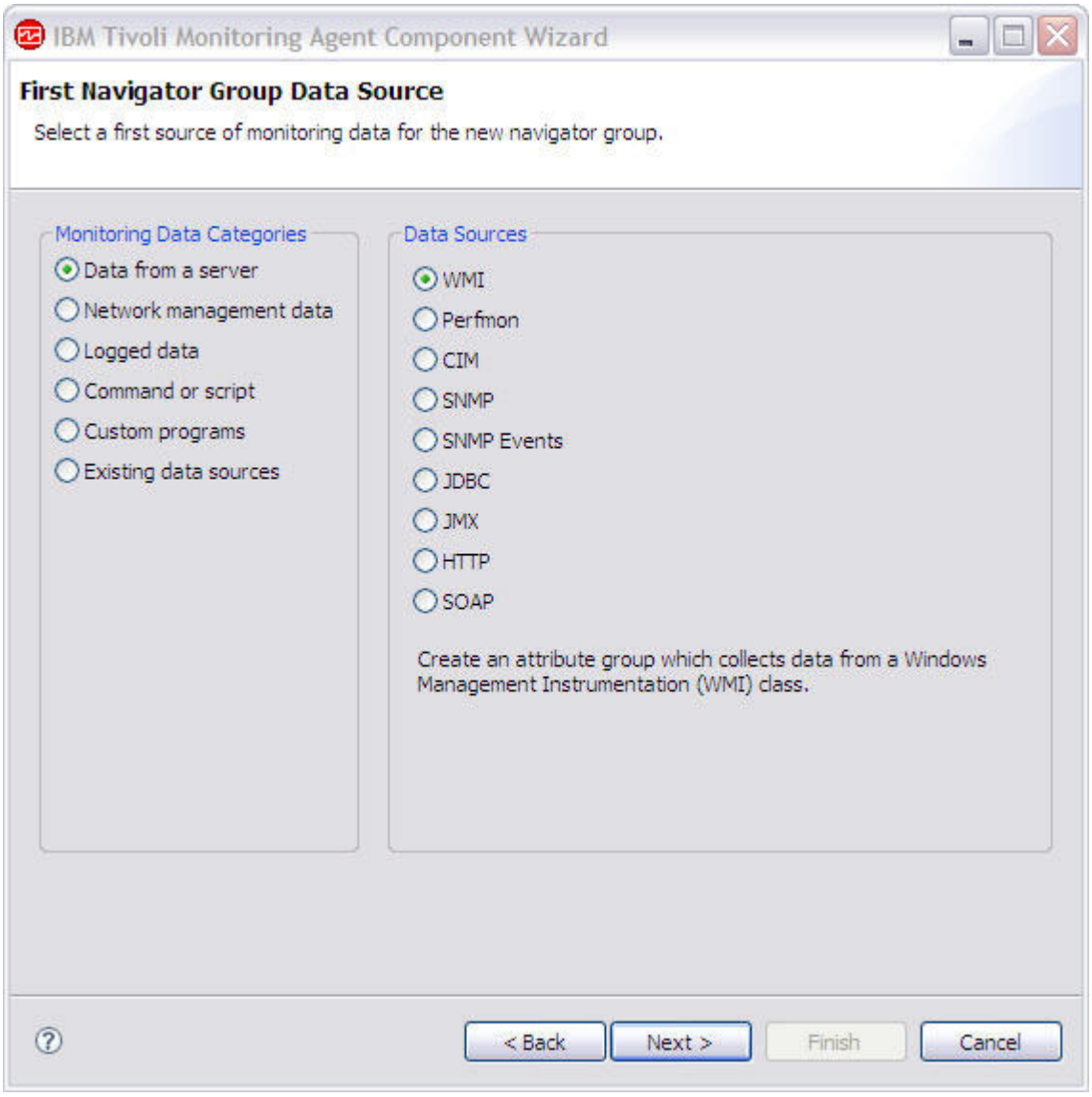


Figure 217. Selecting the first source of monitoring data

Note: The data source can be created as usual. Alternatively, click **Existing data sources** and chose to move one or more data sources that you already created into the navigator group.

6. If you want to create a data source within a navigator group, on the Data Source Definition page, select the navigator group, and click **Add to Selected**.
7. If you want to move existing data sources into the navigator group, on the Data Source Definition page, select the navigator group, and click **Add to Selected**.
8. If you want to remove a data source from a navigator group, do one of the following steps on the Data Source Definition page:
   - Select the data source, and drag it to the root of the data sources tree.
   - Select the data source, and clicking **Remove**.
9. If you want to create a navigator group, do one of the following steps on the Data Source Definition page:
   - Click **Add to Agent**.
   - Select a subnode and click **Add to Selected**.

Chapter 29. Use of subnodes

How an agent can use subnodes.

For a general introduction to subnodes, and how they are used, see (“Subnodes” on page 30).

There are two ways that a single agent can use subnodes:

- The agent can have different subnodes of the same type.
- The agent can have subnodes of different types.

Monitoring the same data from different sources

When one agent represents multiple instances of a monitored entity type, each subnode includes the same attribute groups and the correct values for the specific monitored entity instance. The number of subnodes varies based on agent configuration. The example in Figure 218 shows the monitoring of different systems.

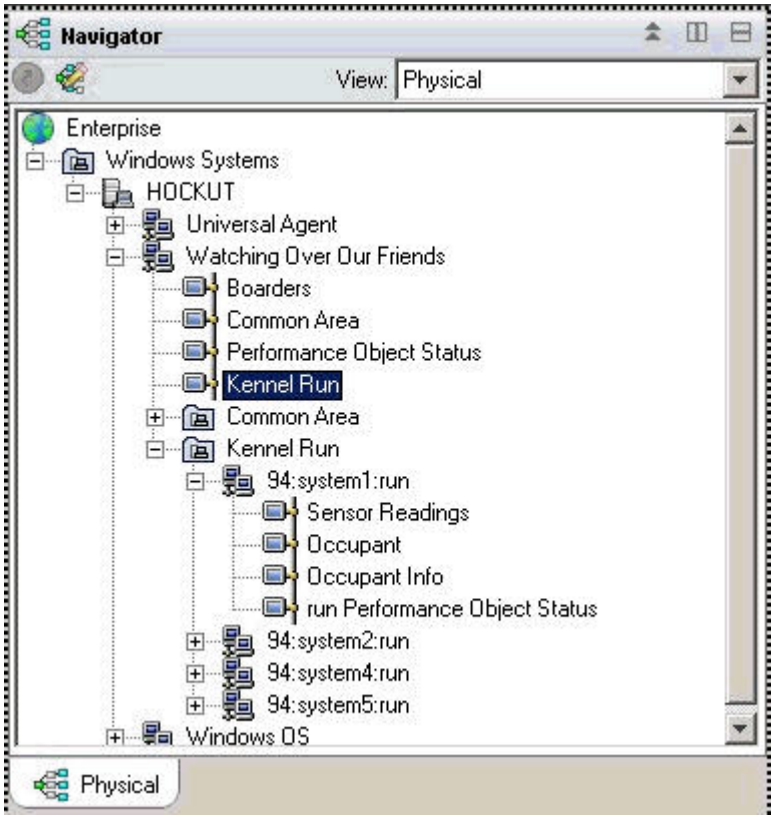


Figure 218. Subnodes monitoring different systems

Monitoring multiple types of information

When one agent monitors multiple types of monitored entities, each entity type is displayed in a separate subnode in the Tivoli Enterprise Portal physical Navigation tree. Each subnode includes the information that is defined in that subnode type. The following example shows two subnode types. Each type is monitoring a different type of entity, with different types of data available for each entity:

© Copyright IBM Corp. 2007, 2013

- Common Area
- Kennel Run

The agent in (Figure 219) runs one copy of each subnode type. A particular agent might create any subset of the defined agents. Subnodes can be used to mimic Tivoli Monitoring V5 profiles.

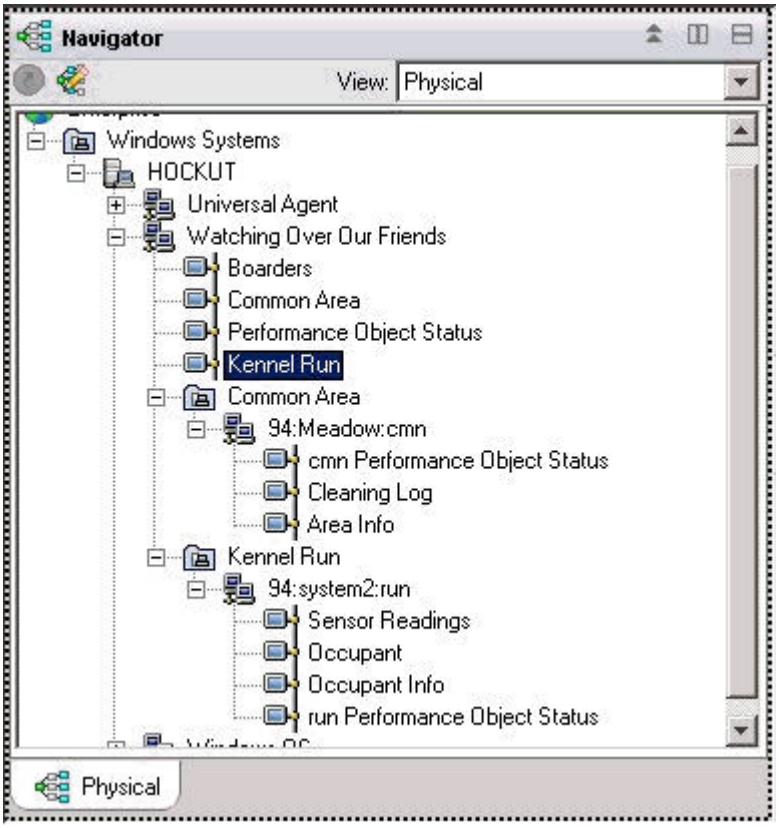


Figure 219. Subnode types in Navigator tree

There are two ways of using subnodes, different subnodes of the same type, and subnodes of different types. Both ways of using subnodes can be used in the same agent, where each type can have more than one subnode instance.

(Figure 219) shows two types of subnodes that monitor two types of entities: Common Areas and Kennel Runs. In addition, there are several subnodes that are defined for each type. There are three subnodes of type Common Area; these subnodes have the following IDs: Meadow, Hill, and Tree. There are also four subnodes of type Kennel (each collecting data from a different system that is dedicated to a Kennel Run); these subnodes have the following IDs: system1, system2, system4, and system5.

Note: The first 24 characters of subnode IDs must be unique for all instances of the subnode type in the IBM Tivoli Monitoring installation. For more information, see *Subnode name unique* in (Table 50 on page 510).

Data Providers in subnodes

A subnode can contain any mixture of data from the different data provider types. Most current Agent Builder data providers can be used in a subnode, including the following data providers:

- WMI
- Perfmon

- Windows Event Log
- SNMP
- SNMP Events
- JMX
- ICMP ping
- Script
- Log
- CIM
- JDBC
- HTTP
- SOAP
- Socket
- Java API

A subnode can also contain a joined attribute group that combines data from two other attribute groups from the same subnode or from agent-level attribute groups.

Status of subnodes

There are two ways to determine status for a subnode agent. The first way is to look at the data that is displayed in the Performance Object Status attribute group. This attribute group displays the status for each of the other attribute groups at the same level in the agent. The Performance Object Status attribute group at the agent level displays the collection status for the other attribute groups at the agent level. The Performance Object Status attribute group in each subnode displays the collection status for the attribute groups in that subnode.

The Agent Builder also creates one attribute group for each subnode type, which displays one row for each configured subnode of that type. In the example in (Figure 220 on page 334), four subnodes are running to collect data.

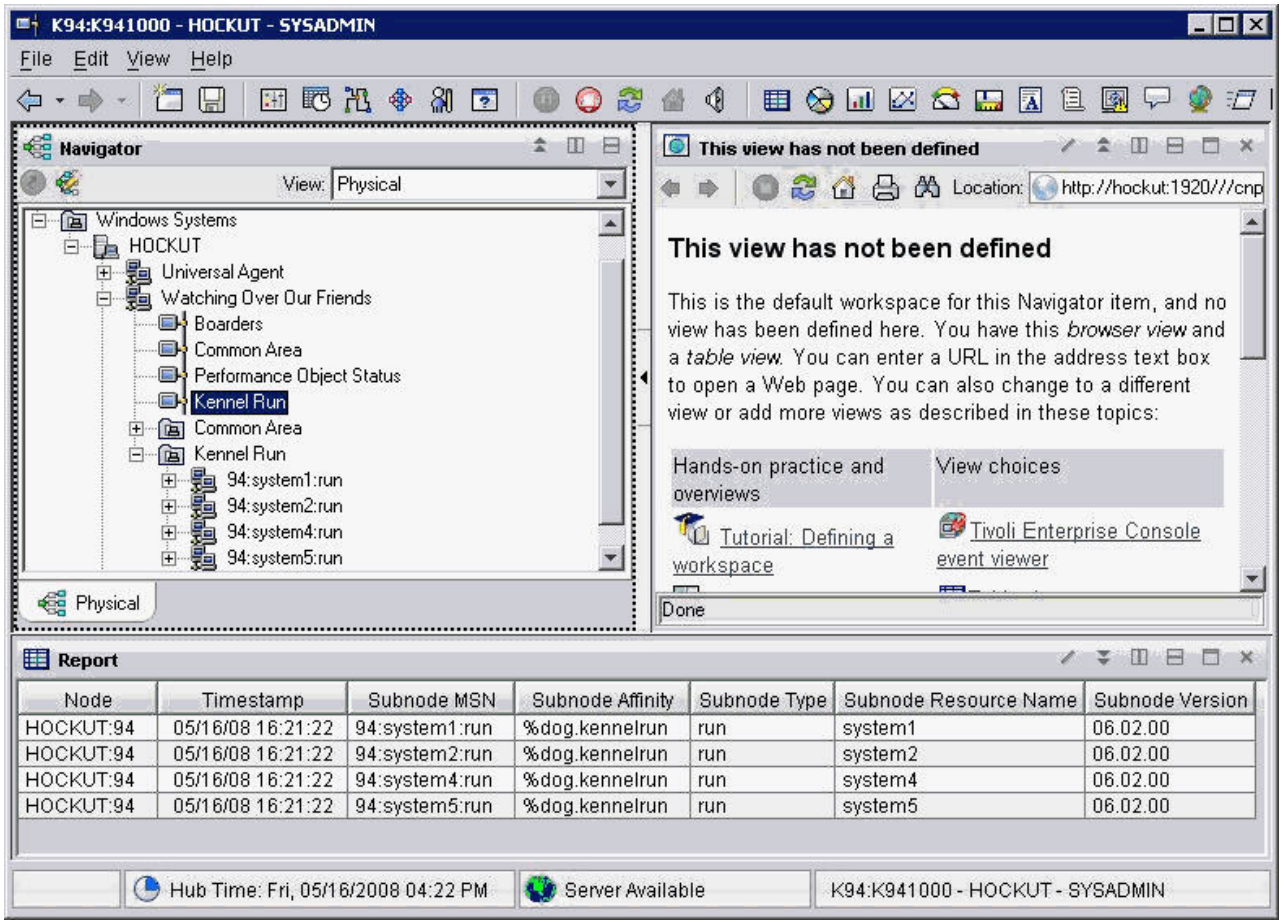


Figure 220. Monitoring multiple subnode instances of the same subnode type

The Performance Object Status subnode contains data visible in the Navigator tree and can have situations that monitor the status of the other data collections.

The example in Figure 221 on page 335 shows a case where the data collection failed (The script was renamed causing it to fail). Typically, any value other than NO_ERROR indicates that there is a problem. For each of the data collectors that are defined in the subnode, there is one row in the table.

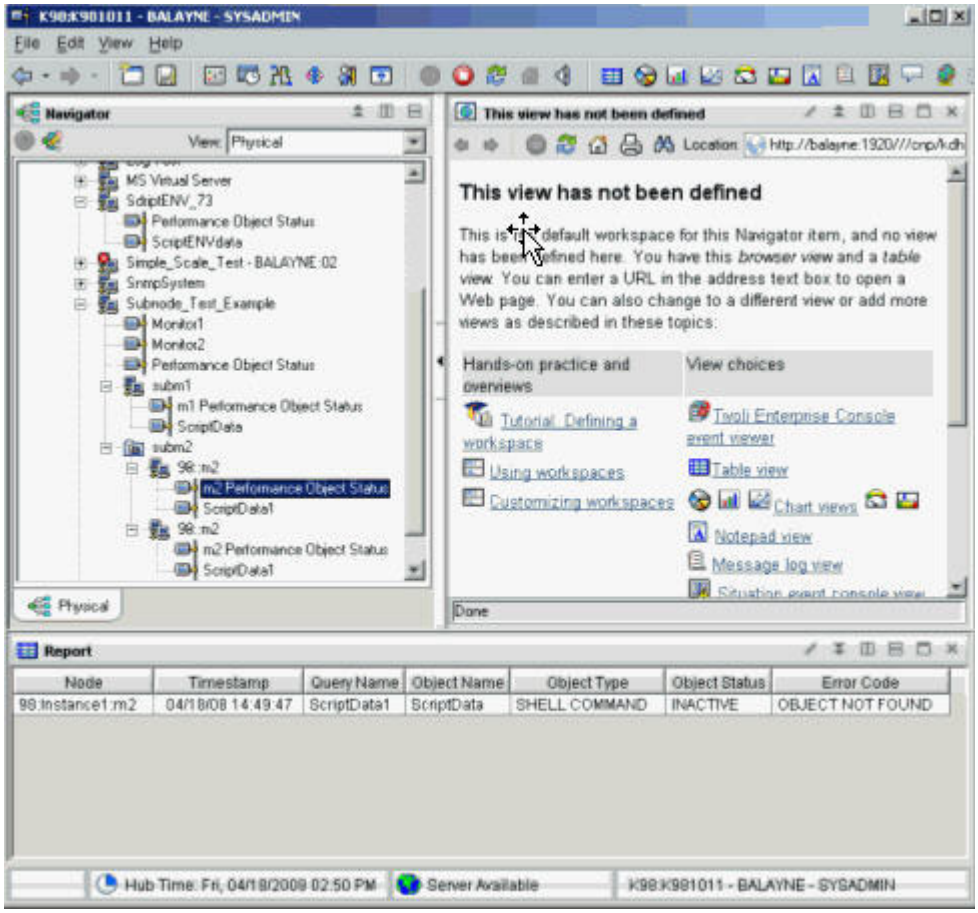


Figure 221. Example: data collection in a subnode

Creating subnodes

Subnodes can be created by using a number of methods.

About this task

You can create a subnode by using one of the following methods:

- "Creating a subnode by using the New Agent wizard"
- "Creating a subnode from the data sources tree" on page 339
- "Creating a subnode in the Agent Editor" on page 343

Creating a subnode by using the New Agent wizard

Use the Data Source Location page in the New Agent wizard to create a subnode.

Procedure

1. On the Data Source Location page, click **Data source groupings** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **A Subnode Definition** .

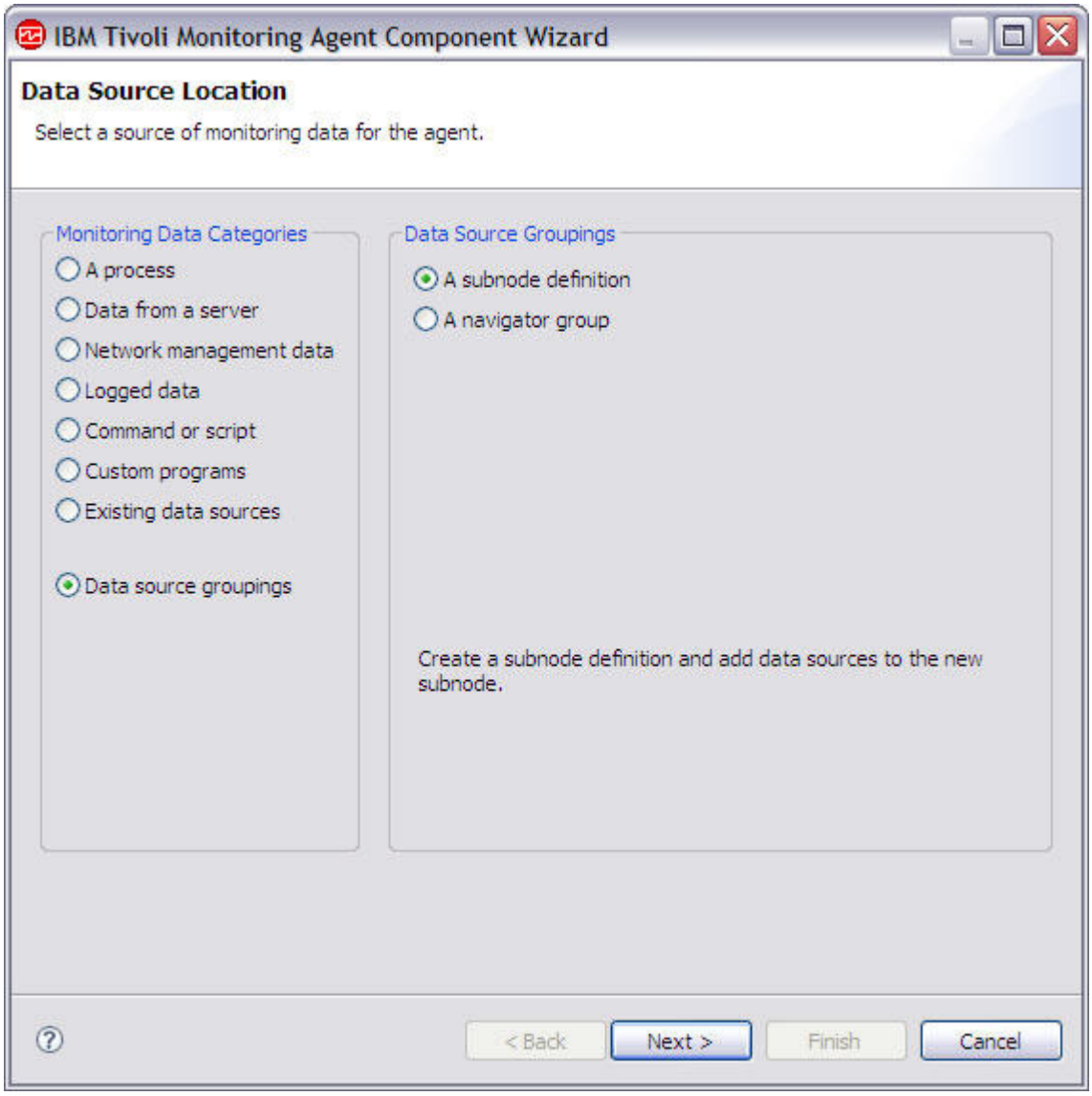


Figure 222. Data Source Location page - Creating a subnode

3. Click **Next**.
4. Complete the Subnode Information page as follows to define the new subnode:
   a. In the **Name** field, type the name of the subnode you are creating.
   b. In the **Type** field, enter 1-3 characters (by using numbers, letters, or both) to identify the type of the subnode you are creating.
   c. In the **Description** field, type a description for the subnode you are creating.
   d. Click the **Show nodes attribute group for this type of subnode** check box to hide or display the attribute group ("Availability node" on page 541).

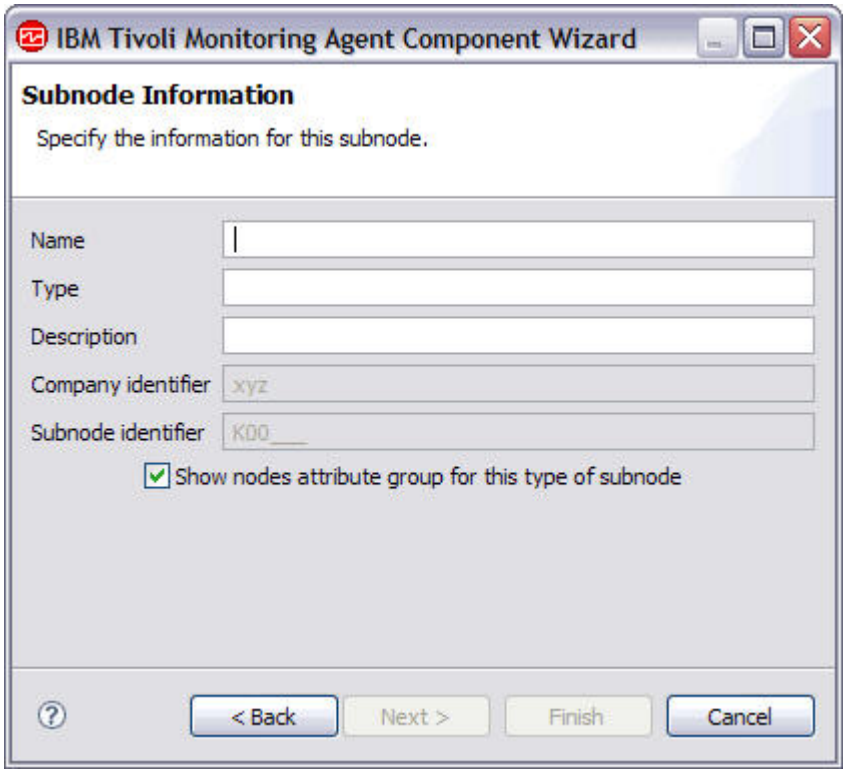


Figure 223. Subnode Information page

e. Click **Next**.

5. Complete the Initial Subnode Data Source Location page as follows to select a data source as the first item in the new subnode:
   a. In the **Monitoring Data Categories** area, click **Data from a server**.
   b. On the **Initial Subnode Data Source** page, select a type of data source for the new subnode, and click **Next**.

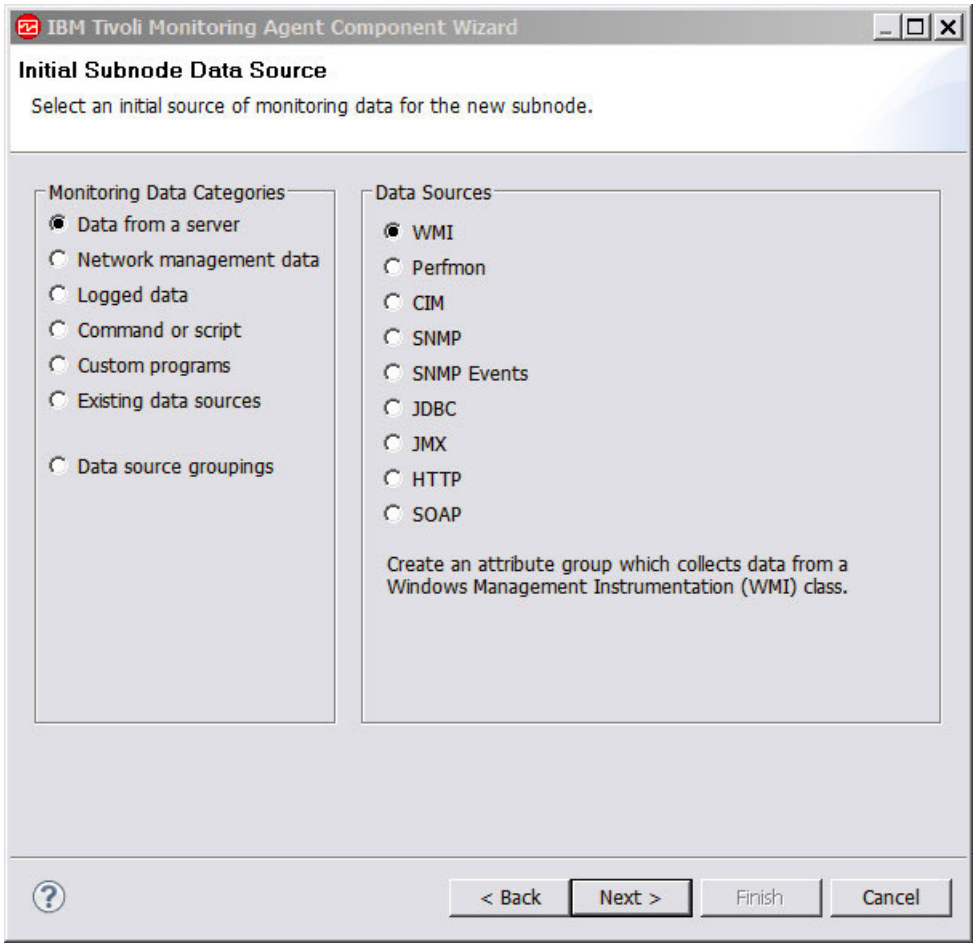


Figure 224. Initial Subnode Data Source page

c. Click **Next**.

6. Define the initial data source for the subnode. See the chapters about specific data sources in this user's guide. Click **Next** after the final data source-specific page.
7. If your agent contains custom configuration properties or if the data source selected requires configuration, use the Subnode Configuration Overrides page to choose the configuration properties. In the Subnode Configuration Overrides page, choose the configuration properties that you want for the subnode at the agent level. Then, choose the configuration properties that you want to vary for each subnode.

   Use **Move**, **Copy**, and **Remove** to specify the configuration properties as described in ("Configuring a subnode" on page 345).

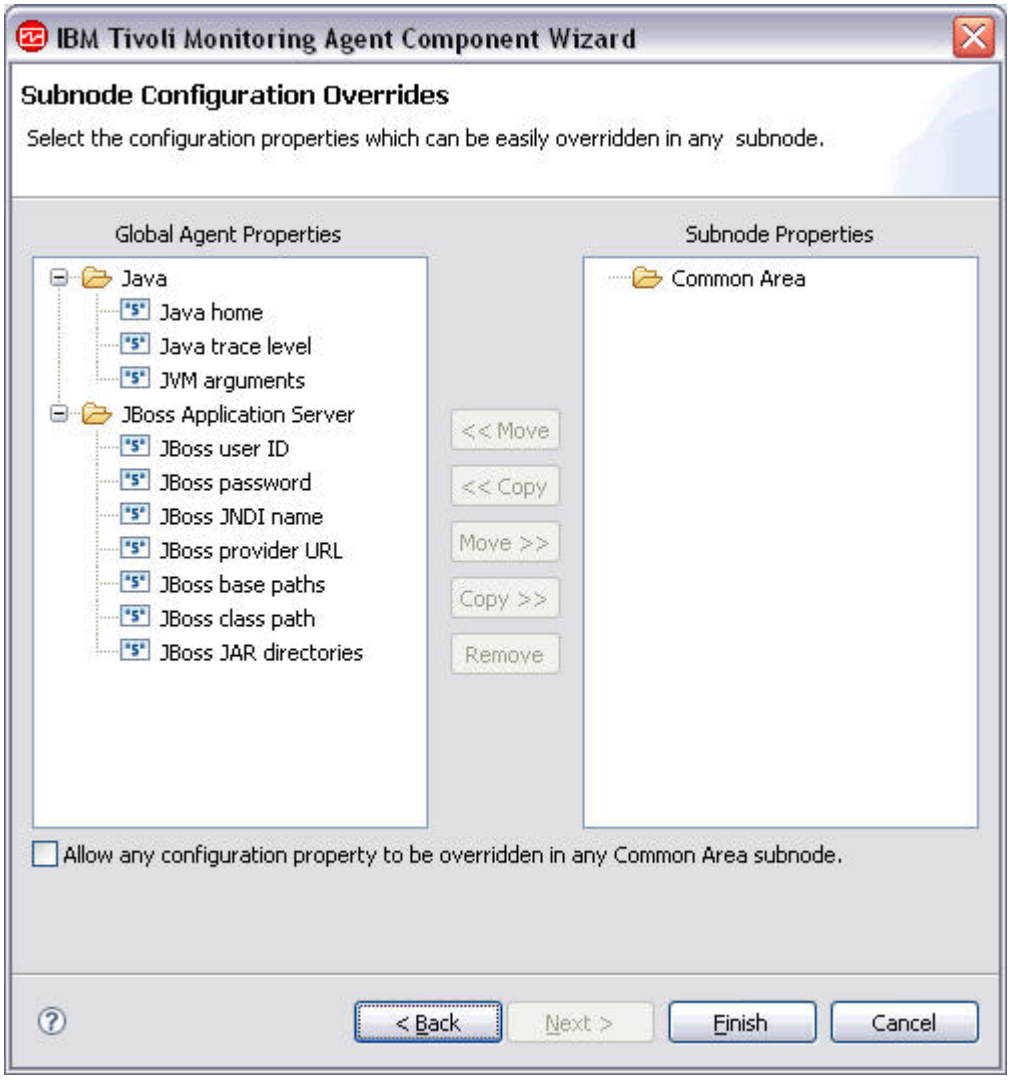


Figure 225. Subnode Configuration Overrides page

8. Click **Next**. The Data Source Definition page is displayed.

Creating a subnode from the data sources tree

Use the data sources tree in the New Agent wizard to create a subnode

Procedure

1. On the Data Source Definition page, right-click an agent name in the Attribute Group Information pane. To create a subnode definition click **Add Subnode Definition**.

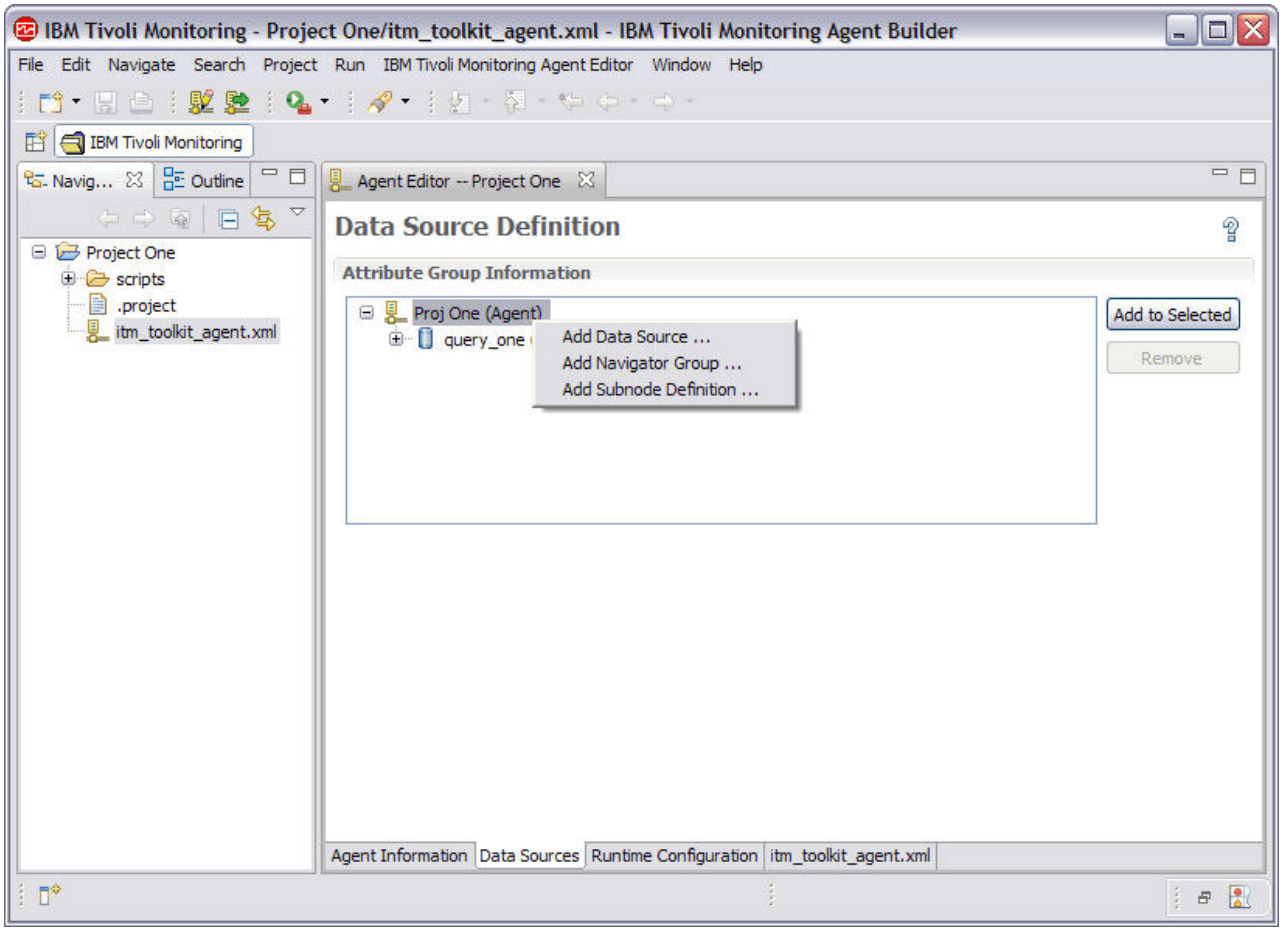


Figure 226. Data Source Definition page

2. Enter information about the new subnode definition in the wizard and then click **Next**.

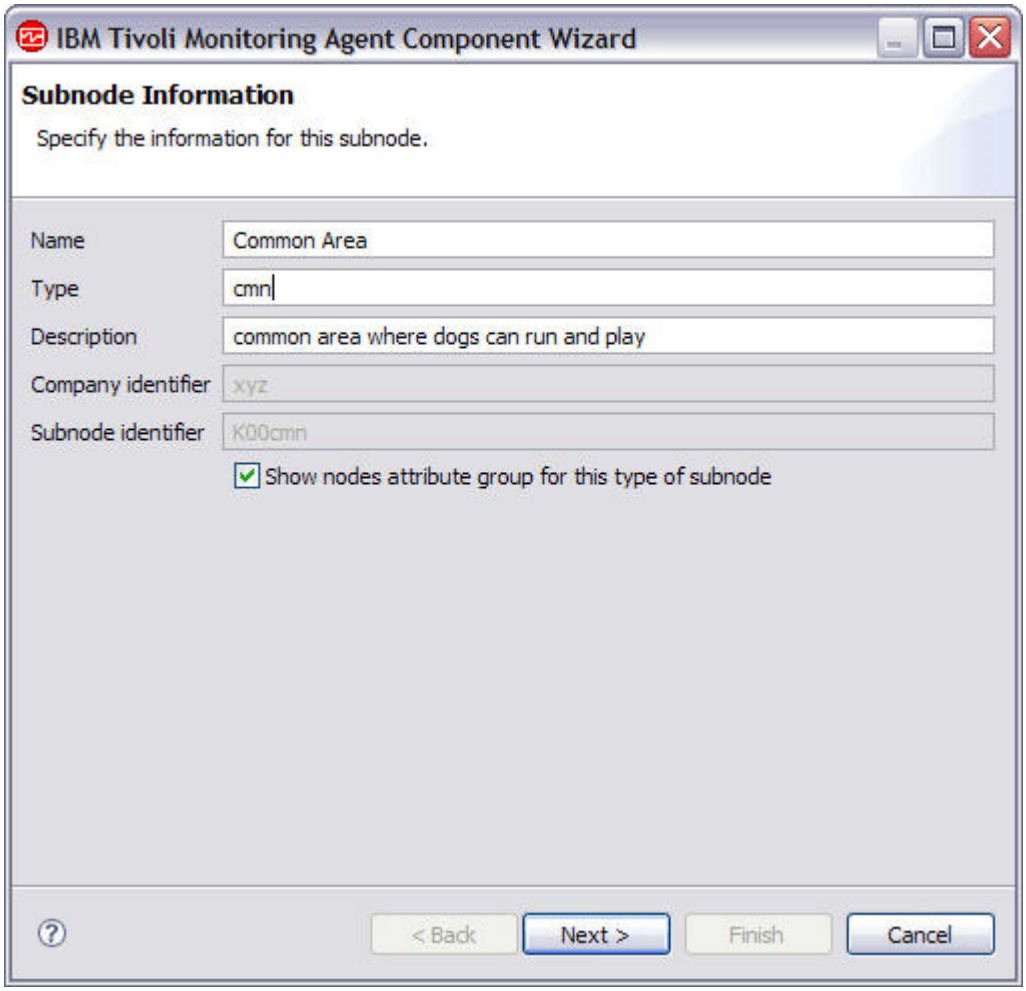


Figure 227. Defining a new subnode

3. On the Initial Subnode Data Source page, select a type of data source for the new subnode, and then click **Next**.

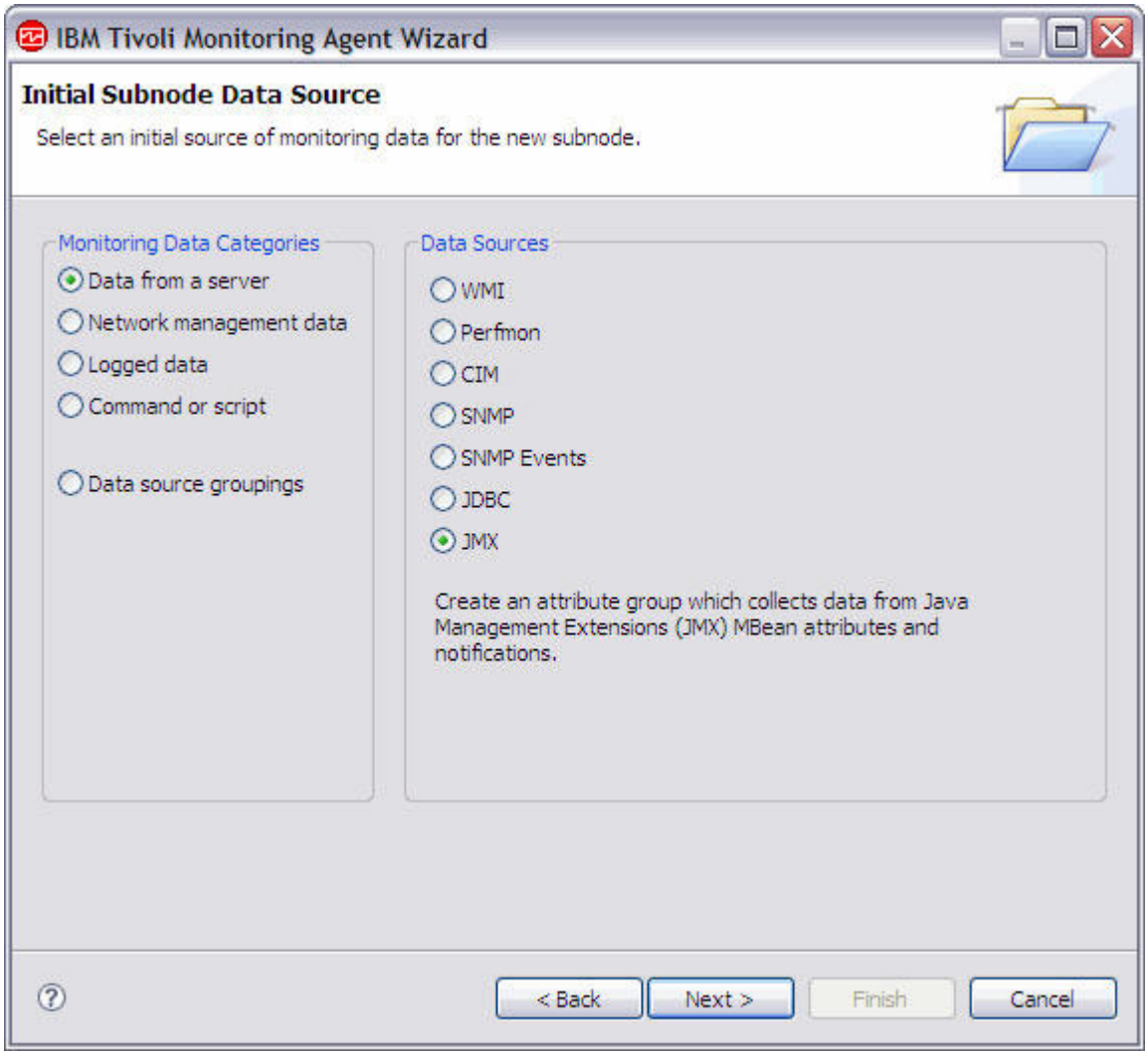


Figure 228. Initial Subnode Data Source page

4. Define the initial data source for the subnode. See the chapters about specific data sources in this user's guide. Then, click **Next** after you complete the final data source-specific page.
5. On the Subnode Configuration Overrides page, identify which configuration properties are to remain at the agent level and which ones to vary for each subnode instance.

   Note: Properties in the **Global Agent Properties** list are configured at the agent level. Properties in the **Subnode Properties** list can be configured for each subnode. Properties can be in both lists. If the subnode property is set, it overrides the global property for that subnode. Some properties must be configured at the agent level and cannot be moved to the subnode level. For SNMP, **SNMP Version** cannot be copied or moved under **Subnode Properties**.

   a. Use **Move**, **Copy**, and **Remove** to specify the configuration properties as described in ("Configuring a subnode" on page 345).

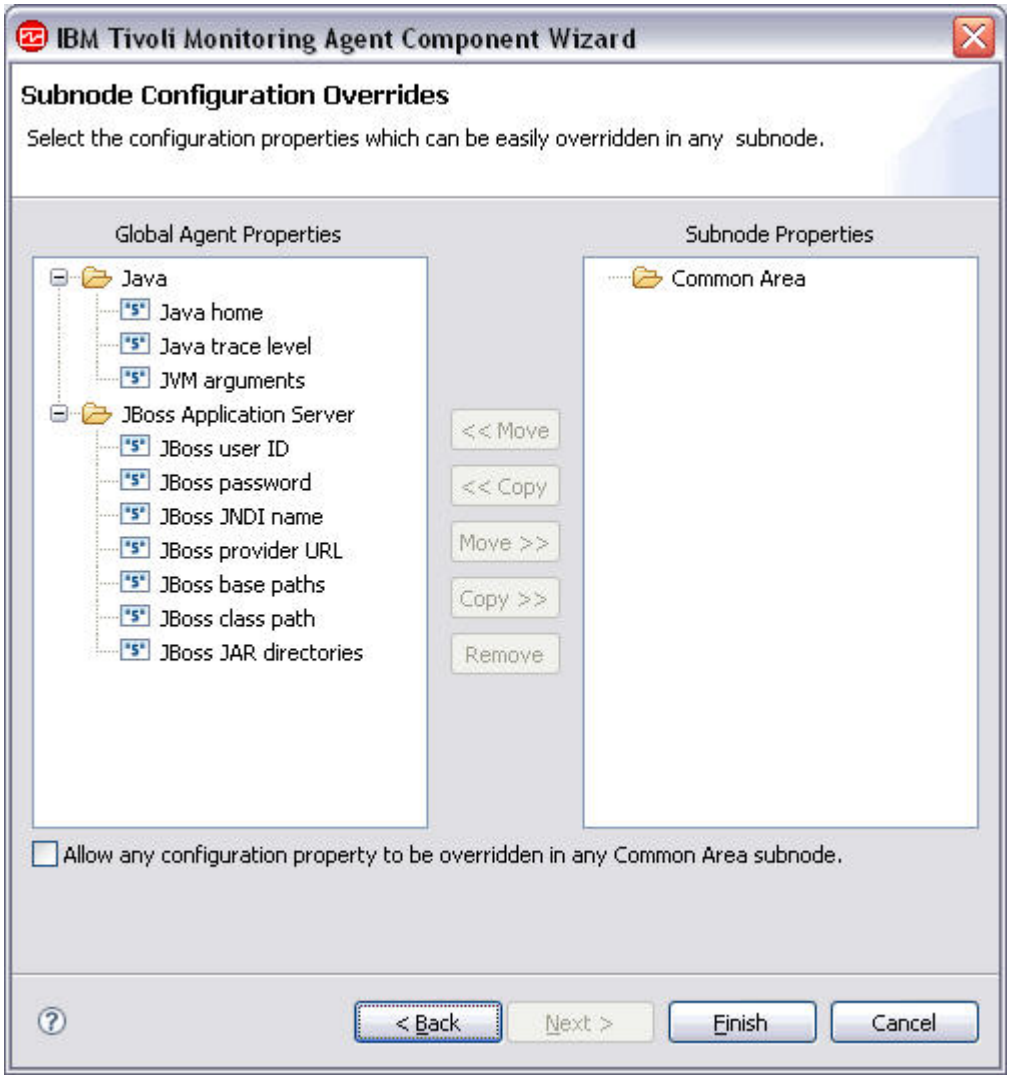


Figure 229. Subnode Configuration Overrides page

6. When finished identifying the properties, click **Finish**. The Data Source Definition widow is displayed.

Creating a subnode in the Agent Editor

Use the Agent Editor to create a subnode

Procedure

1. Start the Agent Editor and click the **Data Sources** tab.
2. Select the agent node, and click **Add to Selected**, or right-click on the agent node and click **Add Subnode Definition** from the menu.

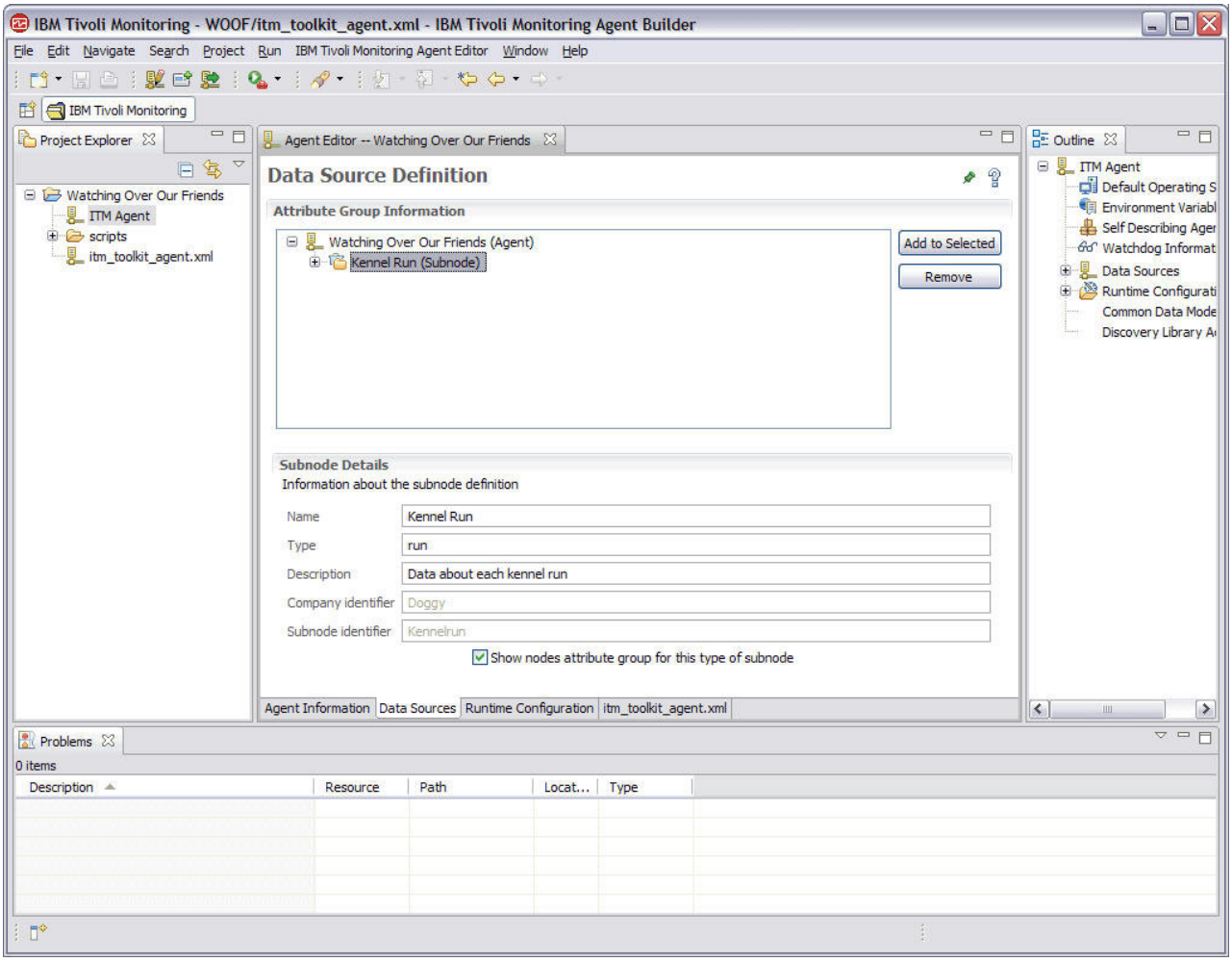


Figure 230. Agent Editor Data Source Definition page

Subnode configuration

When a subnode type is defined, a single configuration section is defined specifically for that subnode.

There are several ways that a subnode configuration section differs from other configuration sections:

- The set of properties in a subnode section can be duplicated, so there are multiple sets of properties. Each set of properties forms its own section. The layout of all sections is identical, but different values can be entered in each section.

  In contrast, the properties in other sections (which are referred to as agent-level sections) are shown only one time during runtime configuration. They do not form subsections and cannot be duplicated or removed.

  See “Subnode configuration example” on page 352 for GUI and command-line examples of configuring subnodes.

-

  For each copy of a subnode section that is created at runtime configuration, the agent creates a separate subnode instance. All of those subnode instances are of the same type.

- The property names in subnode sections can be duplicates of property names in agent-level sections. When duplicate names occur, the subnode property value overrides the agent-level property value.

- In IBM Tivoli Monitoring V6.2.1 and later, a subnode section can have default property values that apply to all instances of subnodes of that type. This feature makes it possible to have a three-level lookup of a single property value as follows:
  1. The agent obtains the property value from the subnode instance subsection.
  2. If no value is configured at the subnode instance level, the property value is obtained from the subnode default level.
  3. If no value is configured at either of those two levels, then the property value is obtained from an agent-level section.

See "Subnode configuration example" on page 352 for GUI and command-line examples of configuring subnodes.

Configuring a subnode

Use the **Subnode Configuration Overrides** page to configure a subnode data source.

Before you begin

Use the steps in "Creating subnodes" on page 335 to create a subnode.

About this task

When you add a data source to a subnode, the Subnode Configuration Overrides page is presented if the data source requires configuration. It shows custom configuration properties and any other configuration properties that are applicable to the subnode type.

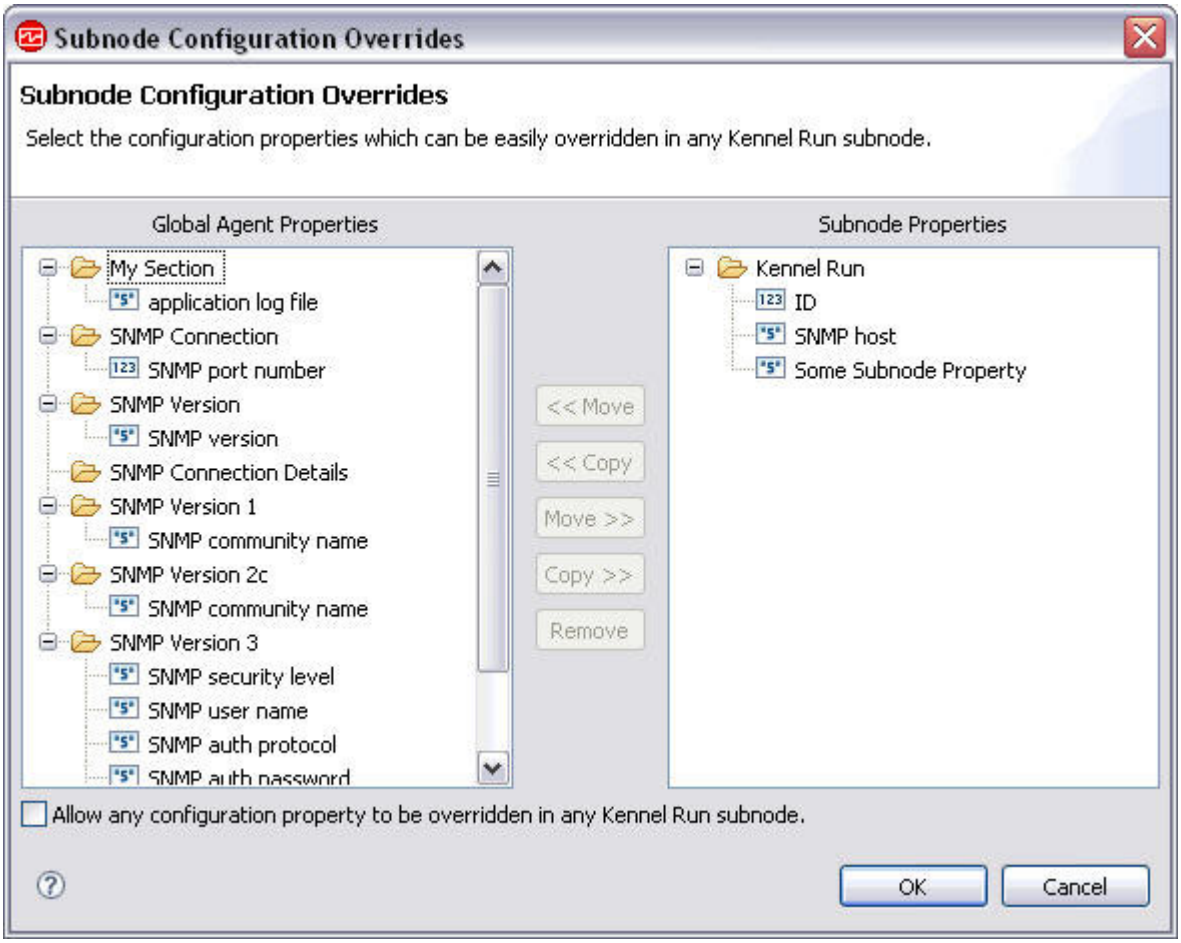


Figure 231. Subnode Configuration Overrides window

Procedure

In the following example, SNMP is the data source for the subnode that is being configured.

- In the Subnode Configuration Overrides window, choose the configuration properties that you want for the subnode at the agent level. Also, choose the configuration properties that you want to vary for each subnode.
- Use **Copy >>** to copy configuration properties so that they are both at the agent level and the subnode level. The agent looks for a value first at the subnode level, and if it does not find a value, it looks at the agent level. If a property at both levels is a required property, it is required only at the agent level, it is optional at the subnode level.
- Use **Move >>** to move properties from the agent level to the subnode level. **Move>>** is not available for properties that are required by an agent-level data source or by a subnode of a different type.
- Use **Remove** to remove one of the two lists. Properties can be removed only if they are listed at both the agent-level and the subnode level. This function cannot be used to remove a property completely.
- Use **<< Copy** to copy a property from the subnode level to the agent-level.
- Use **<< Move** to move a property from the subnode to the agent-level.

What to do next

You can change the configuration for an existing subnode by using the Agent Editor.

Subnode configuration overrides

Use Subnode Configuration Overrides to override agent configuration properties with subnode-specific properties.

The example in (“Configuring a subnode” on page 345) describes how to manage subnode configuration for automatically generated properties. Managing custom configuration properties is similar. Any custom configuration properties that are defined are displayed in the **Subnode Configuration Overrides** window.

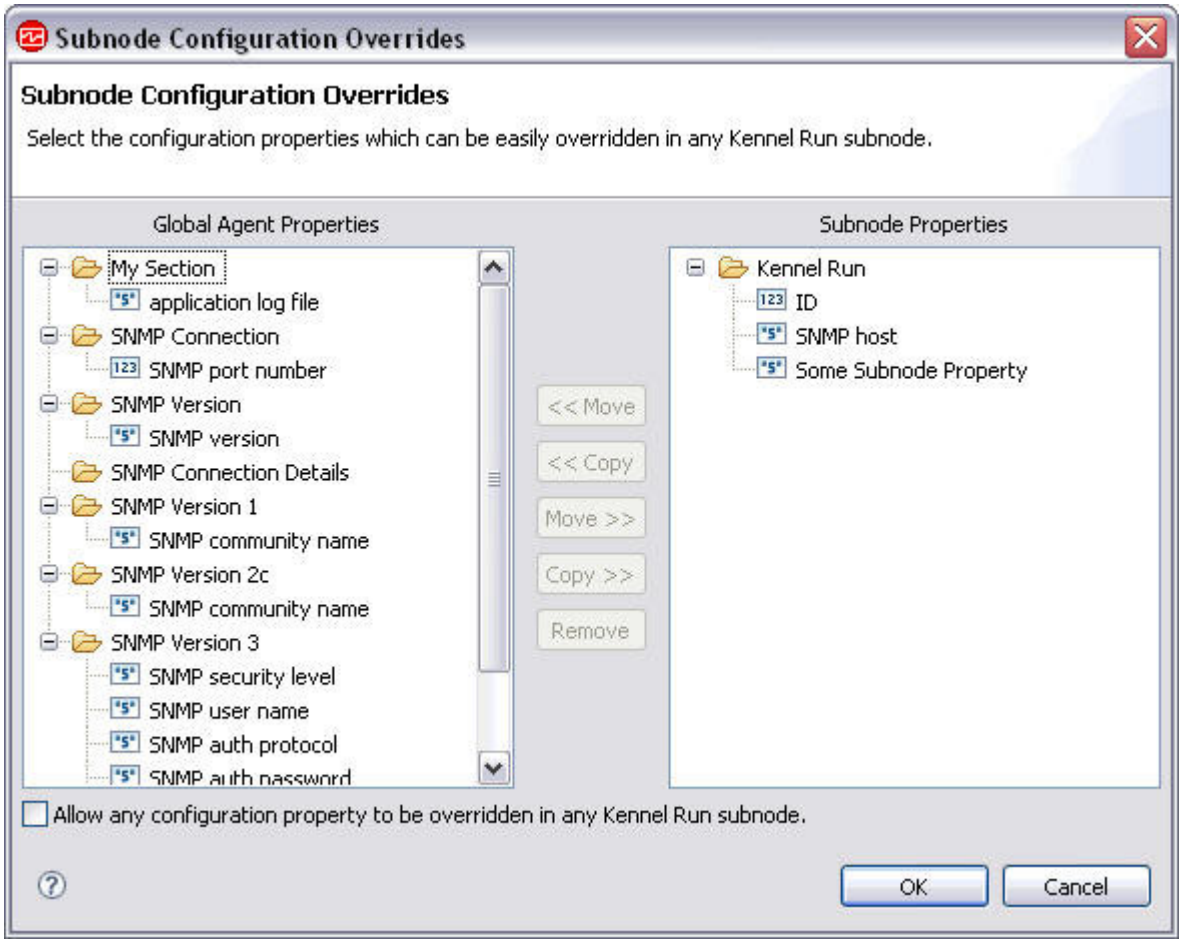


Figure 232. Overriding custom configuration

When you copy or move a custom property from the subnode level to the agent-level, you are prompted for the section to place the property in. As shown in the (Figure 233 on page 348). You can select an existing custom section, or enter the name of a new custom section.

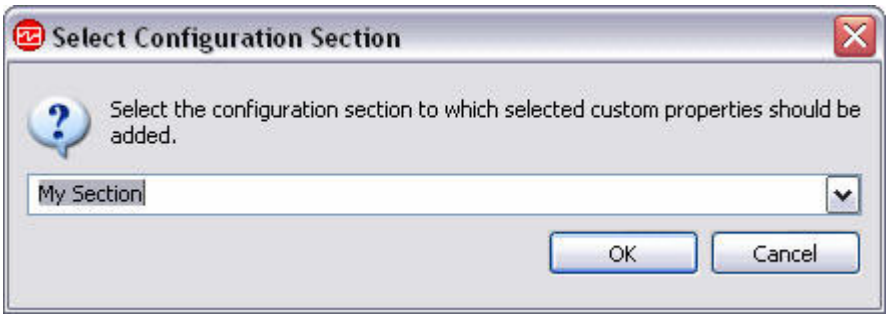


Figure 233. Select Configuration Section window

Selecting subnode configuration properties

Without subnodes, all instances of a data source type share the configuration parameters. For example, all SNMP attribute groups connect to the same host by using the same community name. With subnodes, each instance of a subnode can connect to a different host if the `SNMP_HOST` property is placed at the subnode level.

Selecting properties to be overridden at the subnode level is an important consideration when you are developing an agent. If too many properties are selected, the subnode configuration section becomes cluttered and difficult to manage. If too few properties are selected, then the agent functions might be limited when someone wants to vary a property from one subnode to the next.

The following properties cannot be copied to the subnode level. (All attribute groups in all subnodes and in the base agent must use the same SNMP version and JMX connection type):

- SNMP version
- JMX MBean server connection type
- Java home
- Java trace level
- JVM arguments
- Class path for external JAR files
- Socket data source port number
- JMX or JDBC class path settings

Advanced subnode configuration

Use advanced subnode configuration to override an agent configuration property in a subnode.

About this task

There is an option in IBM Tivoli Monitoring V6.2.1 and later agents that you can enable to override properties from any agent-level configuration section in a subnode instance. On the Subnode Configuration Overrides page, there is a check box labeled **Allow any configuration property to be overridden in any subnode**. For more information, see ("Subnode configuration overrides" on page 347). For this option to be enabled, you must select **6.2.1** as the **Minimum ITM version** when you name your agent ("Naming your agent" on page 19). If you choose this option, each subnode instance can override any property from any agent-level configuration section. But this property can be overridden only from the GUI and not from the **`itmcmd`** command line.

Procedure

The **Allow any configuration property to be overridden in any subnode** option causes an **Advanced** field that contains a list to be displayed on each subnode configuration panel (Figure 234 on page 349).

The initial selection in the **Advanced** field provides the brief directions: **Select a section to override values**.

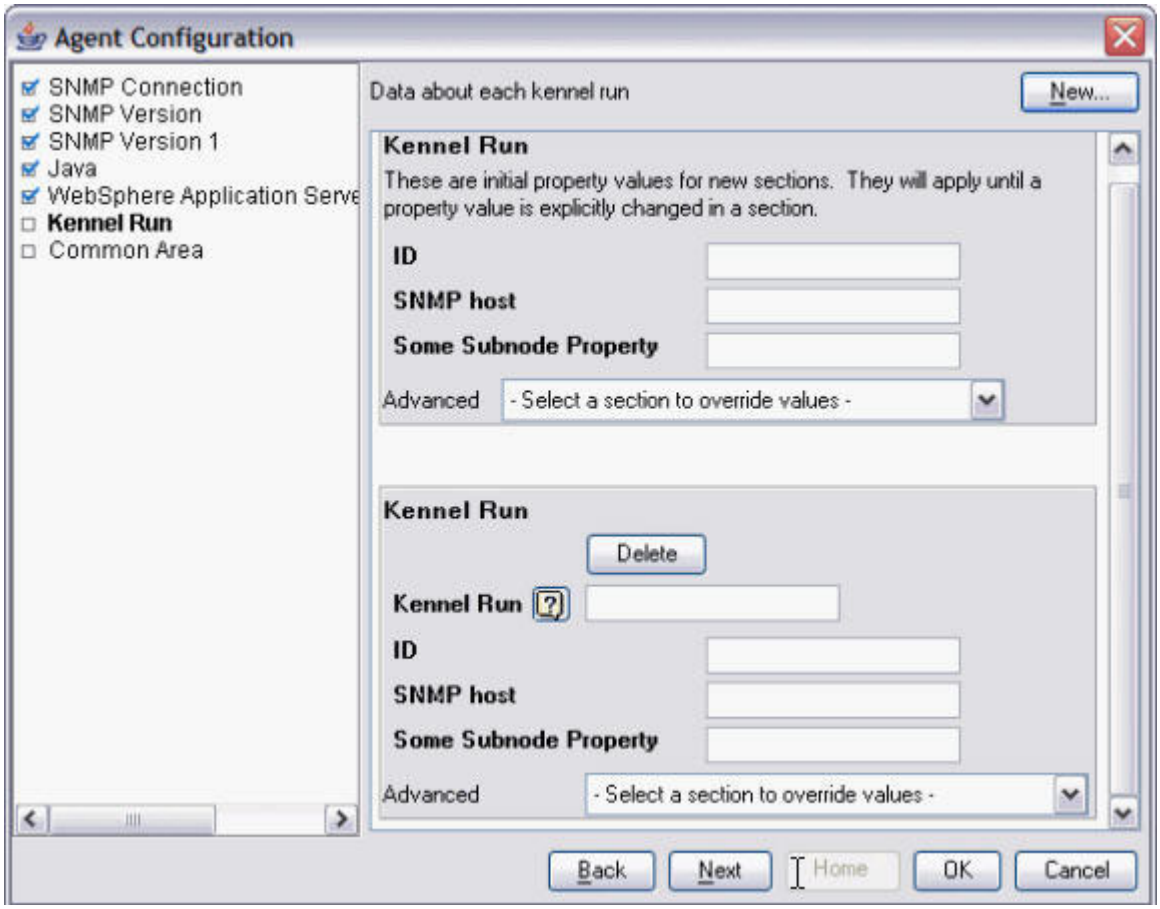


Figure 234. Override configuration lists

- When you click the list, you see a list of all the non-subnode sections that contain configuration properties (Figure 235 on page 350).

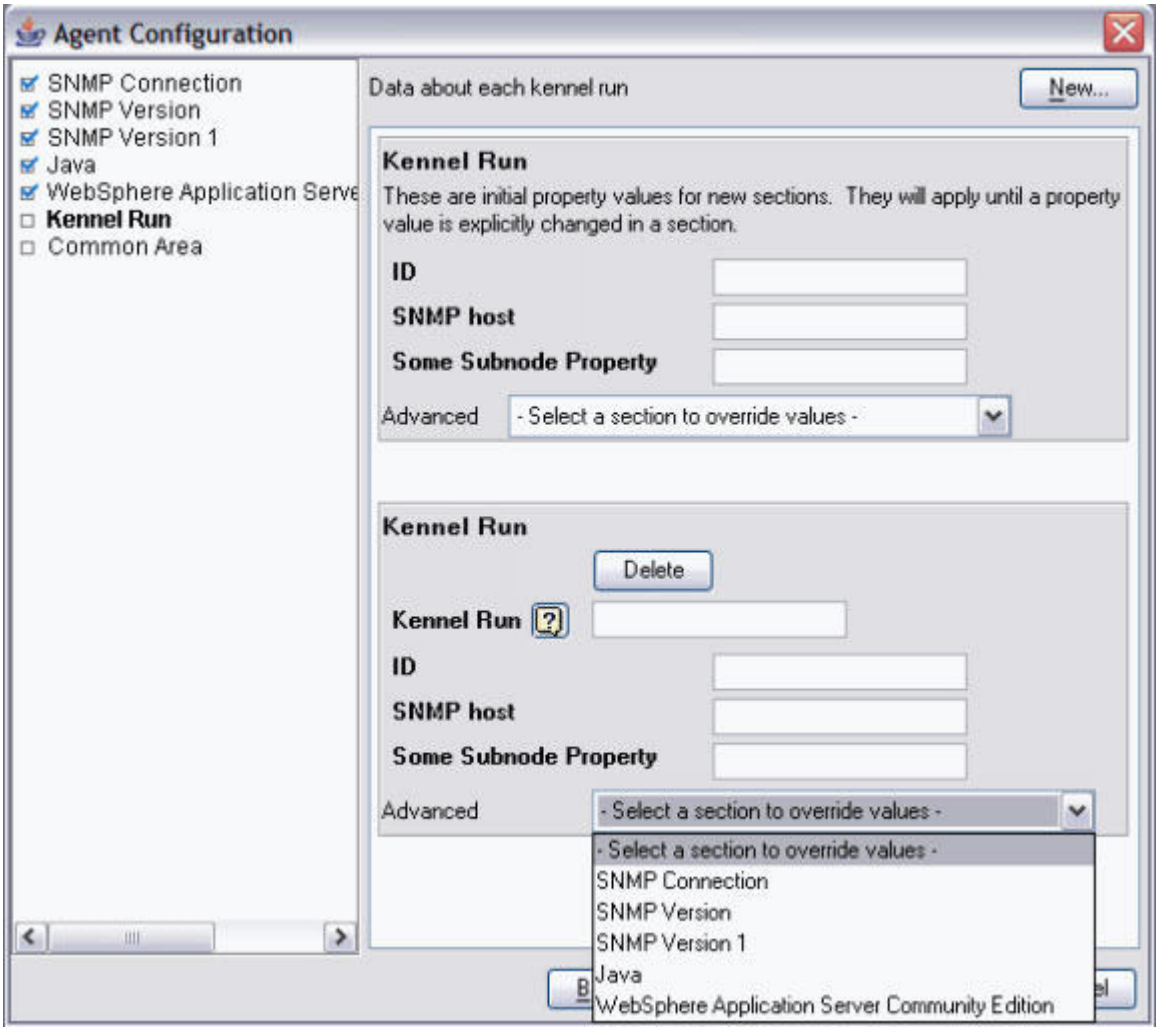


Figure 235. Sections to override

- Select a section. The properties from that section are temporarily added to the subnode panel. The value of any property that you change is added to the set of properties that are defined for the subnode. A data source in the subnode looks for property values in the subnode before it looks in the agent-level sections. See (Figure 236 on page 351).

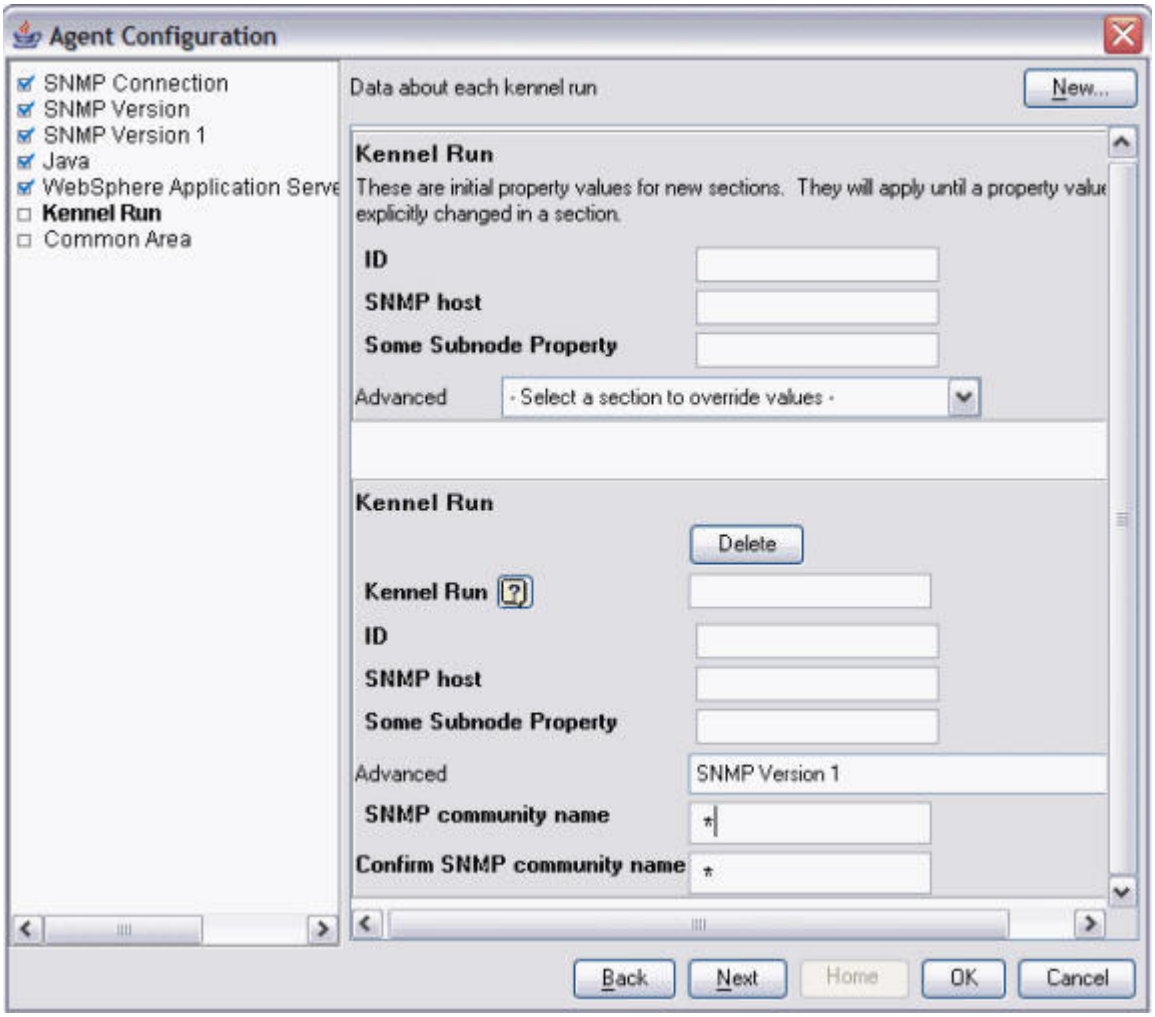


Figure 236. SNMP Version 1 Properties expanded

The following further information applies to overriding properties from agent-level sections:

- Properties that were copied to the subnode section are not shown when the agent-level section is selected in the **Advanced** list. For example, in (Figure 236), SNMP host is not displayed after the **Advanced** list because it was copied to the subnode properties and is already displayed.
- Sections that contain no properties to override do not have a selection in the **Advanced** list.
- Overridden values that you enter for one section are retained even if you select a different section to display different properties.
- Select **Allow any configuration property to be overridden in any subnode** to enable this feature in your agent. The minimum IBM Tivoli Monitoring version of the agent must be 6.2.1 or later. If it is set to 6.2, it is changed to 6.2.1, although you are prompted before the change is made. If you do not want to change the minimum version, you can cancel the setting at the prompt. If you set the minimum IBM Tivoli Monitoring version to 6.2.1, you cannot install on a version earlier than 6.2.1. (Figure 9 on page 22).

Configuring a subnode from the command line

You can also configure a subnode by using the command line.

Before you begin

For more information about subnode configuration, see “Subnode configuration” on page 344

About this task

Procedure

To configure a subnode instance from the command line, use the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p
section_name:subnode_instance_id.property_name=value
```

Where:

section_name
Same as the subnode type

subnode_instance_id
ID for the subnode that is defined during configuration.

property_name
Name of the configuration property

value
Value for the property

Subnode configuration example

How to configure a sample agent with one defined subnode.

Example:

This example shows how to configure a sample agent that has one subnode named Example Subnode of type exs and the following three configuration properties:

- `Agent Cfg` (actual property name is `K00_AGENT_CFG`) is defined only at the agent level.
- `Subnode Cfg` (actual property name is `K00_SUBNODE_CFG`) is defined only in the example subnode.
- `Overridable Cfg` (actual property name is `K00_OVERRIDABLE_CFG`) is defined at the agent level and was copied to the example subnode.

(Figure 237 on page 353) shows these configuration properties on the Runtime Configuration Information page of the Agent Editor.

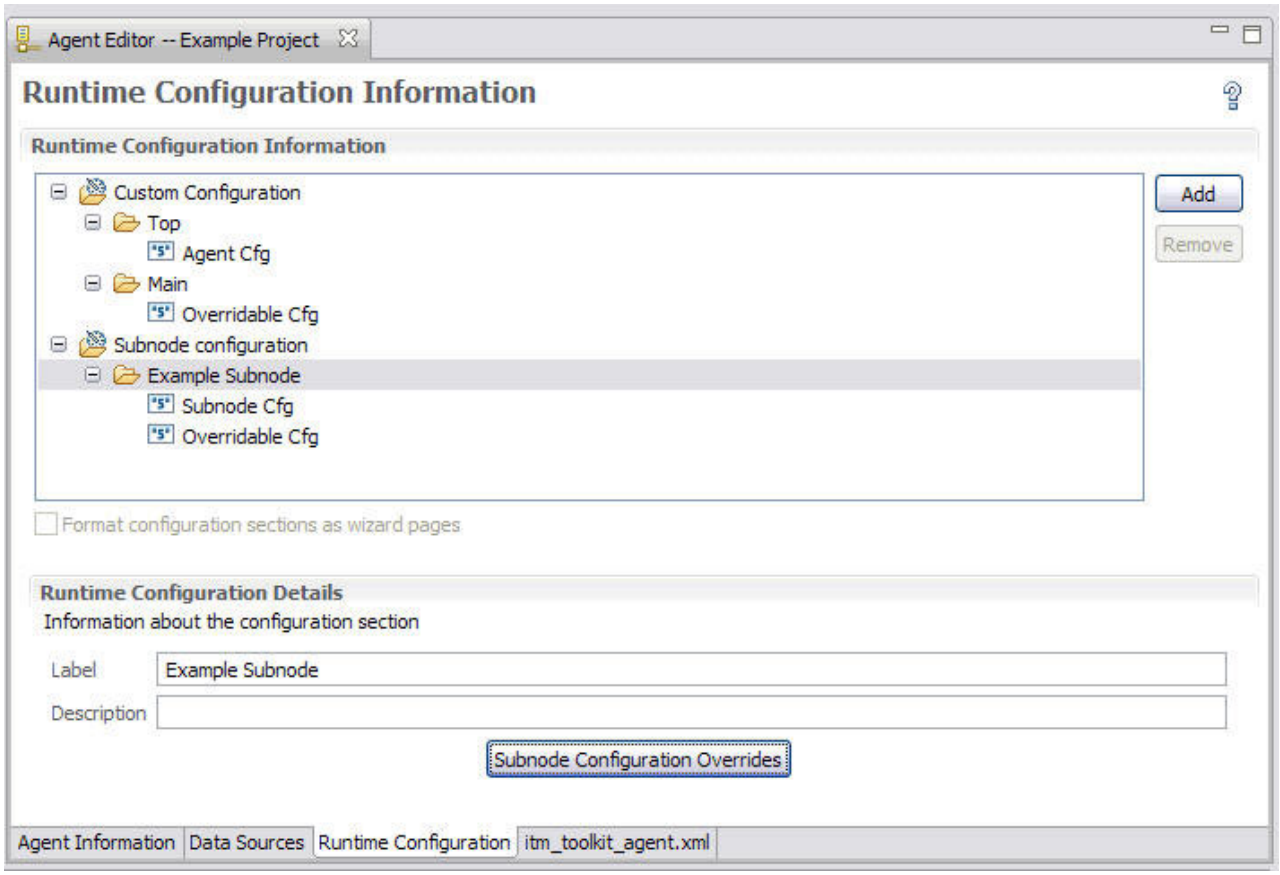


Figure 237. Configuration property definitions in the Agent Builder

When this example agent is configured, the first page that is displayed is the **Top** section, which contains the **Agent Cfg** property as shown in (Figure 238 on page 354). Because this property is an agent-level property, it is shown one time during agent configuration. Any instance of the Example Subnode can see this property value, but all instances see the same value.

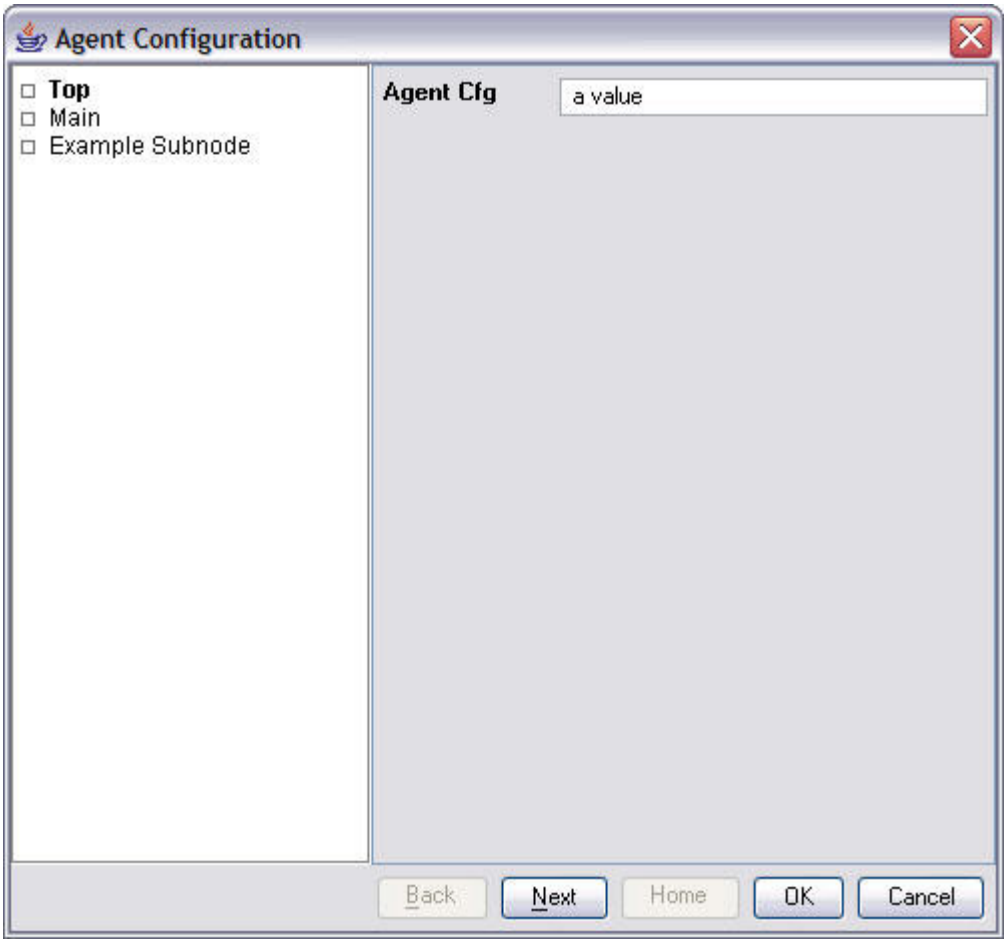


Figure 238. **Top** *section with agent-level configuration for the* **Agent Cfg** *property*

If you are configuring from the Tivoli Enterprise Monitoring Server command line, the **Agent Cfg** property can be set by using the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "TOP.K00_AGENT_CFG=a value"
```

The next section that is displayed is the **Main** section as shown in (Figure 239 on page 355). It is also an agent-level section and contains the agent-level **Overridable Cfg** property. This property differs from the **Agent Cfg** property because this property was copied to the Example Subnode in the Agent Builder. This means that a default value for the property can be entered on the Main page. However, any Example Subnode instance can override the value that is entered here with a different value.

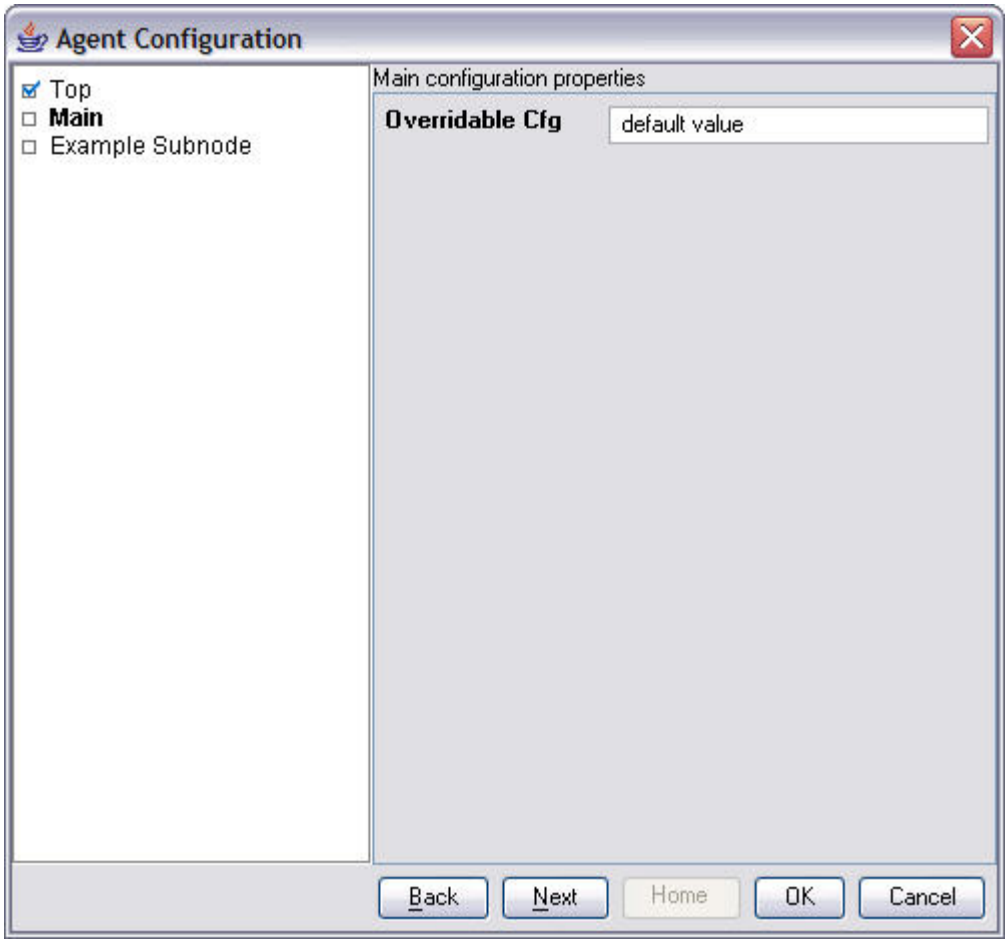


Figure 239. **Main** *section with the agent-wide default value for the* **Overridable Cfg** *property*

If you are configuring from the Tivoli Enterprise Monitoring Server command line, this property can be set by using the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "MAIN.K00_OVERRIDABLE_CFG=default value"
```

You can place both of these properties in the same agent-level section. You can decide how many custom agent-level sections to create and how to distribute custom properties among them.

The next section that is displayed is the **Example Subnode** section as shown in (Figure 240 on page 356). Because this agent is being configured for the first time, there are no defined subnode instances and no subnode instance subsections are shown. The initial property values subsection is shown, although it is optional and some subnode types might not show it. Because the initial property values subsection is shown, default values can be entered for any of the configuration properties. The **Overridable Cfg** property already has a default value that was obtained from the agent-level property of the same name.

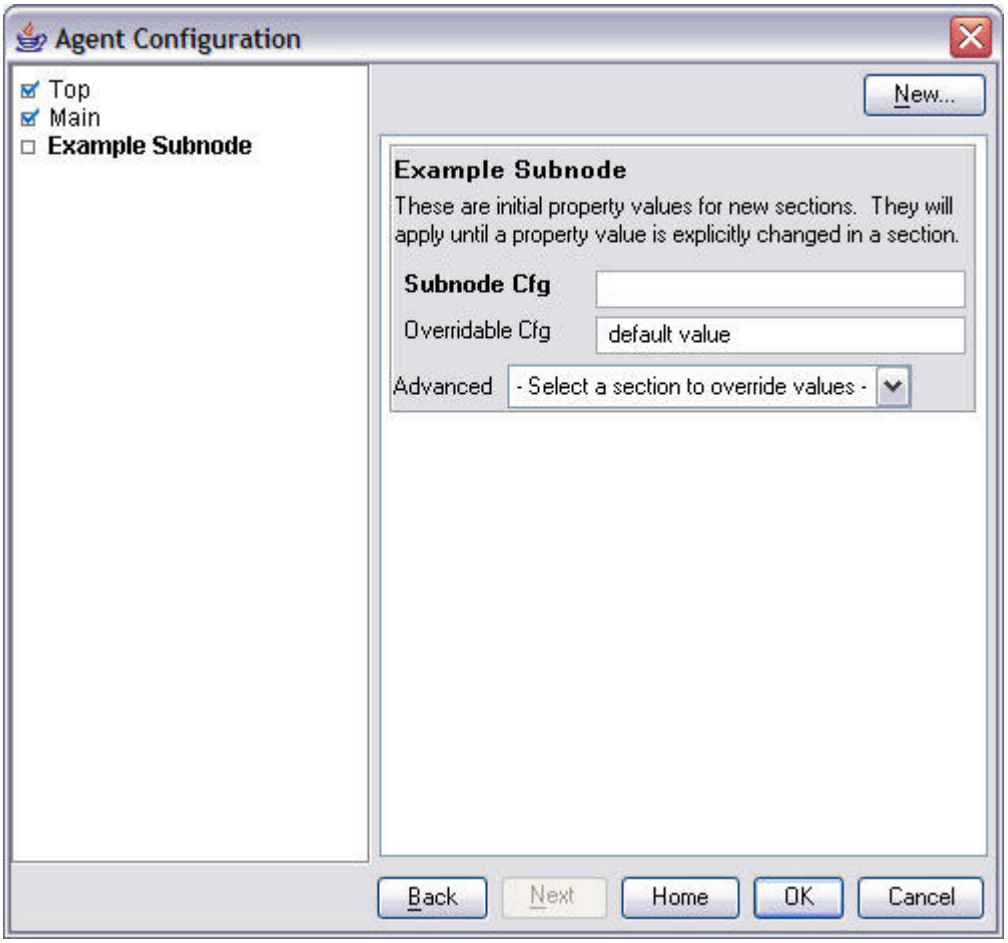


Figure 240. **Example Subnode** *section page with no subnode*

Subnode instances are defined by doing the following actions on the empty **Example Subnode** section page (Figure 241 on page 357):

1. In the initial **Example Subnode** section, in the **Subnode Cfg** field, type the following default string for the property: `sub-default value`.
2. Click **New**. An **Example Subnode** subsection is displayed after the initial properties subsection.
3. In the **Example Subnode** field, type the following subnode instance ID: `do`.
4. Click **New**. A second **Example Subnode** subsection is shown after the first.
5. In the second **Example Subnode** field, type the following subnode instance ID: `re`.
6. In the **Subnode Cfg** field, type the following value for the **Subnode Cfg** property: `sc override`.
7. In the **Overridable Cfg** field, type the following value for the **Overridable Cfg** property: `oc override`.

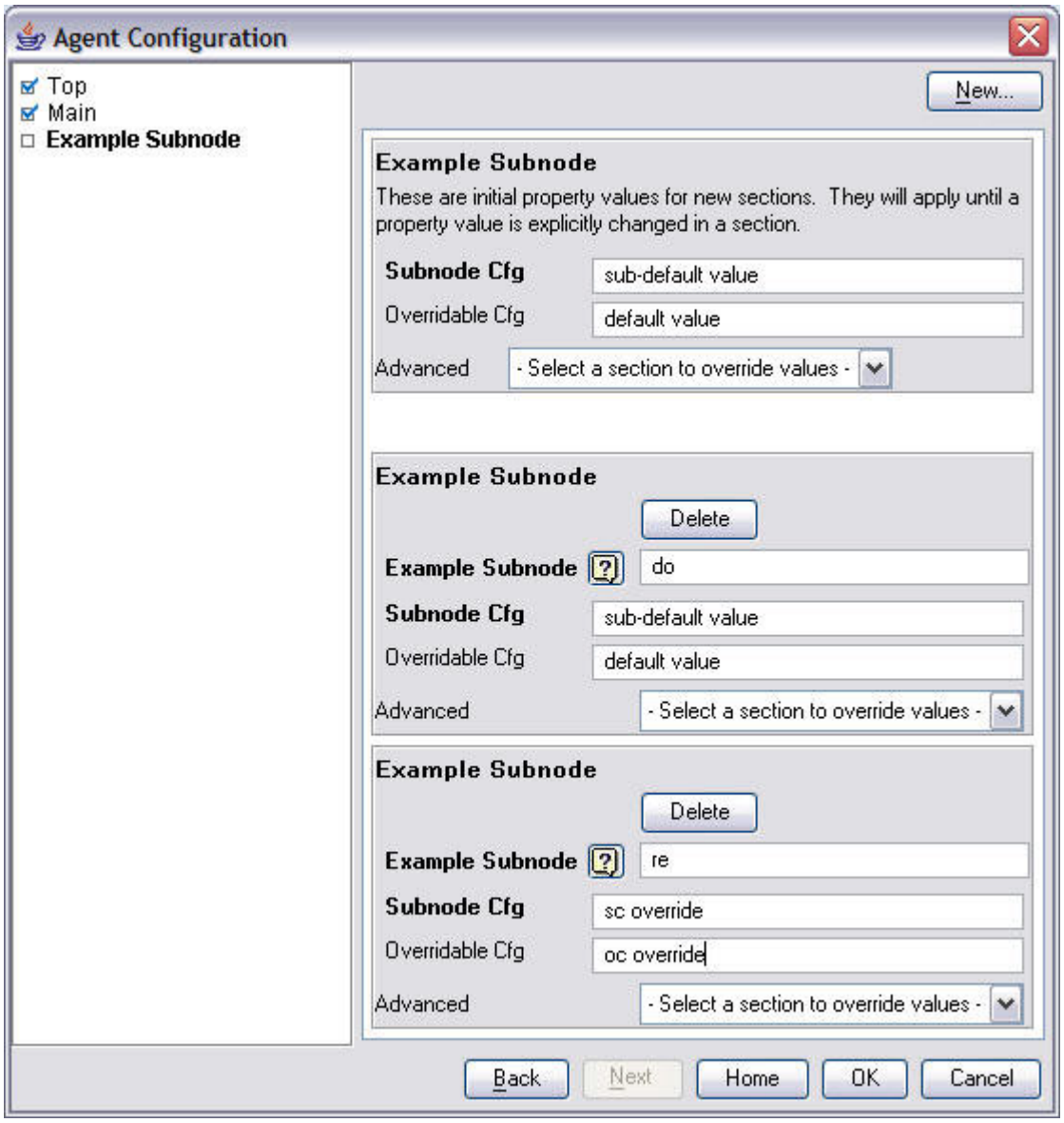


Figure 241. **Example Subnode** *section page with two subnode instances defined*

The two new subsections cause the agent to create two subnode instances when it is started. Because the properties of the **do** subnode subsection were not changed, the default property values are used by that subnode instance. Since different values were entered for the properties in the **re** subsection, the **re** subnode instance uses those values that were typed.

You can set a default value from the Tivoli Enterprise Monitoring Server command line with the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "exs.K00_SUBNODE_CFG=sub-default value"
```

The format for setting subnode default values is exactly like the format for setting agent-level properties, except that the section name identifies a subnode section.

You can create the subnode instances from the Tivoli Enterprise Monitoring Server command line with the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "exs:do.K00_OVERRIDABLE_CFG=default value" \
 "exs:re.K00_SUBNODE_CFG=sc override" "exs:re.K00_OVERRIDABLE_CFG=oc override"
```

The subnode instance ID is inserted between the section name and property name. When you use the command line to create a subnode instance, at least one property must be specified, even if all the properties use default values. Otherwise, default values are not required to be specified on the command line when you define subnode instances.

All of the agent configuration properties can be set in a single command. The following command is equivalent to all of the preceding individual commands:

```
tacmd configureSystem -m HOSTNAME:00 -p "TOP.K00_AGENT_CFG=a value" \
 "MAIN.K00_OVERRIDABLE_CFG=default value" \
 "exs.K00_SUBNODE_CFG=sub-default value" \
 "exs:do.K00_OVERRIDABLE_CFG=default value" \
 "exs:re.K00_SUBNODE_CFG=sc override" "exs:re.K00_OVERRIDABLE_CFG=oc override"
```

Subnodes and Windows data sources

Choose to include Windows Remote Connection properties in the agent or not.

About this task

If an agent has Windows data sources at the agent level and not in subnodes, including Windows Remote Connection configuration properties in the agent are optional. Windows data sources are Windows Event Log, Windows Management Instrumentation, Windows Performance Monitor. If configuration properties are not included, these data sources monitor the local Windows system by default and need no configuration. By default, no Windows data sources are included in any subnode.

To choose whether to include Windows Remote Connection properties in the agent, do the following steps:

Procedure

1. On the Windows Management Instrumentation (WMI) Information page, click **Global Options** when data source properties are displayed. Select **Global Options** either while you are creating the data source or from the Agent Editor, Data Sources page.

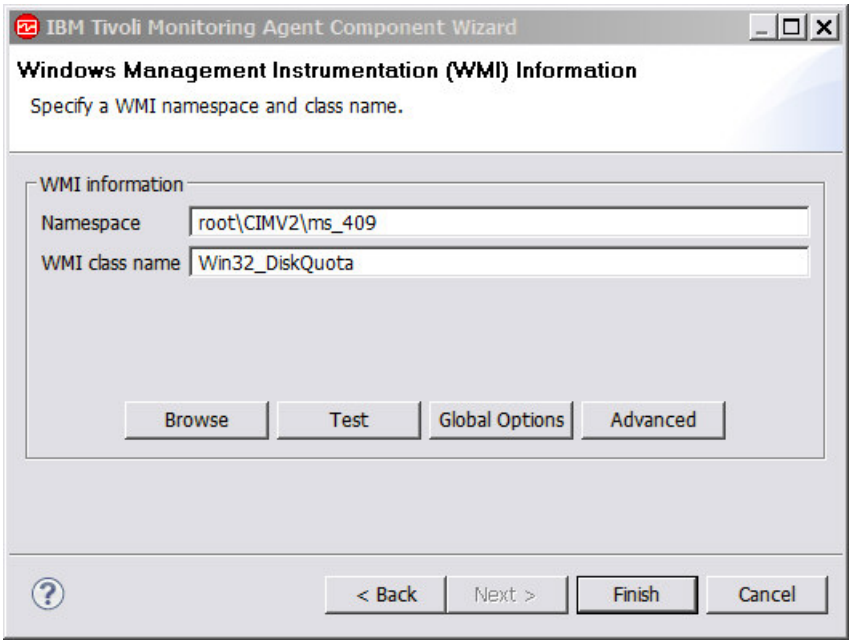


Figure 242. Windows Management Instrumentation (WMI) Information page

2. In the Global Windows Data Source Options window, select **Include Windows Remote Connection configuration** if you want to include these properties in the agent.

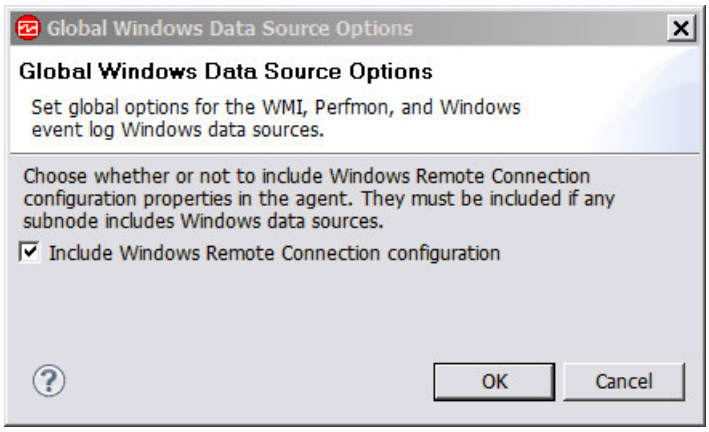


Figure 243. Global Windows Data Source Options

Subnodes and Script data sources

Subnode instance configuration properties are accessed in subnode scripts just as they are in agent-level scripts.

Scripts have access to all agent-level configuration properties and all subnode instance configuration properties. If an agent-level property is overridden at the subnode level, the script has access only to the subnode level property value.

Chapter 30. Customizing agent configuration

Customize the configuration of process, log file, and script data sources.

Before you begin

If you are adding SNMP, JMX, CIM, JDBC, HTTP, and SOAP data sources to your agent, configure these data sources as described in the following chapters:

- Chapter 12, "Monitoring data from a Simple Network Management Protocol (SNMP)," on page 121
- Chapter 14, "Monitoring Java Management Extensions (JMX) MBeans," on page 143
- Chapter 15, "Monitoring data from a Common Information Model (CIM)," on page 171
- Chapter 21, "Monitoring data from Java Database Connectivity (JDBC)," on page 239
- Chapter 23, "Monitoring HTTP availability and response time," on page 261
- Chapter 24, "Monitoring data from a SOAP data source.," on page 273

About this task

Use this task to customize the configuration of process, log file, and script data sources so an agent can access the application that it is monitoring.

All IBM Tivoli Monitoring agents must be configured before they can be started. All agents must have basic configuration information such as the method of connecting to the Tivoli Enterprise Monitoring Server. Many times, an agent must have more configuration information so it has access to information specific to the system on which it is running. For example, if you must know the installation location of a software product, add configuration properties to prompt for this information. Another example of information you might prompt for is the user ID and password to access an interface,

Custom configuration is defined by the agent developer. It is not required for all agents, but can be used in the following areas of data collection:

- Matching an argument in a Process Monitor
- Matching the command line in a Process Monitor
- Forming a log file path or name
- Defining an environment variable in a script

Note: Certain data sources such as JMX and SNMP add this configuration automatically.

Note: When data source specific configuration is added automatically by the Agent Builder, this configuration is added in English only.

If during data source definition your agent requires system-specific information for an area of data collection, **Insert Property** or **Insert Configuration Property** is shown.

For example, when you create an attribute group that monitors a log file, **Insert Configuration Property** is shown.

© Copyright IBM Corp. 2007, 2013



Figure 244. **Insert Configuration Property** *on the Log File Information page*

Procedure

1. Click **Insert Configuration Property** to display the Configuration Properties window (Figure 245 on page 363).

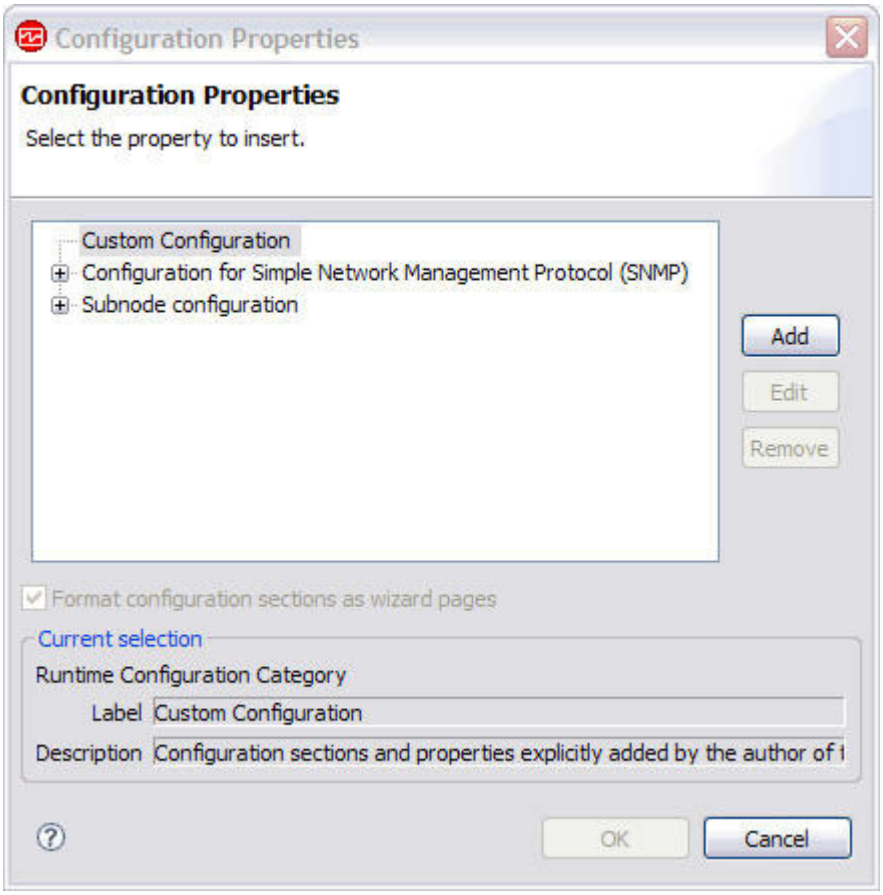


Figure 245. Initial Configuration Properties window

2. In the Configuration Properties window, click a property, and click **Add**.

   Note: Initially there are no configuration properties that are defined for the agent.
3. In the Runtime Configuration Property window (Figure 246 on page 364), complete the fields as described in this step.

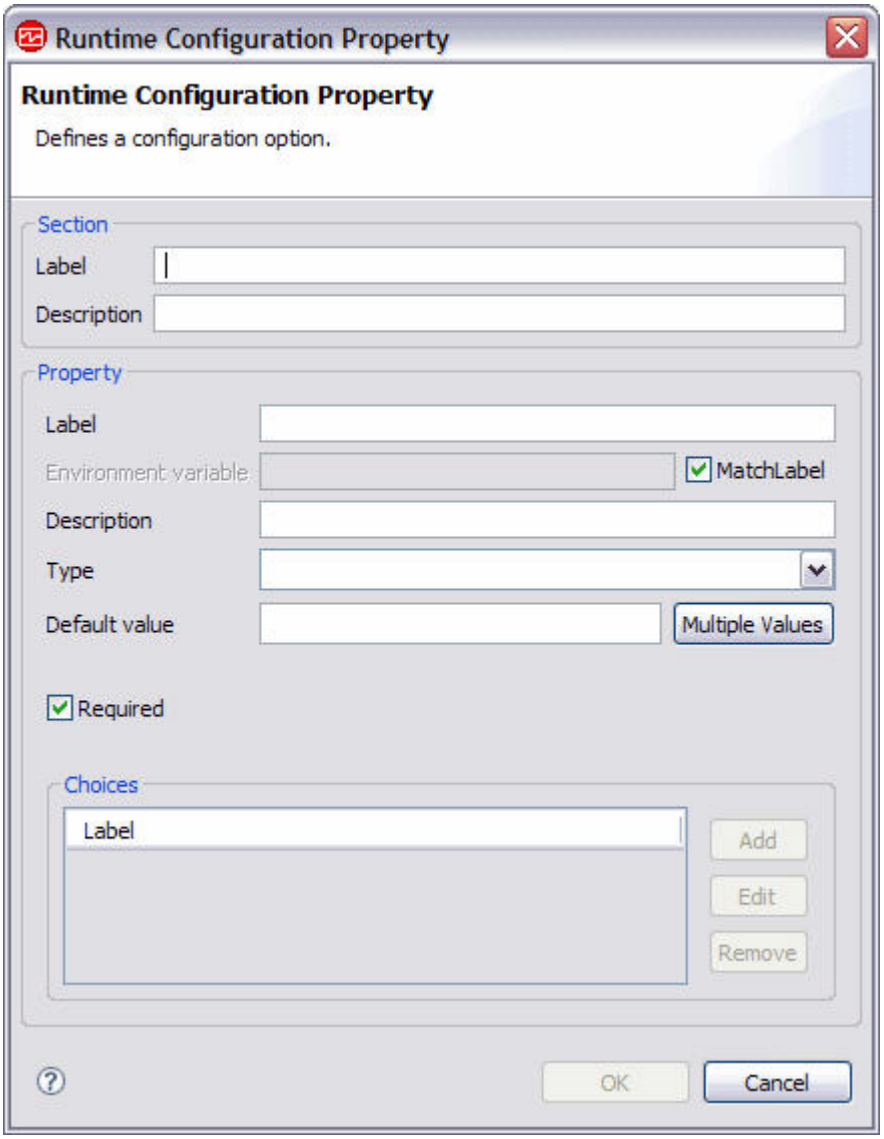


Figure 246. Runtime Configuration Property window

a. In the Section area, complete the following fields:

   Label Text that describes the properties

   Description
   (optional) Description of the properties

b. In the Property area, complete the following fields:

   Label Text that is displayed in the agent configuration panel that identifies the information you must enter.

   Environment variable
   The environment variable is displayed in the **Environment variable** field and is updated as you type in the label field. The Agent Builder automatically constructs the name of the environment variable from the product code and the label. If you want to change the environment variable independently from the label, you can clear **Match Label**.

   Description
   (optional) Description of the property that is being defined.

Type Type of information that is collected, one of the following options:

String For any alphabetic information that must be collected (for example, installation locations, user names, and host names).

Password
For any information that must be encrypted when stored. In addition to providing encryption of the data, the data that is typed into the text box is obscured by asterisks. In addition, you are required to type this information twice to validate the data.

Numeric
For any numeric information (for example, port numbers).

Choice
For a list of specific values. This option enables the Choices table. You can define specific values by clicking **Add**. The values that are entered are displayed in the agent configuration panel as a group of selections, you can make only one selection from the group.

Read Only Text
Displays text when you configure the agent, but no information is collected.

Separator
Displays a horizontal separator, but no information is collected.

File Browser
Collects a string, which is a file name. Click **Browse** to browse the file system for the wanted file.

Default value
(Optional) Specify the value that is shown in the configuration panel at run time when the agent is configured for the first time. If you want a default value for UNIX/Linux that is different from a default value for Windows, click **Multiple Values**.

In the Configuration Property Default Values window (Figure 247), specify the default values that you want for Windows and for UNIX and Linux.

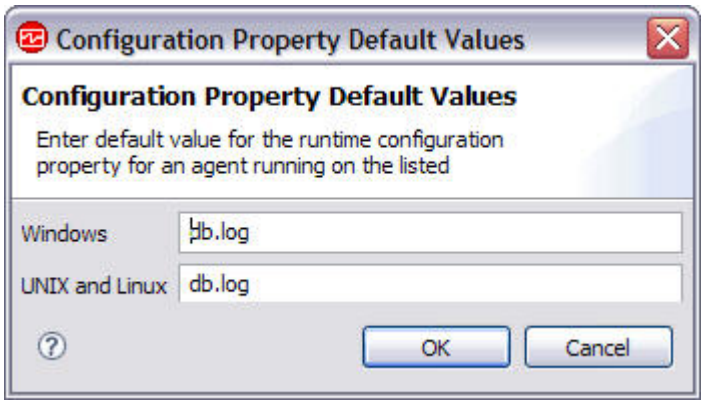


Figure 247. Configuration Property Default Values window

Note: Support for multiple default values is a feature that is only supported in IBM Tivoli Monitoring V6.2.1 and higher. If your agent is compatible with IBM Tivoli Monitoring V6.2, a prompt warns you about this requirement and you can cancel or continue with V6.2.1 compatibility enabled.

Required
Check this field if the user must enter a value when the agent is configured. Clear this field if it is optional for the user to enter a value.

c. To add a choice, click **Add**

4. In the Configuration Property Value window (Figure 248), complete the **Label** and **Value** fields. The Label is displayed as one of the choices. If this choice is taken, the value becomes the property value.

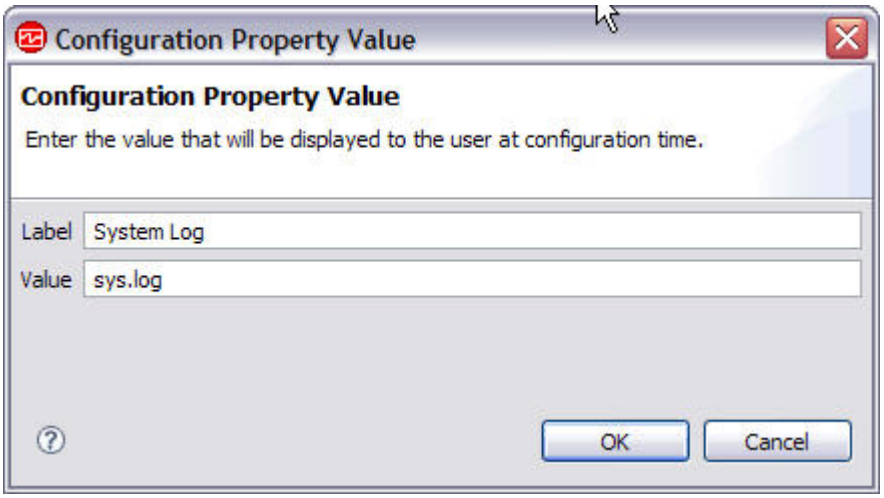


Figure 248. Adding a configuration property choice

5. Click **OK**. The new configuration section and property are displayed in the Configuration Properties window under **Custom Configuration** (Figure 249 on page 367).

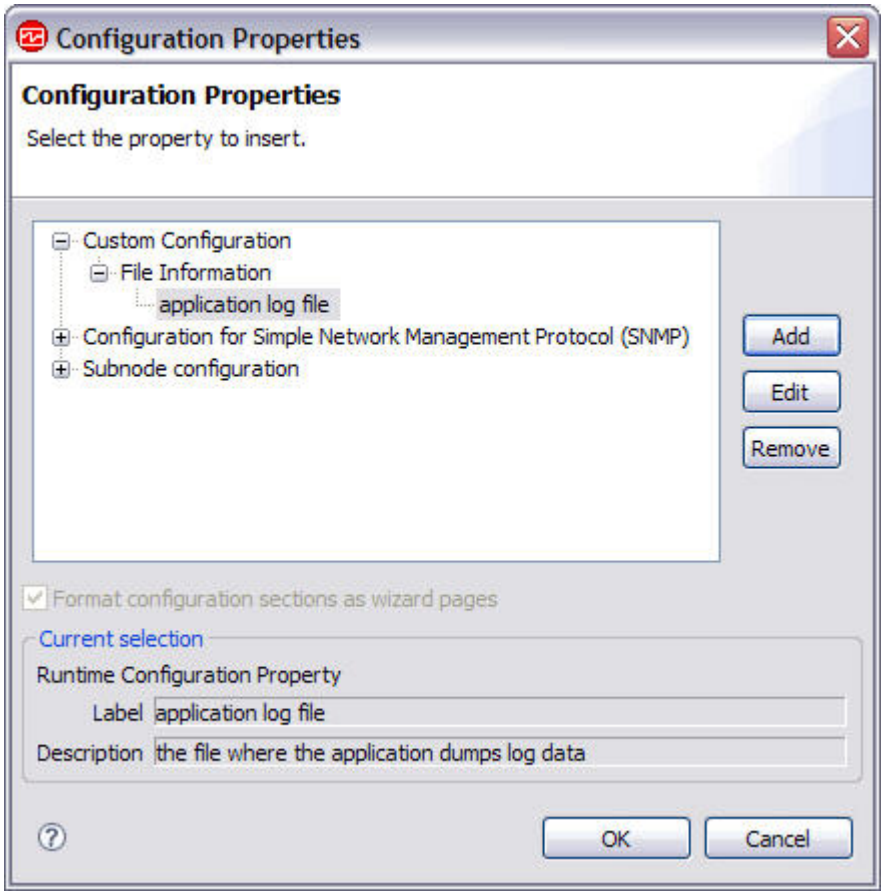


Figure 249. New configuration property displayed

6. Optional: To add another property to an existing section, select the section or an existing property in the section and click **Add**. You make the selection in the runtime configuration tree of the Configuration Properties window.

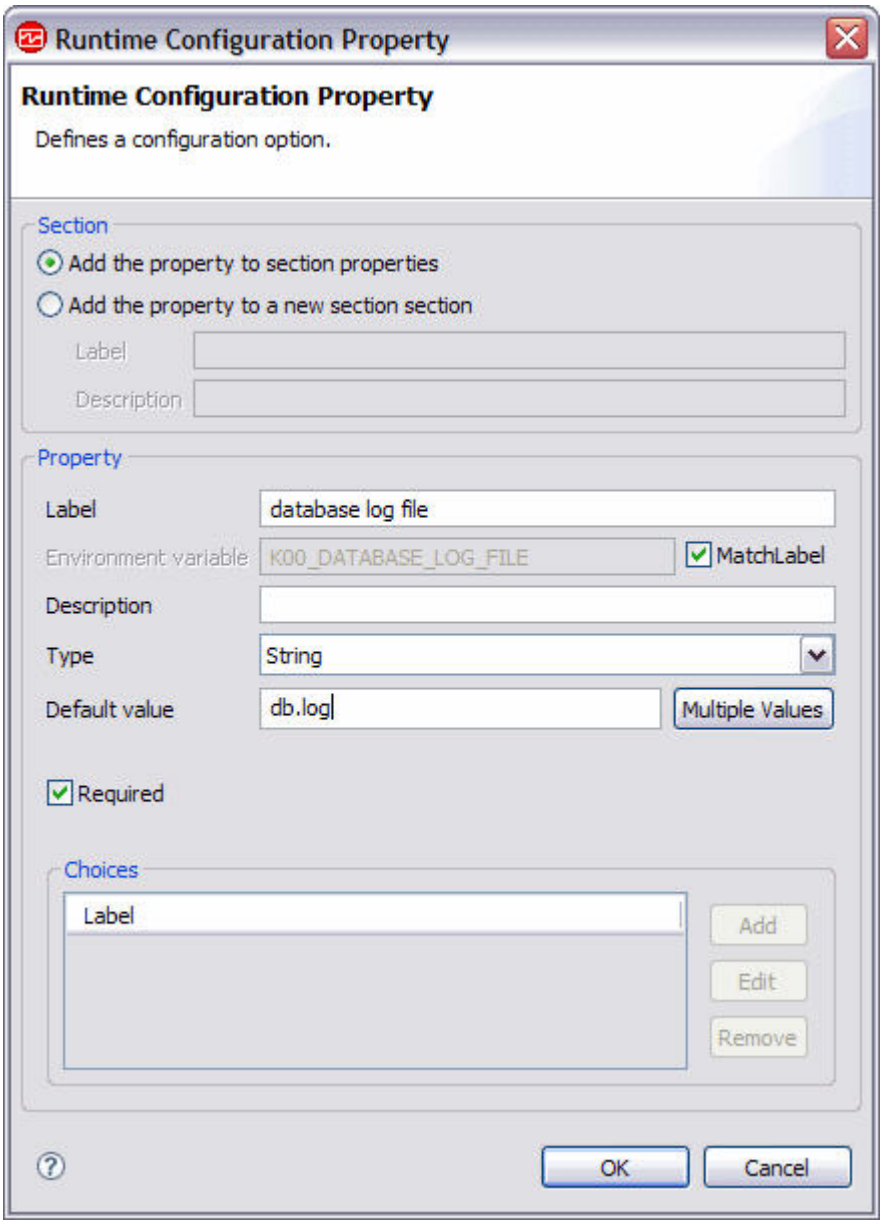


Figure 250. Adding a second property to an existing section

7. Complete the fields for the new property. (Complete the same fields as in step (3 on page 363).
8. Click **OK**. The property that you most recently added is selected.
9. Keep the selection or select the property that you want to insert into the log file name.
10. Click **OK**. The property is inserted into the log file name.

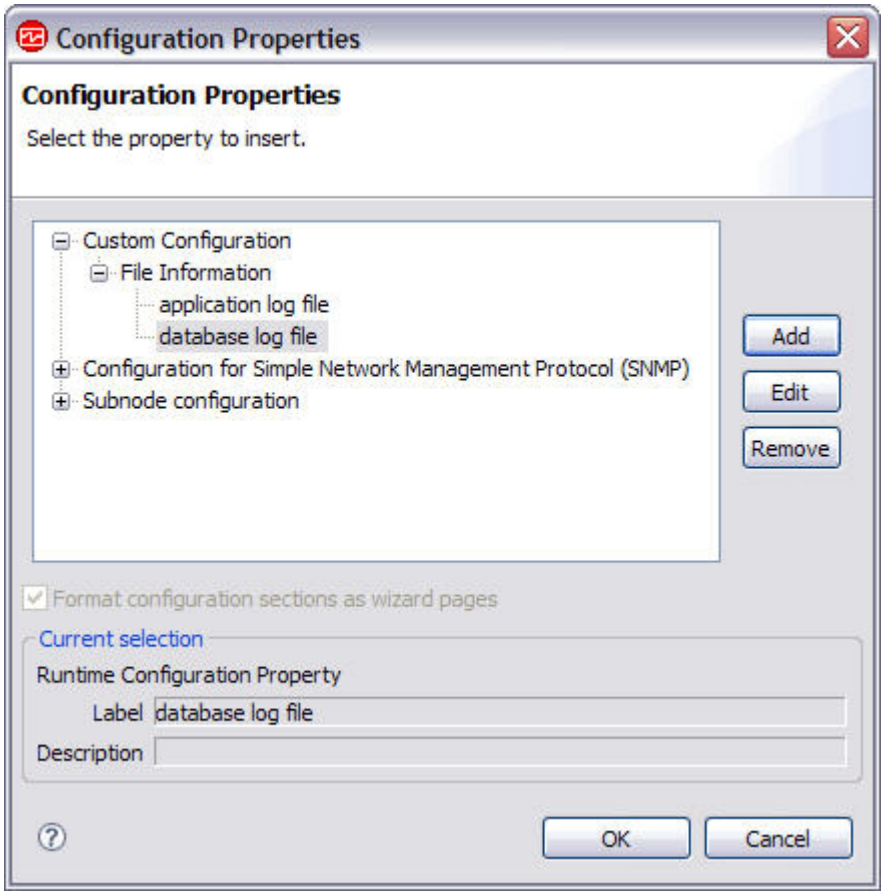


Figure 251. Property inserted into the log file name

You can then continue through the wizard to complete defining your log file attribute group.

Note: Even though a configuration property is defined in the context of a log file name, it can be used in other locations. For instance, another location that accepts a configuration property is a script data source. This flexibility means that you can access the value for the configuration element **File Information** with the script variable *$K00_APPLICATION_LOG_FILE* if the product code is K00. You can also use the Windows batch file variable *%K00_APPLICATION_LOG_FILE%*.

Changing configuration properties by using the Agent Editor

Use the Agent Editor to change configuration properties of your agent.

About this task

This task provides information about viewing, adding, and changing configuration properties by using the Agent Editor.

Procedure

1. Click the **Runtime Configuration** tab.

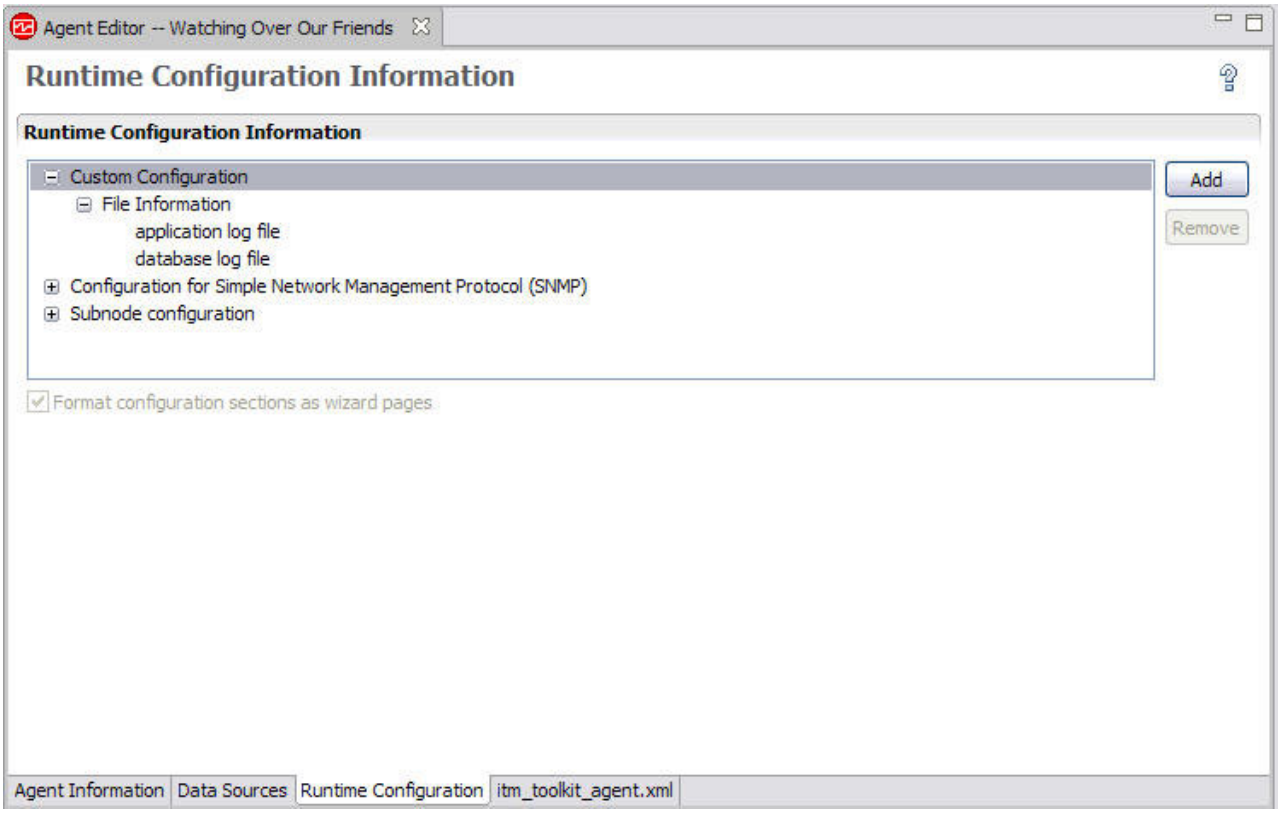


Figure 252. Runtime Configuration Information page in the Agent Editor

2. Select a configuration section, and click **Add**.

   Add works just like it does in (Chapter 30, "Customizing agent configuration," on page 361). There is no **Edit** selection because a configuration section or property is edited when it is selected.

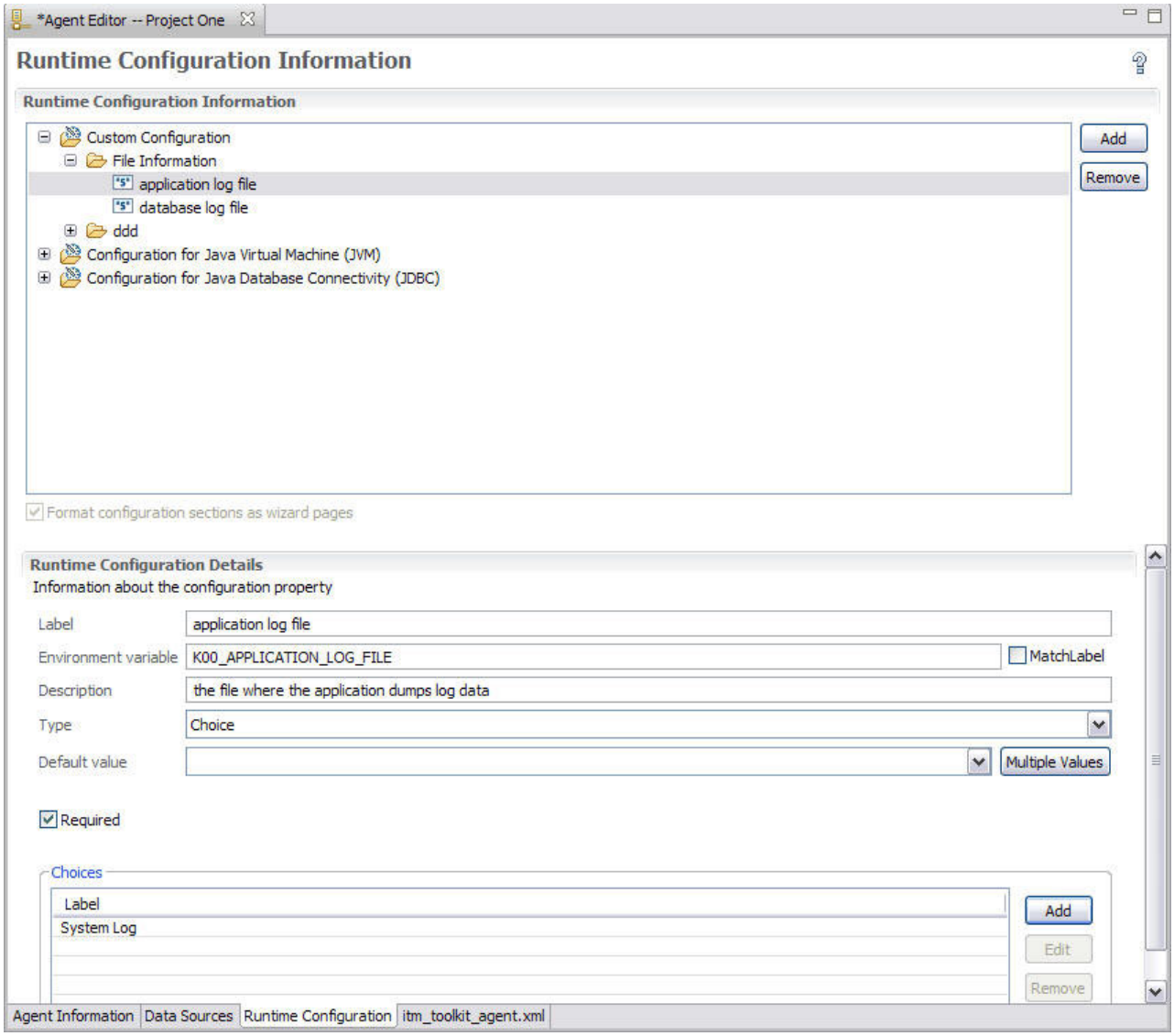


Figure 253. Editing a configuration section in the Agent Editor

3. Select a configuration property to display the Runtime Configuration Details area.

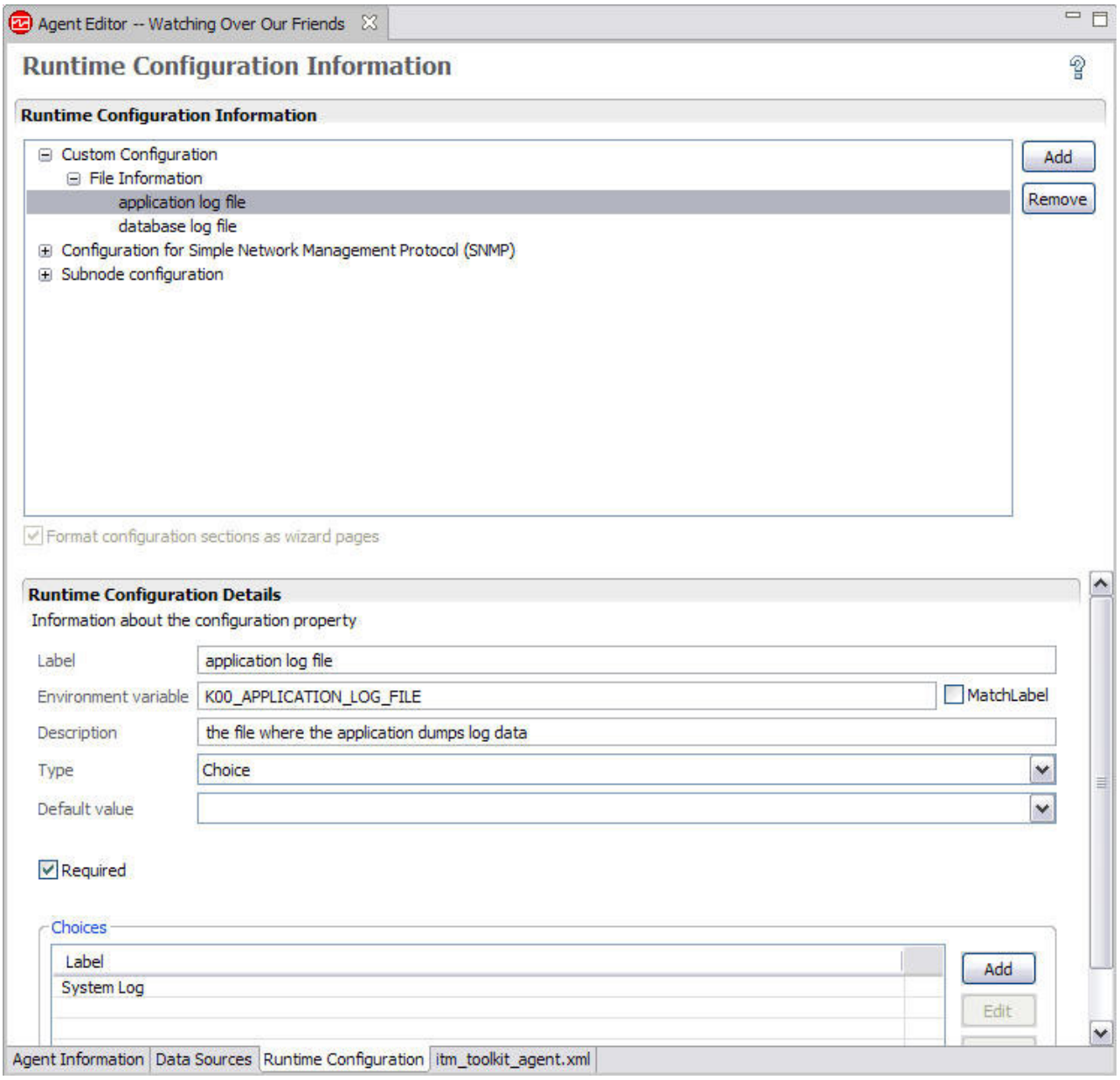


Figure 254. Editing a configuration property in the Agent Editor

4. In the Runtime Configuration Details area, edit the fields to configure the property.

Configuring a Windows remote connection

Information about Configuring a Windows remote connection

About this task

Windows Management Instrumentation (WMI), Windows Performance Monitor (Perfmon), and Windows Event Log data sources can monitor data on the system where the agent is installed. These data sources can also monitor data on remote Windows systems. These three data source types are known as Windows data sources. If these Windows data sources are monitoring data remotely, they all share Windows Remote Connection configuration properties for the agent level where they are defined.

If you define a Windows data source in the base level of your agent, Windows Remote Connection configuration properties are not added to the agent automatically. They are not added to maintain compatibility with earlier versions of agents that might use the Windows data provider before remote monitoring was enabled. The Windows data source in your agent monitors data on the local Windows system where the agent is installed.

If you define a Windows data source in a subnode in your agent, Windows Remote Connection configuration properties are added to the agent automatically. The Windows data source must support Windows Remote Connection if it is in a subnode. You cannot clear the option until all windows data sources are removed from all subnodes in the agent. Each instance of a subnode might be configured to monitor a different remote Windows system. All Windows data sources in the subnode share Windows Remote Connection configuration properties.

To configure a base agent to remotely monitor a single remote Windows system, use the following procedure.

Procedure

1. In the Agent Editor, Data Source Definition window, click **Global Options**. The Global Windows Data Source Options window opens.
2. Select **Include Windows Remote Connection configuration**.
3. Click **OK**.

Results

The following connection-specific configuration properties can be accessed from the Agent Editor, Runtime Configuration Information page by selecting **Configuration for Windows remote access** > **Windows Remote Connection**

Remote Windows host
Host name of remote Windows computer

Remote Windows password
Password for remote Windows

Remote Windows `DOMAIN\user name`
User name for the remote Windows host

What to do next

You can view, add, and change the configuration properties by using the Agent Editor. See ("Changing configuration properties by using the Agent Editor" on page 369). If a Windows data source is defined in a subnode, you can also specify Subnode Configuration Overrides. See ("Subnode configuration" on page 344).

Creating a user with Windows Management Instrumentation (WMI) permissions

You can add and configure a user on a Windows system with permissions to allow WMI browsing.

About this task

If your agent collects data from a remote system by using Windows Management Instrumentation (WMI), it requires permissions to access WMI data on the remote system. The agent can access WMI data on a remote system when you provide credentials of an account with permissions to access WMI data on the system. The procedure applies to Windows 7, Windows 2008 Server and Windows Vista.

Note: Your agent can also access data on a remote Windows system by using Windows Performance Monitor (Perfmon), and Windows Event Log data sources. However, in the case of Windows Performance Monitor (Perfmon), and Windows Event Log data sources, you must provide Administrator credentials for the remote system.

Procedure

1. Create a user account:
   a. Go to Windows **Start** > **Administrative Tools** > **Computer Management**. The Computer Management window opens.
   b. Expand **Local Users and Groups**.
   c. Right click the **Users** folder and select **New User**.
   d. Complete the user details and click **Create** and **Close**.
2. Configure the group membership for the new user account:
   a. In the Computer Management window, select the **Users** folder.
   b. Right-click the new user account and select **Properties**.
   c. Click the **Member Of** tab.
   d. Click **Add**.
   e. Click **Advanced**.
   f. Click **Find Now**.
   g. Select the following groups:
      - Distributed COM Users
      - Performance Log Users
      - Remote Desktop Users

      Tip: Press Ctrl and click to select multiple groups.
   h. Click **OK** until you return to the Computer Management window.
   i. Select **File** > **Exit** to exit the Computer Management window.
3. Assign Distributed Component Object Model (DCOM) rights:
   a. Go to Windows **Start** > **Administrative Tools** > **Component Services**. The Component Services window opens.
   b. Expand **Component Services** > **Computers** > **My Computer**.
   c. Right-click **My Computer** and select **Properties**. The My Computer Properties window opens.
   d. Click the **COM security** tab.
   e. In the Access Permissions area, click **Edit Limits**
   f. In **Distributed COM Users**, verify that **Local Access** and **Remote Access** are selected.
   g. Click **OK** to save settings.
   h. In the My Computer Properties window, Launch and Activation Permissions area, click **Edit Limits**
   i. In **Distributed COM Users**, verify that **Local Launch**, **Remote Launch**, **Local Activation**, and **Remote Activation** are selected.
   j. Click **OK** to save settings and click **OK** again to close the My Computer Properties window.
   k. Select **File** > **Exit** to exit the Component Services window.
4. Configure the WMI namespace security assignments
   a. Go to Windows **Start** > **Run...**.
   b. Enter `wmimgmt.msc` and click **OK**.
   c. Right-click **WMI Control (Local)** and select **Properties**.
   d. Click the **Security** tab.

e. Click **Security**.
f. Click **Add**.
g. Click **Advanced**.
h. Click **Find Now**.
i. Select the new user account, and click **OK** until you return to the Security for Root window.
j. Click **Advanced** and select the newly added user account.
k. Click **Edit**.
l. From the **Apply to:** menu selection, select **This namespace and subnamespaces**.
m. In **Execute Methods**, verify that **Enable Account**, **Remote Enable**, and **Read Security** are selected.
n. Click **OK** until you return to the wmimgmt window.
o. Select **File** > **Exit** to exit the wmimgmt window.

What to do next

For more information about collecting WMI data from a remote system, see Chapter 10, "Monitoring data from Windows Management Instrumentation (WMI)," on page 105.

Configuring a Secure Shell (SSH) remote connection

Information about configuring an SSH remote connection

About this task

Script data sources can monitor data on the system where the agent is installed and also on remote systems. If script data sources are monitoring data remotely, they all share SSH remote connection configuration properties for the agent level where they are defined. Earlier versions of an agent might use the script data provider before remote monitoring was enabled. To maintain compatibility with earlier versions of agents, SSH remote connection configuration properties are not automatically added to the agent. The script data source in your agent monitors data on the local system where the agent is installed.

If you define a Script data source in a subnode and you select **Enable data collection using SSH**, you can configure each subnode instance to monitor a different remote system. All script data sources in the subnode share SSH remote connection configuration properties.

If you want the agent to remotely monitor a remote system, use the following procedure.

Procedure

In the Agent Editor, Data Source Definition window for the script data source, select **Enable data collection using SSH**.

Results

The following connection-specific configuration properties can be accessed from the Agent Editor, Runtime Configuration Information page by selecting **Configuration for Secure Shell (SSH)** > **SSH Remote Connection**

Network address
The IP address or host name of the remote computer.

SSH Port Number
The IP port number on which the SSH server is running. The default value is 22.

Authentication Type
Type of authentication to use when you are logging on to the remote SSH server. You can choose Password or Public Key.

Disconnect from the remote system after each collection interval
An option to determine whether the script data provider drops the login session to the remote system after it collects data. By default, the value is No.

Remove script from the remote system after each collection interval
An option to delete the script from the remote system after each data collection interval. By default, the value is No.

If the Authentication Type is set to Password, the following configuration properties can be accessed from the Agent Editor, Runtime Configuration Information page by selecting **Configuration for Secure Shell (SSH)** > **Password**:

Username
User name for the remote system

Password
Password for the remote system

If Authentication Type is set to Public Key, the following configuration properties can be accessed from the Agent Editor, Runtime Configuration Information page by selecting **Configuration for Secure Shell (SSH)** > **Public Key**:

Username
User name that is associated with the public key file

Public Keyfile
Public key file that is associated with the user

Private Keyfile
Private key file that is associated with the user

Password
Password that is used to unlock the private key file

What to do next

You can view, add, and change the configuration properties by using the Agent Editor. See ("Changing configuration properties by using the Agent Editor" on page 369). If the SSH Remote Connection configuration properties are included in a subnode, you can also specify Subnode Configuration Overrides. See ("Subnode configuration" on page 344).

Chapter 31. Testing and debugging your agent

After you use the Tivoli Monitoring Agent Builder to create an agent, test the agent.

Test the agent to ensure that the monitoring data you are expecting is the data that is being displayed. By testing your agent, you can learn to modify or tweak settings in the agent to ensure that the data displayed is beneficial and accurate.

You can test your agent from the bottom-up by using the following methods:

1. Begin by using the attribute group test function of Agent Builder to test individual attribute groups one at a time. For more information, see "Attribute group testing."
2. After you complete attribute group testing, you can use the agent test function of Agent Builder to test all attribute groups in your agent together. For more information, see "Full agent testing" on page 382.
3. You can complete testing by installing your agent and doing further testing together with the Tivoli Enterprise Portal. For more information, see "Installing and testing an agent" on page 387.

Attribute group testing

You can use attribute group testing to test the attributes groups of the agent you created with Agent Builder, one attribute group at a time. You can test many attribute groups before you complete the attribute group definition. For example, you can initiate testing from the IBM Tivoli Monitoring Agent Wizard when you are defining the attribute groups of a new agent. You can also initiate testing from the IBM Tivoli Monitoring Agent Component Wizard when you are adding attribute groups to an existing agent.

Before you begin

Before you start testing an attribute group, you can optionally:

- Set attribute group testing preferences. For more information, see "Attribute group testing - preferences" on page 379.
- Set environment variables, configuration properties, and where applicable Java information. For more information, see "Attribute group testing - configuration" on page 380.

About this task

Agent Builder supports an attribute group test function for most data sources

Procedure

- Start the Testing procedure in the following ways:
  1. During agent or attribute group creation click **Test** on the relevant data source Information page.
  2. After agent creation, select an attribute group on the Agent Editor Data Source Definition page and click **Test** . For more information about the Agent Editor, see (Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35).

  After you click **Test** in one of the previous two steps, the Attribute group Test window is displayed, for example the WMI Test window is shown in (Figure 255 on page 378).

© Copyright IBM Corp. 2007, 2013

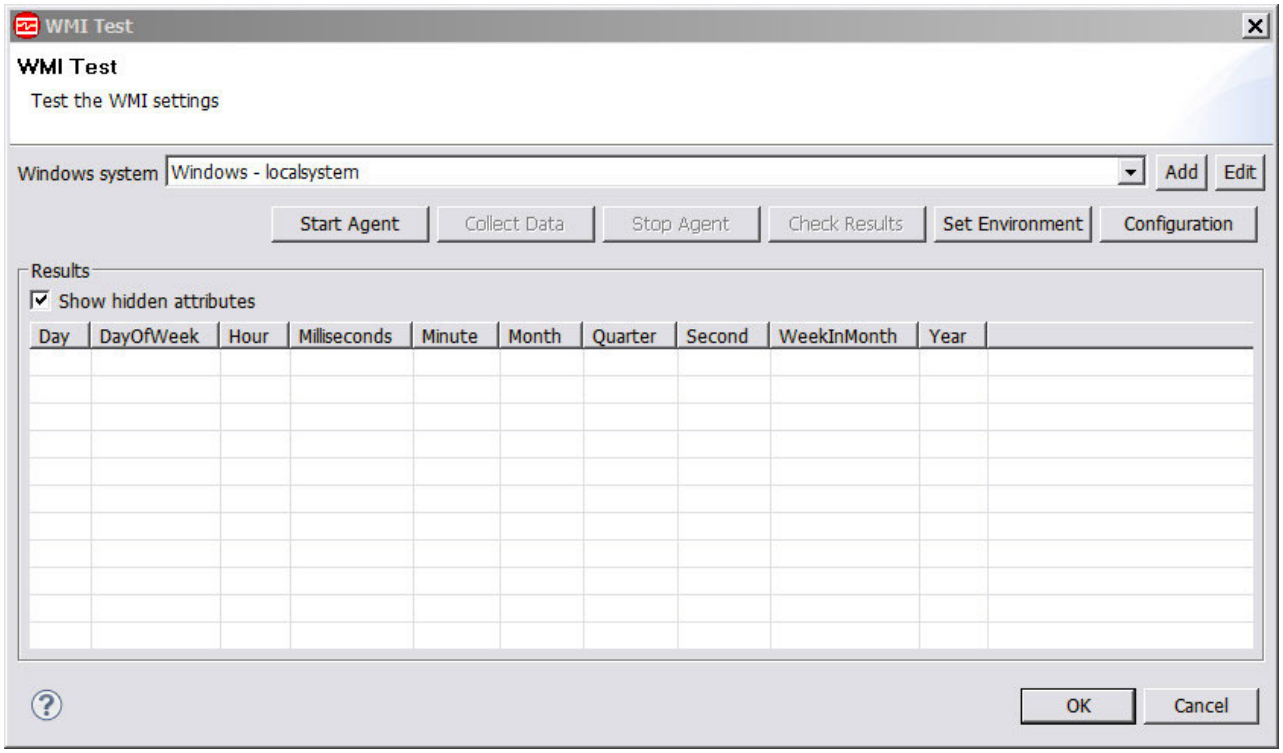


Figure 255. WMI Test window

Agent Builder supports an attribute group test function for most data sources.

For more information about the test procedures for specific attribute groups, see the following Testing sections:

- Windows Management Instrumentation (WMI), for more about the WMI test procedure, see "Testing WMI attribute groups" on page 111
- Windows Performance Monitor (Perfmon), for more about the Perfmon test procedure, see "Testing Perfmon attribute groups" on page 118
- Simple Network Management Protocol (SNMP), for more about the SNMP testing, see "Testing SNMP attribute groups" on page 130
- Simple Network Management Protocol (SNMP) event sender, for more about the SNMP event test procedure, see "Testing SNMP event attribute groups" on page 140
- Java Management Extensions (JMX), for more about the JMX test procedure, see "Testing JMX attribute groups" on page 169
- Common Information Model (CIM), for more about the CIM test procedure, see "Testing CIM attribute groups" on page 174
- Log file, for more about the log file test procedure, see "Testing log file attribute groups" on page 193
- Script, for more about the script test procedure, see "Steps for monitoring output from a script" on page 229
- Java Database Connectivity (JDBC), for more about the JDBC test procedure, see "Testing JDBC attribute groups" on page 253
- Internet Control Message Protocol (ICMP) ping, for more about the ICMP test procedure, see "Testing ping attribute groups." on page 259
- Hypertext Transfer Protocol (HTTP) Availability, for more about the HTTP test procedure, see "Testing HTTP attribute groups" on page 270
- SOAP, for more about the SOAP test procedure, see "Testing SOAP attribute groups" on page 282

– Transmission Control Protocol socket (TCP) socket, for more about the socket test procedure, see “Testing socket attribute groups” on page 298
– Java application programming interface (API), for more about the Java API test procedure, see “Testing Java application attribute groups” on page 316

Some data sources do not have an attribute group test function, for example:

– When you can use the Agent Builder browser to view live data on a system. For example, you can view the processes that are currently running on the system (processes). Other examples are when you can view the services that are installed on the system (windows services) and the Windows Event Logs that are present.
– There is little or no customization that you can do in the agent (AIX Binary Log, command return code).
– Joined and Filtered attribute groups cannot be tested by using the attribute group test function because these groups are based on multiple attribute groups.

Note:

1. Use full agent test to test data sources that cannot be tested by using the attribute group test function. For more information about full agent test, see (“Full agent testing” on page 382).
2. When you test data sources, after you click **Collect Data**, data might not be displayed at all or might not be current after the first click. In such cases, click **Collect Data** a second time to display current data.

- **Debugging:**

  Each data source that is tested has a test directory that is created for it by Agent Builder. This directory is used for the test runtime environment of the data source. Log files that relate to tests run on the data source are stored under this directory. The log files can be useful to help debug issues that are found during testing.

  Note:

  1. The location of the test log file is shown as a status message in the Test window after you click **Start Agent** and also after you click **Stop Agent**. For example, see (Figure 256).

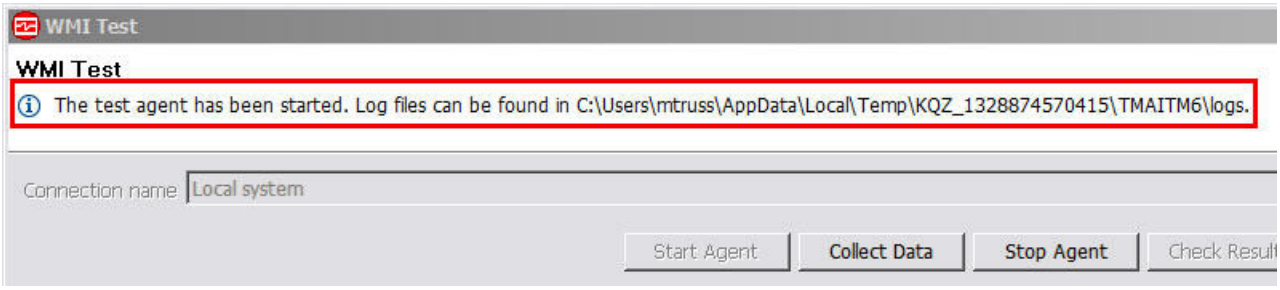


Figure 256. WMI Test window that shows test log location.

2. All test data source directories are deleted when the Agent Builder is shut down.

Attribute group testing - preferences

Set preferences before you test an attribute group.

About this task

Before you start testing an attribute group, you can optionally set some preferences that determine how attributes are treated during testing.

Procedure

1. Select **Window** > **Preferences** from the Agent Builder menu bar. The Preferences window opens.
2. Select **IBM Tivoli Monitoring Agent Builder**.

   The preferences that are associated with testing attribute groups are shown:

   Show data types changed dialog when testing
   When selected, Agent Builder suggests changes to the data type of an attribute. Agent Builder suggests changes when the data type of an attribute does not match the data that is returned by a test for that attribute. For example, if the string length defined for an attribute is too short to hold a value that is returned by a test. In this example, Agent builder suggests redefining the attribute to have a longer string length. When this option is cleared, Agent Builder does not check or suggest data types during testing. This option is selected by default.

   Maximum script or log attributes created
   The value that is entered in this field determines the maximum number of attributes that Agent Builder parses during the initial test of a log file or script attribute group. The default value is 25.
3. When you are finished setting your preferences, click **OK** to save your settings and close the Preferences window. If you want to restore the default settings, click **Restore Defaults** before you click **OK**

Attribute group testing - configuration

Set environment variables, configuration properties, and Java information before you test an attribute group.

About this task

Before you start testing an attribute group, you can optionally set environment variables, configuration properties, and where applicable Java information from the data source Test window. The Java information is a subset of the configuration data. Some environment variables have special values that are set by default for attribute group testing. For more information about environment variables with special values for attribute group testing, see “Test Environment variables” on page 386.

Procedure

1. Optional: Click **Set Environment** from the data source Test window. The Environment Variables window opens. When populated, the Environment Variables window lists all of the environment variables that are used during the running of the test. The initial view of the Environment variable window contains any existing environment variables that are defined in your agent. It also contains any environment variables that you added from previous tests of this agent.

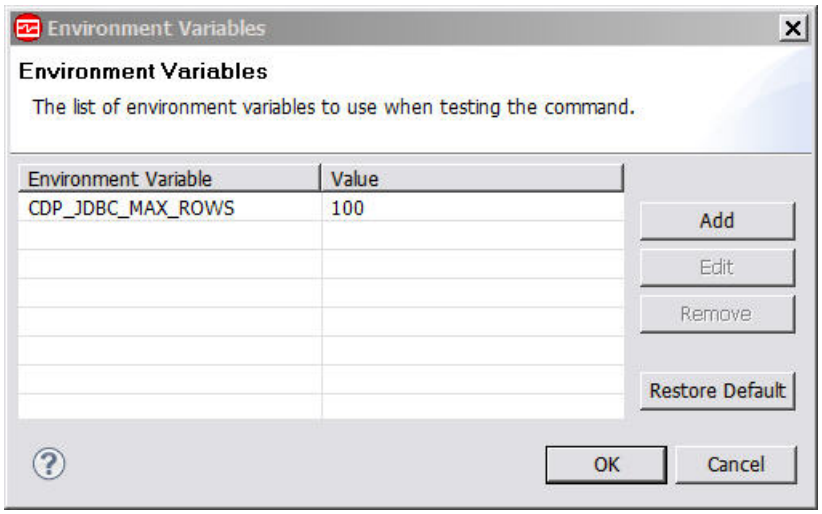


Figure 257. Environment Variables window

a. Click **Add** or **Edit** to add or edit individual variables.
b. Click **Remove** to remove individual variables, or **Restore Default** to restore default variables and remove all others.
c. Click **OK** to save your changes and return to the Test window.

2. Optional: Click **Configuration** from the data source Test window. The Runtime Configuration window opens.

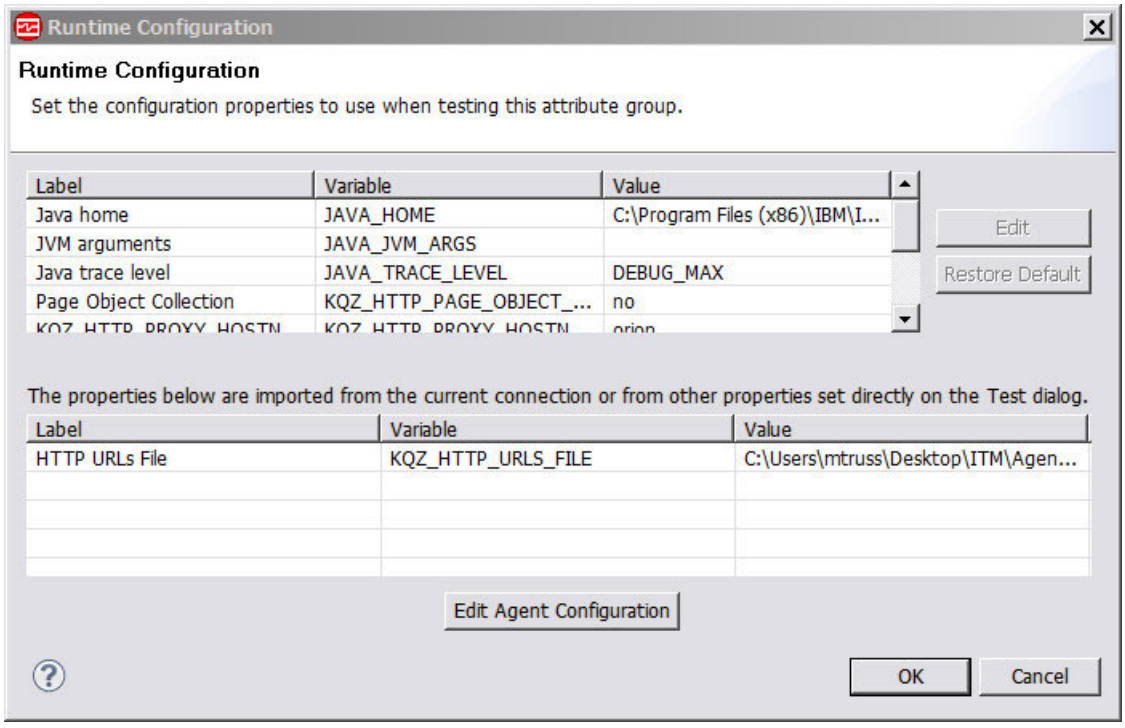


Figure 258. Runtime Configuration window

a. Click **Edit Agent Configuration** to add a configuration property or to edit existing agent configuration properties by using the Configuration Properties window.

b. Select a configuration property and click **Edit** to edit an existing configuration property that relates to the attribute group you are testing.
c. Select a configuration property and click **Restore Default** to restore a configuration property to its default value.

3. Click **OK** to save your changes and return to the Test window.
4. Optional:

   Note: You can set Java information for following types of attribute groups:
   - Java Management Extensions (JMX)
   - Java Database Connectivity (JDBC)
   - Hypertext Transfer Protocol (HTTP) Availability
   - SOAP
   - Java application programming interface (API)

   The Java information is a subset of the configuration data described in step 2 on page 381
   Click **Java Information** from the data source Test window. The Java Information window opens.

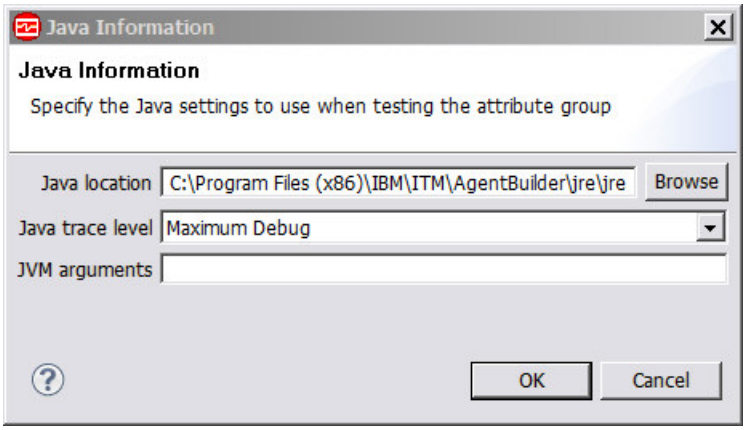


Figure 259. Java Information window

a. Enter Java Information. For example, Browse to or type the location of the Java Runtime Environment (JRE), select a **Java trace level**, or enter **JVM arguments**
b. Click **OK** to save your changes and return to the Test window.

Full agent testing

Use full agent testing to test all attribute groups of your agent together. You can also use full agent testing to test data sources that cannot be tested by using the attribute group test function.

About this task

You can use full agent testing to run the agent in the same way it runs in IBM Tivoli Monitoring without needing an IBM Tivoli Monitoring installation.

Procedure

1. Open the Agent Test perspective:
   - Click **Test the agent** in the Test Agent section of the Agent Editor, Agent Information page.

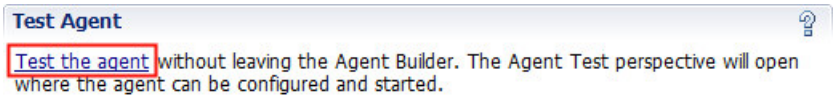


Figure 260. Test Agent section of the Agent Editor, Agent Information page.

- Alternatively, from the Agent Builder menu select **Window** > **Open Perspective** > **Other**, select **Agent Test** and click **OK**.

The Agent Test perspective opens (Figure 263 on page 385). The Agent Test view shows agents that are currently being edited and are eligible to be tested. An Attribute Group Test view is also displayed. The Attribute Group Test view is initially empty. The Attribute Group Test view shows data that is collected from a selected attribute group when the agent is running.

Tip: If no agents are being edited, the Agent Test perspective is empty. To populate the view, go to the IBM Tivoli Monitoring perspective and open an agent in the Agent Editor. When an agent is opened in the Agent Editor return to the Agent Test perspective to test the agent.

2. Optional: Configure environment variables and configuration properties before you start the test. You can access the Environment Variables and Runtime Configuration windows in two ways from the Agent Test view:
   - Right-click the agent in the Agent Test view to open a selection menu. You can select **Set Environment** from the menu to open the Environment Variables window. You can select **Configuration** from the menu to open the Runtime Configuration window.

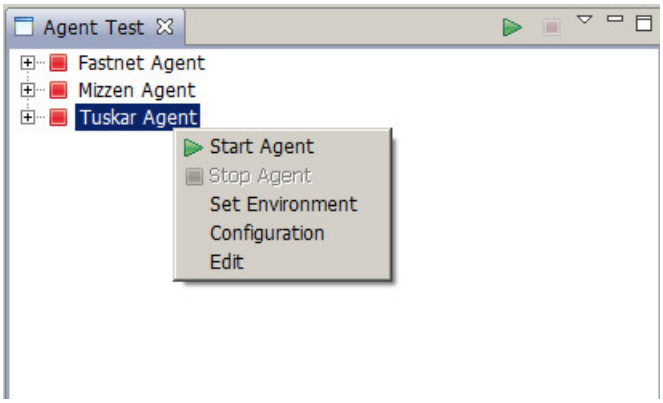


Figure 261. Agent Test menu.

   - Click the view menu icon on the Agent Test view toolbar to access the **Set Environment** and **Configuration** menu items as in the previous choice.

   For more information about using the Environment Variables and Runtime Configuration windows, see (“Attribute group testing” on page 377).

   Note:

   a. The agent is populated automatically with the last set of configuration that relates to each tested attribute group.
   b. Some environment variables can have different default values for attribute group testing and for full agent testing. For more information about environment variables with special values for attribute group testing, see (“Test Environment variables” on page 386).

3. In the Agent Test view, select the agent that you want to test and click the start agent icon .

A window indicates that the agent is starting. When the agent starts, its attributes groups are shown as children of the agent in the Agent Test view. The attribute groups are indicated by the attribute group icon .

The status attribute groups that give information about the agent (**Performance Object Status**, **Thread Pool Status** and **Take Action Status**) are also shown as children of the agent in the Agent Test view. The status attribute groups are indicated by the information icon .

You can start and run more than one agent at the same time.

The stop agent icon becomes available when the agent is started.

If your agent has subnodes or navigator groups, they are shown as nodes in the Agent Test view. Subnode definitions are shown under the agent. A subnode instance node is shown under the subnode definition node. Attribute groups and navigator groups are shown under the subnode instance node. For example:

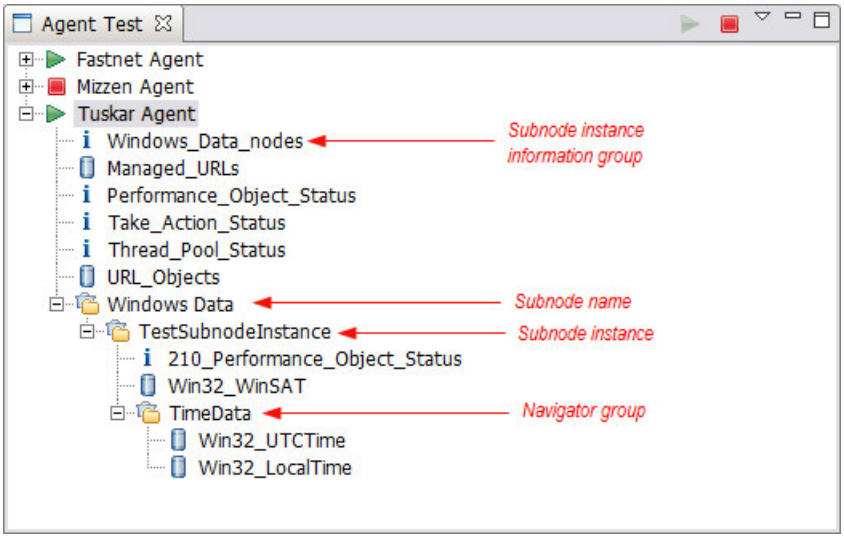


Figure 262. Agent Test view with example subnode and navigator group highlighted.

You can right-click on any of the nodes in the Agent Test view to access menu selections like **Edit** and **Stop Agent**. **Edit** opens the Data Source Definition for the selected node in the Agent Editor.

Note: Changes that you make with the Agent Editor are not visible in the running agent until you stop and restart the agent.

4. In the Agent Test view, select the first attribute group that you want to test.

   When you select an attribute group, a data collection begins for the selected attribute group. If the collection takes some time, a window indicates that the data collection is in progress. When the data collection is complete the collected data is displayed in the Attribute Group Test view, for example:

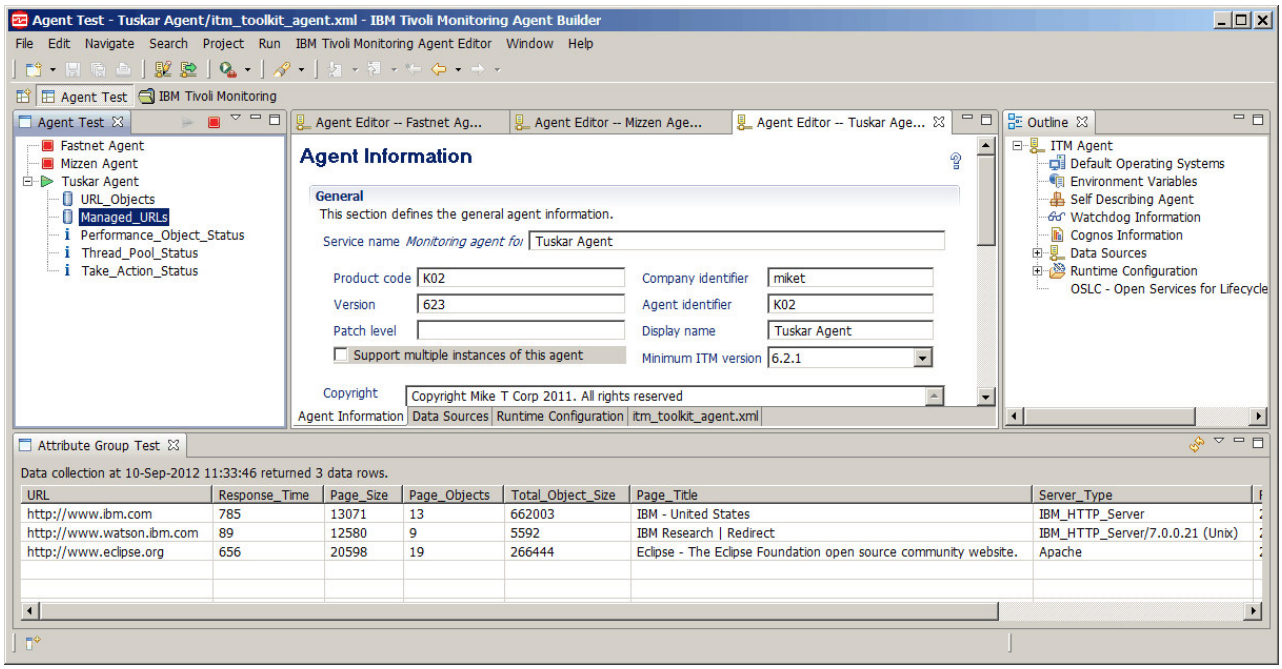


Figure 263. Agent Test perspective

If no data is displayed, a message `0 data rows returned` is shown in the Attribute Group Test view. There are several reasons why the agent might not return data. These reasons include:

- There is no data
- Incorrect definition
- Incorrect configuration

You can check the reason why no data is returned by looking at the value of the **Error_Code** in the **Performance Object Status** attribute group. For more information about viewing the **Performance Object Status** attribute group, see step 9 on page 386

To collect data for another attribute group in the running agent, select the required attribute group. When you select an attribute group in the Agent Test view, the corresponding attribute group is displayed in the Agent Editor view.

5. Optional: Run a second data collection, after the initial data collection, for certain attribute group types, to get useful data values. To run a data collection, click the collect data icon in the Attribute Group Test view.

   If the collection takes some time, a window indicates that a data collection is in progress. When the data collection is complete, the newly collected data is displayed in the Attribute Group Test view.

6. Optional: Click an attribute column heading in the Attribute Group Test view to open the Attribute Information in the Agent Editor Data Source Definition tab. You can also access the same Attribute Information by right-clicking on any data cell in the table and choosing **Edit** from the menu. You can edit properties of the attribute in the normal way. Changes that you make are not visible in the running agent until you stop and restart the agent.

7. Optional: Open multiple Attribute Group Test views at the same time. To open an additional Attribute Group Test view, click the view menu icon on the Attribute Group Test view toolbar and then select **Open view for attribute group**.

   Note: When an additional Attribute Group Test view is opened, it displays the same attribute information as the original Attribute Group Test view. You can then select another attribute group in the Agent Test view to display different attribute group information in the original Attribute Group

Test view. The first time another Attribute Group Test view is opened, it opens in the same location as the original view but with its own tab. If you want to see the two views simultaneously, you can drag the tab to another location in the workspace.

8. Optional: Select the subnode instance information attribute group, if your agent has subnodes, to see how the subnodes are listed in your agent (Figure 262 on page 384). Selecting the subnode instance information attribute group shows subnode instance information in the Attribute Group Test view (for all online subnodes of the selected type).
9. Optional: To see more information about the operation of the agent, you can select the **Performance Object Status** and **Thread Pool Status** attribute groups in the Agent Test view. These status attribute groups are indicated by the information icon . Select these groups to see status information about earlier data collections for your attribute groups.

   For example:

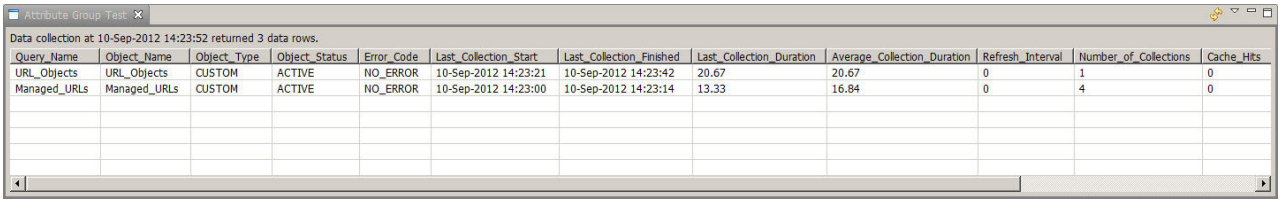


Figure 264. The Attribute Group Test view that shows more information (Performance Object Status) about data collections for the **Managed_URLs** *and* **Managed_Nodes** *attribute groups*

10. When you are finished testing your agent, click the stop agent icon

Test Environment variables

Use these environment variables to control the behavior of the agent during testing.

Environment variables are dynamic named values that determine how the agent runs. For attribute group test, some agent environment variables are set to special values. The special values are used so that the agent responds in a way that suits the testing of a single attribute group. For full agent test special values are not used and instead the default values are used. The default values mean that the agent behaves as it normally would, which is more appropriate to full agent testing.

The environment variables that have special values for attribute group testing are summarized in the following table. For more information about all agent environment variables, see (“Environment variables” on page 435). For more information about setting environment variables, see (“Environment variables” on page 39).

Table 45. Environment variables

Environment variable	Default value (full agent test)	Attribute group test value	Reason for changed value for attribute group test
CDP_DP_INITIAL_COLLECTION_DELAY	varies	1	This value applies to an agent with a thread pool. This value is the time in seconds that the thread pool waits before the initial data collection request is sent to a data provider. **Note:** If CDP_DP_INITIAL_COLLECTION_DELAY is not set, the thread pool waits for a time that is specified by CDP_DP_REFRESH_INTERVAL or CDP_ATTRIBUTE_GROUP_REFRESH_INTERVAL. This wait time is the same time the thread pool waits between data collections, and might be too long to wait for the first data collection.

Table 45. Environment variables (continued)

Environment variable	Default value (full agent test)	Attribute group test value	Reason for changed value for attribute group test
CDP_DP_CACHE_TTL	60	1	When set to 1 a Collect Data request is much more likely to cause the data provider to collect data immediately. Otherwise it might return cached data that is up to 60 seconds old.

Installing and testing an agent

There are two methods for installing and testing the agents that you create with the Tivoli Monitoring Agent Builder.

If Tivoli Monitoring is running on the same system as the Agent Builder, then you can install the agent into the local Tivoli Monitoring installation. If Tivoli Monitoring is not running on the same system, then you can generate a compressed file that you can transfer to another system for testing.

When you finish testing your agent in Agent Builder, there are more benefits to installing and testing your agent with the Tivoli Monitoring Infrastructure. For example, you can:

- Build workspaces, situations, actions, queries in the Tivoli Enterprise Portal.
- Configure and test multiple instances of an agent that run simultaneously.
- Configure and test multiple instances of subnodes that run simultaneously.

Note:

1. After you install an agent, you can see performance metrics in the Tivoli Enterprise Portal tables. For support of situations or workspaces, see Chapter 33, "Importing application support files," on page 409.
2. After you install the agent, you can use the Tivoli Enterprise Portal to verify the data from the agent. For more information, see "Changes in the Tivoli Enterprise Portal" on page 396. If after you view the data in the Tivoli Enterprise Portal, you want to modify the agent, see Chapter 6, "The Tivoli Monitoring Agent Editor," on page 35.
3. For an agent that supports UNIX, generate the installer image on a UNIX system because a UNIX system creates the files with the appropriate permissions.

Installing the agent locally

Install the agent with an IBM Tivoli Monitoring installation on the local system.

About this task

Use the following procedure when you finish creating or editing the new agent and want to install it with an IBM Tivoli Monitoring installation on the local system:

1. Select the `itm_toolkit_agent.xml` file by using one of the following methods:
   a. Right-click the `itm_toolkit_agent.xml` file and select **IBM Tivoli** > **Generate Agent**.
   b. Select the `itm_toolkit_agent.xml` file and select the **Generate Agent** icon on the toolbar.

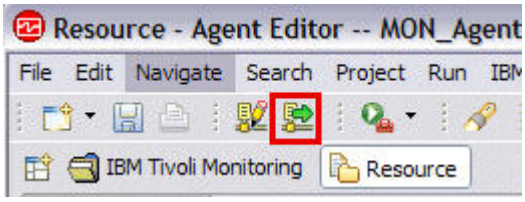


Figure 265. Generate agent icon

c. Double-click the `itm_toolkit_agent.xml` file and select IBM Tivoli Monitoring Agent Editor > **Generate Agent** (Figure 266).

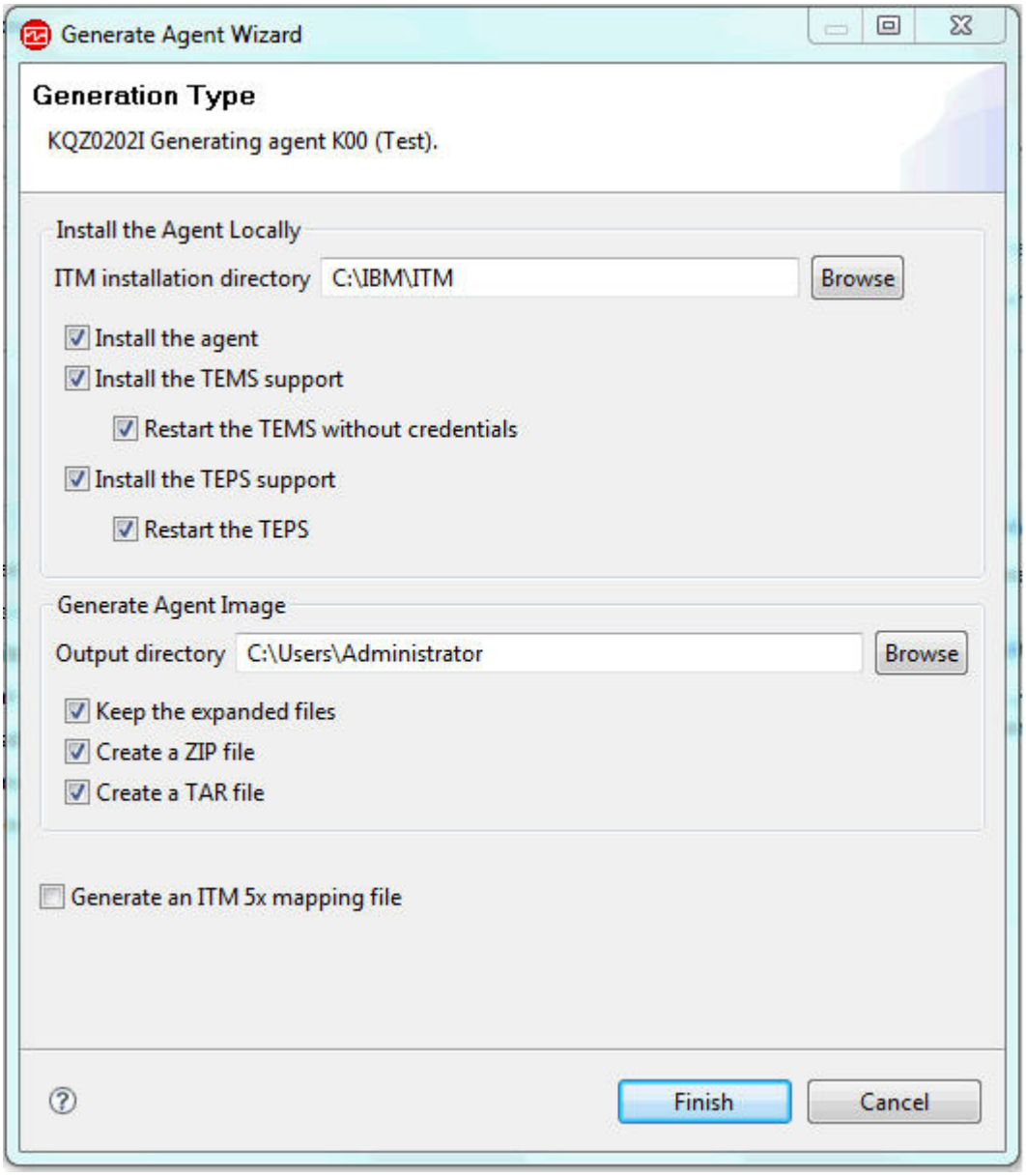


Figure 266. Generate agent menu option

2. In the **Install the Agent Locally** section, type the installation directory for IBM Tivoli Monitoring. The Agent Builder completes the value that is found in the `CANDLE_HOME` environment variable. If this variable is not set, the default value for Windows, `C:\IBM\ITM`, is displayed.

   The check boxes are enabled as follows:

   Install the Agent
   Is enabled if the Agent Builder detects an appropriate TEMA in the location specified. Appropriate means that the agent supports the local operating system and is the correct minimum version.

   Install the TEMS support
   Is enabled if the Agent Builder detects a Tivoli Enterprise Monitoring Server in the location specified.

   Install the TEPS support
   Is enabled if the Agent Builder detects a Tivoli Enterprise Portal Server in the location specified.
3. Select the components to install (agent, Tivoli Enterprise Monitoring Server support, Tivoli Enterprise Portal Server support).
4. Click **Finish**.
5. Configure and start the agent, see ("Configuring and starting the agent" on page 391).

   For Tivoli Monitoring v6.2 FP1 or later, you can install the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server support without restarting the servers. In this case, the **Restart TEMS without credentials** and **Restart TEPS** check boxes are active in the **Install the Agent Locally** section of the Generate Agent wizard. You can clear the check boxes to install the support without recycling the servers. When you clear the **Restart TEMS without credentials** check box, you are prompted for the Tivoli Enterprise Monitoring Server user ID and password. Enter the Tivoli Enterprise Monitoring Server user ID and password and click **Logon**. If you are running Tivoli Monitoring with security off, enter "sysadmin" for the user ID, leave the password blank, and click **Logon**. You can also continue without entering credentials (click **Logon** without specifying a user ID and password or click **Cancel**. Doing so causes the Tivoli Enterprise Monitoring Server to be recycled).

   Note: The Tivoli Enterprise Monitoring Server must be running to install support files without recycling the Tivoli Enterprise Monitoring Server.

Creating the agent image

Use this procedure to create the agent package.

Procedure

1. Select the `itm_toolkit_agent.xml` file by using one of the following methods:
   - Right-click the `itm_toolkit_agent.xml` file and select **IBM Tivoli** > **Generate Agent**.
   - Select the `itm_toolkit_agent.xml` file and select the **Generate Agent** icon on the toolbar.
   - Double-click the `itm_toolkit_agent.xml` file and select **IBM Tivoli Monitoring Agent Editor** > **Generate Agent** (Figure 266 on page 388).
2. Enter the name of the directory where you want to keep the expanded files or the compressed file or files in the Generate Agent Image section,Figure 267 on page 390

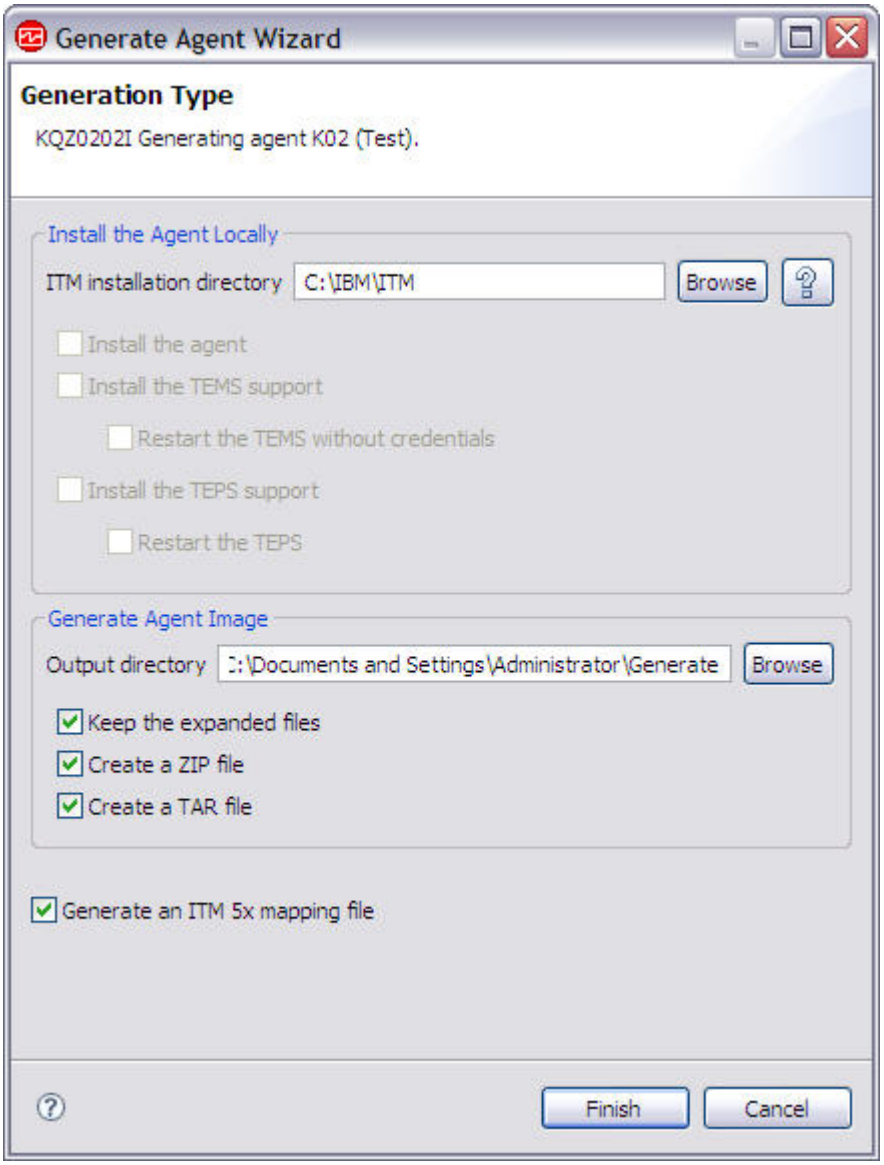


Figure 267. Create an agent image

3. Optional: Select the **Keep the expanded files** check box to keep the generated files.
4. Select the **Create a ZIP file** check box to create a compressed file in the specified directory. The compressed file is named *product_code*`.zip` for Windows systems by default.
5. Select the **Create a TAR file** check box to create a tar file in the specified directory. The compressed file is named *product_code*`.tgz` for UNIX and Linux systems by default.
6. Click **Finish**.
7. Install the package. For more information, see (“Installing the package”).

Installing the package

Install the agent package on other systems:

Procedure

1. Copy the compressed file, which is named *product_code*`.zip` for Windows systems or *product_code*`.tgz` for UNIX and Linux systems by default.
2. Extract the file in a temporary location.

Note: For UNIX, this temporary location cannot be `/tmp/product_code`, where the product code is lowercase.
You can install remotely by using the compressed file.

- On a Linux system, use the following commands to extract your `.tgz` file:

  ```
  tar -xvzf filename
  ```

- On an AIX system, use the following commands to extract your `.tgz` file:

  ```
  gunzip filename
  tar -xvf filename
  ```

3. Run the appropriate installation script.
   - To install the agent, Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal support all at together:

     ```
     InstallIra.bat/.sh itm_install_location [[-h Hub_TEMS_hostname] -u
     HUB_TEMS_username -p Hub_TEMS_password]
     ```

   - To install only the agent:

     ```
     installIraAgent.bat/.sh itm_install_location
     ```

   - To install the Tivoli Enterprise Monitoring Server support:

     ```
     installIraAgentTEMS.bat/.sh itm_install_location [[-h Hub_TEMS_hostname] -u
     HUB_TEMS_username -p Hub_TEMS_password]
     ```

   - To install the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal support:

     ```
     installIraAgentTEPS.bat/.sh itm_install_location
     ```

   The installation location is mandatory on all scripts: `installIra.bat/.sh`, `installIraAgent.bat/.sh`, `installIraAgentTEMS.bat/.sh`, and `installIraAgentTEPS.bat/.sh`, and must be the first argument. The rest is optional. If you are installing on Tivoli Monitoring v6.2 FP1 and a user ID is not provided, the Tivoli Enterprise Monitoring Server is recycled after support file installation. The user ID if provided is provided to `installIra` or `installIraAgentTEMS`. Tivoli Enterprise Monitoring Server restarts during the process of loading the support. If the system on which you are installing is Tivoli Monitoring v6.2, then the new arguments are not allowed. The arguments are not allowed because the new function that uses them is not available. On Tivoli Monitoring v6.2 FP1, if you provide the new arguments, Tivoli Enterprise Monitoring Server support is loaded without causing the Tivoli Enterprise Monitoring Server to restart.

   Note: The following components must be installed before the agent is displayed correctly in the Tivoli Enterprise Portal:
   - The agent
   - Tivoli Enterprise Monitoring Server
   - Tivoli Enterprise Portal Server
   - Tivoli Enterprise Portal support
4. Configure and start the agent, see (“Configuring and starting the agent”).

What to do next

If you changed the layout of your agent in a way that causes Navigator items to be moved or removed, restart the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal. The restart ensures that your changes are correctly recognized.

Configuring and starting the agent

Configure and start your agent.

Procedure

1. Open the **Manage Tivoli Monitoring Service**. The new entry **Monitoring Agent for** *agent_name* is shown.

Note: If the **Manage Tivoli Monitoring Service** window is already open, you can refresh the screen to see the new service.

2. Right-click the entry and select **Configure Using Defaults**. Click **OK** to accept the defaults if you are prompted.

   Note:

   a. On UNIX systems, the option to select is **Configure**.

   b. For multi-instance agents, when you are configuring, you are prompted for an instance name.

   If you added runtime configuration elements to your agent, or if you selected a data source, then you are presented with configuration panels. You use these panels to collect the required information for your agent.

3. Start the agent.
4. Open the Tivoli Enterprise Portal and go to the new agent.

Agent post-generation and installation results

Results to expect after creation and installation of an agent with Agent Builder.

Changes that you see after you generate and install the new agent that is created with the Tivoli Monitoring Agent Builder:

New files on your system

After you generate and install the agent that you created with the Tivoli Monitoring Agent Builder, you can see the following new files on your agent system:

Note: *xx* denotes the two character product code.

Windows systems:

TMAITM6\k*xx*agent.exe
: Agent binary

TMAITM6\K*xx*ENV
: Environment variable settings

TMAITM6\K*xx*.ref
: Agent provider configuration

TMAITM6\SQLLIB\k*xx*.his
: SQL description of agent attribute information

TMAITM6\SQLLIB\k*xx*.atr
: Agent attribute information

TMAITM6*xx*_dd_062000000.xml
: Product description

TMAITM6*xx*_dd.properties
: Product name

TMAITM6\kxxcma.ini
: Agent service definition file

TMAITM6*your files*
: Supplemental Files included from the Java API or Socket data sources with a file type of *executable* or *library*. Scripts included from the Script or Command return code data sources.

UNIX/Linux systems:

registry/*xxarchitecture***.ver**
Internal versions and prerequisites file

architecture/*xx*/**bin/***xx***_dd_061000000.xml**
Product description

architecture/*xx*/**bin/k***xx***agent**
Agent binary

architecture/*xx*/**bin/***xx***_dd.properties**
Product name

architecture/*xx*/**work/k***xx***.ref**
Agent provider configuration

architecture/*xx*/**tables/ATTRLIB/k***xx***.atr**
Agent attribute information

architecture/*xx*/**hist/k***xx***.his**
SQL description of agent attribute information

architecture/*xx*/**bin/***your files*
Supplemental Files included from the Java API or Socket data sources with a file type of *executable*. Scripts included from the Script or Command return code data sources.

architecture/*xx*/**lib/***your files*
Supplemental Files included from the Java API or Socket data sources with a file type of Library.

config/.*xx***.rc**
Internal setup file

config/*xx***.config**
Environment settings

config/*xx***_dd_062000000.xml**
Product description

config/*xx***_dd.properties**
Product name

config/.ConfigData/k*xx***env**
Environment variable settings

config/*xx***.ini**
Environment settings

Note: Run the following command to find out the architecture of the system:

```
cinfo -p xx
```

where *xx* is the two-character product code.
For example, for a Solaris 8 64-bit system that is running an agent with product code 19, here is the output:

```
# /opt/IBM/ITM/bin/cinfo -p 19

*********** Fri Aug 17 11:23:58 EDT 2007 *****************
User : root Group: other
Host name : guadalajara  Installer Lvl:06.20.00.00
CandleHome: /opt/IBM/ITM
***********************************************************
Platform codes:
sol286 : Current machine
sol286 : Product (19)
tmaitm6/sol286 : CT Framework (ax)
```

The line in bold is the relevant one. The string before the colon, `sol286` indicates the architecture in use for this agent. This string is different for different combinations of operating system and computer hardware type. The agent must already be installed in order for this feature to work.

The following are the files for the Java based data sources. These files are only created if the agent contains JMX, JDBC, HTTP, or SOAP data sources:

- `cpci.jar`
- `jlog.jar`
- `common/jatlib-1.0.jar`

The following are the files for JMX runtime support. These files are only created if the agent contains JMX data sources:

- `common/jmx-1.0.jar`
- `common/connectors/jboss/connJboss-1.0.jar`
- `common/connectors/jsr160/connJSR160-1.0.jar`
- `common/connectors/was/connWas-1.0.jar`
- `common/connectors/weblogic/connWeblogic-1.0.jar`

The following are the files for JDBC runtime support. These files are only created if the agent contains JDBC data sources:

- `common/jdbc-1.0.jar`

The following are the files for HTTP or SOAP runtime support. These files are only created if the agent contains HTTP or SOAP data sources:

- `http-1.0.jar`

The following are the files for the Java API runtime support. These files are only created if the agent contains a Java API data source:

- `cpci.jar`
- `custom/`*`your JAR file`* The name of this JAR file is specified in the **Global settings** of a Java API data source.
- `custom/`*`your JAR file`* Supplemental Files with a file type of Java resource

The same files exist on both Windows and on UNIX/Linux systems for Java based data sources, but they are in different directories:

- Windows path: `TMAITM6\kxx\jars`
- UNIX/Linux path: *`architecture`*`/`*`xx`*`/jars`

The following are the files for log file Monitoring runtime support. These files are only created if the agent contains log file data sources:

- On Windows systems: TMAITM6\kxxudp.dll
- On Solaris/Linux systems: *architecture*/*xx*/lib/libkxxudp.so
- On HP-UX systems: *architecture*/*xx*/lib/libkxxudp.sl
- On AIX systems: *architecture*/*xx*/lib/libkxxudp.a

The following are the files for SSH Script Monitoring runtime support. These files are only created if the agent contains a script data source that is enabled for SSH collection:

- On Windows systems: `TMAITM6\kxxssh.dll`
- On Solaris/Linux systems: *`architecture`*`/`*`xx`*`/lib/libkxxssh.so`
- On HP-UX systems: *`architecture`*`/`*`xx`*`/lib/libkxxssh.sl`

- On AIX systems: *architecture*/*xx*/lib/libkxxssh.a

Changes in the Manage Tivoli Enterprise Monitoring Services window

After you generate and install the agent that you created with the Tivoli Monitoring Agent Builder, you can see an entry for the agent in the **Manage Tivoli Enterprise Monitoring Services** window. The entry is named **Monitoring Agent for** *agent_name*.

On Windows systems, this entry contains a Task/Subsystem column that identifies whether your agent supports multiple instances:

- A single instance agent displays a new application in the **Manage Tivoli Enterprise Monitoring Services** window named Monitoring Agent for agent_name. A service is created for the agent (Figure 268 on page 396). The Task/Subsystem column contains the value Primary.
- A multiple instance agent displays a new application template in the **Manage Tivoli Enterprise Monitoring Services** window named Monitoring Agent for agent_name. A service is not created for the agent until you create an instance of the agent from this template. The Task/Subsystem column contains the value Template to indicate that this entry is a template that is used to create instances of the agent.

On UNIX systems, the entry for the agent is the same whether your agent supports multiple instances or not.

Note: The following screens are taken from a Windows system. UNIX systems have similar screens.

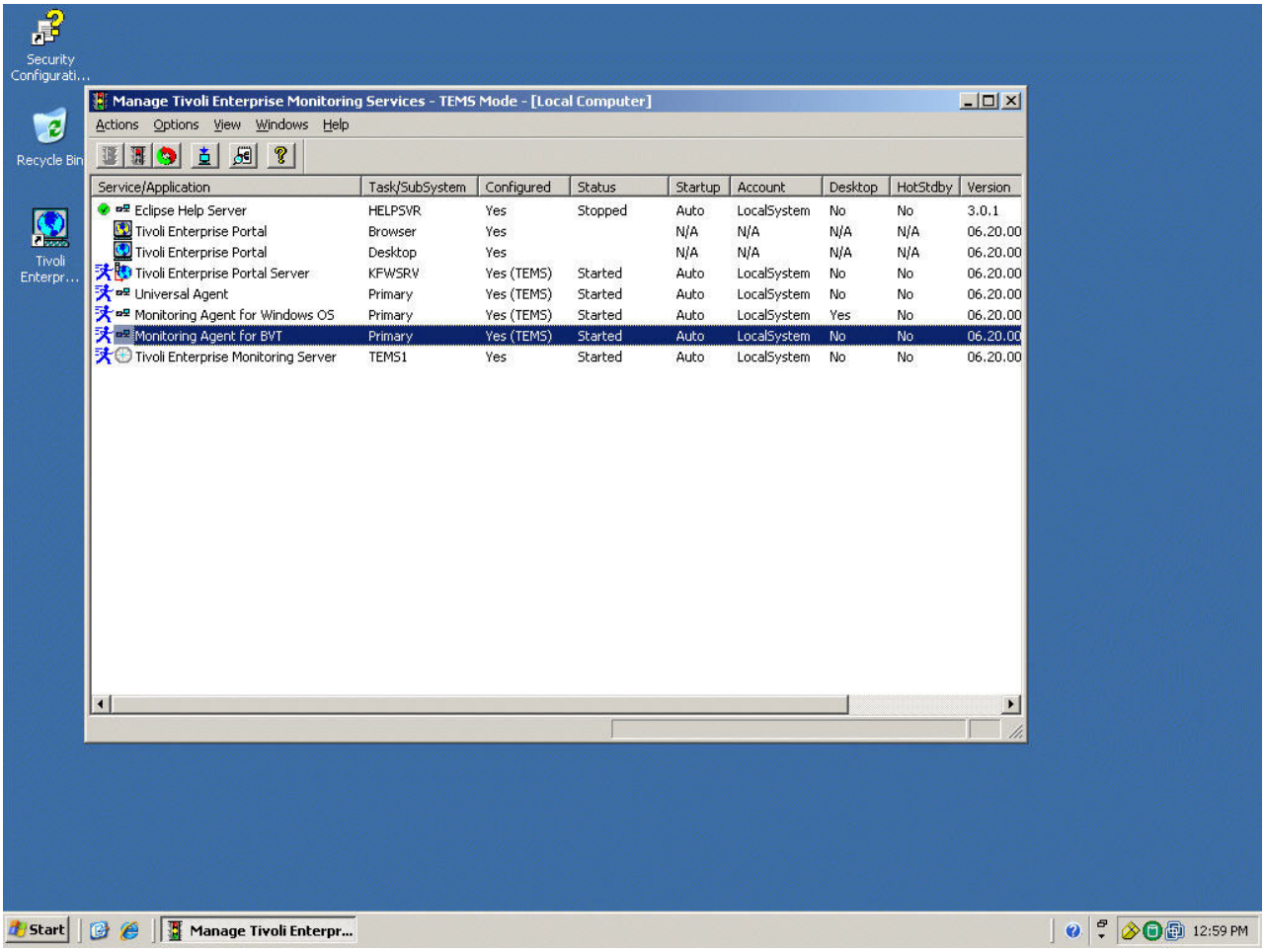


Figure 268. Manage Tivoli Enterprise Monitoring Services window

Changes in the Tivoli Enterprise Portal

After you generate, install, and start the agent, and click the green **Refresh** icon. The new agent that you created with the Tivoli Monitoring Agent Builder can be viewed in the Tivoli Enterprise Portal. You can see the following changes in the portal:

- A new subnode for the agent in the Tivoli Enterprise Portal physical view
- Nodes for every navigator group and top-level data source that you defined by using the Agent Builder (Figure 269 on page 397).

  Note: For each navigator item, you must define a default query.

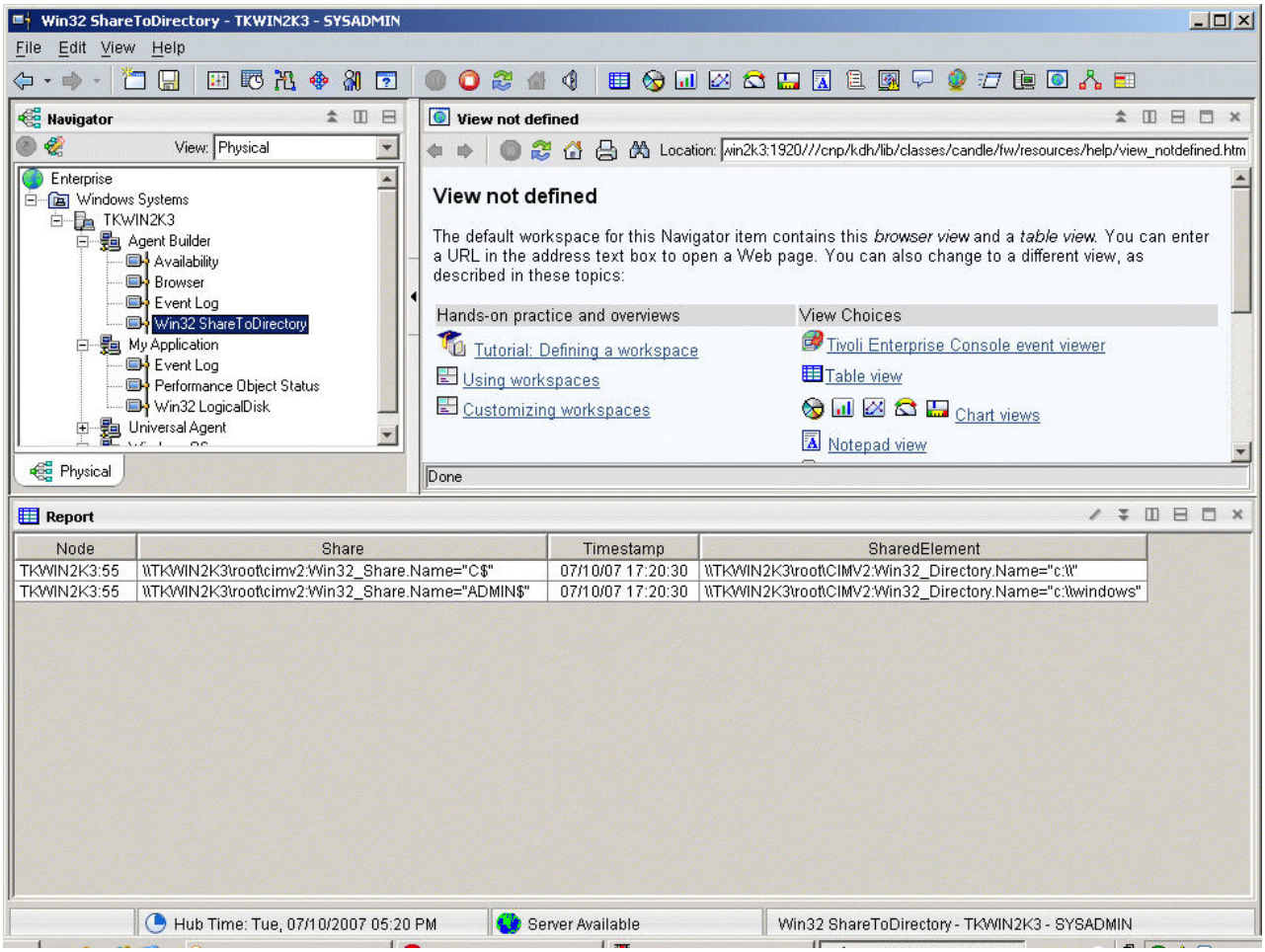


Figure 269. Nodes for attribute groups you defined

- If your agent contains subnodes, an expandable node is present for each subnode that is defined in your agent. The following nodes are shown under the expandable node:
  - xxx performance object status, where xxx is the three-letter subnode type
  - Nodes for every Navigator group and data source that you defined in the subnode
  - xxx event log node if you have event logs
  - xxx JMX monitors node if you have JMX and you included JMX monitors
- The following automatic nodes:
  - An availability node if your agent contains an availability data source (Figure 270 on page 398)

    Note: This node behaves differently depending on the contents of the agent. If the agent monitors only availability, the availability node represents the availability data source. If the agent monitors availability and performance, the availability node becomes the Navigator item that represents the availability, and performance object status data sources.

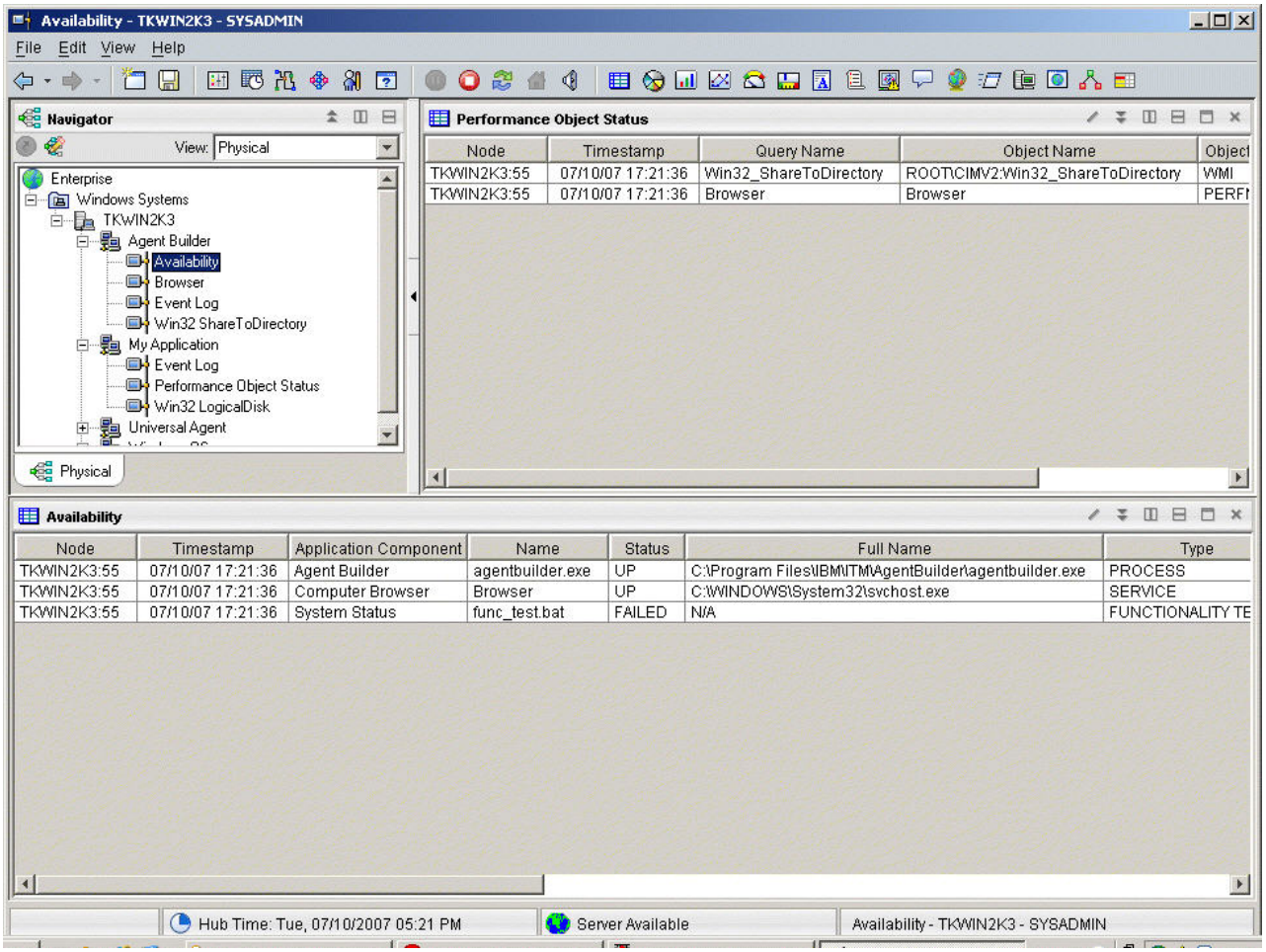


Figure 270. Availability node

– Performance Object Status (if you specified that you want the agent to do performance monitoring and not availability) (Figure 271 on page 399)

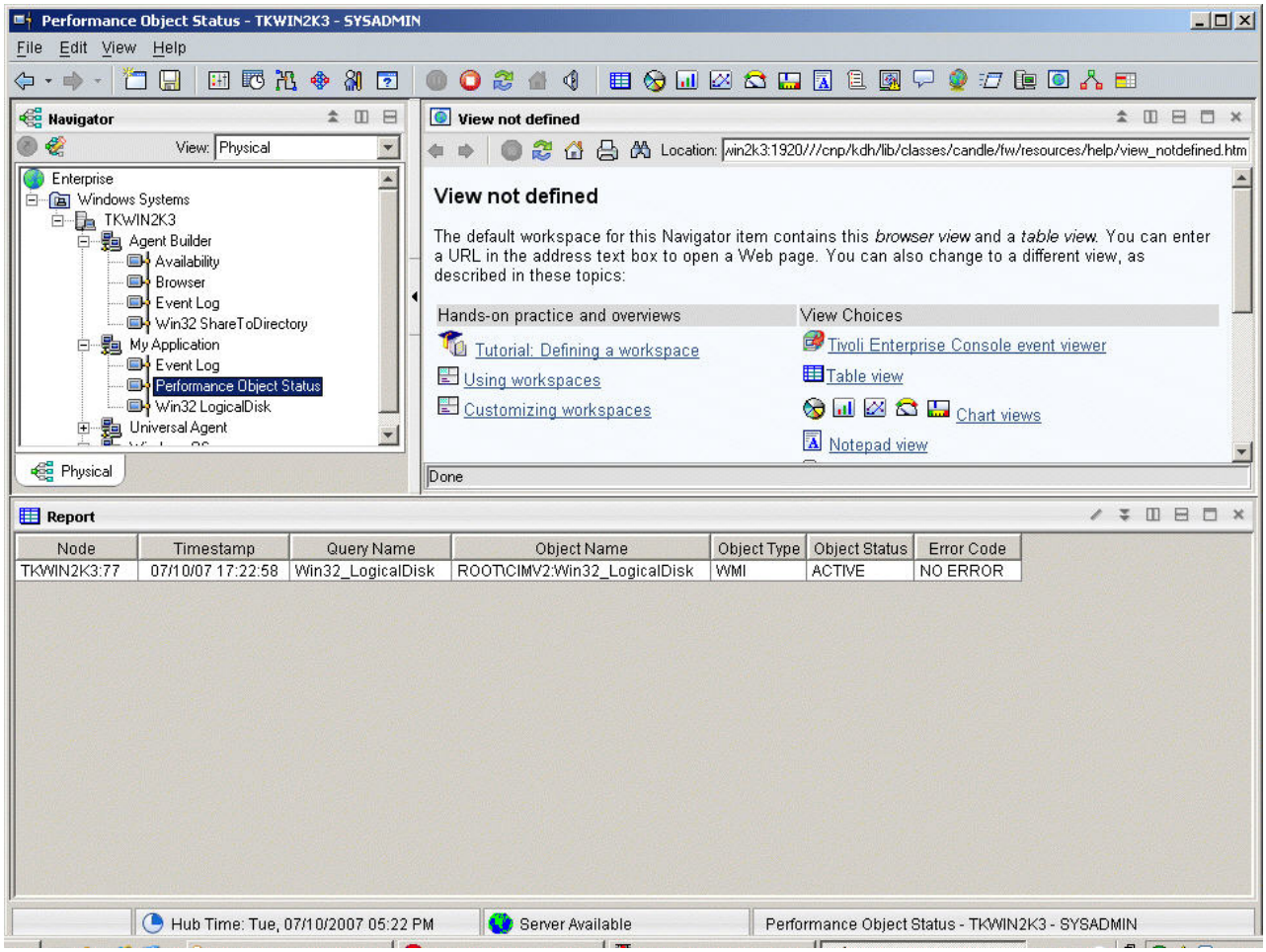


Figure 271. Performance Object Status node

– Event log (if you specified that filter for your agent) (Figure 272 on page 400)

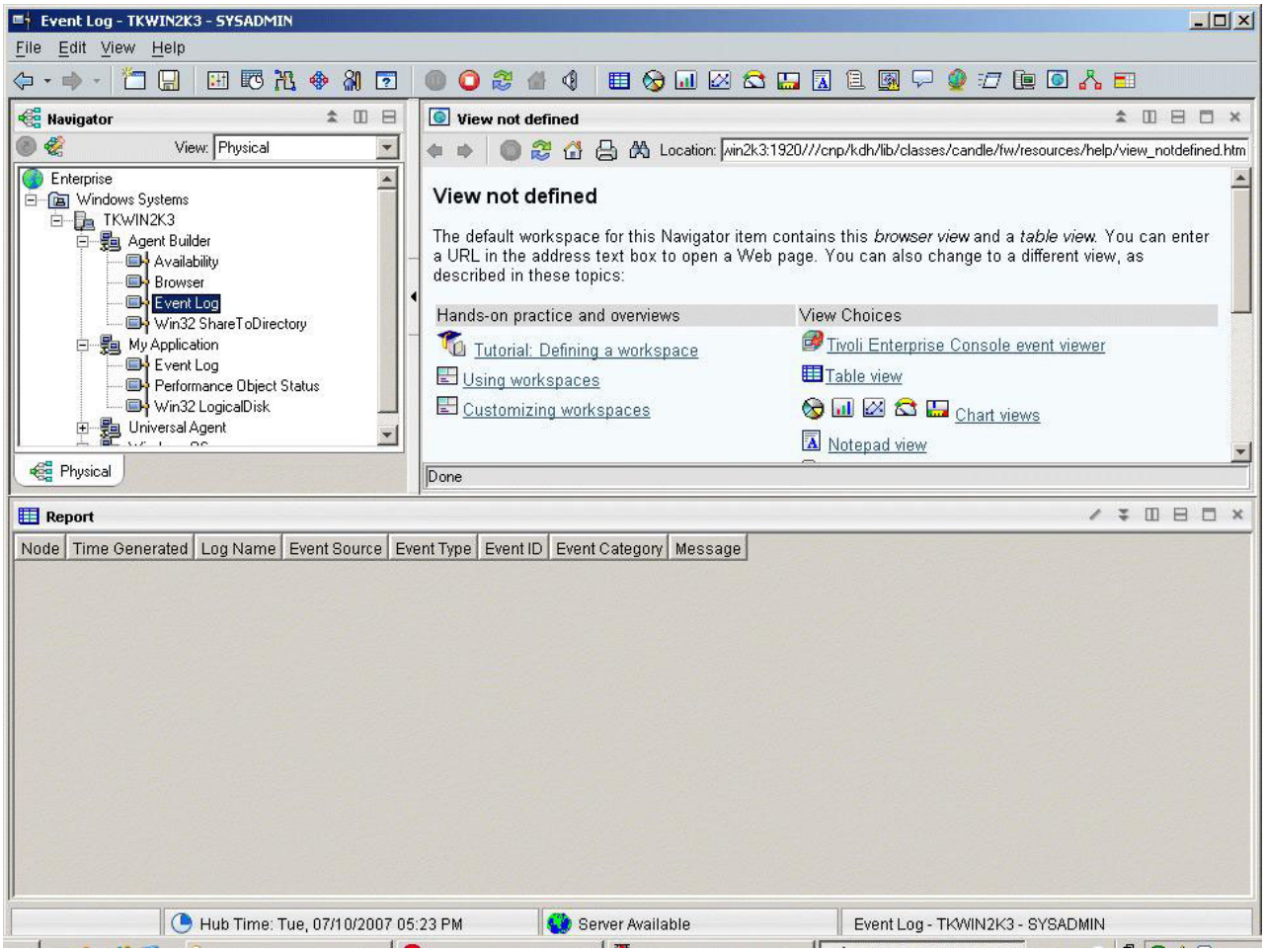


Figure 272. Event log node

See Appendix C, "Attributes reference," on page 541 for descriptions of the attribute groups and attributes for Agent Builder.

Uninstalling an agent

You can use one of two procedures to remove an agent that the Agent Builder generated

About this task

The uninstallation process uninstalls only the agent from the agent system. This process does not uninstall Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, or Tivoli Enterprise Portal Desktop support.

You can use one of the following procedures to remove an agent that the Agent Builder generated:

- "Removing an agent by using the Tivoli Enterprise Portal"
- "Removing an agent without using the Tivoli Enterprise Portal" on page 401

Removing an agent by using the Tivoli Enterprise Portal

You can use the Tivoli(r) Enterprise Portal to remove an agent.

Before you begin

see (“Uninstalling an agent” on page 400).

About this task

To use the Tivoli(r) Enterprise Portal to remove an agent, use the following step:

Procedure

Right-click the agent from the Tivoli Enterprise Portal Navigation tree and select **Remove**.

Note: Your OS agent must be running in order to remove your created agent.

Removing an agent without using the Tivoli Enterprise Portal

You can use Operating System scripts and commands to remove an agent when you do not have a Tivoli Enterprise Portal available.

Before you begin

See (“Uninstalling an agent” on page 400).

About this task

To remove an agent that the Agent Builder generated from the target system, when you do not have a Tivoli Enterprise Portal, use the following commands:

Procedure

Windows systems

- Command:

  ```
  cd ITM_INSTALL/TMAITM6
  kxx_uninstall.vbs ITM_INSTALL
  ```

  where *xx* is the product code for the agent
- Uninstallation script:

  If you want to run the k*xx*_uninstall.vbs script from a script or from the command line, use the **cscript** k*xx*_uninstall.vbs command. cscript.exe is the command-line interface parser for vbs scripts and does not display a window; instead, a message is displayed on the console.

UNIX systems

- Use the uninstall.sh file that is found in *ITM_INSTALL*/bin:

  ```
  uninstall.sh [-f] [-i] [-h ITM_INSTALL] [product platformCode]
  ```

Clearing the agent from the Tivoli Enterprise Portal

After you remove the agent, clear the agent from the Tivoli Enterprise Portal.

Before you begin

Remove the agent. For more information, see (“Uninstalling an agent” on page 400).

Procedure

1. Ensure that your Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server are up and running.

2. Log on to your Tivoli Enterprise Portal client.
3. From the Tivoli Enterprise Portal client Physical Navigator views, right-click **Enterprise** and select **Workspace** > **Managed System Status**. The Managed System Status workspace is displayed.
4. Select all of the IBM Tivoli Managed Systems for your agent.
5. Right-click and select **Clear off-line entry**, which clears all of the entries from that table.

Chapter 32. Creating workspaces, Take Action commands, and situations

Prepare to create workspaces, queries, Take Action commands, and situations.

After you create, install, and test your agent solution, you can prepare to create workspaces, queries, Take Action commands, and situations for the solution.

The situations, workspaces, Take Action commands, and queries that you create can be included in the installation package. To have one installation image for situations, workspaces, and the agent itself, the situation, and workspace files must be in the same project as the agent. The Agent Builder provides a wizard to create the appropriate files in the agent project. For information about importing application support files, see Chapter 33, “Importing application support files,” on page 409.

Creating situations, Take Action commands, and queries

Find information to help create situations, Take Action commands, and queries.

To create situations, Take Action commands, and queries, use the Tivoli Enterprise Portal and the embedded Situation editor. For detailed information about how to create situations and queries, see the Tivoli Enterprise Portal User's Guide. The User's Guide is online at the IBM Tivoli Monitoring Information Center. You can also use the help documentation that is installed with your Tivoli Enterprise Portal Server. An Agent Builder monitoring agent can recognize and perform special processing for a set of specific Take Action commands. For more information about these special Take Action commands, see (Appendix K, “Take Action commands reference,” on page 651).

Situations for system monitor agents are created differently from the Enterprise situations that are created with the Tivoli Enterprise Portal Situation editor or the **tacmd createSit** command. For system monitor agents, private situations are created in a local private situation configuration XML file for the agent. For information about creating situations for system monitor agents, see "Private situations" in the "Agent Autonomy" chapter of the *IBM Tivoli Monitoring Administrator's Guide*.

Creating workspaces

Place the Tivoli Enterprise Portal in the “Administrator” mode to create workspaces that you can export and include in your solution.

About this task

Build the workspaces in the environment from which they are used. When you build workspaces, change the display settings on your computer to build workspaces at the minimum resolution that is normally used in your environment. Building workspaces at a greater resolution can create views that are too cluttered to be used reasonably at lesser resolutions.

To create workspaces that you can export and include in your solution, the Tivoli Enterprise Portal must be placed in the "Administrator" mode. To place the Tivoli Enterprise Portal in "Administrator" mode, use the following steps:

Procedure

1. Go to the *ITM_INSTALL*/CNP directory and open the cnp.bat file. If you used the default installation, the directory is C:\IBM\ITM\CNP. In the cnp.bat file, you must update the set _CMD= %_JAVA_CMD% line to include option -Dcnp.candle.mode="$_KCJ_$".

© Copyright IBM Corp. 2007, 2013

If you want to create extensions on Linux or AIX systems, use the following path:

`/opt/IBM/ITM/li263/cj/bin/cnp.sh`

Where *li263* is the operating system on which the Tivoli Enterprise Portal is running. The updated `set _CMD= %_JAVA_CMD%` looks similar to the following example:

```
set _CMD= %_JAVA_CMD% -Dcnp.candle.mode="$_KCJ_$" -Xms64m -Xmx256m -showversion -noverify
 -classpath %CPATH% -Dkjr.trace.mode=LOCAL -Dkjr.trace.file=C:\IBM\ITM\CNP\LOGS\kcjras1.log
-Dkjr.trace.params=ERROR -DORBtcpNoDelay=true -Dibm.stream.nio=true
-Dice.net.maxPersistentConnections=16 -Dice.net.persistentConnectionTimeout=1
-Dcnp.http.url.host=SKINANE -Dvbroker.agent.enableLocator=false  -Dnv_inst_flag=%NV_INST_FLAG%
-Dnvwc.cwd=%NVWC_WORKING_DIR% -Dnvwc.java=%NVWC_JAVA% candle.fw.pres.CMWApplet
```

Note: The command is shown here on multiple lines for formatting reasons only.

2. Open a new Tivoli Enterprise Portal Client, and log in with the `sysadmin` user ID.
3. Set the "sysadmin" user ID in “Administrator” mode. In the Tivoli Enterprise Portal, select **Edit** > **Administer Users**. Select **sysadmin** and then under the **Permissions** tab, select **Workspace Administration**. Select the **Workspace Administration Mode** check box. If you make the selection correctly, *ADMIN MODE* is displayed in the desktop title bar.

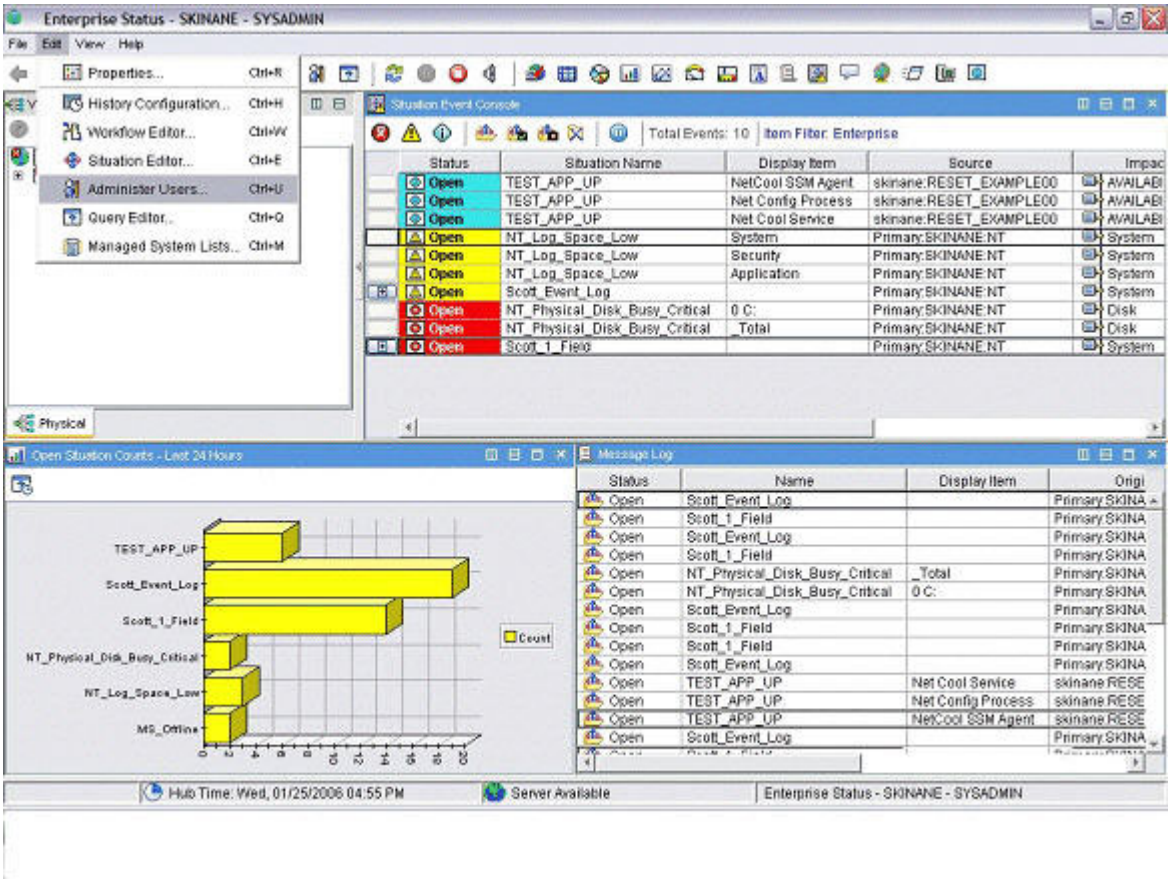


Figure 273. Setting the `sysadmin` user ID

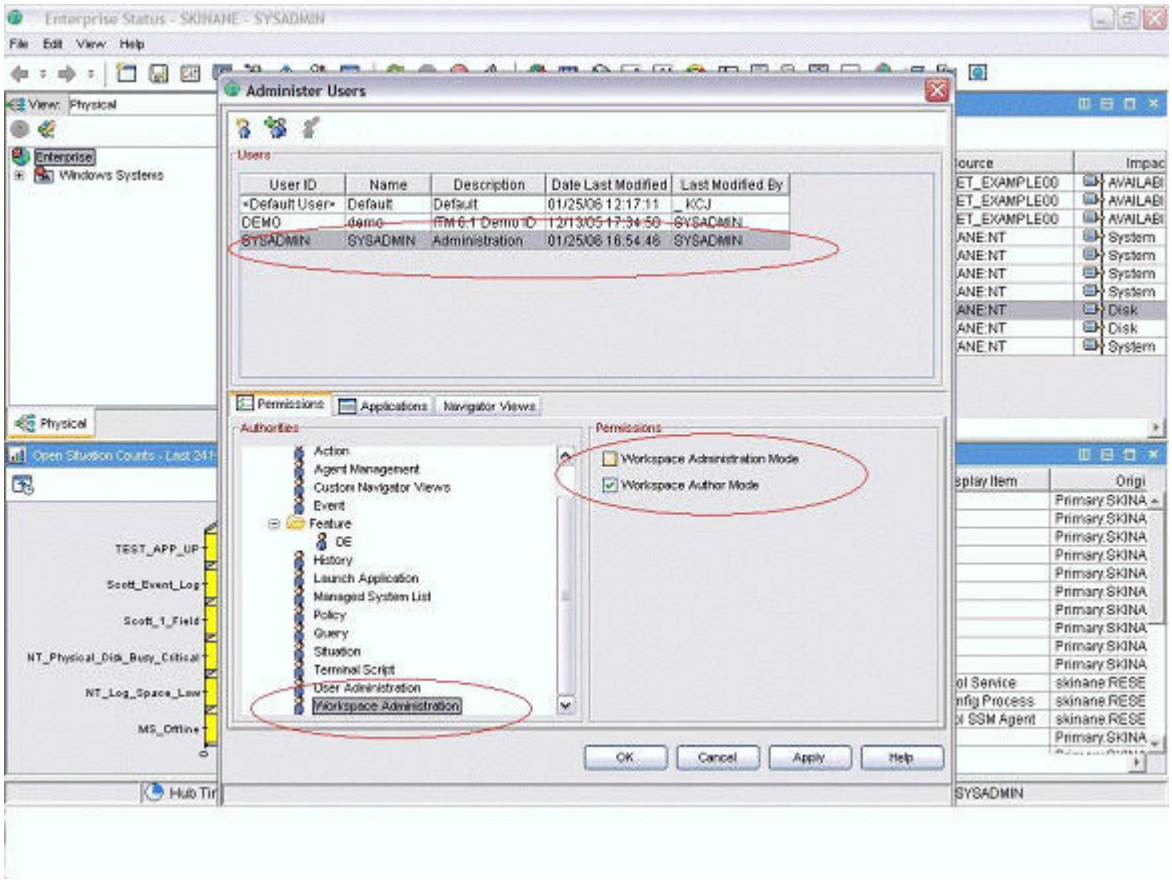


Figure 274. Setting the `sysadmin` user ID (continued)

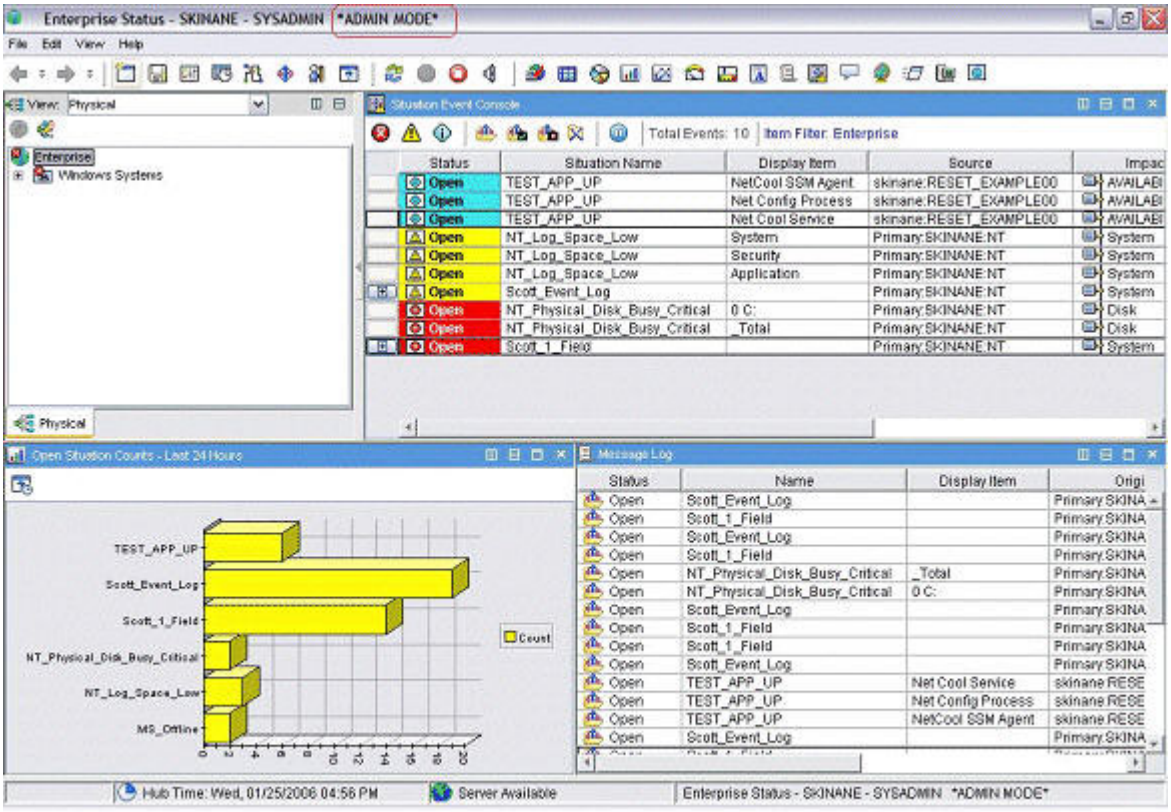


Figure 275. Setting the `sysadmin` user ID (continued)

What to do next

After you are in "Administrator" mode as depicted in (Figure 275), you are now ready to create workspaces for your application. For information about how to customize and create workspaces, see the Tivoli Enterprise Portal User's Guide online at the IBM Tivoli Monitoring Information Center. Alternatively, use the help documentation that is installed with your Tivoli Enterprise Portal component.

If you want your workspaces to be "read-only" and to not be deleted by a customer, set the "not-editable" and "non-deletable" properties for each workspace. In the workspace properties, you must select the following properties:

- **Do not allow modifications**
- **Product-provided by IBM (mark as non-deletable)**

You can go to the properties by either viewing a workspace or clicking the icon with the controls on it. You can also go to one of the view property pages and then going to the workspace level in the properties tree. If you have more than one workspace for each navigator item, remember to set the properties for each workspace. As indicated in the following example screen capture:

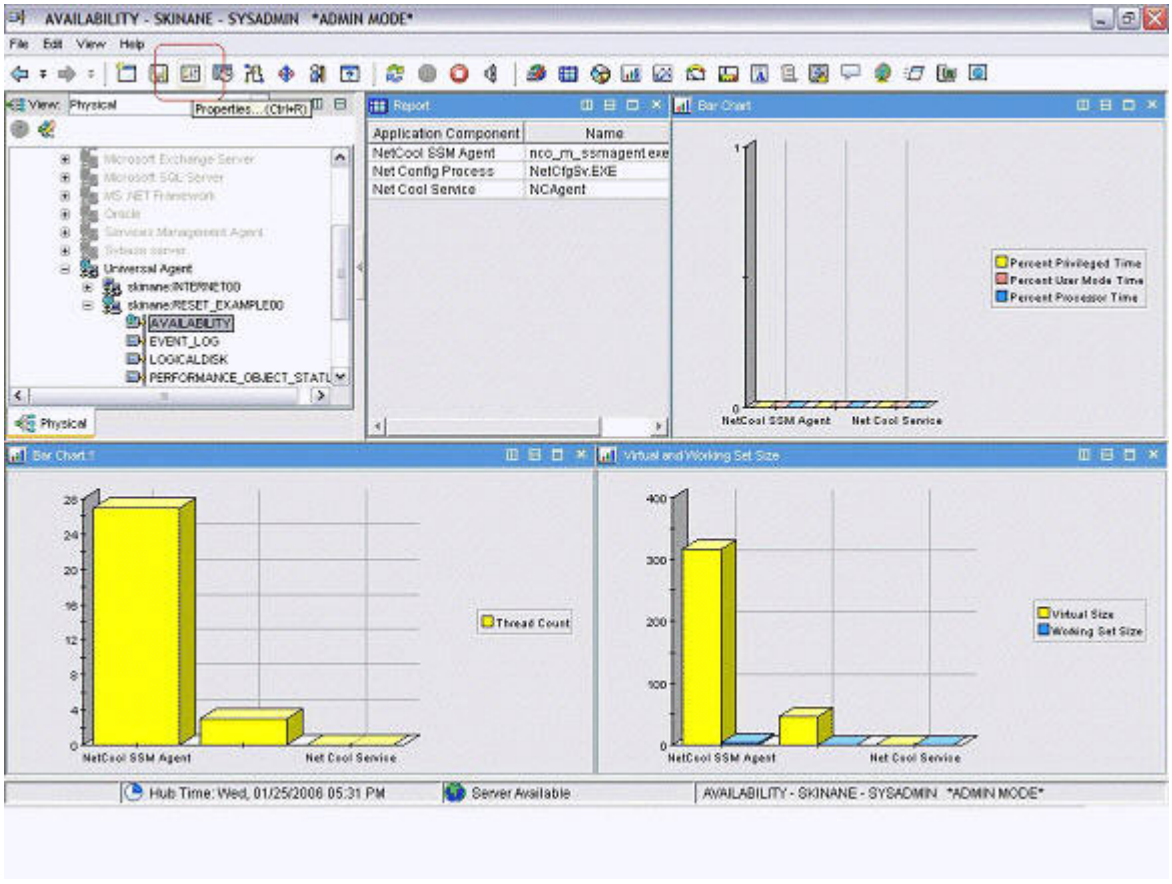


Figure 276. Setting workspace properties

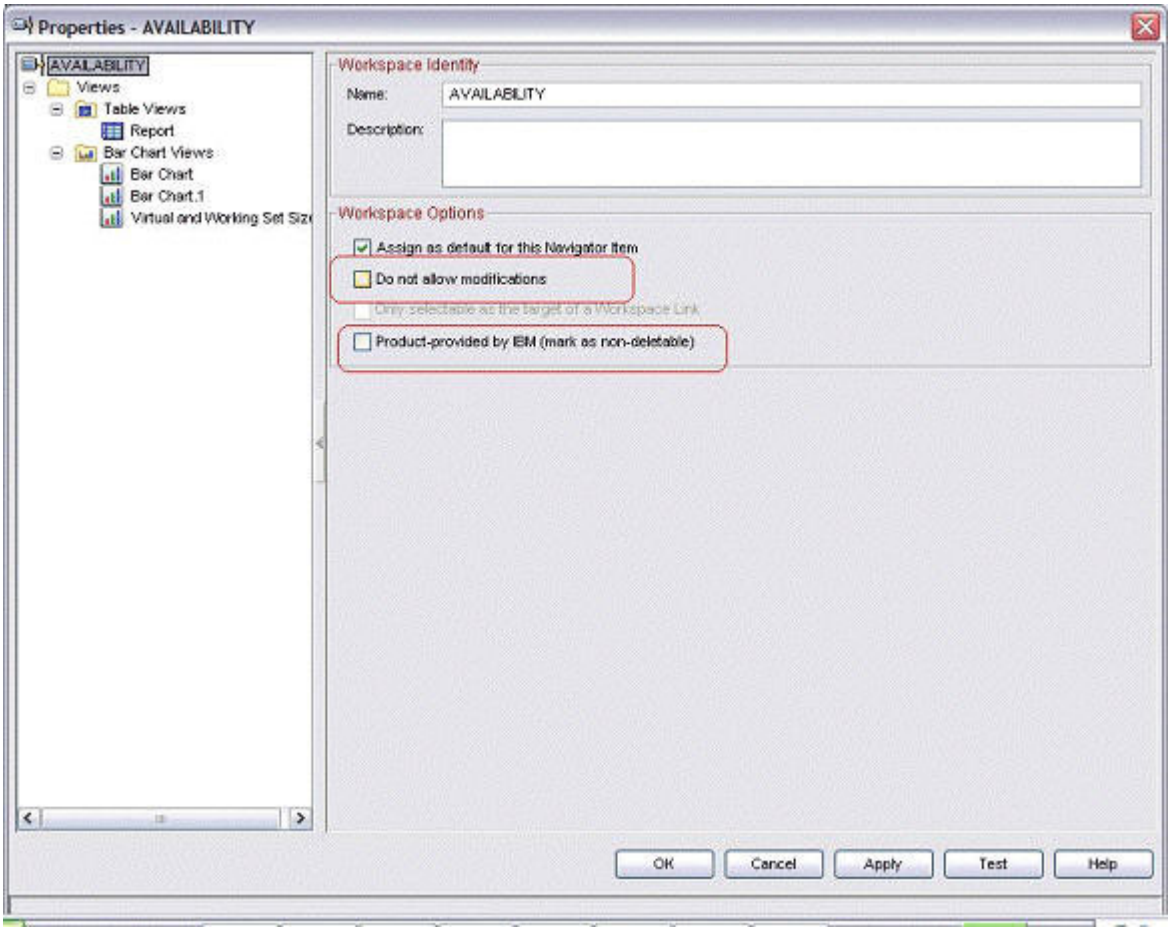


Figure 277. Setting workspace properties (continued)

Chapter 33. Importing application support files

About this task

Custom situations, workspaces, Take Action commands, and queries can also be included in the installation package. To have a single installation image for situations, workspaces, and the agent, the situation, and workspace files must be in the same project as the agent. The Agent Builder provides a wizard to create the appropriate files in the agent project.

Definitions that are associated with an agent can also be included in the installation package. The content of these definitions is different for an agent that is used in an enterprise monitoring environment and in a system monitor environment. An enterprise monitoring agent image can include custom situations, workspaces, Take Action commands and queries. A system monitor agent image can include private situations, trap definitions, and agent configuration information.

To have a single installation package that includes the appropriate definitions and the agent itself, the files must be in the same project as the agent. The Agent Builder provides a wizard to create the appropriate files for an enterprise monitoring installation. The files for a system monitor agent environment are created by using the process that is described in the *Agent Autonomy* chapter in the *IBM Tivoli Monitoring Administrator's Guide*. The resulting files are copied into the root of the Eclipse project for the agent.

Exporting and importing files for Tivoli Enterprise Monitoring Agents

About this task

After you create situations, workspaces, queries, and Take Action commands in the Tivoli Enterprise Portal, you can export and import them into another Tivoli Monitoring Version 6.2 environment. For more information about creating situations and workspaces, see (Chapter 32, "Creating workspaces, Take Action commands, and situations," on page 403). Use the following steps to extract the situations, workspaces, Take Action commands, and queries:

Procedure

1. From the **Project Explorer** tab, right-click the agent project folder.
2. Select **IBM Tivoli** > **Import Application Support Files**.

© Copyright IBM Corp. 2007, 2013

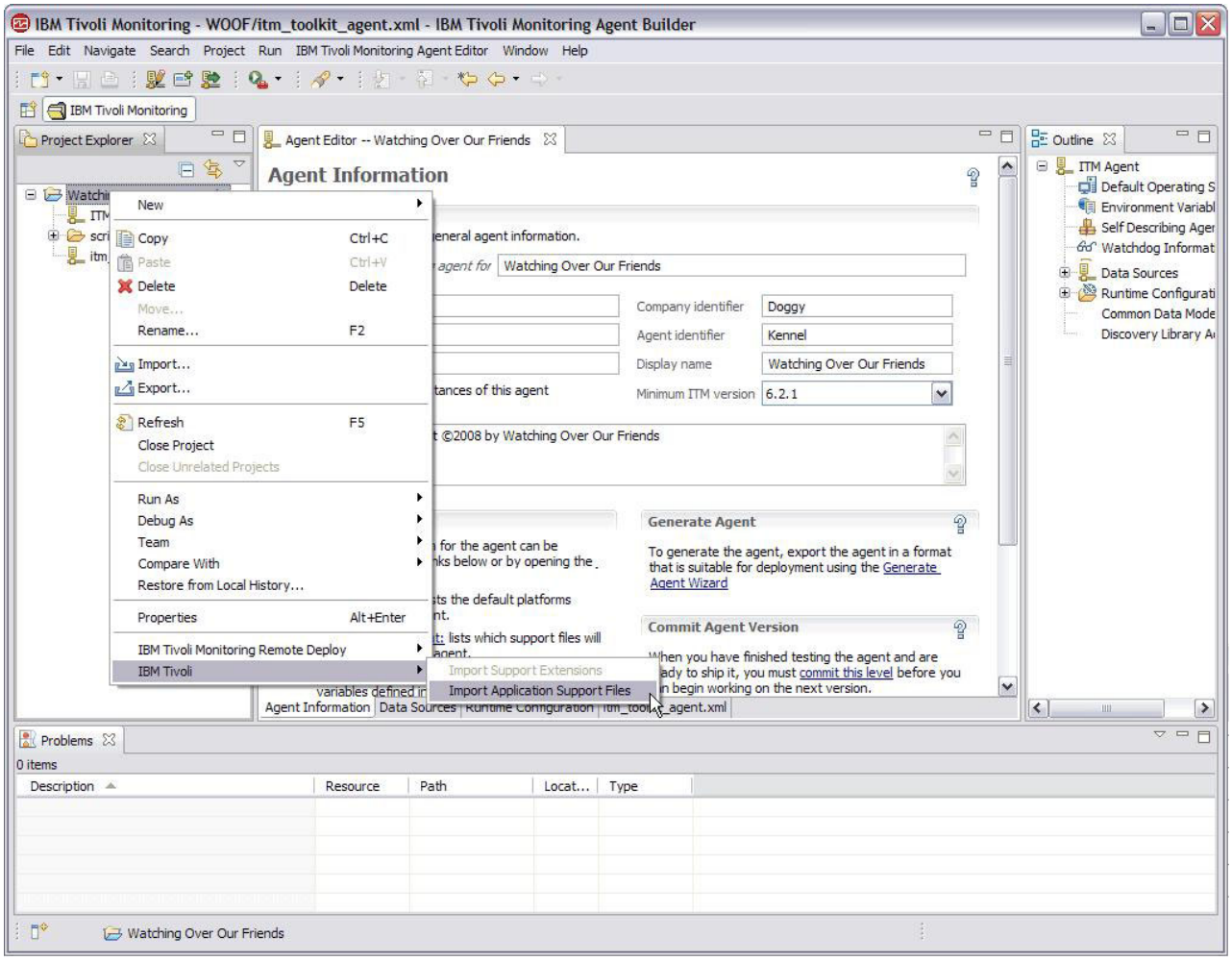


Figure 278. Importing application support files

3. Enter the host name of the Tivoli Enterprise Portal Server.
4. Enter the user name and password for the Tivoli Monitoring environment you are connecting to.
5. If you defined situations for your agent, a dialog box is presented that lists the situations that are defined for the agent.

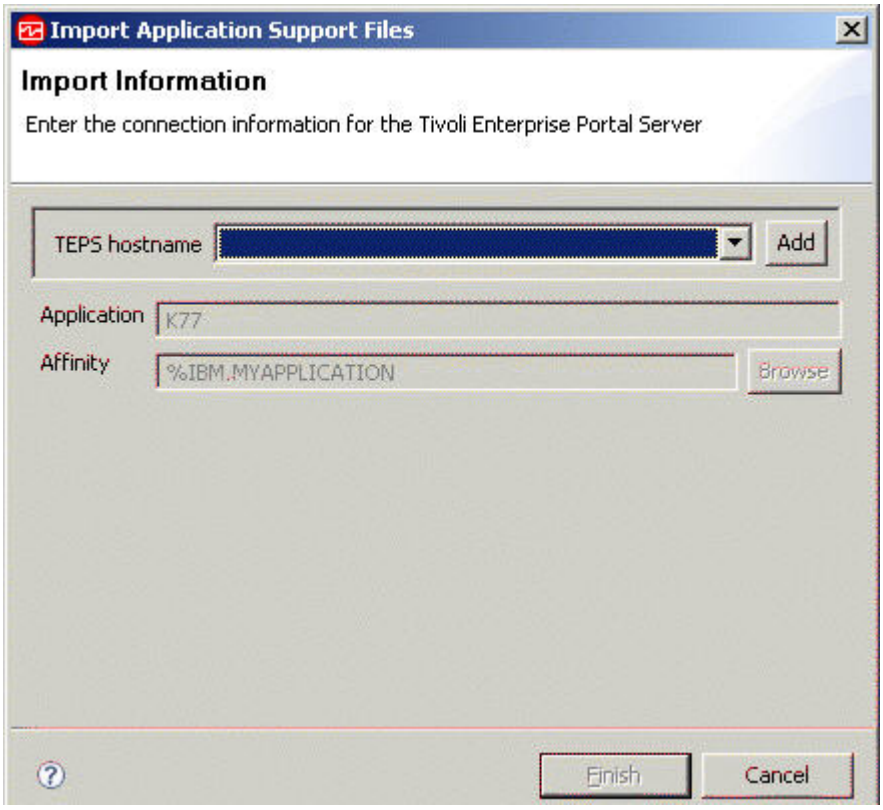


Figure 279. Importing situations

6. Select the situations that you want to export from the list and click >> to add them to the selected situations table and click **OK**.

   The import might take a few moments. When the task completes, you see the SQL files in the appropriate folders in the agent project.

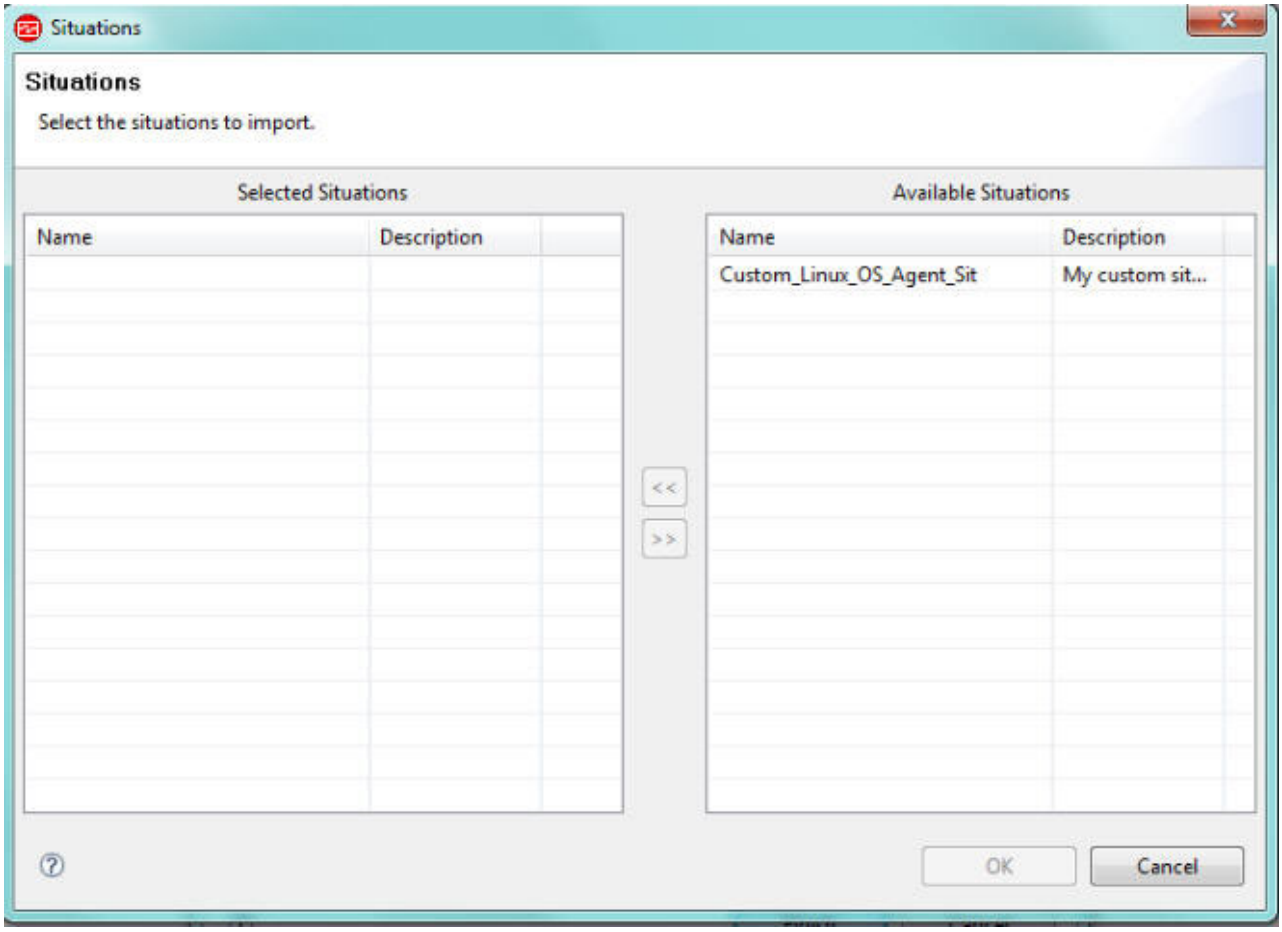


Figure 280. Selecting situations

7. If you defined Take Action commands for your agent, a dialog presents the Take Action commands defined. Choose the Take Action commands that you want to export from the list and click >> to add them to the Selected Take Actions table and click **OK**.

   The import might take a few moments. When the task completes, you see the SQL files in the appropriate folders in the agent project.

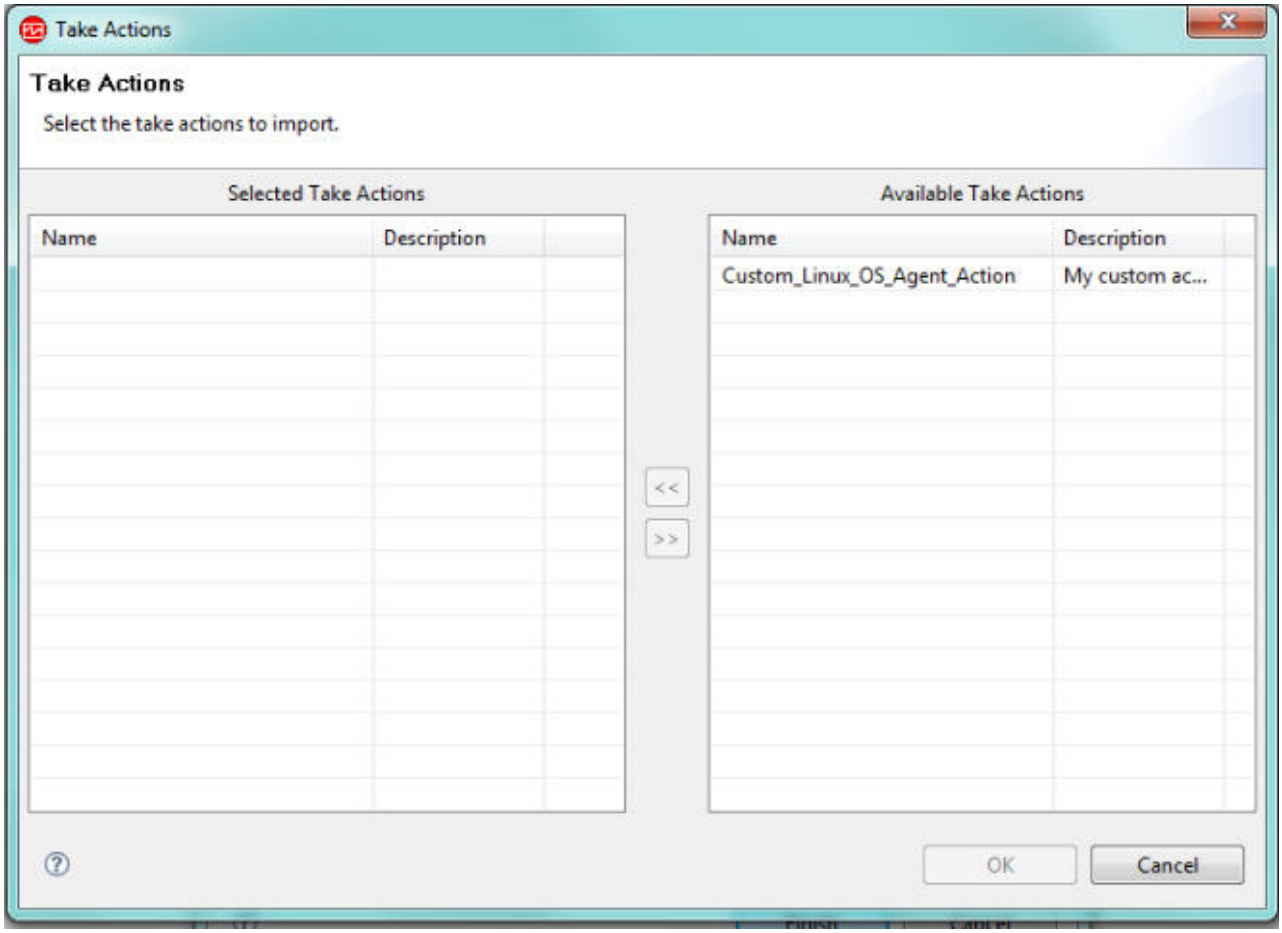


Figure 281. Take Actions window

8. If you defined custom queries for your agent, a dialog presents the Take Action commands defined. Select the queries that you want to export from the list and click >> to add them to the Selected Queries table and click **OK**.

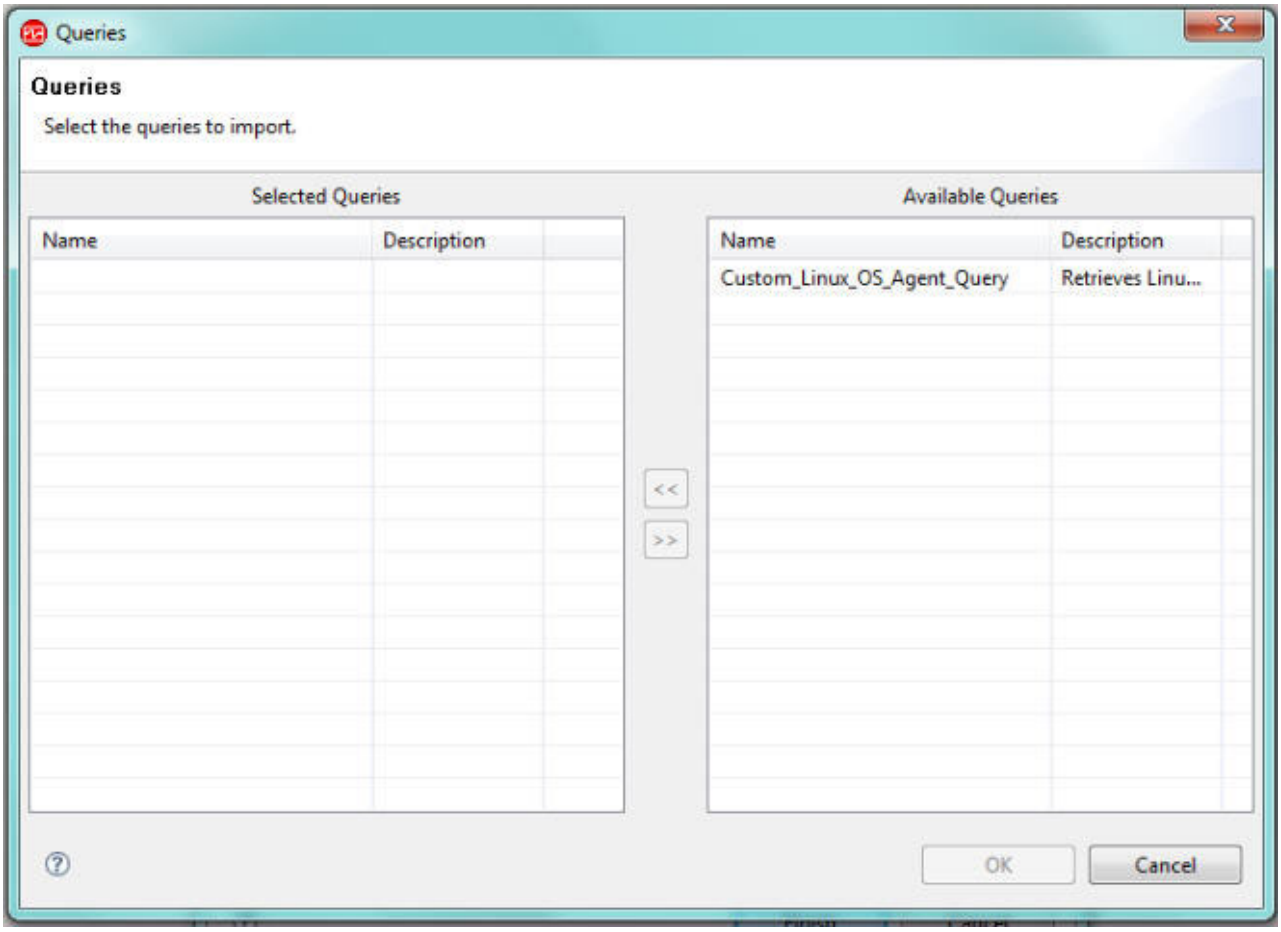


Figure 282. Importing queries

The import might take a few moments. When the task completes, you see the SQL files in the appropriate folders in the agent project.

Note: Workspaces are imported automatically.

Exporting and importing files for Tivoli System Monitor Agents

About this task

The system monitor agent definitions are contained in three types of files:

- Private situations are defined in a file named *xx*`_situations.xml`, where *xx* is the two-character product code
- Trap configuration information is defined in a file named *xx*`_trapcnfg.xml`, where *xx* is the two-character product code
- For agents that require configuration, the configuration is defined in one file for each instance of the agent. When the agent is a single instance agent, the file is named *xx*`.cfg`. When the agent is a multi-instance agent, there is a file is present for each instance. The file names are *xx*`_`*instance name*`.cfg`, where *xx* is the two-character product code and *instance name* is the name of the agent instance.

Procedure

Create the files by using the process that is described in the *Agent Autonomy* chapter in the *IBM Tivoli Monitoring Administrator's Guide*. Copy the files into the root of the project directory manually, or use the Eclipse import function to select the files to be imported: **File** > **Import** > **General** > **File System**. These files are included in the agent image and installed by the installer.
When the agent is installed the installation:

- Copies the included files into the appropriate locations.
- Any private situations that are defined in the `pc_situations.xml` file that is run on the agent.
- The trap definitions that are defined in the `pc_trapcnfg.xml` are used to forward traps that are based on the situations.
- The agent is automatically configured and started if:
  - The agent is a single instance agent with no configuration defined as part of the agent.
  - The agent is a single instance agent with configuration defined as part of the agent and the image includes a `pc.cfg`file.
  - The agent is a multi-instance agent (all multi-instance agents require configuration): the installer starts one instance of the agent for each `pc_`*`inst`*`.cfg` file.

Chapter 34. Event filtering and summarization

An attribute group is defined to be *pure event* or *sampled*. Pure event attribute groups contain data rows that occur asynchronously. As each new row of data arrives, it is processed immediately by Tivoli Monitoring. Sampled attribute groups collect the current set of data rows each time the data is requested. The following attribute groups illustrate the difference:

- An `SNMPEvent` attribute group is created that represents all of the SNMP Traps and informs that are sent to the agent. Traps or informs arrive asynchronously as they are sent by the monitored systems. As each event arrives, it is passed to Tivoli Monitoring.
- A `Disk` attribute group is created to represent information about all of the disks on a system. The disk information is collected periodically. Each time disk information is collected, the agent returns a number of rows of data, one for each disk.

The difference between pure event and sampled attribute groups affects various aspects of Tivoli Monitoring. These aspects include: situations, warehouse data, and Tivoli Enterprise Portal views.

Each situation is assigned (or *distributed*) to one or more managed systems to be monitored for a specific condition of a set of conditions. When the determination of the event must be made based on observations that are made at specific intervals, the event is known as a *sampled event*. When the event is based on a spontaneous occurrence, the event is known as a *pure event*. Therefore, situations for sampled events have an interval that is associated with them, while situations for pure events do not. Another characteristic of sampled events is that the condition that caused the event can change, thus causing it to be no longer true. Pure events cannot change. Therefore, alerts that are raised for sampled events can change from true to false, while a pure event stays true when it occurs.

An example of a sampled event is `number of processes > 100`. An event becomes true when the number of processes exceeds 100 and later becomes false again when this count drops to 100 or less. A situation that monitors for `invalid logon attempt by user` is a pure event; the event occurs when an invalid logon attempt is detected, and does not become a False event. While you can create situations that are evaluated on a specific interval for sampled attribute groups, such evaluations are not possible for pure event attribute groups.

Similarly, for historical data, you can configure how frequently sampled data is collected. However, when you turn collection on for pure event data, you get each row as it happens.

The data that is displayed in the Tivoli Enterprise Portal for sampled data is the latest set of collected rows. The data that is displayed for pure event attribute groups is the contents of a local cache that is maintained by the agent. It does not necessarily match the data that is passed to Tivoli Monitoring for situation evaluation and historical collection.

Controlling duplicate events

Use the event filtering and summarization options to control how duplicate events are sent to Tivoli Monitoring.

Before you begin

For more information about event filtering and summarization, see Chapter 34, “Event filtering and summarization.”

© Copyright IBM Corp. 2007, 2013

About this task

The Agent Builder defines attribute groups that represent event data as *pure event* in Tivoli Monitoring. These attribute groups include log file, AIX Binary Log, SNMP events, and JMX notifications. These attribute groups can produce multiple duplicate events. You can control how these duplicate events are sent to Tivoli Monitoring. You can activate these controls for log file, SNMP events, and JMX notifications attribute groups in the **Event Information** tab under **Advanced Data Source Properties** in the **Advanced** window (Figure 283 on page 419).

Whether an event is treated as a duplicate of other events is determined by the key attributes, you define in the attribute group. A duplicate event occurs when the values for all key attributes in the event match the values for the same key attributes in an existing event. When event filtering and summarization is enabled, the attributes for the `isSummary`, `occurrenceCount`, `summaryInterval`, and `eventThreshold` functions are added automatically.

Procedure

In the **Event Filtering and Summarization Options** area, select one of the following options:

- **No event filtering or summarization**: Sends all events without any event filtering or summarization. This option is the default option.
- **Filter and summarize events**: Creates a summary record for each event with duplicates and each unique event that is based on the key attributes. Select also to choose the event filtering option. In the **Summarization Options** area, enter the summary interval. You can enter either a value in seconds or insert a configuration property.

  The event filtering options are:

  – **Only send summary events**: Sends only the summary records for the specified interval.
  – **Send all events**: Sends all events and summary records.
  – **Send first event**: For each event, sends only the first event that is received in the summary interval that is specified and no duplicate events. This option also sends the summary records.
  – **Event threshold**: Sends an event to Tivoli Monitoring when the number of duplicate events that are received in the interval is evenly divisible by the threshold. For example, if you set the event threshold to 5 and you receive less than five duplicates (including the first event) in the interval, no event is sent to Tivoli Monitoring. If you receive 5, 6, 7, 8, or 9 duplicates, one event is sent. If you receive 10 duplicates, 2 events are sent. In the **Event threshold** field, you can enter a number or insert a configuration property. This option also sends the summary records.

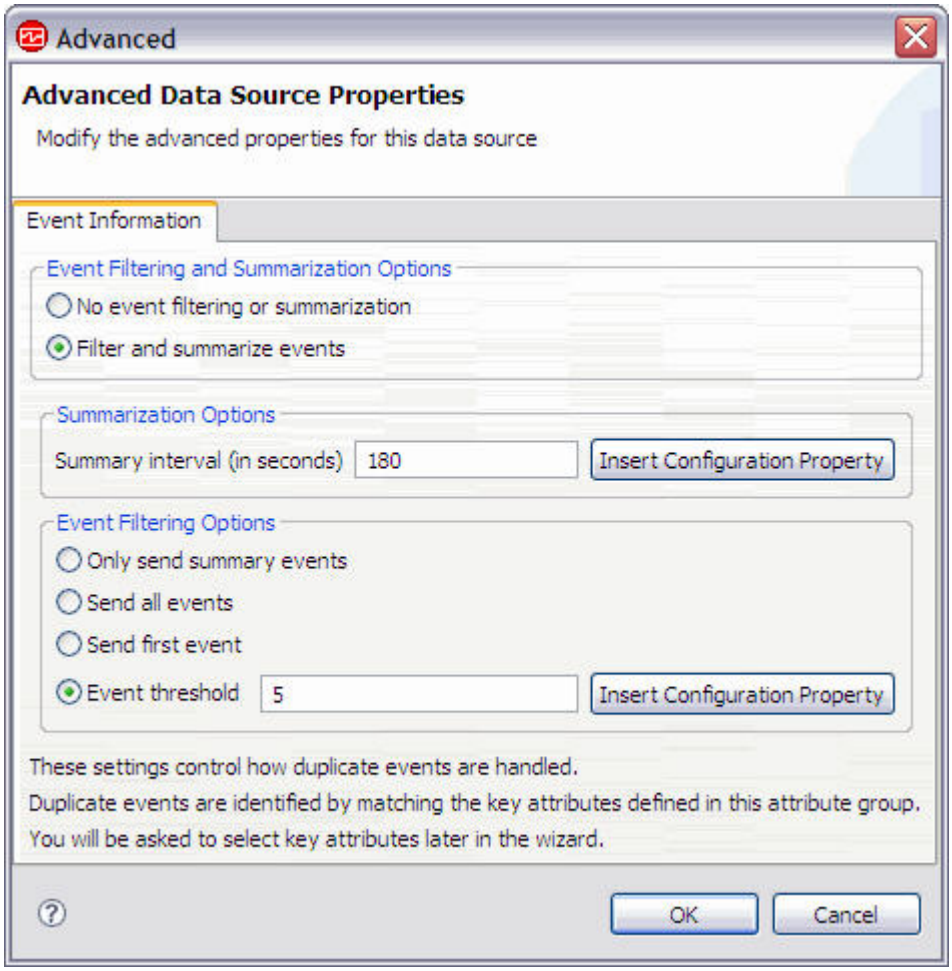


Figure 283. Advanced Data Source Properties page.

Viewing event filtering and summarization in the Tivoli Enterprise Portal

Examples of how data is treated depending on your event filtering and summarization choices.

The agent maintains a cache of the last events received. By default, this cache is 100 in size. If you enable agent event filtering and summarization, differences can occur between the number of events in the cache and the number sent to IBM Tivoli Monitoring. Additional events in the cache might not reach the designated threshold for sending. Or you might have fewer events in the cache if you selected the **Send all events** option. If the **Send all events** option is set, an event is sent each time a duplicate occurs. However, only one copy of the event is kept in the cache, and the occurrence count is incremented each time that the event occurs. To view the events that are sent to IBM Tivoli Monitoring, create a historical view. For information about creating historical views, see Historical Reporting. You can compare this view with the real-time cache view in the Tivoli Enterprise Portal. You can also use situations to make the same comparison.

The following examples indicate how the same log data is treated depending on your choice, if any, of event filtering and summarization. The example agent was created to illustrate different behaviors. Each attribute group was defined to monitor the same log file. In each example, a historical view and a real-time (cache) view is shown. The names of the nodes in the Tivoli Enterprise Portal reflect the settings

selected. By default, the historical view displays the newest events last. The default real-time view of the cache displays the newest events first. In these examples, the historical view shows the last 1 hour.

As new events arrive, you can see them in the cache view. As duplicates of an event arrive, the data is updated in the existing row. When a summary interval elapses, the existing events are converted to summary events and sent. New rows are then added for the next summary interval.

(Figure 284) shows the historical view and cache view if you did not enable event filtering or summarization. Both views display the same data, but in reverse order. To display the corresponding events, the historical view is scrolled down and the real time (cache) view is scrolled up.

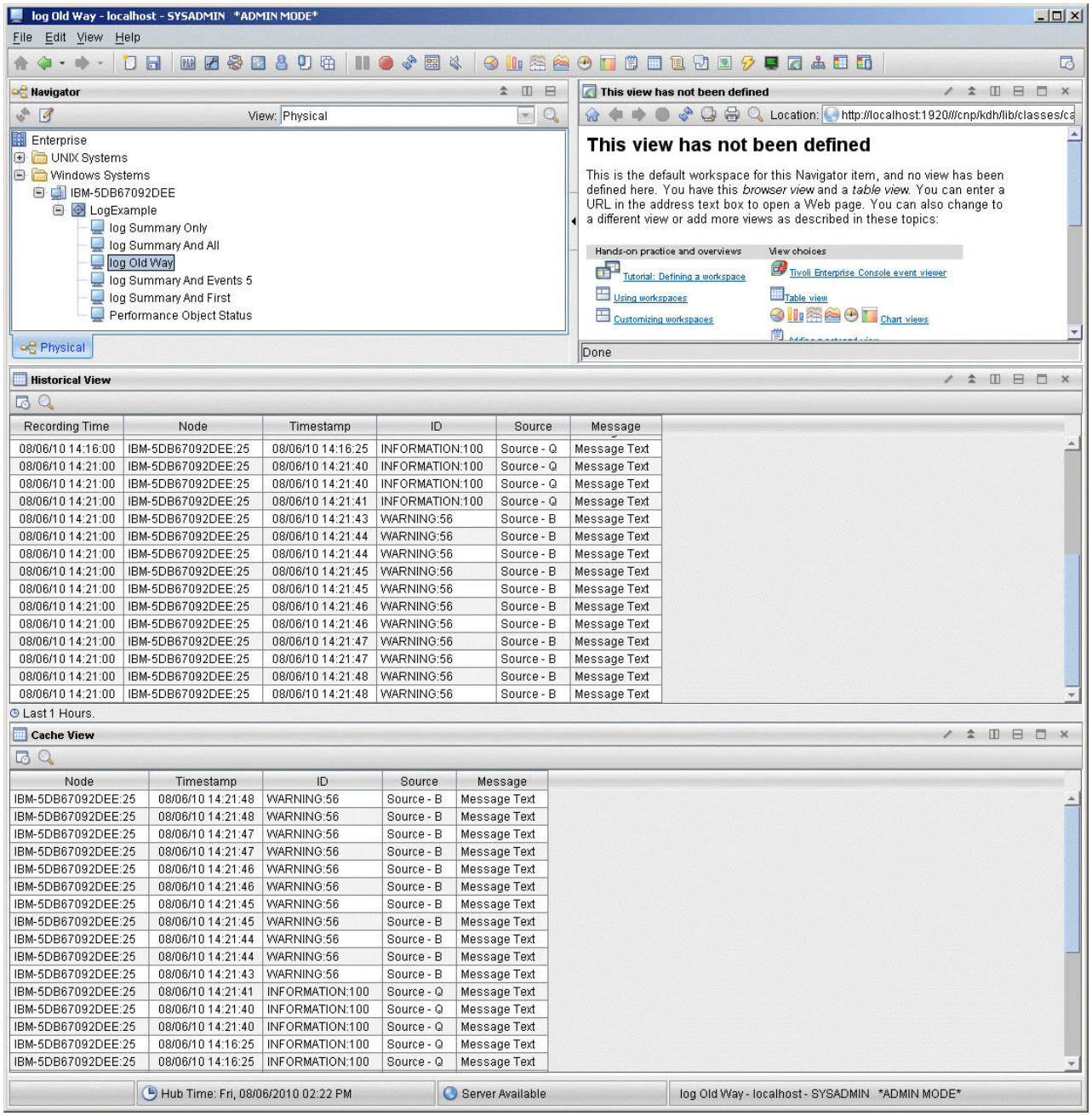


Figure 284. Historical view and cache view when event filtering or summarization is not enabled

(Figure 285) shows the historical view and cache view if you selected the **Only send summary events** option in the Event Information tab. The summary events are displayed in both views, but the new events are only displayed in the real-time (cache) view.

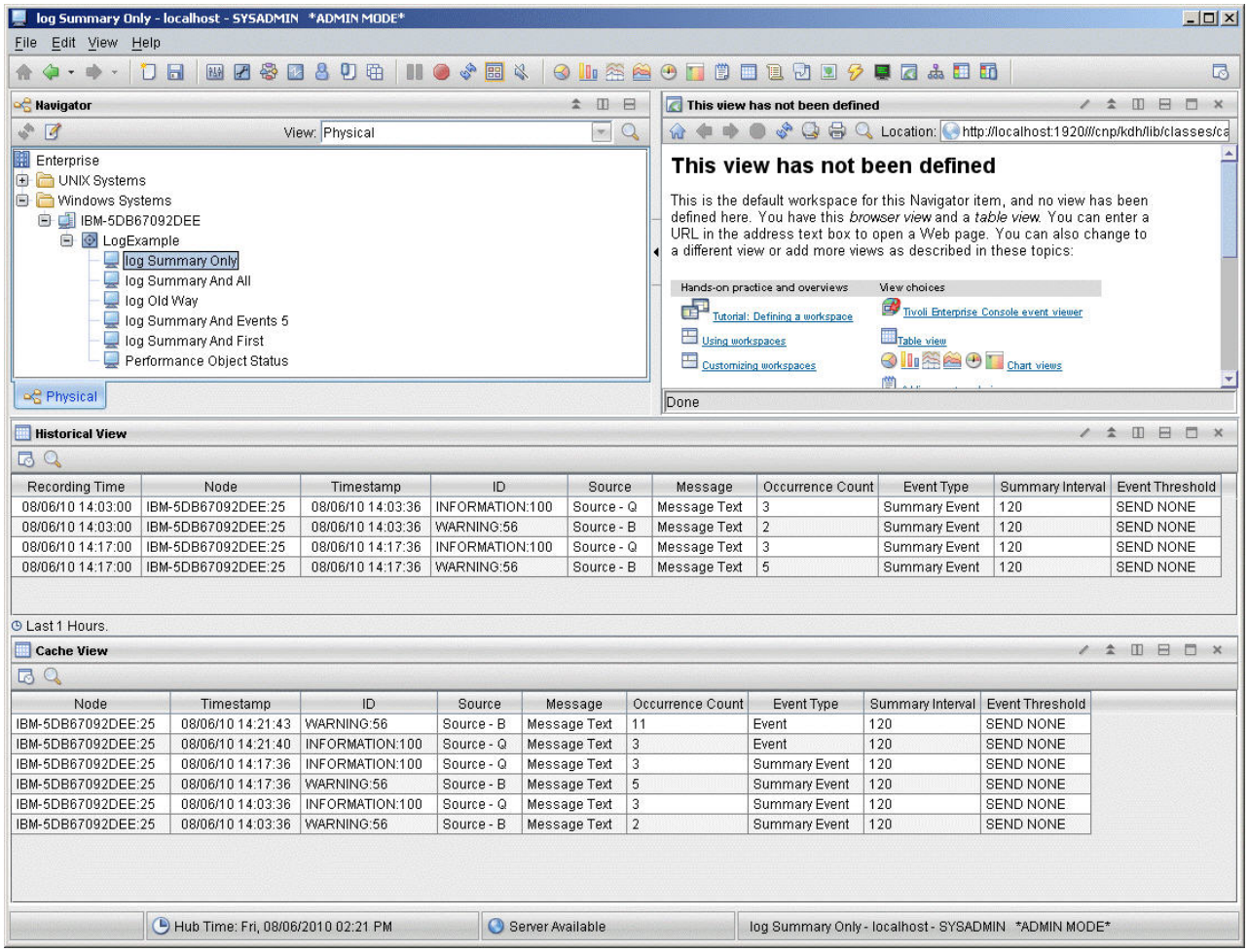


Figure 285. Historical view and cache view when **Only send summary events** *is selected*

(Figure 286 on page 422) shows the historical view and cache view if you selected the **Send all events** option in the **Event Information** tab. All of the events are shown in both views, but you also see the summary events that are created at the end of each interval. The real-time view changes when the interval elapses. The existing events are converted into summary records and then the new events are added. The addition of the other two available event attributes that are used to display the summary interval (120 seconds in this example) and the *SEND ALL* threshold.

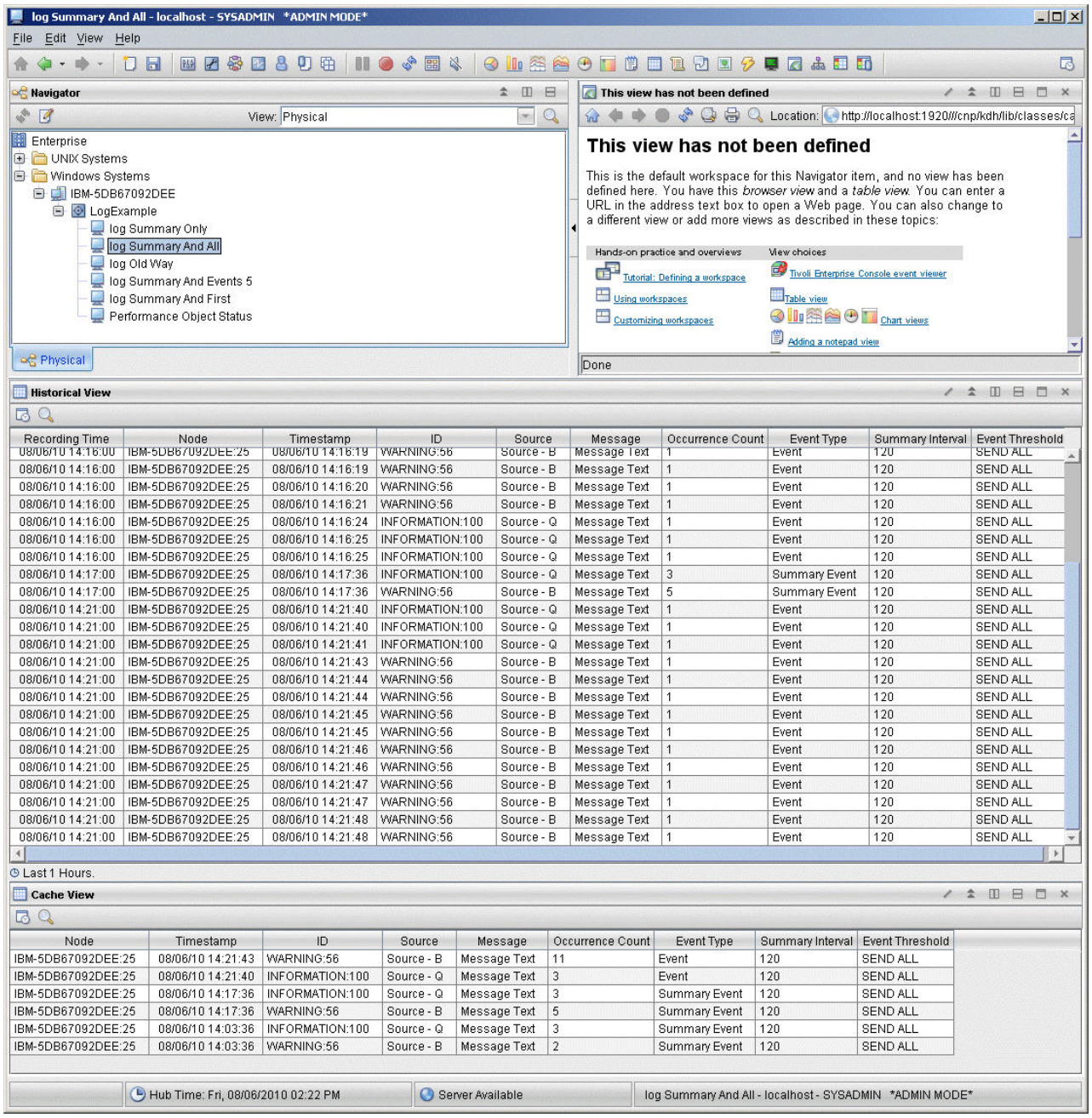


Figure 286. Historical view and cache view when **Send all events** *is selected*

(Figure 287 on page 423) shows the historical view and cache view if you selected the **Send first event** option in the Event Information tab. The summary events are displayed in both views, but all the new events are only displayed in the real-time (cache) view. For each event, the historical view displays only the first event that is received in the interval and no duplicate events.

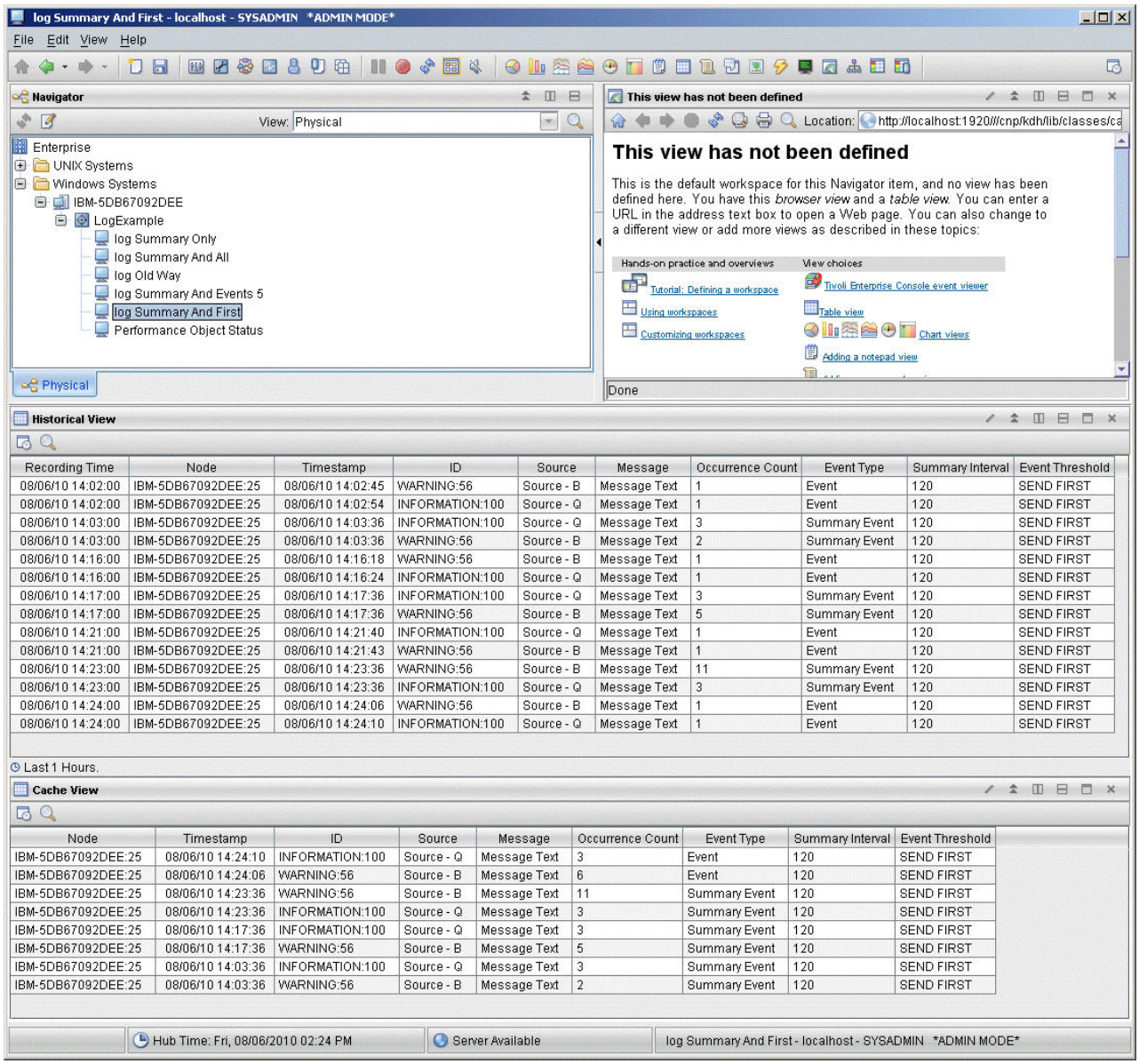


Figure 287. Historical view and cache view when **Send first event** *is selected*

(Figure 288 on page 424) shows the historical view and cache view if you selected the **Event threshold** option and entered a value of 5. The summary events are displayed in both views, but all the new events are only displayed in the real-time (cache) view. In this example, a threshold of 5 is specified. The historical view displays an event only when five duplicates of an event (including the first event) are received in the interval. If less than 5 are received, no event is displayed. If 6, 7, 8, or 9 duplicates are received in the interval, one event is displayed. If 10 duplicates are received, 2 events are displayed.

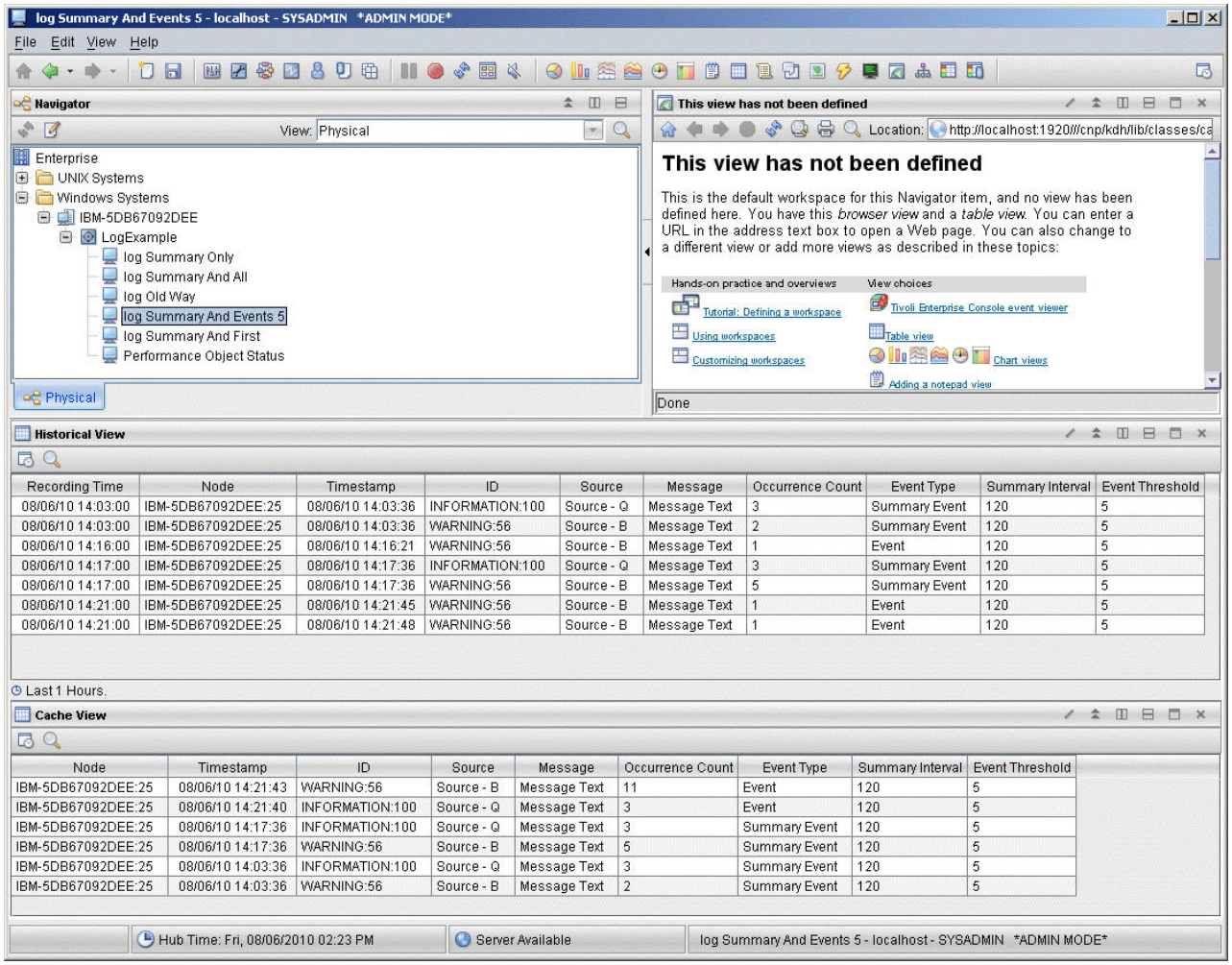


Figure 288. Historical view and cache view when **Event threshold** *is selected*

Related concepts:

Chapter 34, "Event filtering and summarization," on page 417

Chapter 35. Troubleshooting

Use troubleshooting to diagnose why Agent Builder or your agent is malfunctioning.

This documentation explains how to troubleshoot the Tivoli Monitoring Agent Builder and the resulting agents. Troubleshooting, or problem determination, is the process of determining why a certain product is malfunctioning. You might be able to resolve some problems by ensuring that your system matches the system requirements that are listed in Chapter 3, “Installing the Tivoli Monitoring Agent Builder,” on page 5.

Gathering product information for IBM Software Support

Gather certain product information before you contact IBM Software Support.

About this task

Gather the product information that is detailed in: Table 46.

Procedure

Gather the following information before you contact IBM Software Support:

Table 46. Information to gather before you contact IBM Software Support

Information type	Description
Log files	Collect trace log files from failing systems. Most logs are in a logs subdirectory on the host computer. See “Agent Builder Trace logging” on page 426 for lists of all trace log files and their locations.
Operating system	Operating system version number and patch level
Messages	Messages and other information that is displayed on the screen
Version numbers for Tivoli Monitoring	Version number of the following members of the monitoring environment • Tivoli Monitoring. Also, provide the patch level, if available. • Tivoli Monitoring: Windows OS Agent • Tivoli Monitoring: UNIX OS Agent • Tivoli Monitoring: Linux OS Agent • Tivoli Universal Agent
Screen captures	Screen captures of incorrect output, if any
`itm_agent_toolkit.xml` file	When an agent exhibits a problem, send a small non-confidential agent `.xml` file of a special case or sample test scenario that illustrates the problem.

© Copyright IBM Corp. 2007, 2013

Agent Builder Trace logging

Trace logging is built into the software to assist IBM Software Support in gathering information to determine why a problem is occurring.

Note: Trace logs contain detailed internal information and require an internal understanding. These logs are for use only by IBM Software Support.

If you encounter an issue, IBM Software Support might request that you obtain the trace logs, to review the information. Here Trace logging is introduced with an explanation on how to locate the appropriate files.

Trace logging captures information about the operating environment when the code fails to operate as intended.

The Tivoli Monitoring Agent Builder supports tracing through the Java Logging (JLog) Toolkit. The trace logger is configurable through the Eclipse plug-in preferences. For more information about configuration of the Trace logging, see ("Logging configuration" on page 428).

Agent Builder Trace logging overview

Trace logs capture information about the operating environment when component code fails to operate as intended. These logs are in the English language only. IBM Software Support uses the information that is captured by trace loggers to trace a problem to its source or to determine why an error occurred. Generally, this information is not enabled by default, but logging can be enabled without stopping the application.

Trace log location

Trace logging is kept in a configurable number of rotating log files. The log file names follow the format `traceKQZ.log, traceKQZ1.log...` where `traceKQZ.log` is the most recent log file.

You can retrieve trace logs in the following location:

workspace_directory`/.metadata/tivoli/KQZ/logs`

Where:

workspace_directory
: Identifies the workspace directory that you specified when you initially ran the Agent Builder, see ("Starting the Agent Builder" on page 11).

To save the log data, you must select the Enable Trace logging to files option on the Output page of the Tivoli Monitoring Agent Builder preferences. For information about the Output page, see ("Customize trace file settings" on page 429).

Trace log format

The trace log record format is as follows:

[*Date*]*-Level-Server-Thread-FileName.Method-LogText Exception*(StackTrace)

Where:

Date
: Time the log was produced specified in milliseconds, since January 1st 1970, for example, 1015343592000.

Server
host name on which the program is running.

Level
The level of detail the trace represents, which can be one of the following options:

ERROR
error message.

WARN
Warning.

INFO
Informational message.

MIN
The minimum level of tracing provides some information about the program state with only a minimal affect on the performance of the program.

MID
The medium level of tracing follows the paths of some high-level APIs and their parameters.

MAX
The maximum level of tracing follows the paths of most of the code and provide a detailed program flow. This level includes minimum and medium levels of tracing as well.

For more information about setting these levels, see ("Configuration by using the IBM Java Logging Toolkit" on page 428).

FileName
Name of the file or class to which the trace refers.

Method
The method in the class to which the trace refers.

Thread
Thread to which the trace refers. Thread element that refers to the operating system-specific notion of a thread.

LogText
A description of the error

Exception
Exception element depends upon the specific language or platform. For Java, it is a stack trace.

Trace log example

The following text shows an example of a trace that is recorded in the trace log:

```
[2007-08-17T17:50:17.527-04:00] -  MIN  - myhost - main - AgentXMLDocument.parseAgent
 - begin parsing document
```

Where:

2007-08-17
Specifies the date that the log entry was produced.

17:50:17.527
Specifies the time that the log entry was produced.

-04:00
Specifies the Coordinated Universal Time offset.

MIN
Specifies the level of detail the trace entry represents, in this case, minimum debug level.

myhost
Specifies the host name of the system that produced the trace entry.

main
Specifies name of the thread to which the trace entry refers.

AgentXMLDocument
Specifies name of the file or class to which the trace refers.

parseAgent
Specifies the method in the class to which the trace refers.

begin parsing document
Specifies the description of the condition or error.

Logging configuration

You can configure logging on components of the Tivoli Monitoring Agent Builder.

About this task

Tivoli Monitoring Agent Builder logging options can be configured after installation. If you are experiencing problems, IBM Software Support might request that you configure logging and tracing information. The following steps explain how to configure the product to provide logging data.

Configuration by using the IBM Java Logging Toolkit

You can configure trace logs by using the IBM Java Logging Toolkit to set the trace level for the appropriate component of the Tivoli Monitoring Agent Builder.

About this task

You can do all logging configuration in the IBM Java Logging Toolkit. Access the Eclipse preference window by using the following steps:

Procedure

1. Select **Window** > **Preferences** to display the **Window** toolbar menu.
2. Click **IBM Java Logging Toolkit** to display the IBM Java Logging Toolkit window.
   The Logger Configuration table displays the trace configuration for the following components:

   JLog IBM Tivoli Monitoring Logger Trace Logger

   KQZGeneratorTrace
   IBM Tivoli Monitoring Generator Trace Logger

   KQZTrace
   IBM Tivoli Monitoring Agent Builder Trace Logger

   The trace level for each component can be set independently to one of the following values. Selecting a trace level causes trace entries of that level and all higher levels to be included in the log file.
   - **Error** - error messages.
   - **Warning** - Warnings.
   - **Information** - Informational messages.
   - **Minimum** - The Minimum level of tracing provides some information about the program state with only a minimal affect on the performance of the program.
   - **Medium** - The Medium level of tracing follows the paths of some high-level APIs and their parameters.
   - **Maximum** - The Maximum level of tracing follows the paths of most of the code and provide a detailed program flow.

Logging for each component can also be enabled or disabled by selecting **On** or **Off** for Logging.

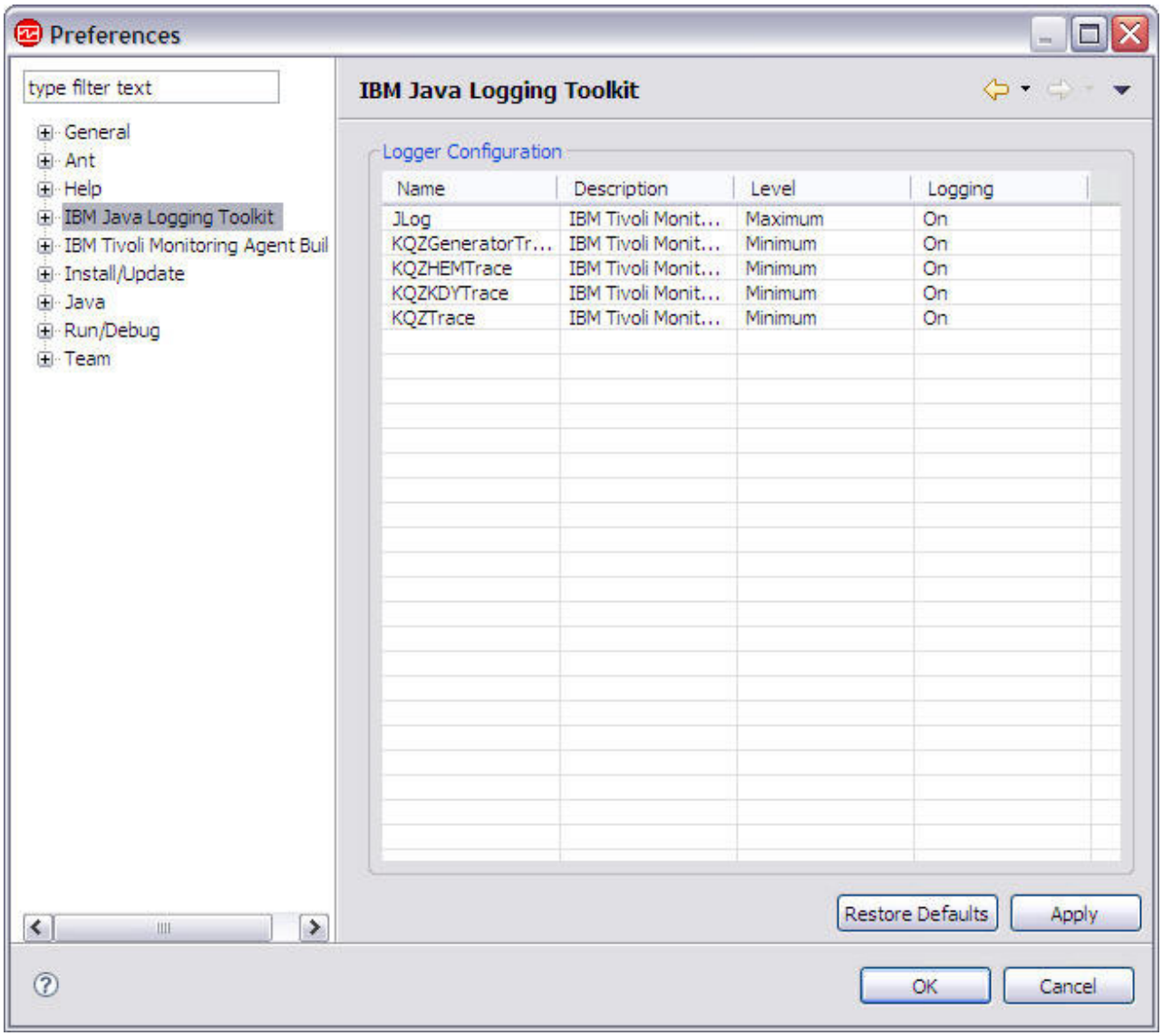


Figure 289. IBM Java Logging Toolkit

3. By expanding the IBM Java Logging Toolkit, you can set the following Trace Attributes for each component:
   - **Log file Directory**
   - **Maximum number of revolving trace files**
   - **Maximum size of each trace file (in KB)**
   - **Enable trace logging to files**
   - **Enable trace logging to console**
   - **Restore Defaults**

Customize trace file settings

Use the appropriate output windows to customize the trace file settings.

About this task

Access the Eclipse output page by doing the following steps:

Procedure

1. Click **Window** to display the **Window** toolbar menu.
2. Click **Preferences** to display the Preferences window.
3. Click the plus sign next to the IBM Java Logging Toolkit to display the output windows.
4. Click the appropriate output window to display the output settings for that component.
   The window displays the following information:

   Log file directory
   Specifies the location of the log file directory.

   Maximum number of revolving trace files
   Specifies the number of trace files that are used and reused to store the trace data.

   Maximum size of each trace file (in Kilobytes)
   Specifies the maximum size in KB for each of the revolving trace files.

   Enable trace logging to files
   Check this box to store the trace log data.

   Enable logging to console
   Check this box to send logging information to a development console.

   Note: If you enable this option, you must restart the Tivoli Monitoring Agent Builder as described in the following procedure:

   a. Close the Tivoli Monitoring Agent Builder.
   b. Modify the *install_dir*/agentbuilder.ini file.
      - On Windows systems, change *install_dir*/_jre/jre/bin/javaw.exe to *install_dir*/_jre/jre/bin/java.exe.
      - On Linux or AIX systems, change *install_dir*/_jre/jre/bin/javaw to *install_dir*/_jre/jre/bin/java.
   c. Relaunch the Tivoli Monitoring Agent Builder.

Configuring and Tuning data collection

When an Agent Builder agent is created, you can configure and tune its data collection to achieve the best results.

How you configure and tune your agent can be different for different Agent Builder agents and even between attribute groups in a single agent. Agent Builder agents can include two types of data and they support two basic methods of data collection for the most common type of data.

Data types

An agent collects two types of data:

1. Most Tivoli Monitoring attribute groups represent snapshots of data. Someone asks for the data and it is returned. Agents use this type of data to represent configuration, performance, status, and other information where a one time collection of a set of data makes sense. This data is called *sampled data*.
2. Some Tivoli Monitoring data represents events. In this case, an event happens and the agent must forward data to Tivoli Monitoring. Examples of events are SNMP Traps, Windows Event Log entries, and new records that are written to a log file. For simplicity, these types of data are grouped and referred to as *event data*.

Sampled data

When sampled data is required, a request is sent to the agent for a specific attribute group. The request might be initiated by clicking a workspace in the Tivoli Enterprise Portal. Other things that might initiate a request are a situation that is running, a data collection for the Warehouse, or a SOAP request. When the agent receives the request, the agent returns the current data for that attribute group. Tivoli Enterprise Portal requests target a specific attribute group in a particular Managed System Name (MSN). Situations and historical requests are more interesting, especially in an agent which includes subnodes. When a situation needs data for an attribute group in a subnode, the agent receives one request with a list of the targeted subnodes. The agent must respond with all the data for the requested attribute group for all of the subnodes before Tivoli Monitoring can work on the next request.

The most straightforward way for an agent to satisfy a request is to collect data every time it receives a request from Tivoli Monitoring. Agent Builder agents do not collect data every time. Data is not collected every time because it often takes time or uses resources to collect data. And in many cases the same data is requested many times in a short period. For example, a user might define several situations that run at the same interval on an attribute group and the situations can signal several different conditions. Each of these situations results in a request to the agent, but you might prefer each of the situations to see the same data. It is likely that as each situation sees the same data, more consistent results are obtained, minimizing the demand for system resources by the monitoring agent.

The agent developer can configure agents to optimize data collection by choosing to run the collection in one of the following two modes:

1. **On-demand collection**: The agent collects data when it receives a request and returns that data.
2. **Scheduled collection**: The agent runs data collection in the background on scheduled intervals and returns the most recently collected data when it receives a request.

The agent uses a short-term cache in both of these modes. If another request for data is received while the cache is valid, the agent returns data from the cache without collecting new data for each request. Using data from the cache solves the problem that is caused by multiple concurrent situations (and other types of) requests. The amount of time the data remains valid, the scheduled collection interval, the number of threads that are used for collection and whether the agent runs in on demand or scheduled mode are all defined by environment variables. Using the environment variables, you can tune each agent for the best operation in its environment.

See the following examples that illustrate how the agent works in both modes:

- Agent 1 (*on-demand* collection): A simple agent that collects a small amount of data that is normally accessed only by situations or on an infrequent basis in the Tivoli Enterprise Portal. Data collection is reasonably fast, but it can use up computing and networking resources. This agent is normally defined to run on demand. If no situations are running or no one clicks the Tivoli Enterprise Portal, the agent does nothing. When data is needed, it is collected and returned. The data is placed into the short-term cache so that further requests at about the same time return the same data. This type of collection is likely the most efficient way for this agent to run because it collects data only when someone actually needs it.
- Agent 2 (*scheduled* collection): A complex agent that includes subnodes and collects data from multiple copies of the monitored resource. Many copies of the resource can be managed by one agent. It is normal to run situations on the data on a relatively frequent basis to monitor the status and performance of the monitored resource. This agent is defined to run a *scheduled*collection. One reason for running a *scheduled* collection is the way that situations are evaluated by Tivoli Monitoring agents. Because situations are running on the attribute groups in the subnodes, the agent receives one request for the data from all of the subnodes simultaneously. The agent cannot respond to other requests until all of the data is returned for a situation. If the agent collected all of the data when the request arrived, the agent would freeze when you click one of its workspaces in theTivoli Enterprise Portal. To avoid freezing the agent, the agent builder automatically defines all subnode agents to run as scheduled

collection. The agent developer tunes the number of threads and refresh interval to collect the data at a reasonable interval for the data type. For example, the refresh interval can be one time a minute, or one time every 5 minutes.

Environment variables:
An agent determines which mode to use and how the scheduled data collection runs based on the values of a set of environment variables. These environment variables can be set in the definition of the agent on the Environment Variables panel. Each environment variable is listed in the menu along with the default values. The environment variables can also be set or modified for an installed agent by editing the agent's environment (`env`) file on Windows or initialization (`ini`) file on UNIX. The environment variables that control data collections for sampled attribute groups are:

- `CDP_DP_CACHE_TTL=<validity period for the cached data - default value 60 seconds>`
- `CDP_DP_THREAD_POOL_SIZE=<number of threads to use for concurrent collection - default value 15 for subnode agents>`
- `CDP_DP_REFRESH_INTERVAL=<number of seconds between collections - default value 60 seconds for subnode agents>`
- `CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT=<amount of time to wait for new data after validity period expires - default value 5 seconds>`

The most important of these variables are `CDP_DP_CACHE_TTL`, `CDP_DP_REFRESH_INTERVAL`, and `CDP_DP_THREAD_POOL_SIZE`.

If `CDP_DP_THREAD_POOL_SIZE` has a value greater than or equal to 1 or the agent includes subnodes, the agent operates in *scheduled* collection mode. If `CDP_DP_THREAD_POOL_SIZE` is not set or is 0, the agent runs in *on-demand* collection mode.

If the agent is running in *scheduled* mode, then the agent automatically collects all attribute groups every `CDP_DP_REFRESH_INTERVAL` seconds. It uses a set of background threads to do the collection. The number of threads is set by using `CDP_DP_THREAD_POOL_SIZE`. The correct value for the `CDP_DP_THREAD_POOL_SIZE` varies based on what the agent is doing. For example:

- If the agent is collecting data from remote systems by using SNMP, it is best to have `CDP_DP_THREAD_POOL_SIZE` similar to the number of remote systems monitored. By setting the pool size similar to the number of monitored remote systems, the agent collects data in parallel, but limits the concurrent load on the remote systems. SNMP daemons tend to throw away requests when they get busy. Discarding requests forces the agent into a try-again mode and it ends up taking more time and more resources to collect the data.
- If the agent includes a number of attribute groups that take a long time to collect, use enough threads so that long data collections can run in parallel. You can probably add a few more for the rest of the attribute groups. Use threads in this way if the target resource can handle it. Examples of when attribute groups can take a long time to collect are if the script runs for a long time, or a JDBC query takes a long time.

Running an agent with a larger thread pool causes the agent to use more memory (primarily for the stack that is allocated for each thread). It does not however increase the processor usage of the process or increase the actual working set size of the process noticeably. The agent is more efficient with the correct thread pool size for the workload. The thread pool size can be tuned to provide the wanted behavior for a particular agent in a particular environment.

When data is collected, it is placed in the internal cache. This cache is used to satisfy further requests until new data is collected. The validity period for the cache is controlled by `CDP_DP_CACHE_TTL`. By default the validity period is set to 60 seconds. When an agent is running in scheduled mode, it is best to set the validity period to the same value as `CDP_DP_REFRESH_INTERVAL`. Set it slightly larger if data collection can take a long time. When set the validity period in this way, the data is considered valid until its next scheduled collection.

The final variable is CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT. This variable comes into play only when CDP_DP_CACHE_TTL expires before new data is collected. When the cache expires before new data is collected, the agent schedules another collection for the data immediately. It then waits for this collection to complete up to CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT seconds. If the new collection completes, the cache is updated and fresh data is returned. If the new collection does not complete, the existing data is returned. The agent does not clear the cache when CDP_DP_CACHE_TTL completes to prevent a problem that is seen with the Universal Agent. The Universal Agent always clears its data cache when the validity period ends. If the Universal Agent clears its data cache before the next collection completes, it has an empty cache for that attribute group and returns no data until the collection completes. Returning no data becomes a problem when situations are running. Any situation that runs after the cache cleared but before the next collection completes sees no data and any of the situations that fire are cleared. The result is floods of events that fire and clear just because data collection is a little slow. The Agent Builder agents do not cause this problem. If the 'old' data causes a situation to fire generally the same data leaves that situation in the same state. After the next collection completes, the situation gets the new data and it either fires or clears based on valid data.

Attribute groups:
Agent Builder agents include two attribute groups that you can use to inspect the operation of data collection and to tune the agent for your environment. The attribute groups are Performance Object Status and Thread Pool Status. These attribute groups are described in (Appendix C, "Attributes reference," on page 541). When these attribute groups are used to tune data collection performance, the most useful data is:

- Performance Object Status, Average Collection Duration attribute. This attribute shows you how long each attribute group is taking to collect data. Often a small percentage of the attribute groups in an agent represents most of the processor usage or time that is used by the agent. You might be able to optimize the collection for one or more of these attribute groups. Or you can modify the collection interval for one or more groups, if you do not need some data to be as up-to-date as other data. For more information, see ("Examples and advanced tuning" on page 434).
- Performance Object Status, Intervals Skipped attribute. This attribute shows you how many times the agent tried to schedule a new collection for the attribute group and it found that the previous collection was still on the queue, waiting to be run, or already running. In a normally behaved agent this attribute value is zero for all attribute groups. If this number starts growing, you tune the data collection, by adding threads, lengthening the interval between collections, or optimizing the collection.
- Thread Pool Status, Thread Pool Avg Active Threads attribute. You can compare this value to the Thread Pool Size attribute group to see how well your thread pool is being used. Allocating a thread pool size of 100 threads when the average number of active threads is 5 is probably just wasting memory.
- Thread Pool Status, Thread Pool Avg Job wait and Thread Pool Avg Queue Length attributes. These attributes represent the time an average data collection spends waiting on the queue to be processed by a thread and the average number of collections on the queue. Because of the way this data is collected, even an idle system indicates that at least an average of one job is waiting on the queue. A larger number of waiting jobs or a large average wait time indicates that collections are being starved. You can consider adding threads, lengthening the interval between collections or optimizing the collection for one or more attribute groups.

Event data

Agent Builder agents can expose several types of event data. Some behavior is common for all event data. The agent receives each new event as a separate row of data. When a row of event data is received, it is sent immediately to Tivoli Monitoring for processing, and added to an internal cache in the agent. Situations and historical collection are performed by Tivoli Monitoring when each row is sent to Tivoli Monitoring. The cache is used to satisfy Tivoli Enterprise Portal or SOAP requests for the data. The agent can use the cache to perform duplicate detection, filtering, and summarization if defined for the attribute group. The size of the event cache for each attribute group is set by CDP_PURE_EVENT_CACHE_SIZE. This

cache contains the most recent CDP_PURE_EVENT_CACHE_SIZE events with the most recent event returned first. There are separate caches for each event attribute group. When the cache for an attribute group fills, the oldest event is dropped from the list.

The Agent Builder agent can expose events for:

- Windows Event Log entries
- SNMP Traps or Informs
- Records added to log files
- JMX MBean notifications
- JMX monitors
- Events from a Java API provider or socket provider.
- Joined attribute groups (where one of the data sources is an event data source)

These events are handled in the most appropriate way for each of the sources. SNMP Traps and Informs, JMX notifications and events from the Java API and socket providers are received asynchronously and forwarded to Tivoli Monitoring immediately. There is no requirement tune these collectors. The agent subscribes to receive Windows Event Log entries from the operating system by using the Windows Event Log API. If the agent is using the older Event Logging API, it polls the system for new events by using the thread pool settings. For joined attribute groups where one of the data sources is an event data source, there is no tuning to apply to the joined attribute group. Though the joined attribute group does benefit from any tuning applied to the event source group.

File monitoring is more complicated. The agent must monitor the existence of the files and when new records are added to the files. The agent can be configured to monitor files by using patterns for the file name or a static name. As the set of files that matches the patterns can change over time, the agent checks for new or changed files every KUMP_DP_FILE_SWITCH_CHECK_INTERVAL seconds. This global environment variable governs all file monitoring in an agent instance. When the agent determines the appropriate files to monitor, it must determine when the files change. On Windows systems, the agent uses Operating System APIs to listen for these changes. The agent is informed when the files are updated and processes them immediately. On UNIX systems, the agent checks for file changes every KUMP_DP_EVENT seconds. This global environment variable governs all file monitoring in an agent instance. When the agent notices that a file changed, it processes all of the new data in the file and then waits for the next change.

Examples and advanced tuning

Example

Environment variables that are used for more advanced tuning are defined at the agent level. You set the following variables one time and they apply to the all of the attribute groups in the agent:

- CDP_DP_CACHE_TTL
- CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT
- KUMP_DP_FILE_SWITCH_CHECK_INTERVAL
- KUMP_DP_EVENT

You can make the following variables apply to individual attribute groups. They still have a global setting that applies to all other attribute groups in the agent:

- CDP_DP_REFRESH_INTERVAL
- CDP_PURE_EVENT_CACHE_SIZE

If you defined an agent to include the following six attribute groups:

- EventDataOne
- EventDataTwo
- EventDataThree

- SampledDataOne
- SampledDataTwo
- SampledDataThree

You might set the following default variables:

- `CDP_DP_CACHE_TTL=60`
- `CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT=2`
- `CDP_DP_REFRESH_INTERVAL=60`
- `CDP_PURE_EVENT_CACHE_SIZE=100`

As a result, all of the attribute groups which contain sampled data (SampledDataOne, SampledDataTwo, and SampledDataThree) would be collected every 60 seconds. Each of the event attribute groups (EventDataOne, EventDataTwo, and EventDataThree) would store the last 100 events in their cache.

These settings might work perfectly, or there might be reasons that you must control the settings at a more granular level. For example, what if EventDataOne generally receives 10 times as many events as EventDataTwo and EventDataThree? To further complicate things, there really is a link between EventDataOne and EventDataTwo. When one event is received for EventDataTwo, there are always multiple events for EventDataOne and users want to correlate these events. There is not a single correct setting for the cache size. It would be nice to be able to have EventDataOne store a larger number of events and EventDataTwo store a smaller number. You can achieve this storage by setting `CDP_PURE_EVENT_CACHE_SIZE` to the size that makes sense for most of the event attribute groups, 100 seems good. Then, you can set `CDP_EVENTDATAONE_PURE_EVENT_CACHE_SIZE` to 1000. That way all of the corresponding events are visible in the Tivoli Enterprise Portal.

The same thing can be done with `CDP_DP_REFRESH_INTERVAL`. Set a default value that works for the largest number of attribute groups in the agent. Then set `CDP_`*`attribute group name`*`_REFRESH_INTERVAL` for the attribute groups which must be collected differently. To optimize collection, set the default `CDP_DP_REFRESH_INTERVAL` to match the `CDP_DP_CACHE_TTL` value. `CDP_DP_CACHE_TTL` is a global value so if set to a value less than a refresh interval, unexpected collections might occur.

Environment variables

Use environment variables to control the behavior of the agent at run time.

Environment variables can be built into the agent by using the Environment Variables page, for more see (“Environment variables” on page 39). On Windows systems, environment variables are defined in the agent `K`*`XX`*`ENV` file. On UNIX and Linux systems, these variables can be defined in the agent `$CANDLEHOME/config/`*`XX`*`.ini` file. *XX* is the two-letter product code. The agent must be restarted for the new settings to take effect.

Note: Environment variables are not set correctly on a remote system that runs C Shell. Use a different shell if you want to use environment variables.

Table 47. Environment variables

Environment variable	Default value	Valid values	Description
`CDP_`*`ATTRIBUTE_GROUP`*`_REFRESH_INTERVAL`	Not Applicable	Any non-negative integer	The interval in seconds at which a particular specified attribute group is updated in the background. This variable works in the same way as `CDP_DP_REFRESH_INTERVAL`, except it targets only the specified attribute group. The attribute group name in the variable name must be in uppercase, even if the actual attribute group name is not.

Table 47. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_CACHE_TTL	60	Any integer greater than or equal to 1.	Data that is collected for an attribute group is cached for this number of seconds. Multiple requests for the same data in this time interval receive a cached copy of the data. This value applies to all attribute groups in the agent. (You cannot set different values for each attribute group.)
CDP_DP_IMPATIENT_ COLLECTOR_TIMEOUT	5 if subnodes are defined, otherwise not set	Any positive integer	The number of seconds to wait for a data collection before a timeout and cached data is returned, even if the cached data is stale. (Cached data is stale if older than CDP_DP_CACHE_TTL seconds). If this variable is not set, the agent waits until the data collection completes. The wait at times can make the Tivoli Enterprise Portal timeout and give up waiting. If no thread pool is configured, this variable is ignored and data collection is done synchronously.
CDP_DP_REFRESH_INTERVAL	60 if subnodes are defined, otherwise not set	Any non-negative integer	The interval in seconds at which attribute groups are updated in the background. If this variable is not set or is set to 0, background updates are disabled. If a thread pool is configured (see variable CDP_DP_THREAD_POOL_SIZE), then the attribute groups can be refreshed in parallel. If there is no thread pool, the updates happen serially, which can take a long time. Logically equivalent to a thread pool size of 1.
CDP_DP_THREAD_POOL_SIZE	15 if subnodes are defined, otherwise not set	Any non-negative integer	The number of threads that are created to run background data collections at an interval that is defined by CDP_DP_REFRESH_INTERVAL. If this variable is not set or is set to 0, there is no thread pool. The Thread Pool Status attribute group shows how the thread pool is running. Use the Thread Pool Status to adjust the thread pool size and refresh interval for best results. By default, the query for this attribute group is not displayed on the agent Navigator tree. You might not remember to include the query in a custom workspace for the agent. However, you can easily view it by assigning the Thread Pool Status query to a base agent level workspace view.
CDP_JDBC_MAX_ROWS	1000	Any positive integer	The maximum number of rows of data that the JDBC data provider returns. A result set that contains more than this number of rows is processed only up to this maximum value. Queries can be developed to prevent too much data from being returned to IBM Tivoli Monitoring.

Table 47. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_NT_EVENT_LOG_GET_ALL _ENTRIES_FIRST_TIME	NO	YES, NO	If set to YES, the agent sends an event for every event in the Windows event log. If set to NO, only new events in the Windows event log are sent.
CDP_NT_EVENT_LOG_CACHE _TIMEOUT	3600	Any integer greater than or equal to 300.	The number of seconds Windows Event log events are cached by the agent. All cached events are returned when the event log attribute group is queried. **Note:** This variable is no longer used. Use the CDP_PURE_EVENT_CACHE_SIZE variable.
CDP_PURE_EVENT_CACHE_SIZE	100	Any positive integer greater than or equal to 1.	Maximum number of events to cache for a log file data source that is configured to process new records, for the Windows Event Log attribute group. And also for JMX monitors and notifications. Each new record in the log causes an event to be sent. This environment variable defines how many events are remembered in a cache by the agent. The cached values are returned when the attribute group is queried.
CDP_DP_ACTION_TIMEOUT	20 seconds	Any positive integer greater than or equal to 1.	The number of seconds to wait for a Take Action that is being handled by the agent to complete.
CDP_DP_SCRIPT_TIMEOUT	30 seconds	Any positive integer greater than or equal to 10.	The number of seconds to wait for the program started by a script-based attribute group to complete.
CDP_DP_PING_TIMEOUT	30 seconds	Any positive integer greater than or equal to 10.	The number of seconds to wait for the program started by a command return code to complete. **Note:** This variable is not related to the ICMP ping data provider.
CDP_SNMP_MAX_RETRIES	2	Any positive integer	The number of times to try sending the SNMP request again. The total number of requests that are sent to the SNMP agent is this value plus one if no responses are received.
CDP_SNMP_RESPONSE_TIMEOUT	2 seconds	Any positive integer	The number of seconds to wait for each SNMP request to timeout. Each row in an attribute group is a separate request. This timeout value is the number of seconds to wait for a response before you try again. The total timeout for a single row of data is (CDP_SNMP_MAX_RETRIES + 1) * CDP_SNMP_RESPONSE_TIMEOUT. The total default timeout value is (2+1) * 2 = 6 seconds.

Table 47. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_HOSTNAME	Name of the first installed network interface	An IP address or host name	Sets the preferred host name (network interface) on a multiple interface system. Use this environment variable if the agent binds its listening ports to a non-default network interface address. Used by the SNMP data provider. For Socket data sources, this variable applies if CDP_DP_ALLOW_REMOTE is also set.
CDP_SNMP_ALLOW_ DECREASING_OIDS	NO	YES, NO	If set to YES, the SNMP data providers do not check whether returned OIDs are increasing. Set to YES with caution because the monitored agent might have problems that this check would normally catch.
KUMP_DP_COPY_MODE_SAMPLE_INTERVAL	60	Wait time in seconds	For a log file data provider, specifies how long to wait before it rereads the contents of a file when the agent is defined to **Process all records when the file is sampled**. The time is specified in seconds.
KUMP_MAXPROCESS	100%	5-100%	For a log file data provider, specifies the maximum processor usage to use to process file data. Values range from 5 to 100 percent. The default is 100 percent.
KUMP_DP_SAMPLE_FACTOR	5	Any non-negative integer	For a log file data provider, sets the sampling factor when you select **Process all records when the file is sampled** on the Agent Builder. This wait time ensures that patterns that span multiple records are written before scans are logged for the pattern.
KUMP_DP_EVENT	5	Any non-negative integer	For a log file data provider, sets the sampling frequency for Event data, in seconds.
KUMP_DP_FILE_EXIST_WAIT	YES	YES, NO	For a log file data provider, specifies that the file monitoring thread continues to run if it detects that the monitored file is absent or empty. The thread waits for the file to exist, rechecks every few seconds, and starts or restarts monitoring when the file becomes available.
KUMP_DP_FILE_SWITCH_ CHECK_INTERVAL	600	Any non-negative integer	The frequency in seconds that the log file Data Provider searches for a different monitoring file to switch to when dynamic file name support is enabled.
KUMP_DP_FILE_ROW_ PAUSE_INCREMENT	None	Any non-negative integer	For a log file data provider, specifies how many file records are read before the file monitoring thread pauses. The pause is so that previous updates can be processed. Use this environment variable only if the monitored file receives high-volume bursts of new records and you are concerned that some record updates might be lost.

Table 47. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_COLLECTION_TIMEOUT	60 seconds	Any positive integer	The number of seconds that the agent waits for a response from a data collector that was started in another process. JMX, JDBC, HTTP, and SOAP data collectors are examples.
CDP_SSH_TEMP_DIRECTORY	. (period)	Any valid path string on the remote system	For an SSH enabled Script data provider, specifies the location on the remote system to upload the script files that are provided with the agent to. A relative location is relative to the user's home directory. The default of . (period) denotes the user's home directory.
CDP_SSH_DEL_COMMAND	rm -Rf	Any valid delete command string on the remote system	For an SSH enabled Script data provider, specifies the command to start to delete the uploaded script files that are provided with the agent.
CDP_SNMP_SEND_DELAY_ FACTOR	0 milliseconds	Any positive integer	The initial SNMP send is delayed from 0 to the number of milliseconds specified. This variable is only enabled if the thread pool is also enabled. The delay does not apply to all sends, only to the first send made by an attribute group. This variable is useful if the device that is being monitored has difficulty to respond when multiple requests are received simultaneously.
CDP_ICMP_PING_REFRESH_ INTERVAL	60 seconds	Any integer greater than or equal to 1	The systems in a device list file are pinged at this interval. If the pings use too much time, there is always a delay of at least CDP_PING_MIN_INTERVAL_DELAY seconds before the pings begin again. Data is refreshed no more frequently than this setting. Data can be refreshed less frequently based on the number of entries in the device list file and the time it takes to receive responses.
CDP_ICMP_PING_MIN_ INTERVAL_DELAY	30 seconds	Any integer greater than or equal to 1 and less than the CDP Ping refresh interval	After the devices in a device list file are pinged, the next ping refresh interval does not begin until at least this number of seconds elapses.
CDP_ICMP_PING_BURST	10	Any integer greater than or equal to 0	The number of pings that are sent before the agents pauses for the amount of time that is specified by the CDP_ICMP_PING_BURST_DELAY variable. A value of 0 disables this function.
CDP_ICMP_PING_BURST_DELAY	10	Any integer greater than or equal to 0	The amount of time in milliseconds to wait after a set number of pings are sent as defined by the CDP_ICMP_PING_BURST variable. A value of 0 disables this function.

Table 47. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_ICMP_PING_TIMEOUT	2000 milliseconds	Any integer greater than or equal to 1	The number of milliseconds to wait for a ping response. This setting applies to each ping attempt that is made. Ping attempts are made 3 times for each host. If no response is received from any of the 3 attempts, the total time waited for a reply is CDP_ICMP_PING_TIMEOUT multiplied by 3. By default, this value is 6000 milliseconds. Changing the value for CDP_ICMP_PING_TIMEOUT causes the default TIMEOUT enumeration for the Current® Response Time attribute to no longer apply. Change the TIMEOUT enumeration to the new value of CDP_ICMP_PING_TIMEOUT multiplied by 3.
CDP_JDBC_CONNECTIONLESS	false	true, false	If set to true, JDBC connections are closed after each data collection attempt. That is, all attribute groups attempt to create their own connection each time data is collected. Connections are not reused if this variable is enabled. If set to false, one connection to the database is made and that connection is shared among the attribute groups.
CDP_SSH_EXCLUDED_ ENVIRONMENT_VARIABLES	None	A comma-separated list of environment variable names	For an SSH enabled Script data provider, specifies the set of local environment variables that must not be set in the environment of the remote system.

Table 47. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_EVENT_LOG_MAX_ BACKLOG_TIME	0 seconds	0, 1, or any integer greater than 1	If set to 0, and CDP_DP_EVENT_LOG_MAX_BACKLOG_EVENTS is not set to 1 or a greater integer, does not process events that are generated while the agent is shut down. 0 is the default. If set to 1, and CDP_DP_EVENT_LOG_MAX_BACKLOG_EVENTS is not set to an integer greater than 1, processes all events that are generated while the agent is shut down. If set greater than 1, and CDP_DP_EVENT_LOG_MAX_BACKLOG_EVENTS is not set greater than 1, processes events that are generated within that value in seconds of the current computer time. For example, if the value is set to 300, at startup, the agent processes all events that are generated within 300 seconds of the current time. Where a value greater than 1 is entered for both CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME and CDP_DP_EVENT_LOG_MAX _BACKLOG_EVENTS variables, either that time interval of events or that number of events is processed. Which variable is chosen depends on which is matched first.

Table 47. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_EVENT_LOG_ MAX_BACKLOG_EVENTS	0 events	0, 1, or any integer greater than 1	If set to 0, and CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME variable is not set to 1 or a greater integer, does not process events that are generated while the agent is shut down. 0 is the default. If set to 1, and the CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME variable is not set to an integer greater than 1, processes all events that are generated while the agent is shut down. If set greater than 1, and CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME is not set greater than 1, processes at most that number of events that are generated while the agent is shut down. For example, if the value is set to 200, then at startup of the agent the 200 events that generated directly before startup are processed. Where a value greater than 1 is entered for both CDP_DP_EVENT_LOG_MAX_BACKLOG _EVENTS and CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME, either that number of events or that time interval of events is processed. Which variable is chosen depends on which is matched first.
CDP_HTTP_READ_TIMEOUT	10	Any positive integer	The number of seconds to wait for a reply to the HTTP request.
CDP_JAT_THREAD_POOL_SIZE	15	Any positive integer	The number of threads that are used by the Java providers for handling data collection requests. JMX, JDBC, HTTP, and SOAP data providers are the providers that can benefit from this thread pool.
CDP_HTML_OBJECTS_THREAD_ POOL_SIZE	10	Any positive integer	The number of threads that are used to download page objects that are found in URLs monitored with the HTTP data provider.
CDP_HTTP_SOAP_MAX_ROWS	500	Any positive integer	The maximum number of rows that are returned by the HTTP SOAP data provider.
CDP_DP_ALLOW_REMOTE	NO	NO, YES	If set to Yes, the agent allows remote socket connections. If set to No, the agent allows only socket connections from the local host. No is the default. **Note:** For more information about configuring remote socket ports, and restrictions that apply, see “Remote socket port connection” on page 296
CDP_DP_INITIAL_COLLECTION_ DELAY	varies	Any positive integer	The number of seconds, after the agent starts, until the thread pool begins its scheduled data collections.

Informational, warning, and error messages overview

Messages relay information about how the system or application is running and can alert you to exceptional conditions when they occur.

Messages are sent to an output destination, such as a file, database, or console screen.

If you receive a warning or error message, you can do one of the following actions:

- Follow the instructions that are listed in the Detail window of the message if this information is included there.
- Consult the message details that are listed in here to see what action you can take to correct the problem.
- Consult the message log for message ID, text, time, and date of the message, and other data that you can use to diagnose the problem.

Message format

The message format contains a message ID and text, an explanation, and an operator response.

Tivoli Monitoring Agent Builder messages return in the form of *<CCCYYYnnnS><message text>*.

The following example shows a typical message and explains its identifying components:

`KQZ0010E The field_name must be an integer between -32768 and 32767.`

Where:

KQZ0010E
The message ID of the message. This ID is composed of the following parts:

KQZ Identifies the message as belonging to Tivoli Monitoring Agent Builder.

0010 Identifies the unique serial number of the message.

E Identifies the severity of the message as one of the following options:

I *Informational messages* provide feedback about something that happened in the product or system that might be important. These messages can provide guidance when you are requesting a specific action from the product. Informational messages are not documented.

W *Warning messages* call your attention to an exception condition. The condition might not be an error, but might cause problems if not resolved.

E *Error messages* indicate that an action cannot be completed because of a user or system error. These messages require user response.

The *field_name* **must be an integer between -32768 and 32767.**
The log text of the error message.

The *Text* of the message provides a general statement that relates to the problem or condition that occurred. The *Explanation* provides further information about the message and the possible cause for the condition. The *Operator Response* provides actions to take in response to the condition, particularly for error messages (messages with the "E" suffix).

Note: Many message texts and explanations contain variables, such as the specific name of a server or application. Those variables are represented here as symbols, such as &1. Actual messages contain values for these variables.

Agent messages

The following messages apply to Tivoli Monitoring Agent Builder.

KQZ0005E **Unable to log on to the *namespace* namespace.**

Explanation: The IBM Tivoli Monitoring Agent Builder could not connect to the WMI namespace.

Operator response: Ensure that WMI is running on the computer you specified, that the namespace exists, and that you have the proper credentials to log on to the namespace.

KQZ0007I **Gathering the system Perfmon information.**

Explanation: This is an informational message only.

KQZ0009E **The *field_name* field must start with an alphabetic character and can contain only alphanumeric and underscored characters.**

Explanation: String values can contain only alphabetic and underscored characters.

Operator response: Enter only alphabetic and underscored characters for the value.

KQZ0011E **The *field_name* must be an integer between 0 and 4294967295.**

Explanation: The integer must contain only decimal digits and must be between 0 and 4294967295.

Operator response: Enter an valid integer.

KQZ0012E **The agent file *file_name* could not be read.**

Explanation: The IBM Tivoli Monitoring Agent Builder could not read the agent definition file.

Operator response: If the file contains a valid agent definition, rename the file to itm_toolkit_agent.xml.

KQZ0013E **Unable to access the *file_name* file.**

Explanation: The IBM Tivoli Monitoring Agent Builder cannot read the contents of the file.

Operator response: Ensure that you have permission to access the file. You might have deleted the file from outside of Eclipse and failed to refresh the project.

KQZ0014E **Unable to generate the IBM Tivoli Monitoring Agent in *dir_name*.**

Explanation: The IBM Tivoli Monitoring Agent Builder cannot generate the IBM Tivoli Monitoring Agent.

Operator response: Ensure that the output directory exists and that you have permission to write to it.

KQZ0015E **Error saving *file_name*.**

Explanation: The IBM Tivoli Monitoring Agent Builder could not save the agent file.

Operator response: Ensure that you have write permissions to the file and that the disk is not full.

KQZ0018I **Generating the IBM Tivoli Monitoring Agent**

Explanation: This is an informational message only.

KQZ0021E **The *filename* file was not found.**

Explanation: The selected file does not exist.

Operator response: Specify an existing file.

KQZ0022E **Unexpected error parsing MIB file *filename*.**

Explanation: The syntax of the MIB file is incorrect.

Operator response: Correct the syntax of the MIB file and rerun the import action.

KQZ0025I **Loading the initial MIB data. This operation may take a few moments.**

Explanation: This is an informational message only.

KQZ0026I **The IBM Tivoli Monitoring Agent Builder successfully generated the files for agent *agent_name*.**

Explanation: This is an informational message only.

KQZ0027E **No attributes are defined for the *datasource* attribute group.**

Explanation: This is an informational message only.

KQZ0028E **An attribute with identifier *id* was defined previously.**

Explanation: This is an informational message only.

KQZ0029E **A attribute group with identifier *id* was defined previously.**

Explanation: This is an informational message only.

KQZ0030E **Cannot remove attribute *attr* because it is referenced in derived attribute *derived_attr*.**

Explanation: This is an informational message only.

KQZ0031E ***file_name* is the name of an existing file, not a directory.**

Explanation: The name of a file was specified instead of a directory.

Operator response: Specify the correct name of a project directory.

KQZ0032E **The *dir_name* directory does not exist.**

Explanation: The program could not find the specified directory.

Operator response: Ensure that the directory exists and that you entered the path to it correctly.

KQZ0033E **The following exception was caught while trying to run: *command*:**

Explanation: This error message is generic.

Operator response: Access the stack trace to investigate the problem.

KQZ0034E **The *directory* directory does not contain an agent XML file, *agent_XML_file*.**

Explanation: The program could not find the specified file.

Operator response: Ensure that the file exists and that you entered the path to it correctly.

KQZ0035E **A file named *directory* already exists. Do you want to continue and overwrite the contents of the file?**

Explanation: The file that the IBM Tivoli Monitoring Agent Builder is trying to generate already exists.

Operator response: Select Yes to continue and overwrite the file or No to cancel and select a new directory.

KQZ0036E **The IBM Tivoli Monitoring agent project contains errors as listed in the Problems view. The errors must be corrected before the agent can be generated.**

Explanation: The IBM Tivoli Monitoring Agent Builder project contains errors as listed in the Problems view.

Operator response: Correct the errors and save the IBM Tivoli Monitoring Agent.

KQZ0038I **The file *exception_message* has changed on the file system. Do you want to load the changes?**

Explanation: This is an informational message only.

KQZ0039E **The IBM Tivoli Monitoring Agent *agent_name* was modified. You must save the agent before you can generate it.**

Explanation: This is an informational message only.

KQZ0040E **The IBM Tivoli Monitoring Agent Builder was unable to load required libraries for the browsing features. Follow the instructions in the documentation for copying the required libraries to the correct location on your system.**

Explanation: The IBM Tivoli Monitoring Agent Builder requires native libraries to perform browsing functions.

Operator response: Follow the instructions in the documentation for copying the required libraries to the correct location on your system.

KQZ0041E **No instances were found for the *object_name* Performance Monitor object.**

Explanation: This is an informational message only.

KQZ0042E **Cannot remove the message because it is referenced in command return code data source *func_test*.**

Explanation: This is an informational message only.

KQZ0043W **The selected object has more than *max_properties* properties. Attributes are created for the first *max_properties* properties only.**

Explanation: This is an informational message only.

KQZ0048E **A version of the agent to be generated is currently running on the system. You must stop the agent before you can generate.**

Explanation: This is an informational message only.

KQZ0049E **An error occurred while generating the agent.**

Explanation: The IBM Tivoli Monitoring Agent Builder encountered errors while attempting to generate the agent.

Operator response: Check the details view of the error

dialog for exception information, or examine the Eclipse and IBM Tivoli Monitoring Agent Toolkit trace files to see the cause of the error.

KQZ0050E **A file with the name *directory* exists. Either delete the file or specify another output directory.**

Explanation: This is an informational message only.

KQZ0054E **A return code with value *rc* was defined previously.**

Explanation: This is an informational message only.

KQZ0058I **Generating contents of *agent_XML_file*.**

Explanation: This is an informational message only.

KQZ0059E **No sources were found for the *logName* Windows Event Log.**

Explanation: This is an informational message only.

KQZ0061I **Getting namespaces for *host_name***

Explanation: This is an informational message only.

KQZ0062W **The IBM Tivoli Monitoring Agent Builder was unable to determine if an instance of the agent is currently running. Ensure that an instance of the agent is not running before continuing.**

Explanation: Native libraries needed to determine if an instance of the agent is running could not be loaded. Follow the instructions in the documentation for copying the required libraries to the correct location on your system.

Operator response: Ensure an instance of the agent is not currently running.

KQZ0063I **The specified output directory does not exist. Would you like to create it?**

Explanation: This is an informational message only.

KQZ0071E **The IBM Tivoli Monitoring Agent Builder was unable to import a file due to an internal error.**

Explanation: An invalid parameter was specified in the file import software.

Operator response: Report this problem to IBM.

KQZ0073W **This agent project contains warnings which are indications of potential problems. Attempting to run an agent that contains warnings may cause unpredictable results. Do you wish to continue the generation of the agent?**

Explanation: The IBM Tivoli Monitoring Agent Builder project generated warnings for the agent. The warnings can be viewed in the Problems View. Agents with warnings may not function properly.

Operator response: Select yes to continue generation. Select no if you wish to review the warnings. The warnings can be viewed in the Problems View. Select Window->Show View->Problems if the view is not visible.

KQZ0074W **The builder was unable to resolve the imports in the specified MIB files.**

Explanation: The IBM Tivoli Monitoring Agent Builder could not resolve the import in the MIB file.

Operator response: Ensure that all dependencies for the desired MIB are imported first.

KQZ0075E **No return codes defined for command *command* defined in Command Return Code data source "*test*".**

Explanation: There were no global or command specific return codes for the command in the specified Command Return Code data source.

Operator response: Either define a global return code or define a return code for the specific command.

KQZ0076W **The project file "*file*" is no longer referenced by the agent.**

Explanation: A file was found in the project directory that is not referenced in the agent.

Operator response: Either remove the file from the project or associate the file with a monitor.

KQZ0078E **The version is invalid. The version must contain 3 numbers in the format VRM and cannot start with a 0.**

Explanation: The version contains an unsupported character.

Operator response: A version must contain 3 numbers.

The version is in the format VRM

KQZ0079E **The *field_name* must be an integer between -2147483648 and 2147483647.**

Explanation: The integer must contain only decimal digits and must be between -2147483648 and 2147483647.

Operator response: Enter an valid integer.

KQZ0080E **Could not connect to remote host *host*. The Windows error code was *errcode*.**

Explanation: Could not connect and authenticate with the remote host. The host might be down, the host name might have been typed incorrectly, or the user name or password could be wrong.

Operator response: Enter a correct host name, user name, and password.

KQZ0081E **The Managed System *msys* was not found or was not a supported operating system.**

Explanation: The Managed System that was specified either did not exist or was not a supported operating system.

Operator response: Enter a valid, running Managed System name of a supported operating system type.

KQZ0082E **The Product Code *pc* of Managed System *msys* is not supported.**

Explanation: Only the Product Codes UX (UNIX), LZ (Linux), and NT (Windows) are supported.

Operator response: Enter a valid, running Managed System name of a supported operating system.

KQZ0083E **No processes were returned from query to Managed System *msys*.**

Explanation: The Managed System that was specified was offline or provided no data.

Operator response: Ensure that the specified system is running and accessible on the network.

KQZ0084E **The Managed System *msys* is offline.**

Explanation: The Managed System that was specified is offline.

Operator response: Bring the system back online or choose a different system.

KQZ0085E **Login to WMI namespace *namespace* for user *user* failed.**

Explanation: The system was unable to login to the specified WMI server with the given username and password.

Operator response: Enter a correct username and password for the WMI namespace.

KQZ0086E **Error retrieving classes in WMI namespace *namespace*.**

Explanation: The system was unable to retrieve the classes for the specified namespace.

Operator response:

KQZ0090E **The JMX MBean name or pattern *MBean_pattern* is not valid.**

Explanation: The JMX MBean name or pattern does not follow the format allowed by the JMX specification.

Operator response: Correct the name so that it is a valid MBean name or MBean pattern.

KQZ0091E **Could not write to the output directory *outputDir*.**

Explanation: The current user is not able to write to the chosen output directory.

Operator response: The agent builder must generally be run as root in order to install directly into an ITM installation on the system.

KQZ0092E **Could not create the temporary directory *tempdir*.**

Explanation: The current user is not able to write to the temporary directory. It may be full, or the user may not have permission.

Operator response: Free up some space in the temporary directory if it is full, or run as a user with authority to write there.

KQZ0093E **Could not set execute permission on the installation script *script*.**

Explanation: The installation script must have execute permission set on it before it can be launched, and the attempt to set this failed.

Operator response: Make sure that the script was actually created and that the chmod executable is in the PATH.

KQZ0094E **The installation of the agent on the local system failed. For more information see the installation log in: *log***

Explanation: The installation failed. The reason should be in the installation log.

Operator response: Consult the installation log and correct any problems found there.

KQZ0095E **Could not launch the installation script** *script*

Explanation: The builder was unable to launch the installation script. It might not be executable, or the system shell might not be found.

Operator response: Ensure that the shell (sh) is in the PATH and that the user has permission to run it.

KQZ0096E **A navigator group with identifier** *id* **was defined previously.**

Explanation: This is an informational message only.

KQZ0097E **Navigator group "***navgroup***" does not contain at least** *attribute_group_count* **attribute groups.**

Explanation: This is an informational message only.

KQZ0098E **Navigator group "***navgroup***" does not contain any attribute groups.**

Explanation: This is an informational message only.

KQZ0099E **The affinity is invalid. See the documentation for rules about affinities.**

Explanation: The affinity tag contains an unsupported character.

Operator response: An affinity tag must start with a letter, number, or percent.

The remaining characters can be alphanumeric.

If the affinity starts with a percent, the affinity must contain a period that is not in the second or final position.

KQZ0100E **Unknown ITM 5x interp:** *interp*

Explanation: The specified ITM 5x interp is not recognized.

Operator response: Specify one of the following ITM 5x interps: aix4-r1 hpux10 linux-ix86 linux-ppc linux-s390 os2-ix86 os400 solaris2 solaris2-ix86 w32-ix86

KQZ0101W **You are about to commit this level of the agent. Only perform this action when you are done testing your agent and are ready to ship it.**

Explanation: There are a limited number of versions available so only commit the level when you are ready to ship the agent.

Operator response: Click OK to commit the level or cancel to abort.

KQZ0102E **Select whether the filters are inclusive or exclusive.**

Explanation: The inclusive selection causes the attribute to be processed if the attribute value matches the filter set. The exclusive selection causes the attribute to be processed if the attribute value does not match the filter set.

Operator response: Select either the Inclusive or Exclusive radio button.

KQZ0103E **Select Match all filters or Match any filter.**

Explanation: Match all filters causes a match when all of the filters match the attribute value (logical and). Match any filter causes a match when any of the filters match the attribute value (logical or).

Operator response: Select either the Inclusive or Exclusive radio button.

KQZ0104E **Enter a string which separates fields in a record.**

Explanation: The separator Text selection allows you to specify the string which is used to separate fields in a record.

Operator response: Enter the separator text or select another option for defining fields.

KQZ0105E **Enter two strings, one which identifies the beginning of a field and one which identifies the end of the field.**

Explanation: The Begin and End Text selection allows you to specify the string which precedes the beginning of an attribute value, and one which follows the end of the attribute value.

Operator response: Enter both the begin and end separator text or select another option for defining fields.

KQZ0106E **Enter the name of the source file.**

Explanation: Each source file must have a name that identifies the source file being monitored.

Operator response: Enter the name of the source file.

KQZ0107E **Enter the text found at the beginning of a line that indicates the end of the record.**

Explanation: The pattern selection allows you to enter a string to identify the end of a multi-record record.

Operator response: Enter the pattern or make another selection for identifying a record in a log file. The record is used to collect a single group of attributes.

KQZ0108E Enter the maximum number of non-blank lines that comprise a record.

Explanation: The rule selection allows you to identify the maximum number of lines that form a record. Blank lines, which contain no characters (not even space characters) are not included in the count. The record is used to collect a single group of attributes.

Operator response: Enter the maximum number of non-blank lines or make another selection for identifying a record in a log file.

KQZ0109E The interval for creating a summary record must be from 60 to 86400 seconds (one minute to one day).

Explanation: The summary selection allows you to summarize the detailed data from a log file. You must specify the interval, in seconds, that this summary is to take place. The interval must be at least 60 seconds but not more than 86400 seconds (one day).

Operator response: Enter the summary interval or uncheck the Summary checkbox.

KQZ0110E Enter the number of records, from 1 to 5000, to be processed when file monitoring begins.

Explanation: With the selection "Process existing records from the file", a fixed number of records of the file are processed when monitoring begins. That number must be from 1 to 5000.

Operator response: Enter a number from 1 to 5000 or make a different record processing selection.

KQZ0111E Cannot remove attribute *attr* because it was created in a previous version of the agent.

Explanation: This is an informational message only.

KQZ0112E Cannot remove attribute group *name* because it was created in a previous version of the agent.

Explanation: This is an informational message only.

KQZ0113E Cannot remove navigator group *name* because it contains an attribute group that was created in a previous version of the agent.

Explanation: This is an informational message only.

KQZ0114E Each log file must have a unique label.

Explanation: When two or more files are specified, each must have a different label. If desired, the label can be included in an attribute by itself. The label cannot contain spaces, angle brackets, or @.

Operator response: Enter a label for each file.

KQZ0115E The label cannot contain spaces, angle brackets, at sign (@), or non-ASCII characters.

Explanation: When two or more files are specified, each must have a different label. If desired, the label can be included in an attribute by itself. The label cannot contain spaces, angle brackets, @ or non-ASCII characters.

Operator response: Remove the space from the label.

KQZ0116E Label *label* is entered more than once.

Explanation: When two or more files are specified, each must have a different label. If desired, the label can be included in an attribute by itself. The label cannot contain spaces, angle brackets, or @.

Operator response: Enter a unique label for each file.

KQZ0118E No comparison string entered.

Explanation: A string filter compares an attribute value, or some portion of the attribute value, with a comparison string. The comparison string cannot be blank.

Operator response: Enter a value for the comparison string.

KQZ0119E The Agent Builder encountered an error while generating file *name*.

Explanation: The builder encountered an unexpected error.

Operator response: Ensure you have the correct permissions to the output directory. If the error persists, try restarting the Agent Builder.

KQZ0120W The Agent Builder determined that a TEMS is not installed in directory *directory*. Manually install the TEMS support files after the agent installation completes.

Explanation: A TEMS server is not present in the specified directory.

Operator response: Generate a zip file, transfer it to the TEMS system and run the TEMS installation script.

KQZ0121W **The Agent Builder determined that a TEPS is not installed in directory *directory*. Manually install the TEPS support files after the agent installation completes.**

Explanation: A TEPS server is not present in the specified directory.

Operator response: Generate a zip file, transfer it to the TEPS system and run the TEPS installation script.

KQZ0122E **Enter a value to compare with the attribute.**

Explanation: A number filter compares an attribute value with a comparison value. The comparison value cannot be blank.

Operator response: Enter a value for the comparison number.

KQZ0123E **No regular expression entered.**

Explanation: A regular expression filter compares an attribute value, or some portion of the attribute value, with a regular expression. The regular expression cannot be blank.

Operator response: Enter a value for the regular expression.

KQZ0124E **The filter cannot contain the character sequence "*string*".**

Explanation: A filter is a function applied to an attribute value that returns TRUE or FALSE. Filters can be combined together to form a filter set. The combined results of the filters in the filter set, along with filter sets from other attributes in the attribute group, determine whether the current sample is presented to TEMS or discarded.

A filter cannot contain certain sequences of characters as that would make the resulting filter statement ambiguous when parsed at runtime.

Operator response: Remove the offending sequence from the expression in the filter.

KQZ0125E **No replacement string entered.**

Explanation: Regular expression matches can have the matched portion of the attribute value replaced with a different value. If you want this replacement to take place, you must specify a replacement string.

Operator response: Enter a replacement string or uncheck the replacement value checkbox.

KQZ0126E **Offset must be a number.**

Explanation: The offset is the number of characters at the beginning of the attribute value that do not participate in the filter's matching operation. An offset of 0 means that the entire attribute value is used in the filter.

Operator response: Enter an offset.

KQZ0127E **An attribute can be added only when the attribute type has been set to a string or numeric type.**

Explanation: Different types of filters are applied to string vs. numeric attributes. Therefore the attribute type must be set before a filter can be defined.

Operator response: Return to the basic attribute information and select an attribute type.

KQZ0130E **The affinity tag is invalid. See the documentation for rules about affinity tags.**

Explanation: The affinity tag contains an unsupported character.

Operator response: An affinity tag must start with AFF_ and contain only letters.

KQZ0134E **An agent with product code "*pc*" has already been defined in project "*project_name*".**

Explanation: Product codes must be unique

Operator response: Enter a unique product code.

KQZ0135E **An agent with affinity "*affinity*" has already been defined in project "*project_name*".**

Explanation: Affinities must be unique

Operator response: Enter a unique affinity.

KQZ0136E **An agent with affinity tag "*tag*" has already been defined in project "*project_name*".**

Explanation: Affinity tags must be unique

Operator response: Enter a unique affinity tag.

KQZ0138E **An agent with name "*name*" has already been defined in project "*project_name*".**

Explanation: Agent names should be unique.

Operator response: Enter a unique agent name.

KQZ0139E **Navigator group "*navgroup*" contains more than *max_attr_groups* attribute groups.**

Explanation: This is an informational message only.

KQZ0140E **A file with the name *name* has already been defined for this command.**

Explanation: This is an informational message only.

KQZ0142E **Navigator group name *id* is reserved for internal use.**

Explanation: This is an informational message only.

KQZ0143E **Enter the number of bytes to be copied into this attribute.**

Explanation: The number entered determines how many bytes are copied from the record to fill this attribute. It must be a positive number up to the maximum number of bytes that this attribute can hold.

Operator response: Enter the number of characters to be read from the log file to fill this attribute value, or select a different field delimiter.

KQZ0144E **The number of characters exceeds the maximum of *maximum_byte_count*.**

Explanation: The number entered determines how many characters are copied from the record to fill this attribute. It must be a positive number up to the maximum number of byte that this attribute can hold.

Operator response: Reduce the number so that it is less than or equal to the maximum number of bytes in this attribute, or select a different field delimiter.

KQZ0145I **The reference to file *file_name* is being removed from the agent. Do you want to remove that file from the project?**

Explanation: This is an informational message only.

KQZ0148E **The *field_name* must contain only ASCII characters.**

Explanation: String values can contain only ASCII characters.

Operator response: Enter only ASCII characters for the value.

KQZ0149W **If you exit the wizard, all changes will be lost. Do you wish to exit?**

Explanation: The user clicked cancel on a wizard.

Operator response: Click Yes to exit or No to return to the wizard.

KQZ0150E **The maximum number of lines that comprise a log file record must be from 2 to 32767.**

Explanation: The rule selection allows you to identify the maximum number of lines that form a record. The record is the portion of the file used to parse a single group of attributes.

Operator response: Enter a number between 2 and 32767 inclusive for the maximum number of lines, or make another selection for identifying a record in a log file.

KQZ0151E **The file, *filename*, does not appear to be an IBM Tivoli Monitoring Agent file.**

Explanation: The Agent Builder was unable to parse the specified file.

Operator response: Specify an IBM Tivoli Monitoring Agent file.

KQZ0153E **This action will remove *section_count* runtime configuration sections or properties. Do you wish to continue?**

Explanation: This is an informational message only.

KQZ0154E **Removing runtime configuration section *section_name* will remove all properties defined in the section. Do you wish to continue?**

Explanation: This is an informational message only.

KQZ0155E **A single quote cannot be included in the file name.**

Explanation: A single quote character is not permitted in the log file name specification. It may be permitted by some operating systems or file systems but is not supported for log file monitoring.

Operator response: Do not enter a single quote in the log file name specification.

KQZ0157I **Canceling the installation may result in the agent or support files being partially installed. Are you sure you want to cancel?**

Explanation: This is an informational message only.

KQZ0158I **The installation script *script* was canceled.**

Explanation: The cancel button was pressed while the installation script was running. The script may have partially run before the script was ended and the agent may be partially installed.

Operator response: Reinstall the agent before trying to run it.

KQZ0159I **The agent generation action was canceled.**

Explanation: The cancel button was pressed while the agent files were being generated.

Operator response: Rerun the Generate Agent action to generate the agent.

KQZ0160I **Can the attribute group to which this attribute belongs produce more than one data row?**

Explanation: A attribute group may or may not be capable of producing more than one row of data. If it can produce more than one row, key attributes allow you to tell the rows apart. A key attribute is being defined, but this attribute group is marked as being able to produce only one row. A key attribute for a attribute group that can only produce one row is not useful.

Operator response: Answer Yes to mark the attribute group as being able to produce more than one data row, or No to leave it marked as being able to produce only a single data row.

KQZ0161W **There is no agent installation script for this operating system. The agent is not supported on this operating system.**

Explanation: The agent builder creates installation scripts only for the operating systems on which the agent is supported. The agent cannot be installed on a operating system it was not intended to run on.

Operator response: Press OK on the dialog. The installation of TEMS and TEPS support will continue normally.

KQZ0162I **The IBM Tivoli Monitoring Agent Builder successfully installed components for agent** *agent_name***.**

Explanation: This is an informational message only.

KQZ0163I **The IBM Tivoli Monitoring Agent Builder successfully created a Solution Installer project for agent** *agent_name***.**

Explanation: This is an informational message only.

KQZ0164I **The IBM Tivoli Monitoring Agent Builder successfully generated a mapping file for agent** *agent_name***.**

Explanation: This is an informational message only.

KQZ0165E **No agent components to install.**

Explanation: This is an informational message only.

KQZ0166E **A separator string cannot be more than** *maximum_character_count* **characters.**

Explanation: The separator Text selection allows you to specify the string which is used to separate fields in a record of the log file.

Operator response: Enter the separator text or select another option for defining fields.

KQZ0167I **Getting classes in namespace** *namespace*

Explanation: This is an informational message only.

KQZ0171I **"***operation_title***" was canceled.**

Explanation: This is an informational message only.

KQZ0174E **The** *field_name* **must contain only letters, numbers, spaces, underscores, and hyphens.**

Explanation: String values can contain only ASCII characters.

Operator response: Enter only ASCII characters for the value.

KQZ0175E **The names of files included in the agent must not be over** *max_name_length* **bytes long. The name** *file_name* **is** *actual_name_length* **bytes long.**

Explanation: The agent generator will not generate agents that include files whose names are too long. This is to prevent path names from becoming longer than what is allowed by the operating system.

Operator response: Remove the file from the agent, give it a shorter name, and add the renamed file to the agent.

KQZ0176E **The trace file** *trace_file_name* **could not be created.**

Explanation: This is an informational message only.

KQZ0177E **Changing the** *pc_or_affinity* **has invalidated the contents of the agent's support files. The files will be removed.**

Explanation: Previously created workspaces, situations, queries and take actions are associated with the agent by the product code and affinity.

Operator response: You will need to re-install the agent and recreate and re-extract any workspaces, situations, custom queries and take actions.

KQZ0179E **Make a selection for** *data_source***.**

Explanation: You must select two attribute groups in order to create a joined attribute group.

Operator response: Select one more attribute group.

KQZ0180E **Cannot join an attribute group with itself.**

Explanation: You must select two different attribute groups in order to create a joined attribute group.

Operator response: Select two different attribute groups.

KQZ0182E **Select an attribute to join on for** *attribute_group***.**

Explanation: When both selected attribute groups are defined as returning multiple rows, you must select an attribute from each in order to correlate the data.

Operator response: Select an attribute from each attribute group.

KQZ0183E **Select a key attribute for the joined attribute group.**

Explanation: By default, the keys for the base attribute groups are used as the keys for the joined attribute group. However, you can specify one attribute to be the key.

Operator response: Select the key attribute.

KQZ0184E **The selection cannot be removed because joined attribute group** *referencing_attribute_group_name* **references attribute group** *referenced_attribute_group_name***.**

Explanation: The attribute group you are trying to delete is referenced in a joined attribute group which is not being deleted.

Operator response: You must delete the joined attribute group before or along with any referenced attribute groups.

KQZ0185E **Cannot remove attribute** *attribute_name* **because it is defined as a join attribute for joined attribute group** *attribute_group_name***.**

Explanation: The attribute you are attempting to delete is defined as the attribute on which to join in a joined attribute group.

Operator response: Modify or remove the joined attribute group if you would like to remove the attribute.

KQZ0186E **Cannot remove attribute** *attribute_name* **because it is defined as the key for joined attribute group** *attribute_group_name***.**

Explanation: The attribute you are attempting to delete is defined as the key attribute in a joined attribute group.

Operator response: Modify or remove the joined attribute group if you would like to remove the attribute.

KQZ0187E **Cannot install the agent to** *directory* **because the Tivoli Enterprise Monitoring Agent Framework is not present in that location.**

Explanation: In order to install and run the agent, the Tivoli Enterprise Monitoring Agent Framework must be installed in the same location you wish to install the agent.

Operator response: Install the Tivoli Enterprise Monitoring Agent Framework or install the agent into a different location.

KQZ0188E **Cannot install the agent to** *directory* **because no other IBM Tivoli Monitoring Agents are present in that location.**

Explanation: In order to install and run the agent, at least one other agent must be installed.

Operator response: Install the another agent or install this agent in another location.

KQZ0189W **The file** *file_name* **already exists in the project. Do you want to overwrite the existing files with the one from** *new_file***?**

Explanation: A file with the same name already exists.

Operator response: Click "Yes" to overwrite the file or "No" to keep the existing file.

KQZ0190E **A section with label** *section_label* **already exists.**

Explanation: Each configuration section needs to have a different label.

Operator response: Change the label to one that has not been used by another section.

KQZ0191E **A property with label** *property_label* **already exists in this section.**

Explanation: Each configuration property in a section needs to have a different label.

Operator response: Change the label to one that has

not already been used by another property in this section.

KQZ0192E **A choice with label *choice_label* already exists in this property.**

Explanation: Each choice in a configuration property needs to have a different label.

Operator response: Change the label to one that has not already been used by another choice in this property.

KQZ0193E **A choice with value *choice_value* already exists in this property.**

Explanation: Each choice in a configuration property needs to have a different value.

Operator response: Change the value to one that has not already been used by another choice in this property.

KQZ0194E **Configuration section *section_name* does not contain any configuration properties.**

Explanation: A configuration section must contain at least one configuration property.

Operator response: Add a configuration property to the section or delete the section.

KQZ0195E **Return code *return_code* cannot be returned by UNIX or Linux commands, but only UNIX or Linux operating systems are selected.**

Explanation: Return codes outside the range 0 to 255 cannot be returned by UNIX or Linux commands. Only UNIX and Linux operating systems are selected so this return code can never be issued.

Operator response: Add the Windows operating system which allows a wider range of return codes (-2147483648 to 2147483647), change the return code value to be in the range 0 to 255, or remove the return code.

KQZ0196E **Return code *return_code* cannot be returned by UNIX or Linux commands, but only UNIX or Linux commands are defined.**

Explanation: Return codes outside the range 0 to 255 cannot be returned by UNIX or Linux commands. Only UNIX and Linux commands are defined so this return code can never be issued.

Operator response: Add a Windows command which allows a wider range of return codes (-2147483648 to 2147483647), add the Windows operating system to one of the existing commands, change the return code value to be in the range 0 to 255, or remove the return code.

KQZ0197E **The combined length of company identifier and agent identifier cannot exceed *character_count* characters.**

Explanation: Ensure that the combined length of the company identifier and agent identifier does not exceed 11 characters.

Operator response: Modify the company identifier and/or agent identifier to make the combined length 11 or fewer characters.

KQZ0198I **You have made changes to the configuration. Do you want to keep those changes in the agent?**

Explanation: Some changes were made to the configuration while performing some other operation, such as creating a new attribute group. The other operation is being canceled and you have a chance to keep or discard the configuration changes.

Operator response: Click Yes to keep the configuration changes or No to discard them.

KQZ0199E **The specified directory does not appear to be a valid ITM installation directory.**

Explanation: The Agent Builder determined that the specified directory does not appear to be an ITM installation directory.

Operator response: Re-run the task and specify a valid ITM installation directory.

KQZ0200E **Cannot remove the final availability filter because joined attribute group *referencing_attribute_group_name* references the Availability group.**

Explanation: A joined attribute group exists which includes availability information.

Operator response: You need to remove the Availability Joined Attribute Group before you can remove the final availability filter.

KQZ0201E **The property environment variable *duplicate_name* has already been used and is being replaced with *unique_name*.**

Explanation: Configuration property environment variable names must be unique across all configuration properties in the agent. The variable name that was entered has already been used, so it was changed to a unique variable name.

Operator response: Accept the changed variable name or enter a different one.

KQZ0202I **Generating agent** *product code* **(***display_name***).**

Explanation: This is an informational message only.

KQZ0203E **No agent is selected for the generate operation.**

Explanation: No agent file was selected or no Agent Editor had focus.

Operator response: Select an agent file from the Navigator View or give focus to an Agent Editor and rerun the command.

KQZ0204E **The specified version of** *new_version* **is not greater than the current version of** *current_version***.**

Explanation: The version specified must be greater than the current version

Operator response: Specify a version greater than the current version.

KQZ0205E **The specified directory does not exist.**

Explanation: The Agent Builder determined that the specified directory does not exist.

Operator response: Re-run the task and specify a valid ITM installation directory.

KQZ0206E **The specified location is not a directory.**

Explanation: The Agent Builder determined that the specified location is not a directory.

Operator response: Re-run the task and specify a valid ITM installation directory.

KQZ0208E **The specified username or password is incorrect.**

Explanation: The specified username or password is incorrect.

Operator response: Specify the correct username and password.

KQZ0209E **The Tivoli Enterprise Monitoring Server is not running.**

Explanation: The Tivoli Enterprise Monitoring Server is not running.

Operator response: Start the Tivoli Enterprise Monitoring Server and rerun the command.

KQZ0210E **Name "***name***" has already been used by the agent or another subnode.**

Explanation: The agent display name and subnode names should be unique within an agent.

Operator response: Enter a unique name.

KQZ0211E **A subnode with type "***type***" has already been defined.**

Explanation: Subnode types should be unique within an agent.

Operator response: Enter a unique subnode type.

KQZ0212E **The subnode type** *type* **is not valid. It must be 1 to 3 characters and contain only letters and numbers.**

Explanation: The subnode type entered contained invalid characters or too many characters.

Operator response: Enter a subnode type from 1 to 3 characters consisting of letters A-Z or numbers.

KQZ0213E **The combined length of company identifier and subnode identifier cannot exceed** *character_count* **characters.**

Explanation: Ensure that the combined length of the company identifier and subnode identifier does not exceed 14 characters.

Operator response: Modify the company identifier and/or subnode identifier to make the combined length 14 or fewer characters.

KQZ0214I **The following items are selected and will be removed. Do you want to continue?**

Explanation: This is an informational message only.

KQZ0215E **Could not launch command** *script*

Explanation: The builder was unable to launch the command. It might not be executable, or the system shell might not be found.

Operator response: Ensure that the shell (sh) is in the PATH and that the user has permission to run it.

KQZ0216I **The command** *command* **was canceled.**

Explanation: The cancel button was pressed while the command was running. The command may have partially run before the command was ended.

Operator response: None.

KQZ0217I **Do you wish to cancel the command?**

Explanation: This is an informational message only.

KQZ0218I **Based on this data, some attributes might be better represented by other data types. Do you wish to view the suggested data types?**

Explanation: The Agent Builder detected a discrepency in the data types specified and the data returned.

Operator response: Click Yes to view the attributes or No to leave the data types unchaged.

KQZ0219I **Attribute** *attribute* **is defined as numeric, however, the string data was returned. Do you want to change the type to string?**

Explanation: The Agent Builder determined that the the data returned for the attribute is string data, however, the attribute is defined as numeric. This may cause a problem at runtime.

Operator response: Click Yes to change the type or No to leave it as numeric.

KQZ0220E **There is not a command defined for the operating system on which the Agent Builder is currently running.**

Explanation: In order to test a command, you must have a command defined that will run on the operating system on which the Agent Builder is currently running.

Operator response: Define a command for the current operating system and retry the action.

KQZ0222I **The command returned with a non-zero return code. Click the "Check Results" button to view the results.**

Explanation: There are already attributes defined for the script attribute group.

Operator response: Click Yes to replace the attributes or No to keep the attributes.

KQZ0223E **Could not connect to the CIM server on host** *host***.**

Explanation: The agent builder could not connect to the server on the specified host.

Operator response: Check that the host name is typed correctly and that the CIM server is accepting connections.

KQZ0224E **An unknown error occurred when attempting to connect to the CIM server on host** *host***.**

Explanation: The agent builder could not connect to the server on the specified host.

Operator response: Check that the host name is typed correctly and that the CIM server is accepting connections.

KQZ0225E **The selection cannot be moved because joined attribute group** *referencing_attribute_group_name* **references attribute group** *referenced_attribute_group_name***.**

Explanation: A joined attribute group at the agent level (i.e. not part of a subnode definition) can only reference other attribute groups which are also at the agent level. A joined attribute group in a subnode definition can only reference other attribute groups which are in the same subnode definition or at the agent level. This move would cause a joined attribute group to be unable to reference the source attribute groups.

Operator response: If moving a joined attribute group out of a subnode, also move any referenced attribute groups out of that subnode at the same time. If moving an attribute group into a subnode, also move any joined attriubte groups that reference it into the subnode at the same time.

KQZ0226E **Subnode definition "***subnode_name***" does not contain any attribute groups.**

Explanation: This is an informational message only.

KQZ0229E **Two attribute groups which collect event data cannot be joined together.**

Explanation: Both attribute groups selected collects event data rather than sampled data. At least one attribute group must collect sampled data.

Operator response: Choose an attribute group which collects sampled data for one or both attribute groups to be joined.

KQZ0230E **An attribute group that collects event data cannot be joined to an attribute group which can produce more than one row.**

Explanation: One selected attribute group collects event data and the other collects sampled data but can contain more than one row of data in the sample. Only an attribute group which collects a single row in its sample can be joined to an attribute group which collects event data.

Operator response: Choose an attribute group which

collects a single row of sampled data instead of the attribute group which can contain more than one row in its sample.

KQZ0231I IBM Tivoli Monitoring Agent Builder, Version *versionplugin_information*

Explanation: The output lists the version of the IBM Tivoli Monitoring Agent Builder.

Operator response: None.

KQZ0254E The XML element which identifies data rows in this attribute group must be specified.

Explanation: One data row will be collected for each instance of this element in the XML file. Elements and attributes within this element can be used as attribute values.

Operator response: Enter an XML element name.

KQZ0255E The XML tag whose value is collected by this attribute must be specified.

Explanation: Each ITM attribute in an XML file attribute group must be identified by an XML element or XML attribute name.

Operator response: Enter an XML element or attribute name.

KQZ0256E The combined length of the agent company identifier and agent identifier cannot exceed *character_count* **characters if the agent is to have subnodes.**

Explanation: Ensure that the combined length of the company identifier and agent identifier, does not exceed 11 characters.

Operator response: On the Agent Information page, modify the company identifier and/or agent identifier to make the combined length 11 or fewer characters, then add a subnode to the agent. If you do not want to change vital agent information, you will need to create a new agent to contain subnodes.

KQZ0257E Collect namespace operation was canceled.

Explanation: This is an informational message only.

KQZ0258E Collect classes operation was canceled.

Explanation: This is an informational message only.

KQZ0259E Unable to list services on *host***.**

Explanation: This is an informational message only.

KQZ0260I Getting services on *host***.**

Explanation: This is an informational message only.

KQZ0261I The server connection was successful.

Explanation: The connection to the server from the connection wizard was made successfully.

Operator response: Press "Finish" to save the connection definition for browsing the server.

KQZ0262E The server connection was not successful: *exception_message*

Explanation: The connection to the server from the connection wizard was not successful. The message describes the exception that occurred when attempting to make the connection.

Operator response: Modify the parameters and attempt the connection again, press "Finish" to save the connection definition, or press "Cancel" to abandon any changes you have made.

KQZ0263E The following files already exist. Do you want to continue and overwrite the contents of the files?

Explanation: The files that the IBM Tivoli Monitoring Agent Builder is trying to generate already exist.

Operator response: Select Yes to continue and overwrite the files or No to cancel and select a new directory.

KQZ0264E The underlying file for *file_name* **could not be found or has become corrupted. Close this editor and reopen it from the Navigator tree.**

Explanation: The file was deleted or moved while it was being edited.

Operator response: Close this editor for this agent and reopen it from the Navigator tree.

KQZ0265E At least two return codes must be defined.

Explanation: All return codes that the command can return should be defined. The command should return at least two distinct return codes to distinguish between states that the command detects.

Operator response: Add the return code or return codes to complete the command definition.

KQZ0266E **The selection contains at least one global return code. Do you want to continue?**

Explanation: Global return codes can apply to any command in the availability filter, not just the command you are currently editing. If you remove it, you may remove return codes that other commands depend on.

Operator response: Press OK to remove the selected return codes or Cancel to end the action without removing any return codes.

KQZ0267E **The IBM Tivoli Monitoring agent contains errors. The errors must be corrected before the agent can be committed.**

Explanation: The IBM Tivoli Monitoring Agent contains errors that would prevent it from operating correctly.

Operator response: Correct the errors and save the IBM Tivoli Monitoring Agent.

KQZ0268E **The Tivoli Enterprise Monitoring Server logon failed.**

Explanation: This is an informational message only.

KQZ0270E **XML tag** *tag_name* **was not found in the specified file.**

Explanation: This is an informational message only.

KQZ0273E **No override filter comparison string entered.**

Explanation: Regular expression matches can have an additonal pattern that if matched overrides the previous match. You must specify a comparison string.

Operator response: Enter a comparison string or uncheck the override filter checkbox.

KQZ0274E *directory* **is a directory not a filename.**

Explanation: The user specified the name of a directory instead of the name of a file.

Operator response: Specify a file name.

KQZ0275E **No comparison string entered.**

Explanation: Regular expression matches search for a pattern that identifies the record. You must specify a comparison string.

Operator response: Enter a comparison string or select a different record identification option.

KQZ0276E **Cannot remove the Perfmon instance name because it is defined as a join attribute for joined attribute group** *attribute_group_name***.**

Explanation: The Perfmon instance name you are attempting to remove is defined as the attribute on which to join in a joined attribute group.

Operator response: Modify or remove the joined attribute group if you would like to remove the attribute.

KQZ0277E **Cannot remove attribute** *attribute_name* **because it is referenced as** *joined_attribute_name* **in derived attribute** *derived_attribute_name* **in joined attribute group** *joined_attribute_group_name***.**

Explanation: The attribute you are attempting to delete is defined as the key attribute in a joined attribute group.

Operator response: Modify or remove the joined attribute group if you would like to remove the attribute.

KQZ0278E **The** *tivoli_product_component* **was modified. You must save the** *tivoli_product_component* **before you can add another component.**

Explanation: This is an informational message only.

KQZ0279E **Cannot remove attribute** *attribute_name* **because it is referenced in filtered attribute group** *filtered_attribute_group_name***, attribute** *filtered_attribute_name***.**

Explanation: The attribute you are attempting to delete is defined as the key attribute in a joined attribute group.

Operator response: Modify or remove the joined attribute group if you would like to remove the attribute.

KQZ0280I **One or more event summarization attributes should be added to this attribute group. Do you wish to add them now?**

Explanation: Attribute groups that have event summaraziation enabled should have additional attributes.

Operator response: Click "OK" to add the attributes or "Cancel" continue without adding the attributes.

KQZ0281W The command you are attempting to test on the local machine is not defined as running on the local platform. Testing the command locally may produce different results than when the command is run in the agent. It is recommended that you either test the command locally on a supported platform, or test the command remotely using SSH. Do you wish to continue testing the command on the local system?

Explanation: The Agent Builder determined that the the local platform type is not selected for the command being tested.

Operator response: You should add the local platform type to the command definition, test the command locally on a supported platform or test the command remotely using SSH.

KQZ0282E Cannot remove automatically created attribute *attr*.

Explanation: This attribute is needed for this attribute group to function properly. It can only be deleted if the entire attribute group is deleted.

Operator response: You must leave this attribute in place. If you do not want it to sent to the TEMS, you can remove the checkmark from the "Display attribute in the Tivoli Enterprise Portal" flag.

KQZ0283E The row selection XPath contains a syntax error.

Explanation: The XPath contains a syntax error.

Operator response: Edit the row selection XPath to correct the error.

KQZ0284E The XPath for attribute *attr* contains a syntax error.

Explanation: The XPath contains a syntax error.

Operator response: Edit the XPath to correct the error.

KQZ0285E Could not collect data from *url*.

Explanation: An error occurred while trying to collect data from the URL specified.

Operator response: Ensure the URL is correct, the user id and password is correct, and the post data, if applicable, is correct.

KQZ0286E Error collecting data from *url*. The response received was *response*

Explanation: An error occurred while trying to collect data from the URL specified.

Operator response: Ensure the URL is correct, the user id and password is correct, and the post data, if applicable, is correct.

KQZ0287E Cannot remove subnode *name* because it was created in a previous version of the agent.

Explanation: This is an informational message only.

KQZ0288I The XPath specified in the *dialog* dialog does not match the XPath here. Do you wish to update the XPath?

Explanation: This is an informational message only.

KQZ0289E The XPath contains a syntax error.

Explanation: The XPath contains a syntax error.

Operator response: Edit the XPath to correct the error.

KQZ0290E The following configuration variables are used; however, no values are set: *var_list*.

Explanation: A configuration variable is specified, but no value for that variable was found in the test environment.

Operator response: Remove the reference to the configuration variable or set a value in the test environment.

KQZ0291W No rows returned for the queries specified.

Explanation: This is an informational message only.

KQZ02923I Attribute *attribute* is defined as 32-bit; however, the data returned suggests that it should be 64-bit. Do you want to update the attribute to be 64-bit?

Explanation: The Agent Builder determined that the the data returned needs 64-bits to represnt it. Leaving the attribute as a 32-bit attribute may cause an overflow condition.

Operator response: Click Yes to make the attribute 64-bit or No to leave it as it is.

KQZ0292I **The scale for attribute *attribute* is *scale*; however, the data returned suggests that it should be *newscale*. Do you want to update the scale to *newscale*?**

Explanation: The Agent Builder determined that the the data returned should have the specified scale. This will not cause an error at runtime.

Operator response: Click Yes to change the scale or No to leave it as it is.

KQZ0298E **The selection cannot be removed because filtered attribute group *referencing_attribute_group_name* references attribute group *referenced_attribute_group_name*.**

Explanation: The attribute group you are trying to delete is referenced in a filtered attribute group which is not being deleted.

Operator response: You must delete the filtered attribute group before or along with any referenced attribute groups.

KQZ0299E **The selection cannot be moved because filtered attribute group *referencing_attribute_group_name* references attribute group *referenced_attribute_group_name*.**

Explanation: A filtered attribute group at the agent level (i.e. not part of a subnode definition) can only reference other attribute groups which are also at the agent level. A filtered attribute group in a subnode definition can only reference other attribute groups which are in the same subnode definition or at the agent level. This move would cause a filtered attribute group to be unable to reference the source attribute group.

Operator response: If moving a filtered attribute group out of a subnode, also move any referenced attribute groups out of that subnode at the same time. If moving an attribute group into a subnode, also move any filtered attriubte groups that reference it into the subnode at the same time.

KQZ0300E **The selection cannot be removed because configuration section *configuration_section_name* is required by data sources defined in this agent.**

Explanation: The configuration section you are trying to delete contains properties that are needed by data sources in this agent. It cannot be removed unless the data sources are also removed, or in some cases, configured so that the properties are not needed.

Operator response: Delete the data sources that use this configuration section if you do not wish to have it in your agent any longer.

KQZ0301E **The selection cannot be removed because configuration section *configuration_section_name* is needed for configuration of subnode *subnode_type*.**

Explanation: The configuration section you are trying to delete contains properties that are needed when configuring a subnode. They cannot be removed unless the subnode is removed.

Operator response: Delete the subnode if you do not wish to have it in your agent any longer.

KQZ0302E **The selection cannot be removed because configuration property *configuration_property_name* is required by data sources defined in this agent.**

Explanation: The configuration property you are trying to delete is needed by data sources in this agent. It cannot be removed unless the data sources are also removed, or in some cases, configured so that the properties are not needed.

Operator response: Delete the data sources that use this configuration property if you do not wish to have it in your agent any longer.

KQZ0303E **The selection cannot be removed because configuration property *configuration_property_name* overrides a property of the same name in the base agent.**

Explanation: Configuration property overrides can not be added or removed from the context menu on the runtime configuration page or the outline view. fThe configuration property you are trying to delete is needed by data sources in this agent. It cannot be removed unless the data sources are also removed, or in some cases, configured so that the properties are not needed.

Operator response: Delete the data sources that use this configuration property if you do not wish to have it in your agent any longer.

KQZ0304E **An error occurred generating the agent configuration file.**

Explanation: An unknown error occurred generating the agent configuration file.

Operator response: Retry the operation.

KQZ0305I **The agent is taking a longer than expected amount of time to start. Do you want to continue to wait?**

Explanation: The agent hasn't started in the expected amount of time. This could indicate an error.

Operator response: Click "Yes" to continue waiting or "No" to cancel.

KQZ0306E The agent returned an unexpected response.

Explanation: The agent returned a response that was not expected.

Operator response: Retry the operation.

KQZ0307E An error occurred while communicating with the agent.

Explanation: The agent returned an error condition in response to the request.

Operator response: Retry the operation.

KQZ0308W There should be at least one configuration property in subnode *subnode_name* **that the agent can use to differentiate one instance of the subnode from another.**

Explanation: Sample code for Java API provider cannot be created if there is a sampled attribute group in a subnode and there is no configuration property in the subnode. One or more configuration properties are needed to distinguish one subnode instance from another.

Operator response: Add a configuration property to the subnode and save the agent.

KQZ0309I The TEPS server connection was successful up to the loading of the managed system names.

Explanation: The TEPS connection was successful, but a more complete test cannot be completed until a managed system is selected.

Operator response: Select a managed system and press Test Connection again or press Finish.

KQZ0310W The agent process *agent_process_name* **did not end as expected. You may need to end the process manually in order to test this agent again.**

Explanation: The Agent Builder started an agent process in order to test the functionality of the agent. The agent process did not respond in a reasonable amount of time to the request to stop the agent. The process may eventually respond to the request to stop it (for example, after a network connection from an attempted data collection times out), or it may be in an error state and never stop on its own. The same agent cannot be tested again while this process is still running.

Operator response: Wait for a little longer if desired. If the process does not end on its own, end it manually using tools from the operating system or elsewhere.

KQZ0311W The following agent processes did not end as expected. You may need to end the processes manually in order to test these agents again. Processes that did not end: *agent_process_names*

Explanation: The Agent Builder started an agent process for each agent in order to test the functionality of the agent. The agent processes did not respond in a reasonable amount of time to the request to stop the agents. The processes may eventually respond to the request to stop it (for example, after a network connection from an attempted data collection times out), or it may be in an error state and never stop on their own. An agent cannot be tested again while its process is still running.

Operator response: Wait for a little longer if desired. If the processes do not end on their own, end them manually using tools from the operating system or elsewhere.

KQZ0312E The selection cannot be removed because OSLC resource *referencing_attribute_group_name* **references attribute group** *referenced_attribute_group_name***.**

Explanation: The attribute group you are trying to delete is referenced in an OSLC resource which is not being deleted.

Operator response: You must delete the OSLC resource before or along with any referenced attribute groups.

KQZ1000E The Eclipse preferences for the Jlog Plug-in were not found. Reverting to default preferences.

Explanation: The preferences that control the Resource Model Builder logging behavior could not be loaded. The default values will be used instead.

Operator response: The Resource Model Builder installation may be corrupt. Reinstall the program. If this does not correct the error, contact IBM support.

KQZ1002E An error occurred while saving the preference information.

Explanation: The IBM Tivoli Monitoring Agent Toolkit was unable to save the user preferences.

Operator response: Check the Eclipse error log for more information

---

KQZ2008W **There were no situations defined for the application. Do you wish to continue?**

Explanation: There were no situations defined.

Operator response: Select yes to continue with the operation or no to cancel.

KQZ2009E **Unable to connect to the Tivoli Enterprise Portal server on host** *hostname***.**

Explanation: Unable to connect to the TEPS on the specified host.

Operator response: Ensure that the system is up and the Tivoli Enterprise Portal Server Service is running.

KQZ2010E **Unable to extract the application support information from the Tivoli Enterprise Portal Server on host** *hostname***.**

Explanation: Unable to extract information from the TEPS on the specified host.

Operator response: Ensure that the system is up and the Tivoli Enterprise Portal Server Service is running.

KQZ2011I **There were no further operations to perform.**

Explanation: There were no situations to extract and the option to import workspaces was not selected.

Operator response: Re-run the task and ensure that the options for importing situations are correct.

KQZ2013W **The operation was canceled. Would you like to continue without importing situations?**

Explanation: The user canceled the operation to select the situations to import.

Operator response: Click yes to continue or no to cancel.

KQZ2014E **Error copying file** *src* **to** *dst***.**

Explanation: Could not copy the specified file.

Operator response: Ensure the disk is not full and that the tool has the required permissions.

KQZ2015W **The toolkit has determined that Fix Pack 1 is not installed on the Tivoli Enterprise Portal Server. This fix pack is required in order to export workspaces using the toolkit. Do you wish to continue and only export situations?**

Explanation: Fix Pack 1 is not installed on the Tivoli Enterprise Portal Server

Operator response: Create the workspaces on a system that is running Fix Pack 1.

KQZ2016E **The toolkit has determined that Fix Pack 1 is not installed on the Tivoli Enterprise Portal Server. This fix pack is required in order to export workspaces using the toolkit.**

Explanation: Fix Pack 1 is not installed on the Tivoli Enterprise Portal Server

Operator response: Create the workspaces on a system that is running Fix Pack 1.

KQZ2017I **There were no further operations to perform.**

Explanation: There were no situations to extract and the option to import workspaces and queries were not selected.

Operator response: Re-run the task and ensure that the options for importing situations are correct.

KQZ2018W **There were no user defined queries for the application. Do you wish to continue?**

Explanation: There were no user defined queries.

Operator response: Select yes to continue with the operation or no to cancel.

KQZ2019W **The operation was canceled. Would you like to continue without importing queries?**

Explanation: The user canceled the operation to select the queries to import.

Operator response: Click yes to continue or no to cancel.

KQZ2020E **No managed systems found in response from TEPS query.**

Explanation: The system queried the TEPS to get the list of all known managed systems, and none were returned.

Operator response: Start the operating system monitoring agent on the system you want to manage. Make sure it is configured to connect to the same TEMS that the TEPS connects to.

KQZ2023E **Unable to determine the Tivoli Enterprise Portal Server version on *host*.**

Explanation: The Solution Installer was unable to check the TEPS version.

Operator response: Ensure that the TEPS is running on the specified system and that it is accessible from this system.

KQZ2024W **There were no user defined take actions for the application. Do you wish to continue?**

Explanation: There were no user defined take actions.

Operator response: Select yes to continue with the operation or no to cancel.

KQZ2025W **The operation was canceled. Would you like to continue without importing take actions?**

Explanation: The user canceled the operation to select the take actions to import.

Operator response: Click yes to continue or no to cancel.

KQZ2027E **Select a TEPS host name from the drop-down list or add a new host to the list.**

Explanation: You must select or define the host to contact to retrieve the desired information.

Operator response: Select or define a new TEPS host.

KQZ2028E **No Windows managed systems found in response from TEPS query.**

Explanation: The system queried the TEPS to get the list of all known Windows managed systems, and none were returned.

Operator response: Start the Windows system monitoring agent on the system you want to manage. Make sure it is configured to connect to the same TEMS that the TEPS connects to.

KQZ2029E **Select a managed system from the drop-down list.**

Explanation: You must select a managed system to retrieve the desired information.

Operator response: Select a managed system.

KQZ2030E **Could not connect to the Tivoli Enterprise Portal Server on host *hostname*.**

Explanation: Either the host name is incorrect or there is no TEPS server running on the host.

Operator response: Correct the host name or ensure the TEPS is running.

KQZ2032E **The Tivoli Enterprise Portal Server on host *hostname* has not completed initialization.**

Explanation: The Tivoli Enterprise Portal Server has started but is not yet ready to handle requests.

Operator response: Wait a few moments for the server to complete initialization and retry the operation.

KQZ2036W **Establishing the connection to the Tivoli Enterprise Portal Server is taking longer than expected. Do you want to continue waiting?**

Explanation: This is an informational message only.

KQZ23553I **Support files for product, *project*, were found in the project. Do you wish to append to the existing files or overwrite the existing files?**

Explanation: This is an informational message only.

KQZ2924E **The directory *directory* already contains an agent image. Do you want to replace it with the new image?**

Explanation: The directory already contains an agent image.

Operator response: Select yes to replace the agent image in the directory, or select no to cancel the operation. After canceling the operation, you can select a different directory in which to create the agent image.

KQZ2925E **A file named *file_name* already exists.**

Explanation: A file already exists in the directory with the specified name.

Operator response: Delete the existing file or rename the new file.

KQZ2926W **The specified files will be removed from the workspace mibs directory when you click "OK" in the "Manage Custom MIBs" dialog.\n If you do not wish to delete the files, click "Cancel" in the dialog to revert the directory to it's previous contents.**

Explanation: The specified MIB files will be deleted.

Operator response: Click "OK" in the dialog to delete the files or "Cancel" to keep the files.

KQZ2927E **Class files from folder** *folder_name* **could not be found to create jar file** *jar_file_name***.**

Explanation: When an agent containing a Java API attribute group is generated, class files for the Java custom provider application are written to a jar file when the agent is generated. The folder that should contain those class files could not be found.

Operator response: Make sure the Java source files for the Java custom provider application exist, and are being compiled without any compiler errors, and the Java output folder is named "bin".

KQZ3000E *path* **is already in the list.**

Explanation: The same path cannot be entered in the list twice.

Operator response: Select another path or press Cancel to keep the existing paths.

KQZ3001E **File** *file_path* **could not be found.**

Explanation: This is an informational message only.

KQZ3002E **Host** *hostname* **has already been defined. Either enter a new host name or select the host name from the drop down box.**

Explanation: The specified host has already been defined.

Operator response: Specify a different host name.

KQZ3003E **The maximum length of the project name is 100 characters.**

Explanation: The Eclipse project name is too long. The length is limited to prevent the maximum fully-qualified path length of files in the project from exceeding operating system limits.

Operator response: Shorten the name of the project.

KQZ3004E **Unable to run the tacmd command.**

Explanation: The program was unable to launch the IBM Tivoli Monitoring tacmd command.

Operator response: Ensure the ITM install directory is specified correctly.

KQZ3005E **A TEMA is not installed in directory** *directory***. The agent will not be installed on this system.**

Explanation: The product requires that the Tivoli Enterprise Monitoring Agent Framework be installed on the system.

Operator response: Install the Tivoli Enterprise Monitoring Agent Framework by installing the OS agent.

KQZ3006E **The product requires ITM version** *min_ver***, however, version** *version* **was found.**

Explanation: The product requires a version of ITM that is not present in the directory specified.

Operator response: Either change the minimum level of ITM required by the agent or upgrade the ITM installation on the system.

KQZ3007E **The product requires ITM version** *min_ver***, however, the Agent Builder was unable to determine the version of ITM installed.**

Explanation: An unexpected error occurred when trying to determine the version of ITM installed on the system.

Operator response: Either change the minimum level of ITM required by the product or upgrade the ITM installation on the system.

KQZ3008I **Cancel checking for the presence of a** *component***.**

Explanation: The user clicked cancel while checking for the presence of and ITM component.

Operator response: Click Yes to cancel or No to continue waiting.

KQZ3009E **A Tivoli Enterprise Monitoring Server was not found in** *directory***.**

Explanation: A TEMS is not installed in the specified directory..

Operator response: Specify a directory that contains a TEMS.

KQZ3011E **Action Failed. See the agent log file for more information.**

Explanation: This is an informational message only.

KQZ3012I Action successfully completed.

Explanation: This is an informational message only.

KQZ3013E Action Failed. Monitors are not supported for this connection.

Explanation: This is an informational message only.

KQZ3014E Action Failed. The Object Name is invalid.

Explanation: This is an informational message only.

KQZ3015E Action Failed. The operation name is invalid.

Explanation: This is an informational message only.

KQZ3016E Action partially failed. See the agent log file for more information.

Explanation: This is an informational message only.

KQZ3021I Cancel logging onto the IBM Tivoli Monitoring Server?

Explanation: This is an informational message only.

KQZ3022E The operation does not have enough information to log on.

Explanation: This is an informational message only.

KQZ3025I Cancel running the IBM Tivoli Monitoring query?

Explanation: This is an informational message only.

KQZ3026I Cancel loading the queries into the the IBM Tivoli Monitoring Server?

Explanation: This is an informational message only.

KQZ3029E "*prop_name*" must be unique but "*prop_value*" has already been used.

Explanation: The property must be unique but this value has already been used in another object.

Operator response: Enter a different value for the property.

KQZ3030E "*prop_name*" is required.

Explanation: A value must be entered for the property.

Operator response: Enter a value for the property.

KQZ3031E The specified user does not have workspace administration permission.

Explanation: The specified user must have workspace administration permission.

Operator response: Specify a user that has workspace administration permission.

KQZ3032I The ITM logon operation was canceled.

Explanation: The cancel button was pressed while attempting to login to the TEMS.

Operator response: Retry the operation.

KQZ3033E The specified user name or password is incorrect.

Explanation: The specified user name or password is incorrect.

Operator response: Specify the correct user name and password.

KQZ3034E The Tivoli Enterprise Monitoring Server is not running.

Explanation: The Tivoli Enterprise Monitoring Server is not running.

Operator response: Start the Tivoli Enterprise Monitoring Server and rerun the command.

KQZ3035E The Tivoli Enterprise Monitoring Server logon failed.

Explanation: This is an informational message only.

KQZ3036I The file "*file_name*" has been changed on the file system. Do you want to overwrite the changes made on the file system?

Explanation: The file has been changed from outside the agent editor. Those changes can be overwritten by the copy of the file currently in the agent editor, or can be retained without overwriting them.

Operator response: Select "Yes" if you want to overwrite the changes made outside the agent editor with changes from the agent editor, or "No" if you want to keep the changes on disk and not save the contents of the agent editor for that file.

KQZ3037I Could not create directory *dir*

Explanation: The Solution Installer was unable to create the directory.

Operator response: Ensure that the disk is not full and that the plug-in has the correct permissions.

KQZ3038E **Error creating file *filename*.**

Explanation: Could not create the specified file.

Operator response: Ensure the disk is not full and that the tool has the required file permissions.

KQZ3039W **The condition of the conditional operator is *actual_type* but *actual_type* is expected.**

Explanation: This is an informational message only.

KQZ3040W **The "true" operand of the conditional operator is *actual_type* but *expected_type* is expected.**

Explanation: This is an informational message only.

KQZ3041W **The "false" operand of the conditional operator is *actual_type* but *expected_type* is expected.**

Explanation: This is an informational message only.

KQZ3042E **Operator *operator* must have 1 operand.**

Explanation: This is an informational message only.

KQZ3043E **Operator *operator* is not recognized.**

Explanation: This is an informational message only.

KQZ3044E **"*Java_identifier*" contains character "*bad_character*" which is not allowed in a Java identifier in field "*field_name*".**

Explanation: The value you entered will be used as a Java identifier, but it contains one or more characters that are not allowed in a Java identifier. Only letters, numbers, the dollar sign '$', and the underscore '_' are allowed in Java identifiers.

Operator response: Remove the character that is not allowed.

KQZ3045E **"*Java_identifier*" starts with character "*bad_character*" which is not allowed is not allowed as the first character of a Java identifier in field "*field_name*".**

Explanation: The value you entered will be used as a Java identifier, but it starts with a character that cannot be the first character in a Java identifier. Only letters, the dollar sign '$', and the underscore '_' are allowed to start a Java identifier.

Operator response: Replace the first character with a valid one.

KQZ3046E **Field "*field_name*" must contain a value.**

Explanation: The field does not have a value.

Operator response: Enter a value in the field.

KQZ3047W **Field "*field_name*" value "*Java_class_name*" should start with an upper-case letter.**

Explanation: The value you entered will be used as a Java class name, but it does not start with an upper-case letter. It is recommended that Java class names start with an upper-case letter.

Operator response: Replace the first character with an upper-case letter.

KQZ3048W **Field "*field_name*" value "*Java_package_name*" should not start with an upper-case letter.**

Explanation: The value you entered will be used as a Java package name, but it starts with an upper-case letter. It is recommended that Java package names not start with an upper-case letter.

Operator response: Replace the upper-case letter with a lower-case letter.

KQZ3049E **Character "*bad_character*" is not allowed in file name value "*file_name*" in field "*field_name*".**

Explanation: You should enter a valid file name without any path components.

Operator response: Remove the character that is not valid.

KQZ3050E **"*Java_package_name*" is not a valid Java package name in field "*field_name*".**

Explanation: The value you entered will be used as a Java package name, but it does not contain a valid Java identifier in each portion of the package name. The package name cannot contain a package separator character without an identifier on either side of it.

Operator response: Remove any leading or trailing dots in the package name, or dots that are next to other dots without an intervening identifier.

KQZ3051E **Directory "*directory_name*" does not exist.**

Explanation: This is an informational message only.

KQZ3052E **File "*file_name*" does not exist in directory "*directory_name*".**

Explanation: This is an informational message only.

KQZ3053E **"*keyword*" is a Java keyword and therefore not a valid Java identifier in field Field "*field_name*".**

Explanation: The value you entered will be used as a Java identifier, but it contains one or more characters that are not allowed in a Java identifier. Only letters, numbers, the dollar sign '$', and the underscore '_' are allowed in Java identifiers.

Operator response: Remove the character that is not allowed.

KQZ3054E **Field "*field_name*" contains an empty Java package name.**

Explanation: The package name you entered is empty or contains only whitespace.

Operator response: Enter a valid Java package name.

KQZ3505E **Field "*field_name*" can only contain letters and digits.**

Explanation: This is an informational message only.

KQZ3506E **Field "*field_name*" can only contain letters, digits and characters "*allowable_characters*".**

Explanation: This is an informational message only.

KQZ3507E **Field "*field_name*" can only contain letters.**

Explanation: This is an informational message only.

KQZ3508E **Field "*field_name*" can only contain letters and characters "*allowable_characters*".**

Explanation: This is an informational message only.

KQZ3540E **Field "*field_name*" can only contain digits and characters "*allowable_characters*".**

Explanation: This is an informational message only.

KQZ3541E **Field "*field_name*" can only contain characters "*allowable_characters*".**

Explanation: This is an informational message only.

KQZ3542E **Field "*field_name*" can only contain characters "*allowable_characters*".**

Explanation: This is an informational message only.

KQZ3543E **Field "*field_name*" must be an integer from *lower_bound* to *upper_bound*.**

Explanation: This is an informational message only.

KQZ3544E **Field "*field_name*" must contain Java package and class name. The class cannot be in the default package.**

Explanation: The field does not have both a package and class name.

Operator response: Enter a Java package and class name in the field.

KQZ3545E **No file name was specified in field "*field_name*".**

Explanation: A required file name is missing.

Operator response: Enter a file name in the field.

KQZ3546E **The file name "*file_name*" in field "*field_name*" must not contain leading or trailing spaces.**

Explanation: The file name started with or ended with a whitespace character, and that is not allowed for this file name.

Operator response: Remove the leading or trailing spaces from the file name.

KQZ3547E **The file name "*file_name*" in field "*field_name*" contains a non-printable character.**

Explanation: The file name contains a non-printable character.

Operator response: Remove any non-printable character from the file name.

KQZ3548E **The file name "*file_name*" in field "*field_name*" must not contain any path separators.**

Explanation: The path separator / or \ was found in the file name. The field needs only a file name without path information.

Operator response: Remove the path separator characters and any path information from the file name.

KQZ3549E **The file name "*file_name*" in field "*field_name*" must not contain a "*invalid_character*" character.**

Explanation: The characters < > : " | ? * are not valid in file names on all files systems and therefore are not allowed in an Agent Builder file name.

Operator response: Remove the character from the file name.

KQZ3550W **The file name "*file_name*" in field "*field_name*" contains a space character.**

Explanation: The space character, while a legal character, can be confusing if used in a script and is discouraged on UNIX and Linux file systems

Operator response: Remove the space character from the file name.

KQZ3551W **The file name "*file_name*" in field "*field_name*" does not end with "*file_type*".**

Explanation: The file name does not end with the expected file type. The file type is the portion of the file name after the last period in the name.

Operator response: Enter the correct file type at the end if the file name, after a period.

KQZ3552W **The file name "*file_name*" in field "*field_name*" contains only the file type.**

Explanation: The file name does not contain any characters before the period and file type.

Operator response: Enter characters in the file name before the period that separates the file name from file type.

KQZ3554E **A project named *project* already exists in the workspace.**

Explanation: Project names must be unique within a workspace.

Operator response: Enter a unique project name.

KQZ3555I **Successfully converted the Solution Install Project to an Application Support Extension Project.**

Explanation: This is an informational message only.

KQZ3556E **The product code must start with the letter K.**

Explanation: The product code contains an unsupported character.

Operator response: An product code must contain only three characters.

The first character must be "K".

The final two characters must be alphanumeric.

KQZ3557E **The the character "*character*" is not supported in an IBM Tivoli Monitoring Agent Builder product code.**

Explanation: The product code contains an unsupported character.

Operator response: An product code must contain only three characters.

The first character must be "K".

The final two characters must be alphanumeric.

KQZ3558E **The product code "*product_code*" is reserved by IBM.**

Explanation: The product code is reserved by IBM and cannot be used in an IBM Tivoli Monitoring Agent Builder agent.

Operator response: An product code must contain only three characters.

The first character must be "K".

The final two characters must be alphanumeric.

KQZ3559E **The product code must be three characters long.**

Explanation: The product code is too long or too short.

Operator response: An product code must contain only three characters.

The first character must be "K".

The final two characters must be alphanumeric.

KQZ3560E **An Application Support Extension image for *pc* exists in directory *dir*. Do you want to replace the existing image with a new one?**

Explanation: The output directory already contains an install image for this Application Support Extension.

Operator response: Click "Yes" to replace the image or "No" to cancel the operation.

KQZ3561E **The operation was canceled.**

Explanation: The operation was canceled.

Operator response: Re-run the operation.

KQZ3562E **Successfully created the install image for Application Support Extension *pc*.**

Explanation: This is an informational message only.

KQZ3563W **The filter formula in attribute group *attribute_group_name* evaluates to *actual_type* but should evaluate to *expected_type*.**

Explanation: This is an informational message only.

KQZ3564E **Error accessing all files needed for the consolidation of *file_name*.**

Explanation: This is an informational message only.

KQZ3565E **The "*file_name*" field must be set to test the connection.**

Explanation: The field is required to make a connection to a remote host.

Operator response: Set the field value and press the Test Connection button again.

KQZ3566E **Field "*field_name*" can hold only printable ASCII characters.**

Explanation: Some fields can cause errors in an agent if they include any characters other than printable ASCII characters.

Operator response: Enter only printable ASCII characters for the value.

KQZ3567E **Expression "*literal_token*" is not a valid string literal value.**

Explanation: This is an informational message only.

KQZ3568E **Expression "*literal_token*" is not a valid numerical value value.**

Explanation: This is an informational message only.

KQZ359E **Field "*field_name*" can only contain digits.**

Explanation: This is an informational message only.

KQZ4000E **The maximum length of the project name is 100 characters.**

Explanation: The Eclipse project name is too long. The length is limited to prevent the maximum fully-qualified path length of files in the project from exceeding operating system limits.

Operator response: Shorten the name of the project.

KQZ4001I **Copy the selected files into the Remote Deploy project.**

Explanation: This is an informational message only.

KQZ4002I **Generate the final remote deploy bundle.**

Explanation: This is an informational message only.

KQZ4004E **The bundle identifier must be an alphanumeric string with a length between 3 and 31 characters that does not start with a K.**

Explanation: The bundle identifier must be between 3 and 31 alphanumeric characters and cannot start with the letter K.

Operator response: Correct the bundle identifier.

KQZ4005E **The version must be 9 digits in length.**

Explanation: The version is in the format VVRRMMFFF and must be 9 digits long.

Operator response: Specify a 9 digit version.

KQZ4006E **A prerequisite with identifier, *id*, has already been defined.**

Explanation: The prerequisite is already defined.

Operator response: Specify a different prerequisite.

KQZ4007E **The bundle identifier must be an alphanumeric string with a length between 3 and 31 characters.**

Explanation: The bundle identifier must be between 3 and 31 alphanumeric characters.

Operator response: Correct the bundle identifier.

KQZ4008I **Copying file *id*.**

Explanation: This is an informational message only.

KQZ4009E **A file named *id* already exists in the project. Do you want to overwrite the file?**

Explanation: The file already exists in the project.

Operator response: Click Yes to overwrite the file or No to skip.

KQZ4010E **The location specified is not a directory.**

Explanation: The specified location is not a directory.

Operator response: Specify a directory.

KQZ4011I **Cancel the installation to the local TEMS depot?**

Explanation: The user opted to cancel the install.

Operator response: Click Yes to cancel or No to continue.

KQZ4012E **Remote Deploy bundle, *bundle_id*, was modified. You must save the bundle information before you can generate it.**

Explanation: This is an informational message only.

KQZ4013I **Successfully generated the Remote Deploy bundle in *directory*.**

Explanation: This is an informational message only.

KQZ4014I **Successfully added the bundle to the local TEMS depot.**

Explanation: This is an informational message only.

KQZ4015E **Error adding the bundle to the local TEMS depot.**

Explanation: This is an informational message only.

KQZ4016I **The bundle generation was canceled.**

Explanation: This is an informational message only.

KQZ4017I **The specified directory already contains a Remote Deploy bundle. Do you wish to continue?**

Explanation: This is an informational message only.

KQZ4018I **The IBM Tivoli Monitoring Depot already contains this version of the bundle. Do you want to continue?**

Explanation: This is an informational message only.

KQZ4019I **The bundle already existed so no work needed to be done.**

Explanation: This is an informational message only.

KQZ4020I **The following operating systems will be added to the bundle: *new_operating_systems*.**

Explanation: This is an informational message only.

KQZ4021I **The following operating systems were removed from the bundle. References to these operating systems in commands and prerequisites will also be removed: *new_operating_systems*.**

Explanation: This is an informational message only.

KQZ4022E **The description cannot be empty.**

Explanation: This is an informational message only.

KQZ4023E **The copy location cannot be empty.**

Explanation: This is an informational message only.

KQZ4024E **No operating systems selected.**

Explanation: This is an informational message only.

KQZ4025E **No files are included in the bundle.**

Explanation: This is an informational message only.

KQZ4026E **No commands defined for operating systems** *operating_system*

Explanation: This is an informational message only.

KQZ4027E **The command string cannot be empty.**

Explanation: This is an informational message only.

KQZ4029E **Invalid command type.**

Explanation: This is an informational message only.

KQZ4030E **The remote deploy bundle contains errors as listed in the Problems view. The errors must be corrected before the bundle can be generated.**

Explanation: The remote deploy bundle contains errors as listed in the Problems view.

Operator response: Correct the errors and save the bundle.

KQZ4031W **This remote deploy bundle contains warnings which are indications of potential problems. Attempting to deploy a bundle that contains warnings may cause unpredictable results. Do you wish to continue the generation of the bundle?**

Explanation: The IBM Tivoli Monitoring Agent Builder generated warnings for the bundle. The warnings can be viewed in the Problems View. Bundles with warnings may not function properly.

Operator response: Select yes to continue generation. Select no if you wish to review the warnings. The warnings can be viewed in the Problems View. Select Window->Show View->Problems if the view is not visible.

KQZ41002I **Do you want to cancel the generation of the remote deploy bundle?**

Explanation: This is an informational message only.

KQZ41003I **Successfully created the probe bundle and the configuration bundle project**

Explanation: This is an informational message only.

KQZ41004I **Successfully added the probe bundle to the local TEMS depot and created the configuration bundle project**

Explanation: This is an informational message only.

KQZ41005E **Error adding the probe bundle to the local TEMS depot.**

Explanation: This is an informational message only.

KQZ41006E **Error creating the configuration bundle project.**

Explanation: This is an informational message only.

KQZ41007E **Error creating the Remote Deploy descriptor file for the probe bundle.**

Explanation: This is an informational message only.

KQZ41008E **Error extracting files from the package archive.**

Explanation: This is an informational message only.

KQZ4100E **Error opening the specified archive file.**

Explanation: An exception occurred while attempting to open the specified archive file.

Operator response: Ensure that you have access to the file and it is a valid archive file.

KQZ4101E **The specified file is not a valid OMNIbus install package.**

Explanation: The archive does not appear to be a valid OMNIbus install package.

Operator response: Ensure that the archive file is a valid OMNIbus install package.

KQZ6000W **Columns for the query have not been found. Continuing will create an attribute group containing no attributes.**

Explanation: An SQL query that is manually defined must be tested for attributes to be automatically created.

Operator response: Select "OK" to continue without testing the query. Select "Cancel" to return to the browser to test the query to detect attribute data.

KQZ6001E **A problem occurred running this SQL statement. Check the query for mistakes. The database server returned the following problem information:** *exception_message***.**

Explanation: The database server returned an error when it attempted to run the SQL statement.

Operator response: Modify the statement and attempt the test again. The message returned by the database server may help identify the problem.

KQZ6100W **Unable to discover the namespaces on the system. If you know the namespace, you can add it to the list by clicking the Add button next to the namespace field.**

Explanation: The Agent Bulider was unable to discover the namespaces on the system.

Operator response: If you know the namespace, add it to the list by clicking the Add button next to the namespace field.

KQZ6101E **The namespace is already defined for this host.**

Explanation: The namespace is already defined for the host.

Operator response: If this is the namespace you desire, select it from the list. Otherwise, specify a new namespace.

KQZ6102E **Namespace,** *namespace***, does not exist on the CIM server.**

Explanation: The namespace is not defined on the server.

Operator response: Specify a namespace that exists on the server.

KQZ6200E **An error code with value** *error_code* **was defined previously.**

Explanation: This is an informational message only.

KQZ6201W The Java source folder *folder_name* does not exist, and Java source code for the agent cannot be written.

Explanation: If an agent contains Java API attribute groups, the Agent Builder attempts to write out Java source code that assists the agent developer in writing a Java custom provider application. If the Java source folder does not exist, the code cannot be written.

Operator response: Create the missing Java source folder:

1. In the Project explorer tree, right-click on the project node for the agent. A context menu will be displayed.
2. From the context menu, select Properties. A properties dialog will be displayed.
3. On the left of the Properties dialog, select Java Build Path from the tree. A Java Build Path panel will be displayed on the right side of the Properties dialog.
4. In the Java Build Path panel, select the Source tab.
5. Click on the Add Folder... button. A Source Folder Selection dialog will be displayed.
6. In the Source Folder Selection dialog, click on the Create New Folder... button. A New Source Folder dialog will be displayed.
7. In the New Source Folder dialog, enter the missing folder name after the "Folder name:" prompt.
8. Press Finish. This will dismiss the New Source Folder dialog.
9. Press OK. This will dismiss the Source Folder Selection dialog.
10. Press OK. This will dismiss the Properties dialog, creates the new folder, and marks it as a Java source folder.

The next time you save the agent, the Java source files will be created in this folder.

KQZ6202W Sample Java code was not created because subnode *subnode_name* does not contain any configuration properties.

Explanation: The example Java code for a sampled Java API attribute group in a subnode depends on the subnode having at least one configuration property. Since the subnode does not have any configuration properties, Java sample code was not created.

Operator response: Add a configuration property to the subnode and save the agent.

KQZ6203E The selection containing custom provider *client_name*. cannot be removed because attribute group *attribute_group* references the provider.

Explanation: The custom provider you are trying to delete is referenced by one or more attribute groups and cannot be removed from the agent.

Operator response: Delete the attribute group or groups that reference this custom provider if you do not wish to have them in your agent any longer.

KQZ6204E The selection containing the Java API custom provider application cannot be removed because attribute group *attribute_group* references it.

Explanation: The Java API custom provider application cannot be removed because is referenced by one or more attribute groups.

Operator response: Delete the attribute group or groups that reference the Java API application if you do not wish to have them in your agent any longer.

KQZ6205E The selection containing the socket custom data provider cannot be removed because attribute group *attribute_group* references it.

Explanation: The socket custom data provider cannot be removed because is referenced by one or more attribute groups.

Operator response: Delete the attribute group or groups that reference the socket custom data provider if you do not wish to have them in your agent any longer.

KQZ6600W The SNMP properties cannot be tested from this dialog when field "*field_name*" contains value "*value*".

Explanation: The Java SNMP implementation used by the Agent Builder cannot use the protocol. The agent itself can use the protocol.

Operator response: Test your agent as you normally would. Connection problems caused by an incorrect property value can be detected when testing the agent but not when editing the properties.

KQZ7001E The file name must be itm_jtk_agent.xml.

Explanation: The JMX Agent Toolkit can only read agents whose file name is itm_jtk_agent.xml

Operator response: If the file contains a valid agent, rename the file to itm_jtk_agent.xml.

KQZ7002E Unable to access the file *file_name*

Explanation: The Resource Model Builder is unable to read the contents of the resource itm_jtk_agent.xml file.

Operator response: Ensure that you have permission to access the file. It's possible you may have deleted the

file from outside of Eclipse and failed to refresh the project.

KQZ7003I **Generating the IBM Tivoli Monitoring Agent**

Explanation: This is an informational message only.

KQZ7004E **A Product Service Name must start with an alphabetic character, and can contain only alphanumeric and underscore characters.**

Explanation: This is an informational message only.

KQZ7005E **The value must be a positive integer less than or equal to 255.**

Explanation: This is an informational message only.

KQZ7006E **An error occurred while saving the file** *file_name***.**

Explanation: An error occurred during the save operation for the specified file.

Operator response: Check the details section of this dialog for more information.

KQZ7010W **An MBean was not selected. Continuing will close the browser and attributes will need to be created manually.**

Explanation: An MBean must be selected to determine the attributes that will be created.

Operator response: Select "OK" to close the JMX browser and define one or more attributes manually in the wizard. Select "Cancel" to return to the browser to select an MBean containing attributes.

KQZ8002E **The Agent Builder encountered an error while generating file** *name***.**

Explanation: The builder encountered an unexpected error.

Operator response: Ensure you have the correct permissions to the output directory. If the error persists, try restarting the Agent Builder.

KQZ8002F **The Agent Builder encountered an error while copying imported files into the agent installation image.**

Explanation: The builder encountered an unexpected error.

Operator response: Ensure you have the correct permissions to the output directory. If the error persists, try restarting the Agent Builder.

KQZ8003E **An error occurred while generating the agent.** *internal_error_message*

Explanation: The IBM Tivoli Monitoring Agent Builder encountered errors while attempting to generate the agent.

Operator response: Check the details view of the error dialog for exception information, or examine the Eclipse and IBM Tivoli Monitoring Agent Toolkit trace files to see the cause of the error.

KQZ8004E **No version specified.**

Explanation: This is an informational message only.

KQZ8005E **Version** *version* **is not valid.**

Explanation: This is an informational message only.

KQZ8006E **Patch level** *patch* **is not valid.**

Explanation: This is an informational message only.

KQZ8007E **Version** *current_version* **is less than version** *prev_version* **committed on** *date***.**

Explanation: This is an informational message only.

KQZ8008E **No affinity specified.**

Explanation: This is an informational message only.

KQZ8009E **Affinity "***affinity_tag***" is not valid.**

Explanation: This is an informational message only.

KQZ8010E **Affinity tag "***affinity_tag***" is not valid.**

Explanation: This is an informational message only.

KQZ8011E **No affinity tag specified.**

Explanation: This is an informational message only.

KQZ8012E **No agent display name specified.**

Explanation: This is an informational message only.

KQZ8013E **No product code specified.**

Explanation: This is an informational message only.

KQZ8014E **Runtime configuration section name not specified.**

Explanation: This is an informational message only.

KQZ8015E **Property name not specified**

Explanation: This is an informational message only.

KQZ8016E **Value name not specified**

Explanation: This is an informational message only.

KQZ8107E **Configuration element name** *name* **is not valid.**

Explanation: This is an informational message only.

KQZ8108E **No label specified for the configuration element.**

Explanation: This is an informational message only.

KQZ8109E **No message identifier specified**

Explanation: This is an informational message only.

KQZ8110E **No description specified for the configuration element**

Explanation: This is an informational message only.

KQZ8111E **Message identifier** *id* **was previously defined.**

Explanation: This is an informational message only.

KQZ8112E **Message identifier** *id* **is not valid.**

Explanation: This is an informational message only.

KQZ8113E **No properties defined for configuration section** *source*

Explanation: This is an informational message only.

KQZ8114E **Property type not specified**

Explanation: This is an informational message only.

KQZ8115E **Property type** *pc* **is not valid**

Explanation: This is an informational message only.

KQZ8116E **The variable name "***property_variable_name***" of property "***property_label***" in section "***seciton_name***" should contain the string "PASSWORD" to ensure that it is never stored in clear text form.**

Explanation: This is an informational message only.

KQZ8117E **The variable name "***property_variable_name***" of property "***property_label***" in section "***seciton_name***" should not contain the string "PASSWORD" to ensure that it does not get confused with a password property.**

Explanation: This is an informational message only.

KQZ8118E **The default value "***value***" for numeric property** *prop_name* **is not valid.**

Explanation: This is an informational message only.

KQZ8119E **No configuration values specified for property** *propname*

Explanation: This is an informational message only.

KQZ8120E **The agent is defined as being multi-instance, but there are no configuration elements defined.**

Explanation: Multi-instance agents require configuration; otherwise, all instances would monitor the same resources.

Operator response: Either add configuration elements to the agent or turn off multi-instance support.

KQZ8121E **Navigator group name "***navgroup_name***" is not valid.**

Explanation: The specified name contains invalid characters.

Operator response: A navigator group must start with a letter and can contain only letters, numbers or underscores.

KQZ8122E **A navigator group with identifier** *id* **was defined previously.**

Explanation: Navigator group names must be unique.

Operator response: Enter a new name for the navigator group.

KQZ8123E **Navigator group name** *id* **is reserved for internal use.**

Explanation: The Agent Builder automatically creates navigator groups in certain situations. The name specified duplicates one of these navigator groups.

Operator response: See the User Guide for the list of reserved navigator group names.

KQZ8124E **A return code with value *rc* was defined previously.**

Explanation: This is an informational message only.

KQZ8125E **Return code "*rc*" is not numeric.**

Explanation: This is an informational message only.

KQZ8126E **Attribute group name *attribute_group_name* is not valid.**

Explanation: This is an informational message only.

KQZ8127E **An attribute group with identifier *id* was defined previously.**

Explanation: This is an informational message only.

KQZ8128E **Attribute group name *id* is reserved for internal use.**

Explanation: This is an informational message only.

KQZ8129E **Attribute group *attribute_group_name* does not have help text.**

Explanation: This is an informational message only.

KQZ8130E **The help text for *object_name* exceeds the maximum length of *max_length*.**

Explanation: This is an informational message only.

KQZ8131E **No operating systems selected for attribute group "*attribute_group_name*".**

Explanation: This is an informational message only.

KQZ8132E **Attribute group "*referencing_attribute_group*" references an unknown attribute group "*referenced_attribute_group*".**

Explanation: This is an informational message only.

KQZ8133E **Joined attribute group "*joined_attribute_group*" attempts to join attribute group "*referenced_attribute_group*" with itself.**

Explanation: This is an informational message only.

KQZ8134E **Attribute group *referencing_attribute_group* references the Availability attribute group; however, no availability filters exist.**

Explanation: This is an informational message only.

KQZ8135E **No attribute to join specified for attribute group *attribute_group*.**

Explanation: This is an informational message only.

KQZ8136E **Join attribute "*attribute*" defined in joined attribute group "*joined_attribute_group*" was not found in attribute group *data_source*.**

Explanation: This is an informational message only.

KQZ8138E **Key attribute "*attribute*" defined in joined attribute group "*joined_attribute_group*" was not found in either base attribute group.**

Explanation: This is an informational message only.

KQZ8139E **Attribute name *attribute_name* is not valid.**

Explanation: This is an informational message only.

KQZ8140E **Attribute name *attribute_name* exceeds the limit of *limit* characters.**

Explanation: This is an informational message only.

KQZ8141E **Attribute name *id* is reserved for internal use.**

Explanation: This is an informational message only.

KQZ8142E **Attribute *attribute_name* does not have help text.**

Explanation: This is an informational message only.

KQZ8143E **Attribute *attribute_name* is *attribute_bytes* bytes long which exceeds the maximum size of *maximum_size* bytes.**

Explanation: This is an informational message only.

KQZ8144E **The formula contains a syntax error.**

Explanation: This is an informational message only.

KQZ8145E **Attribute group *id* contains *attribute_count* attributes, more than the limit of *maximum_attribute_count*.**

Explanation: There are too many attributes in the attribute group set to display on the TEP.

Operator response: Remove some attributes, mark some attributes as not being displayed on the TEP, or split the group into two smaller groups.

KQZ8146E **Attribute group *id* contains zero attributes.**

Explanation: This is an informational message only.

KQZ8147E **Attributes in attribute group *attribute_group_name* total *attribute_group_bytes* bytes which exceeds the maximum of *maximum_size* bytes.**

Explanation: This is an informational message only.

KQZ8148E **Attribute group *source* can produce more than one data row but does not contain a key attribute.**

Explanation: This is an informational message only.

KQZ8149E **Perfmon attribute group *source* can produce more than one data row but does not return the Perfmon instance name or contain an other key attribute.**

Explanation: This is an informational message only.

KQZ8150E **Range value "*range_endpoint_value*" is not a valid numerical value.**

Explanation: This is an informational message only.

KQZ8151E **Range value "*range_endpoint_value*" is not a number from *minimum_value* to *maximum_value*, the minimum and maximum for a *integer_size*-bit number.**

Explanation: This is an informational message only.

KQZ8152E **Range value "*range_endpoint_value*" is not a number from *minimum_value* to *maximum_value*, the minimum and maximum for a *integer_size*-bit number with a decimal adjustment of *decimal_adjustment*.**

Explanation: This is an informational message only.

KQZ8153E **Minimum range value *minimum_range_endpoint_value* is not less than maximum range value *maximum_range_endpoint_value*.**

Explanation: This is an informational message only.

KQZ8154E **Decimal adjustment (scale) must be a number from 0 to *maximum_scale* for *integer_size*-bit numbers.**

Explanation: This is an informational message only.

KQZ8155E **You can only join an attribute group that produces events to an attribute group that produces a single data row.**

Explanation: This is an informational message only.

KQZ8156E **Attribute *attribute_name* is defined as a key, however it is defined as hidden.**

Explanation: This is an informational message only.

KQZ8157E **The metric name is missing in attribute *attribute_name*.**

Explanation: This is an informational message only.

KQZ8158E **No type is specified for attribute *attribute_name*.**

Explanation: This is an informational message only.

KQZ8159E **Attribute *attribute_name* does not have a valid maximum size.**

Explanation: This is an informational message only.

KQZ8160E **The object identifier is missing in attribute *attribute_name*.**

Explanation: This is an informational message only.

KQZ8161E **"*token*" is not expected in column *column_number*.**

Explanation: This is an informational message only.

KQZ8162E **More tokens expected at the end of the formula.**

Explanation: This is an informational message only.

KQZ8163E **Closing parenthesis for function call "*function_name*" not found.**

Explanation: This is an informational message only.

KQZ8164E **Two unary operators, "*unary_operator*" and "*unary_operator*" found without an intervening expression.**

Explanation: This is an informational message only.

KQZ8165E **"*bad_function_name*" is not a valid function name.**

Explanation: This is an informational message only.

KQZ8166E **"*token*" found where identifier or literal was expected.**

Explanation: This is an informational message only.

KQZ8167E **Missing argument in function "*function_name*".**

Explanation: This is an informational message only.

KQZ8168E **"*bad_operator*" is not a valid operator.**

Explanation: This is an informational message only.

KQZ8169E **Lexical error at column "*column_number*". Encountered: "*bad_character*" after "*reference_character*".**

Explanation: This is an informational message only.

KQZ8170E **No function name has been specified.**

Explanation: This is an informational message only.

KQZ8171E **A function name cannot start with a number.**

Explanation: This is an informational message only.

KQZ8172E **A function name cannot contain a space character.**

Explanation: This is an informational message only.

KQZ8173E **A function name cannot contain character "*bad_character*".**

Explanation: This is an informational message only.

KQZ8174W **"*function_name*" is not a recognized derived formula function.**

Explanation: This is an informational message only.

KQZ8175E **No attribute name has been specified.**

Explanation: This is an informational message only.

KQZ8176E **An attribute name cannot start with a number.**

Explanation: This is an informational message only.

KQZ8177E **An attribute name cannot contain a space character.**

Explanation: This is an informational message only.

KQZ8178E **An attribute name cannot contain character "*bad_character*".**

Explanation: This is an informational message only.

KQZ8179W **Attribute "*attribute_name*" is not found in this attribute group.**

Explanation: This is an informational message only.

KQZ8180W **Function "*function_name*" has *argument_count* arguments, but *parameter_count* arguments are expected.**

Explanation: This is an informational message only.

KQZ8181W **Function "*function_name*" argument *argument_number* is *actual_argument_type*, but *expected_parameter_type* is expected.**

Explanation: This is an informational message only.

KQZ8182E **An object identifier is specified in attribute *attribute_name* which also has an index reference.**

Explanation: This is an informational message only.

KQZ8184E **Operator *operator* must have 2 operands.**

Explanation: This is an informational message only.

KQZ8185E **Operator *operator* must have 1 or 2 operands.**

Explanation: This is an informational message only.

KQZ8186W **The left operand of the *operator* operator is *left_operand_type* and the right is *right_operand_type*. Both types should be the same.**

Explanation: This is an informational message only.

KQZ8187W **The left operand of the *operator* operator is *actual_type_with_indefinite_article* but *actual_type_with_indefinite_article* operand is expected.**

Explanation: This is an informational message only.

KQZ8188W **The right operand of the *operator* operator is *actual_type_with_indefinite_article* but *actual_type_with_indefinite_article* operand is expected.**

Explanation: This is an informational message only.

KQZ8189W **The derived formula in attribute** *attribute_name* **evaluates to** *actual_type* **but should evaluate to** *expected_type* **to match the attribute type.**

Explanation: This is an informational message only.

KQZ8190E **There are no process or service availability filters defined with the TADMM sensor "Use these criteria" box checked.**

Explanation: One process or service availability filter must have TADDM sensor criteria defined, and the check box labeled "Use these criteria to identify where the sensor should run" checked.

Operator response: Define a process availability filter, select the "TADDM sensor" tab, check the "Use these criteria to identify where the sensor should run" checkbox, define criteria to identify the process, save the file and regenerate the sensor.

KQZ8191E **There are more than one process or service availability filters defined with the TADMM sensor "Use these criteria" box checked.**

Explanation: Only one process or service availability filter can have the check box labeled "Use these criteria to identify where the sensor should run" checked.

Operator response: Examine all the process and service availability filters, select the "TADDM sensor" tab, and make sure only one of them has the "Use these criteria to identify where the sensor should run" checkbox checked. Then save the file and regenerate the sensor.

KQZ8192E **Cannot write to file** *maximum_count***.**

Explanation: An unsuccessful attempt was made to write the file to the local file system.

Operator response: Change the path where the file is being written, or check that the specified path is not read-only or the disk partition is not full.

KQZ8195E **Java resource** *resource_name* **cannot be found or read.**

Explanation: This is an informational message only.

KQZ8196E **Closing parenthesis after "***expression_fragment***" not found.**

Explanation: This is an informational message only.

KQZ8197E **An agent with affinity "***affinity***" has already been defined.**

Explanation: Affinities must be unique

Operator response: Enter a unique affinity.

KQZ8198E **Index,** *index***, is not defined.**

Explanation: The specified SNMP index is not defined in the set of MIBs loaded by the parser.

Operator response: Find the MIB that defines the index and import it.

KQZ8199E **Could not determine an OID for value,** *value***.**

Explanation: The parser could not determine the OID for the specified value.

Operator response: Find the MIB that defines the value and import it.

KQZ8200W **The size of attribute,** *name***, was changed from** *original_size* **to** *new_size***.**

Explanation: The size reported by the MIB file was larger than expected.

Operator response: If the size needs to be larger, modify the attribute to increase the size.

KQZ8201E **"***supposed_directory_name***" is not a valid directory name.**

Explanation: The value is not a valid directory.

Operator response: Enter or select a valid directory name.

KQZ8202E **File "***file_name***" does not exist in directory "***directory_name***"**

Explanation: The file does not exist in the directory.

Operator response: Enter a valid directory name that contains the file.

KQZ8203E *schema_file_name* **version** *schema_version* **and** *naming_rules_file_name* **version** *naming_rules_version* **from directory** *directory_name* **do not match.**

Explanation: The two files must be from the same version of the Common Data Model.

Operator response: Get files from the same version of the Common Data Model (at least CDM.xsd, NamingRules.xml, and oneschema.xml) and place them in the directory.

KQZ8204E **The join attributes selected for *attribute_group* have different types.**

Explanation: The two join attributes for a joined attribute group must both be numeric or both be a string.

Operator response: Select two join attributes of the same type, either both string or both numeric.

KQZ8205E **The event threshold attribute can only be set if attribute group, *attribute_group*, is pure event.**

Explanation: The attribute group is not a pure event group, so the value set for the event threshold should not be set.

Operator response: Remove the event threshold attribute from the attribute group definition.

KQZ8206E **The event summary interval attribute can only be set if attribute group, *attribute_group*, is pure event.**

Explanation: The attribute group is not a pure event group, so the value set for the event summary interval should not be set.

Operator response: Remove the event summary interval attribute from the attribute group definition.

KQZ8207E **The event filtering threshold value for attribute group *attribute_group* is not valid. It should be one of "send_none", "send_first", "send_all", a positive integer or a configuration variable.**

Explanation: The attribute does not contain a valid value.

Operator response: Set a valid value for the attribute.

KQZ8208E **The event summary interval for attribute group *attribute_group* is not valid. It should be a positive integer or a configuration variable.**

Explanation: The attribute does not contain a valid value.

Operator response: Set a valid value for the attribute.

KQZ8209W **An event threshold exists for attribute group *attribute_group*; however, an event summary interval is not set.**

Explanation: The event threshold should only be set if an event summary interval is set.

Operator response: Either remove the event threshold or specify an event summary interval.

KQZ8210E **Event filtering and summarization is enabled for attribute group *attribute_group*; however, no key attribute exists.**

Explanation: The attribute group does not contain a key attribute.

Operator response: Specify a key attribute.

KQZ8211E **Operating system *operating_system* is selected in more than one command for "*command_definition_name*".**

Explanation: Two commands for a single command definition are configured to run on the same operating system. When alternate commands are defined in a single script definition, the commands must to run on different operating systems. For any given operating system, only one command can be run.

Operator response: Remove the operating system selection on one of the command definitions.

KQZ8212E **No operating systems selected for command "*command*".**

Explanation: This is an informational message only.

KQZ8213E **Filtered attribute group *attribute_group_name* has no formula for the filter.**

Explanation: This is an informational message only.

KQZ8214E **Attribute group "*referencing_attribute_group*" references attribute group "*referenced_attribute_group*" which belongs to a different subnode.**

Explanation: This is an informational message only.

KQZ8215E **Attribute "*attribute*" has source type "*source_class*" in attribute group "*attribute_group*" which has settings type "*settings_class*". This attribute is expected to be in a group with settings type "*settings_class*".**

Explanation: This is an informational message only.

KQZ8216E **Filtered attribute group "*attribute_group*" is missing a reference to a source attribute group.**

Explanation: This is an informational message only.

KQZ8217E **Filtered attribute group "*attribute_group*" has attribute "*attribute*" with a missing referenced attribute name.**

Explanation: This is an informational message only.

KQZ8218E **Attribute "*referencing_attribute*" in attribute group "*referencing_attribute_group*" references an attribute named "*source_attribute_name*" that does not exist in attribute group "*source_attribute_group*".**

Explanation: This is an informational message only.

KQZ8219E **Attribute "*referencing_attribute*" of type "*referencing_attribute_type*" in attribute group "*referencing_attribute_group*" references attribute "*source_attribute*" of type "*source_attribute_type*" in attribute group "*source_attribute_group*". The attribute types should be the same.**

Explanation: This is an informational message only.

KQZ8220W **Function *function_name* should only be called for event attribute groups, but is called for sampled attribute group *attribute_group_name*.**

Explanation: The functions isSummaryEvent, cumulativeSum, and occurrenceCount are applicable only to pure event attribute groups, but one of them was called for a sampled attribute group.

Operator response: Remove the function call from the formula.

KQZ8221E **The formula is missing.**

Explanation: This is an informational message only.

KQZ8222W **Event summarization is enabled for attribute group *attribute_group*; however, no event summarization attributes exist.**

Explanation: Although the attributes are not necessary, they are useful when performing event summarization.

Operator response: Add the event summarization attributes by right clicking on the attribute group.

KQZ8223W **Event summarization is enabled for attribute group *attribute_group*; however, the event summarization occurrence count attribute does not exist.**

Explanation: Although the attribute is not necessary, it is useful when performing event summarization.

Operator response: Add the event summarization attributes by right clicking on the attribute group.

KQZ8224W **Event summarization is enabled for attribute group *attribute_group*; however, the summary event flag attribute does not exist.**

Explanation: Although the attribute is not necessary, it is useful when performing event summarization.

Operator response: Add the event summary attributes by right clicking on the attribute group.

KQZ8225W **Event summarization is enabled for attribute group *attribute_group*; however, the event threshold attribute does not exist.**

Explanation: Although the attribute is not necessary, it is useful when performing event summarization.

Operator response: Add the event summarization attributes by right clicking on the attribute group.

KQZ8226W **Event summarization is enabled for attribute group *attribute_group*; however, the summary interval attribute does not exist.**

Explanation: Although the attribute is not necessary, it is useful when performing event summarization.

Operator response: Add the event summarization attributes by right clicking on the attribute group.

KQZ8227W **Attribute *attribute_name* must not be both a key and a rate or delta.**

Explanation: This is an informational message only.

KQZ8228E **The subnode name cannot be empty**

Explanation: No subnode name was entered.

Operator response: Enter a name for the subnode.

KQZ8229E **The subnode description cannot be empty**

Explanation: No subnode description was entered.

Operator response: Enter a description for the subnode.

KQZ8230E **The errpt command cannot be empty**

Explanation: The errpt command line was not specified

Operator response: Enter the errpt command that should be run.

KQZ8231E The command specified does not invoke the errpt command or is not a configuration variable.

Explanation: The command specified must run the errpt command or it must be a configuration variable.

Operator response: Specify the errpt utility or a configuration variable.

KQZ8232E The command specified does not provide the -c option.

Explanation: The -c option is required for the agent to properly gather data from errpt.

Operator response: Specify the -c option.

KQZ8233E The configuration variable must start with ${ and end with }.

Explanation: Ensure that the configuration variable is specified properly.

Operator response: Specify a correct configuration variable.

KQZ8234W Event filtering and summarization is not enabled for attribute group, *attribute_group*; however, event summarization attributes exist.

Explanation: The derived functions being use only return valid data when event filtering and summarization is turned on.

Operator response: Remove the event summarization attributes from the group or enable filtering and summarization for the group.

KQZ8235E *schema_file_name* has a format error and cannot be parsed.

Explanation: There was an error parsing a Common Data Model file.

Operator response: Download the files from Common Data Model again (at least CDM.xsd, NamingRules.xml, and oneschema.xml) and place them in the directory specified in the Common Data Model preferences.

KQZ8236E The sensor has two or more layouts with the same display name "*layout_display_name*".

Explanation: Each TADDM GUI layout must have a unique display name.

Operator response: In the TADDM GUI section, click one of the layouts with the duplicate display name, click the Edit button, and change the display name.

KQZ8237E The sensor has two or more layouts which reference the same Common Data Model class "*layout_class_name*".

Explanation: Each TADDM GUI layout must reference a unique Common Data Model class.

Operator response: In the TADDM GUI section, click one of the layouts with the duplicate class name, click the Edit button, and browse to a different class.

KQZ8238E TADDM GUI layout does not have a display name.

Explanation: A TADDM GUI layout definition does not have a display name.

Operator response: In the TADDM GUI section, select the layout without a name, click the Edit button, and enter a display name.

KQZ8239E TADDM GUI layout "*layout_name*" does not define a CDM class.

Explanation: A TADDM GUI layout definition does not have a Common Data Model class associated with it.

Operator response: In the TADDM GUI section, select the layout, click the Edit button, select Browse, and choose a CDM class.

KQZ8240E TADDM GUI layout "*layout_name*" does not contain any tab definitions.

Explanation: Each TADDM GUI layout must define at least one tab.

Operator response: In the TADDM GUI section, select the layout and click the Add button to define a tab in the layout.

KQZ8241E TADDM GUI layout "*layout_name*" has two or more tabs with the same display name "*tab_display_name*".

Explanation: Each TADDM GUI tab in a layout must have a unique display name.

Operator response: In the TADDM GUI section, open the layout, click one of the tabs with the duplicate display name, click the Edit button, and change the display name.

KQZ8242E TADDM GUI tab in layout "*layout_name*" does not have a display name.

Explanation: A TADDM GUI tab definition does not have a display name.

Operator response: In the TADDM GUI section, select the tab without a name, click the Edit button, and enter a display name.

KQZ8243E **TADDM GUI tab "*tab_name*" in layout "*layout_name*" does not contain any table definitions.**

Explanation: Each TADDM GUI tab in a layout must define at least one table.

Operator response: In the TADDM GUI section, select the tab and click the Add button to define a table in the tab.

KQZ8244E **TADDM GUI tab "*tab_name*" in layout "*layout_name*" has two or more tables with the same display name "*tab_display_name*".**

Explanation: Each TADDM GUI table in a tab must have a unique display name.

Operator response: In the TADDM GUI section, open the layout and tab, click one of the tables with the duplicate display name, click the Edit button, and change the display name.

KQZ8245E **TADDM GUI table in tab "*tab_name*" in layout "*layout_name*" does not have a display name.**

Explanation: A TADDM GUI table definition does not have a display name.

Operator response: In the TADDM GUI section, select the table without a name, click the Edit button, and enter a display name.

KQZ8246E **TADDM GUI table in tab "*tab_name*" in layout "*layout_name*" does not define a CDM class.**

Explanation: A TADDM GUI table definition does not have a Common Data Model class associated with it.

Operator response: In the TADDM GUI section, select the table, click the Edit button, select Browse, and choose a CDM class.

KQZ8247E **TADDM GUI table "*table_name*" in tab "*tab_name*" in layout "*layout_name*" does not contain any field definitions.**

Explanation: Each TADDM GUI table must define at least one field.

Operator response: In the TADDM GUI section, select the table and click the Add button to define a field in the table.

KQZ8248E **TADDM GUI field in table "*table_name*" in in tab "*tab_name*" in layout "*layout_name*" has no attribute selected.**

Explanation: A TADDM GUI field must select an attribute from the parent table's class or parent field's class.

Operator response: In the TADDM GUI section, select the field, click the Edit button, and choose an attribute name.

KQZ8249E **TADDM GUI field "*attribute_name*" in table "*table_name*" in tab "*tab_name*" in layout "*layout_name*" does not contain any field definitions.**

Explanation: A TADDM GUI field which identifies another Common Data Model object must define at least one field from that object.

Operator response: In the TADDM GUI section, select the field and click the Add button to identify a field from the associated attribute.

KQZ8250E **TADDM GUI field in table "*tab_name*" in tab "*tab_name*" in layout "*layout_name*" does not have a display name.**

Explanation: A TADDM GUI field which identifies a primitive attribute value to display in a table must define a display name to be displayed along with the attribute.

Operator response: In the TADDM GUI section, select the table without a name, click the Edit button, and enter a display name.

KQZ8251E **Class "*class_name*" is not a valid Common Data Model class.**

Explanation: The class specified could not found in the Common Data Model or does not inherit from core.ManagedElement.

Operator response: Select a valid class from the Common Data Model browser, or update the builder with a more recent version of the Common Data Model schema files (Window - Preferences - IBM Tivoli Monitoring - Common Data Model).

KQZ8252E **CDM schema directory "*directory_name*" does not exist.**

Explanation: A directory containing the CDM schema was set with the preferences and that directory no longer exists.

Operator response: Download the files from Common Data Model (at least CDM.xsd, NamingRules.xml, and oneschema.xml) and identify the location of those files using Window - Preferences - IBM Tivoli Monitoring - Common Data Model.

KQZ8253E **CDM schema file "*file_name*" does not exist in directory "*directory_name*".**

Explanation: A directory containing the CDM schema was set with the preferences and one or more Common Data Model files no longer reside in that directory (or cannot be read from that directory).

Operator response: Download the files from Common Data Model (at least CDM.xsd, NamingRules.xml, and oneschema.xml) into the directory, or identify a new directory location of those files using Window - Preferences - IBM Tivoli Monitoring - Common Data Model.

KQZ8254E **Common Data Model attribute "*attribute_name*" has type "*attribute_type*", but embedded class information shows that it is type "*attribute_type*".**

Explanation: This is an informational message only.

KQZ8255E **Common Data Model attribute "*attribute_name*" has no topology link finder class defined. Attribute parent is "*attribute_path*".**

Explanation: If the Common Data Model class in which an attribute resides has a topology node, and the attribute itself is a class which has a topology node, then a link finder custom class and method must be defined. The custom link finder class is missing.

Operator response: Select the attribute node on the Common Data Model tree and enter a custom link finder class name.

KQZ8256E **Common Data Model attribute "*attribute_name*" has no topology link finder method defined. Attribute parent is "*attribute_path*".**

Explanation: If the Common Data Model class in which an attribute resides has a topology node, and the attribute itself is a class which has a topology node, then a link finder custom class and method must be defined. The custom link finder method is missing.

Operator response: Select the attribute node on the Common Data Model tree and enter a custom link finder method name.

KQZ8257E **Common Data Model class "*class_name*" has no topology node finder class defined.**

Explanation: If the Common Data Model class has a topology node and specifies that it is to use a custom class and method to locate the appropriate nodes, then a node finder custom class and method must be defined. The custom node finder class is missing.

Operator response: Select the node on the Common Data Model tree and enter a custom node finder class name.

KQZ8258E **Common Data Model class "*class_name*" has no topology node finder class defined. The class is defined in attribute "*attribute_path*".**

Explanation: If the Common Data Model class has a topology node and specifies that it is to use a custom class and method to locate the appropriate nodes, then a node finder custom class and method must be defined. The custom node finder class is missing.

Operator response: Select the node on the Common Data Model tree and enter a custom node finder class name.

KQZ8259E **Common Data Model class "*class_name*" has no topology node finder class defined.**

Explanation: If the Common Data Model class has a topology node and specifies that it is to use a custom class and method to locate the appropriate nodes, then a node finder custom class and method must be defined. The custom node finder method is missing.

Operator response: Select the node on the Common Data Model tree and enter a custom node finder method name.

KQZ8260E **Common Data Model class "*class_name*" has no topology node finder class defined. The class is defined in attribute "*attribute_path*".**

Explanation: If the Common Data Model class has a topology node and specifies that it is to use a custom class and method to locate the appropriate nodes, then a node finder custom class and method must be defined. The custom node finder method is missing.

Operator response: Select the node on the Common Data Model tree and enter a custom node finder method name.

KQZ8261E **Could not copy file from "*source_file_path*" to "*target_file_path*".**

Explanation: This is an informational message only.

KQZ8262E **Could not copy directory from "*source_directory*" to "*target_directory*".**

Explanation: This is an informational message only.

KQZ8263E **Could not add directory from "*source_directory_or_file*" to zip file "*zip_file*".**

Explanation: This is an informational message only.

KQZ8264E **The HTTP function specified for attribute, *attribute_name*, is not valid.**

Explanation: The function specified is incorrect.

Operator response: Select a valid type for the attribute.

KQZ8265E **The HTTP function specified for attribute, *attribute_name*, requires an argument; however, none was specified.**

Explanation: The function selected requires an argument.

Operator response: Specify an argument or choose a different attribute type.

KQZ8266E **The XPath for attribute group, *attribute_name*, contains a syntax error.**

Explanation: The XPath contains a syntax error.

Operator response: Correct the XPath syntax.

KQZ8267E **The XPath for attribute, *attribute_name*, contains a syntax error.**

Explanation: The XPath contains a syntax error.

Operator response: Correct the XPath syntax.

KQZ8268E **An attribute cannot have an empty name.**

Explanation: There is no name specified for the attribute.

Operator response: Specify a unique name for the attribute.

KQZ8269E **The Agent Builder encountered an error while generating the Cognos data model.**

Explanation: The builder encountered an unexpected error.

Operator response: Ensure you have the correct permissions to the output directory. If the error persists, try restarting the Agent Builder.

KQZ8270E **No URL specified for attribute group *attribute_group*.**

Explanation: The attribute group must contain a URL.

Operator response: Specify the URL for the attribute group.

KQZ8271E **No HTTP request type specified for attribute group *attribute_group*.**

Explanation: The attribute group must contain a request type.

Operator response: Specify the request type for the attribute group.

KQZ8272E **The request type specified for attribute group *attribute_group* requires post data to be specified.**

Explanation: The attribute group must contain post data.

Operator response: Specify post data for the attribute group or change the request type to GET.

KQZ8273W **The request type specified for attribute group *attribute_group* is GET; however, post data is specified.**

Explanation: The post data will not be used.

Operator response: Remove the post data for the attribute group or change the request type either POST or PUT.

KQZ8274E **The post data specified for attribute group *attribute_group* contains a syntax error.**

Explanation: The post data is not well formated XML.

Operator response: Correct the post data for the attribute group or change the request type to GET.

KQZ8280E **Common Data Model file *schema_file_name* could not be found at location *location*.**

Explanation: The Common Data Model file could not be found.

Operator response: Download the files from Common Data Model again (at least CDM.xsd, NamingRules.xml, and oneschema.xml) and place them in the directory specified in the Common Data Model preferences.

KQZ8281E **Common Data Model zip file** *schema_file_name* **at location** *location* **could not be opened.**

Explanation: The Common Data Model zip schema file could not be opened or read.

Operator response: Download the files from Common Data Model again (at least CDM.xsd, NamingRules.xml, and oneschema.xml) and place them in the directory specified in the Common Data Model preferences.

KQZ8282I **Attributes from more than one IBM Tivoli Monitoring attribute group are being queried.**

Explanation: A simple mapping section copies attribute values from one attribute group to attributes in one Common Data Model class. This section gets attributes from more than one IBM Tivoli Monitoring attribute group and thus cannot be shown under the Attributes tab.

Operator response: Click on the Queries tab and select a query from the Query field to see which IBM Tivoli Monitoring attributes' values are being used.

KQZ8283I **No IBM Tivoli Monitoring attribute groups are being queried.**

Explanation: A simple mapping section copies attribute values from one attribute group to attributes in one Common Data Model class. This section does not query any IBM Tivoli Monitoring attribute groups and thus cannot be shown under the Attributes tab.

Operator response: If appropriate for this mapping section, click on the Queries tab and click on the Add button select a query from the Query field at the right of the Query field to add an IBM Tivoli Monitoring attribute group to query.

KQZ8284I **Attributes from more than one Common Data Model class are being set.**

Explanation: A simple mapping section copies attribute values from one attribute group to attributes in one Common Data Model class. This section copies attributes to more than one Common Data Model class and thus cannot be shown under the Attributes tab.

Operator response: Click on the CDM Attributes tab and select a Common Data Model class from the Class field to see how that class' variables will be set.

KQZ8285I **No Common Data Model class mappings are defined.**

Explanation: A simple mapping section copies attribute values from one attribute group to attributes in one Common Data Model class. This section does not copy attributes to any Common Data Model class and thus cannot be shown under the Attributes tab.

Operator response: If appropriate for this mapping section, click on the CDM Attributes tab and click on the Add button at the right of the Class field to add a Common Data Model class whose attributes will be mapped.

KQZ8286E **An attribute group with table ID** *table ID* **mentioned in the TMS DLA was not found.**

Explanation: An attribute group specified in a TEMS query from the TMS DLA was not found.

Operator response: Open the Common Data Model mapping and select the attribute group whose variables are to be mapped to Common Data Model attributes.

KQZ8287E **A TEMS query in TMS DLA CDM mapping section** *section_name* **does not contain an attribute group.**

Explanation: An attribute group specified in a TEMS query from the TMS DLA was not found.

Operator response: Open the Common Data Model mapping and select the attribute group whose variables are to be mapped to Common Data Model attributes.

KQZ8289E **A TMS DLA variable definition without a name was found in a query of attribute group** *attribute_group_name***.**

Explanation: This is an informational message only.

KQZ8290E **TMS DLA variable** *attribute_group_name* **for attribute group** *attribute_group_name* **does not have a column ID specified.**

Explanation: This is an informational message only.

KQZ8291E **TMS DLA Variable** *attribute_group_name* **has column ID Variable** *column_ID* **that does not belong to any attribute in attribute group** *attribute_group_name***.**

Explanation: This is an informational message only.

KQZ8292E **A TMS DLA variable definition without a name was defined to transform variable** *parent_variable_name***.**

Explanation: This is an informational message only.

KQZ8293E **TMS DLA Variable definition** *variable_name* **has no method name.**

Explanation: This is an informational message only.

KQZ8294E **TMS DLA variable *variable_name* is defined more than once.**

Explanation: This is an informational message only.

KQZ8295E **TMS DLA variable name *variable_name* is already defined as a predefined variable.**

Explanation: This is an informational message only.

KQZ8296E **A TMS DLA filter in mapping section *mapping_section* had no type or multiple types specified.**

Explanation: This is an informational message only.

KQZ8297E **TMS DLA filter *filter_name* has no variable name.**

Explanation: This is an informational message only.

KQZ8298E **Node *node_name* is not recognized in TMS DLA mapping section *mapping_section*.**

Explanation: This is an informational message only.

KQZ8299E **No ID found in the TMS DLA mapping to CDM class *CDM_class_name*.**

Explanation: This is an informational message only.

KQZ8300E **No source token found in the TMS DLA mapping to CDM class *CDM_class_name*.**

Explanation: This is an informational message only.

KQZ8301E **No source found in the TMS DLA mapping of CDM relationship *CDM_relationship_name*.**

Explanation: This is an informational message only.

KQZ8302E **No source found in the TMS DLA mapping of CDM relationship *CDM_relationship_name*.**

Explanation: This is an informational message only.

KQZ8303E **Attribute *attribute_name* mentioned in the TMS DLA is not found in CDM class *class_name*.**

Explanation: This is an informational message only.

KQZ8304E **TMS DLA variable *variable_name* is not defined.**

Explanation: This is an informational message only.

KQZ8305E **The custom command "*command*" is not a Java invocation command.**

Explanation: This is an informational message only.

KQZ8306E **Grammar "*filter_grammar_string*" in filtered attribute group *attribute_group_name* is not a valid grammar.**

Explanation: This is an informational message only.

KQZ8307E **Function *function_name* is called from the filter in attribute group *attribute_group_name*. It is a cumulative function which is not supported in filters.**

Explanation: The functions count, cumulativeSum, and last are not supported in filters and may return incorrect values.

Operator response: Remove the function call from the formula.

KQZ8308E **The filter formula in attribute group *attribute_group_name* contains a syntax error.**

Explanation: This is an informational message only.

KQZ8309E **A fixed number of bytes and a delimiter are both specified in attribute *attribute_name* in attribute group *attribute_group_name*.**

Explanation: There should be only one way to specify how an attribute value is copied from a record, but both a byte count and a delimiter are specified.

Operator response: Remove the delimiter or the byte count from the specification.

KQZ8310E **A fixed number of bytes and a delimiter are both specified in attribute group *attribute_group_name*.**

Explanation: There should be only one way to specify how an attribute value is copied from a record, but both a byte count and a delimiter are specified.

Operator response: Remove the delimiter or the byte count from the specification.

KQZ8311E **Both a special delimiter and string delimiter are specified in attribute** *attribute_name* **in attribute group** *attribute_group_name***.**

Explanation: There should be only one way to specify how an attribute value is copied from a record, but both a special delimiter and a string delimiter are specified.

Operator response: Remove the special delimiter or the string delimiter from the specification.

KQZ8312E **Both a special delimiter and string delimiter are specified in attribute group** *attribute_group_name***.**

Explanation: There should be only one way to specify how an attribute value is copied from a record, but both a special delimiter and a string delimiter are specified.

Operator response: Remove the special delimiter or the string delimiter from the specification.

KQZ8313E **A starting delimiter was specified without an ending delimiter in attribute** *attribute_name* **in attribute group** *attribute_group_name***.**

Explanation: You cannot describe how an attribute value is copied from a record by specifying only a beginning delimiter. You must specify an ending delimiter alone or both a beginning and ending delimiter.

Operator response: Add an ending delimiter or choose a different way to describe what data is copied into an attribute.

KQZ8314E **A starting delimiter was specified without an ending delimiter in attribute group** *attribute_group_name***.**

Explanation: You cannot describe how an attribute value is copied from a record by specifying only a beginning delimiter. You must specify an ending delimiter alone or both a beginning and ending delimiter.

Operator response: Add an ending delimiter or choose a different way to describe what data is copied into an attribute.

KQZ8315E **The number of bytes copied to the attribute must be an integer greater than** *minimum_byte_count* **and less than or equal to** *maximum_byte_count* **in attribute** *attribute_name* **in attribute group** *attribute_group_name***.**

Explanation: The byte count specified is not a valid integer or not a positive integer.

Operator response: Change the byte count to a positive integer.

KQZ8316E **The number of bytes copied to the attribute must be an integer greater than** *minimum_byte_count* **and less than or equal to** *maximum_byte_count* **in attribute group** *attribute_group_name***.**

Explanation: The byte count specified is not a valid integer or not a positive integer.

Operator response: Change the byte count to a positive integer.

KQZ8317E **The XML element is missing in attribute group** *attribute_group_name***.**

Explanation: An XML log file is indicated but no XML element was given to identify a record.

Operator response: Enter an XML element name.

KQZ8318E **The TEMS query for attribute group** *id* **is** *size* **bytes, which exceeds the limit of** *max_size* **bytes.**

Explanation: The query string for an attribute group is limited in size. If too many attributes are included in an attribute group, then this query string exceeds its limit. Only attributes that are actually sent to the TEMS are included in this count.

Operator response: Remove some attributes, mark some attributes as not being displayed on the TEP, or split the group into two smaller groups.

KQZ8319E **An error occurred generating a TEMS query for the agent.**

Explanation: An error occurred while creating the TEMS queries for the agent.

Operator response: See the log file for more information about the error.

KQZ8320E **A zero-length starting delimiter was specified in attribute** *attribute_name* **in attribute group** *attribute_group_name***.**

Explanation: You cannot describe how an attribute value is copied from a record by specifying an empty starting delimiter.

Operator response: Fill in the starting delimiter or select "Separator text" and fill in only an ending delimiter.

KQZ8321E **A zero-length starting delimiter was specified in attribute group** *attribute_group_name***.**

Explanation: You cannot describe how an attribute value is copied from a record by specifying an empty starting delimiter.

Operator response: Fill in the starting delimiter or select "Separator text" and fill in only an ending delimiter.

KQZ8322E **The file "***file_name***" that is referenced in** *display_name***, does not exist in the project.**

Explanation: This is an informational message only.

KQZ8323E **The file "***file_name***" that is referenced in** *display_name***, does not exist in the project folder** *folder_name***.**

Explanation: This is an informational message only.

KQZ8324E **The file "***file_name***" that is referenced in** *display_name* **does not exist in the "***scripts_folder_name***" folder for operating systems** *operating_system_list***.**

Explanation: This is an informational message only.

KQZ8325E **The file "***file_name***" that is referenced in** *display_name* **was found in the "***folder_name***" folder with letters of a different case "***actual_file_name***".**

Explanation: The file referenced by the agent does not contain the same case in the reference as it does on the file system. This will cause errors if the project is moved to a file system that uses case-sensitive file names, or if the agent is deployed on a file system that uses case-sensitive file names.

Operator response: Fix the reference in the agent to match the case of the file on the file system.

KQZ8326E **No object ID (OID) filter is specified in SNMP event attribute group "***attribute_group_name***".**

Explanation: An SNMP event attribute group must have a filter which describes which events are shown in the attribute group. This attribute group has no filter.

Operator response: Select one of the event options in the "SNMP event information" section of the panel. If select Custom Events, enter the object IDs that should be shown in this attribute group, separated by commas.

KQZ8327E **JDBC statement is missing.**

Explanation: A JDBC attribute group must have a JDBC statement which can retrieve data from the database. This attribute group has no statement.

Operator response: Enter an SQL query or stored procedure which will run to collect data from the database.

KQZ8328E **WMI namespace is missing.**

Explanation: A WMI attribute group must have a namespace and class name which is used to retrieve data from WMI. This attribute group has no namespace.

Operator response: Enter a WMI namespace.

KQZ8329E **WMI class name is missing.**

Explanation: A WMI attribute group must have a namespace and class name which is used to retrieve data from WMI. This attribute group has no class name.

Operator response: Enter a WMI class name.

KQZ8330E **WMI class name is missing.**

Explanation: A WMI attribute group must have a namespace and class name which is used to retrieve data from CIM. This attribute group has no class name.

Operator response: Enter a CIM class name.

KQZ8332E **SNMP performance object display name is missing.**

Explanation: An SNMP attribute group must have a display name by which the attribute group is identified in the Performance Object Status table. This attribute group has no performance object display name.

Operator response: Enter a performance object name.

KQZ8333E **Log file name is missing.**

Explanation: A log file attribute group must specify the name for each log file to be monitored. This attribute group does not specify a name for one or more files.

Operator response: Enter a file name, optionally using wild cards, for each log file to be monitored.

KQZ8334E **The JMX MBean name or pattern is missing.**

Explanation: The JMX MBean name or pattern has not been entered.

Operator response: Enter a valid MBean name or MBean pattern.

KQZ8335E The Director REST object name is missing.

Explanation: The REST object name has not been entered.

Operator response: Enter a REST object name.

KQZ8336E Agent *agent_product_code* does not contain any monitoring actions.

Explanation: No data sources were specified in this agent, and therefore there is nothing that it can monitor.

Operator response: Add one or more data sources to the agent.

KQZ8337E CIM namespace is missing.

Explanation: A CIM attribute group must have a namespace and class name which is used to retrieve data from CIM. This attribute group has no namespace.

Operator response: Enter a CIM namespace.

KQZ8338E OSLC type is missing.

Explanation: An OSLC resource definition must specify an OLSC type of the resource.

Operator response: Add the OSLC type to the resource.

KQZ8339E OSLC type "*agent_product_code*" is not valid.

Explanation: Only OSLC types Location, ServiceInstance, SoftwareServer, SoftwareModule, and StorageVolume are allowed

Operator response: Select one of the valid OSLC types.

KQZ8340E Attribute group is missing from the OSLC resource definition.

Explanation: The resource definition represents the same entity that one of the agent's attribute groups represents, and no attribute group was specified.

Operator response: Select an attribute group whose rows represent the same object that this resource represents.

KQZ8341E Attribute group "*attribute_group_name*" is not valid.

Explanation: The resource definition represents the same entity that one of the agent's attribute groups represents, and the attribute group specified is not defined in this agent.

Operator response: Select an attribute group whose rows represent the same object that this resource represents.

KQZ8342E Dependent attribute group "*attribute_group_name*" is not valid.

Explanation: The resource definition specified a dependent attribute group that does not exist.

Operator response: Select a valid attribute group.

KQZ8343E Attribute or value is missing from an OSLC property definition.

Explanation: An OSLC property value is obtained from an ITM attribute value. No attribute was specified in this property definition.

Operator response: Select an attribute from the resource's main or dependent attribute groups.

KQZ8344E Attribute group "*attribute_group_name*" referenced on OSLC property "*attribute_group_name*" is not valid.

Explanation: An attribute group that is referenced in an OSLC property value is not a valid attribute group.

Operator response: Edit the property value and change the reference to an attribute from either the resource's main attribute group or from one of the dependent attribute groups.

KQZ8345E Relationship type not specified.

Explanation: An OSLC relationship must specify a relationship type such as "depends on". This relationship does not specify a type.

Operator response: Enter the OSLC relationship type.

KQZ8346E Relationship target resource definition not specified.

Explanation: An OSLC relationship is defined in a source relationship and must reference a target resource by name. This relationship does specify a target resource.

Operator response: Select a target resource definition.

KQZ8347E The specified target resource type *resource_type* does not match the actual type *resource_type* of resource *resource_name*.

Explanation: An OSLC relationship is defined in a source relationship and must reference a target resource by name. A target resource type can also be specified. In this case, the type specified does not match the resources defined type.

Operator response: Remove or correct the specified

target resource type or select a different target resource.

KQZ8348E **No criteria specified to match resource** *source_resource_name* **to** *target_resource_name***.**

Explanation: The criteria used to match a source and target resource in a relationship is missing.

Operator response: Enter one or more property paris to match source and target resources.

KQZ8349E **Property** *property_name* **is not defined by resource** *resource_name***.**

Explanation: The criteria used to match a source and target resource specified a property in one of those resources that does not exist.

Operator response: In the matching criteria, enter a valid property for the source resource and the target resource.

KQZ8350E **Matching criteria property missing. for resource** *resource_name***.**

Explanation: The criteria used to match a source and target resource did not specify a property for one or both resources.

Operator response: In the matching criteria, enter a valid property for the source resource and the target resource.

KQZ8351E **Method name is missing.**

Explanation: When an OSLC method is specified, a method name is required.

Operator response: Specify a method name.

KQZ8352E **Log source specifies both regExFileName and regExStrictFileName attributes.**

Explanation: A file name pattern may either be a legacy regExFileName or a preferred regExStrictFileName, it can't be both. The agent will be generated assuming regExStrictFileName is correct.

Operator response: Only set one of the 2 flags in the agent definition.

KQZ8353E **The directory containing the OSLC definitions "***directory_name***" does not exist.**

Explanation: A directory containing the RDF resource shapes XML file was set with the preferences and that directory no longer exists.

Operator response: Extract the RDF resource shapes XML file from a running instance of OSDP and save it as resource-shapes.xml. Identify the location of that file using Window - Preferences - IBM Tivoli Monitoring - OSLC.

KQZ8354E **The OSLC definitions file "***file_name***" does not exist in directory "***directory_name***".**

Explanation: A directory containing the OSLC definition file was set with the preferences and the OSLC definition file longer resides in that directory (or cannot be read from that directory).

Operator response: Extract the resource-shapes.xml file from a running instance of OSDP and identify the location of that file using Window - Preferences - IBM Tivoli Monitoring - OSLC.

KQZ8355W **Attribute group "***attribute_group_name***" referenced by OSLC property** *OSLC_property_name***, has no row selection criteria defined.**

Explanation: An OSLC property value references an attribute from an attribute group other than the main attribute group for the resource. The attribute group can return multiple rows, but there is no criteria defined to select which row corresponds to the particular row from the main attribute group.

Operator response: Open the resource dependencies dialog and add criteria to match a row from the resource's main attribute group to a row from the referenced attribute group.

KQZ8357E **Attribute "***attribute_name***" does not exist in attribute group "***attribute_group_name***". This occurred in the specification of dependent attribute group "***dependent_attribute_group_name***"**

Explanation: An attribute used to match data rows from the main attribute group to the dependent attribute group is not a valid attribute.

Operator response: Select a valid attribute from the main attribute group.

KQZ8358E **Attribute "***attribute_name***" does not exist in dependent attribute group "***attribute_group_name***".**

Explanation: An attribute used to match data rows from the main attribute group to the dependent attribute group is not a valid attribute.

Operator response: Select a valid attribute from the dependent attribute group.

KQZ8359E **Attribute "*attribute_name*" does not exist in attribute group "*attribute_group_name*".**

Explanation: An attribute that is referenced in an OSLC property value does not exist in the referenced attribute group.

Operator response: Edit the property value and change the reference to an attribute from either the resource's main attribute group or from one of the dependent attribute groups.

KQZ9000I **The agent has successfully completed startup processing.**

Explanation: This is an informational message only.

KQZ9001I **The agent is shutting down.**

Explanation: This is an informational message only.

KQZ9002I **Received Take action command <*command*>**

Explanation: This is an informational message only.

KQZ9003I **Starting the custom data provider client named '*client*'**

Explanation: This is an informational message only.

KQZ9004I **Restarting the custom data provider client named '*client*'**

Explanation: This is an informational message only.

KQZ9005I **Processing a dynamic configuration update request.**

Explanation: This is an informational message only.

KQZ9006I **Custom data client connected from system *client*.**

Explanation: This is an informational message only.

KQZ9007I **Agent shutdown request from data provider on socket *socket*.**

Explanation: This is an informational message only.

KQZ9999E **RKANDATV Member *filename* not found.**

Explanation: This is an informational message only.

Agent trace logs

Trace logs capture information about the operating environment when component software fails to operate as intended.

The principal log type is the RAS (Reliability, Availability, and Serviceability) trace log. These logs are in the English language only. The RAS trace log mechanism is available for all components of IBM Tivoli Monitoring. Most logs are in a `logs` subdirectory on the host computer. See the following sections to learn how to configure and use trace logs:

- "Principal trace log files" on page 492
- "Example for using trace logs" on page 494
- "Setting RAS trace parameters" on page 495

Note: The documentation refers to the RAS facility in IBM Tivoli Monitoring as **`RAS1`**.

IBM Software Support uses the information that is captured by trace logs to trace a problem to its source or to determine why an error occurred. Although the default configuration for trace logs is to be enabled, it can be disabled.

Overview of log file management

(Table 48 on page 493) provides the names, locations, and descriptions of RAS1 log files. The log file names adhere to the following naming convention:

`hostname_productcode_``[`*`instance_`*`]k`*`product_code`*`agent_`*`timestamp-nn`*`.log`

where:

- *hostname* is the host name of the machine on which the monitoring component is running.
- *product_code* is the last two characters of the product code (for example, for K33, it would be 33).

- *instance* is the instance ID of a multi-instance agent. The instance ID can be up to 26 characters long.
- *timestamp* is an eight-character hexadecimal time stamp that represents the time at which the program started.
- *nn* is a rolling log suffix. See "Examples of trace logs" for details of log rolling.

Single and multi-instance agents build differently formatted log file names. Multi-instance agents include the instance ID in the log file name. The following example is for a multi-instance agent that runs on `balayne`, with product code `k11` and instance ID `inst1`:

```
BALAYNE_11_inst1_k11agent_46c31514-01.log
```

The following example is for a single instance agent that runs on `balayne`, with product code `13`:

```
inst1 BALAYNE_13_k13agent_46c49270-01.log
```

Examples of trace logs

For example, if a monitoring agent is running on computer `server01` with the product code of `K33`, the RAS log file for a data source might be named:

```
server01_33_437fc59-01.log
```

For long-running programs, the *nn* suffix is used to maintain a short history of log files for that startup of the program. For example, the data source might have a series of log files as follows:

```
server01_33_437fc59-01.log
server01_33_437fc59-02.log
server01_33_437fc59-03.log
```

As the program runs, the first log (*nn*=01) is preserved because it contains program startup information. The remaining logs roll: when the set of numbered logs reaches a maximum size, the remaining logs are overwritten in sequence.

Each time a program is started, a new time stamp is assigned to maintain a short program history. For example, if the data source is started twice, it might have log files as follows:

```
server01_33_437fc59-01.log
server01_33_437fc59-02.log
server01_33_437fc59-03.log
server01_33_537fc59-01.log
server01_33_537fc59-02.log
server01_33_537fc59-03.log
```

Principal trace log files

(Table 48 on page 493) contains locations, file names, and descriptions of trace logs that can help determine the source of problems with agents.

Table 48. Trace log files for troubleshooting agents

System where log is located	File name and path	Description
On the Tivoli Enterprise Monitoring Server	• On Windows: The file `IBM Tivoli Monitoring` *`timestamp`*`.log` in the *`install_dir`*`\InstallITM` path. • On UNIX: The `candle_installation.log` file in the *`install_dir`*`/logs` path.	Provides details about products that are installed. **Note:** Trace logging is enabled by default. A configuration step is not required to enable this tracing.
	The `Warehouse_Configuration.log` file is in the following path on Windows: *`install_dir`*`\InstallITM`. On UNIX, the file is in the following path: *`install_dir`*`/Install`	Provides details about the configuration of data warehousing for historical reporting.
	The name of the RAS log file is as follows: • On Windows: *`install_dir`*`\logs\ hostname_ms_HEXtimestamp-nn.log` • On UNIX: *`install_dir`*`/logs/ hostname_ms_timestamp.log` **Note:** File names for RAS1 logs include a hexadecimal time stamp. Also on UNIX, a log with a decimal time stamp is provided: *`hostname_productcode_timestamp`*`.log` and *`hostname_productcode_timestamp`*`.pid`*`nnnnn`* in the *`install_dir/logs`* path, where *nnnnn* is the process ID number.	Traces activity on the monitoring server.
On the Tivoli Enterprise Monitoring Server	The name of the RAS log file is as follows: • On Windows: *`install_dir`*`\logs\ hostname_cq_timestamp-nn.log` • On UNIX: `install_dir/logs/` *`hostname`*`_cq_`*`timestamp-nn`*`.log` **Note:** File names for RAS1 logs include a hexadecimal time stamp. Also on UNIX, a log with a decimal time stamp is provided: *`hostname_productcode_timestamp`*`.log` and *`hostname_productcode_ timestamp`*`.pid`*`nnnnn`* in the *`install_dir/logs`* path, where *nnnnn* is the process ID number.	Traces activity on the portal server.
	The `teps_odbc.log` file is in the following path on Windows:*`install_dir`*`\InstallITM` path. On UNIX, the file is in the following path: *`install_dir`*`/logs`	When you enable historical reporting, this log file traces the status of the warehouse proxy agent.
On the computer that hosts the monitoring agent	The RAS1 log files are named *`hostname_productcode`*`_[`*`instance`*`]_k`*`[productcode]`* `agent_`*`timestamp-nn`*`.log` **Note:** 1. The files are in the *`install_dir`*`\tmaitm6\logs` path. 2. File names for RAS1 logs include a hexadecimal time stamp. 3. Multi-instance agents include the instance ID in the log file name.	Traces activity of the monitoring agent.

Table 48. Trace log files for troubleshooting agents (continued)

System where log is located	File name and path	Description
On the computer that hosts the monitoring agent that includes JDBC or JMX data sources	The JMX, JDBC, or Java log file is named as follows, depending on whether the agent uses JMX, JDBC, or both JMX and JDBC (Java): • For a single instance agent *KXX_JMX*|*JDBC*|*JAVA*_trace.log • For a multiple instance agent *KXX_JMX*|*JDBC*|*JAVA_instanceName*_trace.log	Traces activity of the data provider Java process for JMX, JDBC, or both.
The RAS1 log files are named *hostname_productcode*_[*instance*_]_k*[productcode]*agent_*timestamp-nn*.log and are in the *install_dir*\tmaitm6\logs path. **Note:** File names for RAS1 logs include a hexadecimal time stamp. **Note:** Multi-instance agents include the instance ID in the log file name. Definitions of variables: *timestamp* is time stamp whose format includes year (y), month (m), day (d), hour (h), and minute (m), as follows: **yyyymmdd hhmm** *HEXtimestamp* is a hexadecimal representation of the time at which the process was started. *install_dir* represents the directory path where you installed the IBM Tivoli Monitoring component. *install_dir* can represent a path on the computer that host the monitoring system, the monitoring agent, or the portal. *instance* refers to the name of the database instance that you are monitoring. *hostname* refers to the name of the computer on which the IBM Tivoli Monitoring component runs. *nn* represents the circular sequence in which logs are rotated. The range is 1-5, by default, though the first is always retained because it includes configuration parameters.		

For more information about the complete set of trace logs that are maintained on the monitoring server, see the *IBM Tivoli Monitoring Installation and Setup Guide*.

Example for using trace logs

Typically IBM Software Support applies specialized knowledge to analyze trace logs to determine the source of problems. However, you can open trace logs in a text editor to learn some basic facts about your IBM Tivoli Monitoring environment. The following example is from the log for a Tivoli Universal Agent data source that is created by the Agent Builder:

```
!47CC1929.0000!==========>  IBM Tivoli RAS1 Service Log  <=========
+47CC1929.0000  System Name: BALAYNE Process ID: 2152
+47CC1929.0000  Program Name: k01agent  User Name: SYSTEM
+47CC1929.0000  Task Name: k01agent  System Type: WinXP;5.1-SP2
+47CC1929.0000  MAC1_ENV Macro: 0xC112  Start Date: 2008/03/03
+47CC1929.0000  Start Time: 10:28:41 CPU Count: 2
+47CC1929.0000  Page Size: 4K  Phys Memory: 2047M
+47CC1929.0000  Virt Memory: 2048MPage Space: 3429M
+47CC1929.0000  Service Point: system.balayne_01 UTC Start Time: 47cc1929
+47CC1929.0000  ITM Home: C:\IBM\ITM ITM Process: balayne_01
+47CC1929.0000  Executable Name: C:\IBM\ITM\TMAITM6\k01agent.exe
+47CC1929.0000  KBB_RAS1: ERROR
+47CC1929.0000  KBB_RAS1_LOG: C:\IBM\ITM\TMAITM6\logs\BALAYNE_01_k01agent_47cc1929-.log
INVENTORY=C:\IBM\ITM\TMAITM6\logs\BALAYNE_01_k01agent.inv COUNT=03 LIMIT=5 PRESERVE=1 MAXFILES=9
+47CC1929.0000  KBB_ENVPATH: C:\IBM\ITM\TMAITM6\KQRENV
+47CC1929.0000 ================================================
(47CC1929.0000-2390:RAS1,400,"CTBLD")
+47CC1929.0000 Component: kbb
+47CC1929.0000 Driver: d7310a/3784490.1
+47CC1929.0000 Timestamp: Nov  6 2007 17:51:39
+47CC1929.0000 Target: wv7i386
(47CC1929.0001-2390:RAS1,400,"CTBLD")
```

```
+47CC1929.0001 Component: kdy
+47CC1929.0001 Driver: d7310a/3784490.23
+47CC1929.0001 Timestamp: Nov  6 2007 18:36:38
+47CC1929.0001 Target: wv7i386
(47CC1929.0002-2390:RAS1,400,"CTBLD")
+47CC1929.0002 Component: kns
+47CC1929.0002 Driver: d7310a/3784490.9
+47CC1929.0002 Timestamp: Nov  6 2007 18:20:58
+47CC1929.0002 Target: wv7i386
(47CC1929.0003-2390:RAS1,400,"CTBLD")
+47CC1929.0003 Component: ira
+47CC1929.0003 Driver: dev/3751458.660
+47CC1929.0003 Timestamp: Mar  3 2008 10:22:31
+47CC1929.0003 Target: wv7i386
(47CC1929.0004-2390:kbbssge.c,52,"BSS1_GetEnv") CANDLE_HOME="C:\IBM\ITM"
(47CC1929.0005-2390:kbbssge.c,52,"BSS1_GetEnv") CDP_DP_ACTION_TIMEOUT="20"
(47CC1929.0006-2390:kbbssge.c,52,"BSS1_GetEnv") CDP_DP_CACHE_TTL="30"
(47CC1929.0007-2390:kbbssge.c,52,"BSS1_GetEnv") CDP_PURE_EVENT_CACHE_SIZE="100"
(47CC1929.000A-2390:kbbssge.c,52,"BSS1_GetEnv") AGENT_REF_FILE="C:\IBM\ITM\TMAITM6\KQR.ref"
(47CC1929.000B-2390:query.cpp,698,"CtPreInit") AGENT_REF_FILE = C:\IBM\ITM\TMAITM6\KQR.ref
(47CC1929.000C-2390:query.cpp,241,"ApplicationXMLConfigParser::startElement") Returning
 service name = KQRCMA
(47CC1929.000D-2390:kbbssge.c,52,"BSS1_GetEnv") CANDLE_HOME="C:\IBM\ITM"
(47CC1929.000E-2390:kbbssge.c,52,"BSS1_GetEnv") AGENT_REF_FILE="C:\IBM\ITM\TMAITM6\K01.ref"
```

This example shows that the data source started up correctly. You can see the values for variables that can be used to configure the data source (`CDP_PUSH_INTERVAL_SECS` and `CDP_NT_EVENT_LOG_GET_ALL_ENTRIES_FIRST_TIME`).

On Windows, you can use the following alternative method to view trace logs:

1. In the Windows **Start** menu, choose **Program Files** > **IBM Tivoli Monitoring** > **Manage Tivoli Enterprise Monitoring Services**. The Managing Tivoli Enterprise Monitoring Services window is displayed.
2. Right-click a component and select **Advanced** > **View Trace Log** in the menu. For example, if you want to view the trace log of an agent, right-click the name of the agent in the window. You can also use the viewer to access remote logs.

   Note: The viewer converts time stamps in the logs to a format that is easier to read.

For information about the **`rastlog`** tool, see the *IBM Tivoli Monitoring Troubleshooting Guide*.

Setting RAS trace parameters

Pinpoint a problem by setting detailed tracing of individual components of the monitoring agent and modules.

Before you begin

To ensure that you understand log rolling and can reference the correct log files when you manage log file generation, see ("Overview of log file management" on page 491).

About this task

The Agent Builder agents and agent runtime use RAS1 tracing and generate the logs that are described in (Table 48 on page 493). The default RAS1 trace level is ERROR.

Specify RAS1 trace options in the *agent_name*`ENV` file on Windows systems, and in the *agentname*`.ini` file on UNIX systems.

Procedure

Use one of the following procedures:

- Optional: Manually edit the configuration file to set trace log preferences:
  1. Open the file which contains the trace options.
     - On Windows: *install_dir*\TMAITM6\K*PRODUCTCODE*ENV[*_instance*] or *install_dir*\TMAITM6_x64\K*PRODUCTCODE*ENV[*_instance*]
     - On UNIX: *install_dir*\config*productcode*.ini for all new instances or *productcode*[*_instance name*].config
  2. Edit the line that begins with KBB_RAS1= to set trace log preferences. For example, if you want detailed trace logs, set the Maximum Tracing option:KBB_RAS1=ERROR (COMP:kqz ALL) (UNIT:kra ALL)
  3. Edit the line that begins with KBB_RAS1_LOG= to manage the generation of log files:
     - MAXFILES: the total number of files that are to be kept for all starts of a program. After this value is exceeded, the oldest log files are discarded. Default value is 9.
     - LIMIT: the maximum size, in megabytes (MB) of an RAS1 log file. Default value is 5.
     - IBM Software Support might guide you to modify the following parameters:
       - COUNT: the number of log files to keep in the rolling cycle of one program startup. Default is 3.
       - PRESERVE: the number of files that are not to be reused in the rolling cycle of one program startup. Default value is 1.

     Note: The KBB_RAS1_LOG parameter also provides for the specification of the log file directory, log file name, and the inventory control file directory and name. Do not modify these values or log information can be lost.
  4. Restart the monitoring agent so that your changes take effect.
- Optional: For Windows operating systems only, you can use the following alternative method to edit trace log parameters:
  1. Open the Manage Tivoli Enterprise Monitoring Services window.
  2. Right-click the icon of the monitoring agent whose logging you want to modify.
  3. Select **Advanced** > **Edit Trace Parms**. The Tivoli Enterprise Monitoring Server Trace Parameters window is displayed.
  4. Select a new trace setting in the menu in the **Enter RAS1 Filters** field or type a valid string.
  5. Modify the value for **Maximum Log Size Per File (MB)**to change the log file size (changes LIMIT value).
  6. Modify the value for **Maximum Number of Log Files Per Session** to change the number of log files per startup of a program (changes COUNT value).
  7. Modify the value for **Maximum Number of Log Files Total** to change the number of log files for all starts of a program (changes MAXFILES value).
  8. Optional: Click **Y** (Yes) in the **KDC_DEBUG Setting** menu to log information that can help you diagnose communications and connectivity problems between the monitoring agent and the monitoring server. The **KDC_DEBUG** setting and the Maximum error trace setting can generate many trace log messages. Use them only temporarily, while you are troubleshooting problems. Otherwise, the logs can occupy excessive amounts of hard disk space.
  9. Click **OK**. You see a message report a restart of the monitoring agent so that your changes take effect.

What to do next

Monitor the size of the `logs` directory. Default behavior can generate a total of 45 to 60 MB for each agent that is running on a computer. See the "Procedure" section to learn how to adjust file size and numbers of log files to prevent logging activity from occupying too much disk space.

Regularly prune log files other than the RAS1 log files in the `logs` directory. Unlike the RAS1 log files that are pruned automatically, other log types can grow indefinitely. For example, the logs in (Table 48 on page 493) that include a process ID number (PID).

The following list comprises the RAS1 trace parameters settings:

- No error tracing. `KBB_RAS1=-none`
- General error tracing. `KBB_RAS1=ERROR`
- Intensive error tracing. `KBB_RAS1=ERROR (COMP:kqz ALL)`
- Agent interaction with the Tivoli Enterprise Monitoring Server. `KBB_RAS1=ERROR (UNIT:genericagent ALL) (UNIT:kra ALL)`
- Trace data collection. `KBB_RAS1=ERROR (UNIT: ALL)`
- Trace Script data collection. `KBB_RAS1=ERROR (UNIT:shell ALL) (UNIT:commandwithtimeout ALL)`
- Trace Availability data collection. `KBB_RAS1=ERROR (UNIT:availability ALL) (UNIT:winavailability ALL)`
- Trace Windows Event Log data collection. `KBB_RAS1=ERROR (UNIT:winlog ALL) (UNIT:eventlog ALL)`
- Trace log file data collection. `KBB_RAS1=ERROR (UNIT:logmonitor ALL)`
- Trace SNMP data collection. `KBB_RAS1=ERROR (UNIT:snmp ALL)`
- Trace ICMP Ping data collection. `KBB_RAS1=ERROR (UNIT:ping ALL)`
- Trace CIM data collection. `KBB_RAS1=ERROR (UNIT:cim ALL)`
- Trace PerfMon data collection. `KBB_RAS1=ERROR (UNIT:queryclass ALL)`
- Trace WMI data collection. `KBB_RAS1=ERROR (UNIT:wmi ALL)`
- Trace separate collector interface. `KBB_RAS1=ERROR (UNIT:custom ALL) (UNIT:CPS_Socket ALL) (UNIT:cpci ALL)`

Note: The **KDC_DEBUG** setting and the Maximum error trace setting can generate many trace logs. Use them only temporarily, while you are troubleshooting problems. Otherwise, the logs can occupy excessive amounts of hard disk space.

Problem classification

Symptom descriptions and detailed workarounds for troubleshooting when you are using the Agent Builder or when you are working with your agent.

See the *Tivoli Monitoring: Troubleshooting Guide, GC32-9458* for general problem determination. Also, see the *IBM Tivoli Monitoring Messages Guide* for general IBM Tivoli Messages.

Troubleshooting: Agent Builder

(Table 49 on page 499) shows solutions for installation, configuration, and uninstallation problems.

- Maximum data size exceeded
- Receive error when installing Tivoli Enterprise Monitoring support on Windows system
- Display name is not refreshed
- Demo CD problems
- The -console flag does not work
- No help is displayed when clicking help links within the Agent Builder

- AgentBuilder does not start on a 64 bit Linux system
- Linux Gtk library warning messages
- Problems launching the Agent Builder on a non-English system
- Remote service browsing error
- Key attribute missing
- Change command return code or script data provider executable
- Error when attempting to launch Eclipse
- Browsing processes or services on a remote system do not work
- When I generate an agent and omit the .bat or .cmd in the script data source command, the script does not run.
- Resource out of sync
- Parsing MIBs
- MIBs frequently have errors
- Agent Builder version on UNIX
- Adding new attributes to log file data source
- Agent Builder does not allow duplicate data source names in different subnodes
- CIM integer array value
- No CIM support for certificates
- How do I display large numeric values?
- Parse Log window characters corrupted
- Adding data source Browse button not active
- Agent Builder CIM Browser does not show CIM Classes for AIX OpenPegasus 2.6.1.
- Agent Builder CIM Browser does not show CIM classes for Solaris WBEM Server
- JDBC connection to zOS DB2 database is failing
- Specifying the internal_logon connection property for Oracle
- The Agent Builder uninstall ends with an error.
- Project default location error
- Product code and affinity errors not cleared
- Agent Builder script argument parameterization does not work.
- Error message in any of the Agent Builder wizards is truncated.
- JRE or JDK not available
- Installing the agent locally fails with error KQZ0208E
- iFixes applied to the Agent Builder are not removed when the Agent Builder is uninstalled
- Agent Builder workspace screen not responding in UNIX over a Cygwin Xserver
- Using the Parse function on the Parse Log window, returns incorrect results, such as no rows appear
- JMX browser fails to connect to WebLogic 10.x

Table 49. Problems and solutions for installation and use of the Agent Builder

Problem	Solution
Maximum data size exceeded When you save an agent, you might see the following error in the Problems View: `Data source DataSourceName exceeds the maximum data size of 8192 bytes`	Tivoli Monitoring currently limits the amount of data that can be returned for each row in a table to 8192 bytes. For each data source in the agent, the Agent Builder computes the sizes of the attributes in the following ways: • Only attributes marked `Display attribute in the Tivoli Enterprise Portal` are counted. • Each 32-bit numeric attribute is 4 bytes. Each 64-bit numeric attribute is 8 bytes. • For each string attribute, the value for maximum size is used (with a maximum value of 2048 bytes). Ensure that the sizes listed for string attributes are set to reasonable values (under 100 bytes). • Any request for data from an attribute group causes collection of all the data. Defining large attribute groups can increase processor and memory usage in the monitoring system. Where the data does not need to be displayed together, form smaller groups of data to optimize the behavior of the system.
An error occurs when you install Tivoli Enterprise Monitoring support on Windows system I tried to install Tivoli Enterprise Monitoring support on a Windows Tivoli Enterprise Monitoring with an Agent Builder script, and received the following error: `F:\wspace1\CCMDB\export>installIra.bat f:\IBM\ITM6.2` `Install of K41 Agent successful.` `FIND: `VRMF=': No such file or directory` `06200100 was unexpected at this time.`	Reset your PATH to ensure that the Windows **find** command is found first. It is possible that something was installed that is now the first **find** command in the PATH. For example, if you installed Cygwin (a set of UNIX tools for Windows). If Cygwin is first in your PATH, you hit the Cygwin **find** command rather than the built-in Windows **find** command.
Display name is not refreshed If you are browsing to add a process or a service and select one, and then you select a different one before you click **OK**, the display name is not refreshed for the new process or service that is selected.	Delete the characters that are in the display name field, and then browse.
Demo CD problems I am running the demo CD as my Tivoli Monitoring system. I selected the managed node to browse. I am shown an unusual set of processes and UNIX operating systems is automatically selected for my process instead of Windows.	You selected the wrong agent. The Demo CD contains many agents which would not normally be found together on a system. Stop the other OS agents and browse by using the real Monitoring Agent for Windows. Alternatively browse by using a system that has a normal Tivoli Monitoring installation.
The **-console** flag does not work You receive a message: `The installer is unable to run in graphical mode. Try running the installer with the -console or -silent flag` You try to run the installer with the **-console** flag, but it does not work.	The silent installation does not support the **-console** option. It does, however, support the **-silent** option. For more information, see (“Silent installation” on page 7).

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
No help is displayed when you click help links within the Agent Builder On an AIX system, you might see the following messages: `The context help for this user interface element could not be found. Open in dynamic help.` Clicking the **Open in dynamic help** link does not do anything.	On some AIX and Linux systems, Eclipse is unable to determine the default system browser. To correct this issue, you must set the default browser in Agent Builder. For more information, see "Set the default browser in Agent Builder" on page 9.
Agent Builder does not start on a 64-bit Linux system Agent Builder does not start on my 64-bit Linux system. No errors are displayed or logged.	Make sure that you have the prerequisite software installed. Agent Builder is a 32-bit application so the 32-bit version of the prerequisites must be installed. For more information, see "Requirements" on page 5

Linux `Gtk` library warning messages
 	
Even after I install the latest Linux 32-bit libraries for Agent Builder I see the following messages:
`(Agentbuilder:10801): Gtk-WARNING **: Unable to locate theme engine in module_path: "clearlooks",`
 	
`Gtk-Message: Failed to load module "pk-gtk-module": libpk-gtk-module.so: cannot open shared object file: No such file or directory`
 	
`Gtk-Message: Failed to load module "canberra-gtk-module": libcanberra-gtk-module.so: cannot open shared object file: No such file or directory`
	These messages can be ignored. The `pk-gtk-module` and `canberra-gtk-module` packages do not have to be installed.

Problems running the Agent Builder on a non-English system When you run the Agent Builder on a non-English system, the JVM might have difficulty to determine the locale of the system. This issue prevents the Agent Builder from running.	To get around this issue, edit *install_location*`/agentbuilder.ini` and add the following 2 lines: `-Dfile.encoding` *LOCALE_PREFIX* Where *LOCALE_PREFIX* is the prefix for the locale the system is running (that is, FR for French). Also, the **-vm** line and the line immediately after must remain the first two lines in the file. Add this option AFTER those lines but before the `-vmargs`line: `-vm`*install_location*`/_jvm/jre/bin/javaw -Dfile.encoding` *LOCALE_PREFIX* `-vmargs`
Remote service browsing error A failure occurs when you enter the host name and IP address of the local system when you try to browse WMI or services remotely.	Browsing in this way is not supported. You receive an error that the builder was unable to open a connection to the host. If you want to browse WMI or services locally, select **localhost** from the list instead of entering the host IP address.

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
Key attribute is missing When you save an agent, you might see the following error in the Problems View: `Data source DataSourceName does not contain a key attribute.`	You must have key attributes in the following situation only: If the data source can return multiple rows of data, select an appropriate attribute or attributes as keys. If the data source can return only one row of data, select **Produces a single data row** for the data source. There are other side effects too. Situations can combine data from multiple attribute groups which are defined to return only one row.
Change command return code or script data provider executable file When you change a command return code or a script data provider executable file, newly defined return codes do not show up.	If you change the executable file, then you must copy the new executable file into the agent's project directory under the scripts subdirectory. Copying ensures that the new executable file is picked up when you regenerate the agent.
Error when you attempt to start Eclipse: Agent Builder fails to start and displays an error that points to a log file in the configuration directory. The log file contains: `No application id has been Found.`	Uninstall Agent Builder, and reinstall it in a fresh directory.
Browsing processes or services on a remote system do not work	If use the browse remote system function, the Tivoli Enterprise Monitoring Server, the Tivoli Enterprise Portal Server and the OS agent are required to be running. Restarting the Tivoli Enterprise Monitoring Server or starting it after the OS agent causes the OS agent to go offline. The agent goes offline until its heartbeat with the Tivoli Enterprise Monitoring Server is done. Alternatively the OS agent can be restarted.
When I generate an agent and omit the .bat or .cmd in the script data source command, the script does not run.	Scripts in Windows are frequently started without specifying the .bat or .cmd extension on the command line. Do not omit the .bat or .cmd in the script data source command.
Resource out of sync I edited my agent xml file outside the project. I then get this error when I open the project: `Unable to create this part due to an internal error.Reason for the failure: Resource is out of sync with the file system: /bounds/itm_toolkit_agent.xml.`	To fix the error, click the project and press **F5** to refresh. Then, close the Agent Editor and reopen it.

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
Parsing MIBs When I load my MIB into the Agent Builder, it did not parse. I see is something like the following messages: `KQZ0022E Unexpected error parsing MIB file D:\mibs\mymib.MIB.Stack trace: com.ibm.tivoli.monitoring.agentkit.AgentException: KQZ0022E Unexpected error parsing MIB file D:\mibs\mymib.MIB.at com.ibm.tivoli.monitoring. agentkit.actions.LoadInitialMIBData.run (Unknown Source)at org.eclipse.jface.operation .ModalContext$ModalContextThread.run (ModalContext.java:113)Caused by: java.lang. NullPointerExceptionat com.tivoli.snmp.metadata .MibParser.OTdefinition(MibParser.java:1198)at com.tivoli.snmp.metadata.MibParser.assignment (MibParser.java:1081)at com.tivoli.snmp. metadata.MibParser.mibModule(MibParser.java:599)`  , and other similar messages  `MibParser.java:1198` and `MibParser.java:1081` might point to other areas of the parser.	To get the MIB to load: The MIB parser that is used by the Agent Builder uses the grammar that is defined by ASN.1 to parse the MIBs. Some MIBs do not follow the grammar correctly. The parser can relax certain rules to accommodate the most common errors. The MIB must be corrected, but until that can be done you can turn off checking for common errors by doing the following steps: 1. On the Agent Builder menu bar, click **Window** > **Preferences**. 2. Click **IBM Tivoli Monitoring Agent Generator** in the selection pane. 3. Select one of the following options in the **MIB Parsing Options** list: **Allow types to start with lowercase letters** Allows types that people write in MIBs, such as values **Allow numeric named numbers** Allows numbers that start with uppercase letters **Allow underscore in value name** Allows underscore characters **Allow values to begin with uppercase letters** Allows various (technically) invalid things **Ignore duplicate MIBs** Turns off warning for duplicate MIB modules
MIBs frequently have errors	As many of these errors as possible are fixed by the Agent Builder. However, some errors are so severe that they cannot be corrected. In many cases, it is possible to edit the MIB, correct the file, and then attempt to import the MIB into the Agent Builder. As MIB errors are discovered, IBM, customers, and IBM Business Partners add information to the A&BSM Technical Exchange Wiki, IBM Tivoli AA&BSM Technical Exchange. The information indicates the MIBs on which errors were discovered and the corrections that can be imported into the Agent Builder. See Parsing MIBs earlier in this table.
Agent Builder version on UNIX The Agent Builder is installed on a UNIX system, and you cannot start the GUI to see the version.	The directory name that matches the following pattern contains the current version of Agent Builder: *install dir*`\features\ com.ibm.tivoli.monitoring.agentkit_X.Y.Z. vYYYYMMDDHHmm` or *install dir*`\features\ com.ibm.tivoli.monitoring.agentkit_*`

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
Adding new attributes to log file data source **Next field** is by definition a way to build a log record from a log file. The record is built in the order in which fields are encountered in the line that is being parsed. IBM Tivoli Monitoring requires that all new attributes be added to the end of an attribute group.	Generally, application logs are what they are and when they change they can change dramatically. It is also often necessary to support the new and old log file formats to correctly support current and old versions of the application. The best way to handle this issue in an agent is to add an attribute group that represents the new log file format. When the agent is deployed to a system, you normally configure the name of the file to be monitored in the appropriate configuration variable. Use the configuration variable so that only the old or new attribute group is used in a specific instance of the agent.
Agent Builder does not allow duplicate data source names in different subnodes In the portal Navigator Physical view, I want the same Navigator node name for data sources that are in different subnode types.	The Agent Builder prevents you from adding warehouse data from two different attribute groups that have the same name in the same agent. A possible workaround to get duplicate names in the Tivoli Enterprise Portal Navigator is to define Navigator groups in the subnode definition. You define in the subnode by using the name that you want to display in the portal. Move the data source into the Navigator group and the data source name is effectively hidden.
CIM integer array value Only the first value of my CIM Integer Array was returned.	The Agent Builder currently does not have an array data type. String Arrays can contain variable length data, all of the strings are concatenated into a single value that is separated by a comma (,). Verify that the overall length of the attribute is large enough to contain all of the expected values. The Integer data type attribute does not accept the, delimiter. Rather than convert the integer array to string data, only the first value in the integer array is returned.
No CIM provider support for certificates No information is displayed in the Tivoli Enterprise Portal. The Agent Builder CIM provider does not support the use of certificates.	CIM provider support is provided only for base, plain HTTP, and HTTPS. The CIM server was configured to require SSL certificates for all communication. The only workaround is to reconfigure the CIM server to allow communication without the use of SSL certificates.
How do I display large numeric values? The browsers create all of my numeric attributes as signed 32-bit values when compatibility with IBM Tivoli Monitoring V6.2 is required. These 32-bit values do not work for unsigned data or large values that I want to collect.	See Negative or wrong attribute value for a technique to use derived attributes to display large numeric values by using signed 32-bit attributes.
Parse Log window characters corrupted Characters that are displayed in the **Results** area of the Parse Log window are corrupted. I expected to see my non-ASCII file contents, displayed correctly in the browser and in the portal after I deploy my agent.	The Agent Builder log file parsing does not support parsing files with names or contents that contain non-ASCII characters.

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
Browse not active when you add a data source When you add a data source, **Browse** is not active when the data source type is WMI, Perfmon, or Windows Event Log; and Agent Builder is running on Linux or AIX.	For the Agent Builder on Linux or AIX, you cannot use the Browse function when you are creating WMI, Perfmon, or Windows Event Log data sources. The Windows APIs must be available on the Agent Builder system to Browse these data sources. You can still define these types of data sources when you run the Agent Builder on Linux or AIX by specifying the data source information manually.

Agent Builder CIM Browser does not show CIM Classes for AIX OpenPegasus 2.6.1.
 	
`Efix 644427.080123` for `sysmgt.pegasus.cimserver.rte` - Fix for PAM stack overflow Vulnerability in OpenPegasus 2.6.1 for AIX is applied to the IBM Pegasus CIM Server.
	This issue is resolved in IBM Pegasus CIM Server V2.6.1.35
	
	You can display the current version of the IBM Pegasus CIM Server file sets by using the following file: `lslpp -l sysmgt.pegasus.cimserver.rte`
	
	Upgrade to IBM Pegasus CIM Server to 2.6.1.35 or later
	
	For current details, see the AIX Common Information Model overview
	
	OR
	
	Temporarily remove `EFIX 644427` to browse the IBM Pegasus CIM Server on a development CIM Server and allow browsing of the CIMOM.
	Note: Be sure that you have access to the `EFIX 644427.080123.epkg.Z` file to reapply the `EFIX`.
	
	Remove the `EFIX`:
	1. Stop **`cimlistener`** and **`cimserver`**:
	• On an AIX system, with IBM Director Agent or Server installed:
	`/opt/ibm/icc/cimom/bin/stopcimlist`
	`/opt/ibm/icc/cimom/bin/stopcim`
	• On an AIX system, without IBM Director Agent or Server installed:
	`cimlistener -s`
	`cimserver -s`
	2. Remove the `efix`:
	`emgr -r -L 644427`
	3. Start **`cimserver`** and **`cimlistener`**:
	• On an AIX system, with IBM Director Agent or Server installed:
	`/opt/ibm/icc/cimom/bin/startcim`
	`/opt/ibm/icc/cimom/bin/startcimlist`
	• On an AIX system, without IBM Director Agent or Server installed:
	`cimserver`
	`cimlistener`
	
	(continued on the next page)

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution

Agent Builder CIM Browser does not show CIM Classes for AIX OpenPegasus 2.6.1.
 	
(continued)
	Reapply the `EFIX`:
	1. Stop **`cimlistener`** and **`cimserver`**:
	• On an AIX system, with IBM Director Agent or Server installed:
	`/opt/ibm/icc/cimom/bin/stopcimlist`
	`/opt/ibm/icc/cimom/bin/stopcim`
	• On an AIX system, without IBM Director Agent or Server installed:
	`cimlistener -s`
	`cimserver -s`
	2. Install the `efix`:
	`emgr -e /tmp/644427.080123.epkg.Z`
	3. Start **`cimserver`** and **`cimlistener`**:
	• On an AIX system, with IBM Director Agent or Server installed:
	`/opt/ibm/icc/cimom/bin/startcim`
	`/opt/ibm/icc/cimom/bin/startcimlist`
	• On an AIX system, without IBM Director Agent or Server installed:
	`cimserver`
	`cimlistener`

Agent Builder CIM Browser does not show CIM classes for Solaris WBEM Server. The Agent Builder CIM Browser connects to a Solaris WBEM Server, but when a Namespace is selected, the CIM Browser displays the following error: `KQZ0224E An unknown error occurred when attempting to connect to the CIM server on host` *`hostname`*`.`	This issue is reported to SUN and is a Vendor Limitation. There is an error in the Solaris WBEM CIMOM. The enumerateClasses method is logging a NullPointerException in the WBEM log. There is no workaround.
JDBC connection to z/OS DB2 database is failing.	Make sure that you are using the correct set of JAR files. The z/OS DB2 connection fails if you are not using the JAR files that include the correct licensing information.

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution

Specifying the internal_logon connection property for Oracle
 	
How do I specify the internal_logon connection property for Oracle?
	If you want to allow users to specify the internal_logon connection property, add the following configuration property to the agent XML file. Add this property to the runtime configuration section of the XML file after you create and saved the agent with at least 1 JDBC data source. Find the KQZ_JDBC_PASSWORD property and insert the following text after that property:
	`<property defaultValue="NONE" editable="false"`
	`  name="KQZ_JDBC_ORACLE_INTERNAL_LOGON"`
	`  required="true" type="restricted">`
	`<label`
	`  msgKey="KQZ_JDBC_INTLOGON">Internal Logon</label>`
	`<description msgKey="KQZ_JDBC_INT_LOGON_D">`
	`The Oracle user name when doing an`
	`internal logon or none to logon normally.`
	`</description>`
	`<values>`
	`<value name="NONE">`
	`<label`
	`  msgKey="KQZ_JDBC_LOGON_NONE">None</label>`
	`</value>`
	`<value name="sysdba">`
	`<label`
	`  msgKey="KQZ_JDBC_LOGON_DBA">sysdba</label>`
	`</value>`
	`<value name="sysoper">`
	`<label`
	`  msgKey="KQZ_JDBC_LOGON_OPER">sysoper</label>`
	`</value>`
	`</values>`
	`</property>`

The Agent Builder uninstallation ends with an error. If the Agent Builder is running when you attempt to uninstall, the uninstaller ends with an error.	Close the Agent Builder and manually delete the Agent Builder installation directory to complete the uninstallation.
Project default location error On the New IBM Tivoli Monitoring Agent Project page, if you clear the **Use default location** check box, and then browse to, or type, the default location, Eclipse gives an error that the specified directory overlaps an existing workspace. If you type a subdirectory underneath, Eclipse gives an error that the directory overlaps an existing project.	If you want to use the default location, select the **Use default location** check box. Do not browse to, or type, the default location.
Product code and affinity errors not cleared In the Agent Builder Editor, if you modify the product code or affinity (company ID or agent ID) of an agent, Agent A, so that it overlaps with the product code or affinity of another agent, Agent B, in the workspace, an error is created against Agent A. If you then modify the product code or affinity of Agent B so that the product codes or affinities are now unique, the error still exists on Agent A.	1. Click **Project** > **Clean** from the Agent Builder menu bar. 2. Click **Clean projects selected**. 3. Select the project that contains Agent A and click **OK**. The agent is revalidated and the error marker is removed.

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution

Agent Builder script argument parameterization does not work.
 	
When you create a script agent, passing a runtime Parameter as an argument to the script at execution time does not work.
	The command line cannot include environment variables. Runtime parameters must be referenced inside the script.
	
	The command cannot be **script ${K51_PROCNAME}**
	
	Instead, inside the script, reference the variable:
	• On Windows:
	`PROCNAME=%K51_PROCNAME%`
	• Linux or UNIX:
	`PROCNAME=$K51_PROCNAME`

An error message in any of the Agent Builder wizards is truncated.	Resize the window to make it wider so that the entire message can be viewed.
JRE or JDK not available You receive the following message: `A Java Runtime Environment (JRE) or Java Development Kit (JDK) must be available in order to run Agentbuilder. No Java virtual machine was found after searching the following locations: C:\Program Files\IBM\ITM\ AgentBuilder/jre/jre/bin/javaw.exe`.	Open the *install_dir*`/agentbuilder.ini` file and modify any lines that have a leading space by removing the leading space.
Installing the agent locally fails with error message KQZ0208E When you generate the agent, install it locally and specify a password with special characters, the login to the monitoring server fails with error message:`KQZ0208E The specified user name or password is incorrect.`	The password cannot include special characters.
iFixes applied to the Agent Builder are not removed when the Agent Builder is uninstalled If you apply an interim fix to the Agent Builder, when you uninstall the Agent Builder, the interim fix is not removed. When uninstalling, an installer message informs you that it was unable to remove all directories.	Remove the interim fix directories manually before you reinstall the Agent Builder.
Agent Builder workspace screen not responding in UNIX over a `Cygwin Xserver` The Agent Builder workspace screen does not respond in UNIX over a `Cygwin Xserver`. When you start Agent Builder, the options on the workspace screen are unavailable and you are unable to do actions.	An Eclipse issue with the `Xming` and `Cygwin` tools and not an issue with the Agent Builder. There is no fix available.
Using the Parse function on the Parse Log window, returns incorrect results, such as no rows shown When you use the Parse function on the Parse Log window, incorrect results are returned, such as no rows shown.	The Test log file parser reads only the last 1,000 lines of the log file in tail mode. Make sure the data that you are trying to test for occurs in the last 1000 lines of your sample log file.

Table 49. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
JMX browser fails to connect to WebLogic 10.x The JMX browser fails to connect to WebLogic 10.x. The configuration settings are correct, but connection errors in the `traceKQZ.log` file show that a `CORBA.MARSHAL` error occurred.	Copy the `wlclient.jar` and the `wljmxclient.jar` files from your WebLogic server installation into the `jre/jre/lib/ext` directory of the Agent Builder installation and restart Agent Builder. This action loads the WebLogic JAR files in the system class loader which makes sure the correct classes are loaded to create the JMX connection. After you create your attribute groups by using the JMX browser, you must remove these two JAR files and restart the Agent Builder.
When I click **Collect Data** in a Test window, no data is returned or the data is not what I expect.	Examples: Log file provider might not detect and process the file yet. Ping might report all zeros since the ping did not happen yet Click **Collect Data** a second time If this solution does not resolve the issue, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data collected. The data that is collected and shown by the Data Collection Status window is described in “Performance Object Status node” on page 545 You can attempt further debugging by looking at the test log files, see (Figure 256 on page 379).

Troubleshooting: Agents

A table that shows problems that might occur with your agents:

- Program fails
- Negative or wrong attribute value
- JMX Notifications
- Locally configuring an agent fails on Windows
- Command line does not allow me to configure an agent
- JMX monitors
- JMX browser connecting to WebSphere 6.1
- Error code for attribute group, but data being returned
- JMX data provider connecting to WebSphere 7.0
- JMX data provider connecting to Oracle WebLogic on AIX
- Title bar not displaying the workspace name
- The log file is not monitored properly
- Agents not translated
- RAS1 log errors
- Log file data mismatch
- Installed agent does not show up
- Situations not showing up
- Queries not showing up

- Error states the startagent command failed
- When running command return code, the return code is not accurately shown
- Attribute group's data missing from Tivoli Enterprise Portal
- Changing version number causes errors
- Changing product code, company identifier, or agent identifier causes problems
- A script provider behaves oddly when edited
- Core dump after editing the ref file
- The Tivoli Enterprise Portal does not show any columns or column headers
- Bad string values collected from SNMP are displayed in attribute groups
- No data in the Tivoli Enterprise Portal
- Cannot remove agent from the Tivoli Enterprise Portal
- Service monitoring returns zeros for metrics
- Trying to monitor a non-existent Performance Monitor object
- Trying to monitor a non-existent WMI class
- Agent runs a command return code but does not return data
- UNKNOWN status displayed when trying to monitor a service
- No data shown for an attribute group
- Do not see situations in the console
- Data being sent by the agent does not look like it should in the Tivoli Enterprise Portal
- A re-installed agent appears to be configured in Managed Tivoli Monitoring Services
- Do not see situations in the navigator tree
- Agent crashes
- Agent configuration hangs
- Subnode name unique
- Installing 2 agents with the same script name
- Agent Configuration stops
- SNMP attribute group not collecting data reliably
- An agent monitoring SNMP V2 events, does not receive traps.
- Windows commands not running as expected
- CIM data provider stops collecting data from AIX OpenPegasus 2.6.1 CIM Server
- Collecting metrics through Windows APIs
- CIM data provider stops collecting data from the Solaris WBEM Server
- CIM data provider intermittently fails to collect data from Solaris 9 WBEM Server
- CIM data provider stops collecting data from Solaris 10 WBEM CIM_FileSystem class
- CIM data provider cannot collect data remotely from Solaris 10 WBEM server after applying Solaris Patches
- Local configuration JRE warning
- Password is not stored when locally configuring the agent on a Windows system
- Connecting to Microsoft SQL Server using integrated authentication is failing
- Microsoft SQL Server and JDBC driver compatibility errors with Java Runtime Environment (JRE) version 1.6
- JDBC connections are failing to find my JDBC driver
- Agent support files still exist after uninstalling
- Navigator display in the Tivoli Enterprise Portal shows a combination of old and new nodes or shows the wrong data when you click on a node

- Agent installation fails for agents built with Agent Builder V6.2.2 or earlier
- Installing the agent or application support fails with error message: KQZ0208E
- Ping attribute group on Windows slow to respond
- SDA (self describing agents), problem uploading support.
- Authorization failures when using SSH public key authentication
- Running the installIraAgentTEMS.sh results in an error on UNIX
- Agent with SSH to Windows does not run a command
- On Windows, a mapped drive to a Ping, Log file, JMX, or JDBC configuration file cannot be read
- Missing or unexpected data for a socket attribute group in the Tivoli Enterprise Portal
- HTTP data provider does not properly handle URLs that use the https protocol
- Testing an agent in Agent Builder, but not getting results.
- Tivoli Enterprise Monitoring Server does not start
- Script data providers that use SSH are failing
- An inbound ping is not reaching my agent on Windows 2012 Server.
- An inbound SSH connection from a remote script data source is not reaching my agent on Windows 2012 Server.
- How can I find the available parameters to perform silent command line configuration of an Agent Builder agent?

Table 50. Problems and solutions for agents

Problem	Solution
Program fails No data for a script provider or a command return code is unusable. You see the following in the trace log: `(46C44462.0000-184C:commandwithtimeout.cpp,278, "CommandWithTimeout::threadMain") *Error: Failure in call to CreateProcess() for script script1 not.bat Error The system cannot find the file specified.` This shows that it tried to run the command **`script1`** with the argument **`not.bat`**.	If you call a program with spaces in the name, use quotation marks around the name so that it is not parsed by the command interpreter. For example, `this is a test.bat argument` becomes: `"this is a test.bat" argument`

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Negative or wrong attribute value I created an agent that includes a number that I expect to be a large positive number, but I see a negative or incorrect number.	Tivoli Monitoring 6.2 uses 32-bit signed integers to represent numeric values. A 32-bit signed integer can display values from -2,147,483,648 to 2,147,483,647. The value that you are trying to display overflowed the 32-bit signed number. In many cases, the values are traced to the log file and replaced with enumerations that indicate the value exceeds maximum or minimum. Overflows can usually be handled by creating another attribute that scales the large one to a more reasonable value. For example, if the number represents the size of the disk in bytes, it is more useful to use megabytes or even gigabytes. Use the following procedure to convert the value: 1. Select the **Data Sources** tab in the Agent Builder. 2. Right-click the data source. 3. Select **New Derived Attribute...** 4. Choose a new name. For example, for Size, you can add units like Size_MB. 5. Add a description. 6. Select the appropriate data type. For example, Counter is correct for a total size. 7. Create the formula. • Select the attribute. For example, Size. • Convert the value appropriately. For instance, `Size/1048576` converts from bytes to megabytes You can now either leave the original attribute or hide it by selecting the attribute and clearing **Display attribute in the Tivoli Enterprise Portal**. Hide the original attribute if it is likely that it might overflow a 32-bit signed integer. Tivoli Monitoring V6.2.1 introduces 64-bit numeric attributes. Changing a 32-bit numeric attribute that is overflowing to a 64-bit value is a natural way to represent large numeric values.
JMX Notifications I want create a data source to receive notifications, but I cannot see an option to configure in this way.	When you use the browser, take note of the MBean information that is displayed in the browser panel for the MBean you are working with. If the MBean contains items in the Attributes tab, then you get an attribute group that contains those metrics. If the MBean contains items in the Notifications tab, you get an event attribute group that contains a standard set of metrics for a Notification object. If both tabs contain values, then you get attribute groups for each. If notifications are not defined for the MBean, the browser does not create the event attribute group. You can create the event attribute group manually by not clicking **Browse** to display the JMX browser. Instead, manually type in the Object Name pattern and click **Finish**. This action creates the same event attribute group that you get from the browser had notifications been detected. It also creates an attribute group to receive data with no attributes defined. This other attribute group can be deleted.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Locally configuring an agent fails on Windows An Agent Builder created agent that contains configuration properties requires Tivoli Monitoring to install the Java Runtime Environment. If the Monitoring Agent for Windows OS was installed by using the **tacmd createnode** command to create a Tivoli Monitoring v6.2 Fix Pack 1: Windows OS Agent, then the Java Runtime eEnvironment is not installed. On this system, you cannot locally configure any Agent Builder created agents that contain configuration properties.	The agent can still be configured remotely by using the Tivoli Enterprise Portal. The following solutions can be used to install the Java Runtime Environment that is needed by Tivoli Monitoring: • Install an IBM provided agent that requires the Java Runtime Environment. • Install the Monitoring Agent for Windows OS locally instead of using the **tacmd createnode** command. • Install the Tivoli Monitoring V6.2: Windows OS Agent by using the **tacmd createnode** command instead of the Tivoli Monitoring V6.2 Fix Pack1: Windows OS Agent.
Command line does not allow me to configure an agent I cannot use the **itmcmd config** command (or CandleConfig) to configure an agent that contains JMX, SNMP, or JDBC attribute groups.	You must have Tivoli Monitoring v6.2 Fix Pack 1 to configure an agent that contains JMX, SNMP, or JDBC attribute groups by using the **itmcmd config** (or CandleConfig) commands. You can use the GUI to configure the agent if an upgrade is not an option.
JMX monitors JMX Monitors are not working with the JBoss application server.	The following is needed to make JMX Monitors work. Copy the `connJboss-1.0.jar` file from *CANDLE_HOME*`/TMAITM6/k`*xx*`/jars/common/connectors/jboss` on Windows or *CANDLE_HOME*`/`*dynarch*`/k`*xx*`/jars/common/connectors/jboss` on UNIX to *JBoss_install*`/server/default/lib`. If you have a server configured other than the default server, the *default* part of the path is different for your server. If the JBoss server is running, it must be restarted after you copy this file.
JMX browser cannot connect to WebSphere 6.1 The JMX MBean browser cannot connect to WebSphere 6.1 with security enabled, when you use the SOAP connector protocol.	The browser can collect the MBean names by using the rmi connector protocol.
Error code for attribute group, but data is being returned The Performance Object status error code for my JMX attribute group is `ATTRIBUTE_ERROR`. Data is being returned for the MBeans, so what does this mean?	A JMX attribute group that has an error code set to `ATTRIBUTE_ERROR` in the performance object status attribute group has one or more attributes that cannot be collected. This error not only indicates a problem with one or more attributes, but it is a performance issue as well. To determine which attributes are having a problem, look for the exceptions in the JMX trace log file. The exceptions typically indicate the class path cannot locate a certain class or the attribute object cannot be serialized. When you see this error, the attributes must be collected individually from the MBean Server instead of collecting all attributes in one remote call. This collection method can significantly affect the performance of the agent.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
JMX data provider cannot connect to WebSphere 7.0 JMX data provider fails to connect to WebSphere 7.0 with security enabled, when you use the RMI connector.	An issue with the WebSphere Application Server. Upgrade to WebSphere 7.0.0.1 or later to resolve this connection issue.
JMX data provider cannot connect to Oracle WebLogic Server on AIX The JMX data provider fails to connect to the Oracle WebLogic Server when you run on AIX.	This connection problem is a limitation that is documented by Oracle WebLogic. Connect to Oracle WebLogic remotely from an operating system with a compatible Sun Java runtime environment. In this case, the agent must be configured to use the compatible Sun Java runtime environment instead of the IBM Java runtime environment.
Title bar not displaying the workspace name When you look at a generated agent through the Tivoli Enterprise Portal, the title bar displays *Nav_Node - TEPS_Hostname - UserID* rather than *Workspace_Name - TEPS_Hostname - UserID*.	When an agent is generated, workspaces are not created by default. If no workspace is defined for an attribute group, the Tivoli Enterprise Portal displays a generated workspace for the attribute group that shows all attributes in a table view. The title bar of the Tivoli Enterprise Portal displays the Navigation Node ID because the workspace name is not defined. You can click **File** > **Save** to save the generated workspace as the default workspace. A workspace name is now displayed in the title bar. For more information, see "Creating workspaces" on page 403.
The log file is not monitored properly If you build an agent with a log file data source, and all or part of the log file name comes from a Configuration Property, and that Configuration Property value contains a space, the log file is not monitored properly.	Enter the Windows short name for the file or path in the agent configuration. You can get the short name of any file or directory with the **DIR /X** command issued at a Windows command prompt.
Agents not translated Agents that are created by Agent Builder are not translated when shown on the Tivoli Enterprise Portal.	The Agent Builder does not build language packs. This means that the text that is displayed in the Tivoli Enterprise Portal is in the language you used when you built the agent.
RAS1 log errors I see the following in the RAS1 log for my agent. What does it mean? `(46C30EA0.0000-2180:getprocesscmdline.cpp,387, "GetProcessCmdLine") Unable to read the process environment block. ReadProcessMemory returned 0. (46C30EA0.0001-2180:getprocesscmdline.cpp,589, "getPIDCommandLine") Failed to get process command line, pid(4) error: Only part of a ReadProcessMemory or WriteProcessMemory request was completed.`	This error can occur on some systems for the process that represents the *system* in Windows. The process environment block is not available for this process. System is a special process and normally is not one you are monitoring. If all of the data in your Availability table is completed, then this error does not represent a problem. You can verify by checking the PID printed in the trace against the PIDs for the processes you are monitoring.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Log file data mismatch A record from the log file is not displayed on the Tivoli Enterprise Portal. Or the last entry or last few entries on the Tivoli Enterprise Portal do not match the contents of the log record.	Check for error messages in the trace file, for example, `HOSTNAME_81_k81agent_465c087e-01.log`. Look for the following trace entries: `(465C08D7.0000-2A4:logmonitorqueryclass.cpp,506,"LogMonitorQueryClass::setInstanceData")Agent metric count and UA metric count do no match! Agent count=<7>, UA count=<6>.(465C08D7.0001-2A4:logmonitorqueryclass.cpp,561,"LogMonitorQueryClass::setInstanceData")UA ran out of values for agent metric! Agent name=<rest>`. The `Agent count` indicates the number of attributes that are expected to be completed from a log file record. The `UA count` is the number of records that the data provider parsed from the log record. A mismatch means that some of the attributes might be parsed from the log record. However, others might not because there was less data in the log record than expected.
Installed agent does not show up An installed agent does not show up in the Tivoli Enterprise Monitoring Services utility.	Select **View > Refresh** from the Manage Tivoli Enterprise Monitoring Services window.
Situations not showing up I expect the situations to be true but they are not displayed as true. Or, I expect my node to display the little red circle that indicates that situations are true. I checked and the data exceeded the threshold. Why do I not see the situation on my node in the navigation tree.	Some of the associations between nodes and situations are loaded when the Tivoli Enterprise Portal starts. Restart the Tivoli Enterprise Portal.
Queries not showing up The new queries are not showing up.	Install the agent and then recycle the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal.
Error states the **`startagent`** command failed An error on the Tivoli Enterprise Portal says that the **C:\IBM\ITM\InstallITM\Batch\kincli -startagent -akxx command failed**.	After you install Tivoli Monitoring on a Windows endpoint, the machine must be rebooted before it can successfully be a target of remote deployment, or the start, stop, and remove functions that are available on the Tivoli Enterprise Portal. The reason for this is that the Tivoli Monitoring installation changes the PATH of the system to include some DLLs that must be found for those functions to work. However, the services do not pick up those changes until the machine reboots.
When you run command return code, the return code is not accurately shown When you run the Windows command return code to run a command and analyze the return code, and the command is a `.bat` or `.cmd` script, the return code is not accurately shown. The script exits with something like `exit /b 2`.	To get this value, surround the target script with a script that calls the target script and then runs`"exit %ERRORLEVEL%"`.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Attribute group data is missing from Tivoli Enterprise Portal I built an attribute group by using a script source and I do not see any data in my Tivoli Enterprise Portal.	Read the log file and look for text that looks like: `(46543D0F.0019-1A0C:shellqueryclass.cpp,331, "internalCollectData") Missing metrics. Skipping row. expected 15 tokens, parsed 2. Input:<FINDSTR>, separator:<:>` This text indicates that your script returned data that did not match the defined format (in this case, items that are separated by a colon). It was attempting to parse the string that is contained within the first <> pair. Fix the script to return data in the correct format. You can test the result by editing the script in the agent directory because it calls the script each time it tries to collect data.
Changing version number causes errors If you have an agent that you created with the Agent Builder, and you modify it and change the version number, if you attempt to deploy the agent remotely with the new version, you get an error that says **KFWITM291E An agent configuration schema was not found.**	The Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server support must be installed again, even if you changed nothing but the version number. This error is caused by certain configuration files that contain the version number. You receive failures if the files with the new version number are not present.
Changing product code, company identifier, or agent identifier causes problems You have an agent that you created with the Agent Builder, and you modify it and change the product code, company identifier, or agent identifier after you created workspaces or situations.	All the situations and workspaces must be re-created.
A script provider behaves oddly when edited For instance, you edit the script and add a sleep 30 to simulate a timeout. The timeout occurred as expected during the next refresh of the group. Then you take out the sleep 30 and refresh the group again. The calculated values are now all set to 0.	This behavior is because of the previous data point being lost.
Core memory dump after you edit the `.ref` file Core memory dump with a seemingly innocent modification to the `.ref` file. For example, splitting attributes of an element in the `.ref` file onto different lines still produces valid XML, but the agent coredumps.	Do not edit this file.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
The Tivoli Enterprise Portal does not show any columns or column headers Navigator groups in an agent do not show any columns or column headers. Instead, an error is displayed in the view: `KFWITM220E Request failed during execution.` In addition to navigator groups that the user created, this error is also seen with internally generated navigator groups such as the Availability and JMX Monitors navigator groups.	When you define a navigator item, there can be more than one attribute group with a node in the navigation tree. When there is more than one attribute group, you must assign a query to a workspace. After you assign a query, you can see the data.
Bad string values that are collected from SNMP are displayed in attribute groups	This behavior happens for `OCTETSTRING` types where the value is binary data and not strings. Binary data is not translated for display. It is forwarded as the binary data.

No data in the Tivoli Enterprise Portal
 	
I have a script data source and I am not getting any data in the Tivoli Enterprise Portal.
	Change the setting for the trace to: `ERROR (UNIT:shell ALL)`
	
	If you see a trace like the following, your script is not returning the data in the expected format.
	`(45FEFE30.001F-D14:shellqueryclass.cpp,329, "internalCollectData") Missing metrics. Skipping row`
	`(45FEFE30.0044-D14:shellqueryclass.cpp,329, "internalCollectData") Missing metrics. Skipping row`

Cannot remove agent from the Tivoli Enterprise Portal When you try to remove an agent, you see that the agent is still listed in the desktop navigation view, but it is not available.	This problem occurs if the agent remains in the managed system list in IBM Tivoli Monitoring. Perform the following steps: 1. Select the **Enterprise** node, in the physical tree in the Tivoli Enterprise Portal. 2. Right-click **Workspace** > **Managed System Status**. 3. Select the entry for the agent. **Note:** The name is displayed in the second column. 4. Right-click and select **Clear offline entry**. 5. The element is removed from the tree in the Tivoli Enterprise Portal.

Table 50. Problems and solutions for agents (continued)

Problem	Solution

Service or process monitoring returns zeros for metrics
 	
When you create an agent to monitor Service availability, the column values are not correct. All metrics, including the Process ID, are zero.
 	
For example:
• For Services, you might receive the following information:
`Status=UNKNOWN`
and zeros for the availability metrics.
• For Processes, you might receive the following information:
`Status= PROCESS_DATA_NOT_AVAILABLE`
and zeros for the availability metrics except for Process ID.
	Ensure that the agent is being run under an Administrator ID.

Trying to monitor a non-existent Performance Monitor object The agent runs and tries to monitor a Performance Monitor object, but a status of `INACTIVE` is displayed along with the error code `OBJECT_NOT_FOUND`.	The Performance Monitor object does not exist in the system. Specify a Performance Monitor object that exists in the system, or install the monitored application that creates the Performance Monitor object.
Trying to monitor a non-existent WMI class The agent runs and tries to monitor a WMI class, but a status of `ACTIVE` is displayed along with the error code `NO_INSTANCES_RETURNED`.	The WMI class was not found and does not exist in the system. WMI collection displays the error message in the log each time the agent tries to collect the data. Specify a WMI class that exists in the system, or install the monitored application.

Agent runs a command return code but does not return data
 	
When your agent runs a command return code script, the end of the log might display the information such as the line in the following example:
`FACWIN5B_test_datasource_442816d1-02.log:(4428171C.30A2-1A50:applicationpinglistelement.cpp,100,"isApplicationAvailable")`
`Running Application Ping Command hello.bat`
 	
If this type of information is in the log and is the last to be displayed, the command return code script did not end normally. The agent is locked up and cannot return data.
	Investigate the trace log. An entry similar to this example indicates that the command return code has not run to completion.
	
	Amend your command return code so that it is completed in a reasonable amount of time (for example, no more than 10 seconds).

UNKNOWN status is displayed when you try to monitor a service You try to monitor a service but the message `Status=UNKNOWN` is displayed, even though you already verified that the agent has Administrator privileges.	The service is not installed on the system. When the view is being built for the Availability table, rows where the status is `Unknown` must be filtered out to prevent confusion. Filtering rows out is especially needed when an application is composed of a set of optional services. The lack of a service is not an error in this case; it is normal.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
No data is shown for an attribute group You try to monitor a data source but no data is shown from an attribute group that collects data.	If the systems used for developing and testing the agent are different, the WMI classes and the Performance Monitor objects can be different too. Develop and test the agent on the same version of Windows and the same version of the monitored application that you want to manage.
Do not see situations in the console I installed my agent, situations, and workspaces. I configured the agent and started it. I see it in the Tivoli Enterprise Portal, but I do not see the situations in the navigator tree or the Situation Event Console. I checked, and the situations are associated with the correct nodes.	Restart the Tivoli Enterprise Portal.
Data being sent by the agent does not look like it can be in the Tivoli Enterprise Portal The data looks like the attributes are not being parsed in the correct places.	Reinstalling Tivoli Monitoring is the safest way to ensure that you do not introduce incompatibilities between the phases of your development. **Note:** The agent builder includes features to prevent you from introducing these types of changes into later versions of your agent. These features ensure that you do not have this type of problem as you build and deploy updates to an agent.
The status for a reinstalled agent indicates that the agent is configured in Managed Tivoli Monitoring Services When you develop an agent, you likely generate, install, test, update the agent and then generate, install and test again. When you follow these actions, the agent status indicates that the agent is configured in Managed Tivoli Monitoring Services after the agent is reinstalled.	If you modified the agent configuration, then reconfigure the agent, and restart.
Do not see situations in the navigator tree I installed my agent, situations, and workspaces. I configured the agent and started it. I see it in the Tivoli Enterprise Portal, but I do not see the situations in the navigator tree. They are not associated with the correct node. (I right-click on the node and select the situations, but they do not show in the default list.)	When you create the situation on a live system, it is automatically distributed to the agent by using the host name. This host name is not available in every environment. So you must distribute the situation to a generic managed system list that exists when the agent is installed. Distribute the situations to the CUSTOM_*app_name*00 Managed System List, and reimport the situations. Then rebuild the Solution Installer image, and reinstall the Tivoli Enterprise Monitoring Server support for the agent.

Agent fails
 	
One reason this behavior can occur is that the `ICCRTE_DIR` might not be set in the `ENV` file on Windows systems.
	`cygwin` might be installed so that `ICCRTE_DIR` is set as:
	`ICCRTE_DIR=`
	
	The `Kxxinstall.log` has the following text that shows the problem:
	`find: ICCRTE_DIR=: No such file or directory`
	
	You can fix this problem by taking `cygwin` out of the path and setting `ICCRTE_DIR` manually.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Subnode names Subnodes from different agent instances have the same managed system name.	The Managed System Name for a subnode consists of *2 letter agent Product Code:first 24 characters of the Subnode ID:3 letter Subnode Type* The first 24 characters of subnode IDs must be unique for all instances of the subnode type in the IBM Tivoli Monitoring installation. The agent automatically adds a prefix `PC` to prevent the subnodes from colliding with subnodes created by other agents. It automatically appends the subnode type to prevent collisions with other subnode types in the same agent. It uses the first 24 characters of the Subnode ID (which you control) as the final token. No part of the Agent Instance or the Agent Host System is used in constructing the Subnode Managed System Name. By using the same Subnode ID in 2 instances of the Agent, the Managed System Names collide. This name collision happens even for Agent Instances hosted on separate systems. The result is the subnodes do not function.
Installing 2 agents with the same script name On Windows operating systems, when two different agents are installed on the same Tivoli Enterprise Monitoring Agent, and they both have a script with the same name, the script from the last agent that was installed or deployed overwrites any existing scripts of the same name. Scripts are copied into the `instdir\tmaitm6` directory without a warning.	There is no solution. The files are copied into the TEMA directory so that they are all in a consistent place and easily accessed by the agent and each other. The situation is the same for Windows and UNIX systems.
Agent Configuration stops When you deploy or configure an agent that contains subnodes and requires a minimum IBM Tivoli Monitoring version of 6.2.1 on a Tivoli Enterprise Monitoring Agent that has a IBM Tivoli Monitoring version earlier than 6.2.1, configuration might stop.	1. Upgrade the OS agent on the target system to the prerequisite for the agent, version 6.2.1 or later. 2. Change the **Minimum ITM version** field to 6.2 and rebuild the agent, or upgrade to Tivoli Monitoring 6.2.1 or later. The subnode configuration parameters do not have User Configurable Initial Values, although default values might still be assigned when you build the agent. You also cannot override configuration parameters that are not explicitly listed in the subnode configuration overrides section. This issue can be addressed in 2 ways: • Configure multiple instances of the agent. Each instance can provide a different set of values for parameters that are not included in the subnode overrides. • Rebuild the agent so that the Subnode Configuration Overrides contain all parameters that might require to be overridden.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
SNMP attribute group not collecting data reliably Data is collected intermittently or not at all. The SNMP version and credentials are configured correctly The Performance Object Status Error Code for the attribute group shows `NO RESPONSE RECEIVED`. **Note:** This issue applies to SNMP attribute groups, so the Object Type in the Performance Object Status table is SNMP. The agent trace file shows the following message: `Timeout occurred. No response from agent.` Here is a sample entry: `(48A18C71.000A-12:snmpqueryclass.cpp,1714,"internalCollectData") Timeout occurred. No response from agent.`	The IBM Tivoli Monitoring SNMP data provider is multithreaded to enhance performance. The SNMP data source that is being monitored might not be able to respond to multiple incoming requests in a timely manner. There are tuning options that can improve reliability of data collections: **Reduce the thread pool size** The default thread pool size is 15. Try reducing the size to 5. This setting can be adjusted in the agent ENV file by setting the CDP_DP_THREAD_POOL_SIZE environment variable. **Increase the SNMP Response timeout** The default SNMP timeout is 2 seconds. Try increasing the timeout to 6 seconds. This setting can be adjusted in the agent ENV file by setting the CDP_SNMP_RESPONSE_TIMEOUT environment variable. **Reduce the number of SNMP retry attempts** The default number of SNMP retry attempts is 2. Try reducing the size to 1. This setting can be adjusted in the agent ENV file by setting the CDP_SNMP_MAX_RETRIES environment variable. For more information about setting agent environment variables, see ("Environment variables" on page 435).
An agent monitoring SNMP V2 events, does not receive traps. The MIB for the monitored device indicates the enterprise OID for the trap ends with a `".0"`. The received SNMP V2 traps contain the snmpTrapOID.0 varbind with a value of `enterprise OID.specific type`. Example: The received snmpTrapOID is: `1.3.6.1.4.1.1302.3.8.10.2.1.`**`0.6`** Enterprise OID defined in the MIB for this trap: `1.3.6.1.4.1.1302.3.8.10.2.1.`**`0`**	The Agent Builder SNMP event receiver follows RFC 2089 so it can handle SNMP V1 and V2 traps. This RFC states that the last token of the snmpTrapOID.0 varbind is the specific trap field. This token and the preceding token if it is a 0 is removed from snmpTrapOID.0 to create the enterprise field. If your MIB includes enterprise OIDs that end with a `.0` but receives traps with the last two tokens that are removed from the snmpTrapOID, varbind does not match the enterprise OID in the MIB. To fix, you must make the following modification to your agent: • Edit the agent XML and find the lines that look like: – `global_snmp_event_settings_for_group oids="1.3.6.1.4.1.1302.3.8.10.2.1.`**`0`**`-6"` • Delete the `.0` (including the dot), so it now looks like: – `global_snmp_event_settings_for_group oids="1.3.6.1.4.1.1302.3.8.10.2.1-6"` • Finally regenerate and reinstall the agent.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Windows commands not running as expected Take Action command does not run Windows command as expected. Availability Functionality Test does not run Windows command as expected. Script data source does not run Windows command as expected.	The mechanism that is used to start a process in Take Action commands, and agent runtime data providers is the native Windows process management API, CreateProcess(). With this command you can start processes that are implemented in `.bat`, `.cmd`, or `.exe` files. Windows implements several common command functions as internal commands in the shell and not as programs. These include commands like `echo` and `dir` (common commands that are used to test script execution). Since these commands are not programs, they cannot be started with createProcess(). To call these commands, create a `.bat` or `.cmd` file that contains the commands. It is possible to start the command processor and provide the built-in command as an argument, as shown in the following example: `cmd /c "echo val1;1;val2;2"` Remember that you are collecting the return code from `cmd`, not `echo` in this example.
CIM data provider stops collecting data from AIX OpenPegasus 2.6.1 CIM Server. Data collection resumes if the AIX CIM Server is stopped and restarted.	This issue is resolved in IBM Pegasus CIM Server V2.6.1.35 You can display the current version of the IBM Pegasus CIM Server file sets by using `lslpp -l sysmgt.pegasus.cimserver.rte` Upgrade the IBM Pegasus CIM Server to 2.6.1.35 or later For current details, see the AIX Common Information Model overview
Collecting metrics through the Windows APIs	To collect metrics through the Windows APIs, the agent must be hosted on a Windows operating system. Remote registry administration must be enabled on the remote systems.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
CIM data provider stops collecting data from the Solaris WBEM Server.	Several Solaris patches are required to achieve a stable Solaris WBEM CIM Server. Minimum versions that are required are: Solaris 9:Patch Synopsis • 112945-46 WBEM Patch (Sparc) • 114193-36 WBEM Patch (X86) • 116807-02 SMC Security Patch (Sparc) • 116808-02 SMC Security Patch (X86) • 114501-01 DRM Provider Patch (Sparc) • 114502-01 DRM Provider Patch (X86) • 114503-14 User Manager (VUserMgr.jar) Patch (Sparc) • 114504-14 User Manager (VUserMgr.jar) Patch (X86) • 114711-03 Disk Manager (VDiskMgr.jar) Patch (Sparc) • 114712-03 Disk Manager (VDiskMgr.jar) Patch (X86) • 112943-09 Volume Management (VVolMgr.jar) Patch (Sparc) [DiskSuite/SVM] • 114192-06 Volume Management (VVolMgr.jar) Patch (X86) [DiskSuite/SVM] Solaris 10:Patch Synopsis • 119313-22 WBEM Patch (Sparc) • 119314-23 WBEM Patch (X86) • 121308-14 Console Patch (Sparc) • 121309-14 Console Patch (X86) • 119315-14 Solaris Management Applications Patch (Sparc) • 119316-14 Solaris Management Applications Patch (X86) • 124188-02 Trusted Solaris Attributes Patch (Sparc) • 124189-02 Trusted Solaris Attributes Patch (X86)
CIM data provider intermittently fails to collect data from Solaris 9 WBEM Server after all patches are applied.	This issue is reported to Sun Microsystems and is a vendor limitation. If you are running a typical multi-threaded agent against few remote systems, the agent might send all of the requests concurrently to a single WBEM CIM Server. The CIM Server might not handle all requests. In a typical environment, with multiple remote CIM Servers, the requests are spread out across systems and this problem does not occur. Set the following environment variable in the agent `env` or `inifile`: `CDP_DP_LOCK_CIM_ACCESS=YES` When `CDP_DP_LOCK_CIM_ACCESS=YES` is set, the agent serializes the CIM requests that it sends. The lock covers the lifespan of the request to the other system. When the request is received, the agent unlocks and then processes the result. **Note:** This flag can have a negative effect on the scale of the agent. Use this flag only when it is necessary.

Table 50. Problems and solutions for agents (continued)

Problem	Solution

CIM data provider stops collecting data from Solaris 10 WBEM CIM_FileSystem class.
 	
The Performance Object Status for this attribute group reports GENERAL ERROR.
	This issue is reported to SUN and is a Vendor Limitation. There is an error in the Solaris WBEM CIMOM. Requests for the CIM_FileSystem Class log this error in the WBEM log:
	`nfs_mntinfo: Can't access mnttab.`
	`Too many open files`
	`CIM_ERR_FAILED:`
	`nfs_get_mount_list Failed.`
	
	The workaround is to collect data from the `Solaris_LocalFileSystem` class, or to recycle the WBEM CIM Server.

CIM data provider cannot collect data remotely from Solaris 10 WBEM server after you apply Solaris Patches: • `121308 Console Patch (Sparc)` • `121309 Console Patch (X86)`	If you are running `Solaris 10 6/06` or earlier, you must modify your WBEM configuration file to allow remote connections after you install patch `121308-XX`. See the following SunSolve document for detailed information: Document ID: 211275 Title: *Solaris[TM] 10 WBEM only listens to port 898 on localhost*
Local configuration JRE warning Locally configuring an agent displays this warning: `Kincfgexit Java Runtime Environment was not detected! Extended agent configuration is disabled - if remote configuration for the agent is supported, complete the process using the Tivoli Enterprise Portal.`	The Monitoring Agent for Windows OS was deployed by using **`tacmd createnode`**, so no Java was installed on the local system. The agent is configured enough to allow it to connect to the Tivoli Enterprise Monitoring Server by using the default agent configuration parameters. Start the agent to allow it to connect to the Tivoli Enterprise Monitoring Server. You can then complete the configuration by using the Tivoli Enterprise Portal. Optionally, you can install a supported JRE locally to configure the agent by using the Manage Tivoli Enterprise Monitoring Services interface. This problem is known to occur on 6.2.2 releases. Also, use the steps in the following problem: Password is not stored when locally configuring the agent on a Windows system.

Password is not stored when locally configuring the agent on a Windows system	When the Java Runtime Environment is installed by Tivoli Monitoring, a special patch is applied to encrypt the password. When the JRE is manually installed, this patch is not applied. If you manually installed the JRE, do the following steps so that passwords are correctly encrypted:
	1. Copy the following 3 files from the `%CANDLE_HOME%\InstallITM` directory to the `C:\Program Files\IBM\Java50\jre\lib\security` directory:
	`java.security`
	`local_policy.jar`
	`US_export_policy.jar`
	2. Reconfigure the agent to correctly encrypt the password and store it in the configuration file.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Connecting to Microsoft SQL Server by using integrated authentication is failing	You can connect to a Microsoft SQL Server without a user ID or password by using integrated authentication from Microsoft. This only works on Microsoft Windows operating systems and requires that you have a JDBC driver that supports integrated authentication. Integrated authentication can be used for JDBC connections in the Agent Builder browser or in the agent run time. • To connect by using the JDBC browser, make sure the JDBC driver authentication `dll` file is present somewhere in the system path of the computer. Start Agent Builder after the `dll` file is in the path. After Agent Builder is running, you can use the JDBC browser without specifying a user ID or password in the connection properties dialog box to connect to a Microsoft SQL Server by using integrated authentication. Integrated authentication uses the current user's account to verify the authentication. • The agent run time can connect to a Microsoft SQL Server by using integrated authentication by doing the following steps: – Make sure that the JDBC user name and JDBC password configuration properties are changed so the Required check box is not selected in the Runtime Configuration editor tab. – The authentication `dll` file that comes with the JDBC driver must be in the path of the system. Alternatively when you configure the agent, include a JVM argument in Java properties that specifies the location of the user authentication `dll` file. for example: `-Djava.lib.path=E:\sqljdbc_1.2\enu\auth\x86` – When you configure the agent, do not specify a user name or password in the configuration properties for JDBC.
Microsoft SQL Server and JDBC driver compatibility errors with Java Runtime Environment (JRE) version 1.6	Microsoft requires you to use the JDBC 4.0 driver JAR file `sqljdbc4.jar` when you use Java Runtime Environment (JRE) version 1.6. This driver must be used with any JRE version 1.6 or newer. If you use a JRE version 1.5, you must use the older JDBC JAR file`sqljdbc.jar`.
JDBC connections are failing to find my JDBC driver JDBC connections are failing to find my JDBC driver that was added to the class path.	Make sure that the JDBC driver is compatible with the JRE that you are using. JDBC 4.0 drivers are compiled with Java 1.6. To use a JDBC 4.0 driver, the JRE that you configure the agent to use must be at least Java 1.6. Agent Builder uses Java 1.5, so you must use a JDBC driver that is compatible with Java 1.5 to use the JDBC browser in Agent Builder.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Agent support files remain after you uninstall it. After you uninstall an agent, the agent support files remain on the system.	The support files for the Tivoli Enterprise Monitoring Server and the Tivoli Enterprise Portal are not removed by uninstalling the agent. 1. Remove Tivoli Enterprise Monitoring Server Application support by completing the following steps: a. Use Manage Tivoli Enterprise Monitoring Services. b. Select **Tivoli Enterprise Monitoring Server**. c. Right-click and select **Advanced**. d. Select **Remove TEMS application support**. e. Select the agent to remove its application support. 2. Remove the agent from the Tivoli Enterprise Portal by using the following procedure: a. Ensure that your Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server are up and running. b. Log on to your Tivoli Enterprise Portal client. c. From the Tivoli Enterprise Portal client Physical Navigator views, right-click **Enterprise** and select **Workspace** > **Managed System Status**. The Managed System Status workspace is displayed. d. Select all of the IBM Tivoli Managed Systems for your agent. e. Right-click and select **Clear off-line entry**, which clears all of the entries from that table.
Navigator display in the Tivoli Enterprise Portal shows a combination of old and new nodes or shows the wrong data when you click a node When you remove attribute groups, rename them, or reorganize existing groups into or out of navigator groups, the display in the Tivoli Enterprise Portal might show a combination of old and new nodes.	If you are using Agent Builder 6.2.1.2 or later, resolve this issue by restarting the Tivoli Enterprise Portal Server and then, restart the Tivoli Enterprise Portal.
Agent installation fails for agents that are built with Agent Builder V6.2.2 or earlier For agents built with Agent Builder V6.2.2 or earlier, installing the agent might prevent other agents from being successfully installed. If the TECLIB directory does not exist in TMAITM6 when the Agent Builder agent is installed, then a file is created with the name TECLIB. When subsequent agents attempt to create files in the TECLIB directory, the agent installation fails.	Perform the following steps: 1. Rename the file from `TECLIB` to `kXX.baroc`, where *XX* is the two-character product code for the agent to which the file belongs. 2. Create a directory that is called `TECLIB` and copy the renamed `kXX.baroc` file into the newly created `TECLIB` directory. 3. Try the installation, or deployment of the agent that failed, again.
Installing the agent or application support fails with error message: `KQZ0208E` When you install the agent or application support and specifying a password with special characters, the login to the monitoring server fails with error message: `KQZ0208E`. The specified user name or password is incorrect.	The password cannot include special characters.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
SDA (*self describing agents*), Problem uploading support. I am developing an agent by using Agent Builder and I am using SDA (*self describing agents*). It uploaded my support one time, but it does not seem to be doing it anymore. What can I do?	You can increment the agent version number. or you can use `tacmd deleteappinstallrec -t XX -v version` (*XX* is the two-character product code for the agent. *version* is an 8 digit long number, such as 06230000). This command deletes the tag that tells the system it is uploaded.
Ping attribute group on Windows slow to respond It is a long time before I start getting data back from my ping attribute group on Windows.	Name resolution on Windows can take about 5 seconds to timeout if an IP address does not resolve to a host name. If you defined several devices that have this name resolution issue the ping attribute group takes longer than expected to initialize. To prevent this problem, make sure the devices you define can be resolved in DNS or by using entries in your local host file.
Authorization failures when you use SSH public key authentication	Check the agent log file. An error message similar to the following one, indicates that the agent process cannot open the private key file. This error might happen on Windows systems since the agent runs as a service that might not be able to access another users private key file. Resolve this problem by making sure the agent process runs as a user that can read the private key file. The following is an example of an error message when the agent process cannot open the private key file: `(4C6D417B.0048-1230:userauth.c,631,"file_read_privatekey") -16 - Unable to initialize private key from file`

Running the `installIraAgentTEMS.sh` results in an error on UNIX
 	
Running the `installIraAgentTEMS.sh` results in the following error on UNIX: `Installation failed. Please see the log in /opt/IBM/ITM/logs/`*`product_code`*`_TEMSInstall.log`
 	
The end of the *`product_code`*`_TEMSInstall.log` contains the following lines:
`. . .`
`+ rm -rf /tmp/k4s`
`rm: 0653-610 Cannot remove current directory /tmp \`
`/k4s.`
`+ [ -d /opt/IBM/ITM//tables/cicatrsq/TECLIB ]`
`+ cp ./ira/agent/common/k4s.baroc /opt/IBM/ITM// \`
`tables/cicatrsq/TECLIB/k4s.baroc`
`cp: ./ira/agent/common/k4s.baroc: A file or \`
`directory in the path name does not exist.`
`+ fail`
`~`
	Extract your agent package into a directory other than `/tmp/`*`product_code`*, where *`product_code`* is in lowercase.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
Agent with SSH to Windows does not run a command An agent with SSH to Windows, does not run a command, or, the agent runs the command with SSH to Windows with no effect.	To run a remote command to a Windows host, you must have a Linux like shell environment installed. Cygwin is an example of a Linux like shell environment. To verify whether a shell environment exists, SSH or log on to the remote host and enter the command: `PATH=$PATH:. <command>` If the command runs, then a shell environment exists.
On Windows, a mapped drive to a Ping, log file, JMX, or JDBC configuration file cannot be read On Windows, a mapped drive to a Ping, log file, JMX, or JDBC configuration file cannot be read. The agent runs as a service and cannot see the mapped drives.	On Windows, do not use a mapped drive to store files that are required by the agent.
Missing or unexpected data for a socket attribute group in the Tivoli Enterprise Portal There is missing or unexpected data for a socket attribute group in the Tivoli Enterprise Portal.	Check the agent log if there are missing rows of data for a socket attribute group or if the data is not as expected. If there is missing or unexpected data, check the log even if the Performance Object Status for the attribute group displays `NO_ERROR`, as `NO_ERROR` is displayed if any valid rows were returned.
The HTTP data provider does not properly handle URLs that use the https protocol This problem occurs when you use an IBM Java run time on Solaris or HP-UX systems. You know that you have this problem if the Tivoli Enterprise Portal shows `http://` before each of your `https` URLs. For example, http://https://website.ibm.com	This problem can be solved in one of 2 ways. One solution is to add the following parameter to the `JVM arguments` runtime configuration property: `-Djava.protocol.handler.pkgs=com.ibm.net.ssl.www2.protocol` Another solution is to use a Java run time that was not provided by IBM if you have one available on the system.
I am testing my agent in Agent Builder and I do not seem to be getting any results. Performance Object Status shows `Timed Out` (and possibly some other values that indicate there might be a problem)	Sometimes the trace settings can slow the agent down in the test mode enough that data is not visible in the builder. This problem is most common in data sources like SOAP, JMX, and JDBC that use a Java provider. Lower your `Java Trace Level` setting by choosing a value higher in the list. Usually agents work with a trace level of `DEBUG_MID`. Sometimes it is necessary to set the trace level to `ERROR` (which is the value that is used when the agent is installed).
The Tivoli Enterprise Monitoring Server does not start and I receive a message that the number of installation packages exceeds the maximum of 512.	Install an uninitialized `QA1CDSCA` definition file that causes the Tivoli Enterprise Monitoring Server to extend the space that is used for application definitions. For more information about obtaining and installing the file, see Installed packages exceed the 512 maximum
I have an agent that is built with an Agent Builder version before 6.3. Script data providers that use SSH are failing with messages similar to this: `(5102BB3C.002C-B14:sshlibinterface.cpp,487, "Ssh_Interface_load") Failed to load the dynamic library k03ssh.dll : return code = 126 (5102BB3C.002D-B14:sshlibinterface.cpp,508, "Ssh_Interface_load") could not create libHandle (5102BB3C.002E-B14:shellqueryclass.cpp,247, "ShellQueryClass") Error initializing SSH Connector class`	Rebuild the agent with Agent Builder version 6.3 or later.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
On a Windows Server 2012 system, for an agent with a Internet Control Message Protocol (ICMP) data source, an outbound ping works but an inbound ping fails.	You must enable the server to respond to ping. Use the following procedure: 1. Open the Windows **Control Panel**. 2. Select **System and Security** > **Windows Firewall** > **Advanced Settings** The Windows Firewall with Advanced Security window opens. 3. Click **Inbound Rules**. Another Windows Firewall with Advanced Security window opens. 4. In the Inbound Rules list scroll to **File and Printer Sharing (Echo Request - ICMPv4-In)**. 5. Right-click **File and Printer Sharing (Echo Request - ICMPv4-In)** and select **Enable Rule**. 6. Repeat for **File and Printer Sharing (Echo Request - ICMPv6-In)**. Inbound pings to the server from other systems are now enabled.
On a Windows Server 2012 system, for an agent with a remote script data source, the inbound SSH connection fails.	You must create an inbound rule for SSH connection. Use the following procedure: 1. Open the Windows **Control Panel**. 2. Select **System and Security** > **Windows Firewall** > **Advanced Settings** The Windows Firewall with Advanced Security window opens. 3. Click **Inbound Rules**. Another Windows Firewall with Advanced Security window opens. 4. Right-click **Inbound Rules** and select **New Rule**. The New Inbound Rule Wizard window opens. 5. Select **Port** and click **Next**. 6. Select **TCP** and **Specific local**. 7. Enter 22 in the **Specific local** field. 8. Select **Allow the connection** and click **Next**. 9. In this window, you can choose different domains that you want the rule to apply to. The default is that all domains are selected. When you make your selection, click **Next**. 10. In this window, enter `SSH Connection`, or your preferred name for the connection. You can also optionally enter a description of the connection. When you are finished, click **Finish**. An **Inbound Rule** named **SSH Connection** is created in the Inbound Rules list and it is enabled. The rule allows inbound SSH connections on TCP port 22.

Table 50. Problems and solutions for agents (continued)

Problem	Solution
How can I find the available parameters to perform silent command line configuration of an Agent Builder agent?	1. Create a file named `silent_config_agent.txt` 2. Copy the configuration variables from the *XX*`_dd.properties` file into the `silent_config_agent.txt` file and complete the variable values. 3. Run the following command to configure the agent: `itmcmd config -p /tmp/silent_config_agent.txt -o` *MyInstance* `-A` *XX* **Note:** • *XX* is the 2 character short name you specify when creating the Agent Builder agent. • *MyInstance* is the instance name of the agent. • The file *XX*`_dd.properties` can be found in the top level k*xx* directory of the agent installation tar or zip file.

Troubleshooting: Report Creation

A table that shows problems that might occur when you create reports:

- Cognos data model for an agent loads into Framework Manager but testing a table causes a table or view not found error
- Testing any of the tables in the IBM_TRAM schema in Framework Manager results in an error on DB2
- When running on Windows, testing an Oracle data source in Tivoli Common Reporting fails
- The ManagedSystem dimension can not be utilized when creating reports for an agent
- Warnings when Model Advisor is run against the Cognos data model

Table 51. Problems and solutions for Report Creation

Problem	Solution
Cognos data model for an agent is loaded into Framework Manager but testing a table causes a table or view not found error	This problem can have 2 different causes and solutions: • The agent's tables do not exist in the Tivoli Data Warehouse database. Install and configure the agent, start historical collections, and configure summarization. Allow time for the agent to export data to the warehouse proxy and for the Summarization and Pruning agent to summarize data. • The schema name that is used in the data source of the data model does not match the schema that is used in the Tivoli Data Warehouse database. Change the schema value in the data source to match the schema used.

Table 51. Problems and solutions for Report Creation (continued)

Problem	Solution
Testing any of the tables in the IBM_TRAM schema in Framework Manager results in an error on DB2 Testing any of the tables in the IBM_TRAM schema in Framework Manager results in following error on DB2: `This query contains an error and can not be executed. [IBM][CLI Driver][DB2/AIX64] SQL0551N "ITMUser" does not have the required authorization or privilege to perform operation "SELECT" an object "IBM_TRAM.ComputerSystem".SQLSTATE=42501`	Grant select permission on all the Tivoli Reporting and Analytics Model tables. Change <IBM_TRAM> to the schema used for Tivoli Reporting and Analytics Model and <Tivoli Data Warehouse user ID> to the Tivoli Data Warehouse user ID. 1. Connect to the Tivoli Data Warehouse as a user with SYSDBA privileges. 2. Issue the following grants: a. `Grant select on <IBM_TRAM>."ComputerSystem" to <Tivoli Data Warehouse user ID>` b. `Grant select on <IBM_TRAM>.MONTH _LOOKUP to <Tivoli Data Warehouse user ID>` c. `Grant select on <IBM_TRAM>.TIMEZONE_ DIMENSION to <Tivoli Data Warehouse user ID>` d. `Grant select on <IBM_TRAM>.TIME_DIMENSION to <Tivoli Data Warehouse user ID>` e. `Grant select on <IBM_TRAM>. WEEKDAY_LOOKUP to <Tivoli Data Warehouse user ID>` f. `Grant execute on procedure <IBM_TRAM>. CREATE_TIME_DIMENSION to <Tivoli Data Warehouse user ID>`
When you run on Windows, testing an Oracle data source in Tivoli Common Reporting fails When you run on Windows, testing an Oracle data source in Tivoli Common Reporting fails with an error similar to: `QE-DEF-0285 The logon failed. QE-DEF-0325 The logon failed for the following reason: RQP-DEF-0068 Unable to connect at least one database during a multi-database attach to 1 database(s) in: testDataSourceConnection`	The Oracle client libraries are not in the system PATH environment variable. Edit the PATH environment and add the directory that contains the Oracle client DLL files. Restart the Tivoli Common Reporting server.
The `ManagedSystem` dimension cannot be used when you create reports for an agent I am not able to use the `ManagedSystem` dimension when I create reports for my agent.	Perform the following checks: • Make sure that you created or altered the ManagedSystem table in the Tivoli Data Warehouse. For more information, see (“Tivoli Data Warehouse” on page 609). • Make sure that you created the stored procedure that populates the ManagedSystem table for Agent Builder agents. For more information, see (“Tivoli Data Warehouse” on page 609). • Make sure that you run the stored procedure by using your agent's product code. For more information, see (“Running the DB2 stored procedure” on page 619). • Examine the `ManagedSystem`table to ensure that entries for your agent type exist in the table.

Table 51. Problems and solutions for Report Creation (continued)

Problem	Solution
Warnings when Model Advisor is run against the Cognos data model When the Model Advisor is run against the Cognos data model, the following warnings are shown: **Issue 1: Facts identified by cardinality** The query subject is identified as a fact because of the cardinality of its relationships. It is suggested that you review the implications of this fact to SQL generation and, if appropriate, change the usage of the query subject or its relationships. **Problem Description:** The query subject TEST_1 is on the many side of all its referencing relationships. **Issue 2: Query subjects that can behave as facts or dimensions** This query subject can behave as a fact or a dimension. It is suggested that you evaluate this query subject in the context of the model to ensure that queries are not split improperly or unnecessarily. **Problem Description:** The query subject `ManagedSystem` is referenced by relationship ends of different cardinalities. **Issue 3: Determinants that conflict with relationships** There is a determinant specified that conflicts with the relationship you defined. This ambiguity can cause a query to be split unnecessarily. It is suggested that you examine and resolve this ambiguity in the model. **Problem Description:** The relationship `TIME_DIMENSION <- -> WEEKDAY_LOOKUP` does not reference all the keys of any uniquely identified determinant in the query subject `TIME_DIMENSION`. The relationship has a 0:1 or 1:1 cardinality, indicating that its keys must be an exact match to the keys of a uniquely identified determinant. Review the cardinality of the relationship and the determinant information to determine whether adjustments are required. (continued on the next page)	These warnings are expected and do not signify a problem with the Cognos data model.

Table 51. Problems and solutions for Report Creation (continued)

Problem	Solution
Warnings when Model Advisor is run against the Cognos data model (continued) **Issue 4: Factors that override the Minimized SQL setting** A query subject has attributes that override the SQL Generation type setting of Minimized. Relationships between model query subjects, determinants on model query subjects and data source query subjects with modified SQL override the Minimized SQL Generation type. **Problem Description:** There can be problems to generate minimized SQL for the object `Time Dimension`. The object is set to use minimized SQL. **Problem Description:** There can be problems to generate minimized SQL for the object `ManagedSystem`. The object is set to use minimized SQL. **Problem Description:** There can be problems to generate minimized SQL for the object `ManagedSystemName`. The object is set to use minimized SQL.	These warnings are expected and do not signify a problem with the Cognos data model.

Support information

If you have a problem with your IBM software, you want to resolve it quickly.

IBM provides the following ways for you to obtain the support you need:

Online

The following websites contain troubleshooting information:

- Go to the IBM Software Support website (`http://www.ibm.com/support/entry/portal/software`) and follow the instructions.
- Go to the Application Performance Management Wiki (`http://www.ibm.com/developerworks/servicemanagement/apm/index.html`). Feel free to contribute to this wiki.

IBM Support Assistant

The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, go to the IBM Support Assistant website (`http://www.ibm.com/software/support/isa`).

Appendix A. Sharing project files

Share an IBM Tivoli Monitoring agent project with someone.

Procedure

1. Obtain their files. You need the entire contents of the directory with the same name as the project in your workspace directory. For example, if your workspace directory is `c:\Documents and Settings\User1\workspace` and you want to share your project named `TestProject`. You must make the directory `c:\Documents and Settings\User1\workspace\TestProject` and all of its contents accessible to your system.
2. Select **File** > **Import**.

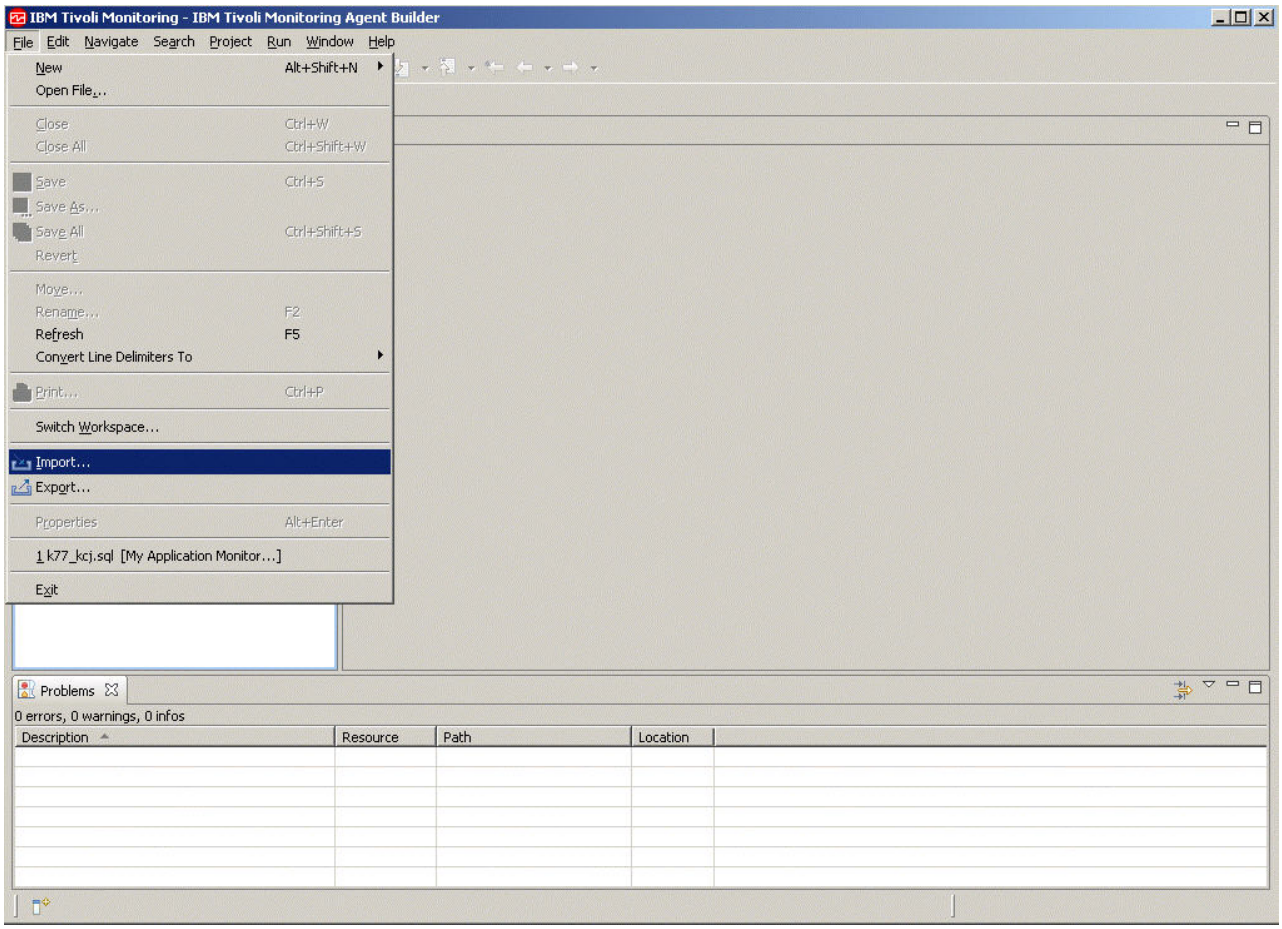


Figure 290. Importing files

3. Open **IBM Tivoli Monitoring**.
4. Select **IBM Tivoli Monitoring Agent** and click **Next**.

© Copyright IBM Corp. 2007, 2013

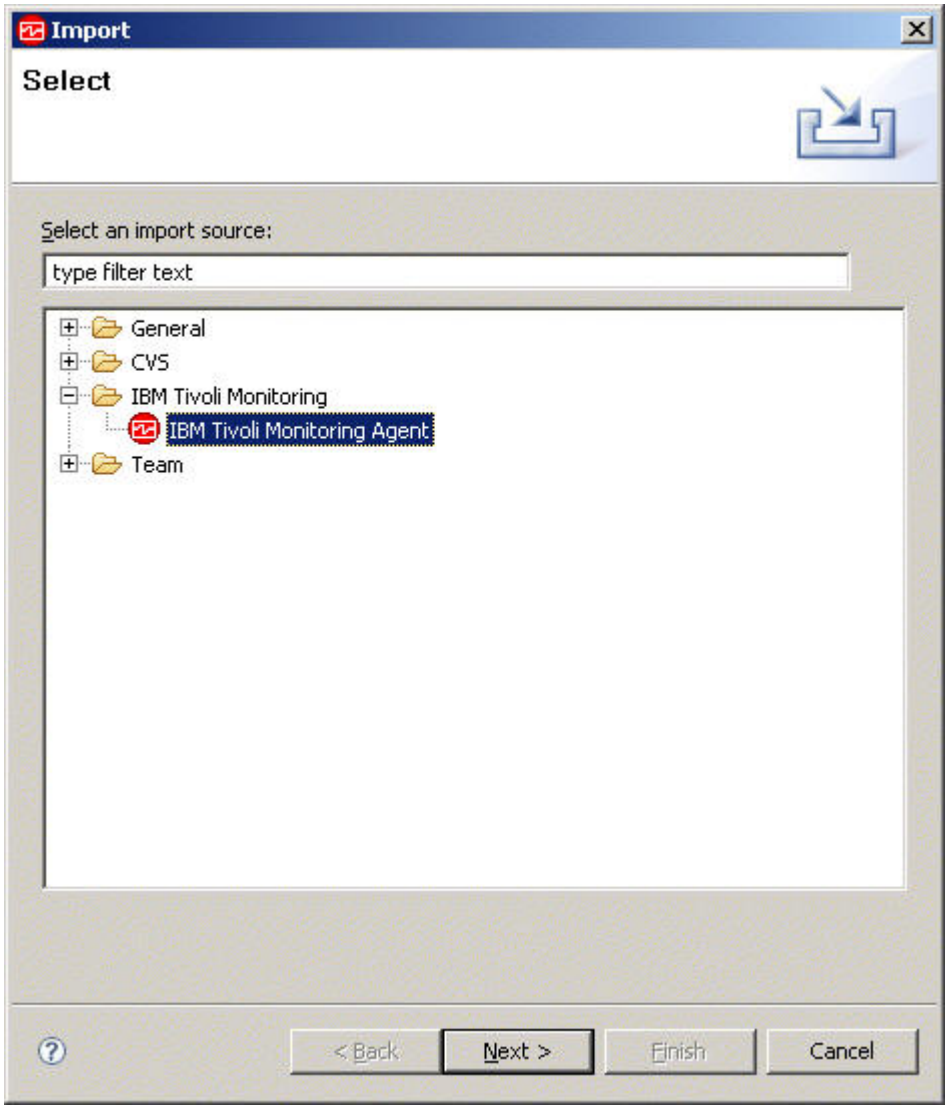


Figure 291. Importing files (continued)

5. Type the full path to the agent xml file or click **Browse** to browse to the file.
6. Click **Finish**.

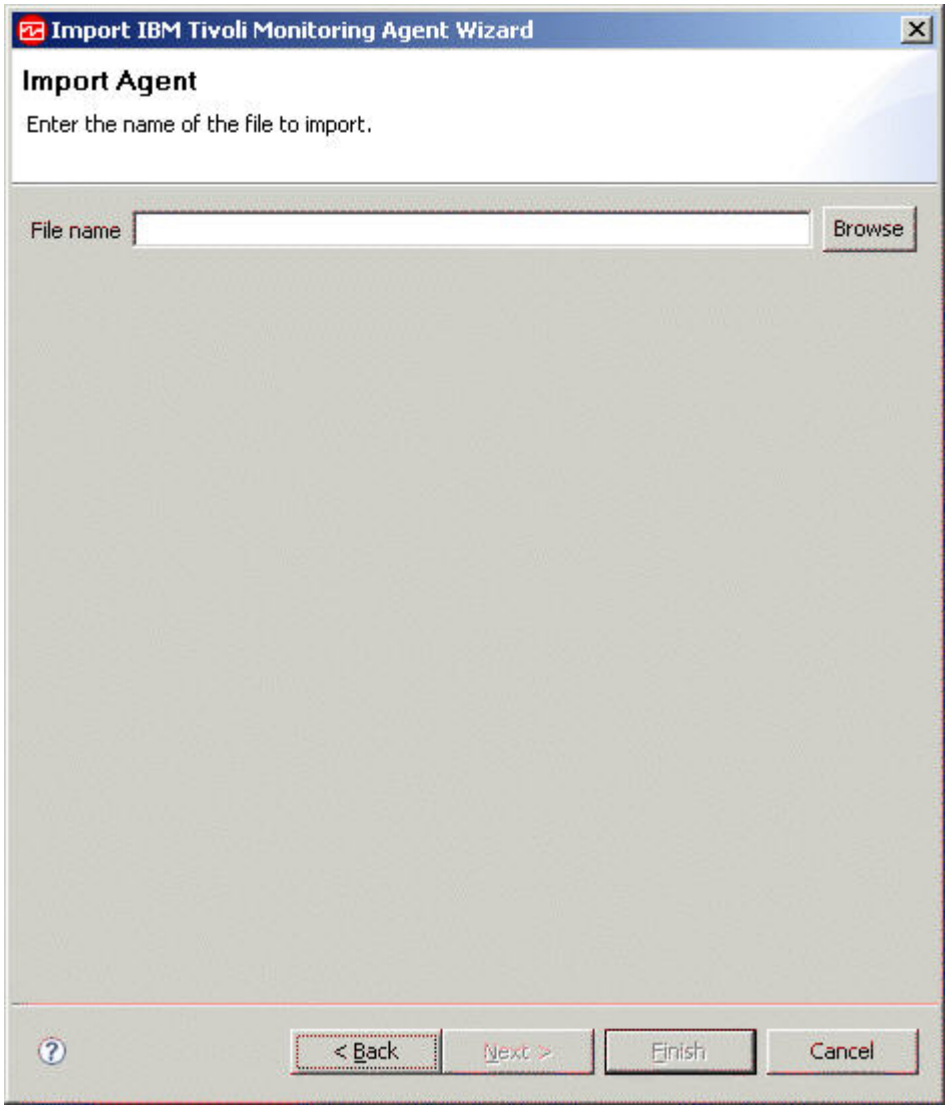


Figure 292. Importing files (continued)

Results

When the wizard completes, you see the new IBM Tivoli Monitoring agent project in your workspace.

Share a Solution Installer Project

Share a Solution Installer Project with someone

Procedure

1. Obtain their files. You must have the entire contents of the directory with the same name as the Solution Installer project in your workspace directory. For example, if your workspace directory is `c:\Documents and Settings\User1\workspace` and you want to share your Solution Installer project named `TestProject Installer`. You must make the directory `c:\Documents and Settings\User1\workspace\TestProject Installer` and all of its contents accessible to your system.
2. Click **File** > **Import**.
3. Open **General**.
4. Select **Existing Projects into Workspace**, and click **Next**.

5. Type the full path to the root directory of the Solution Installer project, or click **Browse** to browse to the root directory of the Solution Installer project. (In this example the `TestProject Installer` directory.) The Project in that directory is displayed in the Projects list and is selected by default.
6. Optional: Click **Copy projects into workspace**.
7. Click **Finish**.

Appendix B. Command-line options

Commands available from the Agent Builder command-line interface (CLI).

The Tivoli Monitoring Agent Builder contains a command-line interface (CLI) that you can use to generate the Tivoli Monitoring Agent without starting the Eclipse graphical user interface (GUI). You can generate the agent as part of a build, for example:

On Windows systems, you can use a batch file in the following directory to access the CLI:
install_location\agenttoolkit.bat

On UNIX and Linux systems, you can use a script in the following directory to access the CLI:
install_location/agenttoolkit.sh

The commands that are described in this documentation are formatted for Windows systems, which use a backslash (\) for directory paths.

For UNIX® or Linux® systems, use the same commands as for Windows systems, but with the following changes:

- Use a forward slash (/) for directory paths instead of a backslash (\).
- Use the `agenttoolkit.sh` script instead of the `agenttoolkit.bat` script.

Commands

Table 52 lists the name and purpose statement for each command option for the text command:

Table 52. Command quick-reference table

Command	Purpose
generatelocal	Loads and validates the `itm_toolkit_agent.xml` file and generates the files that run the Tivoli Monitoring Agent. The installation is into a local Tivoli Monitoring environment.
generatemappingfile	Creates the mapping file for porting custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6 agents.
generatezip	Generates a compressed file named *productcode*`.zip` or *productcode*`.tgz`.

The command descriptions that are referenced from the table describes how to run the commands by covering the following information:

Purpose
Lists the purpose of the command.

Format
Specifies the syntax that you type on the command line. The syntax contains the command name and a list of the parameters for the command. A definition of each parameter follows the command name.

Examples
The example for the command contains a brief description of the example and an example of the syntax.

Usage Provides an explanation of the command and its purpose.

© Copyright IBM Corp. 2007, 2013

Comments
Provides commands or text that can give you more information.

Command - generatelocal

Use this command to load and validate XML and to generate files to run the Tivoli Monitoring Agent.

Purpose

Loads and validates the itm_toolkit_agent.xml file and generates the files for running the Tivoli Monitoring Agent. The installation is into a local Tivoli Monitoring environment.

Format

For Windows systems:

```
install_location\agenttoolkit.bat project_dir -generatelocal itm_install_dir
```

where:

install_location
Directory where the Agent Builder is installed

project_dir
Name of the directory that contains the itm_toolkit_agent.xml file

itm_install_dir
Location where Tivoli Monitoring is installed (for example c:\IBM\ITM)

Examples

In the following example for Windows, the agent definition in C:\ABCAgent is validated and the files that are required to run ABCAgent are generated in C:\IBM\ITM:

```
install_location\agenttoolkit.bat C:\ABCAgent -generatelocal C:\IBM\ITM
```

Command - generatemappingfile

Use this command to migrate custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6 agents.

Purpose

This command creates the mapping file for migrating custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6 agents. For more information, see (Appendix F, "Upgrading custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6.2 agents," on page 625).

Format

For Windows systems:

```
install_location\agenttoolkit.bat project_dir -generatemappingfile output_dir
   itm5_interp_list
```

Where:

install_location
Directory where the Agent Builder is installed

project_dir
Name of the directory that contains itm_toolkit_agent.xml

output_dir
Name of the directory where the mapping file is written

itm5_interp_list
Comma-separated list of the ITM 5x operating systems on which the custom resource model ran. The following values are allowed:

- `aix4-r1`
- `hpux10`
- `linux-ix86`
- `linux-ppc`
- `linux-s390`
- `os2-ix86`
- `os400`
- `solaris2`
- `solaris2-ix86`
- `w32-ix86`

Examples

For Windows systems

```
install_location\agenttoolkit.bat c:\ABCAgent -generatemappingfile c:\output
   linux-ix86,linux-ppc,linux-s390
```

Command - `generatezip`

Use this command to load and validate XML and to generate a compressed file that can be used to install the agent on another system.

Purpose

Loads and validates the `itm_toolkit_agent.xml` file and generates a compressed file named *productcode*`.zip` or *productcode*`.tgz`. The generated compressed file can be used to install the agent on another system. Depending on your environment, both file types can be generated.

Format

For Windows systems:

```
install_location\agenttoolkit.bat project_dir -generatezip output_dir
```

Where:

project_dir
Name of a directory that contains the `itm_toolkit_agent.xml` file

output_dir
Name of a directory where the compressed file is written

Examples

In the following example for Windows, the agent definition in `C:\ABCAgent` is validated and a compressed file that contains the required files for running `ABCAgent` is generated in `C:\Output`:

```
install_location\agenttoolkit.bat\ C:\ABCAgent -generatezip C:\Output
```

Appendix C. Attributes reference

Contains descriptions of the attributes for each attribute generated group included in the Agent Builder.

Availability node

The Availability attribute group contains availability data for the application.

The table provides a common format for representing application availability, which includes relevant information for three aspects of an application: services (Windows only), processes, and command return codes.

The following list contains information about each attribute in the Availability attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Application Component attribute - This attribute is a key attribute

Description
The descriptive name of a part of the application

Type String

Names

Attribute name
Application_Component

Column name
COMPONENT

Name attribute

© Copyright IBM Corp. 2007, 2013

Description
The name of the process, service, or functional test. This name matches the executable name of the process, the service short name, or the name of the process that is used to test the application.

Type String

Names

Attribute name
Name

Column name
NAME

Status attribute

Description
The status of the application component.

- For processes, the values are `UP`, `DOWN`, `WARNING`, or `PROCESS_DATA_NOT_AVAILABLE`. `PROCESS_DATA_NOT_AVAILABLE` is displayed for a process when the matching process is running but the resource use information cannot be collected for that process.
- For services, the values are `UP`, `DOWN`, or `UNKNOWN`. `UNKNOWN` is displayed when the service is not installed.
- For command return codes, the values are `PASSED` or `FAILED`.

Type String

Names

Attribute name
Status

Column name
STATUS

Full Name attribute

Description
The full name of the process which includes information that is process-dependent. The name might include the full path if the process was started that way. The name can also include a partial path or even a path that is changed by the process.

Type String

Names

Attribute name
Full_Name

Column name
FULLNAME

Type attribute

Description
Identifies the type of the application component. Components are processes, services, or command return codes.

Type Integer (gauge)

Names

Attribute name
Type

Column name
TYPE

Virtual Size attribute

Description
The virtual size (in MB) of the process

Type Integer (gauge)

Names

Attribute name
Virtual_Size

Column name
VIRTSIZE

Page Faults Per Sec attribute

Description
The rate of page faults for the process that is measured in faults per second. This value contains only valid data for processes.

Type Integer (gauge)

Names

Attribute name
Page_Faults_Per_Sec

Column name
PAGEFAULTS

Working Set Size attribute

Description
The working set size of the process in MB. This value contains only valid data for processes.

Type Integer (gauge)

Names

Attribute name
Working_Set_Size

Column name
WORKSET

Thread Count attribute

Description
The number of threads that are currently allocated by this process. This value contains only valid data for processes.

Type Integer (gauge)

Names

Attribute name
Thread_Count

Column name
THREADS

PID attribute

Description
The process id that is associated with the process. This value contains only valid data for processes.

Type Integer (gauge)

Names

Attribute name
PID

Column name
PID

Percent Privileged Time attribute

Description
The percentage of the available processor time that is being used by this process for privileged operation

Type Integer (gauge)

Names

Attribute name
Percent_Privileged_Time

Column name
PERCPRIV

Percent User Mode Time attribute

Description
The percentage of the available processor time that is being used by this process for user mode operation

Type Integer (gauge)

Names

Attribute name
Percent_User_Mode_Time

Column name
PERCUSER

Percent Processor Time attribute

Description
The percentage of the elapsed time that this process used the processor to run instructions

Type Integer (gauge)

Names

Attribute name
Percent_Processor_Time

Column name
PERCPROC

Command Line attribute

Description
The program name and any arguments that are specified on the command line when the process was started. This attribute has the value *N/A* if you are running a Service or Functionality test.

Type String

Names

Attribute name
Command_Line

Column name
CMDLINE

Functionality Test Status attribute

Description
The return code of the functionality test. When the monitored application is running correctly, SUCCESS is returned. NOT_RUNNING is returned when the application is not running correctly. N/A is returned when the row does not represent a functionality test.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the numbers. The defined values are: N/A(1), SUCCESS (0), GENERAL_ERROR (2), WARNING (3), NOT_RUNNING (4), DEPENDENT_NOT_RUNNING (5), ALREADY_RUNNING (6), PREREQ_NOT_RUNNING (7), TIMED_OUT (8), DOESNT_EXIST (9), UNKNOWN (10), DEPENDENT_STILL_RUNNING (11), or INSUFFICIENT_USER_AUTHORITY (12). Any other values display the numeric value in the Tivoli Enterprise Portal.

Names

Attribute name
Functionality_Test_Status

Column name
FUNCSTATUS

Functionality Test Message attribute

Description
The text message that corresponds to the Functionality Test Status. This attribute is valid only for command return codes.

Type String

Names

Attribute name
Functionality_Test_Message

Column name
FUNCMSG

Performance Object Status node

Use the Performance Object Status attribute group to see the status of all of the attribute groups that make up the agent. Each of the attribute groups is represented by a row in this table or other type of view. The status of an attribute group reflects the result of the last data collection attempt, or data reception event, for the attribute group. When you check the status information, you can see whether the agent is operating correctly. When your agent does not collect data, but receives it (event data), attributes that relate to sampled data do not contain useful data. Only the first seven attributes that are listed are relevant for event data.

Historical group
This attribute group is eligible for use with Tivoli Data Warehouse.

Attribute descriptions
The following list contains information about each attribute in the Performance Object Status attribute group:

Node attribute: This attribute is a key attribute.

Description
The managed system name of the agent.

Type String

Warehouse name
NODE

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type String

Warehouse name
TIMESTAMP

Query Name attribute: This attribute is a key attribute.

Description
The name of the attribute group.

Type String

Warehouse name
QUERY_NAME or ATTRGRP

Object Name attribute

Description
The name of the performance object.

Type String

Warehouse name
OBJECT_NAME or OBJNAME

Object Type attribute

Description
The type of the performance object.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- WMI (0)
- PERFMON (1)
- WMI ASSOCIATION GROUP (2)
- JMX (3)
- SNMP (4)
- SHELL COMMAND (5)
- JOINED GROUPS (6)
- CIMOM (7)
- CUSTOM (8)
- ROLLUP DATA (9)
- WMI REMOTE DATA (10)
- LOG FILE (11)
- JDBC (12)

- CONFIG DISCOVERY (13)
- NT EVENT LOG (14)
- FILTER (15)
- SNMP EVENT (16)
- PING (17)
- DIRECTOR DATA (18)
- DIRECTOR EVENT (19)
- SSH REMOTE SHELL COMMAND (20)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
OBJECT_TYPE or OBJTYPE

Object Status attribute

Description
The status of the performance object.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- ACTIVE (0)
- INACTIVE (1)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
OBJECT_STATUS or OBJSTTS

Error Code attribute

Description
The error code that is associated with the query.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- NO ERROR (0)
- GENERAL ERROR (1)
- OBJECT NOT FOUND (2)
- COUNTER NOT FOUND (3)
- NAMESPACE ERROR (4)
- OBJECT CURRENTLY UNAVAILABLE (5)
- COM LIBRARY INIT FAILURE (6)
- SECURITY INIT FAILURE (7)
- PROXY SECURITY FAILURE (9)
- NO INSTANCES RETURNED (10)
- ASSOCIATOR QUERY FAILED (11)
- REFERENCE QUERY FAILED (12)
- NO RESPONSE RECEIVED (13)
- CANNOT FIND JOINED QUERY (14)
- CANNOT FIND JOIN ATTRIBUTE IN QUERY 1 RESULTS (15)

- CANNOT FIND JOIN ATTRIBUTE IN QUERY 2 RESULTS (16)
- QUERY 1 NOT A SINGLETON (17)
- QUERY 2 NOT A SINGLETON (18)
- NO INSTANCES RETURNED IN QUERY 1 (19)
- NO INSTANCES RETURNED IN QUERY 2 (20)
- CANNOT FIND ROLLUP QUERY (21)
- CANNOT FIND ROLLUP ATTRIBUTE (22)
- FILE OFFLINE (23)
- NO HOSTNAME (24)
- MISSING LIBRARY (25)
- ATTRIBUTE COUNT MISMATCH (26)
- ATTRIBUTE NAME MISMATCH (27)
- COMMON DATA PROVIDER NOT STARTED (28)
- CALLBACK REGISTRATION ERROR (29)
- MDL LOAD ERROR (30)
- AUTHENTICATION FAILED (31)
- CANNOT RESOLVE HOST NAME (32)
- SUBNODE UNAVAILABLE (33)
- SUBNODE NOT FOUND IN CONFIG (34)
- ATTRIBUTE ERROR (35)
- CLASSPATH ERROR (36)
- CONNECTION FAILURE (37)
- FILTER SYNTAX ERROR (38)
- FILE NAME MISSING (39)
- SQL QUERY ERROR (40)
- SQL FILTER QUERY ERROR (41)
- SQL DB QUERY ERROR (42)
- SQL DB FILTER QUERY ERROR (43)
- PORT OPEN FAILED (44)
- ACCESS DENIED (45)
- TIMEOUT (46)
- NOT IMPLEMENTED (47)
- REQUESTED A BAD VALUE (48)
- RESPONSE TOO BIG (49)
- GENERAL RESPONSE ERROR (50)
- SCRIPT NONZERO RETURN (51)
- SCRIPT NOT FOUND (52)
- SCRIPT LAUNCH ERROR (53)
- CONF FILE DOES NOT EXIST (54)
- CONF FILE ACCESS DENIED (55)
- INVALID CONF FILE (56)
- EIF INITIALIZATION FAILED (57)
- CANNOT OPEN FORMAT FILE (58)
- FORMAT FILE SYNTAX ERROR (59)
- REMOTE HOST UNAVAILABLE (60)

- EVENT LOG DOES NOT EXIST (61)
- PING FILE DOES NOT EXIST (62)
- NO PING DEVICE FILES (63)
- PING DEVICE LIST FILE MISSING (64)
- SNMP MISSING PASSWORD (65)
- DISABLED (66)
- URLS FILE NOT FOUND (67)
- XML PARSE ERROR (68)
- NOT INITIALIZED (69)
- ICMP SOCKETS FAILED (70)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
ERROR_CODE or ERRCODE

Last Collection Start attribute

Description
The most recent time a data collection of this group started.

Type Time stamp with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- NOT COLLECTED (0691231190000000)
- NOT COLLECTED (0000000000000001)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
LAST_COLLECTION_START or COLSTRT

Last Collection Finished attribute

Description
The most recent time a data collection of this group finished.

Type Time stamp with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- NOT COLLECTED (0691231190000000)
- NOT COLLECTED (0000000000000001)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
LAST_COLLECTION_FINISHED or COLFINI

Last Collection Duration attribute

Description
The duration of the most recently completed data collection of this group in seconds.

Type Real number (32-bit counter) with two decimal places of precision

Warehouse name
LAST_COLLECTION_DURATION or COLDURA

Average Collection Duration attribute

Description
The average duration of all data collections of this group in seconds.

Type Real number (32-bit counter) with two decimal places of precision with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- `NO DATA (-100)`

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
AVERAGE_COLLECTION_DURATION or COLAVGD

Refresh Interval attribute

Description
The interval at which this group is refreshed in seconds.

Type Integer (32-bit counter)

Warehouse name
REFRESH_INTERVAL or REFRINT

Number of Collections attribute

Description
The number of times this group is collected since agent start.

Type Integer (32-bit counter)

Warehouse name
NUMBER_OF_COLLECTIONS or NUMCOLL

Cache Hits attribute

Description
The number of times an external data request for this group is satisfied from the cache.

Type Integer (32-bit counter)

Warehouse name
CACHE_HITS or CACHEHT

Cache Misses attribute

Description
The number of times an external data request for this group was not available in the cache.

Type Integer (32-bit counter)

Warehouse name
CACHE_MISSES or CACHEMS

Cache Hit Percent attribute

Description
The percentage of external data requests for this group that are satisfied from the cache.

Type Real number (32-bit counter) with two decimal places of precision

Warehouse name
CACHE_HIT_PERCENT or CACHPCT

Intervals Skipped attribute

Description
The number of times a background data collection was skipped because the previous collection was still running when the next one was due to start.

Type Integer (32-bit counter)

Warehouse name
INTERVALS_SKIPPED or INTSKIP

Thread Pool Status attribute group

The Thread Pool Status attribute group contains information that reflects the status of the internal thread pool that is used to collect data asynchronously.

The following comprises a list of the attributes for this attribute group. The name in bold text shows how the attribute is displayed in the Tivoli Enterprise Portal.

The following list contains information about each attribute in the Thread Pool Status attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time that is collected from the agent system when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Thread Pool Size attribute

Description
The number of threads currently existing in the thread pool.

Type Integer

Names

Attribute name
Thread_Pool_Size

Column name
THPSIZE

Thread Pool Max Size attribute

Description
The maximum number of threads that are allowed to exist in the thread pool.

Type Integer

Names

Attribute name
Thread_Pool_Max_Size

Column name
TPMAXSZ

Thread Pool Active Threads attribute

Description
The number of threads in the thread pool currently active doing work.

Type Integer

Names

Attribute name
Thread_Pool_Active_Threads

Column name
TPACTTH

Thread Pool Avg Active Threads attribute

Description
The average number of threads in the thread pool simultaneously active doing work.

Type Integer

Names

Attribute name
Thread_Pool_Avg_Active_Threads

Column name
TPAVGAT

Thread Pool Min Active Threads attribute

Description
The minimum number of threads in the thread pool simultaneously active doing work.

Type Integer

Names

Attribute name
Thread_Pool_Min_Active_Threads

Column name
TPMINAT

Thread Pool Max Active Threads attribute

Description
The peak number of threads in the thread pool simultaneously active doing work.

Type Integer

Names

Attribute name
Thread_Pool_Max_Active_Threads

Column name
TPMAXAT

Thread Pool Queue Length attribute

Description
The number of jobs currently waiting in the thread pool queue.

Type Integer

Names

Attribute name
Thread_Pool_Queue_Length

Column name
TPQLGTH

Thread Pool Avg Queue Length attribute

Description
The average length of the thread pool queue during this run.

Type Integer

Names

Attribute name
Thread_Pool_Avg_Queue_Length

Column name
TPAVGQL

Thread Pool Min Queue Length attribute

Description
The minimum length the thread pool queue reached.

Type Integer

Names

Attribute name
Thread_Pool_Min_Queue_Length

Column name
TPMINQL

Thread Pool Max Queue Length attribute

Description
The peak length the thread pool queue reached.

Type Integer

Names

Attribute name
Thread_Pool_Max_Queue_Length

Column name
TPMAXQL

Thread Pool Avg Job Wait attribute

Description
The average time a job spends waiting on the thread pool queue.

Type Integer

Names

Attribute name
Thread_Pool_Avg_Job_Wait

Column name
TPAVJBW

Thread Pool Total Jobs attribute

Description
The number of jobs that are completed by all threads in the pool since agent start.

Type Integer

Names

Attribute name
Thread_Pool_Total_Jobs

Column name
TPTJOBS

Event log attribute node

The Event log attribute group contains any recent event log entries that pertain to the application.

By default, the agent displays only events that occur after the agent is started. Events are removed from the Event Log view 1 hour after they occur.

The following list contains information about each attribute in the Event Log attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Log Name attribute

Description
The event log - Application, System, Security, or an application-specific log

Type String

Names

Attribute name
Log_Name

Column name
LOGNAME

Event Source attribute

Description
The event source that is defined by the application

Type String

Names

Attribute name
Event_Source

Column name
EVTSOURCE

Event Type attribute

Description
Event Type - Error(0), Warning(1), Informational(2), Audit_Success(3), Audit_Failure(4), Unknown(5)

Type Integer

Names

Attribute name
Event_Type

Column name
EVTTYPE

Event ID attribute

Description
The ID of the event

Type Integer

Names

Attribute name
Event_ID

Column name
EVTID

Event Category attribute

Description
The category of the event

Type String

Names

Attribute name
Event_Category

Column name
EVTCATEG

Message attribute

Description
The event message

Type String

Names

Attribute name
Message

Column name
MESSAGE

Time Generated attribute

Description
The time the event was generated

Type Time

Names

Attribute name
Time_Generated

Column name
TIMESTAMP

Log File Summary

The attributes of this attribute group are included in summary attribute groups when that option is selected in the advanced properties of the data source.

A Summary node is created for each Log File data source when **Include attribute in summary attribute group** is selected in the advanced properties of the data source. The name of the summary node is the name of the data source with Summary added to the end.

The following list contains information about each of the default attributes in the Log File Summary attribute group. These attributes are always included in summary attribute groups. If you select **Include attribute in summary attribute group**, see step 9 on page 187 in (Chapter 16, "Monitoring a log file," on page 177), then the summary attribute group for that log attribute group also contains each of the attributes you selected. The values are a copy of the corresponding attribute in the log file attribute group.

All of the added attributes together become a key and the summary table includes one row per unique set of keys. The row indicates how many log records are received during the interval where all of the provided keys matched the value reported in the corresponding attributes.

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Interval Unit attribute

Description
The number of seconds between summary attribute generation

Type Integer (gauge)

Names

Attribute name
_Interval_Unit

Column name
IU

Interval attribute

Description
Offset of the current interval within the next larger unit of time (for example, minutes within an hour)

Type Integer (gauge)

Names

Attribute name
_Interval

Column name
INV

Occurrences attribute

Description
The number of occurrences that are recorded during the interval

Type Integer (gauge)

Names

Attribute name
_Occurrences

Column name
OCC

LocalTimeStamp attribute

Description
The time that the summary data was generated

Type Timestamp

Names

Attribute name
_LocalTimeStamp

Column name
LTS

DateTime attribute

Description
The time that the summary data was generated

Type String

Names

Attribute name
_Date_Time

Column name
DT

Interval Unit Name attribute

Description
The word description of the interval unit

Type String

Names

Attribute name
_Interval_Unit_Name

Column name
IUN

AIX Binary Log attribute group

The AIX Binary Log attribute group displays events from the AIX Binary Log as selected by the provided `errpt` command string.

The following list contains information about each attribute in the AIX Binary Log Attribute Group:

Note: The Agent Builder prevents removing, reordering, or changing the size of the `Identifier`, `ErrptTimestamp`, `Type`, `Class`, `ResourceName`, and `Description` attributes. The agent parses the data that comes back from an `errpt` command that is based on columns within the line of text. These columns are defined by the order and size of the `Identifier`, `ErrptTimestamp`, `Type`, `Class`, `ResourceName`, and `Description` attributes. Removing, reordering, or changing the size of these attributes, changes the attribute that the various columns go into. The resulting row as seen in Tivoli Monitoring is then incorrect.

You can, however, rename these attributes.

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Identifier attribute - This attribute is a key attribute

Description
The event identifier reported by `errpt`

Type String

Names

Attribute name
Identifier

Column name
IDENTIFIER

ErrptTimestamp attribute

Description
The time the event is recorded as reported by `errpt`.

Note: This attribute is hidden at run time. This attribute contains a raw value. Other attributes that are derived from this attribute display the value in a more usable form. This attribute is available from within Agent Builder for that purpose, but by default it is not visible in the Tivoli Monitoring environment at run time. If you want to make it visible, select the attribute in the Data Source Definition page in the Agent Editor and select **Display attribute in the Tivoli Enterprise Portal**.

Type String

Names

Attribute name
ErrptTimestamp

Column name
ERRPTTIMES

Type

Description
The single character event type reported by `errpt`, one of `I(NFO)`, `P(END/ERF/ERM)`, `T(EMP)`, and `U(NKN)`

Type String

Names

Attribute name
Type

Column name
TYPE

Class attribute - This attribute is a key attribute

Description
The event class reported by `errpt`, one of `Hardware`, `Software`, `Operator`, and `Undertermined`. These values are enumerated. The raw values for use with situations are `H`, `S`, `O`, and `U`.

Type String

Names

Attribute name
Class

Column name
CLASS

ResourceName

Description
The resource name reported by `errpt`, identifies the origin of the error record

Type String

Names

Attribute name
ResourceName

Column name
RESOURCENA

Description attribute

Description
The description reported by `errpt`, typically a short text message that describes the nature of the error

Type String

Names

Attribute name
Description

Column name
DESCRIPTIO

LogFile attribute

Description
The full name of the binary `errpt` log including the path.

Note: This attribute is hidden at run time. This attribute contains a raw value. Other attributes that are derived from this attribute display the value in a more useable form. This attribute is available from within Agent Builder for that purpose, but by default it is not visible in the Tivoli Monitoring environment at run time. If you want to make it visible, select the attribute in the Data Source Definition page in the Agent Editor and select **Display attribute in the Tivoli Enterprise Portal**.

Type String

Names

Attribute name
LogFile

Column name
LOGFILE

System attribute

Description
The host name of the system where the error was collected

Type String

Names

Attribute name
System

Column name
SYSTEM

LogName attribute

Description
The base name of the binary `errpt` log from which the record was collected

Type String

Names

Attribute name
LogName

Column name
LOGNAME

LogPath attribute

Description
The directory name that contains the binary `errpt` log from which the record was collected

Type String

Names

Attribute name
LogPath

Column name
LOGPATH

EntryTime attribute

Description
The time the event is recorded as reported by `errpt` in Tivoli Timestamp format. This time is not necessarily identical to the time when the agent received the event, as recorded in the **`Timestamp`** field.

Type Time stamp

Names

Attribute name
EntryTime

Column name
ENTRYTIME

Monitor and Notification attribute groups

Definitions for the Monitor and Notification attribute groups.

The first 4 are specific to monitors and the last is for notifications (all are related to JMX).

Each one is listed with an indication whether it is event-based or not. For non-event based attribute groups, data is collected when needed. For event-based attribute groups, the agent maintains a cache of the last 100 events received. These events are used to respond to requests from the Tivoli Enterprise Portal. The events are forwarded immediately for analysis by situations and warehousing.

Counter Notifications

The Counter Notifications attribute group is a non-event based attribute group that sends events that are received by all counter monitors.

The following list contains information about each attribute in the Counter Notifications attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Notification Type attribute

Description
The type of notification received. Describes how the observed attribute of the MBean triggered the notification.

Type String

Names

Attribute name
Notification_Type

Column name
NOTIFICATI

Monitor ID attribute

Description
Monitor ID of the monitor who generated this notification

Type Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Observed MBean attribute

Description
The MBean whose attribute is being monitored

Type String

Names

Attribute name
Observed_MBean

Column name
OBSERVED_M

Observed Attribute attribute

Description
Name of the attribute that is monitored in the Observed MBean

Type String

Names

Attribute name
Observed_Attribute

Column name
OBSERVED_A

Threshold attribute

Description
The current threshold of the monitor

Type String

Names

Attribute name
Threshold

Column name
THRESHOLD

Offset attribute

Description
The value added to the threshold each time the attribute exceeds the threshold. This value forms a new threshold.

Type String

Names

Attribute name
Offset

Column name
OFFSET

Modulus attribute

Description
The maximum value of the attribute. When it reaches this value, it rolls over and begins counting again from zero.

Type Integer

Names

Attribute name
Modulus

Column name
MODULUS

Counter Value attribute

Description
Value of the counter that triggered the notification

Type Integer

Names

Attribute name
Counter_Value

Column name
COUNTER_VA

Notification Time Stamp attribute

Description
Time that the notification was triggered

Type Time

Names

Attribute name
Notification_Time_Stamp

Column name
NOTIFICAT0

Notification Message attribute

Description
The message in the notification

Type String

Names

Attribute name
Notification_Message

Column name
NOTIFICAT1

Gauge Notifications

The Gauge Notifications attribute group is a non-event based attribute group that sends events that are received by all gauge monitors.

The following list contains information about each attribute in the Gauge Notifications attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Notification Type attribute

Description
The type of notification received. Describes how the observed attribute of the MBean triggered the notification.

Type String

Names

Attribute name
Notification_Type

Column name
NOTIFICATI

Monitor ID attribute

Description
Monitor ID of the monitor who generated this notification

Type Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Observed MBean attribute

Description
The MBean whose attribute is being monitored

Type String

Names

Attribute name
Observed_MBean

Column name
OBSERVED_M

Observed Attribute attribute

Description
Name of the attribute that is monitored in the Observed MBean

Type String

Names

Attribute name
Observed_Attribute

Column name
OBSERVED_A

Low Threshold attribute

Description
The threshold that the monitor is watching for the observed attribute to cross

Type String

Names

Attribute name
Low_Threshold

Column name
LOW_THRESH

High Threshold attribute

Description
The threshold that the monitor is watching for the observed attribute to cross

Type String

Names

Attribute name
High_Threshold

Column name
HIGH_THRES

Gauge Value attribute

Description
Value of the gauge that triggered the notification

Type String

Names

Attribute name
Gauge_Value

Column name
MODULUSGAUGE_VALU

Notification Time Stamp attribute

Description
Time that the notification was triggered

Type Time

Names

Attribute name
Notification_Time_Stamp

Column name
NOTIFICAT0

Notification Message attribute

Description
The message in the notification

Type String

Names

Attribute name
Notification_Message

Column name
NOTIFICAT1

Registered Monitors

The Registered Monitors attribute group is an event-based attribute group that shows a list of all JMX Monitors that are created by the agent.

The following list contains information about each attribute in the Registered Monitors attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Monitor ID attribute - This attribute is a key attribute

Description
The unique integer identifier for a monitor

Type Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Monitor Parameters attribute

Description
The parameters that are used to create the monitor

Type String

Names

Attribute name
Monitor_Parameters

Column name
MONITOR_PA

Monitor Name attribute

Description
The JMX Object Name of the monitor MBean

Type String

Names

Attribute name
Monitor_Name

Column name
MONITOR_NA

String Notifications

The String Notifications attribute group is a non-event based attribute group that sends events that are received by all string monitors.

The following list contains information about each attribute in the String Notifications attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Notification Type attribute

Description
The type of notification received. Describes how the observed attribute of the MBean triggered the notification.

Type String

Names

Attribute name
Notification_Type

Column name
NOTIFICATI

Monitor ID attribute - This attribute is a key attribute

Description
The unique integer identifier for a monitor

Type Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Observed MBean attribute

Description
The MBean whose attribute is being monitored

Type String

Names

Attribute name
Observed_MBean

Column name
OBSERVED_M

Observed Attribute attribute

Description
Name of the attribute that is monitored in the Observed MBean

Type String

Names

Attribute name
Observed_Attribute

Column name
OBSERVED_A

Compare String attribute

Description
The string that is used in the comparison operation

Type String

Names

Attribute name
Compare_String

Column name
COMPARE_ST

String Value attribute

Description
Value of the attribute that triggered the notification

Type String

Names

Attribute name
String_Value

Column name
STRING_VAL

Notification Time Stamp attribute

Description
Time that the notification was triggered

Type Time

Names

Attribute name
Notification_Time_Stamp

Column name
NOTIFICAT0

Notification Message attribute

Description
The message in the notification

Type String

Names

Attribute name
Notification_Message

Column name
NOTIFICAT1

SNMP Event attribute groups

SNMP event attribute groups are used to receive traps and informs. These attribute groups are event-based attribute groups

The following list contains information about each attribute in the SNMP Event Attribute Groups:

Note: You can change the default display name of these attributes. These display names are distinct from the internal ID for each attribute.

Enterprise_OID
The enterprise OID that generated the trap.

Source_Address
Host name or IP address of the SNMP agent that sent the trap.

Generic_Trap
Generic trap number that is extracted from the received trap. Possible values:

```
0 ColdStart
1 WarmStart
2 LinkDown
3 LinkUp
4 Authentication Failure
5 EGPNeighborLoss
```

Specific_Trap
Enterprise-specific trap number that is extracted from the received trap. Applies only when Generic_Trap = 6.

Alert_Name
Trap name as specified in the definition in the trap configuration file.

Category
Trap category as specified in the definition in the trap configuration file.

Description
Trap description as specified in the definition in the trap configuration file. The maximum description length is 256 characters.

Enterprise_Name
Trap Enterprise name as specified in the trap configuration file and determined through the trap object identifier.

Source_Status
Status of the agent that originated the trap after the trap is sent as specified in the trap definition in the trap configuration file.

Source_Type
Type of the agent that originated the trap as specified in the trap definition in the trap configuration file.

Event_Variables
Variable binding (VarBind) data that is received in the trap protocol data unit (PDU). The string is constructed as:

```
{OID[type]=value}{OID[type]=value}{oid[type]=value}...
```

Where:

oid MIB variable object identifier

type SMI data type

value Variable value

{} Each triplet is surrounded by braces ({}).

Note: The attributes Alert Name, Category, Description, Enterprise_Name, Source_Status, and Source_Type provide more information. In the SNMP MIB Browser window (Figure 96 on page 136), select the **Include attributes that show information defined in the trap configuration file** check box to include these attributes.

JMX Event attribute groups

JMX event attribute groups are used to receive notifications from an MBean server.

These attribute groups are non-event based attribute groups and are generated with the following attributes that can be edited by the agent developer.

The following list contains information about each attribute in the JMX Event Attribute Groups:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Type attribute

Description
The notification type

Type String

Names

Attribute name
Type

Column name
TYPE

Source attribute

Description
The MBean that caused the notification to be sent

Type String

Names

Attribute name
Source

Column name
SOURCE

Sequence Number attribute

Description
The sequence number from the notification object

Type String

Names

Attribute name
Sequence_Number

Column name
SEQUENCE_N

Message attribute

Description
The notification message

Type String

Names

Attribute name
Message

Column name
MESSAGE

User Data attribute

Description
The user data object from the notification

Type String

Names

Attribute name
User_Data

Column name
USER_DATA

Ping attribute group

The Ping attribute group contains the results of ICMP pings that are sent to lists of devices.

The following list contains information about each attribute in the Ping Attribute Group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time that is collected from the agent system when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Address attribute - This attribute is a key attribute

Description
The IP address of the host that is monitored.

Type String with enumerated value. The value UNKNOWN_ADDRESS is displayed if the IP address is unknown. The warehouse and queries return 0.0.0.0 for this enumeration. Any other IP address values are displayed as is.

Names

Attribute name
Address

Column name
PNGADDR

Device Entry attribute - This attribute is a key attribute

Description
The entry in the device list file for this node.

Type String

Names

Attribute name
Device_Entry

Column name
PINGDEVC

Current Response Time attribute

Description
The current network response time for ICMP requests for the managed node in milliseconds.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `TIMEOUT(-1)` and `SEND_FAILURE(-2)`. Any other values show the numeric value.

Names

Attribute name
Current_Response_Time

Column name
PINGRSTM

Name attribute

Description
The host name of the managed node. If the node address cannot be resolved through DNS, then the dotted decimal IP address is shown.

Type String with enumerated value. The value UNKNOWN_HOSTNAME is displayed if the

host name is unknown. The warehouse and queries return 0.0.0.0 for this enumeration. Any other host name values are displayed as is.

Names

Attribute name
Name

Column name
PNGNAME

Node Description attribute

Description
The description of the managed node.

Type String

Names

Attribute name
Node_Description

Column name
PNGDESC

Node Status attribute

Description
The current operating status of the managed node.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `INVALID(-2)`, `UNKNOWN(-1)`, `INACTIVE(0)`, and `ACTIVE(1)`.

Names

Attribute name
Node_Status

Column name
PNGSTAT

Node Type attribute

Description
The type of the managed node. If the node is online, it is an IP Node. If it is offline, the type is Unknown.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `UNKNOWN(0)` and `IP NODE(1)`.

Names

Attribute name
Node_Type

Column name
PNGTYPE

Status Timestamp

Description
The date and time the node was last checked.

Type Time

Names

Attribute name
Status_Timestamp

Column name
PNGTMSP

HTTP attribute groups

The two HTTP attribute groups, `Managed URLs` and `URL Objects`, are used to receive information from URLs and the objects within theses URLs.

For information about the syntax that is used in the Managed URLs and URL Objects tables, see ("Specific fields for HTTP attributes" on page 264).

Managed URLs

The following list contains information about each attribute in the Managed URL Attribute Group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

URL attribute - This attribute is a key attribute

Description
The URL that is being monitored.

Type String

Names

Attribute name
URL

Column name
HTTPURL

Response Time attribute

Description
The amount of time it took to download the response in milliseconds.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `TIMEOUT (-1)`.

Names

Attribute name
Response_Time

Column name
HTTPURL

Page Size attribute

Description
The size of the page that is returned by the HTTP request.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NO_RESPONSE_RECEIVED(-1)`.

Names

Attribute name
Page_Size

Column name
PAGESZ

Page Objects attribute

Description
The total number of objects that are associated with the monitored page.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NOT_COLLECTED(-1)`.

Names

Attribute name
Page_Objects

Column name
PGOBJS

Total Object Size attribute

Description
The size of the page that is returned by the HTTP request.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NOT_COLLECTED(-1)`.

Names

Attribute name
Total_Object_Size

Column name
TOTOSZ

Page Title attribute

Description
The page title of the received URL page.

Type String

Names

Attribute name
Page_Title

Column name
PAGETTL

Server Type attribute

Description
The type of server that is used at the target URL website.

Type String

Names

Attribute name
Server_Type

Column name
SRVTYP

Response Code attribute

Description
The response code of the HTTP request.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NO_RESPONSE_RECEIVED(-1)`.

Names

Attribute name
Response_Code

Column name
CODE

Status attribute

Description
The current managed URL status (OK or status description).

Type String

Names

Attribute name
Status

Column name
STATUS

URL Alias attribute

Description
The user-specified alias for the URL.

Type String

Names

Attribute name
URL_Alias

Column name
ALIAS

User Data attribute

Description
The user data that is specified with the URL.

Type String

Names

Attribute name
User_Data

Column name
USER

URL Objects

The following list contains information about each attribute in the URL Objects Attribute Group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

URL attribute - This attribute is a key attribute

Description
The URL that is being monitored.

Type String

Names

Attribute name
URL

Column name
HTTPURL

Object Name attribute

Description
The name of the page object within the target URL.

Type String

Names

Attribute name
Object_Name

Column name
ONAME

Object Size attribute

Description
The size (bytes) of the page object within the target URL.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `NOT_COLLECTED (-1)`, `OBJECT_NOT_FOUND (-2)`. Any other values show the numeric value.

Names

Attribute name
Object_Size

Column name
SIZE

Object Response Time attribute

Description
The amount of time it took to download the object in milliseconds.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `NOT_COLLECTED (-1)`, `NO_RESPONSE_RECEIVED (-2)`, `STATUS_CODE_ERROR (-3)`. Any other values show the numeric value.

Names

Attribute name
Object_Response_Time

Column name
ORTIME

Discovery attribute groups

An attribute group that represents the set of subnode instances that are defined for a subnode type

When you create a subnode type, an attribute group is created that represents the set of subnode instances that are defined for that subnode type. Each of these attribute groups includes the same set of attributes.

The following list contains information about each attribute in a Discovery attribute group. The name in bold text shows how the attribute is displayed in the Tivoli Enterprise Portal:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time from the agent system when the data row was built and sent to the Tivoli Enterprise Monitoring Server (or stored for historical purposes). It represents the local time zone of the agent system.

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Subnode MSN attribute

Description
The Managed System Name of the subnode agent.

Type String

Names

Attribute name
Subnode_MSN

Column name
SN_MSN

Subnode Affinity attribute

Description
The affinity for the subnode agent.

Type String

Names

Attribute name
Subnode_Affinity

Column name
SN_AFFIN

Subnode Type attribute

Description
The node type of this subnode.

Type String

Names

Attribute name
Subnode_Type

Column name
SN_TYPE

Subnode Resource Name attribute

Description
The resource name of the subnode agent.

Type String

Names

Attribute name
Subnode_Resource_Name

Column name
SN_RES

Subnode Version attribute

Description
The version of the subnode agent.

Type

Names

Attribute name
Subnode_Version

Column name
SN_VER

Take Action Status attribute group

The Take Action Status attribute group contains the status of actions that the agent processed.

This attribute group is event-based and contains information about each attribute in the Take Action Status attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time that is collected from the agent system, when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Action Name attribute

Description
The name of the action that was run

Type String

Names

Attribute name
Action_Name

Column name
TSKNAME

Action Status attribute

Description
The status of the action.

Type Integer with enumerated values. The values are: OK (0), NOT_APPLICABLE (1), GENERAL_ERROR (2), WARNING (3), NOT_RUNNING (4), DEPENDENT_NOT_RUNNING (5), ALREADY_RUNNING (6), PREREQ_NOT_RUNNING (7), TIMED_OUT (8), DOESNT_EXIST (9), UNKNOWN (10), DEPENDENT_STILL_RUNNING (11), INSUFFICIENT_USER_AUTHORITY (12)

Names

Attribute name
Action_Status

Column name
TSKSTAT

Action Application Return Code attribute

Description
The return code of the application the action started.

Type Integer

Names

Attribute name
Action_App_Return_Code

Column name
TSKAPRC

Action Message attribute

Description
The message that is associated with the return code of the action.

Type String

Names

Attribute name
Action_Message

Column name
TSKMSGE

Action Instance attribute

Description
The instance that is associated with the output produced by running the action. If the action is a system command, the instance is the line number of the output of the command.

Type String

Names

Attribute name
Action_Instance

Column name
TSKINST

Action Results attribute

Description
The output that is produced by running the action.

Type String

Names

Attribute name
Action_Results

Column name
TSKOUTP

Action Command attribute

Description
The command that was run by the action.

Type String

Names

Attribute name
Action_Command

Column name
TSKCMND

Action Node attribute

Description
The node where the action ran.

Type String

Names

Attribute name
Action_Node

Column name
TSKORGN

Action Subnode attribute

Description
The subnode where the action ran.

Type String

Names

Attribute name
Action_Subnode

Column name
TSKSBND

Action ID attribute

Description
The ID of the action.

Type Integer

Names

Attribute name
Action_ID

Column name
TSKID

Action Type attribute

Description
The type of the action.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the numbers. The defined values are: UNKNOWN (0), AUTOMATION (1).

Names

Attribute name
Action_Type

Column name
TSKTYPE

Action Owner attribute

Description
The name of the situation or user that initiated the action.

Type String

Names

Attribute name
Action_Owner

Column name
TSKOWNR

Log File Status attribute group

The Log File Status attribute group contains information that reflects the status of log files this agent is monitoring.

The Log File Status attribute group is included if you have a log attribute group and the agent is at the default minimum Tivoli Monitoring version of 6.2.1 or later. The Log File Status attribute group includes two attributes that are defined as 64-bit numbers so that they can handle large files. 64-bit numeric attribute support is provided by Tivoli Monitoring version 6.2.1 or later.

The following list contains information about each attribute in the Log File Status attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The value is the time that is collected from the agent system, when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Table Name attribute - This attribute is a key attribute

Description
The name of the table in which this log is being monitored

Type String

Names

Attribute name
Table_Name

Column name
TBLNAME

File Name attribute - This attribute is a key attribute

Description
The name of the file that is being monitored

Type String

Names

Attribute name
File_Name

Column name
FILNAME

RegEx Pattern attribute - This attribute is a key attribute

Description
The regular expression pattern (if any) that caused this file to be monitored

Type String

Names

Attribute name
RegEx_Pattern

Column name
REPATRN

File Type attribute

Description
The type of this file (regular file or pipe)

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The defined values are UNKNOWN(0), REGULAR FILE(1), PIPE(2)

Names

Attribute name
File_Type

Column name
FILTYPE

File Status attribute

Description
The status of the file that is being monitored

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The defined values are: OK(0), PERMISSION DENIED(1), FILE DOES NOT EXIST(2), INTERRUPTED SYSTEM CALL(4), I/O ERROR(5), NO SUCH DEVICE(6), BAD FILE NUMBER(9), OUT OF MEMORY(12), ACCESS DENIED(13), RESOURCE BUSY(16), NOT A DIRECTORY(20), IS A DIRECTORY(21), INVALID ARGUMENT(22), FILE TABLE OVERFLOW(23), TOO MANY OPEN FILES(24), TEXT FILE BUSY(26), FILE TOO LARGE(27), NO SPACE LEFT ON DEVICE(28), ILLEGAL SEEK ON PIPE(29), READ-ONLY FILE SYSTEM(30), TOO MANY LINKS(31), BROKEN PIPE(32)

Names

Attribute name
File_Status

Column name
FILSTAT

Num Records Matched attribute

Description
The number of processed records from this log which matched one of the specified patterns

Type Integer

Names

Attribute name
Num_Records_Matched

Column name
RECMTCH

Num Records Not Matched attribute

Description
The number of processed records sent to the UnmatchLog; did not match any patterns

Type Integer

Names

Attribute name
Num_Records_Not_Matched

Column name
RECUNMT

Num Records Processed attribute

Description
The number of records that are processed from this log since agent start (including ones that are not matches/events)

Type Integer

Names

Attribute name
Num_Records_Processed

Column name
RECPROC

Current File Position attribute

Description
The current position in bytes into the monitored file. Data up to this position is processed, data after this position is not processed. Not applicable to pipes.

Type Integer

Names

Attribute name
Current_File_Position

Column name
OFFSET

Current File Size attribute

Description
The current size of the monitored file. Not applicable to pipes.

Type Integer

Names

Attribute name
Current_File_Size

Column name
FILESIZE

Last Modification Time attribute

Description
The time when the monitored file was last written to. Not applicable to pipes.

Type Timestamp

Names

Attribute name
Last_Modification_Time

Column name
LASTMOD

Codepage attribute

Description
The language codepage of the monitored file

Type String

Names

Attribute name
Codepage

Column name
CODEPG

Log File RegEx Statistics attribute group

The Log File RegEx Statistics attribute group contains information that shows the statistics of the log file regular expression search expressions.

Regular expressions can be used to filter records or to define records. This attribute group shows information about both types. When the Result Type attribute contains either `INCLUDE` or `EXCLUDE`, the filter is use to filter records. If the Result Type attribute contains `BEGIN` or `END`, the filter is used to define records. The CPU measurements are approximations that are based on the granularity of the data that is exposed by the operating system. These measurements can result in values of `0.00` when a regular expression takes a small time to evaluate. Use the CPU times to determine the relative cost of regular expressions and to optimize the behavior of specific regular expressions.

The Log File RegEx Statistics attribute group is included if you have a log attribute group and the agent is at Tivoli Monitoring version of 6.2.1 or later. The minimum Tivoli Monitoring Version is selected on the Agent Information page. For more information, see (“Naming your agent” on page 19). The Log File RegEx Statistics attribute group includes attributes that are defined as 64-bit numbers so that they can handle long durations. Support for 64-bit numeric attributes is provided by Tivoli Monitoring version 6.2.1 or later.

The following list contains information about each attribute in the Log File RegEx Statistics attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Table Name attribute - This attribute is a key attribute

Description
The name of the log file attribute group.

Type String

Names

Attribute name
Table_Name

Column name
TBLNAME

Attribute Name attribute - This attribute is a key attribute

Description
The name of the attribute to which this filter is applied.

Type String

Names

Attribute name
Attribute_Name

Column name
ATRNAME

Filter Number

Description
The sequence number, starting at zero, of the filter that is being used for the attribute.

Type Integer (Numeric Property)

Names

Attribute name
Filter_Number

Column name
FLTRNUM

Result Type attribute

Description
The result type can be INCLUDE or EXCLUDE to accept or reject the attribute if the filter matches. The result type can be BEGIN or END to specify the start or end of a record for multi-line records.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. If the filter is used to filter records, the defined values are `INCLUDE(1)` or `EXCLUDE(2)`. If the filter is used to define records, the defined values are `BEGIN(3)` or `END(4)`.

Names

Attribute name
Result_Type

Column name
RSTTYPE

Average Processor Time attribute

Description

The average number of processor seconds used to process the filter for this attribute. The average processor time is the total processor seconds divided by the filter count.

Type Integer (Gauge)

Names

Attribute name
Average_Processor_Time

Column name
CPUTAVG

Processor Time attribute

Description

The total number of processor seconds used to process the filter for this attribute. The processor time is cumulative and is truncated, not rounded. Similar to the Linux `/proc/<pid>/task/thread/stat file`.

Type Integer (Counter)

Names

Attribute name
Processor_Time

Column name
CPUTIME

Max Processor Time attribute

Description

The maximum number of processor seconds used for a single filter processing. It is possible that the maximum is zero if the filter was never used or if each of the filter processing took less than 0.01 seconds.

Type Integer (Gauge)

Names

Attribute name
Max_Processor_Time

Column name
CPUTMAX

Min Processor Time attribute

Description

The minimum number of processor seconds used for a single filter processing. It is possible that the minimum is zero if a filter processing took less than 0.01 seconds.

Type Integer (Gauge)

Names

Attribute name
Min_Processor_Time

Column name
CPUTMIN

Filter Count attribute

Description
The number of times the filter is run. Used with the total processor time to compute the average processor time.

Type Integer (Counter)

Names

Attribute name
Filter_Count

Column name
COUNT

Filter Count Matched attribute

Description
The number of times the filter is run and the attribute matched.

Type Integer (Counter)

Names

Attribute name
Filter_Count_Matched

Column name
COUNTMA

Filter Count Unmatched attribute

Description
The number of times the filter is run and the attribute did not match.

Type Integer (Counter)

Names

Attribute name
Filter_Count_Unmatched

Column name
COUNTUN

RegEx Pattern attribute - This attribute is a key attribute

Description
The regular expression that is used for the match.

Type String

Names

Attribute name
RegEx_Pattern

Column name
REGXPAT

Last Matched Time attribute

Description
The last time the filter was used and the result matched.

Type Time

Names

Attribute name
Last_Matched_Time

Column name
LASTMAT

Last Unmatched Time attribute

Description
The last time the filter was used and the result was unmatched.

Type Time

Names

Attribute name
Last_Unmatched_Time

Column name
LASTUMA

Appendix D. Creating application support extensions for existing agents

Build an installable package to distribute custom workspaces, situations, queries, and Take Action commands that you created, as an application support extension for an existing agent.

Before you begin

For more information about how to create custom situations, workspaces, Take Action commands, and queries, see (Chapter 32, “Creating workspaces, Take Action commands, and situations,” on page 403).

About this task

Attention:: This task is not how you add application support to an agent that you are building. To add application support to an agent that you are building see (Chapter 33, “Importing application support files,” on page 409).

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Select **IBM Tivoli Monitoring Application Support Extension** under **IBM Tivoli Monitoring Wizards**.

© Copyright IBM Corp. 2007, 2013

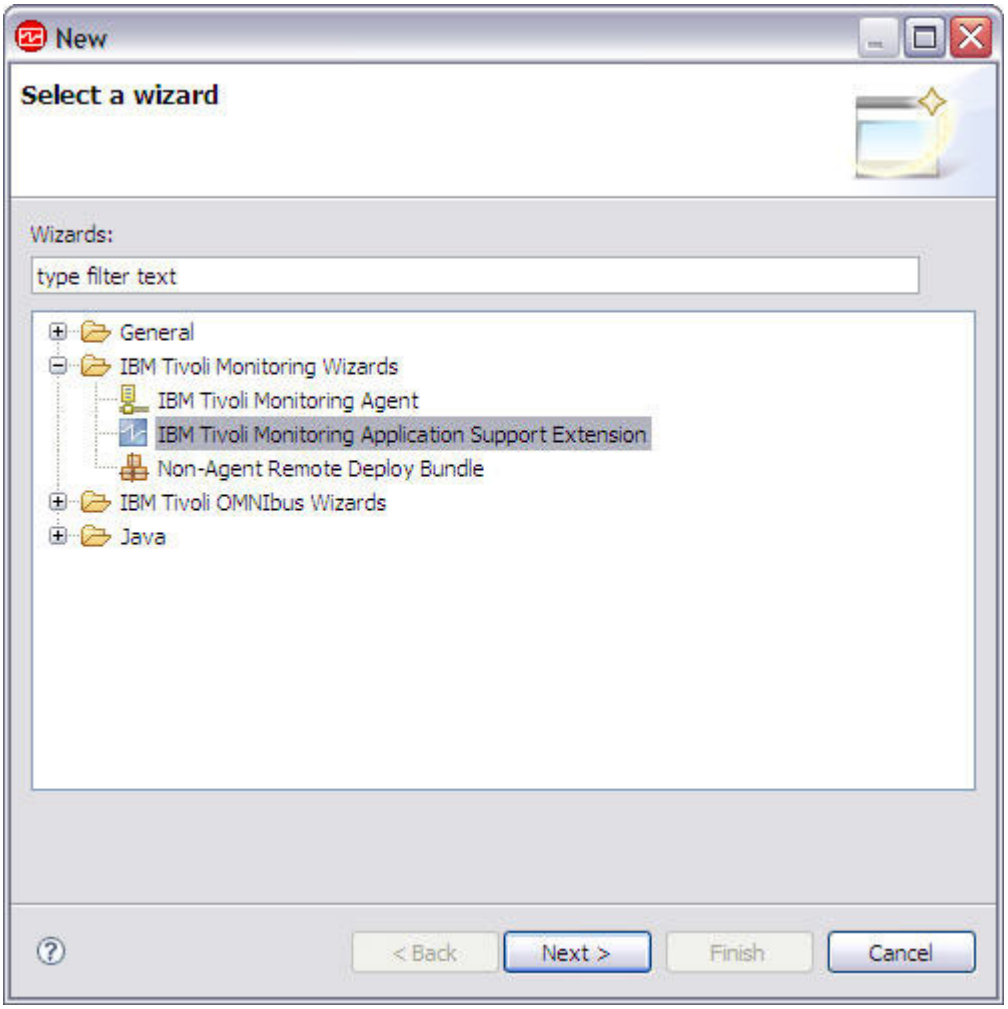


Figure 293. IBM Tivoli Monitoring Application Support Extension

3. Click **Next** to get to the welcome page for the IBM Tivoli Monitoring Application Support Extension wizard.
4. Click **Next** on the welcome page.
5. Enter a name for the project and click **Finish**

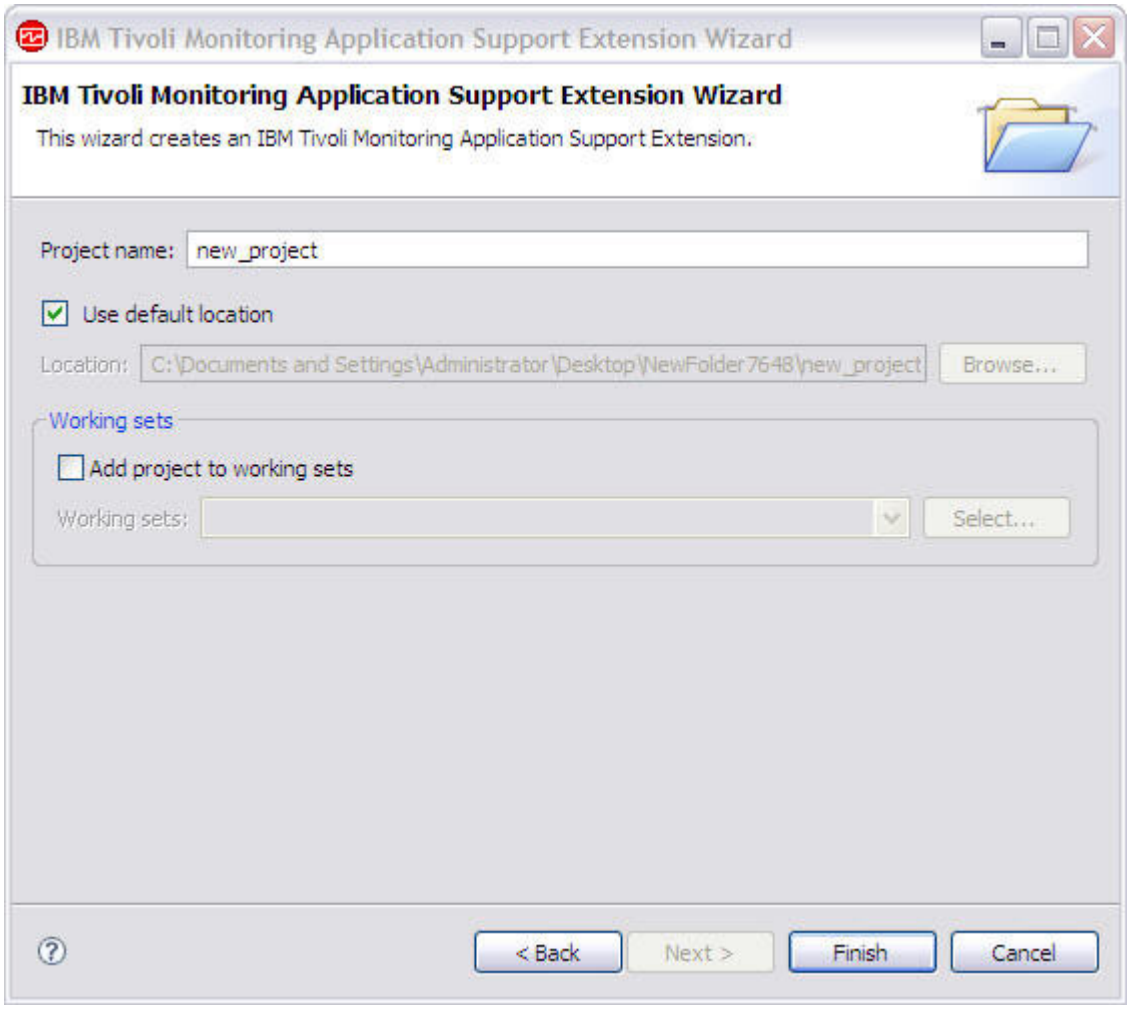


Figure 294. Naming the project

Creating an Application Support Extension project

Create an Application Support Extension project by using Agent builder.

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Select **IBM Tivoli Application Support Extension** under **IBM Tivoli Monitoring Wizards**.

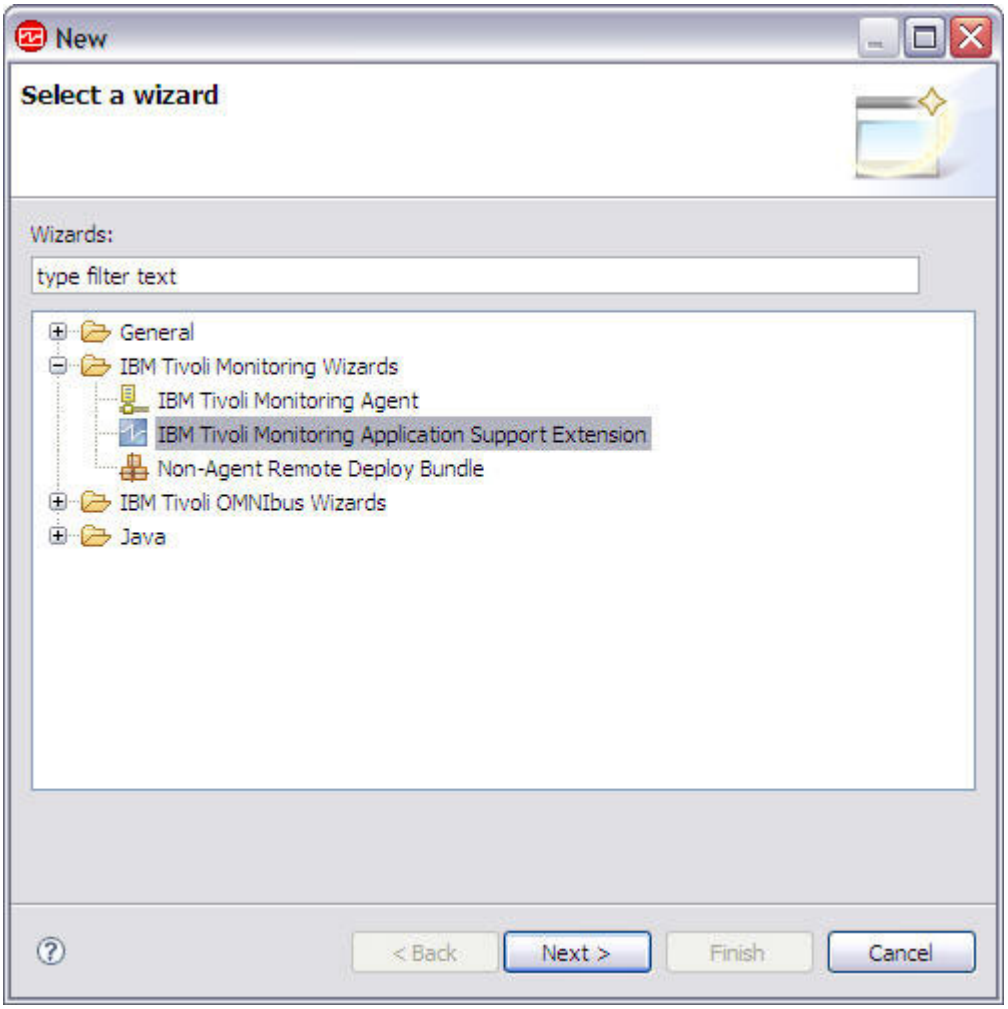


Figure 295. IBM Tivoli Monitoring Application Support Extension

3. Click **Next** to get to the welcome page for the **IBM Tivoli Monitoring Application Support Extension Wizard**.
4. Click **Next** on the welcome page.
5. Enter a name for the project and click **Finish** Figure 296 on page 599.

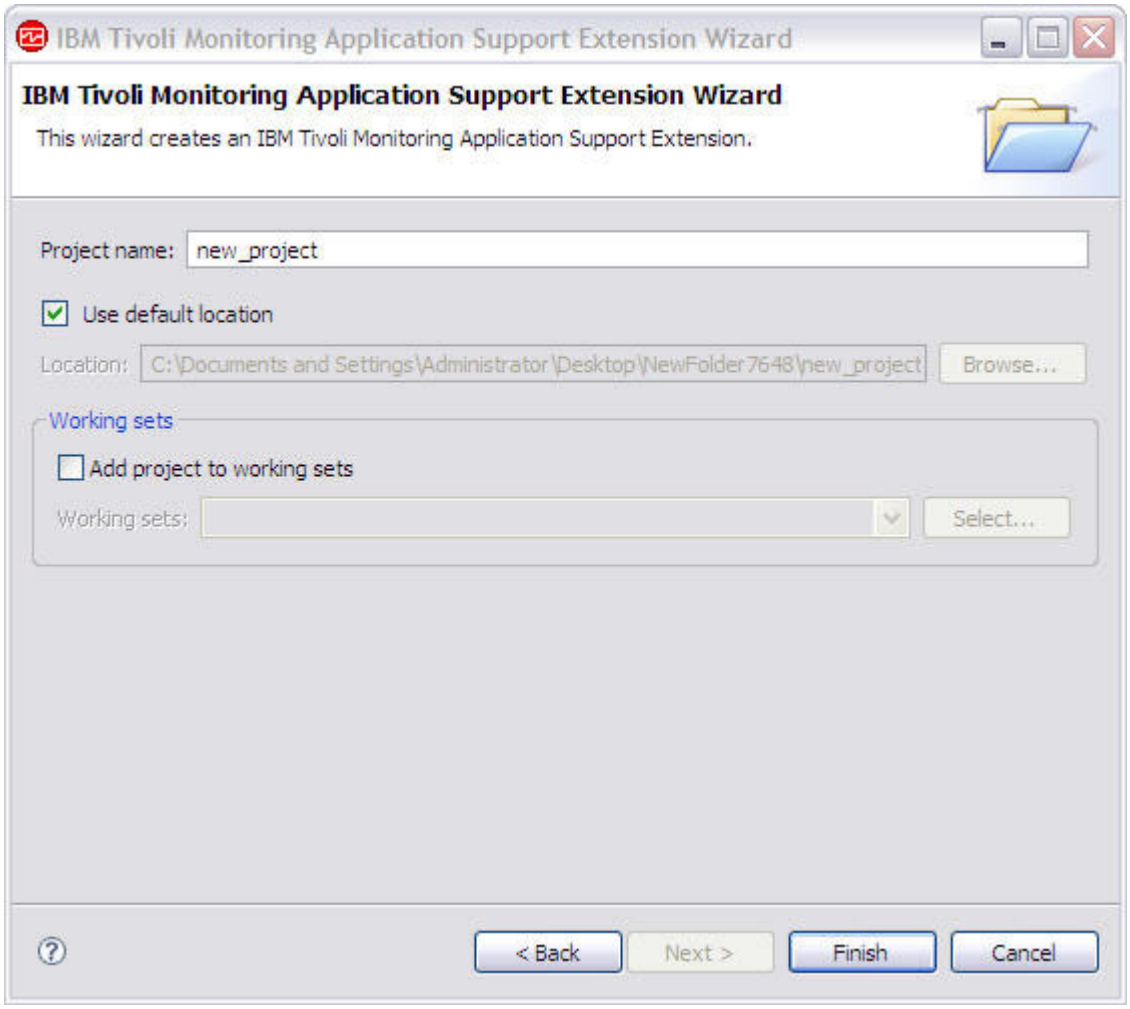


Figure 296. Naming the project

Adding support files to a project

Add your support files to an Application Support Extension project

Before you begin

Create an Application Support Extension project. For more information, see "Creating an Application Support Extension project" on page 597.

Procedure

1. Right-click an Application Support Extension project and select **IBM Tivoli** > **Import Application Support Extensions**

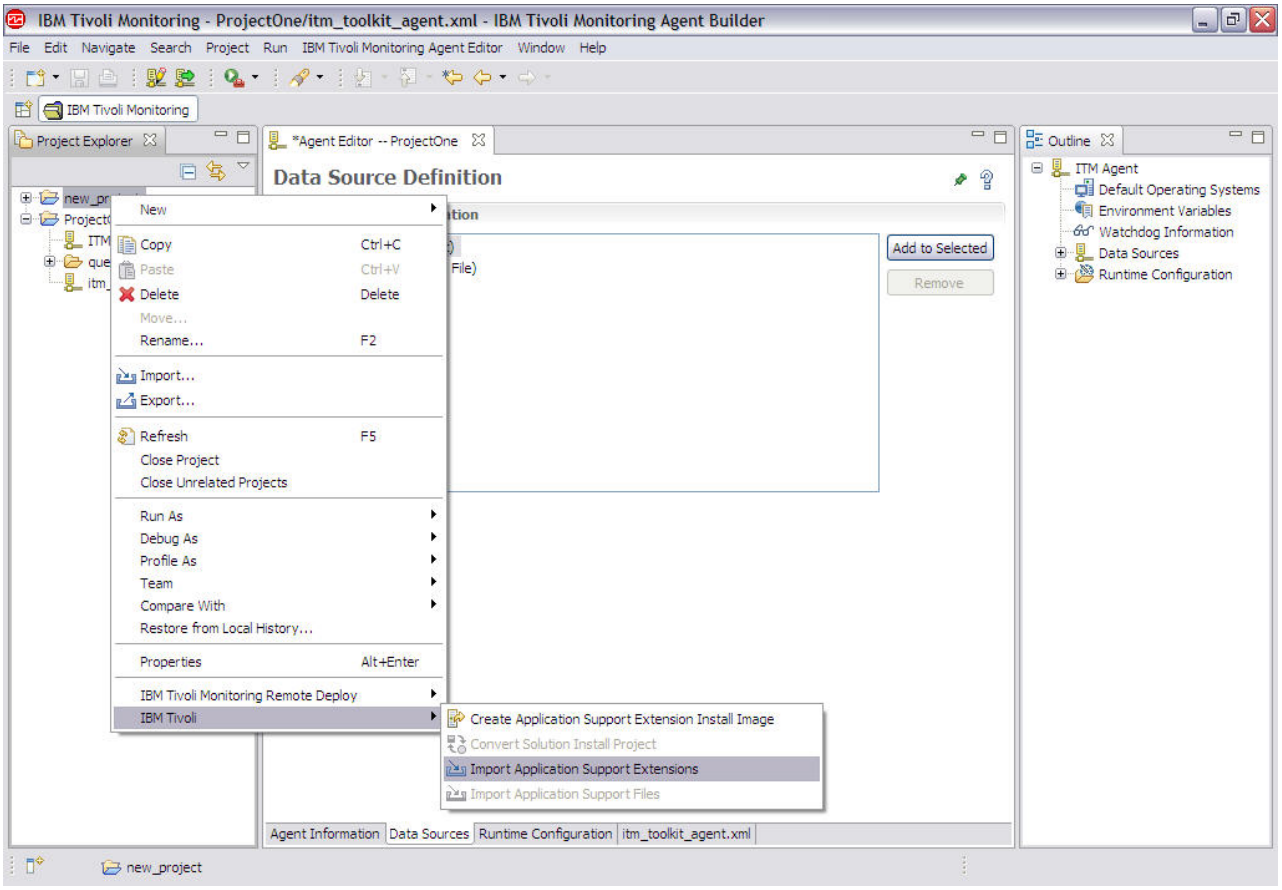


Figure 297. Importing support files

2. In the Import Information window, select the name of the host where the Tivoli Enterprise Portal Server is located or click **Add** to add one.

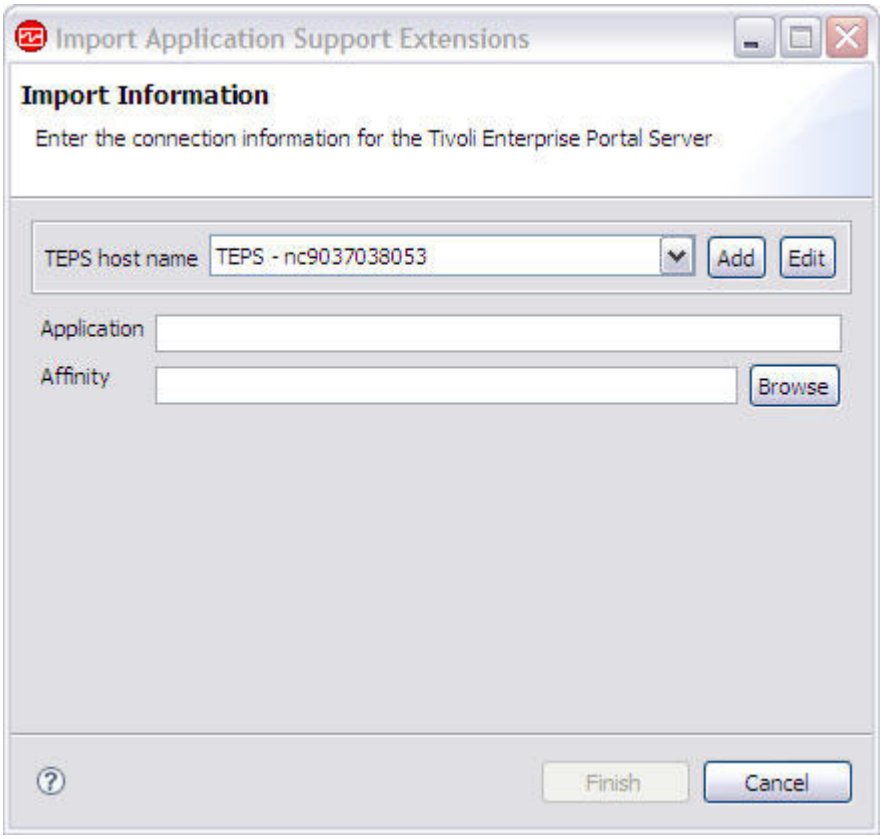


Figure 298. Import Information

3. In the **Application** field, enter the agent product code.
4. Enter the affinity of the agent for which you are creating custom application support. The agent affinity is a Tivoli Monitoring internal identifier that associates workspaces, queries, and other items, with the agent. It must be unique in the Tivoli Monitoring installation. Click **Browse** to open the Node Types window and select this information from a list rather than typing it.

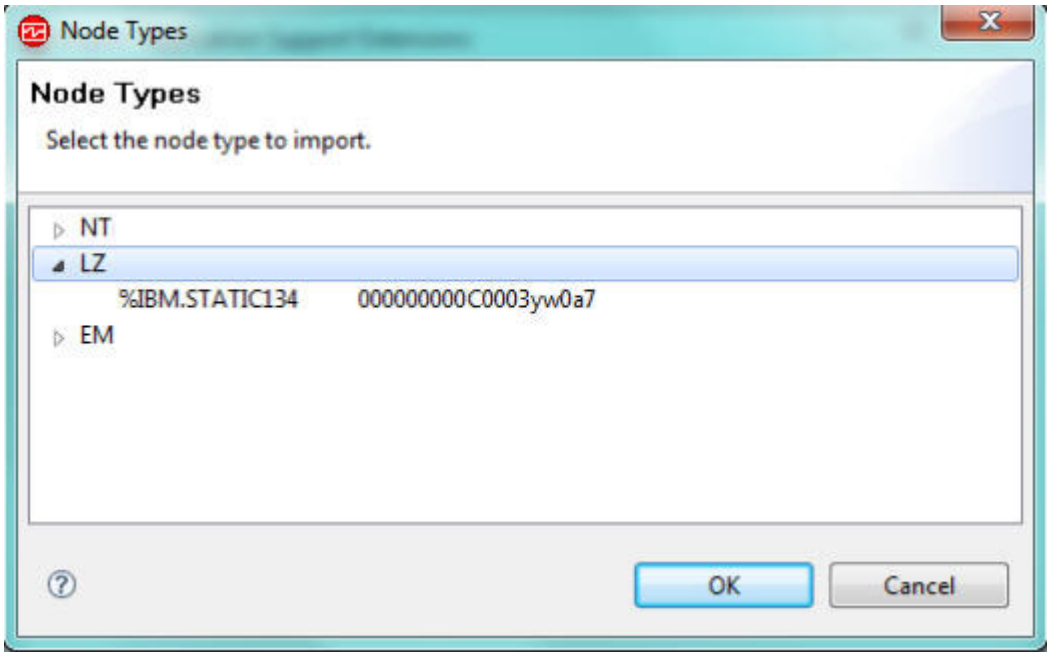


Figure 299. Affinities list

5. When you are satisfied with the import information, click **Finish**.

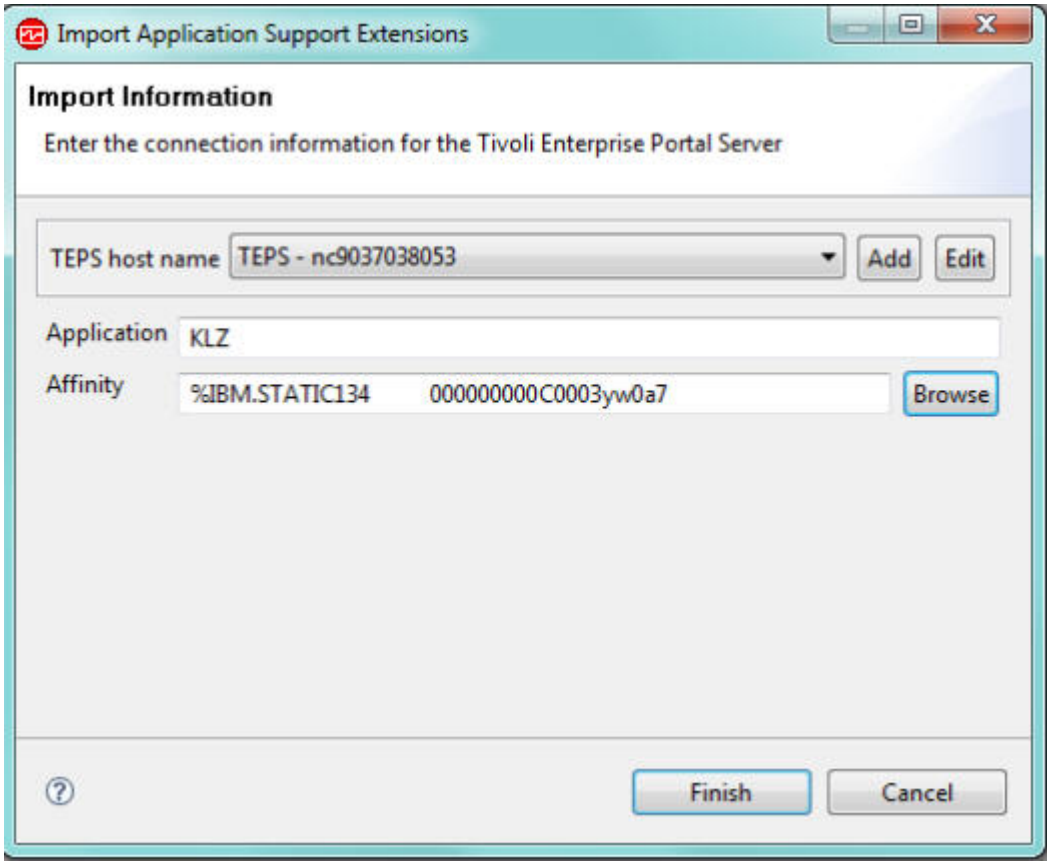


Figure 300. Import Information complete

6. In the Situations window (Figure 301), select the situations that you want to import from the Available Situations list. Click << to add them to the Selected Situations list and click **OK**. A new folder is created under the project, and it contains the necessary files to install the workspaces, situations, and queries.

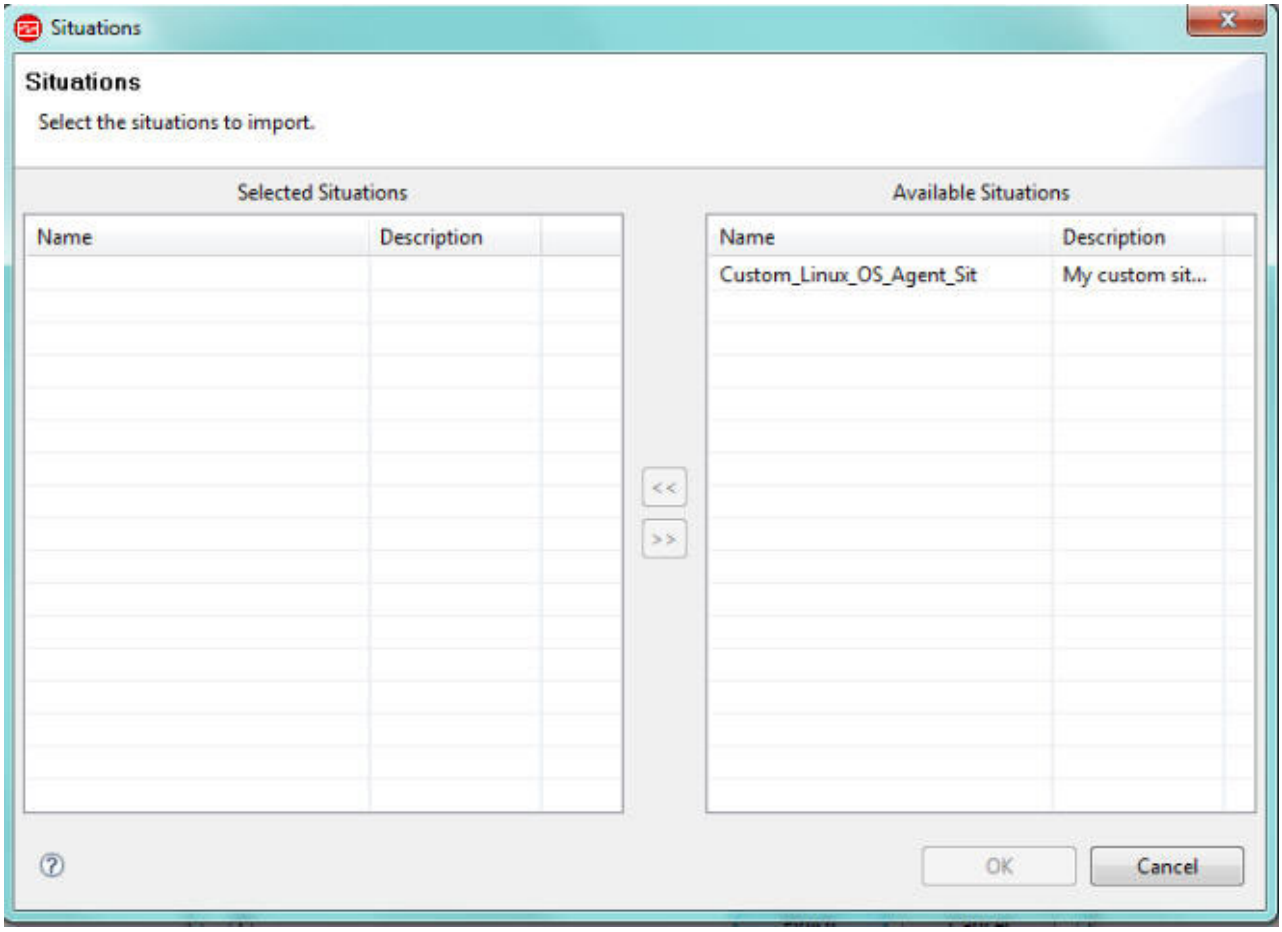


Figure 301. Importing situations

7. In the Queries window (Figure 302 on page 604), select the queries that you want to import from the Available Queries list. Click << to add them to the Selected Queries list and click **OK**.

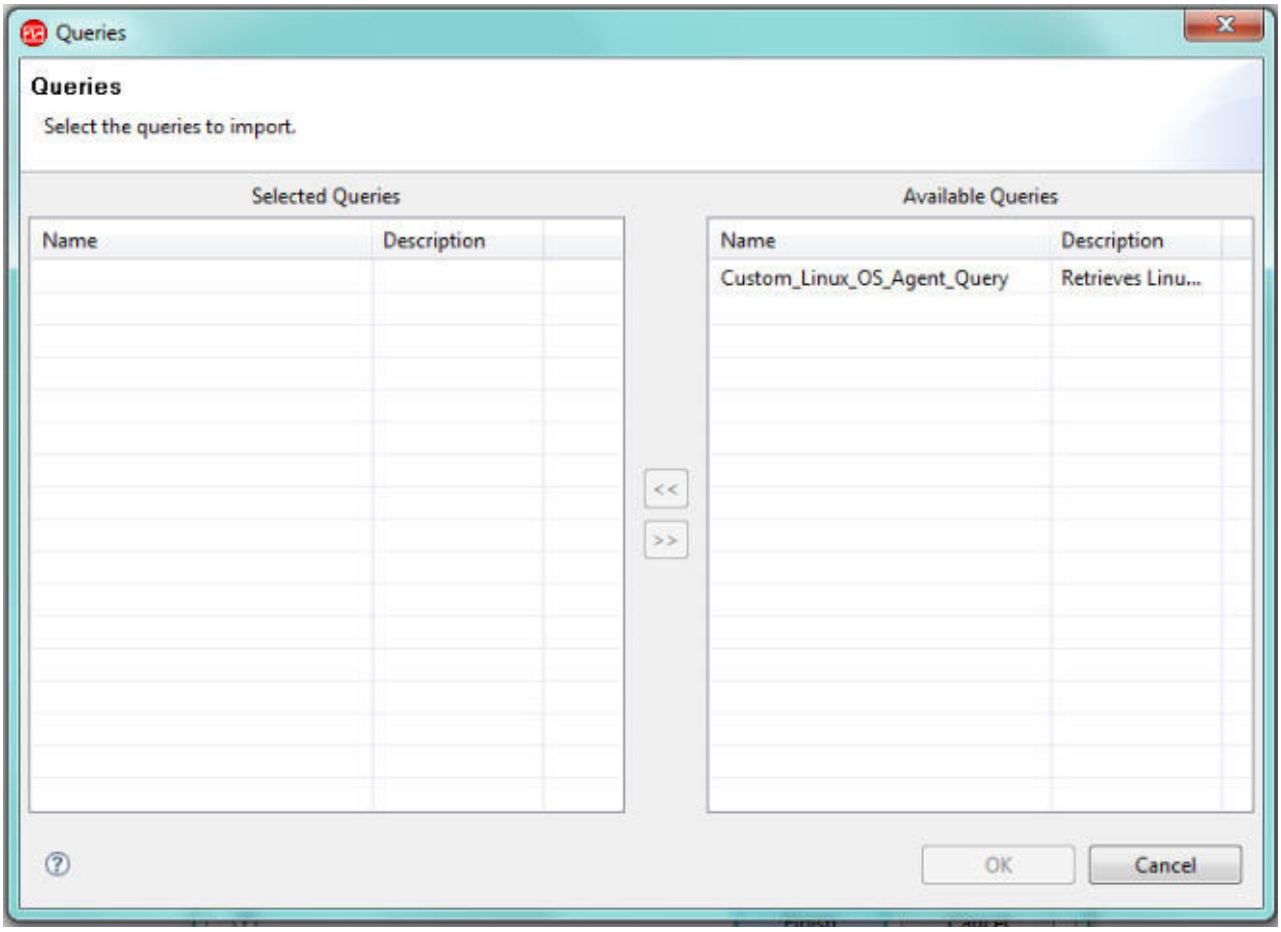


Figure 302. Importing queries

8. In the Take Actions window (Figure 303 on page 605), Choose the Take Action commands that you want to import from the Available Take Actions list. Click << to add them to the Selected Take Actions list and click **OK**.

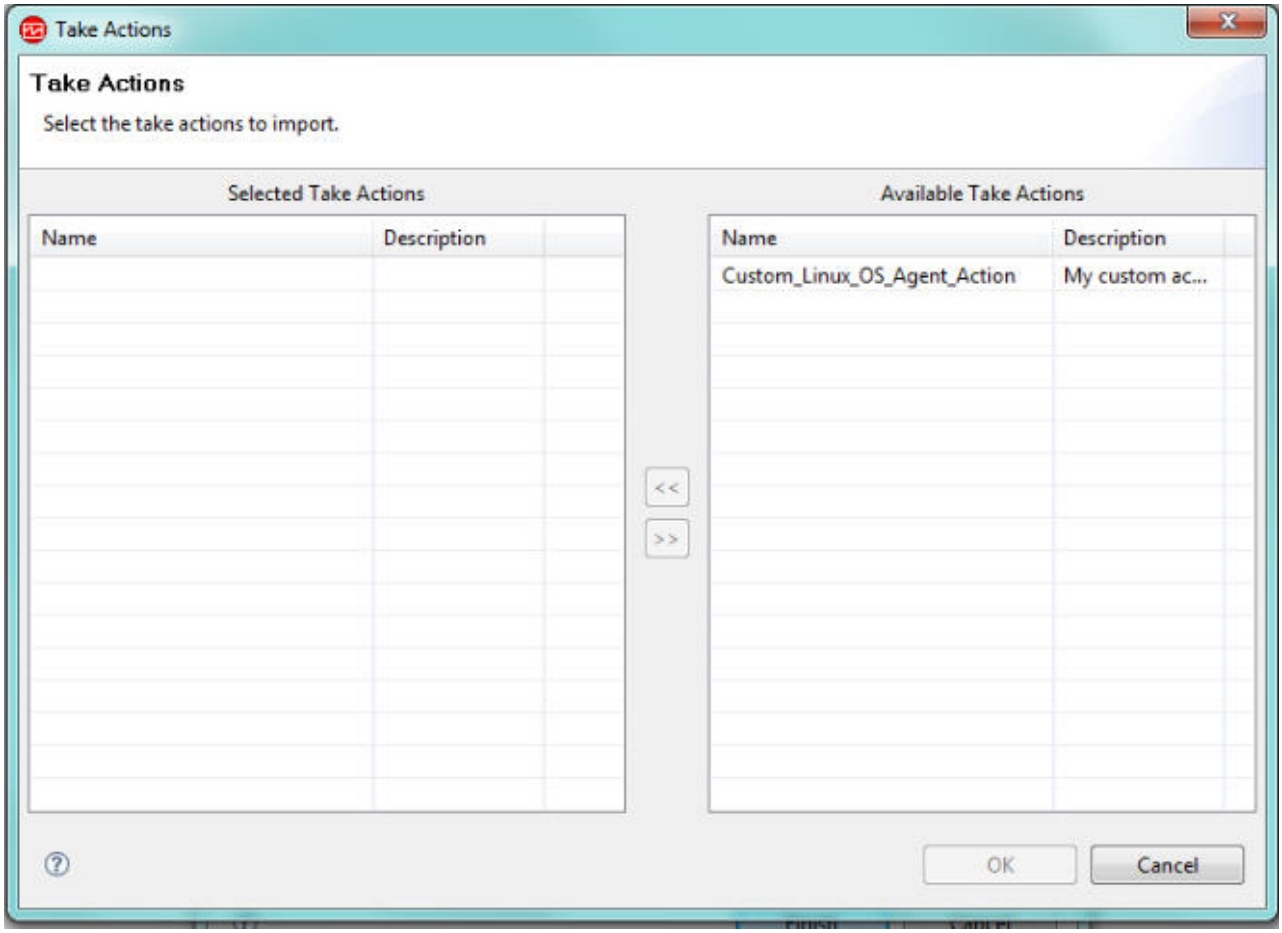


Figure 303. Import Take Action commands

The support files for the agent are put in the project under the appropriate folder.

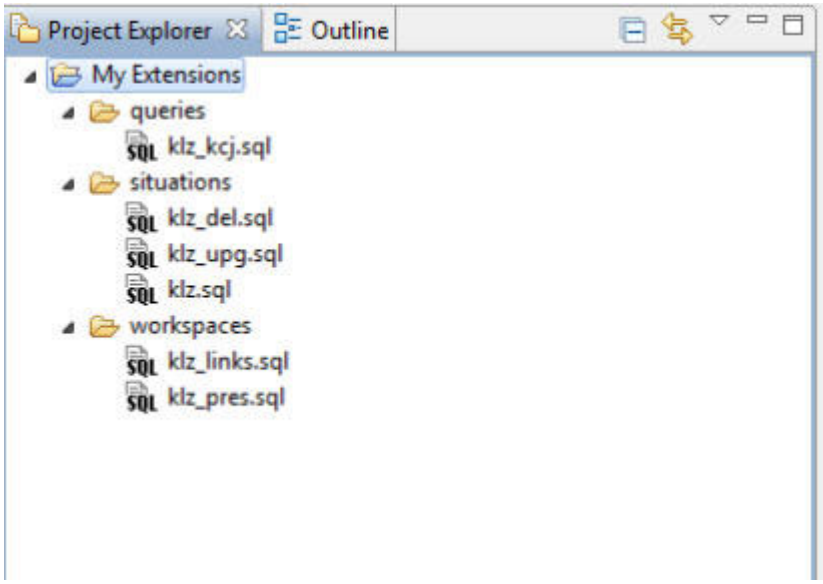


Figure 304. Agent support files

What to do next

You can repeat this process for as many different agents as you want. The Agent Builder creates a single installation image from all of the support files in the Application Support Extension project.

Generating the Application Support Extension installation image

Generate an Application Support Extension installation image.

Procedure

1. Right-click on the Application Support Extension project and select **IBM Tivoli** > **Create Application Support Extension Install Image**.
2. In the Application Support Extension Information window, enter the directory where the image is to be placed.

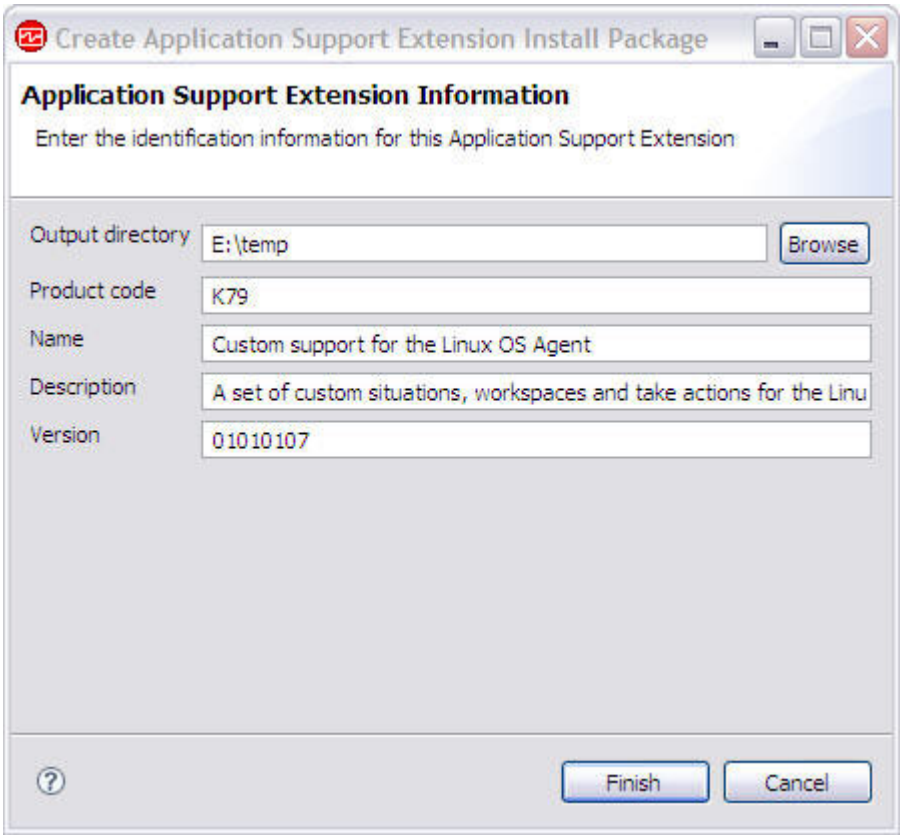


Figure 305. Application Support Extension Information window

3. Your Application Support Extension must have its own product code. Enter the registered product code for your new agent. You can use one of the product codes that are reserved for use with the Agent Builder. The allowed values are `K00-K99`, `K{0-2}{A-Z}`, and `K{4-9}{A-Z}`.

   Note: These values are for internal use only and are not intended for agents that are to be shared or sold. If you are creating an agent to be shared with others, you must send a note to toolkit@us.ibm.com to reserve a product code. The request for a product code must include a description of the agent to be built. A product code is then assigned, registered, and returned to you. When you receive the three-letter product code, you are told how to enable the Agent Builder to use the assigned product code.
4. Enter the name of the Application Support Extension.

5. Enter a description of the Application Support Extension.
6. Enter a version for the Application Support Extension in the `VVRRMMFF` format where `vv` = version number; `rr` = release number; `mm` = modification number (fix pack number); and `ff` = interim fix number.
7. Click **Finish**.

Installing your Application Support Extension

Install your Application Support Extension

Procedure

1. Transfer your image to your Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server servers.
2. To install the Tivoli Enterprise Monitoring Server support, run one of the following commands:
   - On Windows: `install`*`KXX`*`TEMSSupport.bat`
   - On UNIX: `install`*`KXX`*`TEMSSupport.sh`

   The format for the command is as follows:

   ```
   installKXXTEMSSupport[.bat | .sh] <ITM Install Directory> [-s tems_host]
    [-u tems_user]  \[-p tems_password]
   ```
3. To install the Tivoli Enterprise Portal Server support, run one of the following commands:
   - On Windows: `install`*`KXX`*`TEPSSupport.bat`
   - On UNIX: `install`*`KXX`*`TEPSSupport.sh`

   The format for the command is as follows:

   ```
   installKXXTEPSSupport[.bat | .sh] <ITM Install Directory> [-r]
   ```

   where `-r` indicates that the Tivoli Enterprise Portal Server must be restarted after installation

Converting a Solution Install Project to an Application Support Extension project

Convert an existing **Solution Install Project** to an Application Support Extension project

About this task

If you have an existing **Solution Install Project** that you want to convert to an Application Support Extension project, complete the following steps:

Note: In the **Solution Install Project** only Support files are migrated.

Procedure

1. Right-click on the **Solution Install project** and select **IBM Tivoli** > **Convert Solution Install Project**.
2. Enter the name of a new Application Support Extension project or select an existing one from the list
3. Click **Finish**.

Appendix E. Cognos data model generation

Agent Builder can generate a Cognos data model for each agent. Use the data model to import agent information into the Cognos Framework Manager for report creation.

This Cognos data model can be opened and viewed in the Framework Manager, which builds a model package to be published into Tivoli Common Reporting. The data model can also be customized or modified within the Framework Manager before publication.

When a report is created, Agent Builder also allows for a final report package to be imported into the Agent Builder project. This feature enables future agent projects to be generated with the reports that are already part of the agent package. The reports that are packaged as part of the agent installation image can be imported into Tivoli Common Reporting in your production environment.

Note: In this documentation, note the following convention:

- K*xx* or k*xx* refers to the product code given to the agent, for example, k99.
- *dbType* refers to the database that is being used by the Tivoli Data Warehouse, for example, DB2.

Prerequisites to generating a Cognos data model

Complete these tasks before you generate a Cognos data model

About this task

Note:

- These steps must be completed only one time, as all future data models generated with Agent Builder will use this environment.
- It is advisable to create an isolated development environment for agent testing and report creation.

Procedure

1. Install and configure a (“Tivoli Data Warehouse”).
2. Create tables and Procedures in the Tivoli Data Warehouse.
   a. “Create tables and Procedures in the Tivoli Data Warehouse.”
   b. “Populating the Tivoli Data Warehouse with the Tivoli Reporting and Analytics Model” on page 611.
3. Install and configure (“Tivoli Common Reporting” on page 612).
4. Install and configure the (“Framework Manager” on page 612).

Tivoli Data Warehouse

About Tivoli Data Warehouse.

To create reports, you need a Tivoli Data Warehouse, a Warehouse Proxy agent, and a Summarization and Pruning agent, to be installed and configured in your environment. For more information, see the *IBM Tivoli Monitoring Installation and Setup Guide*.

Create tables and Procedures in the Tivoli Data Warehouse

Create or alter the `ManagedSystem` Table and Stored Procedure in the Tivoli Data Warehouse

© Copyright IBM Corp. 2007, 2013

About this task

The generated Cognos data model includes a `ManagedSystem` table which is used to define a `ManagedSystem` dimension. The `ManagedSystem` dimension allows reports to be created that can correlate managed systems. For example, if the agent is a subnode agent, the dimension can be used to determine the subnodes that exist for a specific agent instance.

The `ManagedSystem` table is not created by the Tivoli Data Warehouse. Therefore, when an agent is generated in Agent Builder, SQL scripts are generated for each database platform that will:

- Create the `ManagedSystem` table. Use this script if the table does not exist in the Tivoli Data Warehouse.
- Edit the `ManagedSystem` table. Use this script if the table exists in the Tivoli Data Warehouse. Other reporting products can create the `ManagedSystem` table, but they do not create it with all of the required columns.
- Create a stored procedure that populates the `ManagedSystem` table from tables in the Tivoli Data Warehouse.

Run these scripts one time only.

Running DB2 Scripts to Create tables and Procedures in the Tivoli Data Warehouse:

For a DB2 database, use these scripts to create tables in the Tivoli Data Warehouse

Before you begin

The scripts for DB2 are in the following directory:

`reports/db2/K`*`xx`*`/reports/cognos_reports/itmk`*`xx`*`/db_scripts`

Procedure

1. The generated scripts (`create_table.sql`, `alter_table.sql`, and `create_procedure.sql`) all use *`itmuser`* as the Tivoli Data Warehouse user ID. If *`itmuser`* is not the Tivoli Data Warehouse user ID in your environment, change all occurrences of *`itmuser`* to the correct user ID.
2. Connect to the Tivoli Data Warehouse as the Tivoli Data Warehouse User:

   ```
   db2 connect to <Tivoli Data Warehouse alias name> user
   <Tivoli Data Warehouse user id> using  <password>
   ```

3. Determine whether the ManagedSystem table exists:

   ```
   db2 "select count(*) from sysibm.systables where name = 'MANAGEDSYSTEM'
   and creator=upper ('<Tivoli Data Warehouse user id>')"
   ```

4. Create or alter the table.
   - If the query returns `1`, the table exists. Run the alter script:

     ```
     db2 -tvf alter_table.sql
     ```

   - If the query returns `0`, the table does not exist. Run the create script:

     ```
     db2 -tvf create_table.sql
     ```

5. Run the script to create the stored procedure:

   ```
   db2 -td@ -f create_procedure.sql
   ```

Running Oracle Scripts to Create tables and Procedures in the Tivoli Data Warehouse:

For an Oracle database, use these scripts to create tables in the Tivoli Data Warehouse

Before you begin

The scripts for Oracle are in the following directory:

`reports/oracle/K`*`xx`*`/reports/cognos_reports/itmk`*`xx`*`/db_scripts`

Procedure

1. The generated scripts (`create_table.sql`, `alter_table.sql`, and `create_procedure.sql`) all use *itmuser* as the Tivoli Data Warehouse user ID. If *itmuser* is not the Tivoli Data Warehouse user ID in your environment, change all occurrences of *itmuser* to the correct user ID.
2. Start `sqlplus`:

   ```
   sqlplus <IBM Tivoli Monitoring user ID>/<password>@
   <Tivoli Data Warehouse SID>
   ```
3. Determine whether the ManagedSystem table exists:

   ```
   select count(*) from user_tables where table_name = 'MANAGEDSYSTEM';
   ```
4. Create or alter the table.
   - If the query returns `1`, the table exists. Run the alter script:

     ```
     @<path to alter_table.sql>;
     ```
   - If the query returns `0`, the table does not exist. Run the create script:

     ```
     @<path to create_table.sql>;
     ```
5. Run the script to create the stored procedure:

   ```
   @<path to create_procedure.sql>;
   ```

Running SQL Server 2005 and 2008 Scripts to Create tables and Procedures in the Tivoli Data Warehouse:
Before you begin

The scripts for SQL Server are in the following directory:

```
reports/mssql/Kxx/reports/cognos_reports/itmkxx/db_scripts
```

Procedure

1. The generated scripts (`create_table.sql`, `alter_table.sql`, and `create_procedure.sql`) all use *itmuser* as the Tivoli Data Warehouse user ID. If *itmuser* is not the Tivoli Data Warehouse user ID in your environment, change all occurrences of *itmuser* to the correct user ID.
2. Determine whether the `ManagedSystem` table exists:

   ```
   osql -S <Server> -U <Tivoli Data Warehouse user ID> -P <password> -d
   <Tivoli Data Warehouse database name> -Q "Select count(*)
   from INFORMATION_SCHEMA.TABLES where table_name = 'ManagedSystem'"
   ```
3. Create or alter the table.
   - If the query returns `1`, the table exists. Run the alter script:

     ```
     osql -S <Server> -U <Tivoli Data Warehouse user ID> -P <password> -d
     <Tivoli Data Warehouse database name> -I -n -i <path to alter_table.sql>
     ```
   - If the query returns `0`, the table does not exist. Run the create script:

     ```
     osql -S <Server> -U <Tivoli Data Warehouse user ID> -P <password> -d
     <Tivoli Data Warehouse database name> -I -n -i <path to create_table.sql>
     ```
4. Run the script to create the stored procedure:

   ```
   osql -S <Server> -U <Tivoli Data Warehouse user ID> -P
   <password> -d <Tivoli Data Warehouse database name>
   -I -n -i <path to create_procedure.sql>
   ```

Populating the Tivoli Data Warehouse with the Tivoli Reporting and Analytics Model

Use provided database scripts to populate the Tivoli Data Warehouse

About this task

Tivoli Reporting and Analytics Model (TRAM) contains the base-set of knowledge that is common to all reporting packages. TRAM is installed by a set of scripts unique to each database. The necessary scripts

for populating each supported database are included in the agent installation image, within the reports directory. Use the following procedure to create Tivoli Reporting and Analytics Model Common Dimensions in Tivoli Data Warehouse.

Procedure

1. Browse to the Tivoli Reporting and Analytics Model database scripts.
2. Extract the agent package.
   - On Windows systems, agent package is k*xx*.zip.
   - On Linux and UNIX systems, agent package is k*xx*.tgz.
3. Go to the appropriate database scripts.
   - DB2 scripts are in the Agent package at:
     reports/db2/K*xx*/reports/cognos_reports/itmk*xx*/db_scripts
   - Oracle scripts are in the Agent package at:
     reports/oracle/K*xx*/reports/cognos_reports/itmk*xx*/db_scripts
   - Microsoft SQL Server scripts are in the Agent package at:
     reports/mssql/K*xx*/reports/cognos_reports/itmk*xx*/db_scripts
4. Run the database scripts to generate the common dimensions within the Tivoli Data Warehouse. Each script set provides a readme file for usage instructions.
5. Verify that the scripts added the following tables to the Tivoli Data Warehouse:
   "Computer System", WEEKDAY_LOOKUP, MONTH_LOOKUP, TIMEZONE_DIMENSION, TIME_DIMENSION

Tivoli Common Reporting

Tivoli Common Reporting contains the Cognos Business Intelligence engine, which contains elements to assist with the creation of agent reports.

Tivoli Common Reporting must be installed and configured with a data source that connects to the Tivoli Data Warehouse.

Installing Tivoli Common Reporting

You must install Tivoli Common Reporting. Versions 1.3, 2.1, 2.1.1 or later are supported. For information about installing Tivoli Common Reporting, see Installing Tivoli Common Reporting in the *Tivoli Common Reporting User's Guide*.

Configuring Tivoli Common Reporting

You must configure Tivoli Common Reporting. For information about configuring Tivoli Common Reporting, see Configuring Tivoli Common Reporting in the *Tivoli Common Reporting User's Guide*.

Create a data source between the Tivoli Data Warehouse and Tivoli Common Reporting. For more information, see Configuring database connection in the *Tivoli Common Reporting User's Guide*. Click the appropriate database type. Note the name that is given to the data source. The default is **TDW**.

Note: The data source name must match the name in the **Data source** field of the Cognos Information page. For more information about the Cognos Information page, see "Cognos information" on page 42.

Framework Manager

Framework Manager is an application that ships with the Tivoli Common Reporting application, but must be installed and configured separately.

Framework Manager is used to view and modify data models and to publish data models to Tivoli Common Reporting

Installing Framework Manager

You must install Framework Manager. Versions 8.4, 8.4.1 or later are supported.

The Framework Manager ships with Tivoli Common Reporting, but must be manually installed. Tivoli Common Reporting 1.3 ships with Framework Manager 8.4. Tivoli Common Reporting 2.1 and 2.1.1 ships with Framework Manager 8.4.1. For information about installing Framework Manager, see Installing Framework Manager in the *Tivoli Common Reporting User's Guide*.

Configuring Framework Manager

You must configure Framework Manager. For information about configuring Framework Manager, see Configuring Framework Manager in the *Tivoli Common Reporting User's Guide*.

Creating reports

Use the Framework Manager to publish the agent model, and Report Studio to begin creating reports.

Before you begin

When the agent is completed, it must be installed into the Tivoli Monitoring environment. In addition, historical collection for the agent must be configured and the agent be run for at least one warehouse upload interval. Summarization must be configured, and the summarization setting choices that are made in Tivoli Monitoring must be identical to the summarization choices made in the Agent Builder. The Summarization and Pruning agent must run at least one time after the agent's data is uploaded to the warehouse.

1. Install, configure, and start your agent.
2. Create and distribute to the agent a historical collection for each attribute group you want to create a report for.

   Note: The warehouse upload interval defaults to daily. However, you might want to shorten this interval.

   For information about configuring historical collection, see Managing historical data in the *IBM Tivoli Monitoring Administrator's Guide*.
3. In Tivoli Monitoring, configure summarization for all of the attribute groups you created historical collections for in Step 2.

   Note: When you configure historical collection and summarization, you must wait enough time for data to end up in the summary tables.

   Note: By default, the Summarization and Pruning agent is configured to run one time a day at 2 a.m. You might want to change this setting. For example, you can configure it to run hourly. For information about configuring the Tivoli Data Warehouse, see Setting up data warehousing in the *IBM Tivoli Monitoring Installation and Setup Guide*.

About this task

Generating an agent in Agent Builder creates an entire Framework Manager project, which includes the data model and the Framework Manager project file. Framework Manager can open the project file directly, which opens the data model for modification, customization, or publication.

Procedure

Note: The generated data model for the agent contains all of the summary time dimensions for each attribute group: hourly, daily, weekly, monthly, quarterly, and yearly. The dimensions exist only in the

Tivoli Data Warehouse for the agent if summarization and pruning is configured for the agent. And also if the dimensions are selected, and if the Summarization and Pruning agent, created, and populated the tables. Reports can be defined and published into Tivoli Common Reporting that use dimensions that do not exist. Such reports do not function until the summary tables are created by the Summarization and Pruning agent.

1. Open the Agent Data Model in Framework Manager:
   a. Open the Framework Manager.
   b. From the **Welcome** page, click **Open a project**.

      Tip: If you are in Framework Manager, click **Open** from the **File** menu.
   c. Browse to the Agent data model.
      - For DB2:

        `reports/db2/Kxx/model/`
      - For Oracle:

        `reports/oracle/Kxx/model/`
      - For Microsoft SQL Server:

        `reports/mssql/Kxx/model/`
   d. Select the agent project file, `Kxx.cpf`.

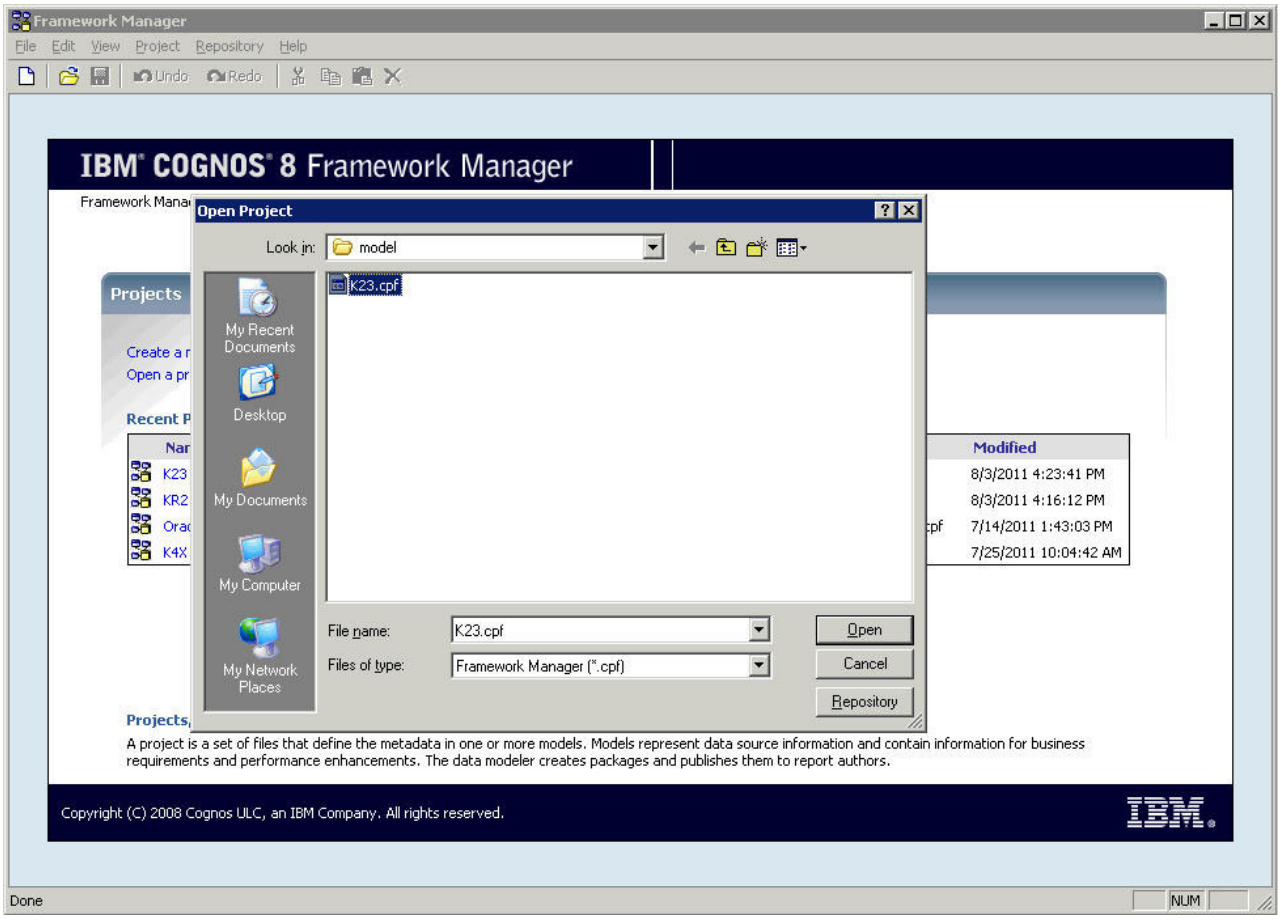


Figure 306. Selecting agent project file

Note: When an agent project is opened in Framework Manager, the agent name is listed under the Recent Projects.

2. Populate the Managed System Table. For more information, see "Populating the ManagedSystem Table" on page 618
3. Use the Framework Manager to publish the Agent Model to Tivoli Common Reporting
   a. Open the Framework Manager.
   b. Open the Agent project.
   c. Expand **Packages** in the navigation tree.
   d. Right-click the agent package and select **Publish Packages**.

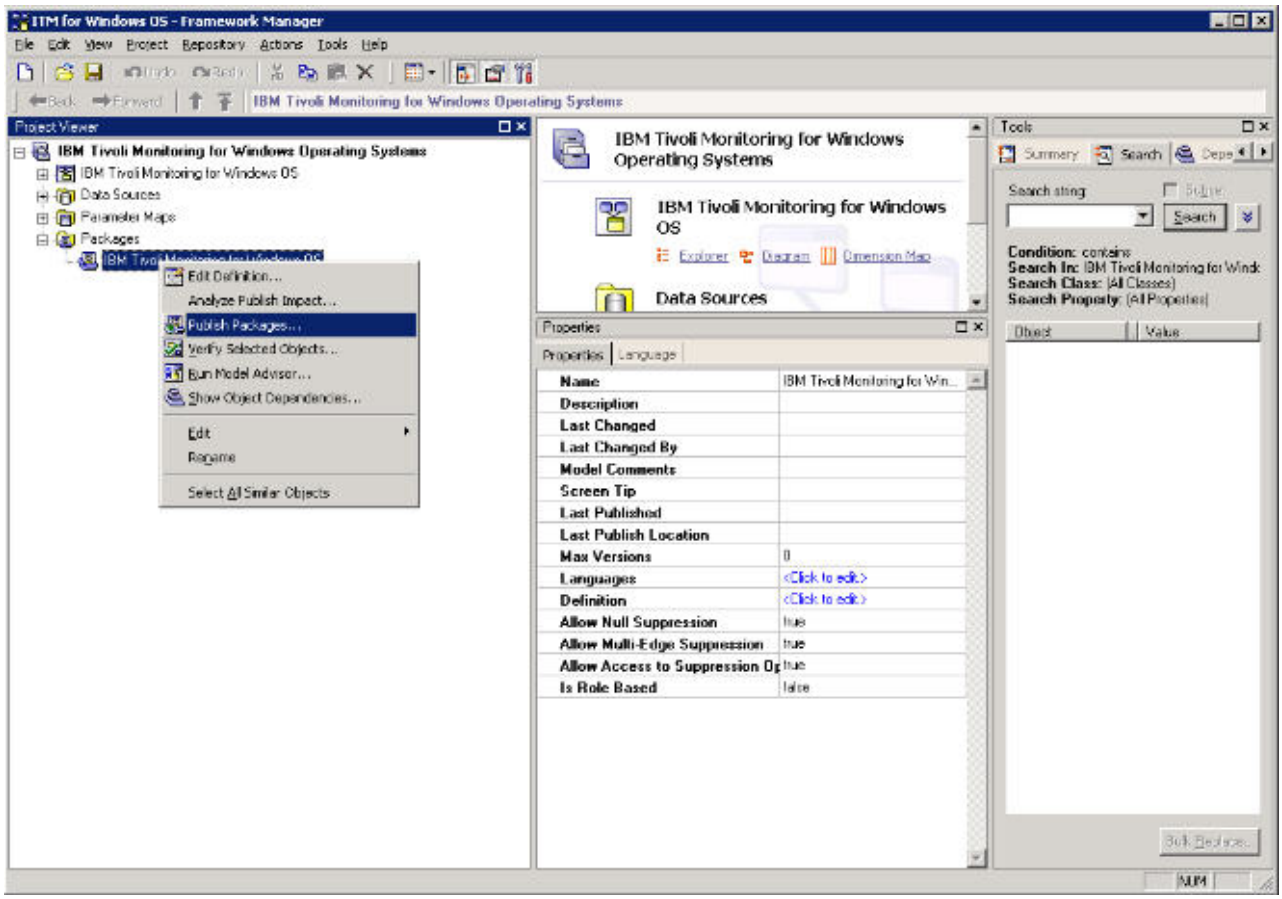


Figure 307. Selecting Publish Packages

4. Use Report Studio to create new reports or templates.
   a. Log on to Tivoli Common Reporting.
   b. Browse to Public Folders, expand **Reporting** in the navigation panel, and select **Common Reporting**.

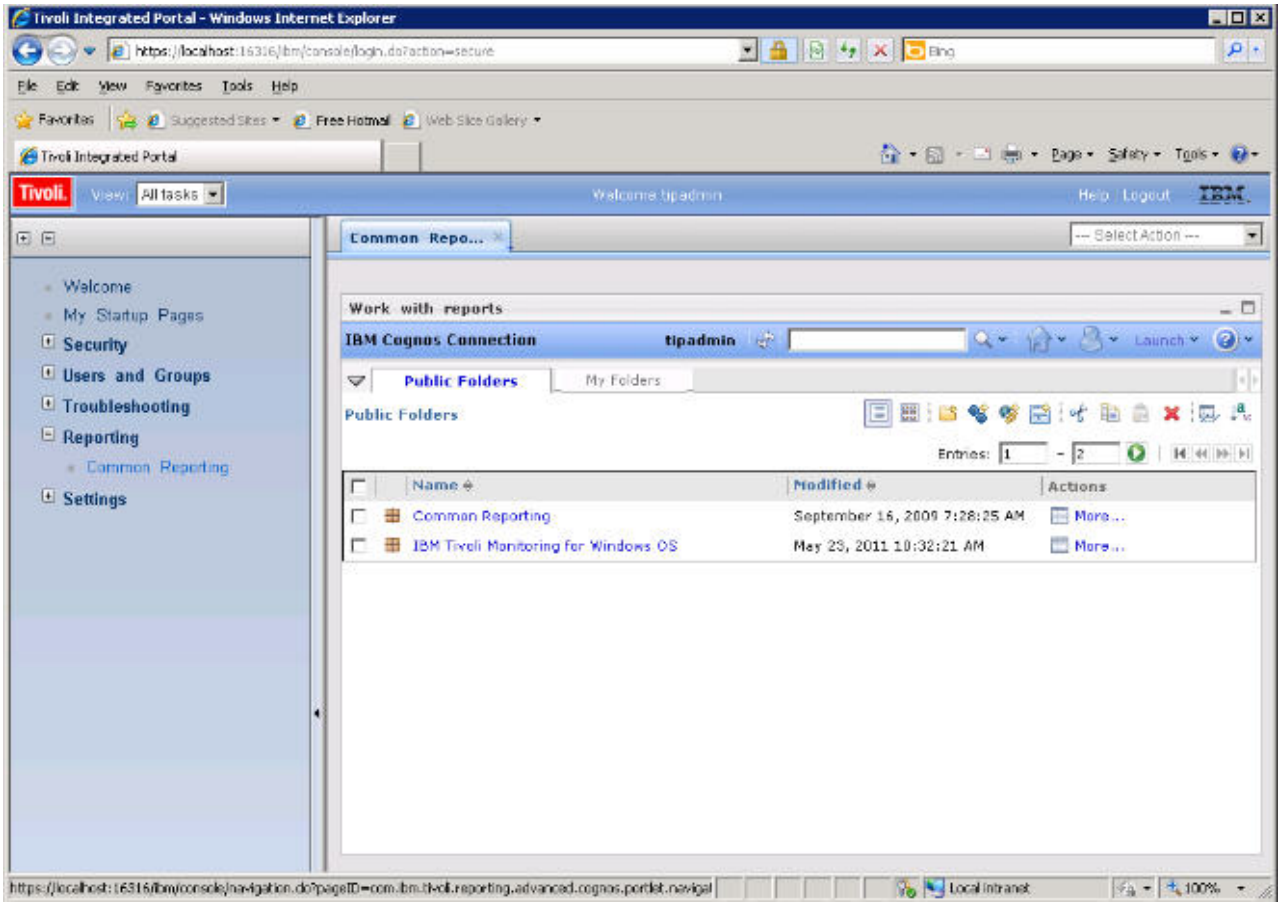


Figure 308. Selecting Common Reporting

c. Select your Tivoli Monitoring agent from the list provided.

d. Open the report creation tool, by clicking the Launch menu and selecting **Report Studio** or **Query Studio**.

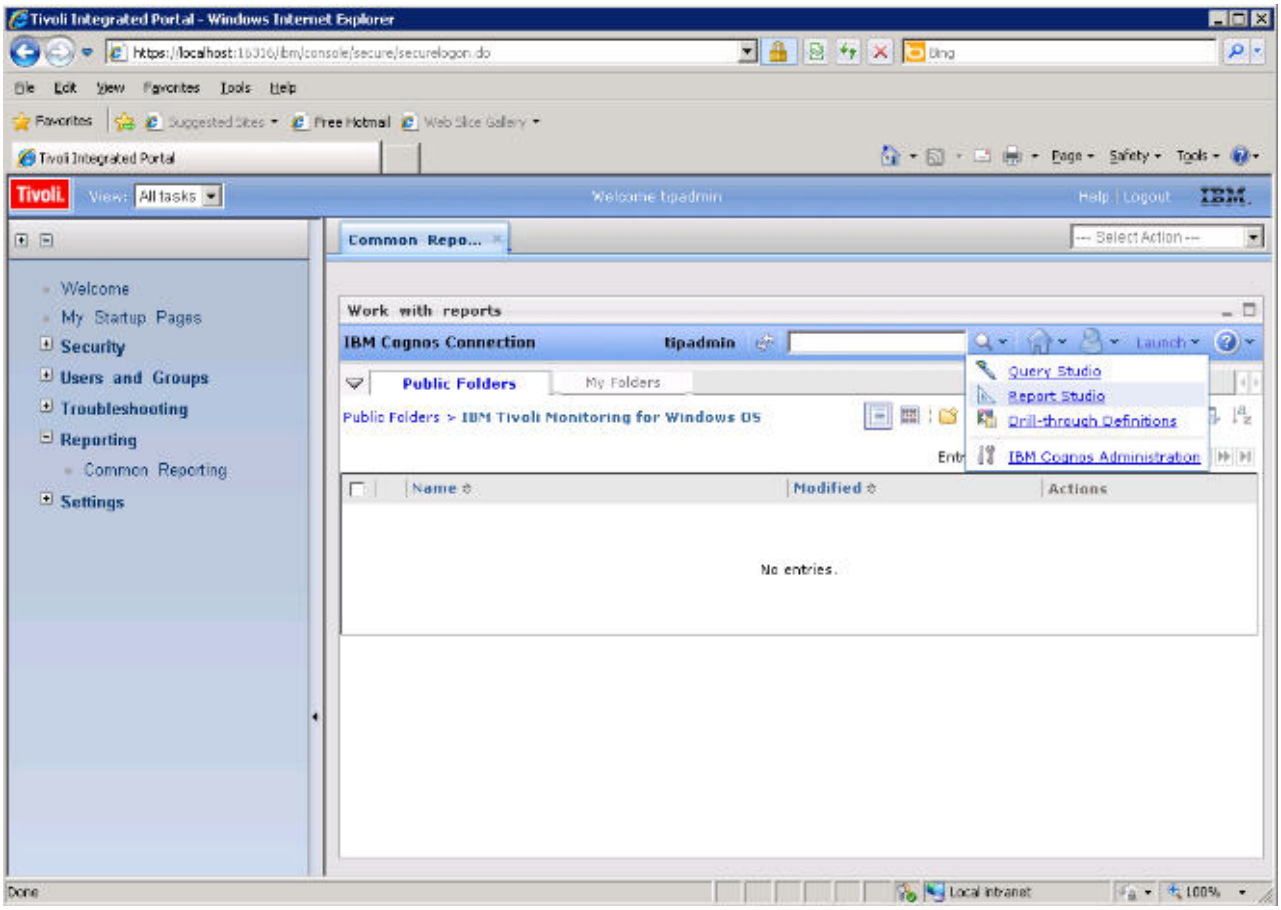


Figure 309. Selecting Report Studio

What to do next

You can use the Report Studio to create new reports or templates, or you can modify an existing report or template.

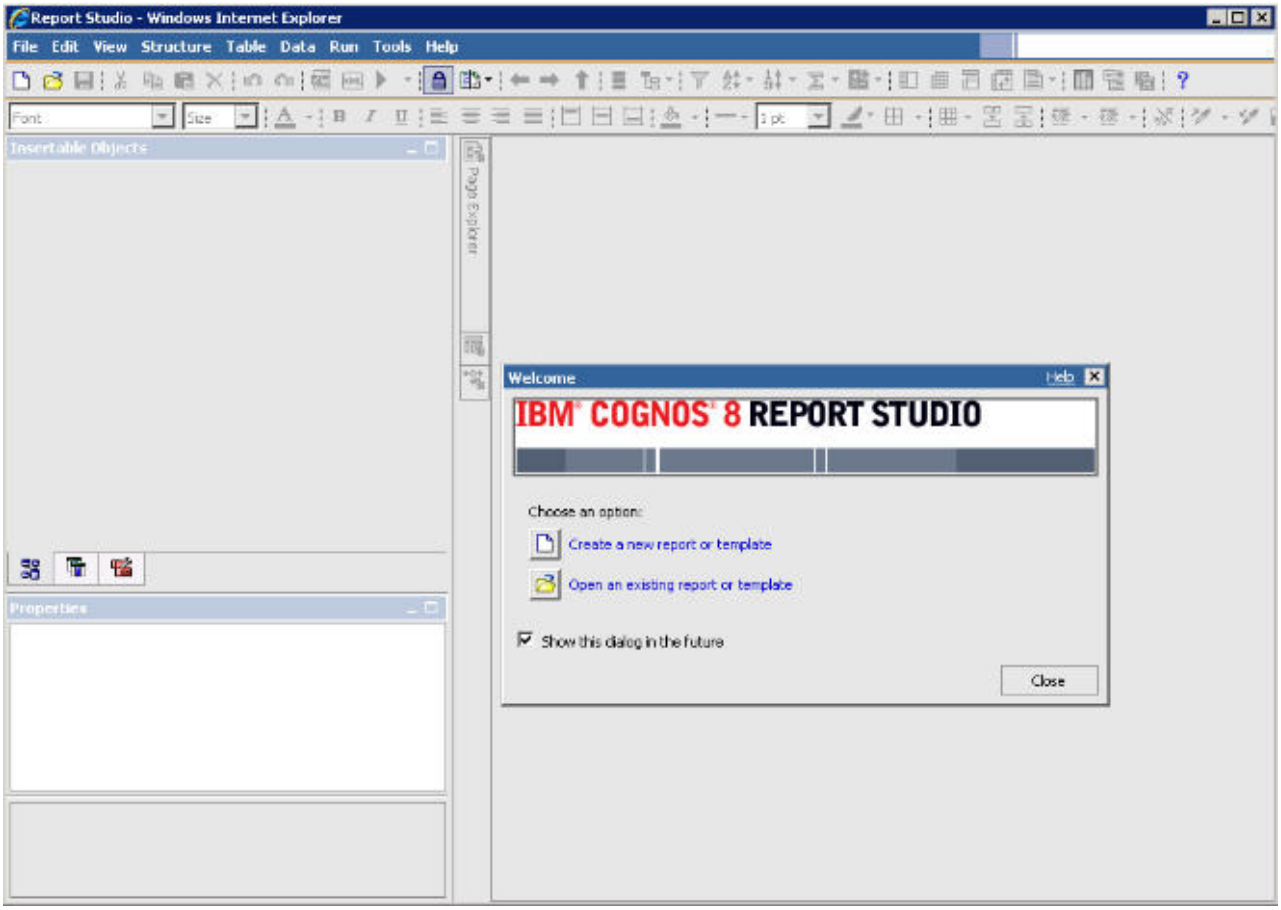


Figure 310. Report Studio

For more information, see the IBM Tivoli Common Reporting Information Center.

Populating the ManagedSystem Table

The `MangedSystem` table is populated by using the `kqz_populate_msn` stored procedure.

For more information, see "Running the DB2 stored procedure" on page 619. This procedure must be run periodically so that the `ManagedSystem` table contains the current list of managed system names.

The stored procedure reads the following historical tables in the Tivoli Data Warehouse if they exist:

- The agent's Performance Object status table
- The agent's availability table. Agents that monitor processes or services have an availability table.
- The agent's discovery tables. Subnode agents create discovery tables.

Historical collection must be started on a particular set of attribute groups. A set of scripts is generated that creates and starts historical collection for these attribute groups. If you do not want to use the scripts, the list of attribute groups is listed in the comment header block of the script.

Sample scripts are created that show which tables must have historical collection enabled:

- `reports/configuretdw.sh`
- `reports/configuretdw.bat`

The following table describes the required arguments:

Note: You must specify either `-n` or `-m`, but not both.

Table 53. Required arguments

Argument	Description
`-h candle_home`	The Tivoli Monitoring installation path.
`-u teps_user`	The Tivoli Enterprise Portal Server user to log in as when you create the historical collections.
`-n tems_name`	The Tivoli Enterprise Monitoring Server where the collections must be started. More than one Tivoli Enterprise Monitoring Server can be specified by using a space separated list. If you specify more than one Tivoli Enterprise Monitoring Server, put the list in quotation marks. For example, `-n "tems1 tems2"`
`-m managed_system_group_or_manged_system`	The managed system group or managed system name against which the collection must be started. More than one managed system group or managed system can be specified by using a space separated list. If you specify more than one managed system group or managed system, put the list in quotation marks. For example, `-m "msg1 msg2"`

The following table describes the optional arguments:

Table 54. Optional arguments

Argument	Description
`-s teps_host`	The host name or IP address of the Tivoli Enterprise Portal Server. If not specified, the default is localhost.
`-p teps_password`	The password for the Tivoli Enterprise Portal Server user that is specified with the `-u` option. If not specified, the script prompts for the password
`-c historical_collection_interval`	The historical collection interval to use when you start the historical collections. If not specified, the default is `1h` (1 hour). The valid values are: `15m`, `30m`, `1h`, `12hor 1d`, where `m` is minutes, `h` is hours and `d` is days.
`-r pruning_interval`	The pruning interval to use for the historical data. The historical data must be pruned so that the tables do not continue to grow in size. If not specified, the default is `2d`(2 days). Use `d` for days, `m` for months, `y` for years.

After historical collection is started, the `kqz_populate_msn` stored procedure must be run periodically. The stored procedure is run periodically so that the `ManagedSystem` table contains the most current list of managed systems in the Tivoli Monitoring environment.

Running the DB2 stored procedure

Run a stored procedure on DB2.

About this task

Perform the following steps to run the stored procedure on DB2:

Procedure

1. Connect to the Tivoli Data Warehouse database as the warehouse user:

```
connect to <Tivoli Data Warehouse database alias> user
<Tivoli Data Warehouse user id> using <password>
```

2. Run the stored procedure:

```
db2 "call <Tivoli Data Warehouse schema>.kqz_populate_msn
('<three letter product code for the agent>')"
```

Running the Oracle stored procedure

Run a stored procedure on Oracle.

About this task

Perform the following steps to run the stored procedure on Oracle:

Procedure

1. Start sqlplus:

```
sqlplus <Tivoli Data Warehouse user id>/<password>@
<Oracle SID>
```

2. Run the stored procedure:

```
execute kqz_populate_msn('<three letter product code for the agent>');
```

Running the stored procedure on SQL Server 2005 and 2008

Run a stored procedure on SQL Server.

About this task

Perform the following steps to run the stored procedure on SQL Server 2005 and 2008:

Procedure

Run the stored procedure:

```
osql -S <server> -U <Tivoli Data Warehouse id> -P
<Tivoli Data Warehouse password> -d
<Tivoli Data Warehouse database name> -Q "EXEC
[<Tivoli Data Warehouse schema>].[kqz_populate_msn]
@pv_productcode = N'<three letter product code>'"
```

Exporting reports and data models from Tivoli Common Reporting

Export reports and data models from Tivoli Common Reporting.

Procedure

1. Log in to the Tivoli Common Reporting.
2. Go to Public Folders, and under **Reporting** in the navigation panel select **Common Reporting**.
3. In the Work with reports section, click the **Launch** menu and select **IBM Cognos Administration**.
4. Click the **Configuration** tab.
5. Click **Content Administration**.

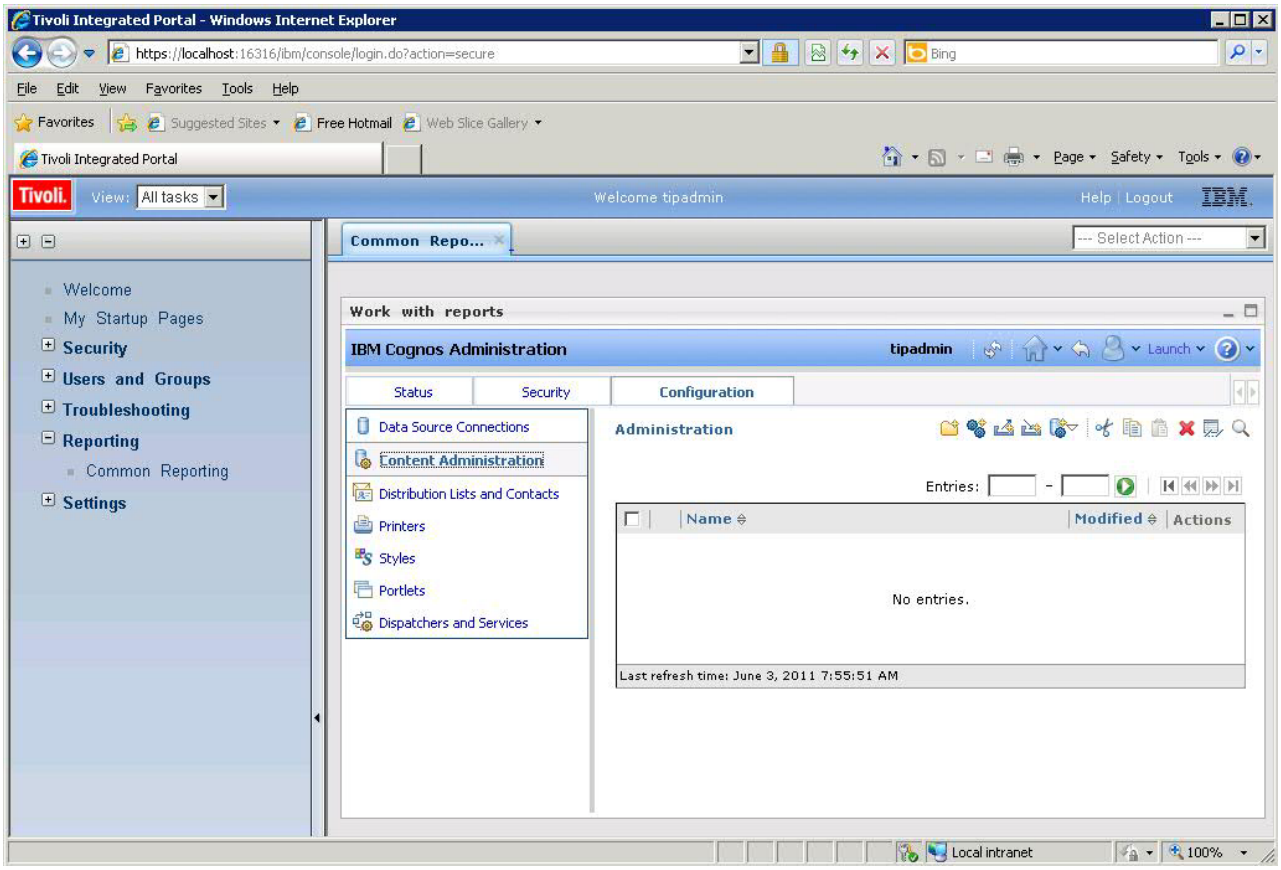


Figure 311. The Content Administration tab

6. Click the **New Export** icon to export a new package.
7. Name the package. Optionally, you can add a screen tip and description.
8. Select **Select public folders and directory content** .
9. In the Public Folders dialog, click the **Add** link.
10. Move your agent package to **Selected entries**.
11. On the last page of the wizard, select **Save Only**. When the wizard completes, the report package is listed on the Content Administration tab.
12. On the Content Administration tab, click the green arrow (Run) to create the compressed `.zip` file.

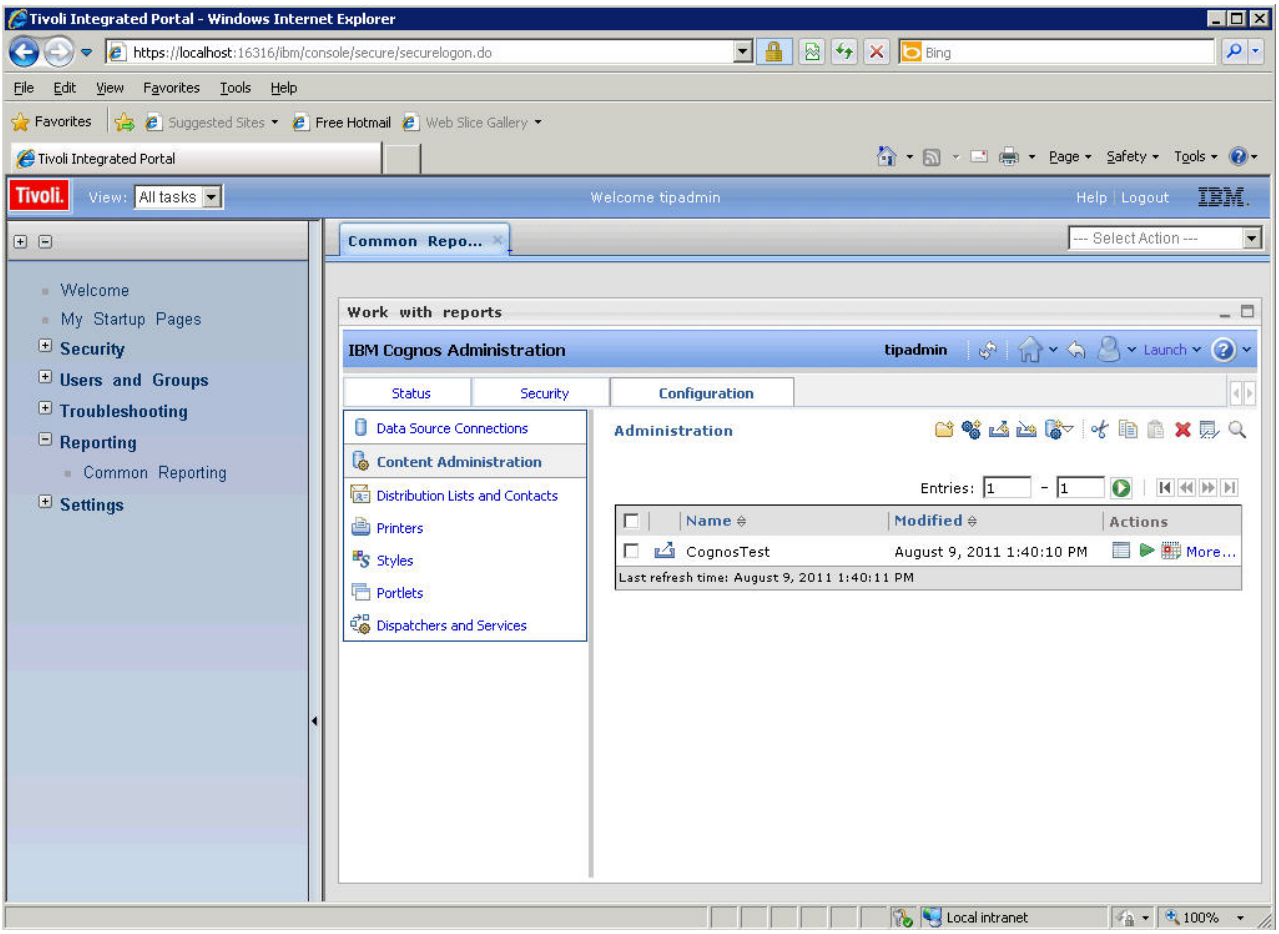


Figure 312. The Content Administration tab with agent package listed

Results

The compressed `.zip` file that is created by the export process is placed in the deployment directory.

- The directory path for Tivoli Common Reporting version 1.3 is:
  `C:\IBM\tivoli\tip\products\tcr\Cognos\c8\deployment`
- The directory path for Tivoli Common Reporting version 2.1 or later is:
  `C:\IBM\tivoli\tipv2Components\TCRComponent\cognos\deployment`

What to do next

For more information about exporting reports, see Exporting Cognos report packages in the *Tivoli Common Reporting User's Guide*.

Importing reports into Agent Builder

When the report package is exported from Tivoli Common Reporting, it can be imported into the Agent Builder project. The report package can then be included in the agent installation image.

Procedure

1. Right-click on the agent project in the Agent Builder.
2. Select **IBM Tivoli** > **Import Report Package**.

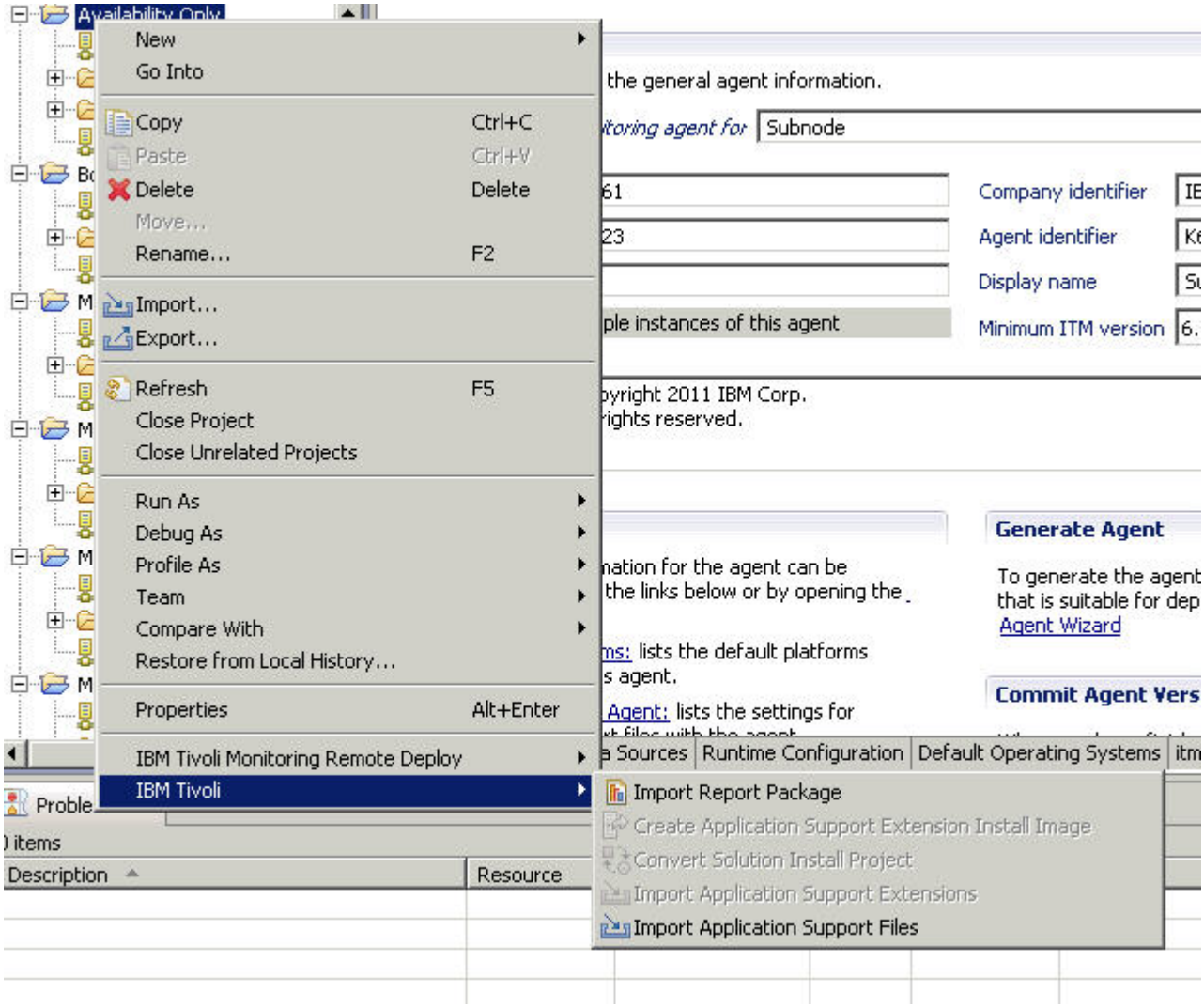


Figure 313. Importing report packages

3. In the Import Report Package window (Figure 314 on page 624), select the **Database Type** on which the report package was created.

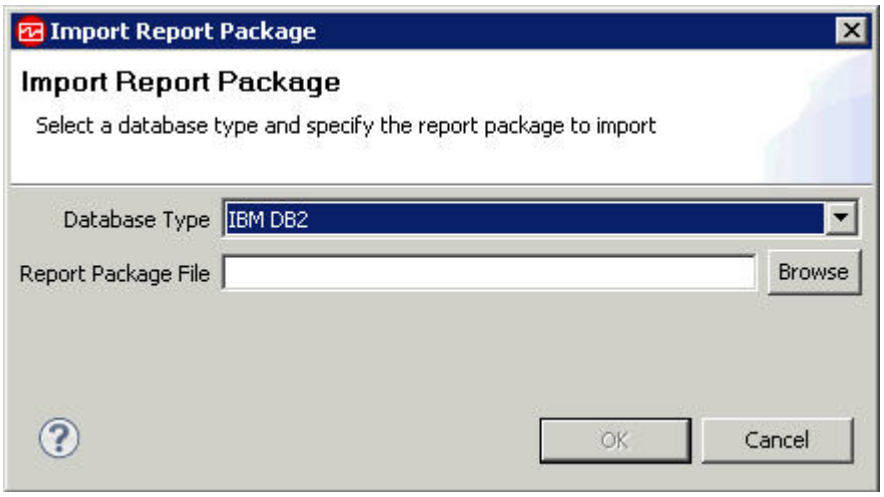


Figure 314. Import Report Package window

4. Enter the fully qualified path to the report package, or click **Browse** to select it.
5. Click **OK**.
6. The report package is now shown in the agent project under the `reports/`*`dbtype`* directory.

   Note: If you create report packages that are database-specific you must import each package into the Agent Builder.

Installing reports from an agent package into Tivoli Common Reporting

Import a report package from your agent to Tivoli Common Reporting

Procedure

1. Follow the steps in the wizard to import a new package from your agent image. In the agent image, the reports are found in: `reports/`*`dbType`*`/K`*`xx`*`/reports/cognos_reports/itmk`*`xx`*`/packages`
2. Copy the reports compressed `zip` file into the Tivoli Common Reporting deployment directory.
   - The directory path for Tivoli Common Reporting version 1.3 is: `C:\IBM\tivoli\tip\products\tcr\Cognos\c8\deployment`
   - The directory path for Tivoli Common Reporting version 2.1 or later is: `C:\IBM\tivoli\tipv2Components\TCRComponent\cognos\deployment`
3. Log in to the Tivoli Common Reporting.
4. Go to Public Folders, and under **Reporting** in the navigation panel select **Common Reporting**.
5. In the Work with reports section, click the **Launch** menu and select **IBM Cognos Administration**.
6. Go to the **Configuration** tab, and open the **Content Administration** section.
7. Click **New Import** to create a package import.
8. Select the agent's reports package.
9. Select the public folders that you want to import.
10. Select save.
11. Click the green (run) arrow to import.

Results

For more information, see Logging in to Tivoli Common Reporting in the *Tivoli Common Reporting User's Guide*.

Appendix F. Upgrading custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6.2 agents

The procedure for upgrading a custom resource model differs in some respects from the procedure that is used for a standard resource model

About this task

If you determine that you want to create an agent to map your IBM Tivoli Monitoring v5.x resource models, complete the procedures for creating an agent in (Chapter 5, "Creating a basic agent," on page 13). When you generate the agent, ensure that you select the **Generate an ITM 5x mapping file** option in the Generate Agent wizard. For more information, see ("Installing the agent locally" on page 387).

For the information about how to upgrade custom resource models, see the *IBM Tivoli Monitoring: Upgrading from IBM Tivoli Monitoring, version 5.1 to version 6.2* document.

The procedure for upgrading a custom resource model differs in some possible respects from the procedure that is used for a standard resource model:

Procedure

- When you do the upgrade, the resource model mapping file that is used by the assess command must be created. Create the file by copying and editing an existing mapping file, or by creating it from scratch.

  Note: For information about the assess command, see the *IBM Tivoli Monitoring: Upgrading from IBM Tivoli Monitoring, version 5.1 to version 6.2* document.
- To evaluate the mapping file that is created in the previous step, you might have to create a Java resource model plug-in class. You can use the Java resource model plug-in class to add your own coding to the standard evaluation used by the Agent Builder.
- If the resource model data collection differs from the collection that is used by the standard resource models, you might be required to create another agent. You can use Agent Builder to create this agent to collect the data. For this agent, you must also create an agent mapping file to enable the migration to work correctly.
- If the agent data collection cannot be provided by standard operating system facilities, you can create a Java agent plug-in class to do the collection.

Results

When these items are developed and tested, the migration follows the same steps as the migration of standard resource models.

© Copyright IBM Corp. 2007, 2013

Appendix G. ICU regular expressions

A description of the specifics of the ICU regular expression implementation.

This reference content is extracted from the *ICU User Guide*. The content describes the specifics of the ICU regular expression implementation. This information is essential if you are using the Agent Builder regular expression feature because different programming languages implement regular expressions in slightly different ways.

Table 55. Regular expression metacharacters

Character	Description
\a	Match a BELL, \u0007
\A	Match at the beginning of the input. Differs from ^ in that \A does not match after a new line within the input.
\b, outside of a [Set]	Match if the current position is a word boundary. Boundaries occur at the transitions between word (\w) and non-word (\W) characters, with combining marks ignored. For more information about word boundaries, see ICU Boundary Analysis.
\b, within a [Set]	Match a BACKSPACE, \u0008.
\B	Match if the current position is not a word boundary.
\cX	Match a Ctrl-X character.
\d	Match any character with the Unicode General Category of Nd (Number, Decimal Digit.)
\D	Match any character that is not a decimal digit.
\e	Match an ESCAPE, \u001B.
\E	Terminates a \Q ... \E quoted sequence.
\f	Match a FORM FEED, \u000C.
\G	Match if the current position is at the end of the previous match.
\n	Match a LINE FEED, \u000A.
\N{UNICODE CHARACTER NAME}	Match the named character.
\p{UNICODE PROPERTY NAME}	Match any character with the specified Unicode Property.
\P{UNICODE PROPERTY NAME}	Match any character not having the specified Unicode Property.
\Q	Place quotation marks around all following characters until \E.
\r	Match a CARRIAGE RETURN, \u000D.
\s	Match a white space character. White space is defined as [\t\n\f\r\p{Z}].
\S	Match a non-white space character.
\t	Match a HORIZONTAL TABULATION, \u0009.
\uhhhh	Match the character with the hex value hhhh.
\Uhhhhhhhh	Match the character with the hex value hhhhhhhh. Exactly eight hex digits must be provided, even though the largest Unicode code point is \U0010ffff.

© Copyright IBM Corp. 2007, 2013

Table 55. Regular expression metacharacters (continued)

Character	Description
\w	Match a word character. Word characters are [\p{Ll}\p{Lu}\p{Lt}\p{Lo}\p{Nd}].
\W	Match a non-word character.
\x{hhhh}	Match the character with hex value hhhh. From one to 6 hex digits can be supplied.
\xhh	Match the character with 2 digit hex value hh.
\X	Match a Grapheme Cluster.
\Z\	Match if the current position is at the end of input, but before the final line terminator, if one exists.
\z	Match if the current position is at the end of input.
\n	Back Reference. Match whatever the nth capturing group matched. n must be a number > 1 and < total number of capture groups in the pattern. **Note:** Octal escapes, such as \012, are not supported in ICU regular expressions.
[pattern]	Match any 1 character from the set. See UnicodeSet for a full description of what can appear in the pattern
.	Match any character.
^	Match at the beginning of a line.
$	Match at the end of a line.
\	Place quotation marks around the following character. Characters that must have surrounding quotation marks to be treated as literals are * ? + [( ) { } ^ $ | \ . /

Table 56. Regular expression operators

Operator	Description
|	Alternation. A|B matches either A or B.
*	Match 0 or more times. Match as many times as possible.
+	Match 1 or more times. Match as many times as possible.
?	Match zero or 1 time. Prefer one.
{n}	Match exactly n times
{n,}	Match at least n times. Match as many times as possible.
{n,m}	Match between n and m times. Match as many times as possible, but not more than m.
*?	Match 0 or more times. Match as few times as possible.
+?	Match 1 or more times. Match as few times as possible.
??	Match zero or 1 time. Prefer zero.
{n}?	Match exactly n times
{n,}?	Match at least n times, but no more than required for an overall pattern match
{n,m}?	Match between n and m times. Match as few times as possible, but not less than n.
*+	Match 0 or more times. Match as many times as possible when first encountered, do not retry with fewer even if overall match fails (Possessive Match)

Table 56. Regular expression operators (continued)

Operator	Description
++	Match 1 or more times. Possessive match.
?+	Match zero or 1 time. Possessive match.
{n}+	Match exactly n times
{n,}+	Match at least n times. Possessive Match.
{n,m}+	Match between n and m times. Possessive Match.
(...)	Capturing parentheses. Range of input that matched the parenthesized subexpression is available after the match.
(?: ...)	Non-capturing parentheses. Groups the included pattern, but does not provide capturing of matching text. More efficient than capturing parentheses.
(?> ...)	Atomic-match parentheses. First match of the parenthesized subexpression is the only one tried. If it does not lead to an overall pattern match, back up the search for a match to a position before the "(?>".
(?# ...)	Free-format comment (?# comment).
(?= ...)	Look-ahead assertion. True if the parenthesized pattern matches at the current input position, but does not advance the input position.
(?! ...)	Negative look-ahead assertion. True if the parenthesized pattern does not match at the current input position. Does not advance the input position.
(?<= ...)	Look-behind assertion. True if the parenthesized pattern matches text that precedes the current input position. The last character of the match is the input character just before the current position. Does not alter the input position. The length of possible strings that is matched by the look-behind pattern must not be unbounded (no * or + operators.)
(?<!...)	Negative Look-behind assertion. True if the parenthesized pattern does not match text that precedes preceding the current input position. The last character of the match is the input character just before the current position. Does not alter the input position. The length of possible strings that is matched by the look-behind pattern must not be unbounded (no * or + operators.)
(?ismx-ismx: ...)	Flag settings. Evaluate the parenthesized expression with the specified flags enabled or disabled.
(?ismx-ismx)	Flag settings. Change the flag settings. Changes apply to the portion of the pattern that follows the setting. For example, (?i) changes to a not case-sensitive match.

Replacement text

The replacement text for find-and-replace operations can contain references to capture-group text from the find. References are of the form $n, where n is the number of the capture group.

Table 57. Replacement text characters

Character	Description
`$n`	The text of the positional capture group n is substituted for `$n`. n must be `>= 0`, and not greater than the number of capture groups. A $ not followed by a digit has no special meaning, and is displayed in the substitution text as itself, a $.
`\`	Treat this character as a literal, suppressing any special meaning. Backslash escaping in substitution text is required only for `'$'` and `'\'`, but can be used on any other character without adverse effects.
`$@n`	The text of capture group n is substituted for the regular expression that matched capture group n. n must be `>= 0`, and not greater than the number of capture groups. A `$@` not followed by a digit has no special meaning, and is displayed in the substitution text as itself, a `$@`.
`$#n`	The text of the matched capture group n is substituted for `$#n`. n must be `>= 0`, and not greater than the number of matched capture groups. A `$#` not followed by a digit has no special meaning, and is displayed in the substitution text as itself, a `$#`.

Flag options

The following flags control various aspects of regular expression matching. The flag values can be specified at the time that an expression is compiled into a RegexPattern object. Or, they can be specified within the pattern itself using the `(?ismx-ismx)` pattern options.

Table 58. Flag options

Flag (pattern)	Flag (API constant)	Description
`i`	`UREGEX_CASE_INSENSITIVE`	If set, matching take place in a case-insensitive manner.
`x`	`UREGEX_COMMENTS`	If set, white space and `#comments` can be used within patterns.
`s`	`UREGEX_DOTALL`	If set, a `"."` in a pattern matches a line terminator in the input text. By default, it does not. A carriage-return / line-feed pair in text behaves as a single-line terminator, and matches a single `"."` in a RE pattern
`m`	`UREGEX_MULTILINE`	Control the behavior of `"^"` and `"$"` in a pattern. By default these patterns match only at the start and end, respectively, of the input text. If this flag is set, `"^"` and `"$"` also match at the start and end of each line within the input text.

Appendix H. Creating Non-agent file bundles

You can create file bundles that can be placed in the Tivoli Monitoring depot. These file bundles can then be deployed to target systems in your environment.

About this task

With this function, you can remotely configure products for which there is no remote configuration option. To use this function, you place pre-populated configuration files into the depot and send them out to the wanted systems.

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Under **IBM Tivoli Monitoring Wizards**, select **Non-Agent Remote Deploy Bundle** (Figure 315).

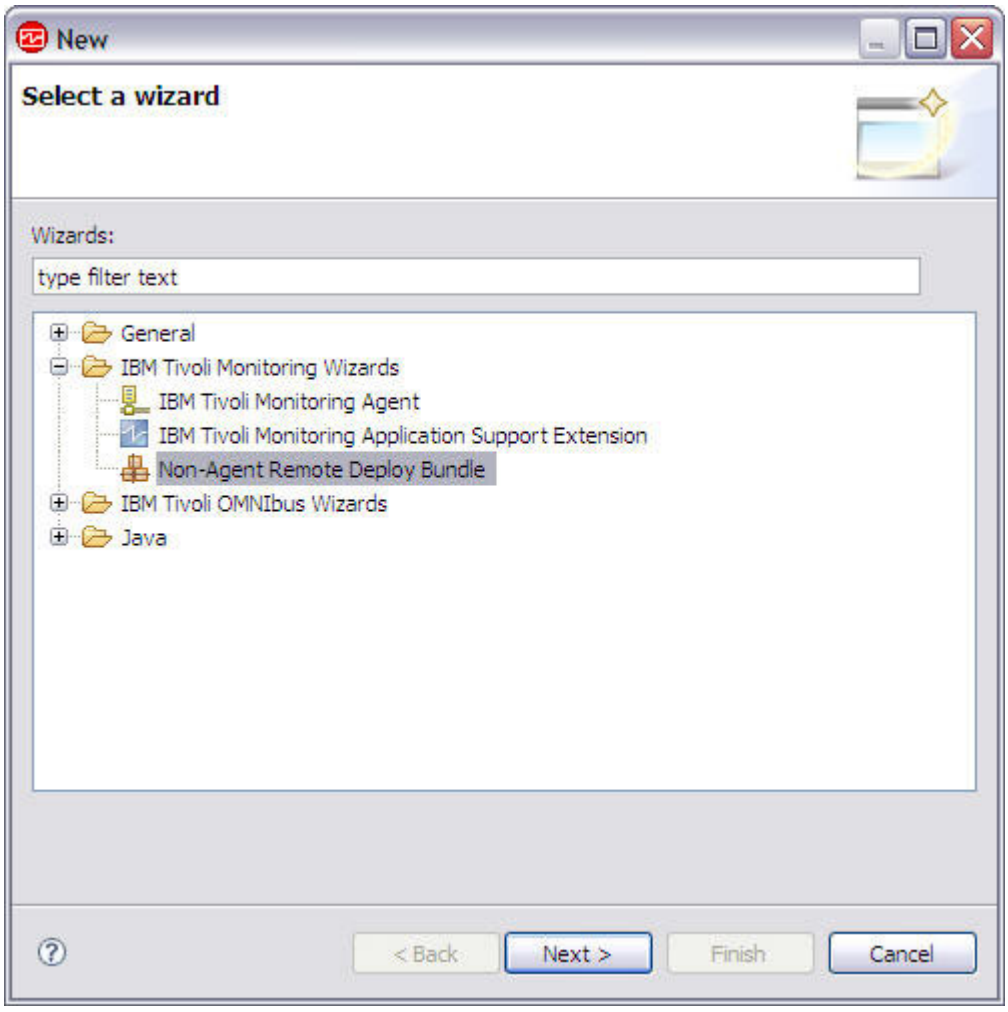


Figure 315. Select a wizard window

3. Click **Next**.
4. In the **Project name** field, enter a name for your project.

© Copyright IBM Corp. 2007, 2013

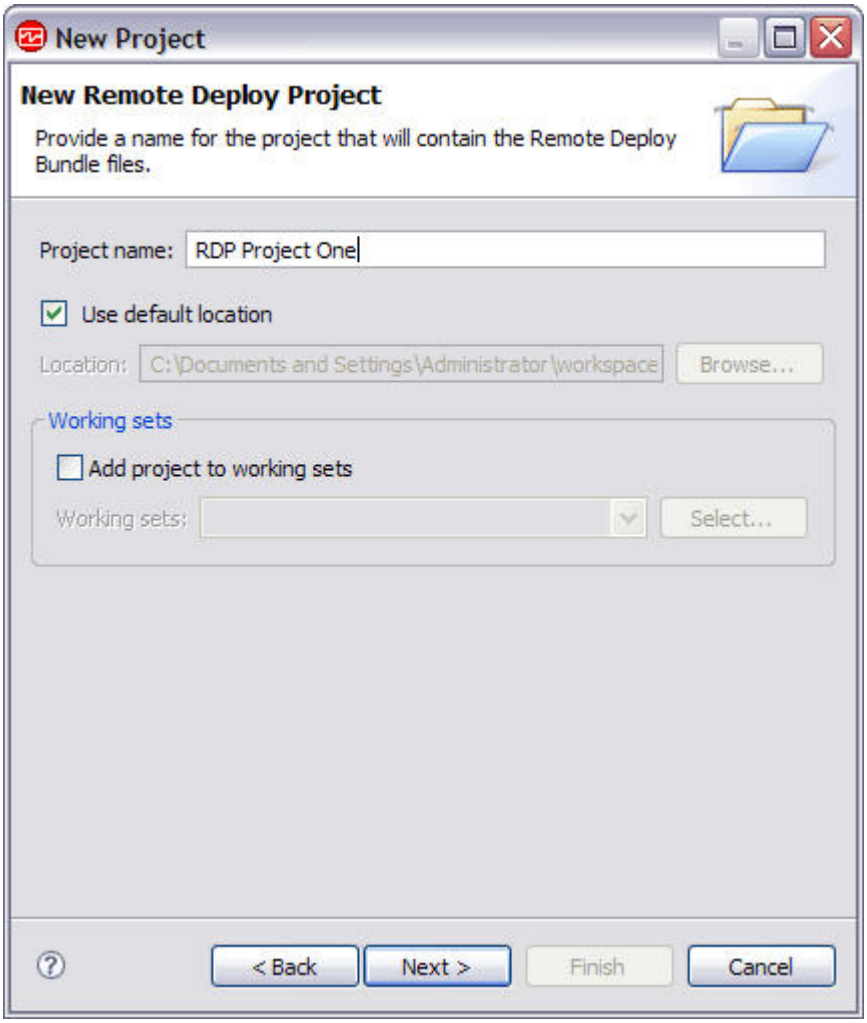


Figure 316. **New Remote Deploy Project** *window*

5. Click **Next**.
6. Complete the information in the **Remote Deploy Bundle Information** window as follows:

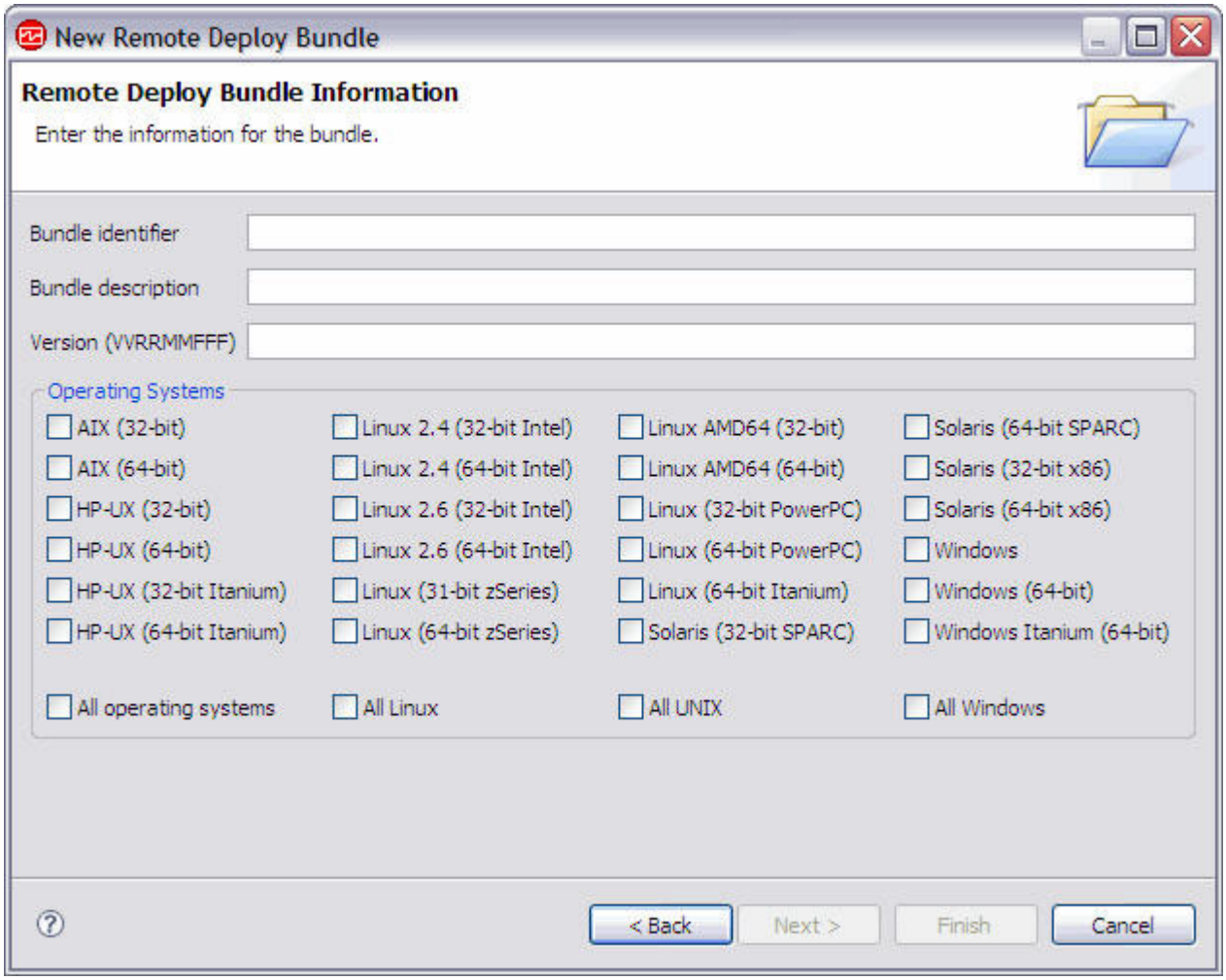


Figure 317. **Remote Deploy Bundle Information** *window*

a. In the **Bundle identifier** field, type an identifier that is a unique alphanumeric string of 3 - 31 characters. This string can contain a hyphen. The string must start with a letter, but it cannot start with a K or a hyphen.
b. In the **Bundle description** field, type a description of the bundle.
c. In the **Version** field, type a version for the bundle in the `VVRRMMFFF` format. Where `vv`= version number; `rr`= release number; `mm`= modification number (fix pack number); and `fff` = interim fix number.

7. In the **Operating Systems** area, select the operating systems to which the bundle can be deployed.
8. Click **Finish** to create a project in the workspace and open the **Remote Deploy Bundle Editor**.

Remote Deploy Bundle Editor

The Remote Deploy Bundle Editor is used to generate commands to help deploy your file bundle.

The Remote Deploy Bundle Editor provides information about the bundle for a project.

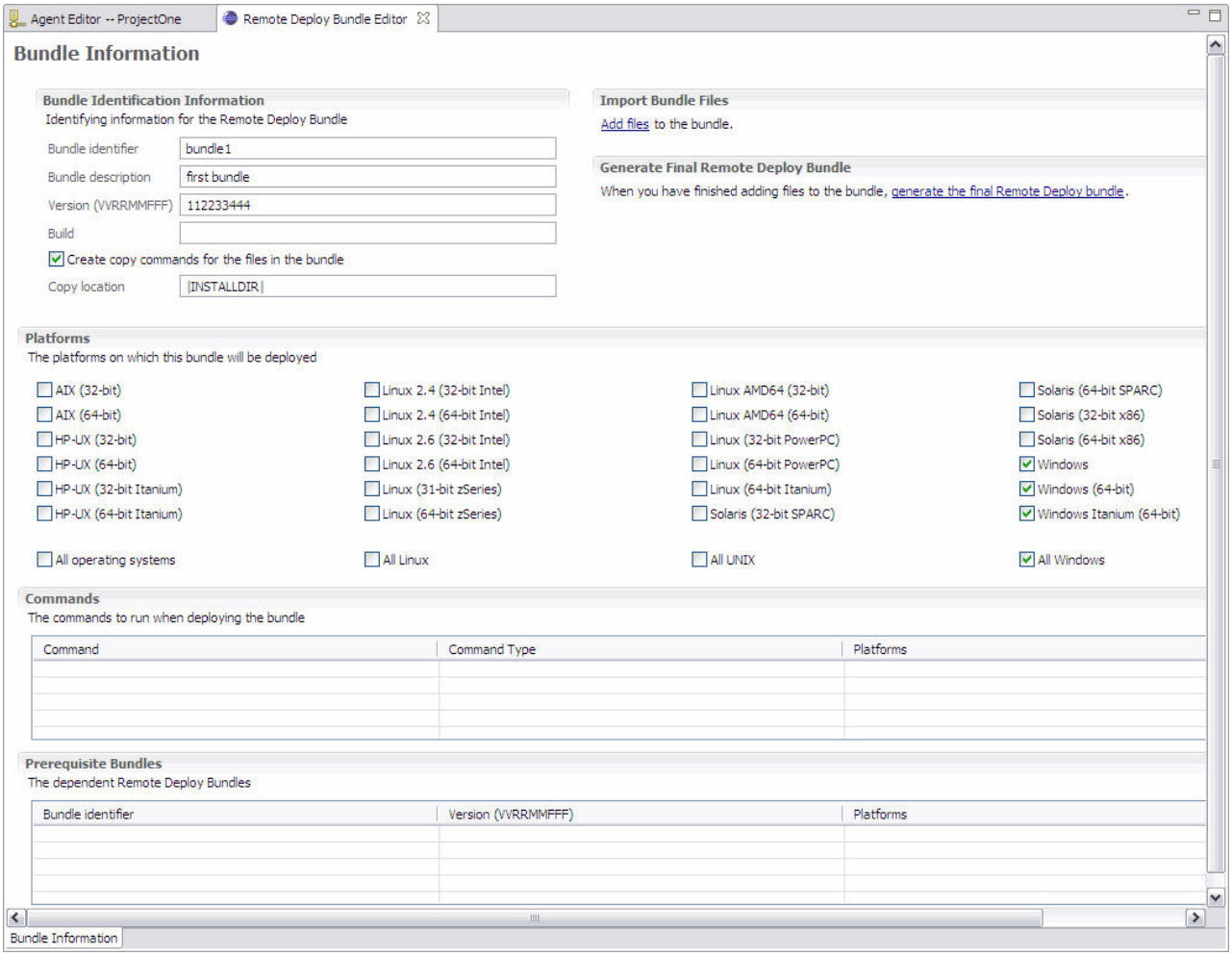


Figure 318. Remote Deploy Bundle Editor

The **Bundle Identification Information** section contains the following information:

Bundle identifier
Unique ID for the bundle

Bundle description
Description for the bundle

Bundle version
Version of the bundle

Build Build identifier for the bundle. Enter a build number here. If no build number is specified, a number is generated from the date and time when the bundle is generated.

Create copy commands for the files in the bundle check box
Click the check box to generate a set of default copy commands that run when the bundle is deployed. The files are copied to the location specified in the **Copy location** text box. The default location is *INSTALLDIR*. Specify this remote deployment variable from the command-line deployment by setting `KDY.INSTALLDIR=...`

The **Operating Systems** section shows the operating systems to which the bundle can be deployed.

The **Commands** section shows the commands to run when the bundle is deployed.

Prerequisite Bundles section shows the bundles that must be present for this bundle to work.

Use the Remote Deploy Bundle Editor to opt for a set of default copy commands that copy the files in your bundle to a set location. If this option is selected, then a copy command is generated for each file in your bundle project. The default copy location is *INSTALLDIR*. A special remote deployment variable that, if not set on the deployment command line, defaults to *CANDLEHOME*. To change the location that is specified by *INSTALLDIR*, specify the **KDY.INSTALLDIR** property when you run the **addSystem** command.

The same directory structure that is specified in your bundle project is replicated in *INSTALLDIR*. For example, if there is a folder named `config` in your bundle project with a file named `myprod.config`, then the generated copy command copies the file to *INSTALLDIR*`/config/myprod.config`when the bundle is deployed.

Adding commands to the bundle

You can specify more commands to run during the deployment.

About this task

You can specify more commands to run during the deployment by using the **Remote Deploy Bundle Editor**.

Procedure

1. To specify more commands to run during the deployment, click **Add** in the **Commands** section of the **Remote Deploy Bundle Editor**.
2. In the Command window, select the type of command **Preinstall**, **Install**, **Post-Install**, or **Uninstall**and then specify the command to run.

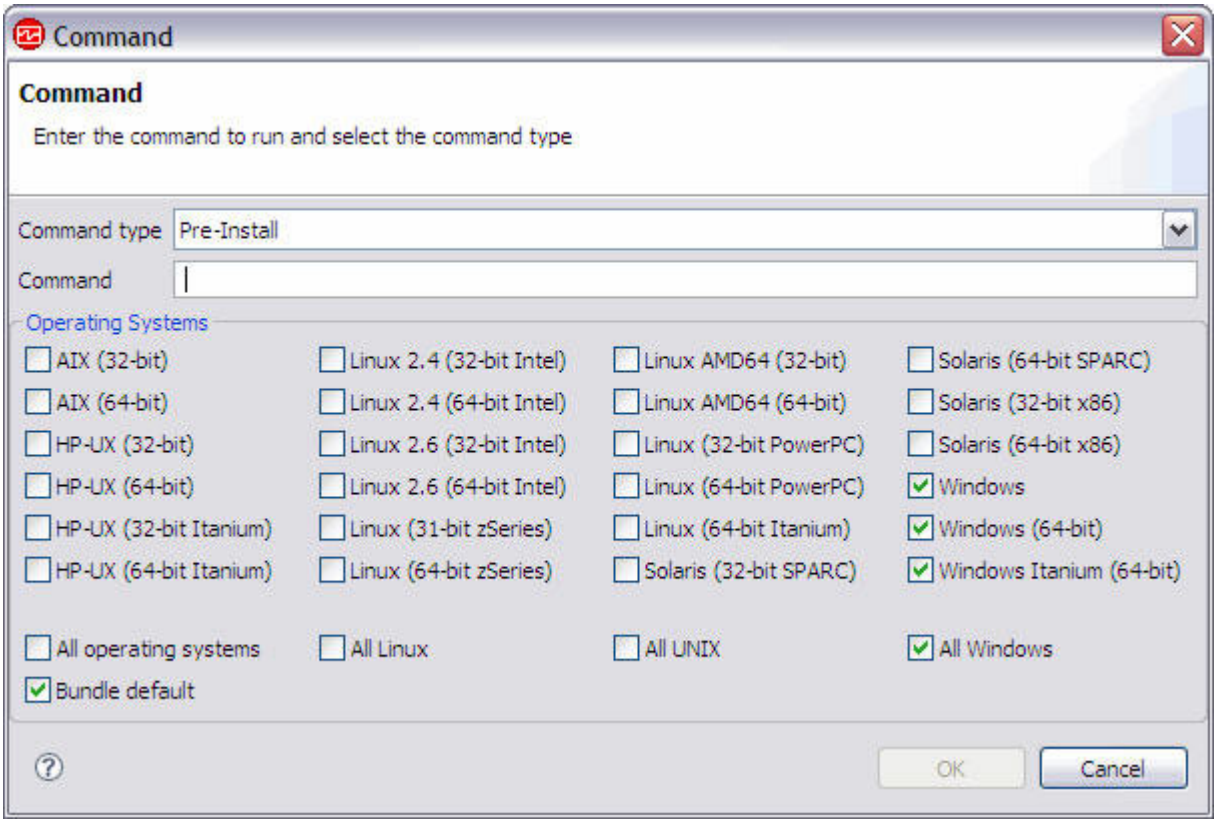


Figure 319. Command window

You must specify the fully qualified path to the command you want to run. For convenience, remote deployment provides a set of predefined variables. To reference the variable for a command, surround the variable with vertical bars, for example, `|DEPLOYDIR|`. For more information about predefined variables for commands, see (Table 59).

Table 59. Predefined Variables for Commands

Variable	Description
DEPLOYDIR	The temporary directory on the endpoint where the bundle is stored during the deployment. For instance, if you want to run `myscript.sh`, a script that is included in your bundle, you specify the following command: `|DEPLOYDIR|/myscript.sh`
INSTALLDIR	Either *CANDLEHOME* or the value of *KDY.INSTALLDIR*if specified on the **addSystem**command.
CANDLEHOME	The Tivoli Monitoring installation directory.

3. Finally, select the **Operating Systems** on which the command is to run.

Adding prerequisites to the bundle

Use the **Remote Deploy Bundle Editor** to specify prerequisites for the bundle.

Procedure

1. To add a prerequisite, click **Add** in the **Prerequisite Bundles** section of the **Remote Deploy Bundle Editor**, Bundle Information page.

2. In the New Prerequisite window (Figure 320), enter the bundle identifier on which this bundle depends and the minimum version required.

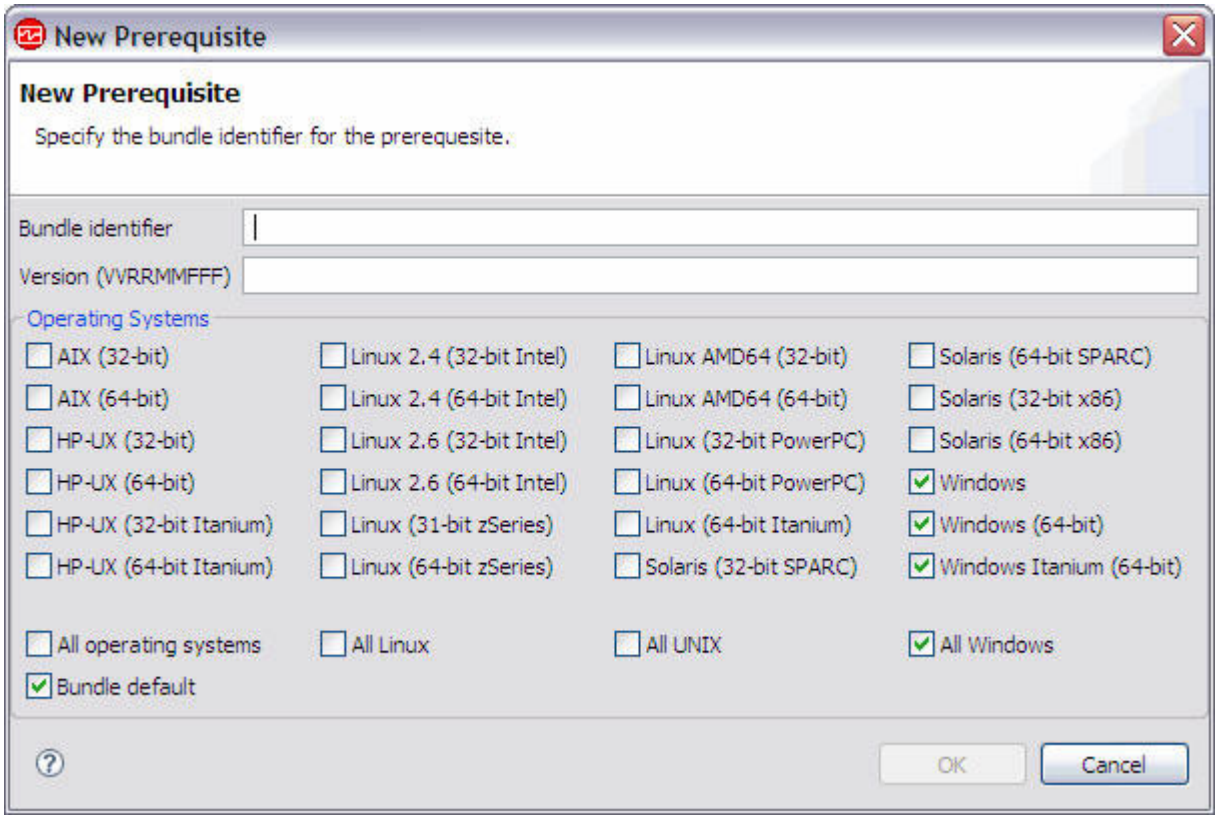


Figure 320. New Prerequisite window

3. Select the operating systems for which this prerequisite is required.
4. Click **OK** to complete and exit.

Adding files to the bundle

Add files to a file bundle by using the **Remote Deploy Bundle Editor**.

Procedure

1. To add files to the remote deployment bundle, do one of the following procedures:
   - In the Bundle Editor, click **Add files to the bundle**.
   - Right-click the project in the Navigator tree, then click **IBM Tivoli Monitoring Remote Deploy** > **Add Files to Bundle**

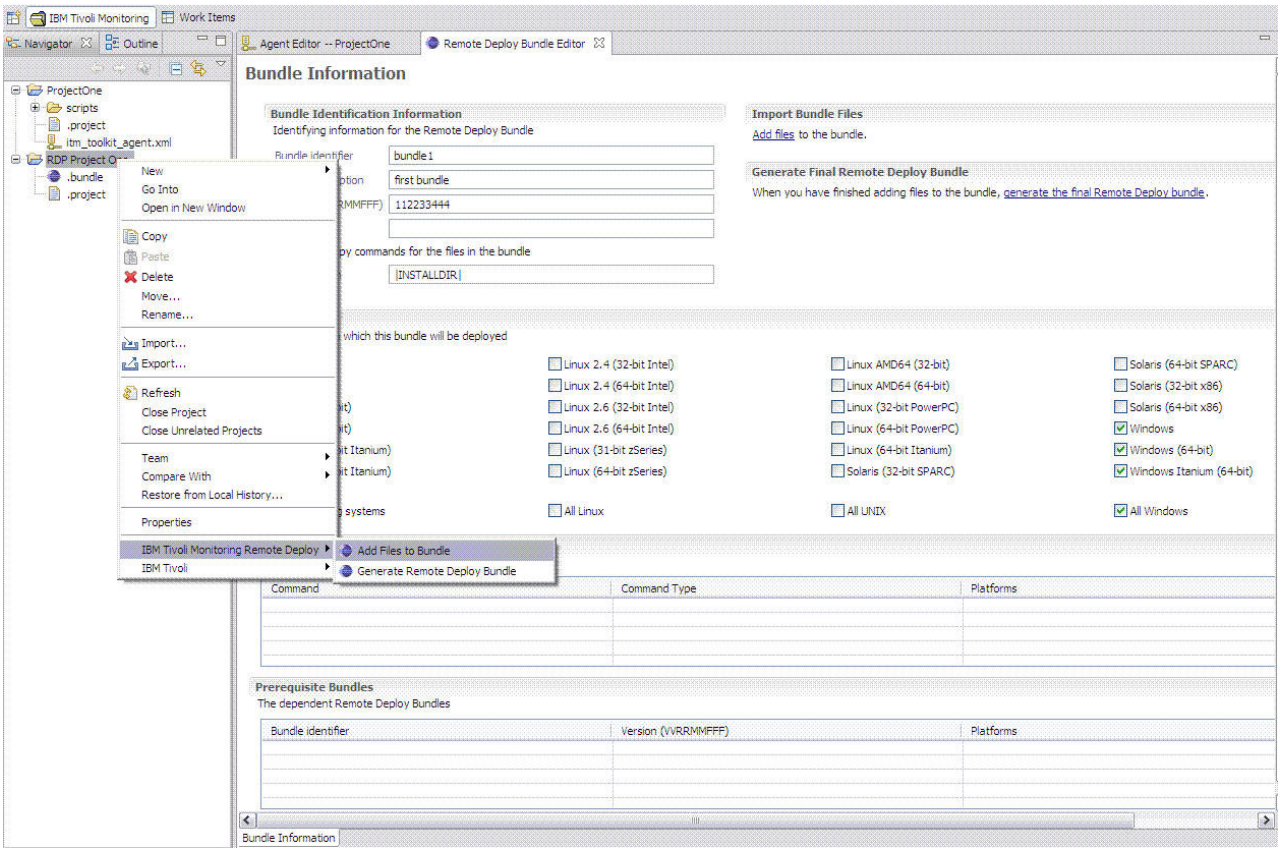


Figure 321. Adding files to the bundle or generating a bundle from the Navigator tree

Both of these actions display the Import Bundle Files window:

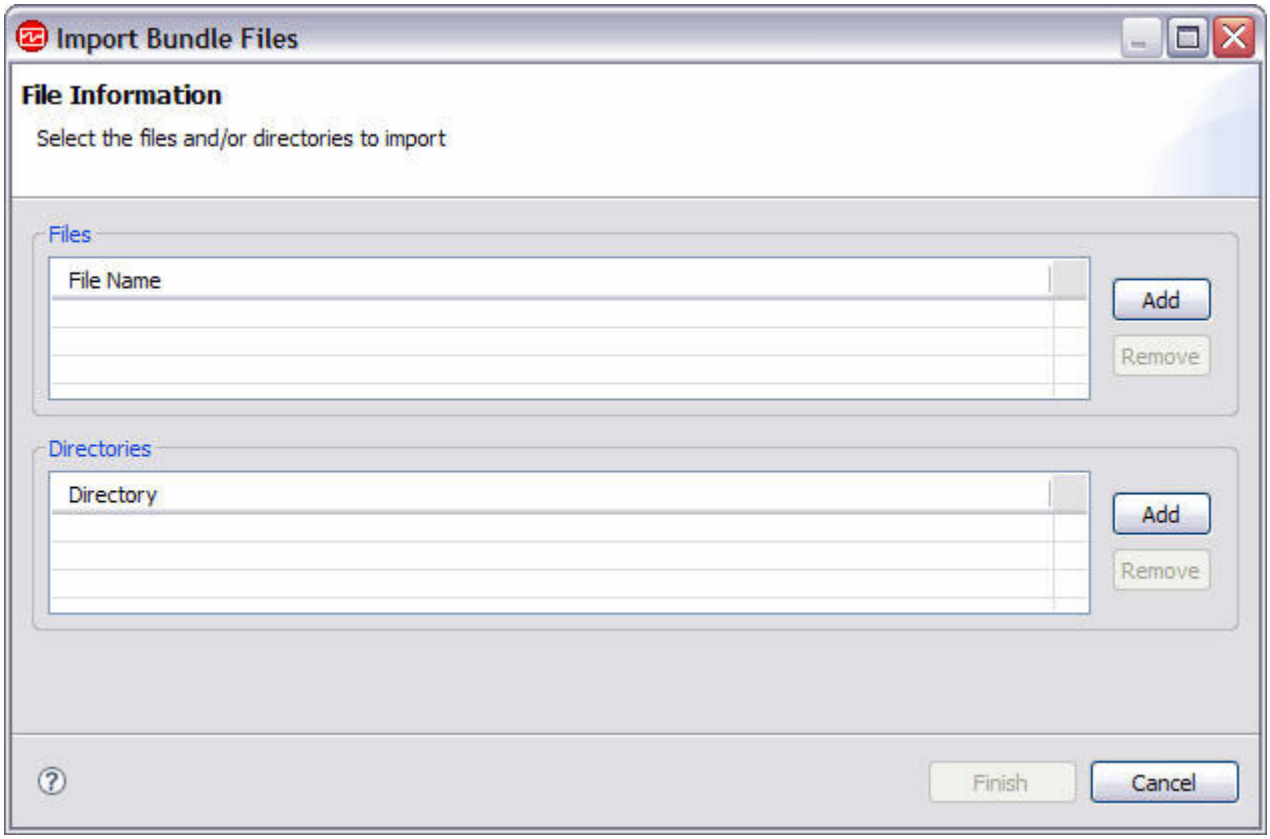


Figure 322. Import Bundle Files window

2. Specify individual files or directories that contain files in **File Information** area.
3. Click **Finish**. The files or directories that are specified are copied into the project directory. The directory structure in the project is maintained when you build the remote deployment bundle. If you want Agent Builder to generate default copy commands, ensure that the files are in the correct directory structure for deployment.

Generating the bundle

Procedure

1. To generate the remote deployment bundle, use one of the following procedures to display the Generate Final Remote Deploy Bundle window
   - In the **Remote Deploy Bundle Editor**, click **generate the final Remote deploy bundle**.
   - Right-click the project in the Navigator tree, then click **IBM Tivoli Monitoring Remote Deploy** > **Generate Remote Deploy Bundle**

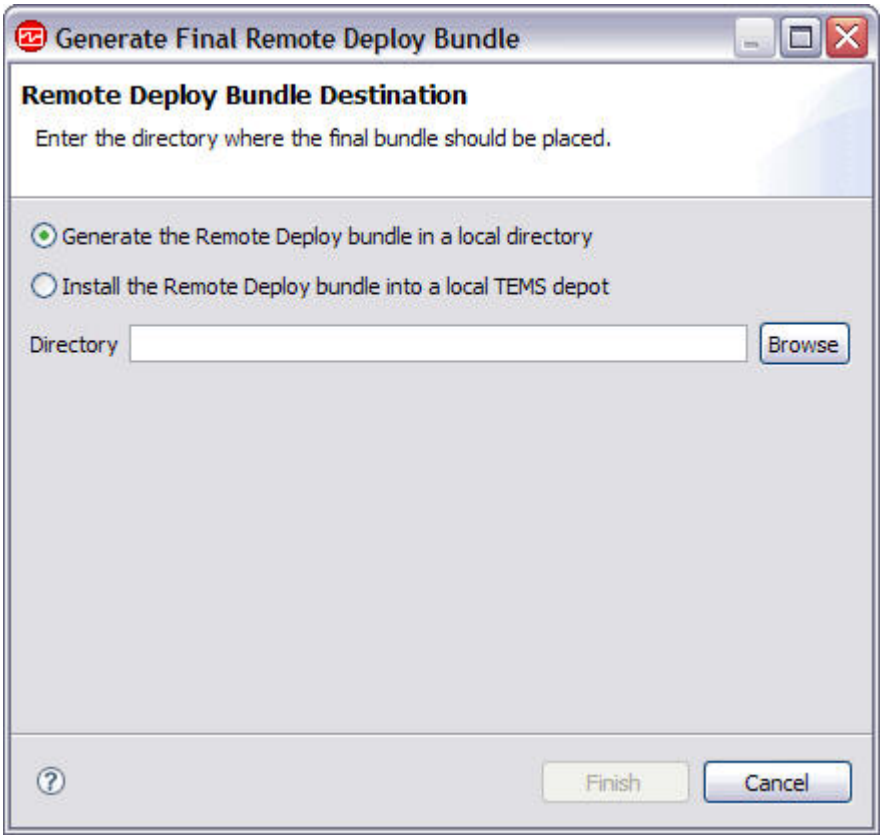


Figure 323. Generate Final Remote Deploy Bundle window

2. You can now generate the bundle in two ways:
   - If there is a Tivoli Enterprise Monitoring Server on the system where you are running the Agent Builder, click **Install the Remote Deploy bundle into a local TEMS depot**.

     The Agent Builder attempts to determine the Tivoli Monitoring installation location and enter it into the **Directory** field. If *CANDLE_HOME* is not set, the default location of `C:\IBM\ITM` or `/opt/IBM/ITM` is used. Ensure that the installation location is correct before you continue.

     You must provide Tivoli Enterprise Monitoring Server login information to install the bundle.
   - To generate the bundle to a directory on your system, click **Generate the Remote Deploy bundle in a local directory**

     After the process is complete, you must transfer this directory to a Tivoli Enterprise Monitoring Server system and use the `tacmd addbundles` command to add the bundle to the depot.

What to do next

When you deploy the bundle, you must use the `tacmd addSystem` command. For example:

```
tacmd addsystem -t MONITORINGCOLLECTION -n Primary:ITMAGT:NT
```

Where `-t` (type) is the Product Code as returned by the `tacmd viewDepot` command:

```
>tacmd viewDepot
Product Code : MONITORINGCOLLECTION
Version : 010000003
Description : MonitoringCollectionScripts
Host Type : WINNT
Host Version : WINNT
Prerequisites:
```

Note: You cannot deploy remotely from the Tivoli Enterprise Portal Desktop or Browser. Deploy remotely from the Tivoli Enterprise Portal Desktop or Browser results in the KFWITM219E message.

See the Tivoli Monitoring documentation for more details.

Creating deployable bundles for Tivoli Netcool/OMNIbus probes

You can use the Agent Builder to create package and configuration bundles that can be used to deploy Tivoli Netcool/OMNIbus probes to remote computers.

About this task

To support the remote deployment of probes, you can also create Tivoli Netcool/OMNIbus bundles that can be deployed to the remote computers before you deploy the probes.

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Under **IBM Tivoli OMNIbus Wizards**, select **Package Bundle**.
3. Click **Next**.

What to do next

Next, use the **OMNIbus Install Bundle** wizard to create the bundles. For information about using this wizard, see the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide* in the Tivoli Netcool/OMNIbus information center at IBM Tivoli Netcool/OMNIbus documentation.

Appendix I. Dynamic file name support

Use dynamic file name support to specify a file name pattern instead of an actual file name.

Some application programs create an output file name that is subject to change. The name changes based on specific criteria such as the current day, month, year, or a file name that includes an incrementing sequence number. In these cases, you can specify the file name pattern instead of the actual file name. There are two pattern formats that are recognized when you specify the file name pattern:

- Regular Expressions (preferred).
- IBM Tivoli Universal Agent dynamic file name syntax (deprecated).

Regular expression file name patterns

To specify file name patterns, you can use regular expressions according to the International Components for Unicode (ICU) syntax that is documented in (Appendix G, "ICU regular expressions," on page 627). To use this capability, you must select the **File names match regular expression** check box on the Advanced Log File Attribute Group Information page. When you specify regular expression patterns, you must also select an option from the **When Multiple Files Match** list on the Advanced Log File Attribute Group Informationpage to specify the guidelines for selecting the most current matching file.

Note: Regular expressions is the preferred method to specify file name patterns.

For more information about how to configure advanced log file attribute group properties, see (Chapter 16, "Monitoring a log file," on page 177), Step (6 on page 179). For example, if you specified a file name pattern:

```
d:\program files\logs\tivoli.*
```

This pattern searches for file names that start with `tivoli` in the `d:\program files\logs` directory. Regular expressions can be specified only for the file name portion, and not the path name.

Dynamic file name syntax

With the dynamic file name syntax, only one file at a time can be monitored. The File Data Provider inspects all files in the designated path location, seeking files that match the defined pattern. The File Data Provider always monitors the most current matching file that is based on whichever matching file name has the highest number or date-time value. The appropriate file to monitor is determined by file name, instead of by file creation or other criteria.

Patterns can be specified for file names with any number of parts. For example, `Log{###}` matches on one-part file names such as `Log010` or `Log456`. In multi-part file names, pattern characters can be specified in any part of the file name or in multiple parts. For example, `aaa.bbb{???}.ccc` is a valid pattern, and `aaa.bbb{???}.ccc{###}` is also valid.

Note: Regular expressions rather than dynamic file name syntax is the preferred method to specify file name patterns, for more about regular expressions, see "Regular expression file name patterns"

The following examples illustrate file name pattern specification:

`{########}.abc`
: Matches numeric file names of length 8 and the file extension *`.abc`*, such as `10252006.abc` or `10262006.abc`. File `10262006.abc` is monitored because 10262006 is greater than 10252006.

© Copyright IBM Corp. 2007, 2013

{########}.*
: Matches numeric file names of length 8 and ignores the file extension. Examples include `20061025.log`, `20061101.log`, and `10252006.abc`. File `20061101.log` is monitored because 20061101 is the largest number.

{######??}.abc
: Matches numeric file names of length 8 and file extension `.abc`, and ignores the last two positions in the name portion. Examples include `02110199.abc`, `02110200.abc`, and `021101AZ.abc`. File `02110200.abc` is monitored because 021102 is the largest number.

Console.{######}
: Matches file names that contain *Console* in the name portion and a six-digit number in the extension portion. Examples include `Console.000133`, `Console.000201`, and `Console.000134`. File `Console.000201`is monitored.

IN{######}.log
: Matches file names that start with `IN` followed by six numerals and the file extension `.log`. Examples include `IN021001.log`, `IN021002.log`, and `IN021004.log`. File `IN021004.log` is monitored.

PS{###}FTP.txt
: Matches file names that start with `PS` followed by three numerals, followed by `FTP`, and the extension `.txt`. Examples include `PS001FTP.txt`, `PS005FTP.txt`, and `PS010FTP.txt`. File `PS010FTP.txt`is monitored.

Follow these guidelines to establish file name patterns:

- Use braces {} to enclose pattern characters in a file name. The presence of pattern characters inside braces indicates that a file name pattern is being used.
- Use an asterisk (*) as a wildcard to ignore file extensions or any trailing characters in the file name. For example, `Myapp{###}.log*` specifies that any file name that starts with `Myapp`, followed by three digits, and followed by `.log`, is a match, regardless of what comes after.

  The asterisk must be specified after the curly braces `({ })` and cannot be used at the beginning of a file name. When you use the asterisk in a file name extension, the asterisk must be used by itself.

  Examples of correct wildcard `(*)`usage:
  err{??}.*
  error{$}.*

  Examples of incorrect wildcard `(*)` usage:
  error.20*
  : No curly braces precede the asterisk `(*)`.

  error*.{###}
  : The asterisk is not used at the end of the file name.

  error.*
  : No curly braces precede the asterisk `(*)`.
- If a specific file extension is defined, then only files with the same extension are considered.
- Use a number sign to indicate each numeric element of a file name.
- Use a question mark to exclude each element of the naming convention that does not serve as search criteria in determining the appropriate file name.
- Use a dollar sign `($)` to represent either any character or no character. For example, if you want to match on two files named `Log` and `LogA`, specify `Log{$}`. The dollar sign has several usage restrictions. When you use one or more dollar signs to prefix a file name as in `{$$$$$$}_abc.log`, the number of dollar signs must exactly match the number of characters in that position in the file name. Also, you cannot specify dollar signs in multiple locations in a file name pattern, for example, `{$$$}b{$$$}.log` does not match `abc.log`. Given these dollar sign restrictions, use regular expression file name patterns if there are an indeterminate number of characters in the file names.

- The total number of number signs and question marks that are enclosed in braces is significant. It must match the portion of file name exactly. For example, the pattern AA{####} instructs the File Data Provider to look for files such as AA0001. File names, such as AA001 or AA00001, are not considered.
- The exact file name pattern, the constant, and the numeric parts, must match the file name exactly. For example, the pattern AA{###} instructs the File Data Provider to check file AA101. File names, such as XAA101, AA222X and AA55555, are not considered.
- Use the reserved pattern string {TIVOLILOGTIME}to substitute for the hex timestamp and file sequence number in a Tivoli Monitoring agent or server log file. This pattern string is useful when you do self-monitoring of Tivoli Monitoring components. For example, if you want to monitor the latest monitoring server log in the /opt/IBM/ITM/logsdirectory, can specify a file name pattern:

  ```
  /opt/IBM/ITM/logs/Host1_ms_{TIVOLILOGTIME}.log
  ```

  If Host1_ms_452053c0-01.log, Host1_ms_451f11f4-01.log, Host1_ms_45205946-01.log, and Host1_ms_451f11f4-02.log are present in the /logsdirectory, the Host1_ms_45205946-01.log file is selected for monitoring.

To precisely specify a file name that consists of date components (year, month, and day), use the capital letters Y, M, and D. These letters must be specified within braces; otherwise they are treated as literal characters in the file name.

See the following examples:

{YYYYMMDD}.log
: Specifies file names such as 20060930.log or 20061015.log.

{MMDDYY}.log
: Specifies file names such as 101106.log or 110106.log.

{DDMMYYYY}.log
: Specifies file names such as 01092006.log or 15082006.log.

{DDMMMYY}.log
: Specifies file names such as 24Jan07 or 13Sep06.

{MM-DD-YY}.log
: Specifies file names such as 11-02-06 or 04-29-07. The (-)separator character is ignored in the date field and does not require a question mark pattern character to skip over it.

MY{YYDDD}.log
: Specifies file names such as MY06202.log, MY06010.log, or MY04350.log.

Complex cases exist, where a date field is embedded within a longer file name, and the date patterns in the previous examples are not sufficient. For complex cases, create patterns that mix number signs and question marks and still perform numeric comparisons that select the most current file for monitoring. For example, the pattern ABC{?####?##?##?##?##?##?}XYZ.TXT can be used for file names such as ABC 2006-04-20 11_22_33 XYZ.TXT. In this example, you are interested in only the #- marked digits and question marks serve as placeholders that ignore other characters in the file name.

The File Data Provider periodically checks for new files that match the defined file pattern in the target path location. When a newer file that matches the pattern is detected, the File Data Provider automatically switches application monitoring to the new file. The File Data Provider searches for the best matching file when:

- The File Data Provider first starts.
- The currently monitored file no longer exists because of possible renaming or deleting.
- The existing file contents, changed because of possible rewriting.
- The check interval expired. The default interval is 10 minutes. You can change the interval to a longer or shorter interval value by specifying the environment variable

KUMP_DP_FILE_SWITCH_CHECK_INTERVAL=*number-of-seconds*

Appendix J. SNMP trap configuration

Description of the configuration file that is used by the SNMP Data Provider to render trap information in a more easily readable form. The file is also used to assign categories, severities, status, and source IDs to traps.

Also contains instructions for modifying the default file or substituting your own configuration file.

SNMP trap configuration file, `trapcnfg`

At startup, the SNMP Data Provider reads a configuration file named `trapcnfg`. One purpose of this file is to translate SNMP trap information into a more readable form. Another is to assign categories, severities, status, and source IDs to specific traps, since these categories are not defined by SNMP.

You can modify the `trapcnfg` file to suit your site-specific needs by adding new trap or enterprise definitions or changing the existing ones. You can also use your own configuration file.

Use the HP OpenView `trapd.conf` file

The `trapcnfg` file is similar in format, but not identical, to the HP OpenView Network Node Manager trap configuration file `trapd.conf`. You can copy the OpenView file and reuse many of the definition statements if necessary.

Types of records

`trapcnfg` contains three types of records or record blocks:

comments
: Comment records begin with a number sign (#).

enterprise definitions
: Enterprise definitions consist of two blank-delimited tokens, where the first token is a name and the second is an object identifier (OID) surrounded by curly brackets ({ }).

trap definitions
: Trap definitions consist of eight blank-delimited tokens. Trap definitions are block records because each definition might consist of multiple records.

The first type is self-explanatory. (Figure 324 on page 648) shows examples of the second and third types.

The first example in Figure 324 on page 648 shows an enterprise definition record which defines enterprise OID 1.3.6.1.4.1.311.1.1.3.1.1 as being Microsoft Windows NT.

The second example shows a trap definition record that defines `trapName` `MSNTCOLD` as being associated with enterprise OID `1.3.6.1.4.1.311.1.1.3.1.1`, generic trap number 0, and specific trap number 0. Notice that the severity is in decimal form whereas the category is in textual form. Severities are translated into their textual form before they are displayed. The next record in the type 3 record block is the short description, which the Agent Builder does not use. The Agent Builder uses the long description that is enclosed within the delimiters `SDESC` and `EDESC`.

© Copyright IBM Corp. 2007, 2013

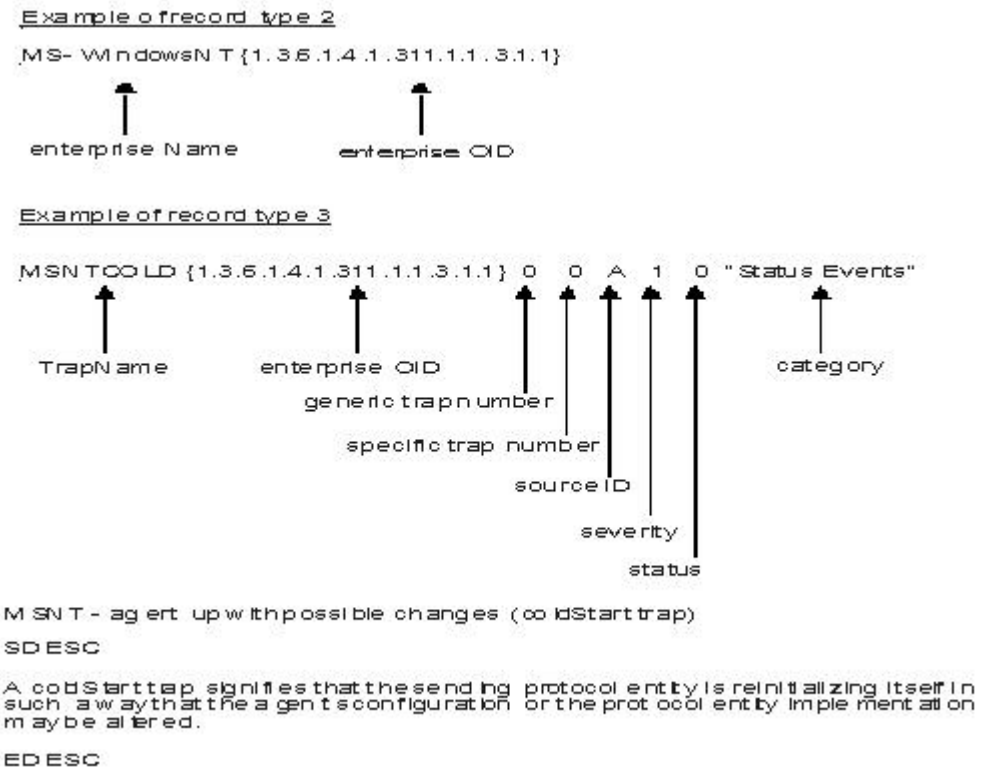


Figure 324. Examples of configuration record types 2 and 3

Defaults for the `trapcnfg` file

Tables that list the defaults that are supported by the SNMP Data Provider.

Supported categories

(Table 60) shows the categories that are supported by the Agent Builder.

Table 60. Categories supported by the SNMP Data Provider

Category	Textual representation
0	Threshold Events
1	Network Topology Events
2	Error Events
3	Status Events
4	Node Configuration Events
5	Application Alert Events
6	All Category Events
7	Log Only Events
8	Map Events
9	Ignore Events

(Table 61 on page 649) lists the severities that are supported by the Agent Builder.

Table 61. Severities supported by the SNMP Data Provider

Severity	Textual representation
0	Clear
1	Indeterminate
2	Warning
3	Minor Error
4	Critical
5	Major Error

Supported statuses

(Table 62) shows the statuses that are defined in the Agent Builder configuration file.

Table 62. Statuses supported by the SNMP Data Provider

Status	Textual representation
0	Unchanged
1	Unknown
2	Up
3	Marginal
4	Down
5	Unmanaged
6	Acknowledge
7	User1
8	User2

Supported source IDs

(Table 63) lists the source IDs supported by `trapcnfg`.

Table 63. Source IDs supported by the SNMP Data Provider

Source ID	Description
a	Application
A	Agent
C	Xnmcollect
d	Demo
D	Data Collector
E	Nvevents
I	Ipmap
L	LoadMIB
m	Shpmon
M	IP topology
n	netmon related
N	netmon-generated traps
O	OSI SA

Table 63. Source IDs supported by the SNMP Data Provider (continued)

Source ID	Description
P	Non-IP traps
r	Tralertd
s	Spappld
S	Security Agent
t	Xnmtrap
T	Trapd
V	Vendor related
?	Unknown

Appendix K. Take Action commands reference

An overview of Take Action commands, references about Take Action commands, and descriptions of special Take Action commands.

About Take Action commands

Take Action commands can be included in an Agent Builder monitoring agent. Take Action commands can be run from the portal client or included in a situation or a policy. When included in a situation, the command runs when the situation becomes true. A Take Action command in a situation is also known as reflex automation. When you enable a Take Action command in a situation, you automate a response to system conditions. For example, you can use a Take Action command to send a command to restart a process on the managed system. You can also use a Take Action command to send a text message to a cell phone.

Advanced automation uses policies to run actions, schedule work, and automate manual tasks. A policy comprises a series of automated steps that are called activities that are connected to create a workflow. After an activity is completed, the Tivoli Enterprise Portal receives return code feedback, and advanced automation logic responds with subsequent activities prescribed by the feedback.

A basic Take Action command displays the return code of the operation in a message box or a log file that is displayed after action completion. After you close this window, no further information is available for this action.

More information about Take Action commands

For more information about working with Take Action commands, see the *Tivoli Enterprise Portal User's Guide*.

For a list and description of the Take Action commands for this monitoring agent, see ("Special Take Action commands"). See also the information in that section for each individual command.

Special Take Action commands

An Agent Builder monitoring agent can recognize and do special processing for a set of Take Action commands:

- SSHEXEC

For more information about creating these commands and including them in an Agent Builder monitoring agent project, see (Chapter 32, "Creating workspaces, Take Action commands, and situations," on page 403).

SSHEXEC action

Before you begin

For more information about Take Action commands, see (Appendix K, "Take Action commands reference").

© Copyright IBM Corp. 2007, 2013

About this task

The SSHEXEC action is recognized for a monitored application that has at least one SSH Script attribute group. It indicates that the command that follows the SSHEXEC keyword is remotely started on the SSH target system. The command is started with the credentials and privileges of the user that is configured to monitor the SSH target system. The command is run on the remote system that is represented by the Managed System Name.

Procedure

To include the Take Action command in a situation or workflow policy, use the following syntax for the system command:

```
SSHEXEC [Command]
```

For example:

```
SSHEXEC [ls &path]
```

Note: You can customize the command or portions of the command during invocation of the Take Action by using the Take Action arguments option with the *Command*.

Note: If the *Command* includes multiple arguments, then consider including the bracket parenthesis to enable invocation of the Take Action command with the **tacmd** command-line interface.

Appendix L. Documentation library

This appendix contains information about the publications that are related to IBM Tivoli Monitoring and to the commonly shared components of Tivoli Management Services.

These publications are listed in the following categories:

- IBM Tivoli Monitoring library
- Related publications

For information about how to access and use the publications, select **Using the publications** in the **Contents** pane of the IBM Tivoli Monitoring and OMEGAMON XE Information Center at (http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp).

To find a list of new and changed publications, click **What's new** on the Welcome page of the IBM Tivoli Monitoring and OMEGAMON XE Information Center. To find publications from the previous version of a product, click **Previous versions** under the name of the product in the Contents pane.

IBM Tivoli Monitoring library

The following publications provide information about IBM Tivoli Monitoring and about the commonly shared components of Tivoli Management Services:

- *Quick Start Guide*

  Introduces the components of IBM Tivoli Monitoring.
- *Installation and Setup Guide*, SC22-5445

  Provides instructions for installing and configuring IBM Tivoli Monitoring components on Windows, Linux, and UNIX systems.
- *Program Directory for IBM Tivoli Management Services on z/OS*, GI11-4105

  Gives instructions for the SMP/E installation of the Tivoli Management Services components on z/OS.
- *High Availability Guide for Distributed Systems*, SC22-5455

  Gives instructions for several methods of ensuring the availability of the IBM Tivoli Monitoring components.
- *IBM Tivoli zEnterprise Monitoring Agent Installation and Configuration Guide*, SC14-7358

  Provides instructions for installing and configuring Tivoli zEnterprise® monitoring agent components on Windows, Linux, and UNIX systems. Also includes migration and backup information, Enterprise Common Collector troubleshooting, Hardware Management Console configuration, and use of the command-line interface or APIs to customize the collector. This guide complements the *Tivoli zEnterprise Monitoring Agent User's Guide*.
- *Administrator's Guide*, SC22-5446

  Describes the support tasks and functions that are required for the Tivoli Enterprise Portal Server and clients, including Tivoli Enterprise Portal user administration.
- *Command Reference*, SC22-5448

  Provides detailed syntax and parameter information, as well as examples, for the commands you can use in IBM Tivoli Monitoring.
- *Messages*, SC22-5450

  Lists and explains messages that are generated by all IBM Tivoli Monitoring components and by z/OS-based Tivoli Management Services components (such as Tivoli Enterprise Monitoring Server on z/OS and TMS:Engine).
- *Troubleshooting Guide*, GC22-5449

© Copyright IBM Corp. 2007, 2013

Provides information to help you troubleshoot problems with the software.

- Tivoli Enterprise Portal online help

  Provides context-sensitive reference information about all features and customization options of the Tivoli Enterprise Portal. Also gives instructions for using and administering the Tivoli Enterprise Portal.

- *Tivoli Enterprise Portal User's Guide*, SC22-5447

  Complements the Tivoli Enterprise Portal online help. The guide provides hands-on lessons and detailed instructions for all Tivoli Enterprise Portal features.

- *Agent Builder User's Guide*, SC32-1921

  Explains how to use the Agent Builder for creating monitoring agents and their installation packages, and for adding functions to existing agents.

- *Performance Analyzer User's Guide*, SC22-5458

  Explains how to use the Performance Analyzer to understand resource consumption trends, identify problems, resolve problems more quickly, and predict and avoid future problems.

- *IBM Tivoli zEnterprise Monitoring Agent User's Guide*, SC14-7359

  Complements the Tivoli zEnterprise monitoring agent online help. The guide provides reference information about the interface, usage scenarios, agent troubleshooting information, and information about Tivoli Common Reporting reports. This guide complements the *Tivoli zEnterprise Monitoring Agent Installation and Configuration Guide*.

Documentation for the base agents

If you purchased IBM Tivoli Monitoring as a product, you received a set of base monitoring agents as part of the product. If you purchased a monitoring agent product that includes the commonly shared components of Tivoli Management Services, you did not receive the base agents. The OMEGAMON XE product is an example of such a product.

The following publications provide information about using the base agents.

- Operating system agents:
  - *Windows OS Agent User's Guide*, SC22-5451
  - *UNIX OS Agent User's Guide*, SC22-5452
  - *Linux OS Agent User's Guide*, SC22-5453
  - *IBM i Agent User's Guide*, SC22-5454
- Agentless operating system monitors:
  - *Agentless Monitoring for Windows Operating Systems User's Guide*, SC23-9765
  - *Agentless Monitoring for AIX Operating Systems User's Guide*, SC23-9761
  - *Agentless Monitoring for HP-UX Operating Systems User's Guide*, SC23-9763
  - *Agentless Monitoring for Solaris Operating Systems User's Guide*, SC23-9764
  - *Agentless Monitoring for Linux Operating Systems User's Guide*, SC23-9762
- Warehouse agents:
  - *Warehouse Summarization and Pruning Agent User's Guide*, SC22-5456
  - *Warehouse Proxy Agent User's Guide*, SC22-5457
- System P agents:
  - *AIX Premium Agent User's Guide*, SA23-2237
  - *CEC Base Agent User's Guide*, SC23-5239
  - *HMC Base Agent User's Guide*, SA23-2239
  - *VIOS Premium Agent User's Guide*, SA23-2238
- Other base agents:
  - *Tivoli Log File Agent User's Guide*, SC14-7484

– *Systems Director base Agent User's Guide*, SC27-2872
– *Monitoring Agent for IBM Tivoli Monitoring 5.x Endpoint User's Guide*, SC32-9490

Related publications

For information about related products and publications, select **OMEGAMON XE shared publications** or other entries in the Contents pane of the IBM Tivoli Monitoring and OMEGAMON XE Information Center at (http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp).

Other sources of documentation

You can also obtain technical documentation about IBM Tivoli Monitoring and related products from the following sources:

- Service Management Connect (SMC)

  For introductory information about SMC, see http://www.ibm.com/developerworks/servicemanagement/.

  For information about Tivoli products, see the Application Performance Management community on SMC at http://www.ibm.com/developerworks/servicemanagement/apm/index.html.

  Connect, learn, and share with Service Management professionals. Get access to developers and product support technical experts who provide their perspectives and expertise. Using SMC, you can:
  – Become involved with transparent development, an ongoing, open engagement between external users and developers of Tivoli products. You can access early designs, sprint demos, product roadmaps, and pre-release code.
  – Connect one-on-one with the experts to collaborate and network about Tivoli and Integrated Service Management.
  – Benefit from the expertise and experience of others by using blogs.
  – Collaborate with the broader user community by using wikis and forums.
- Tivoli wikis on the IBM developerWorks® website

  Tivoli Wiki Central at http://www.ibm.com/developerworks/wikis/display/tivoli/Home is the home for interactive wikis that offer best practices and scenarios for using Tivoli products. The wikis contain white papers contributed by IBM employees, and content that is created by customers and business partners.

  Two of these wikis are of particular relevance to IBM Tivoli Monitoring:
  – Tivoli Distributed Monitoring and Application Management Wiki at http://www.ibm.com/developerworks/wikis/display/tivolimonitoring/Home provides information about IBM Tivoli Monitoring and related distributed products, including IBM Tivoli Composite Application Management products.
  – Tivoli System z® Monitoring and Application Management Wiki at http://www.ibm.com/developerworks/wikis/display/tivoliomegamon/Home. This Wiki provides information about the OMEGAMON XE products, NetView® for z/OS, Tivoli Monitoring Agent for z/TPF, and other System z monitoring and application management products.
- IBM Integrated Service Management Library

  http://www.ibm.com/software/brandcatalog/ismlibrary/

  IBM Integrated Service Management Library is an online catalog that contains integration documentation and other downloadable product extensions.
- Redbooks®

  http://www.redbooks.ibm.com/

  IBM Redbooks and Redpapers include information about products from platform and solution perspectives.
- Technotes

Technotes provide the latest information about known product limitations and workarounds. You can find Technotes through the IBM Software Support website at http://www.ibm.com/software/support/.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corp. 2007, 2013

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM‘s application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not display.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/us/en/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

A

About this guide 2
abs 82
Address attribute 574
Administrator authorization 6
agent
  changing version 52
  name 20
  naming 20
  update agent version 52
Agent Builder
  DVD 6
  guidelines for installation 5
  starting 11
  uninstalling 8
agent editor
  Cognos information 42
  default operating systems 37
  environment variables 39
  runtime configuration information 48
  self-describing agent 38
  watchdog information 41
Agent Editor
  Agent Information page 35
agent testing 377, 382
Agent Watchdog 41
agent wizard, Starting the new 13
Agent XML Editor page 49
agent, creating a basic 13
agent, organizing 29
agents
  after installation 392
  configuring and starting 391
  creating 13
  creation steps 11
  enterprise monitoring 409
  Languages 11
  local installation 387
  modifying 34
  new files on your system 392
  regenerating 34
  removing 400
  solutions to problems 508
  system monitor 414
  testing 387
  uninstalling 400
AIX binary error logs
  monitoring 197
Application Component attribute 541
application support extension
  generating 606
  installing 607
application support extension project
  converting from a solution install project 607
  creating 595
Application Support Extension project
  creating 597
Application Support Extensions project
  adding support files 599
atof 82
atoi 82
attribute group 377
attribute group testing 377
  set configuration properties 379, 380
  set environment variables 379, 380
Attribute group testing 377
attribute groups
  creating a filtered attribute group 324
  deleting 323
  joining 319
attribute Information
  Fields and options for defining attributes 58
attributes 573
  adding 323
  Address 574
  AIX binary log attribute group 558
  Application Component 541
  Availability 541
  Average Collection Duration 550
  Cache Hit Percent 550
  Cache Hits 550
  Cache Misses 550
  Class Entry 559
  Command Line 544
  Compare String 569
  copying 62
  Counter Notifications 562
  Counter Value 564
  creating and editing 56
  DateTime 558
  derived 63
  Description 560
  Device Entry 574
  discovery attribute groups 580
  editing 62
  EntryTime 561
  enumeration 62
  enumeration attribute type 62
  enumeration type 62
  Error Code 547
  ErrptTimestampp 559
  Event Category 555
  Event ID 555
  Event Log 554
  Event Source 555
  Event Type 555
  Full Name 542
  Functionality Test Message 545
  Functionality Test Status 545
  Gauge Value 566
  High Threshold 566
  HTTP 576
  Identifier 559
  Interval 557
  Interval Unit 557
  Interval Unit Name 558
  Intervals Skipped 551
  JMX Event Attribute Groups 572
  joined 323
  Last Collection Duration 549
  Last Collection Finished 549
  Last Collection Start 549

© Copyright IBM Corp. 2007, 2013

attributes *(continued)*
LocalTimeStamp 557
log file RegEx statistics attribute group 589
log file status attribute group 585
Log File Summary 556
Log Name 554
LogFile 560
LogName 561
LogPath 561
Low Threshold 566
Message 556
Message Attribute 573
Modulus 563
monitor and notification attribute groups 561
Monitor ID 562, 565, 567, 569
Monitor Name 568
Monitor Parameters 568
Name 542, 574
Node 541, 546, 551, 554, 556, 558, 562, 564, 567, 568, 572, 573, 576, 579, 580
Node Description 575
Node Status 575
Node Type 575
Notification Message 564, 567, 570
Notification Time Stamp 564, 566, 570
Notification Type 562, 565, 569
Number of Collections 550
numeric 60
numeric options 60
Object Name 546
Object Name Time 580
Object Response Tim 580
Object Size 580
Object Status 547
Object Type 546
Observed Attribute 563, 566, 569
Observed MBean 563, 565, 569
Occurrences 557
Offset 563
Page Faults Per Sec 543
Page Objects 577
Page Size 577
Page Title 577
Percent Privileged Time 544
Percent Processor Time 544
Percent User Mode Time 544
Performance Object Status 545
performing operations 63
PID 544
Query Name 546
Refresh Interval 550
Registered Monitors 567
removing 63
reordering 323
ResourceName 560
Response Code 578
Response Time 574, 577
Sequence Number 572
Server Type 578
Source 572
Status 542, 578
Status Timestamp 575
string 59
String Notifications 568
String Value 570
Subnode Affinity 581
Subnode MSN 581
attributes *(continued)*
Subnode Resource Name 582
Subnode Type 581
Subnode Version 582
System 560
Take Action Status attribute group 582
Thread Count 543
Thread Pool Active Threads 552
Thread Pool Avg Active Threads 552
Thread Pool Avg Job Wait 554
Thread Pool Avg Queue Length 553
Thread Pool Max Active Threads 553
Thread Pool Max Queue Length 553
Thread Pool Max Size 552
Thread Pool Min Active Threads 552
Thread Pool Min Queue Length 553
Thread Pool Queue Length 553
Thread Pool Size 551
Thread Pool Status 551
Thread Pool Total Jobs 554
Threshold 563
Time Generated 556
timestamp 60
Timestamp 541, 546, 551, 557, 562, 565, 567, 568, 572, 573, 576, 579, 581
Total Object Size 577
Type 542, 559, 572
types 59
URL 576, 579
URL Alias 578
User Data 573, 579
Virtual Size 543
WMI 107
Working Set Size 543
attributes reference 541
authorization 6
availability
data types 1
availability filters
removing 323
average 82
Average Collection Duration attribute 550

B

basic agent, creating 13

C

Cache Hit Percent attribute 550
Cache Hits attribute 550
Cache Misses attribute 550
call sp_helpdb 251
call:2 sp_helpdb master 251
call[:index] procedureName [argument] ... 251
changes
saving 9, 50
CIM
configuration 174
monitoring 171
Class attribute 559
clearing the agent from the Tivoli Enterprise Portal 401
cognos 613, 622
Cognos data model
generating 609
Prerequisites 609

Cognos Data Model
Install and Configure Framework Manager 612
Install and Configure Tivoli Common Reporting 612
Tivoli Data Warehouse 609
Cognos information 42
collecting data from a remote system 227
command file
editing 224
Command Line attribute 544
command return code
monitoring 213
command-line interfacecommands 537
commands
configuring a subnode 351
generatelocal 538
generatemappingfile 538
generatezip 539
INSTALL_DIR/_uninst/uninstaller.bin 8
starting Agent Builder 11
tacmd configureSystem 354
committing
steps 45, 51
Common Information Model 1
external data sources 1
monitoring 171
Compare String attribute 569
configuration
Agent Editor 369
CIM 174
custom 361
HTTP 269
JDBC 250
JMX 156
overriding custom 347
Secure Shell remote connection 375
subnodes 344, 348
Windows data sources 358
Windows remote connection 372, 373
configuration problems
solutions 497
configuring
tuning 430
count 82
Counter Value attribute 564
creating a basic agent 13
creating and defining the agent 19
Creating deployable bundles for Tivoli Netcool OMNIbus probes 641
Creating Reports
Populating the ManagedSystem Table 618
Running the DB2 stored procedure 619
Running the Oracle stored procedure 620
Running the stored procedure 620
creating subnodes 335
Agent Editor 343
Data Source Location window 335
Data Sources tree 339
New Agent Wizard 335, 339
cumulativeSum 82
custom configuration 361
overrriding 347

D

data
trace logs 491
types monitored by agent 1
data collection
Windows event log 201
data from different sources
monitoring 331
data providers
subnodes 332
data source 24
Data Source Definition page 46, 47
data sources
AIX binary error logs 197
CIM 171
command return code 213
configuration
java api 316
defining additional 27
editing properties 55
HTTP 261
proxy server 269
ICMP 255
configuration files 257
index 25
java api
generated sample application 308
initializing the java application 307
Java API 301
configuration 316
Java Management Extensions 143, 165
JDBC 239
JMX 143, 165
monitoring 159
starting and stopping monitors 165
joining 319
manipulating attributes 323
log file 177
moving existing 32
ping 255
configuration files 257
process 89
script 229
script output 227
remote system 227
Simple Network Management Protocol 121
events 133
snmp event
configuration properties 138
SOAP 273
socket 285
configuration 296
data encoding 294
errors 295
remote port connection 296
sample script 297
sending information 291
Windows event log 201
Windows Performance Monitor 113
Windows service 97
WMI 105
data types 24
DateTime attribute 558
defining
connections 94
derived attributes 63
editing 67
examples 82
Description attribute 560
developerWorks 655
Device Entry attribute 574

dynamic file name support 643

E

eclipse
  detailed information 2, 66
  working sets 20
edits
  saving 9, 50
enterprise monitoring agents 409
EntryTime attribute 561
environment variables 435
  editing 39
Error Code attribute 547
ErrptTimestamp attribute 559
Event Category attribute 555
event data 433
event duplicate detection 417
event filtering and summarization
  controlling duplicate events 417
Event filtering and summarization
  viewing in the Tivoli Enterprise Portal 419
event flow control 417
Event ID attribute 555
event identifiers
  filtering 209
Event Source attribute 555
event source, filtering 207
event type 206
Event Type attribute 555
event type, filtering 206
eventThreshold 82
excessive data size 499
existing data sources
  moving 32
expression matching flags, regular 630
expressions
  ICU regular 627
  regular metacharacters 627
external data sources 1
  data types 1
  navigator groups 30
external scripts
  external data sources 1

F

fields
  JMX 165
  Process Monitor 89
file
  silent.txt 8
  uninstaller 9
file bundles 641
  adding commands 635
  adding files 637
  adding prerequisites 636
  generating the bundle 639
  remote deploy editor 633
file consolidation
  file separation 225
files
  agent trace 491
  installation trace 491
  log 425
  other trace log 493
files *(continued)*
  revolving message 430
  trace logs 491
filter
  attribute groups 319
filtered attributes
  examples 82
filtering
  event type 206, 207
Filtering
  attribute groups 67
filters
  availability 323
flags
  regular expression matching 630
formula editor
  attribute component 71, 76
  changing the component view 70
  component types 70
  conditional expression component 76
  literal component 73
  operator component 74
Formula Editor 68
  common options 78
  formula errors 80
full agent testing 382
Full Name attribute 542
Functionality Test Message attribute 545
Functionality Test Status attribute 545
functions
  abs 82
  atof 82
  atoi 82
  average 82
  count 82
  cumulativeSum 82
  eventThreshold 82
  getenv 82
  ipAddressToName 82
  isSummaryEvent 82
  itoa 82
  last 82
  max 82
  min 82
  NetWareTimeToTivoliTimestamp 82
  occurrenceCount 82
  replaceAll 82
  replaceFirst 82
  round 82
  sqrt 82
  stddev 82
  StringToTivoliTimestamp 82
  sum 82
  summaryEvent 82
  TivoliLogTimeToTivoliTimestamp 82
  tokenize 82
  UTCtoGMT 82
  UTCtoLocalTime 82

G

Gauge Value attribute 566
generate agent wizard 44
generatelocal command 538
generatemappingfile command 538
generatezip command 539
getenv 82

H

High Threshold attribute 566
HTTP 1
  configuration 269
  monitoring 261
  tables 264
HTTP attributes
  specific fields 264
Hypertext Transfer Protocol 1

I

IBM software support
  gather product information 425
IBM Tivoli Distributed Monitoring and Application Management Wiki 532
Identifier attribute 559
incorporating queries 409
incorporating situations 409
incorporating workspaces 409
install packages
  procedure 390
INSTALL_DIR/_uninst/uninstaller.bin 8
installation
  guidelines 5
  log file 491
  silent 8
installation problems
  solutions 497
installing
  Agent Builder 6
  DVD 6
installing agents
  after you install 390
  local 387
  over an existing agent 34
Integrated Service Management Library 655
Interval attribute 557
Interval Unit attribute 557
Interval Unit Name attribute 558
Intervals Skipped attribute 551
ipAddressToName 82
isSummaryEvent 82
itoa 82

J

Java API 1
  monitoring 301
Java application programming interface 1
Java Database Connectivity
  monitoring 239
Java Management Extensions 143
  external data sources 1
JDBC 1, 239
  monitoring 239
JDBC configuration 250
JDBC data source procedure 239
JMX
  configuration 156
  external data sources 1
  fields 165
  monitoring
    add counter metric watcher 163
    add gauge metric watcher 162
    add string metric watcher 161
JMX *(continued)*
  monitoring *(continued)*
    delete metric watcher 164
    take action commands 161
  notifications 159
  operations 164
joined
  attributes 323
joining
  data sources 319

K

key
  missing attribute 501

L

last 82
Last Collection Duration attribute 549
Last Collection Finished attribute 549
Last Collection Start attribute 549
list of messages 444
local installation 387
LocalTimeStamp attribute 557
log file
  monitoring 177
log file parsing and separators 192
log files
  external data sources 1
Log Name attribute 554
LogFile attribute 560
logging
  agent trace logs 491, 493
  installation log files 491
  trace log files 491
LogName attribute 561
LogPath attribute 561
logs
  trace 426
Low Threshold attribute 566

M

max 82
Message attribute 556, 573
messages
  contents 443
  for Agent Builder 444
  format 443
metacharacters
  regular expression 627
metrics
  availability 517
  service monitoring 517
  zeros returned 517
MIB errors 128
MIB Parsing
  SNMP 128
min 82
Modulus attribute 563
monitor https URLs 269
Monitor ID attribute 562, 565, 567, 569
Monitor Name attribute 568
Monitor Parameters attribute 568
monitoring 229

monitoring *(continued)*
AIX binary error logs 197
CIM 171
command return code 213
Common Information Model 171
Java API 301
Java Database Connectivity 239
JMX 159
log file 177
multiple types of information 331
Perfmon 113
process 89
same data from different sources 331
script output 227
SNMP 121
socket 285
Windows Event Log 201
Windows Management Instrumentation 105
Windows Performance Monitor 113
Windows service 97
monitoring multiple types of information 331
monitoring the same data from different sources 331
monitoring types 24
multiple instances 23
multiple types of information
monitoring 331

N

Name attribute 542, 574
namespaces 107
naming the agent 20
navigator group
creating 327
navigator groups 30
NetWareTimeToTivoliTimestamp 82
new agent wizard, Starting 13
new in this release 2
Node attribute 541, 546, 551, 554, 556, 558, 562, 564, 567, 568, 572, 573, 576, 579, 580
Node Description attribute 575
Node Status attribute 575
Node Type attribute 575
Notification Message attribute 564, 567, 570
Notification Time Stamp 566
Notification Time Stamp attribute 564, 570
Notification Type attribute 562, 565, 569
Number of Collections attribute 550

O

Object Name attribute 546, 580
Object name field 114
Object Response Time attribute 580
Object Size attribute 580
Object Status attribute 547
Object Type attribute 546
objects, browsing 114
Observed Attribute attribute 563, 566, 569
Observed MBean attribute 563, 565, 569
occurrenceCount 82
Occurrences attribute 557
Offset attribute 563
operating system requirements 13
operating systems 5
agent defaults 37
operating systems *(continued)*
specifying 88
organizing the agent 29

P

packages
creating 389
installing 390
Page Faults Per Sec attribute 543
Page Objects attribute 577
Page Size attribute 577
Page Title attribute 577
Percent Privileged Time attribute 544
Percent Processor Time attribute 544
Percent User Mode Time attribute 544
Perfmon 113
external data sources 1
Performance Monitor 1
Performance Monitor objects 114
PID attribute 544
Ping 573
platforms 5
prerequisites, software 5
private situations 414
creating 403
problem determination 425
procedure 239
procedures
stored 251
process
monitoring 89
process data unavailable 517
Process Monitor fields 89
processes
connections 94
product code
changing 52
product information
gathering for IBM Software Support 425
project
contents 20
name 11
project files
sharing 533

Q

queries
creating 403
incorporating 409
preparing to create 403
Query Name attribute 546
quick reference, procedures 3

R

RAS trace parameters
setting 495
Redbooks 655
Refresh Interval attribute 550
regular expression matching flags 630
regular expression metacharacters 627
remote
process browsing 94
removing an agent using the Tivoli Enterprise Portal 401

removing an agent without using the Tivoli Enterprise Portal 401
replaceAll 82
replaceFirst 82
Report creation
  solutions to problems 529
reports
  creating 613
  importing 622
requirements
  software product compatibility report 6
resource models
  upgrading 625
ResourceName attribute 560
Response Code attribute 578
Response Time attribute 577
Response Time Entry attribute 574
round 82
rules for naming agents 20
Runtime Configuration Information page 48

S

same data from different sources
  monitoring 331
Sampled Data 431
Saving your edits and changes 9, 50
script 229
script data source 229
script data sources
  subnode configuration 359
script output
  monitoring 227
script parsing and separators 228
Secure Shell remote connection 375
self-describing agent 38
Sequence Number attribute 572
server definitions 409
Server Type attribute 578
Service Management Connect 655
setup guidelines 3
silent installation 8
silent uninstallation 9
silent.txt file 8
Simple Network Management Protocol 1, 121
Simple Object Access Protocol 1
situations
  creating 403
  exporting 409
  importing 409
  incorporating 409
  preparing to create 403
  private 414
SMC 655
SNMP
  MIB Parsing 128
  monitoring 121
SOAP 1, 273
  configuration 281
  monitoring 273
  Xpaths 279
socket
  monitoring 285
Socket 1
software product compatibility report 6
Software Support 532
solution installer project
  sharing 535
Source attribute 572
sp_helpdb procedure 251
space requirements 5
sqrt 82
SSHEXEC
  action 651
starting the Agent Builder 11
Starting the new agent wizard 13
Status attribute 542, 578
status of subnodes 333
Status Timestamp attribute 575
status unknown 517
stddev 82
stored procedures 251
  Oracle 251, 253
  samples 251
  sp_helpdb 251
String Value attribute 570
StringToTivoliTimestamp 82
Subnode Affinity attribute 581
subnode configuration
  overcoming limitations 348
  script data sources 359
Subnode MSN attribute 581
Subnode Resource Name attribute 582
Subnode Type attribute 581
Subnode Version attribute 582
subnodes 335
  Agent Editor 343
  configuration 344
    examples 352
    overriding custom configuration 347
  configuration properties 348
  configuring 345
  configuring from the command line 351
  creating 331
  creating as first component 335
  data providers 332
  from Data Sources tree 339
  how they work 30
  in Navigator tree 31
  script data sources 359
  status 333
  tasks accomplished with 30
  Windows data sources 358
sum 82
summaryEvent 82
support
  list of messages 444
System attribute 560
system monitor agents 414
  creating situations 403

T

tacmd configureSystem command 354
Take Action Command
  SSHEXEC 651
take action commands
  reference 651
Take Action commands
  creating 403
Technotes 655
testing 377, 382
  CIM data source 174

testing *(continued)*
environment variables 386
HTTP data source 270
Java data source 317
JDBC data source 253
JMX data source 169
log file data source 193
perfmon data source 118
ping data source 259
SNMP data source 130
SNMP event data source 140
SOAP data source 282
socket data source 298
WMI data source 111
testing and debugging your agent 377
Thread Count attribute 543
Thread Pool Active Threads attribute 552
Thread Pool Avg Active Threads attribute 552
Thread Pool Avg Job Wait attribute 554
Thread Pool Avg Queue Length attribute 553
Thread Pool Max Active Threads attribute 553
Thread Pool Max Queue Length attribute 553
Thread Pool Max Size attribute 552
Thread Pool Min Active Threads attribute 552
Thread Pool Min Queue Length attribute 553
Thread Pool Queue Length attribute 553
Thread Pool Size attribute 551
Thread Pool Total Jobs attribute 554
Threshold attribute 563
Time Generated attribute 556
Timestamp attribute 541, 546, 551, 557, 562, 565, 567, 568, 572, 573, 576, 579, 581
Tivoli Common Reporting
exporting reports and data models 620
Installing reports from an agent package to 624
Tivoli Data Warehouse
Install and Configure 610
Populating with the Tivoli Reporting and Analytics Model 611
Tivoli Monitoring Agent Builder, overview 1
TivoliLogTimeToTivoliTimestamp 82
tokenize 82
Total Object Size attribute 577
trace logging 426
configuration 428
example 427
format 426
location 426
trace logs 491
trap configuration 414, 647
trapcnfg configuration file 647
defaults in 647
overview 647
types of records in 647
trapcnfg defaults
supported categories 647
supported severities 647
supported source IDs 647
supported statuses 647
trapd.conf, using 647
troubleshooting 425
installation logs 491
problem classification 497
uninstallation logs 491
Type attribute 542, 559, 572

U

uninstall command
UNIX 401
Windows 401
uninstall script 401
uninstallation
silent 9
uninstallation problems
solutions 497
uninstallation, log file 491
uninstaller file 9
uninstalling
Agent Builder 8
uninstalling agent 400
URL
monitoring 267
URL Alias attribute 578
URL attribute 576, 579
URLs file 267
User Data attribute 573, 579
UTCtoGMT 82
UTCtoLocalTime 82

V

version
agent 22
IBM Tivoli Monitoring 23
version of the agent
limits 45, 51
Virtual Size attribute 543

W

watchdog information 41
Windows data sources
configuration 358
Windows event log
filtering
event type 206, 207
monitoring 201
Windows Event logs
data types 1
Windows Management Instrumentation 1
monitoring 105
Windows Performance Monitor 113
Windows remote connection 372, 373
Windows service
defining connections for service browsing 101
monitoring 97
wizard, Starting the new agent 13
WMI
external data sources 1
monitoring 105
Working Set Size attribute 543
working sets 20
workspace directory
location 11
workspaces
creating 403
exporting 409
importing 409
incorporating 409
preparing to create 403

Z

zeros returned for metrics 517



Printed in USA

SC32-1921-14

