IBM Tivoli Monitoring Agent Builder
Version 6.2.3.1

User's Guide

SC32-1921-13

IBM Tivoli Monitoring Agent Builder
Version 6.2.3.1

User's Guide

SC32-1921-13

Note
Before using this information and the product it supports, read the information in Appendix N, "Notices," on page 647.

This edition applies to version 6.2.3.1 of Tivoli Monitoring Agent Builder and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures ix

Tables xiii

Chapter 1. Introduction 1
About Tivoli Monitoring Agent Builder 1
About Eclipse 2
New in this release 2
About this guide 3

Chapter 2. Getting started quick-reference guide 5

Chapter 3. Installing the Tivoli Monitoring Agent Builder 7
Requirements 7
Authorization 8
Builder installation procedure. 8
After installing 9
Silent installation 9
Uninstalling Agent Builder 10
Silent uninstallation. 11

Chapter 4. Starting the Agent Builder 13
Planning and overview 13
Starting the Agent Builder 13

Chapter 5. Creating a basic agent . . . 15
Operating system requirements. 15
Starting the new agent wizard 15
Using the icon to start the wizard 16
Creating a new file to start the wizard 16
Creating and defining the agent 20
Naming your agent. 20
Defining initial data source 25
Data Sources 26
Selecting key attributes 27
Defining additional data sources 28
Organizing the agent 30
Navigator groups 31
Subnodes 31
Moving existing data sources 33
Preparing your agent for modifications or for regeneration 35

Chapter 6. Modifying your agent by using the Tivoli Monitoring Agent Editor 37
Tivoli Monitoring Agent Editor 37
Agent Information page 38
General agent information 39
Default operating systems 39
Self-Describing Agent 39
Environment variables. 40
Watchdog information 41
Cognos information 43
Generate Agent Wizard link 45
Commit Agent Version link 45
Data Source Definition page 45
Runtime Configuration Information page . . . 47
Agent XML Editor page 48
Saving your edits and changes 49
Committing a version of the agent. 49
Steps to commit the agent 50
Changing a subsequent version of your agent . . 50
Changing the product code 51

Chapter 7. Editing data source and attribute properties 53
Creating attributes 54
Fields and options for defining attributes . . . 55
Attribute types 57
String attributes 57
Numeric 58
Time stamp 58
Numeric attribute options 58
Specifying an enumeration for an attribute . . . 61
Copying attributes 61
Editing attributes 62
Removing attributes 62
Creating derived attributes 62
Editing derived attributes. 66
Filtering attribute groups 66
Formula Editor 66
Changing the component view 69
Component types 69
Attribute component 70
Literal component 72
Operator component 73
Conditional expression component 74
Function component 75
Common options 77
Insert 77
Remove 78
Up one Level 79
Edit 79
Formula errors 79
Formula operators and functions 81
Examples 85
Derived Attributes 85
Filtering 86
Specifying operating systems 86

Chapter 8. Monitoring a process. . . . 89
Defining connections for process browsing 94

Chapter 9. Monitoring a Windows service. 97

Defining connections for service browsing 101

Chapter 10. Monitoring data from Windows Management Instrumentation (WMI) 105
Testing 110

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

Chapter 12. Monitoring data from a Simple Network Management Protocol (SNMP) 121
MIB errors 126
SNMP MIB Parsing options 127
Testing 128

Chapter 13. Monitoring events from a Simple Network Management Protocol event sender. 131
Configuration 135
Testing 137

Chapter 14. Monitoring Java Management Extensions (JMX) MBeans 141
JMX configuration 157
JMX notifications, monitors, and operations . . . 160
JMX notifications 160
JMX monitors 161
Take Action commands for monitors . . . 162
JMX Add String Metric Watcher 162
JMX Add Gauge Metric Watcher 163
JMX Add Counter Metric Watcher . . . 164
JMX Delete Metric Watcher 165
Starting and stopping monitors 165
JMX operations 166
Take Action command syntax 166
Example: Invoke an operation to reset a counter 166
Example: Invoke an action with an argument 166
Running the JMX_INVOKE Take Action command 167
Specific fields for Java Management Extensions (JMX) MBeans 167
Testing 170

Chapter 15. Monitoring data from a Common Information Model (CIM) . . 173
Steps for monitoring data from a CIM 173
CIM configuration 176
Testing 177

Chapter 16. Monitoring a log file . . . 179
Testing 195

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 206
Filtering by event identifier. 208

Chapter 19. Monitoring a command return code 211
Editing a command file definition 222
File Separation & Consolidation 223

Chapter 20. Monitoring output from a script 225
Collecting data from a remote system 225
Script parsing and separators 225
Examples 226
Simple script output 226
Complex script output 226
Steps for monitoring output from a script 227

Chapter 21. Monitoring data from Java Database Connectivity (JDBC) 237
Procedure 237
JDBC configuration 249
Stored procedures 251
SQL Server Samples 251
call sp_helpdb 251
call:2 sp_helpdb master 251
DB2 stored procedure 251
Oracle stored procedures 252
Testing 252

Chapter 22. Monitoring system availability using Ping 255
Ping configuration files 255
Procedure 256
Configuration 258
Testing 258

Chapter 23. Monitoring HTTP availability and response time 261
HTTP tables 263
Specific fields for HTTP attributes 264
Monitoring a URL 268
URLs file 268
Take Action option 269
Monitoring https:// URLs 269
Proxy server 270
HTTP configuration 270
URL Monitoring configuration. 270
Proxy Server configuration 270
Java configuration 270
Testing 271

Chapter 24. Monitoring data using SOAP 273
Specific fields for SOAP attributes 277
XPath options 280
Proxy server 281
SOAP configuration 281
HTTP Server 281
Testing 281

Chapter 25. Monitoring data using Socket 283
Sending information to the agent 288
Sending data 288
Sending errors instead of data 290
Handling take actions 290
Data encoding 291
Special characters 291
Character sets 292
Numeric Data 292
Errors 292
Limitations of the Socket data source 293
Socket configuration 294
Sample script for socket 295
Perl sample 295
Testing 295

Chapter 26. Monitoring data using Java API 297
Initializing the Java application 303
Dependencies 303
Generated sample Java application 304
Class structure 305
Initialization and clean up 306
Collecting sampled attribute group data . . . 307
Collecting sampled data for a subnode 308
Sending events 309
Sending events in a subnode 309
Take actions 310
Handling exceptions 310
Error codes 311
Changes to the agent 312
Using the Java API 313
Java API configuration 314
Testing 314

Chapter 27. Creating attribute groups from existing sources 317
Joining two attribute groups 318
Deleting an attribute group 321
Deleting an attribute 322
Adding an attribute 322
Reordering attributes 322
Removing availability filters 322
Joined attributes 322
Filtered Attribute Groups 323
Creating a Filtered Attribute Group 324

Chapter 28. Creating a navigator group 327

Chapter 29. Creating subnodes 331
Monitoring the same data from different sources 331
Monitoring multiple types of information 332
Data Providers in subnodes 333
Status of subnodes 333
Creating subnodes when creating an agent . . . 335
Creating a subnode as the first component of an agent from a Data Source Location page in the New Agent Wizard 335
Creating a subnode from the Data Sources tree in the New Agent Wizard 339
Creating a subnode in the Agent Editor . . . 343
Subnode configuration 344
Configuring a subnode 345
Overriding custom configuration 346
Selecting subnode configuration properties 347
Advanced subnode configuration 347
Configuring a subnode from the command line 350
Windows data sources 356
Script data sources 357

Chapter 30. Configuring an agent . . . 359
Customizing configuration 359
Changing configuration properties using the Agent Editor 367
Configuring a Windows remote connection . . . 370
Configuring a Secure Shell (SSH) remote connection 371

Chapter 31. Testing and debugging your agent 373
Attribute group testing 373
Configuration prior to testing 374
Setting Java Information 376
Debugging 376
Installing and testing an agent 377
Installing the agent locally 377
Creating the agent image 379
Installing the package 380
After you install the agent 381
Configuring and starting the agent 382
Results after generating and installing the agent with the Agent Builder 382
New files on your system 382
Changes in the Manage Tivoli Enterprise Monitoring Services window 385
Changes in the Tivoli Enterprise Portal 386
Uninstalling an agent 389
Removing an agent using the Tivoli Enterprise Portal 389
Removing an agent without using Tivoli Enterprise Portal 390
Windows systems 390
Command 390
Uninstallation script 390
UNIX systems 390
Clearing the agent from the Tivoli Enterprise Portal after removing the agent 390

Chapter 32. Creating workspaces, Take Action commands, and situations 391
Creating situations, Take Action commands, and queries 391
Creating workspaces 391

Chapter 33. Importing application support files 397
Exporting and importing server definitions . . . 397

Chapter 34. Event filtering and summarization 405
Controlling duplicate events 406
Viewing event filtering and summarization in the Tivoli Enterprise Portal 407

Chapter 35. Troubleshooting 413
Gathering product information for IBM Software Support 413
Agent Builder trace logging 413
Agent Builder trace logging overview 414
Trace log location 414
Trace log format 414
Trace log example 415
Logging configuration 416
Configuration options 416
IBM Java Logging Toolkit 416
Output window 417
Configuring and Tuning data collection 418
Data types 418
Sampled data 418
Environment variables 420
Attribute groups 421
Event Data 422
Examples and Advanced Tuning 423
Example 423
Environment variables 424
Understanding informational, warning, and error messages 430
Product messages 431
Agent Builder messages 431
Agent trace logging 476
Overview of log file management 476
Examples of trace logging 477
Principal trace log files 477
Example for using trace logs 479
Setting RAS trace parameters 480
Objective 480
Background Information. 480
Before you begin 481
After you finish 481
Procedure 481
Problem classification. 483
Troubleshooting: Agent Builder 483
Troubleshooting: Agents 493
Troubleshooting: Report Creation. 513
Support information 517

Appendix A. Sharing project files . . . 519

Appendix B. Command-line options 523
List of commands 523
generatelocal 525
generatemappingfile 526
generatezip 527

Appendix C. Attributes reference . . . 529
Availability node 529
Performance Object Status node 534
Thread Pool Status attribute group 540
Event log attribute node. 544
Log File Summary. 546
AIX Binary Log attribute group 548
Monitor and Notification attribute groups 551
Counter Notifications. 551
Gauge Notifications 554
Registered Monitors 557
String Notifications 558
SNMP Event attribute groups 560
JMX Event attribute groups. 562
Ping attribute group 563
HTTP attribute groups 566
Managed URLs. 566
URL Objects. 569
Discovery attribute groups 571
Take Action Status attribute group 572
Log File Status attribute group 576
Log File RegEx Statistics attribute group 580

Appendix D. Creating application support extensions for existing agents 585
Creating a new Application Support Extension project. 585
Adding your support files to an Application Support Extension project 587
Generating the Application Support Extension install image. 593
Installing your Application Support Extension . . 594
Converting an existing Solution Install Project to an Application Support Extension project 595

Appendix E. Cognos data model generation. 597
Prerequisites. 597
Tivoli Data Warehouse 597
Create tables and Procedures in the Tivoli Data Warehouse 597
Create or alter the ManagedSystem Table and Stored Procedure in the Tivoli Data Warehouse 597
DB2 598
Oracle. 598
SQL Server 2005 and 2008 599
Tivoli Reporting and Analytics Model . . . 599

Populating the Tivoli Data Warehouse with the Tivoli Reporting and Analytics Model . . . 599
Tivoli Common Reporting . . . 600
Installing Tivoli Common Reporting . . . 600
Configuring Tivoli Common Reporting . . . 600
Framework Manager . . . 600
Installing Framework Manager . . . 600
Configuring Framework Manager . . . 601
Creating reports . . . 601
Prerequisites . . . 601
Opening the Agent Data Model in Framework Manager . . . 601
Populating the ManagedSystem Table . . . 602
Running the stored procedure . . . 604
DB2 . . . 604
Oracle . . . 604
SQL Server 2005 and 2008 . . . 604
Publishing the Agent Data Model to Tivoli Common Reporting . . . 604
Creating reports in Tivoli Common Reporting 605
Exporting reports and data models from Tivoli Common Reporting . . . 607
Importing reports into Agent Builder . . . 609
Installing reports from an agent package into Tivoli Common Reporting . . . 611

Appendix F. Upgrading custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6.2 agents . . . 613

Appendix G. ICU regular expressions 615
Replacement text . . . 618
Flag options . . . 619

Appendix H. Non-agent bundles . . . 621
Creating a file bundle . . . 621
Remote Deploy Bundle Editor . . . 623
Adding commands to the bundle . . . 625
Adding prerequisites to the bundle . . . 626
Adding files to the bundle . . . 627
Generating the bundle . . . 628
Creating deployable bundles for Tivoli Netcool/OMNIbus probes . . . 630

Appendix I. Dynamic file name support . . . 631
Regular expression file-name patterns . . . 631
Dynamic file name syntax . . . 631

Appendix J. SNMP trap configuration 635
SNMP trap configuration file, trapcnfg . . . 635
Using the HP OpenView `trapd.conf` file . . . 635
Types of records . . . 635
Defaults for the trapcnfg file . . . 636
Supported categories . . . 636
Supported statuses . . . 637
Supported source IDs . . . 637

Appendix K. Take Action commands reference . . . 639
About Take Action commands . . . 639
More information about Take Action commands 639
Special Take Action commands . . . 639
SSHEXEC action . . . 640

Appendix L. Documentation library 641
Tivoli Monitoring library . . . 641
Documentation for the base agents . . . 642
Related publications . . . 643
Other sources of documentation . . . 643

Appendix M. Accessibility . . . 645
Using assistive technologies . . . 645
Magnifying what is displayed on the screen . . . 645
Documentation in accessible formats . . . 645
Using alternative text . . . 645

Appendix N. Notices . . . 647
Trademarks . . . 649

Index . . . 651

Figures

1. Create New Agent icon 16
2. Starting the wizard with the File > New > **IBM Tivoli Monitoring Agent** option. . . . 16
3. Starting the wizard with the File > New > Other option 17
4. Select a wizard page 18
5. Selecting the agent icon 19
6. Welcome window 20
7. Project page 21
8. General Information page 22
9. Agent Information page 23
10. Agent Initial Data Source page 25
11. Select key attributes page 28
12. Data Source Definition page: defining additional monitors 29
13. Data Source Location page 30
14. Subnodes in the Navigator tree 33
15. Move existing data sources 34
16. Currently Defined Data Sources page 35
17. Choosing Reset from the Tivoli Monitoring perspective button 37
18. Agent Information page 38
19. Default Operating Systems page 39
20. Self-Describing Agent page 40
21. Environment Variables page 40
22. Environment Variable Information window 41
23. Watchdog Information page 42
24. Cognos Information page 44
25. Data Source Definition page for Cognos 45
26. Data Source Definition page 47
27. Runtime Configuration Information page 48
28. Agent XML Editor page 49
29. Updating the agent version after commit 51
30. Agent Product Code change 51
31. Agent support files invalidated 52
32. Attribute Information page 55
33. String attribute type. 58
34. Numeric attribute type. 58
35. Defining enumeration for an attribute. . . . 61
36. Copy Attribute window 62
37. Adding an attribute. 63
38. Attribute Information page 64
39. Attribute Information page with Derived Attribute Details tab open. 65
40. Derived Formula Editor window (default) 68
41. Derived Formula Editor view with formula hierarchy tree view 69
42. Edit the Selected Attribute page. 70
43. Edit the Selected Attribute page with error 71
44. Edit the Selected Function page with warning 72
45. Edit the Selected Literal page 73
46. Edit the Selected Operator page. 74
47. Edit the Selected Operator page (for a conditional operator) 75
48. Edit the Selected Function page 76
49. Insert window example 77
50. Insert window example for getenv function 78
51. Remove function and its Arguments page 78
52. Empty formula example 79
53. Parsing error example 80
54. Insert window for fixing a formula manually 81
55. Operating systems where the agent is to run 87
56. Adding a server process 89
57. Process Monitor page 90
58. Browse the list of currently running processes 91
59. Search for a process. 92
60. Process Monitor page example 93
61. Select a connection type or connection template for process browsing 94
62. Edit connection properties for Tivoli Enterprise Portal Server Managed System 95
63. Adding a Windows service monitor 97
64. Service Monitor page 98
65. Browse the list of all services defined in the system 99
66. Search for a service 100
67. Service information 101
68. Select a connection type or connection template for service browsing 102
69. Edit connection properties for Tivoli Enterprise Portal Server Managed System . . 103
70. Adding WMI data 105
71. WMI Information page 106
72. List of classes with their associated attributes 107
73. Optionally specifying WMI search options 107
74. WMI Information page completed 108
75. WMI shown on Data Source Definition page in the Agent Editor 109
76. Global Windows Data Source Options window 110
77. WMI Test window 111
78. Adding Perfmon data. 113
79. Perfmon Information page 114
80. Browsing for Performance Monitor objects 115
81. Performance Monitor Search window 115
82. Perfmon Information page with Object Name 116
83. Global Windows Data Source Options window 116
84. Perfmon Data Source Definition 118
85. Perfmon Test window. 119
86. Adding SNMP data 121
87. Simple Network Management Protocol Information page 122
88. List of objects 123
89. Selecting an object 124
90. Simple Network Management Protocol Information page 125
91. SNMP Data Source Definition 126
92. MIB Parsing window 127
93. SNMP Attribute Group Test selection 128
94. SNMP Test settings window 129
95. Create Connection Wizard 129

96. Adding SNMP events 131
97. Simple Network Management Protocol Event Information window 132
98. SNMP MIB Browser window 133
99. Select key attributes page 135
100. Runtime Configuration page 136
101. Test Event Settings window 138
102. Test Event Settings window showing collected SNMP event data 138
103. Data Collection Status window 139
104. Adding JMX data 142
105. JMX Information page 143
106. JMX Agent-Wide options window. 144
107. JMX Browser with no connection selected 145
108. JMX Connection Selection page 146
109. JMX connection templates 147
110. JMX connection properties 148
111. Java Management Extensions (JMX) Browser window 150
112. JMX Information page 153
113. JMX Agent-Wide Options window 154
114. New attribute group in Agent Editor 155
115. JMX Agent-Wide Options window 156
116. Runtime Configuration tab of the Agent Editor 157
117. JMX Agent-Wide Options page 161
118. Attribute Information page 167
119. JMX Test window 171
120. Adding CIM data 173
121. Common Information Model (CIM) Information page 174
122. Common Information Model (CIM) Class Browser window 175
123. CIM Test Settings window 177
124. Adding a log file 179
125. Log File Information page 180
126. XML Browser window 181
127. Advanced Data Source Properties page, File Information tab 182
128. Advanced Data Source Properties page, Record Identification tab 185
129. Advanced Data Source Properties page, Filter Expression tab 186
130. Advanced Data Source Properties page, Event Information tab 187
131. Attribute Information page 188
132. Advanced Log File Attribute Information page 190
133. Add Filter window 191
134. Add Filter example 1 193
135. Add Filter example 2 194
136. Parse Log window 195
137. Parse Log window showing parsed logfile attribute values 196
138. Data Collection Status window 196
139. Adding an AIX Binary Log 198
140. Binary Log Information page 199
141. Adding a Windows Event log 202
142. Windows Event Log page 203
143. Windows Event Log Bookmark Settings window 204
144. Global Windows Data Source Options window 205
145. Windows Event Log page 206
146. Event Source window. 207
147. Event Log Source Browser window 207
148. Windows Event Log window 208
149. Event Identifier window 209
150. Windows Event Log window 210
151. Adding data from a command 212
152. Command Return Code page 213
153. Command Information window 214
154. Return Code Definition window: Return code type. 215
155. Return Code Definition window: Return code value 215
156. Messages window 216
157. Message Definition window 217
158. Return Code Definition window completed 217
159. Command Information window completed 218
160. Return Code Definition window: Message text 220
161. Messages window 220
162. Return Code Definition window 221
163. Command Return Code page completed 222
164. Import Command File window 223
165. Example attribute value output when Agent parses complex script output. 227
166. Adding data from a script 228
167. Command List page 229
168. Command Information window for a script 231
169. Test Command window 232
170. Attribute Information page 234
171. Agent Editor Data Source Definition page 235
172. Adding JDBC data 238
173. JDBC Information page 239
174. JDBC Browser window 240
175. Database Connection Wizard: JDBC Connections page 241
176. Database Connections Wizard: Connection Properties page 242
177. Java Database Connectivity (JDBC) Browser window 244
178. JDBC Information page 245
179. Data Source Definition page Attribute Group Information 246
180. Runtime Configuration Information page, Configuration for Java Virtual Machine (JVM) 247
181. Attribute Information page 248
182. Test JDBC statement window 253
183. Adding Ping data 256
184. Ping Information window 257
185. Ping Test Settings window 258
186. Adding HTTP data 262
187. HTTP Information window 263
188. Attribute Information page 265
189. HTTP Test window 271
190. Adding HTTP data 273
191. SOAP Information page 274
192. SOAP Browser window 275
193. SOAP Browser window 276
194. SOAP Browser window 277

195. Attribute Information page 278
196. Test SOAP Collection window 282
197. Monitoring data with Socket 283
198. Socket Information window. 284
199. Global Socket Data Source Information page 285
200. Socket Error Code Definition window, 286
201. Messages window 286
202. Message Definition window 287
203. Test Socket Client window 296
204. Monitoring data with Java API. 298
205. Java API Information window 299
206. Global Java API Data Source Information page 300
207. Java API Error Code Definition window, 301
208. Messages window 301
209. Message Definition window 302
210. Sample agent structure 305
211. Test Java Client window 315
212. Adding existing data sources 318
213. Attribute Group Information window 321
214. Locating source attribute information 323
215. Adding existing data sources 324
216. Filter Information window 325
217. Selecting the initial source of monitoring data for a new agent 327
218. Providing the navigator group name and help text 328
219. Selecting the first source of monitoring data 329
220. Subnodes monitoring different systems 331
221. Subnode types in Navigator tree 332
222. Monitoring multiple subnode instances of the same subnode type 334
223. Example: data collection in a subnode 335
224. Data Source Location page - Creating a subnode 336
225. Subnode Information page 337
226. Initial Subnode Data Source page 338
227. Subnode Configuration Overrides page 339
228. Data Source Definition page 340
229. Defining a new subnode 341
230. Initial Subnode Data Source page 342
231. Subnode Configuration Overrides page 343
232. Agent Editor Data Source Definition page 344
233. Subnode Configuration Overrides window 345
234. Overriding custom configuration 346
235. Select Configuration Section window 347
236. Override configuration drop-down lists 348
237. Sections to override 349
238. SNMP Version 1 Properties expanded 350
239. Configuration property definitions in the Agent Builder 351
240. **Top** section with agent-level configuration for the **Agent Cfg** property 352
241. **Main** section with the agent-wide default value for the **Overridable Cfg** property. . . 353
242. **Example Subnode** section page with no subnode 354
243. **Example Subnode** section page with two subnode instances defined 355
244. Windows Management Instrumentation (WMI) Information page 356
245. Global Windows Data Source Options 357
246. **Insert Configuration Property** button on the Log File Information page 360
247. Initial Configuration Properties window 361
248. Runtime Configuration Property window 362
249. Configuration Property Default Values window 364
250. Adding a configuration property choice 364
251. New configuration property displayed 365
252. Adding a second property to an existing section 366
253. Property inserted into the log file name 367
254. **Runtime Configuration** tab in the Agent Editor 368
255. Editing a configuration section in the Agent Editor 369
256. Editing a configuration property in the Agent Editor 370
257. WMI Test window 373
258. Environment Variables window 375
259. Runtime Configuration window 375
260. Java Information window 376
261. WMI Test window showing test log location. 377
262. Generate agent icon 378
263. Generate agent menu option 378
264. Create an agent image 380
265. Manage Tivoli Enterprise Monitoring Services window 386
266. Nodes for attribute groups you defined 387
267. Availability node 388
268. Performance Object Status node 388
269. Event log node 389
270. Setting the sysadmin user ID 392
271. Setting the sysadmin user ID (continued) 393
272. Setting the sysadmin user ID (continued) 393
273. Setting workspace properties 394
274. Setting workspace properties (continued) 395
275. Importing application support files 398
276. Importing situations 399
277. Selecting situations. 400
278. Select Take Action commands 401
279. Importing queries 402
280. Advanced Log File Attribute Group Information page, Event Information tab . . 407
281. Historical view and cache view when event filtering or summarization is not enabled . . 408
282. Historical view and cache view when Only send summary events is selected 409
283. Historical view and cache view when Send all events is selected 410
284. Historical view and cache view when Send first event is selected 411
285. Historical view and cache view when Event threshold is selected 412
286. IBM Java Logging Toolkit 417
287. Importing files 519
288. Importing files (continued) 520
289. Importing files (continued) 521
290. IBM Tivoli Monitoring Application Support Extension 586
291. Naming the project 587

292. Importing support files 588
293. Import Information 589
294. Affinities list 590
295. Import Information complete 590
296. Importing situations 591
297. Importing queries 592
298. Import Take Action commands. 592
299. Agent support files 593
300. Application Support Extension Install Image window 594
301. Selecting agent project file 602
302. Selecting Publish Packages 605
303. Selecting Common Reporting 606
304. Selecting Report Studio 606
305. Report Studio 607
306. Content Administration tab 608
307. Content Administration tab with agent package listed 609
308. Importing report packages 610
309. Import Report Package window 610
310. Select a wizard window 621
311. New Remote Deploy Project window 622
312. Remote Deploy Bundle Information window 623
313. Remote Deploy Bundle Editor 624
314. Command window 625
315. New Prerequisite window 626
316. Adding files to the bundle or generating a bundle from the Navigator tree 627
317. Import Bundle Files window 628
318. Generate Final Remote Deploy Bundle window 629
319. Examples of configuration record types 2 and 3. 636

Tables

1. Quick-reference information for creating agents 5
2. Quick-reference information for other functions 6
3. Guidelines for installation and configuration 7
4. Minimum version of IBM Tivoli Monitoring required by the agent 24
5. Fields for editing data sources 53
6. Fields and options for defining attributes 56
7. Numeric attribute options 59
8. Valid format parameters for StringToTivoliTimestamp 83
9. StringToTivoliTimestamp examples 84
10. Fields on the Process Monitor page 90
11. SNMP Events configuration properties 136
12. Filter options 243
13. Supported SQL data types for use with a monitoring agent 248
14. Network Management configuration properties 258
15. HTML elements searched for objects to monitor 264
16. HTTP Attribute Information - Managed URLs 266
17. HTTP Attribute Information - URL Objects 268
18. URLs file entries 268
19. URL Monitoring configuration properties 270
20. Proxy Server configuration properties 270
21. Java configuration properties 271
22. SOAP Attribute Information 279
23. HTTP Server configuration properties 281
24. File types for supplemental files 287
25. Sample error code 290
26. Characters to encode in attribute values 292
27. Performance Object Status values 293
28. Socket configuration property 294
29. File types for supplemental files 302
30. Java trace level options 307
31. Attribute field types and their IBM Tivoli Monitoring attribute type equivalents . . . 307
32. Internal error codes for the agent 311
33. Changes to an agent requiring modifications to the Java source 312
34. Java configuration properties 314
35. source attribute group one (single row) 319
36. source attribute group two (single row) 319
37. Resulting join 319
38. source attribute group one (single row) 320
39. source attribute group two (more than one row) 320
40. Resulting join 320
41. source attribute group one (more than one row) 320
42. source attribute group two (more than one row) 320
43. Resulting join (joining on Attribute3 and Attribute7) 320
44. Information to gather before contacting IBM Software Support 413
45. Environment variables 424
46. Trace log files for troubleshooting agents 478
47. Problems and solutions for installation and use of the Agent Builder 484
48. Problems and solutions for agents 495
49. Problems and solutions for Report Creation 514
50. Command quick-reference table 523
51. Required arguments 603
52. Optional arguments 603
53. Regular expression metacharacters 615
54. Regular expression operators 616
55. Replacement text characters 618
56. Flag options 619
57. Predefined Variables for Commands 626
58. Categories supported by the SNMP Data Provider 636
59. Severities supported by the SNMP Data Provider 637
60. Statuses supported by the SNMP Data Provider 637
61. Source IDs supported by the SNMP Data Provider 637

Chapter 1. Introduction

This chapter provides an overview of the IBM® Tivoli® Monitoring Agent Builder.

About Tivoli Monitoring Agent Builder

Tivoli Monitoring Agent Builder is a set of tools used for creating 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, CPU, 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), Performance Monitor (Perfmon), Simple Network Management Protocol (SNMP), SNMP Events, Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (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, and Java application programming interface (API).

Creating installation packages for agents

You can package agents for easy distribution by using a compressed image, which includes install 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. This provides the ability 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.

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, an open source framework for the construction of powerful software development tools and rich desktop applications. Leveraging the Eclipse plug-in framework to integrate technology on the desktop can save technology providers time and money for 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 and provided royalty-free by the Eclipse Foundation. Written in the Java language, Eclipse includes extensive plug-in construction toolkits and examples, and can be extended and run on a range of desktop operating systems including 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 6.2.3.1 of the Tivoli Monitoring Agent Builder, the following enhancements have been made since version 6.2.3:

- You can test data sources in the Agent Builder without the need for a Tivoli Monitoring infrastructure. The data displayed includes the data that was collected and actual values for derived attributes. This feature is available for the following data sources:
  - Windows Management Instrumentation (WMI)
  - Windows Performance Monitor (Perfmon)
  - Simple Network Management Protocol (SNMP)
  - Simple Network Management Protocol (SNMP) event sender
  - Java Management Extensions (JMX)
  - Common Information Model (CIM)
  - Log file
  - Script
  - Java Database Connectivity (JDBC)
  - Internet Control Message Protocol (ICMP) ping
  - Hypertext Transfer Protocol (HTTP) Availability
  - Simple Object Access Protocol (SOAP)
  - Transmission Control Protocol socket (TCP) socket
  - Java Application Programming Interface (API)

  For more information, see “Attribute group testing” on page 373.
- When you add an attribute using the Add data source wizard, you have an option to add additional attributes before you exit the wizard. For more information, see “Creating attributes” on page 54.
- When you create an agent to monitor script output, you can specify separators on a per attribute basis to do the following tasks:
  - Separate one attribute from another with a custom separator that you define
  - Extract a fixed number of bytes from the script output
  - Delimit an attribute value with a string at the beginning and end of the value
  - Return the remainder of a line of text as the attribute value

  For more information, see “Script parsing and separators” on page 225.

- You can now filter the data rows that are returned from attribute groups that return sampled data. You use the formula editor to create and apply filters to determine when to return a data row. For more information, see "Filtering attribute groups" on page 66.
- You can build a new data source by filtering on the data rows of existing attribute groups that return sampled data. You use the formula editor to create and apply filters to the existing attribute group. The result is a new attribute group that contains data rows filtered from the existing attribute group. For more information, see "Filtered Attribute Groups" on page 323.
- For information about how to configure and tune the data collection properties of your agents, see "Configuring and Tuning data collection" on page 418.
- By default, only a single copy of scripts, command files or supplemental files is embedded in the agent. These files in existing Agent Builder projects can also be consolidated. For more information, see "File Separation & Consolidation" on page 223.

For version 6.2.3 of the Tivoli Monitoring Agent Builder, the following enhancements were made since version 6.2.2.7:

- You can use self-describing agents to create agents that bundle support packages with the agent. For more information about self-describing agents, see the *IBM Tivoli Monitoring Installation and Setup Guide* and the *IBM Tivoli Monitoring Administrator's Guide*.
- You can use Agent Builder to generate a Cognos data model to create Tivoli Common Reporting reports for your agent.
- If you monitor log files, a new attribute group, Log File RegEx Statistics, has been added which contains information showing the statistics of the log file regular expression processing.

About this guide

This guide describes how to use the Tivoli Monitoring Agent Builder software to create, modify, debug, and package agents for use with Tivoli Monitoring products. This book explains concepts that agent developers, system administrators, and application administrators must know to use the Tivoli Monitoring Agent Builder product and its integration into the Tivoli environment.

Chapter 2. Getting started quick-reference guide

This chapter provides quick-reference tables of the procedures you can perform with Tivoli Monitoring Agent Builder. The two main objectives for using the Agent Builder (creating agents and creating application support extensions by creating additional workspaces and situations to enhance one or more existing agents) are listed in the information later in this section. The following tables also include cross-references to the locations of additional 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 7 • "Authorization" on page 8 • "Builder installation procedure" on page 8 • "Planning and overview" on page 13 • "Defining initial data source" on page 25 • Chapter 28, "Creating a navigator group," on page 327 • Chapter 29, "Creating subnodes," on page 331
2. Learn how to create an agent using the Agent Builder wizard.	• Chapter 4, "Starting the Agent Builder," on page 13 • Chapter 5, "Creating a basic agent," on page 15
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 373 • Appendix B, "Command-line options," on page 523 • Chapter 35, "Troubleshooting," on page 413 • "Preparing your agent for modifications or for regeneration" on page 35 • Chapter 6, "Modifying your agent by using the Tivoli Monitoring Agent Editor," on page 37
4. Learn how to remove an agent that you created with the Agent Builder.	• "Uninstalling an agent" on page 389
5. Learn how to create workspaces and situations for your agent.	• Chapter 32, "Creating workspaces, Take Action commands, and situations," on page 391 • Chapter 33, "Importing application support files," on page 397
6. Learn how to create reports for your agent.	• Appendix E, "Cognos data model generation," on page 597

The following table contains the procedures for creating other functions:

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 391
2. Learn how to package your application support extension.	• Appendix D, "Creating application support extensions for existing agents," on page 585
3. Learn how to build custom bundles.	• Appendix H, "Non-agent bundles," on page 621

Chapter 3. Installing the Tivoli Monitoring Agent Builder

Table 3 shows some guidelines for installation and running the Tivoli Monitoring Agent Builder.

Note: For information about installing or modifying an *agent*, see "Installing and testing an agent" on page 377.

Table 3. Guidelines for installation and configuration

Goal	Refer to
Verify that your environment meets the requirements.	"Requirements" on page 7
Ensure that you have appropriate authorization.	"Authorization" on page 8

Requirements

To install the Agent Builder, you must have:

- A computer with approximately 400 MB of space, plus the space required for the agents you develop.
- One of the following operating systems installed and running:
  - AIX® 5.3 ML5 x86 (32-bit) or later
  - AIX 6.1
  - AIX 7.1
  - Red Hat Enterprise Linux 4.0 + U2 x86 (32-bit) or later
  - Red Hat Desktop Linux 4.0 + U2 x86 (32-bit) or later
  - Red Hat Enterprise Linux 5.0 x86 (32-bit)
  - Red Hat Enterprise Linux 5.0 x86-64 (64-bit)
  - Red Hat Enterprise Linux 6.0 x86 (32-bit)
  - SUSE Linux Enterprise Server 9 Sp1 x86 (32-bit) or later
  - SUSE Linux Enterprise Server 10 x86 (32-bit)
  - SUSE Linux Enterprise Server 10 x86-64 (64-bit)
  - SUSE Linux Enterprise Server 11 x86 (32-bit)
  - SUSE Linux Enterprise Server 11 x86-64 (64-bit)
  - Windows Server 2000
  - Windows 2000 Advanced Server
  - Windows Server 2003 EE x86 (32-bit)
  - Windows Server 2003 SE x86 (32-bit)
  - Windows Server 2003 Data Center x86 (32-bit)
  - Windows Server 2003 EE x86-64 (64-bit)
  - Windows Server 2003 SE x86-64 (64-bit)
  - Windows Server 2003 Data Center x86-64 (64-bit)
  - Windows XP Professional
  - Windows Server 2008 Data Center x86 (32-bit)
  - Windows Server 2008 EE x86 (32-bit)

– Windows Server 2008 SE x86 (32-bit)
– Windows Server 2008 Data Center x86-64 (64-bit)
– Windows Server 2008 EE x86-64 (64-bit)
– Windows Server 2008 SE x86-64 (64-bit)
– Windows Server 2008 R2 Data Center x86-64 (64-bit)
– Windows Server 2008 R2 EE x86-64 (64-bit)
– Windows Server 2008 R2 SE x86-64 (64-bit)
– Windows 7 x86 (32-bit)
– Windows 7 x86-64 (64-bit)

See "Operating system requirements" on page 15 for information about the operating systems supported by the agents created by the Agent Builder.

Authorization

Before you can run Tivoli Monitoring Agent Builder, you must have Administrator authorization for the following reasons:

- Ability to ensure a consistent running environment for the Agent Builder and the agents 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

Before installing the Tivoli Monitoring Agent Builder, uninstall any previous versions. For more information about uninstalling, see "Uninstalling Agent Builder" on page 10. None of your existing agent information is lost when you uninstall.

To install the Tivoli Monitoring Agent Builder:

1. Download and unzip the Tivoli Monitoring Agent Builder V6.2.3 Fix Pack 1 image.
2. Run the executable for your operating system to start the installation:
   - setup.bat
   - 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 appears 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 you choose for the Agent Builder should not contain the following characters, because having these characters in the directory path prevents the agent builder from launching:

!
#
%
;

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 be displayed. And then, click **Done**.

After installing

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

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

The log files for the Agent Builder installation are in the following locations:
install_dir\IBM_Tivoli_Monitoring_Agent_Builder_InstallLog.xml (Windows)
install_dir/IBM_Tivoli_Monitoring_Agent_Builder_InstallLog.xml (Linux, AIX)

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

Silent installation

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

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 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

Use the following steps to uninstall any earlier versions of Agent Builder:

On AIX and Linux systems

Run the following command:

INSTALL_DIR/uninstall/uninstaller

where *INSTALL_DIR* is the name of the directory where Agent Builder is installed.

On Windows systems

Perform the following steps:

1. From the 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**

Perform the following steps:

1. Open Programs and Features by clicking the **Start** button, clicking **Control Panel**, clicking **Programs**, and then clicking **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: You can also navigate to the Windows Programs and Features window by clicking the **Start** button, clicking **Computer** and clicking **Uninstall or change a program**. Then continue from step 2.

Silent uninstallation

You can also use the silent uninstallation method. Start the silent uninstallation by running the following command:

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

Chapter 4. Starting the Agent Builder

This chapter helps you use the Agent Builder to start the wizards.

Planning and overview

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 20)
- The types of monitoring data that you want your agent to collect (see "Defining initial data source" on page 25)

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 (availability, log events, or external data sources) that you want the agent to collect, the wizard guides you through the process of configuring each of the data collection methods.
4. Testing your agent (see Chapter 31, "Testing and debugging your agent," on page 373)
5. Optionally add workspaces, situations and queries (see Chapter 33, "Importing application support files," on page 397)

Note: The Agent Builder does not build language packs, so the text that is displayed in the Tivoli Enterprise Portal is in the language you use when building the agent.

Starting the Agent Builder

Start (launch) the Agent Builder by typing the following information on the command line:

Windows: *Install_Location*\agentbuilder.exe. Or, go to either **Start > All Programs > IBM Tivoli Monitoring > Agent Builder** or click the Agent Builder desktop icon.

All other supported operating systems: *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.

Chapter 5. Creating a basic agent

This chapter helps you use the Agent Builder to create a basic agent. The following topics are covered:

- "Operating system requirements"
- "Starting the new agent wizard"
- "Creating and defining the agent" on page 20
  - "Naming your agent" on page 20
  - "Defining initial data source" on page 25
  - "Organizing the agent" on page 30
- "Preparing your agent for modifications or for regeneration" on page 35

Operating system requirements

The agents created by the Agent Builder are supported on all of the operating systems supported by the OS monitoring agents, except z/OS® and i5/OS®, as follows:

- AIX
- HP-UX
- Linux
- Linux on System z®
- Solaris
- Windows

For a more detailed list that includes version numbers, refer to 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 will run. Install Tivoli Monitoring V6.2 or later in your environment. Tivoli Monitoring does not have 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 is running 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 is running on the same level as the operating system and monitored applications for which you are developing the agent

Starting the new agent wizard

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.
- Use the **File** option on the top menu bar.

Note: You might not see the wizard from the new menu if you are not using the IBM Tivoli Monitoring perspective in Eclipse. The IBM Tivoli Monitoring perspective opens when you end the wizard, along with the Project Explorer view, the Outline view, and the Problems view.

Using the icon to start the wizard

Click the **Create New Agent** icon (which resembles a pencil) on the top menu (Figure 1).

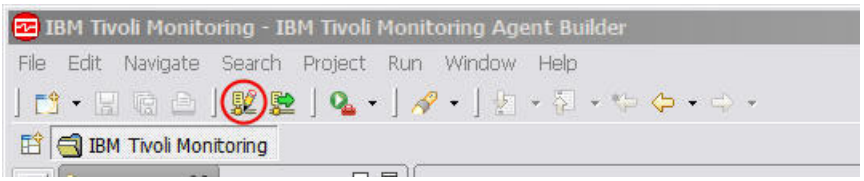


Figure 1. Create New Agent icon

The Welcome window (Figure 6 on page 20) for creating an agent is displayed.

Creating a new file to start the wizard

Select one of the following ways to create a new file:

- **File > New > IBM Tivoli Monitoring Agent**
- **File > New > Other**

Using File > New > IBM Tivoli Monitoring Agent to start the wizard

For the first of these methods, select **File > New > IBM Tivoli Monitoring Agent** (Figure 2). The Welcome window is displayed.

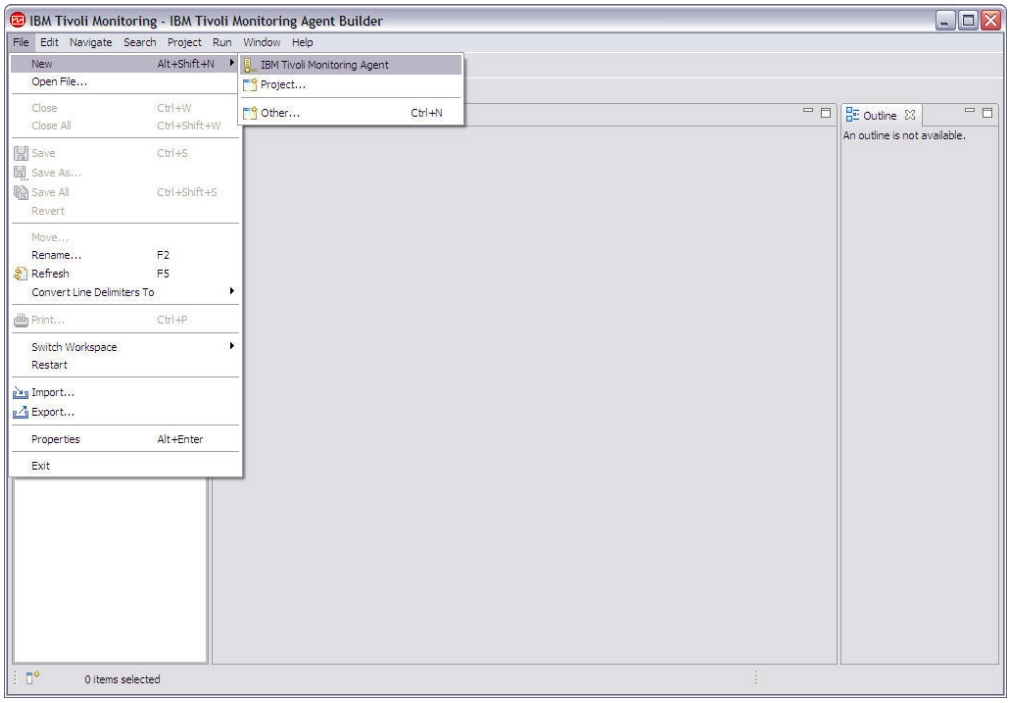


*Figure 2. Starting the wizard with the File > New > **IBM Tivoli Monitoring Agent** option*

Using File > New > Other to start the wizard

For the second of these methods, use the following steps:

1. From the Main Menu, select **File** > **New** > **Other** (Figure 3).

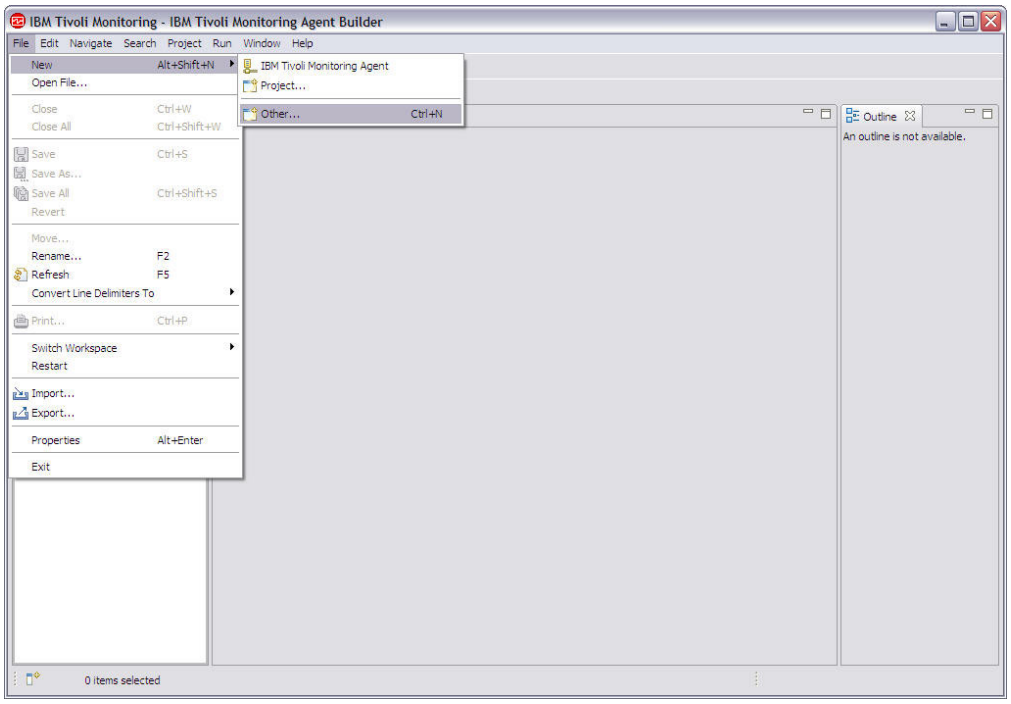


Figure 3. Starting the wizard with the File > New > Other option

2. When the Select a Wizard page is displayed (Figure 4 on page 18), double-click the **IBM Tivoli Monitoring Wizards** folder of wizard options.

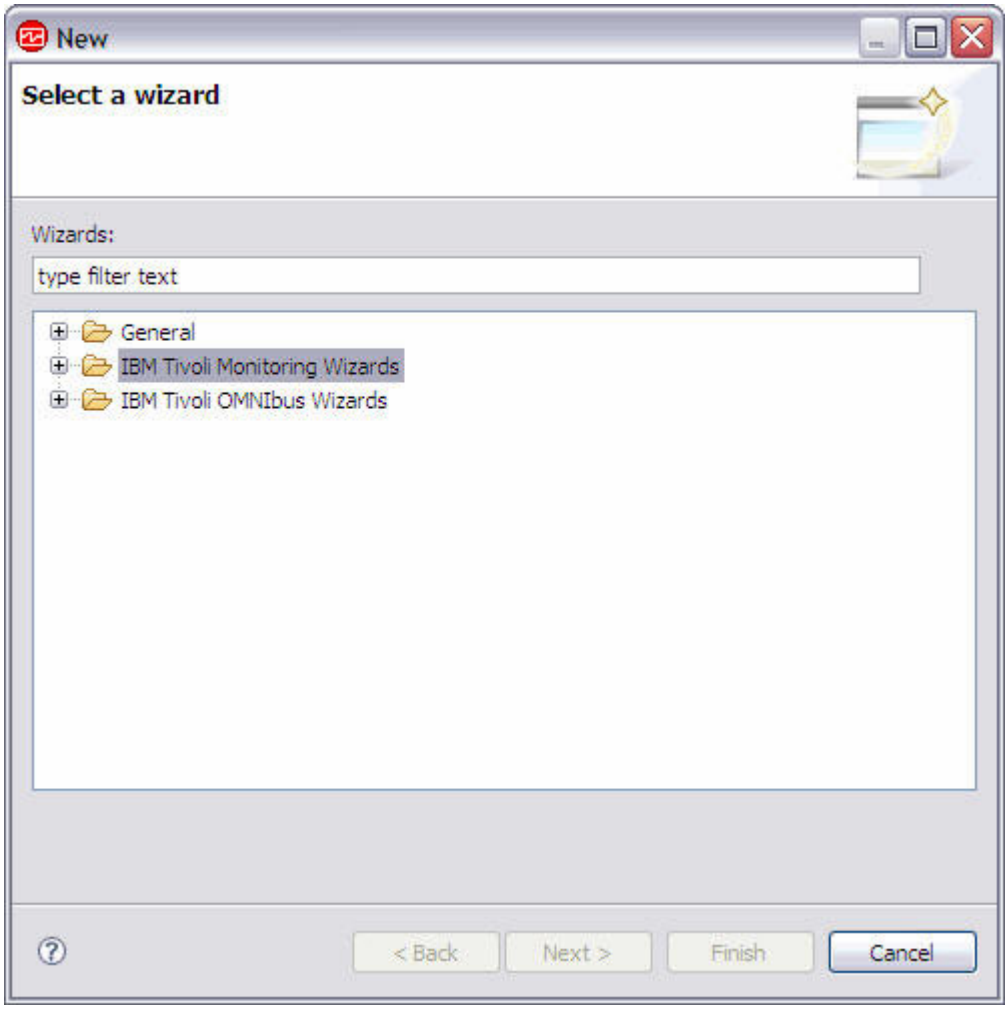


Figure 4. Select a wizard page

3. When the IBM Tivoli Monitoring Agent icon is displayed (Figure 5 on page 19), double-click it.

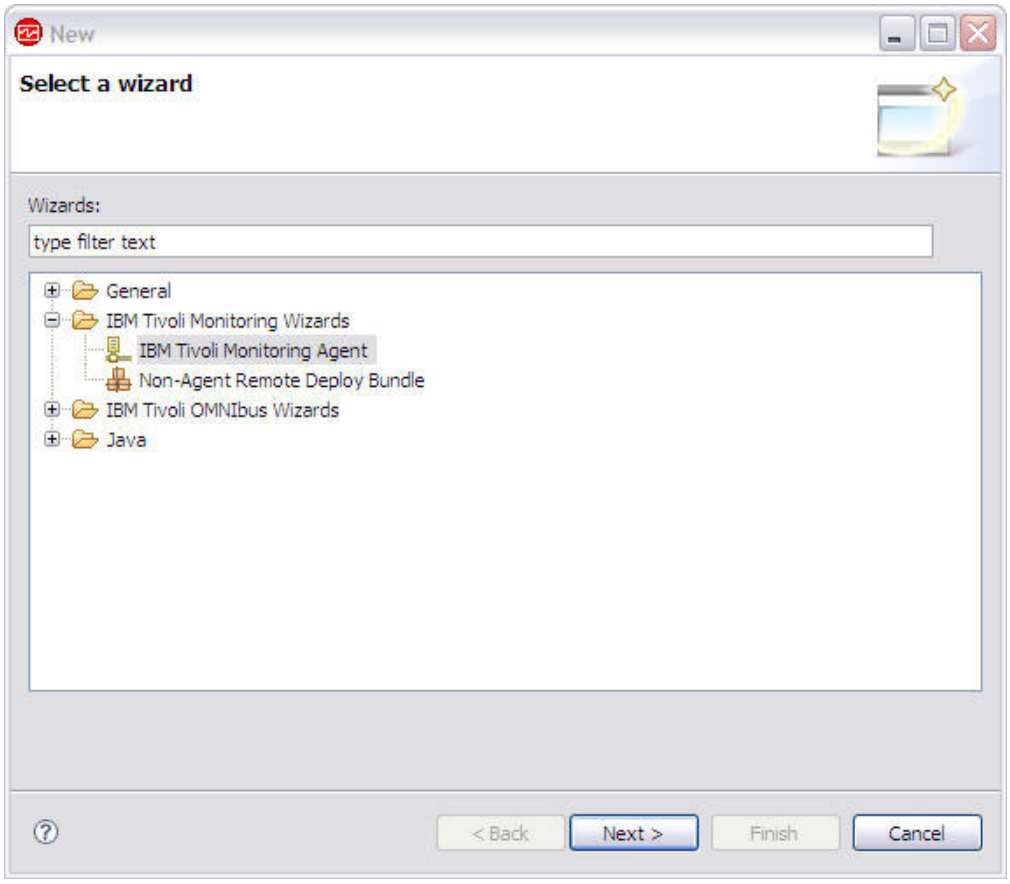


Figure 5. Selecting the agent icon

The Welcome window (Figure 6 on page 20) for creating an agent is displayed.

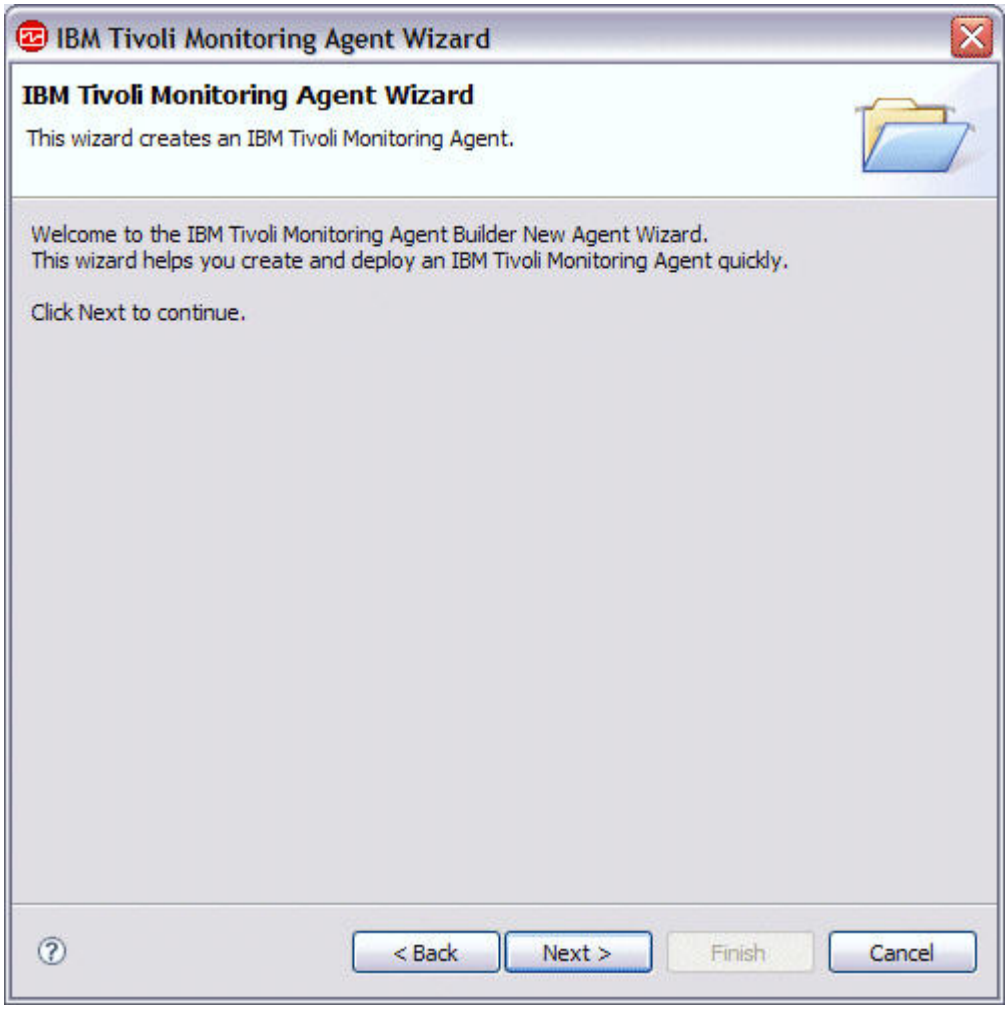


Figure 6. Welcome window

Creating and defining the agent

After selecting a method to start the wizard (see "Starting the new agent wizard" on page 15), 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

The agent name that you designate is displayed in the Tivoli Enterprise Portal when you are ready to run the agent.

Use the following steps to name your agent:

1. After starting the wizard by any of the methods, click **Next** in the Welcome window (Figure 6) to display the New IBM Tivoli Monitoring Agent Project

page (Figure 7), where 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.

Notes:

a. The **Location** field affords an opportunity to navigate to 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 and click **Browse** to navigate 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. See the Eclipse help for more information.

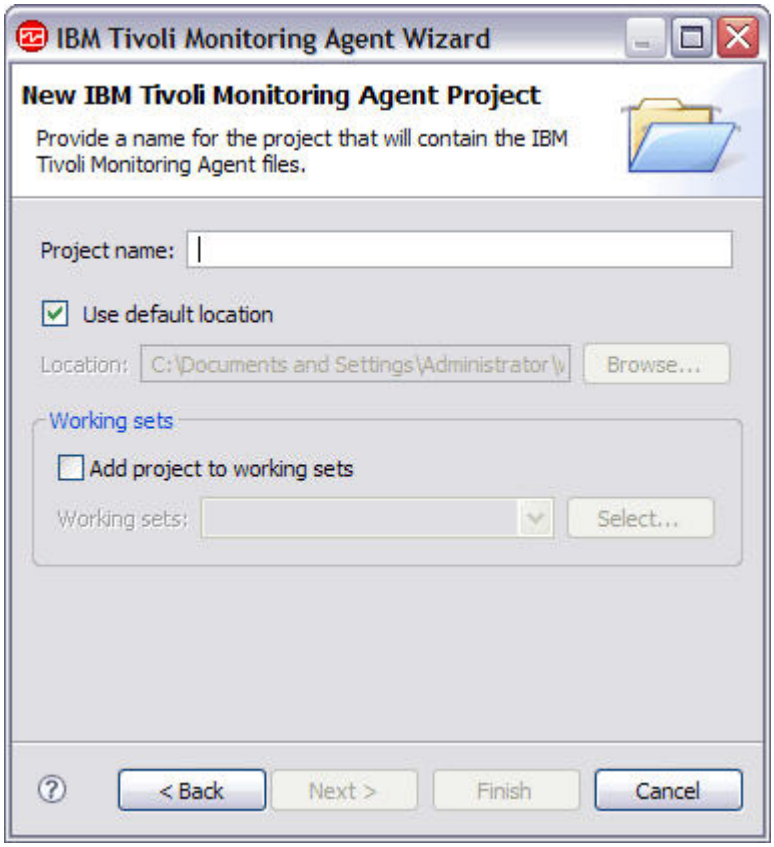


Figure 7. Project page

2. After you type the project name, click **Next** to see the General Information page (Figure 8 on page 22).

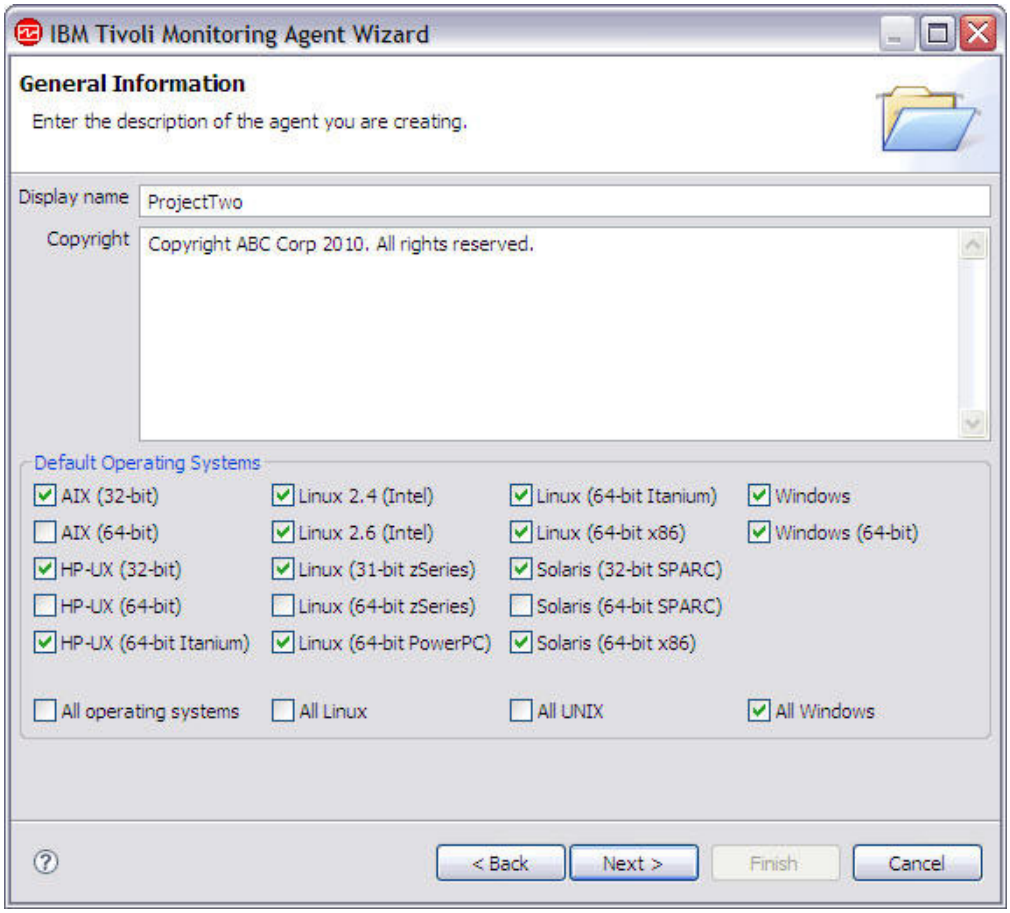


Figure 8. General Information page

The project name is displayed in the **Display name** field. This name is displayed in the Tivoli Enterprise Portal.

3. Type the copyright statement that you want to use for your new agents in the **Copyright** field (Figure 9 on page 23). 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 once, the Agent Builder remembers your copyright statement and displays it in the copyright area each time that you start the wizard to create a new 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: An agent can be deployed on a 64-bit operating system as a 32-bit process if the 32-bit version of that operating system is selected and the 64-bit version of that operating system is not selected.
5. Click **Next** to display the Agent Information page (Figure 9 on page 23).

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 created on Windows and 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 permitted 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 3-letter product code, you will be 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 developing the agent ("IBM" is reserved.). A good candidate for this field can come from your company's URL. 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 being developed (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 3-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 when you need to release a fix for an agent, but you do not want to update 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 so that the Tivoli Monitoring user knows 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 at the same time. Each of these remote systems can be represented separately in Tivoli Monitoring and each can have different configuration values for communication. To do this, select the **Support multiple instances of this agent** check box, which 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. Table 4 provides the reasons to use each level.

Table 4. 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 23. If you select 6.2.1, you can "Allow any configuration property to be overridden in any subnode", (See **Allow any configuration property to be overridden in any subnode** in “Advanced subnode configuration” on page 347.) • This version has 64-bit numeric attribute support. • This version includes support for different default configuration values for Windows and for UNIX and Linux. See Default Value for more information. The default minimum Tivoli Monitoring version is 6.2.1.

Table 4. Minimum version of IBM Tivoli Monitoring required by the agent (continued)

IBM Tivoli Monitoring level	Reason to use this level
V6.2	By selecting 6.2, the resulting agent will run on Tivoli Monitoring, version 6.2 or later, but other features listed above including 64-bit numeric attribute support will not be available to the agent, and Subnode Configuration Overrides must be defined when the agent is built (See "Configuring a subnode" on page 345).

Defining initial data source

After entering the Agent Information, click **Next** on the Agent Information page (Figure 9 on page 23) to display the Agent Initial Data Source page (Figure 10) so you can define the first monitor.

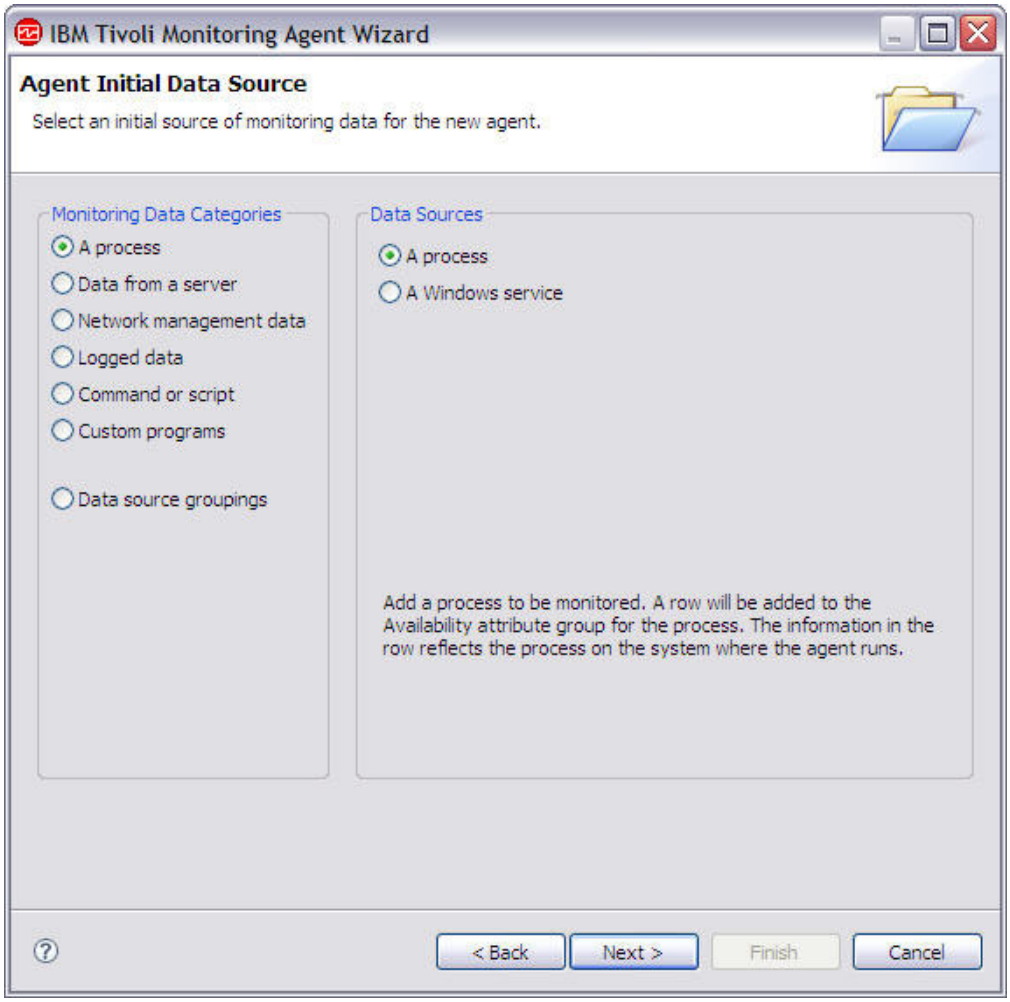


Figure 10. Agent Initial Data Source page

After you select one of the **Monitoring Data Categories** and one of the **Data Sources**, and click **Next**, the wizard guides you through the process of defining and configuring any of the data collection types that you specify. You begin with the Agent Initial Data Source page to define data monitoring types by specifying

the monitoring data categories and the data sources that your new agent is to monitor. For detailed information about each data source, see "Data Sources."

After you define the initial monitor, you can define additional monitors from the Data Source Definition page, the last page of the New Agent Wizard. See "Defining additional data sources" on page 28.

If you decide to add one or more of the monitoring types after you finish the wizard, use the following information to add them using the Agent Editor: Chapter 6, "Modifying your agent by using the Tivoli Monitoring Agent Editor," on page 37.

If you have created a new data source that might return more than one data row, you are prompted to select key attributes, see "Selecting key attributes" on page 27. When you finish defining all of your monitors, click **Finish**, and then generate and install your new agent. See "Installing and testing an agent" on page 377.

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
  - Ping
- 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
  - "Filtered Attribute Groups" on page 323
- Data source groupings monitoring category

- Subnode Definition
- Navigator group

Selecting key attributes

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 needs to a match 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, and do not change from one sample to the next when referring to the same monitored entity.

Rate and delta attributes are calculated by comparing the current sample to the previous sample, and 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.

After you have specified the details about your new data source (using the information in “Defining initial data source” on page 25), if it is possible for multiple data rows to be returned by the data source, the Agent Builder prompts you to select key attributes on the Select key attributes page (Figure 11 on page 28).

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

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

- Click **Produces a single data row** if this attribute group can return only one row. If this option is selected, no key attributes are necessary because only one monitored entity is ever reported in this attribute group.
- 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.

Defining additional data sources

After you define the initial monitor, you can define additional monitors from the Data Source Definition page, by selecting the agent, navigator group, or subnode, and then clicking **Add to Selected**, or right-clicking on the agent, and selecting **Add Data Source** as shown in (Figure 12 on page 29).

Note: You can also add additional monitors after you complete the New Agent Wizard. See (Chapter 6, "Modifying your agent by using the Tivoli Monitoring Agent Editor," on page 37), for information about adding monitors using the Agent Editor. An advantage of using the Agent Editor is that you can save incremental changes as you make them.

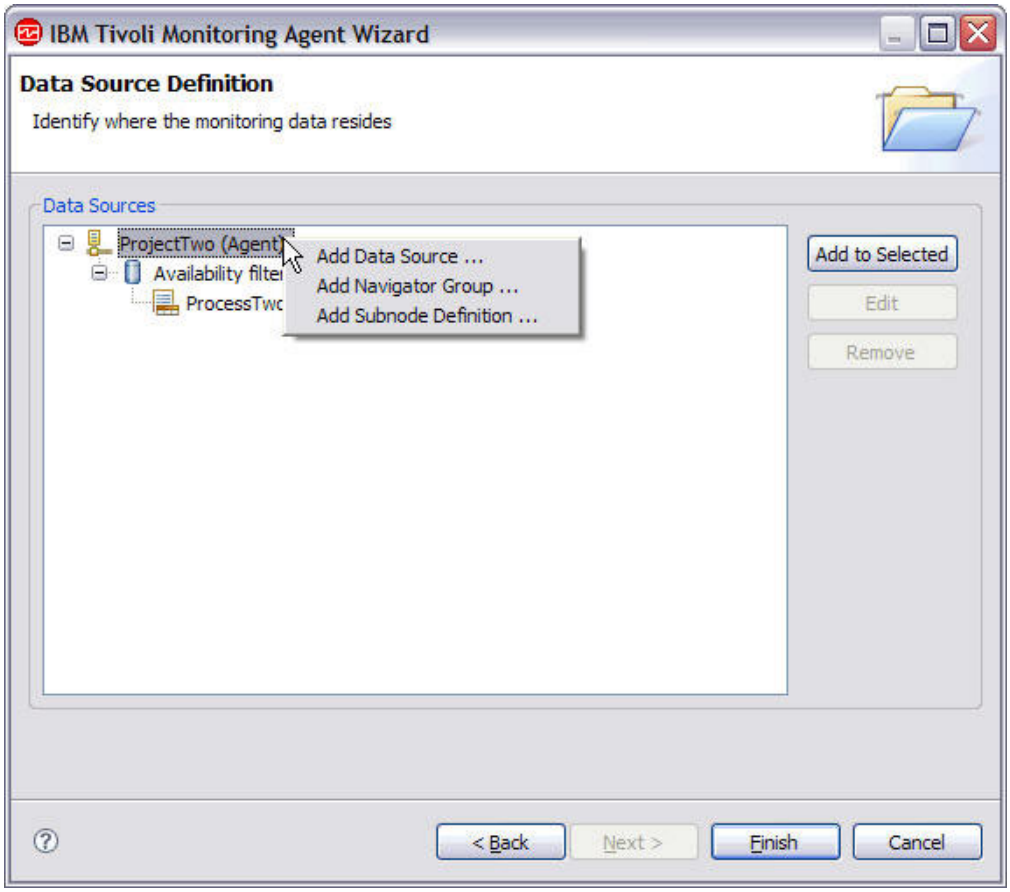


Figure 12. Data Source Definition page: defining additional monitors

When you add a monitor using the Agent Editor Data Source Definition page, the Data Source Location page is displayed (Figure 13 on page 30). This page is the same as the Agent Initial Data Source page (Figure 10 on page 25) except for the name of the page.

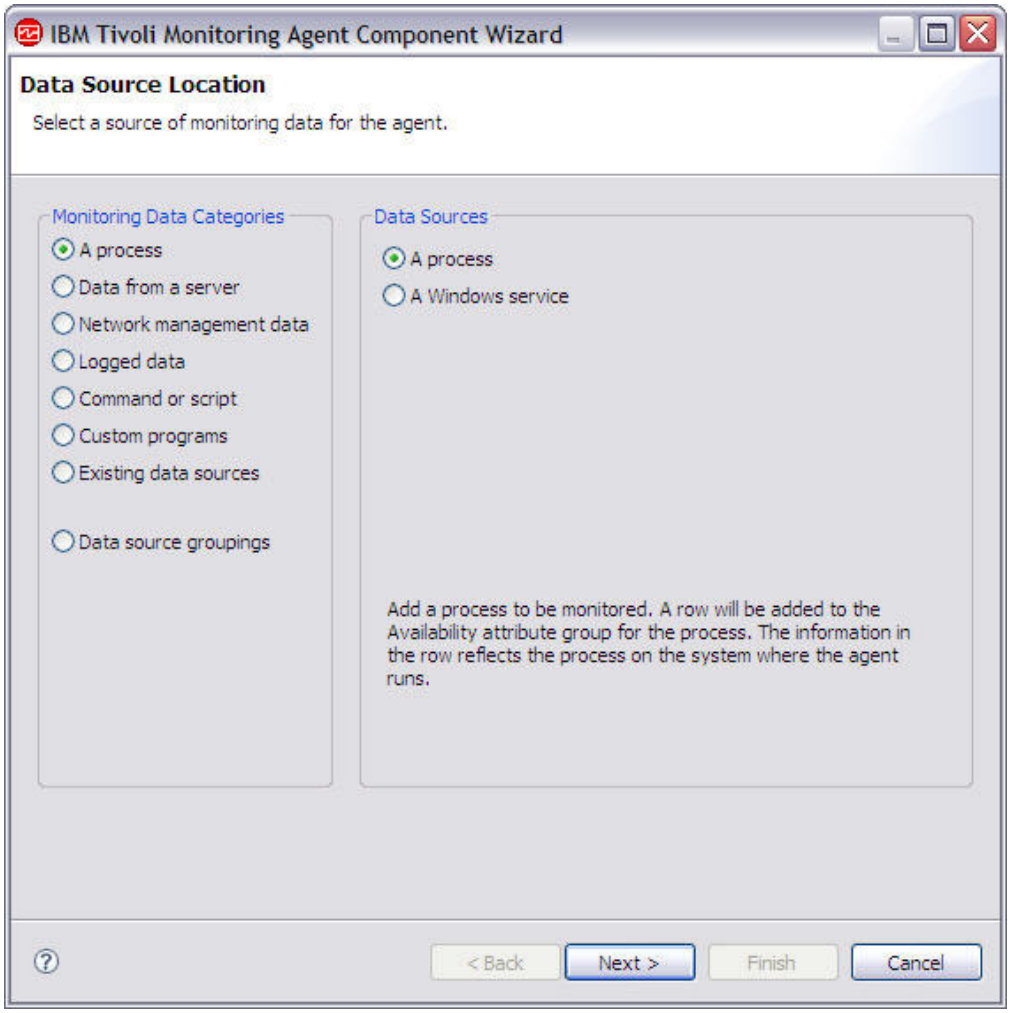


Figure 13. Data Source Location page

After you select one of the **Monitoring Data Categories** and one of the **Data Sources**, and click **Next**, the wizard guides you through the process of defining and configuring any of the data collection types that you specify. You begin with the Agent Initial Data Source page to define data monitoring types by specifying the monitoring data categories and the data sources that your new agent is to monitor. For detailed information about each data source, see "Data Sources" on page 26.

Click **Finish** to add the new data source to the Data Source Definition page.

Organizing 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 the data 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 constructed from all of the data sources contained in the navigator group. In addition to the information in this section (“Navigator groups”), see Chapter 28, “Creating a navigator group,” on page 327 for additional information about navigator groups.

A subnode is a portion of an agent that can be replicated as necessary, depending on the application or enterprise being monitored. In addition to the information in this section (“Subnodes”), see Chapter 29, “Creating subnodes,” on page 331 for additional information about subnodes.

Navigator groups

Navigator groups are generally used 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 certain metrics collected from 2 data sources are most useful if the data sources are joined to create a combined, third data source. You want to see only the data combined in the Joined datasource. 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 317 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

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

- Monitors each instance of an application running on a system instead of having to use separate instances of the agent, one for each 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 allowing additional grouping and customization.
- Defines additional 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, but have 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, at least slightly, 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 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 the agents built using the Agent Builder create the subnode instances based on configuration values, these subnodes have the same life span as the agent. There is still just one heartbeat performed for the agent, not a separate heartbeat for each subnode. Defining monitoring for a number of systems or application instances using subnodes instead of agent instances can significantly increase the possible 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, so that you can manage portions of your environment separately.

Along with attribute groups in subnodes, an agent can define agent-level attribute groups that reside 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 33, "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 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 3 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.

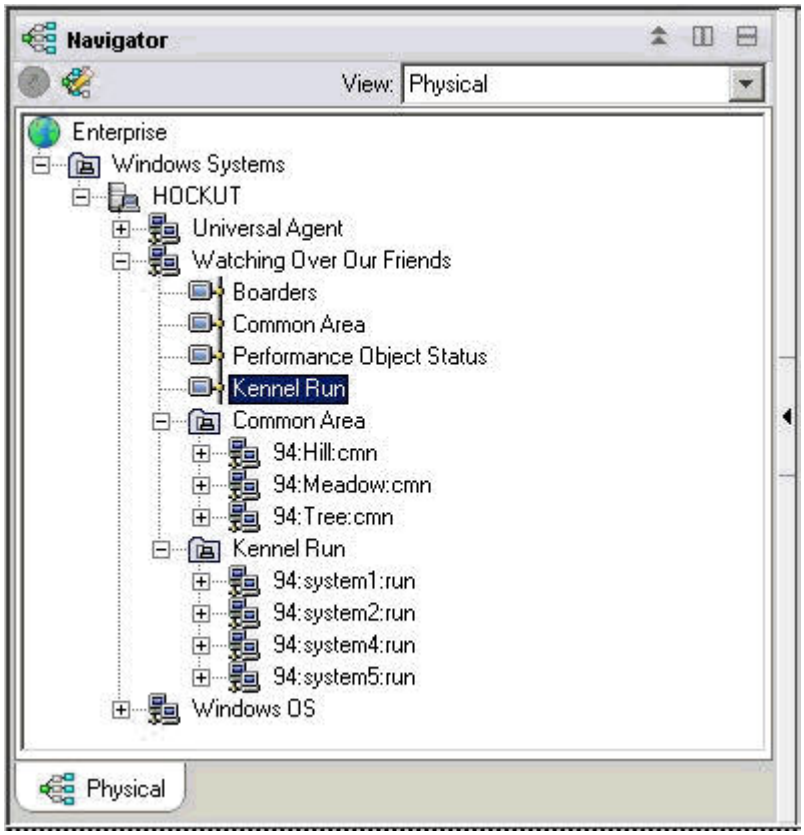


Figure 14. Subnodes in the Navigator tree

Moving existing data sources

If you create a navigator group or a subnode after you have created some attribute groups, and you want some of the existing attribute groups to be moved into the newly-created navigator group or subnode, use the following procedure:

1. Begin creating a navigator group or subnode as described in Chapter 28, "Creating a navigator group," on page 327 or Chapter 29, "Creating subnodes," on page 331.
   - On the Data Source page, click **Existing data sources** in the **Monitoring Data Categories** area (Figure 15 on page 34).

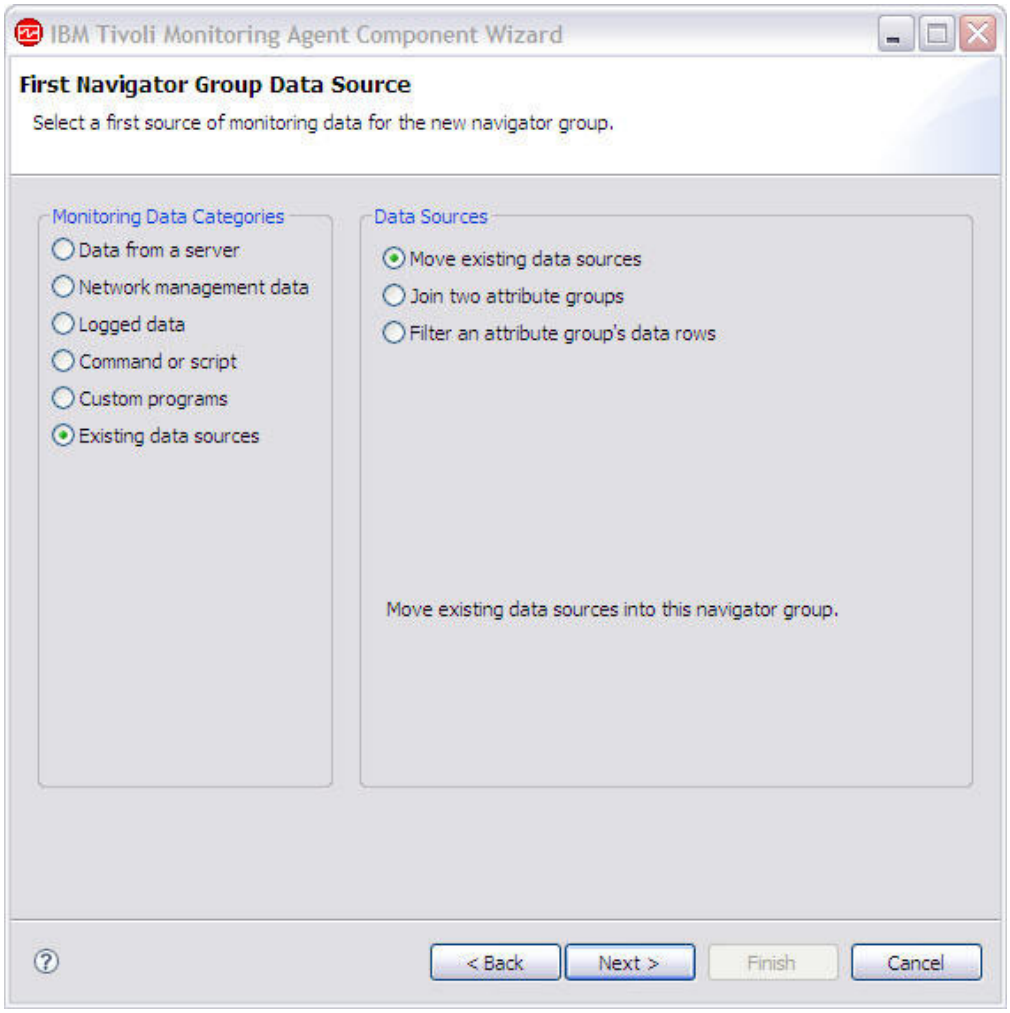


Figure 15. Move existing data sources

- In the **Data Sources** area, click **Move existing data sources** in the Data Sources area.
- Click **Next**.

2. On the Currently Defined Data Sources page, select one or more data sources (Figure 16 on page 35).

Figure 16. Currently Defined Data Sources page

3. Click **Finish**.

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

Note: Once a subnode or navigator group has been created, you can move attribute groups into it from the Data Sources tab of the Agent Editor, 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

To modify an existing agent definition, double-click the `itm_toolkit_agent.xml` file to launch the IBM Tivoli Monitoring Agent Editor (see Chapter 6, "Modifying your agent by using the Tivoli Monitoring Agent Editor," on page 37).

If you installed your agent prior to making modifications, you can uninstall the existing agent before installing the new one, or 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 389), 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 373).

If you have changed the version number of your agent, when you install the new version, files in your Tivoli Monitoring installation that contain the old version number are overwritten, except for the file containing your agent's configuration requirements (named according to the format: *product_code*_dd_*long_version*.xml). For example, for an agent with the product code of 19 and the version of 6.2.3, the

file is called `19_dd_062300000.xml` along with the remote deployment bundle for your agent. Both of these are added to the system alongside the old versions rather than replacing them.

Chapter 6. Modifying your agent by using the Tivoli Monitoring Agent Editor

This chapter covers the following topics

- "Tivoli Monitoring Agent Editor"
- "Saving your edits and changes" on page 49
- "Committing a version of the agent" on page 49

Tivoli Monitoring Agent Editor

This section describes how to open the Tivoli Monitoring Agent Editor and how to use the Editor pages to change your created agent.

Before you modify an agent, see "Preparing your agent for modifications or for regeneration" on page 35.

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 on a node in the Outline view. If the Outline view is missing, or hidden behind another view (as it was in previous versions of Agent Builder), you can reset the Tivoli Monitoring perspective by selecting **Window** > **Reset Perspective**, or by right-clicking on the Tivoli Monitoring perspective button and choosing **Reset** from the context menu.

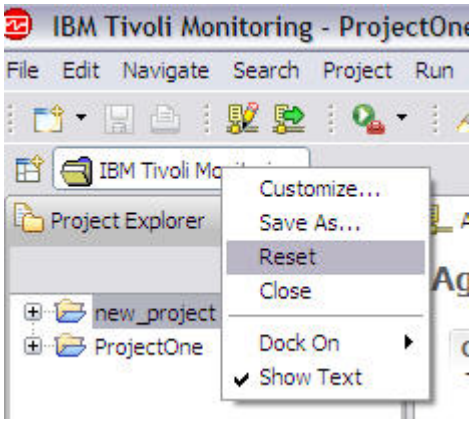


Figure 17. Choosing Reset from the Tivoli Monitoring perspective button

Note: For detailed information and procedures for creating an agent, see Chapter 5, "Creating a basic agent," on page 15.

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 viewing an Editor page, you can also switch to another page by clicking the tab for the page near the bottom of the window. 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 by clicking on the pin icon in the upper right corner so that the pin in the icon points toward the page.

Agent Information page

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

- 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

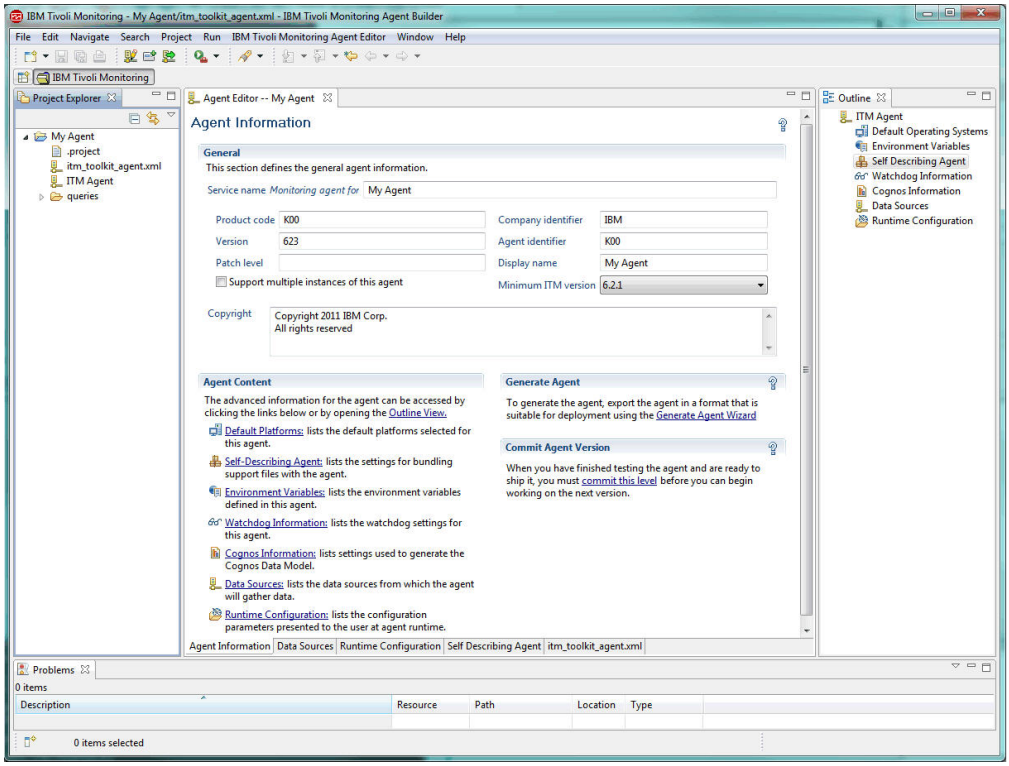


Figure 18. Agent Information page

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 will be built. To open the **Default Operating Systems** page (Figure 19), click **Default Operating Systems** in the **Agent Content** section of the Agent Information page or the **Default Operating Systems** node in the Outline View. When you generate your agent, files unique to each of the operating systems you select here are added to your agent. Unless otherwise specified, data sources you add to your agent that are not specific to the Windows operating system will be available on any of the operating systems that are checked here. The list of operating systems on which any single data source is available can be changed from this list using the details page for that data source. If no operating systems are selected in this list, then operating systems must be selected for each individual data source on the details page for that data source.

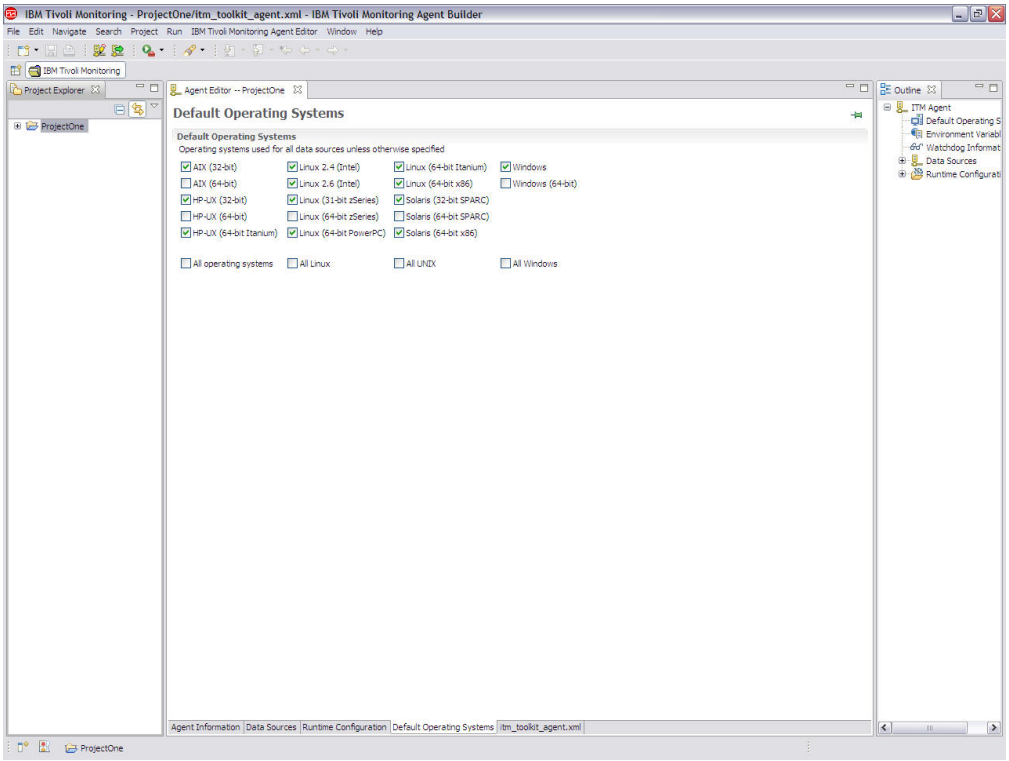


Figure 19. Default Operating Systems page

Self-Describing Agent

Use the Self-Describing Agent page Figure 20 on page 40 to specify whether the agent's support files are bundled with the agent. 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 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 so that the agent can automatically 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.

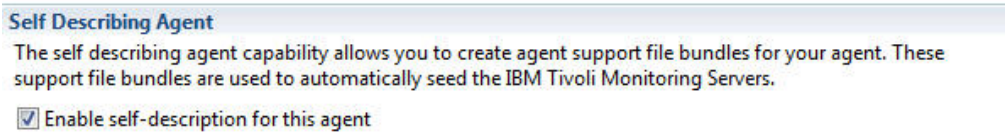


Figure 20. Self-Describing Agent page

Environment variables

Use the Environment Variables page (Figure 21) to view and modify environment variables that are available to your agent while it is running. To open the Environment Variables page, click **Environment Variables** in the **Agent Content** section of the Agent Information page or the **Environment Variables** node in the Outline View.

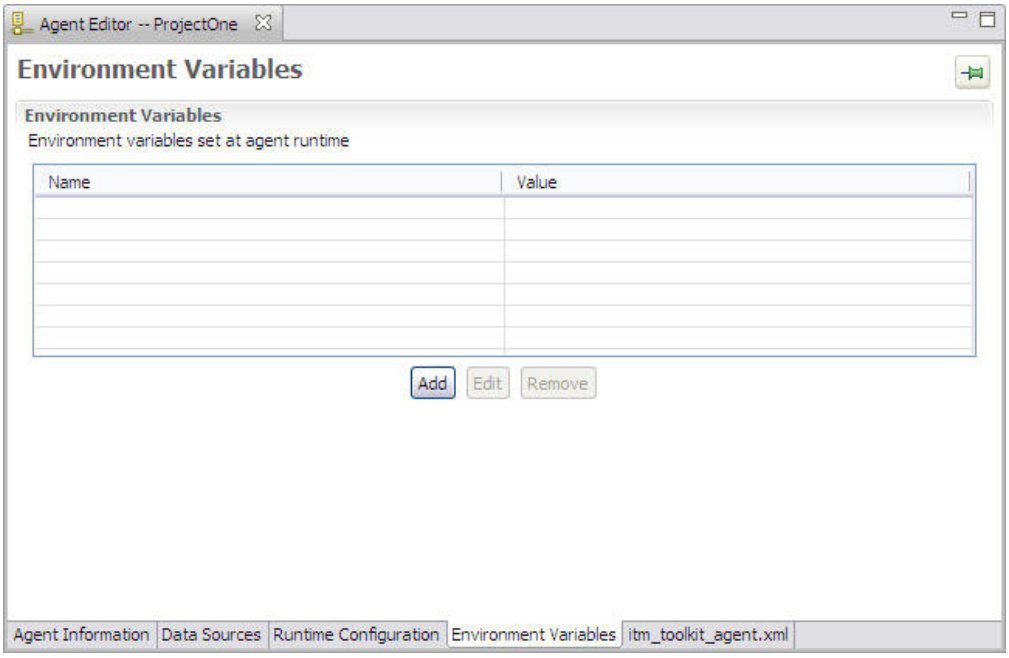


Figure 21. Environment Variables page

The environment variables can be ones that you define, for access inside a script, or predefined variables that cause the agent to behave in a certain way. See Table 45 on page 424 for a list of predefined variables.

Use the following procedure to add an environment variable:

1. Click **Environment Variables**, in the **Agent Content** section of the Agent Information page.
2. In the Environment Variables page (Figure 21 on page 40), click **Add**.
3. In the Environment Variable Information window (Figure 22), complete the information as follows:

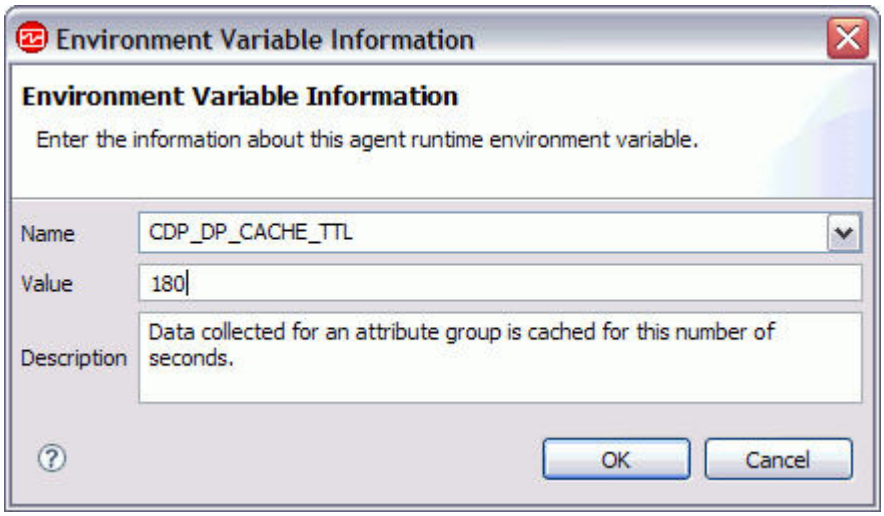


Figure 22. Environment Variable Information window

   a. In the **Name** field, type a variable name or select a predefined name from the drop-down 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

To open the Watchdog Information page (Figure 23 on page 42), click **Watchdog Information** in the **Agent Content** section of the Agent Information page or the **Watchdog Information** node in the Outline View.

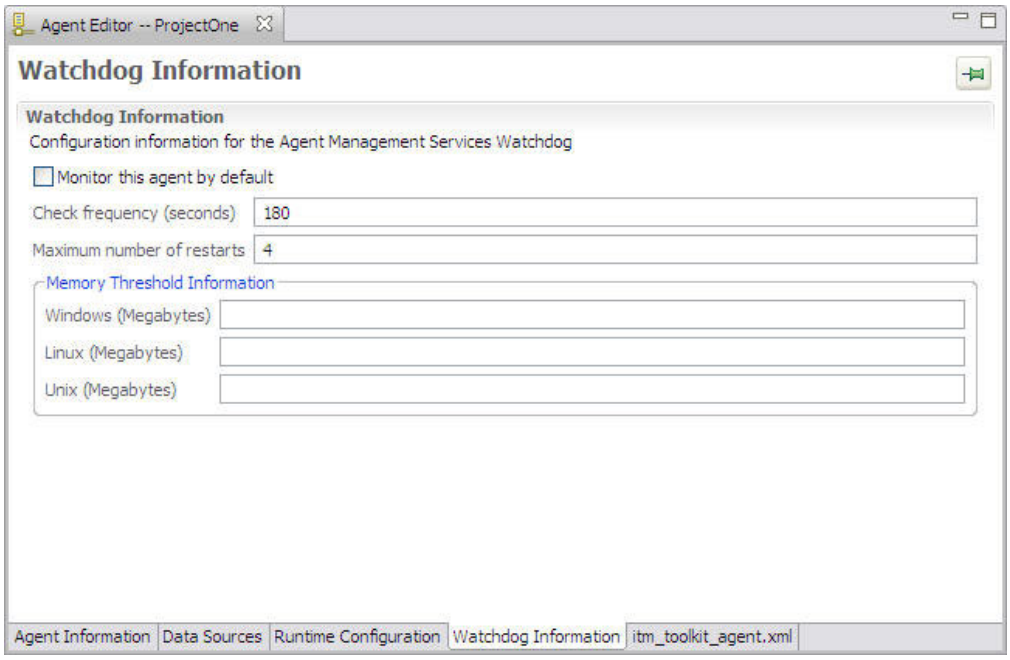


Figure 23. Watchdog Information page

Use the Watchdog Information page to specify 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 watched for unhealthy behavior or abnormal termination and be 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**

  How many times the Watchdog will restart the agent process because of unhealthy behavior or abnormal termination in a 24-hour period before alerting 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 has to 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 4 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 will stop the process and restart it. There are no defaults for these properties. If no value is specified, the Watchdog will not monitor the process size. This 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 has been reached, Watchdog sends an alert that the agent has exceeded its restart count, and

stops doing auto-restarts. Watchdog still reports whether the agent is up or down assuming it is started in some sideband manner such as through the Tivoli Enterprise Portal or Manage Tivoli Enterprise Monitoring Services.

You must manually restart the agent 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 additional 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 (Figure 24 on page 44) to specify the information used when a Cognos data model is generated for your agent. You can use this data model to create Tivoli Common Reporting reports for your agent, see Appendix E, "Cognos data model generation," on page 597. 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.

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. In the **Schema** field, enter the name of the database schema 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.

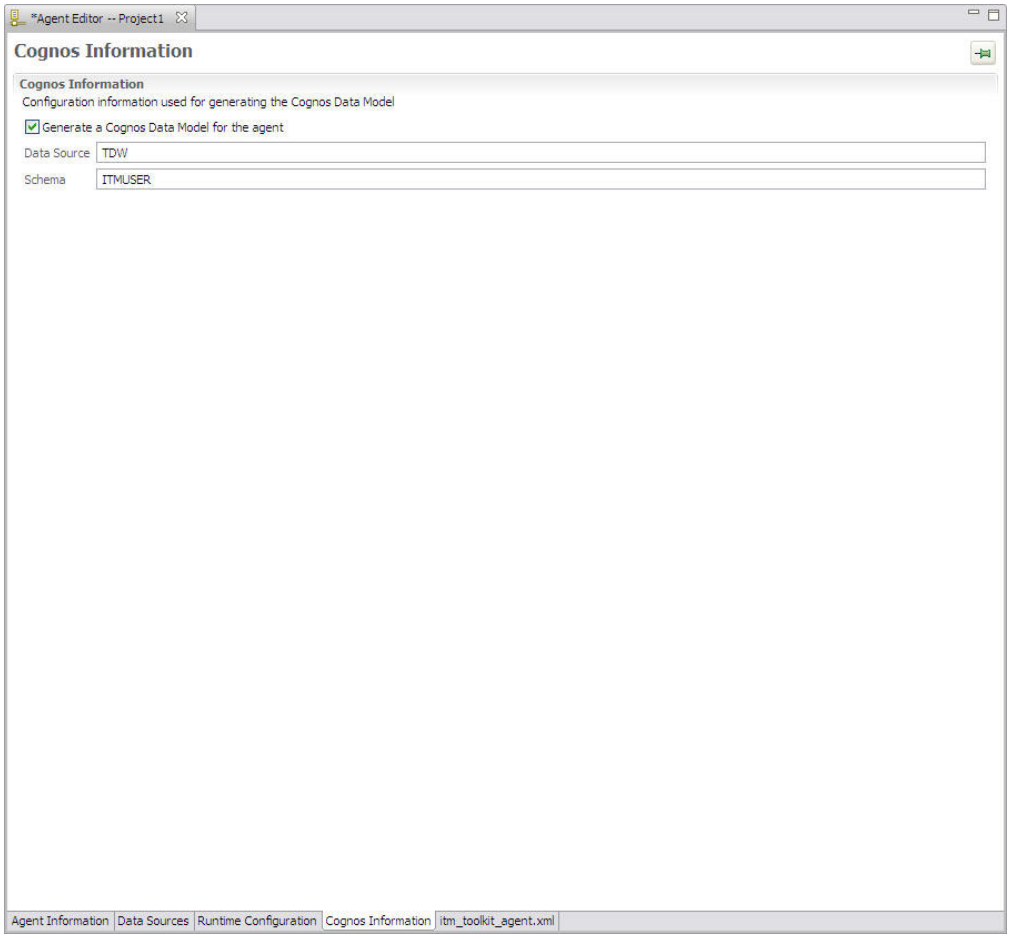


Figure 24. Cognos Information page

The **Add this attribute group to a reporting category** check box in the Data Source Definition page Figure 25 on page 45 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 checked, the attribute group is placed in the selected subfolder (availability or performance) in the Key Metrics folder.

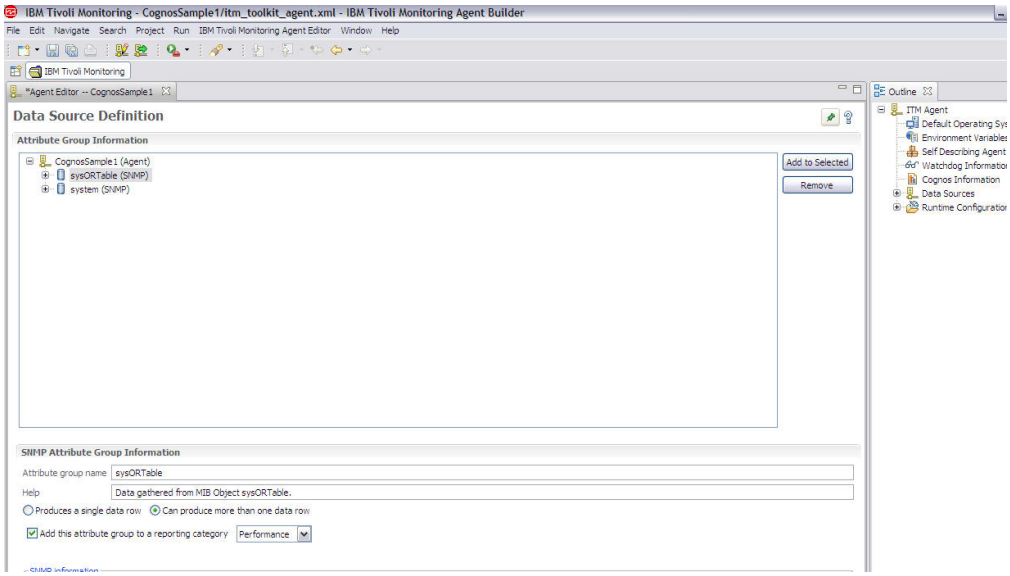


Figure 25. Data Source Definition page for Cognos

For more information about the data source fields, see Table 5 on page 53.

Generate Agent Wizard link

When you have finished creating or editing the new agent, 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 377).
- Create a package so the agent can be installed on other systems. See (“Creating the agent image” on page 379).
- Generate an ITM 5x mapping file. See (Appendix F, “Upgrading custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6.2 agents,” on page 613).

.

Commit Agent Version link

When you are certain you are finished developing this version of the agent and you are ready to deliver it, click the **Commit Agent Version** link. See “Committing a version of the agent” on page 49 for more details.

Note: You are prompted for a new version number the next time you save a change to the agent. To avoid this prompt, you can change the version in the General section as part of your first change after the agent has been committed.

Data Source Definition page

The Data Source Definition page (Figure 26 on page 47) lists the data sources configured for the IBM Tivoli Monitoring Agent. 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 tree view and selecting one of the options. You can remove data sources and attributes by

right-clicking on them and selecting **Remove**. 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 concerning monitoring with data sources, see the chapters in this user's guide about the individual data sources.

Data sources that result in Tivoli Monitoring attribute groups (all except Availability and Windows Event Log data sources) can be copied to the clipboard and pasted back to this agent or another agent.

1. Select the attribute groups that you want to copy.
2. Cut or copy the attribute group using one of the following steps, depending on whether you want to remove the data sources that have been placed on the clipboard (cut) or leave the data sources in place (copy):
   - Click **Edit** > **Cut** or **Edit** > **Copy** from the menu bar.
   - Right-click one of the selected items and click **Cut** or **Copy** from the context menu.
   - Use one of the operating system or Eclipse key strokes that invokes the cut or copy action. For example, on Windows systems, pressing **Ctrl-C** invokes the copy action.
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 steps:
   - 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 context menu.
   - Use one of the operating system or Eclipse key strokes that invokes the paste action. For example, on Windows systems, pressing **Ctrl-V** invokes the paste action.

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

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

You can specify the name and help text for a navigator group by clicking **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 desired, or drag them out to the top level of the tree.

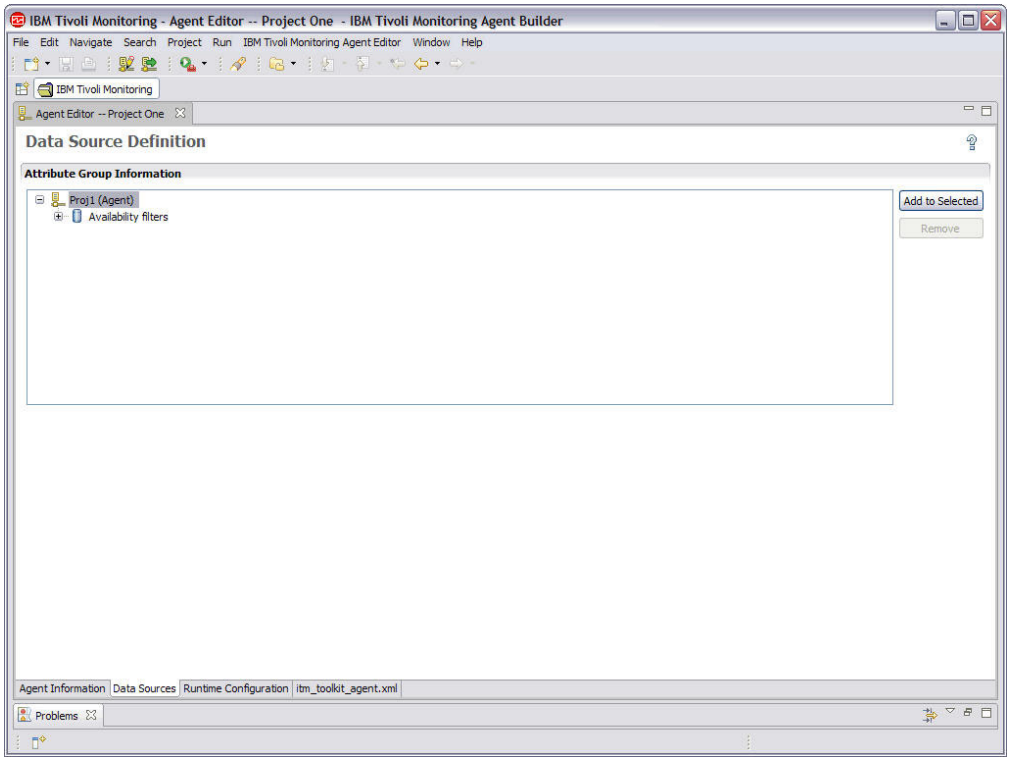


Figure 26. Data Source Definition page

Runtime Configuration Information page

The Runtime Configuration Information page displays the values defined during configuration. These values are made available to command return codes and scripts through the environment. To open the Runtime Configuration 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 using the Runtime Configuration Information page, see ("Changing configuration properties using the Agent Editor" on page 367).

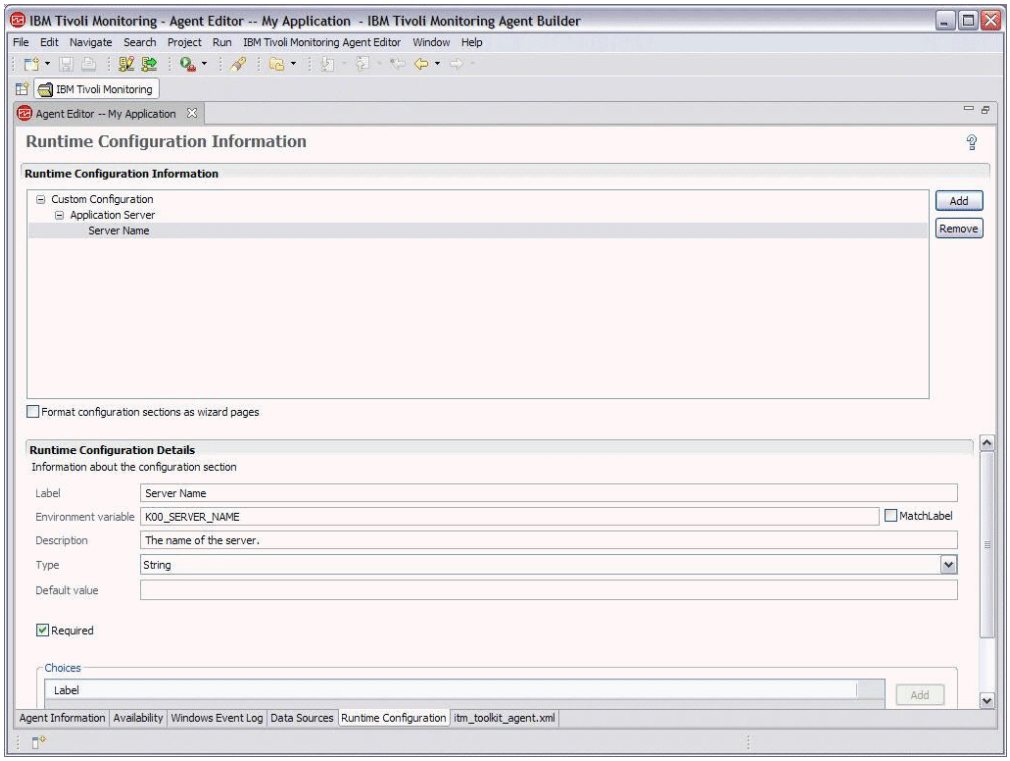


Figure 27. Runtime Configuration Information page

Agent XML Editor page

The Agent XML Editor page (Figure 28 on page 49) displays the XML for the IBM Tivoli Monitoring Agent definition. Changes made in the XML are reflected in the Graphical User Interface (GUI) pages 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 switching to it.

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

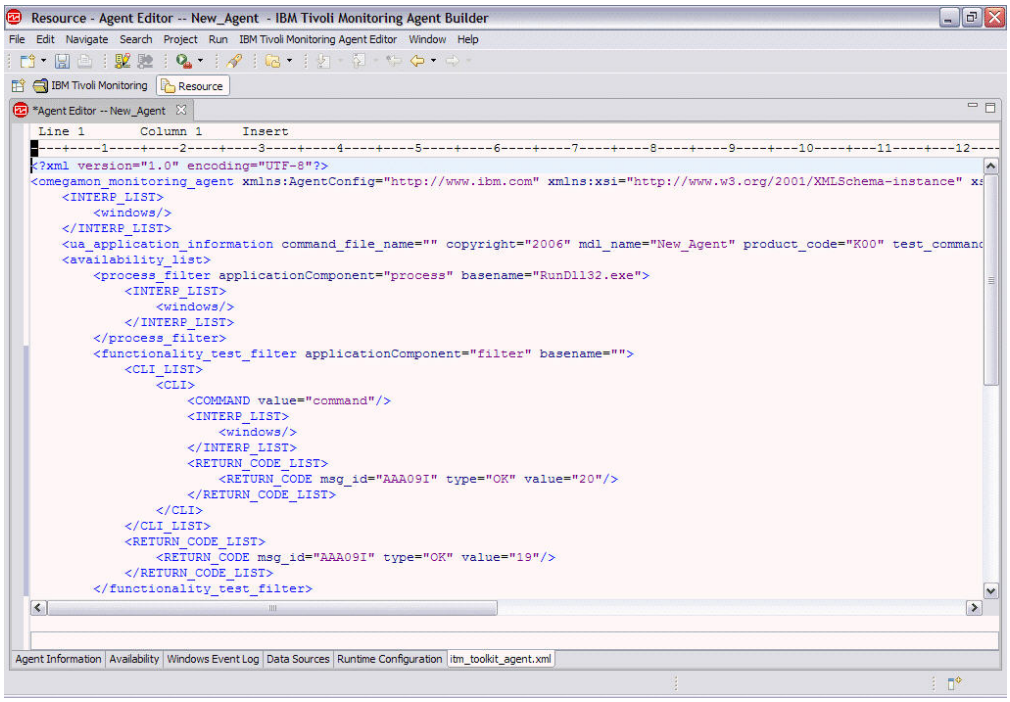


Figure 28. Agent XML Editor page

Saving your edits and changes

Changes you make with the editor are not stored until you **Save**. You can perform a save by either selecting **File** > **Save**, clicking the Save (diskette) icon, or pressing **Ctrl+S**. When saving, 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

Tivoli Monitoring systems require that new versions of an agent include all of the information 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 have completed developing and testing an agent, you must commit the agent. After committing an agent, any additional changes to the agent are part of a new version whose number you must enter before the new changes can be saved. Any changes to the new version must not break compatibility with previous versions of the agent.

There is a limit of 1024 versions.

After committing the agent, you cannot perform 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 (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. See "Changing the product code" on page 51 for additional information.

You can perform the following actions after committing the agent:

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

Steps to commit the agent

When you have finished testing your agent, and you are ready to ship it, commit the agent using the following steps:

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

Changing a subsequent version of your agent

When you are ready to make changes for a subsequent version of your agent, complete the following steps:

1. Open the Agent Editor window, Agent Information page.
2. Enter a new version or patch level after the Version prompt.
3. Make changes to your agent.

If you forget to change the agent version, you are prompted for the new version when you save any of your changes.

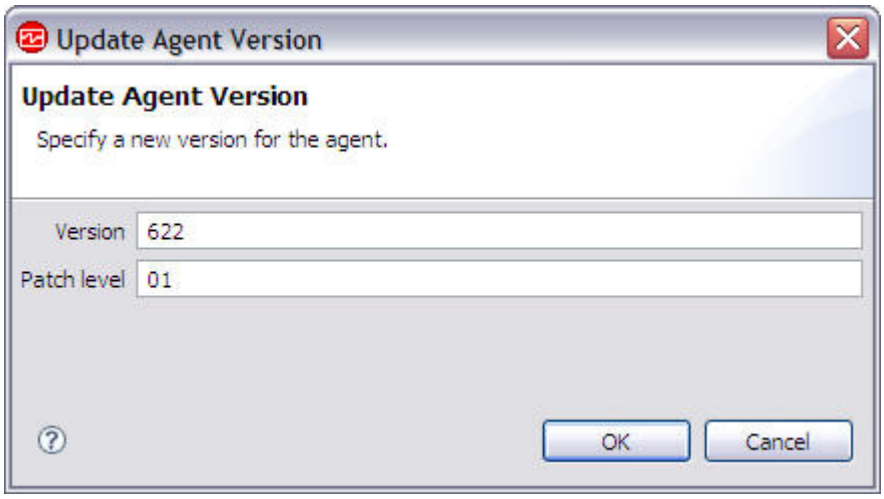


Figure 29. Updating the agent version after commit

Changing the product code

If you change the product code, you will have an agent that is incompatible with any previous version of the agent. Any record of previous commit actions will be lost and you will essentially be developing a new agent. Any files, situations, Take Action commands, or workspaces that you have exported from IBM Tivoli Monitoring and imported into the agent are deleted from the agent. If you try to change the product code, you will be reminded of this and asked if you want to continue as shown in Figure 30.

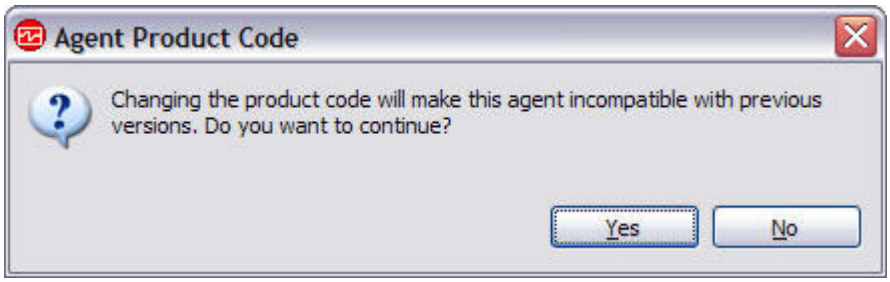


Figure 30. Agent Product Code change

When you click **Yes** in the Agent Product Code window (Figure 31 on page 52), the following information is displayed indicating that the contents of the agent support files are no longer valid, and the files will be removed next time the agent is saved.

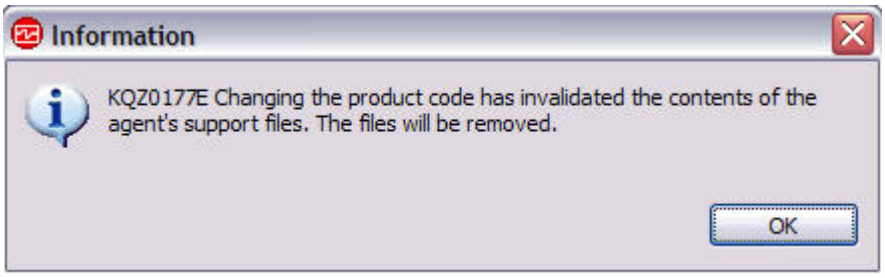


Figure 31. Agent support files invalidated

Chapter 7. Editing data source and attribute properties

This chapter covers the following topics:

- "Creating attributes" on page 54
- "Copying attributes" on page 61
- "Editing attributes" on page 62
- "Removing attributes" on page 62
- "Creating derived attributes" on page 62
- "Editing derived attributes" on page 66
- "Filtering attribute groups" on page 66
- "Formula Editor" on page 66
- "Formula operators and functions" on page 81
- "Specifying operating systems" on page 86

To edit information in an existing data source or to remove information in a data source, select the data source on the Agent Information page. The lower part of the page is updated to display the properties for the selected data source. Alternatively, if you are on the last page of the New Agent wizard (the Data Source Information page), you can double-click the data source to launch the Data Source Properties window with the same information as the lower part of the Agent Information page.

Table 5 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 5. Fields for editing data sources

Field name	Description	Acceptable values and examples
Data Source name	Name of the data source as it is displayed in the Tivoli Enterprise Portal	Acceptable values: User-friendly 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.
Produces a single data row	The data source returns one row of data. Editable in all sampled data sources.	Example: If monitoring physical system memory, choose a single row, because a system typically manages all of its memory in a single pool; so only one row of data can be returned.

Table 5. Fields for editing data sources (continued)

Field name	Description	Acceptable values and examples
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 monitoring disk drives, choose multiple rows, because there can be more than one disk in a system. For keys, choose the attributes that distinguish one disk from another. In the case of 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, one row of data per event.	Example: A 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 will be 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 will be assigned.	Select either Performance or Availability.

Notes:

1. The **Produce a single data row** and **Can produce more than one data row** fields do not impact data for an event data source.
2. For more about sampled and event data types, see "Data types" on page 418
3. For information about the fields for a specific data source, see the data source's chapter in this user's guide.

Creating attributes

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

Add or edit attributes using the following procedure:

1. Right-click the data source and select **Add Attribute** on the menu. The Attribute Information page is displayed (Figure 32 on page 55).

   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 (Figure 32 on page 55).

See “Fields and options for defining attributes” for information about the fields and options for this page.

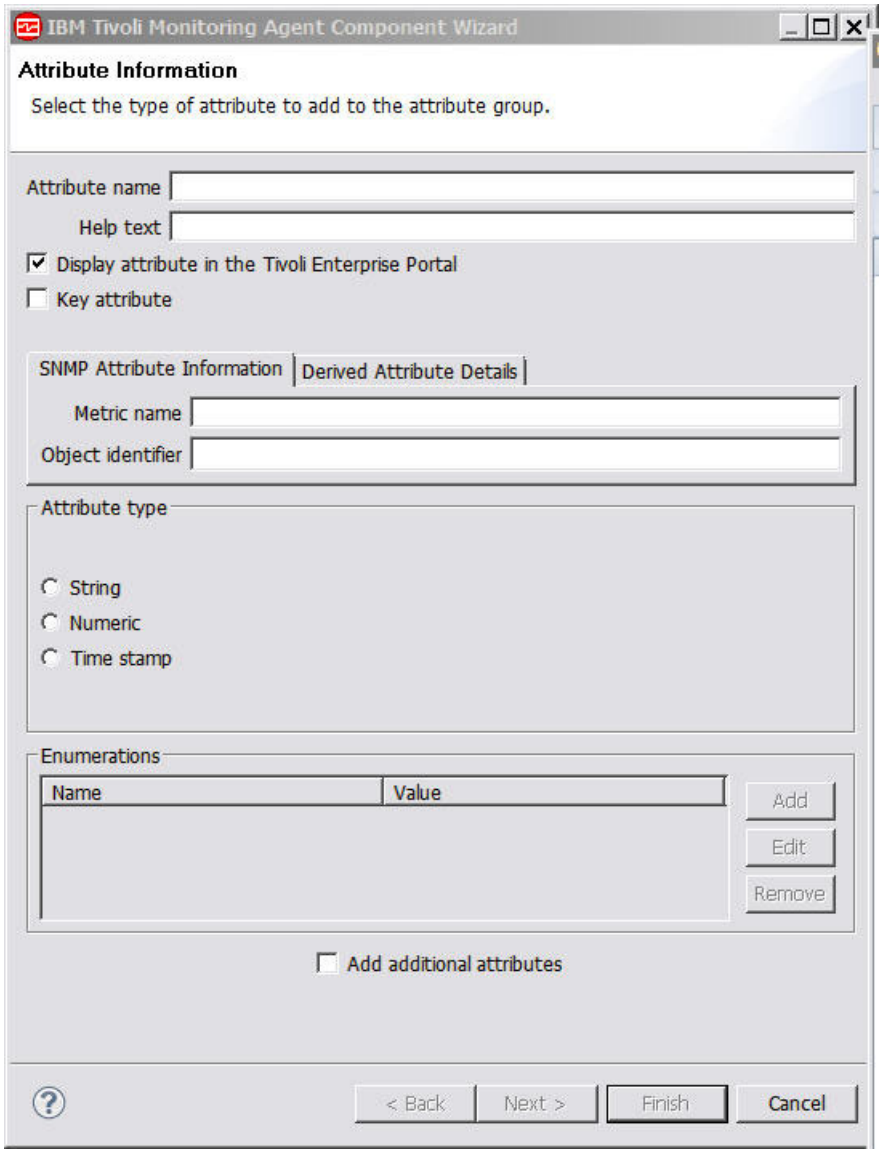


Figure 32. Attribute Information page

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

Fields and options for defining attributes

Table 6 on page 56 describes the field information and options for this page that are applicable to all of the data sources. For information about the specific field information for each of the data sources, refer to the chapter in this user's guide for each data source.

Table 6. 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 below)	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 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 66.	

Table 6. 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 three types: • String • Numeric • Time stamp "Attribute types" contains more information about the attribute types.	Table 7 on page 59 contains descriptions of the numeric attribute type values.
Enumerations	This can be a numeric with scale zero or string value.	Add your enumerations to the table using the procedure in "Specifying an enumeration for an attribute" on page 61. 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 types of attributes:

- 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 on page 58. The default size is 64 bytes.

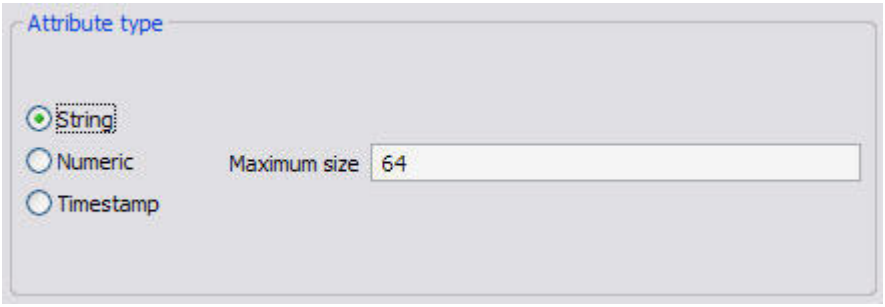


Figure 33. String attribute type

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

Numeric

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

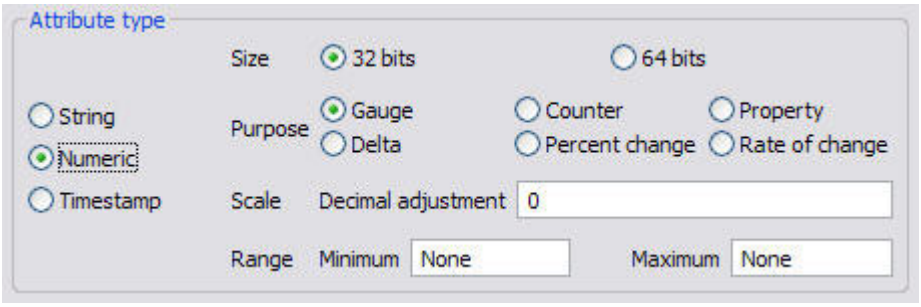


Figure 34. Numeric attribute type

See Table 7 on page 59 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 using 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 attribute options

When you specify a numeric attribute, you must specify the size, purpose, scale, and range of the attribute. Table 7 on page 59 contains descriptions of each of these aspects of an attribute.

Table 7. 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 negative two billion to two billion). The value of 64-bit numbers can range from -9223372036854775808 to 9223372036854775807 (roughly negative nine quintillion to nine quintillion).

Table 7. Numeric attribute options (continued)

Numeric aspects	Options and fields	Description
Purpose	**Gauge**	Integer values where the raw values returned are higher or lower 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 containing 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 the Delta-based calculations, where the detailed data values in one hour are 9, 15, 12, 20, 22, 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 collected during the period.
	Delta	An integer value representing 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 representing the difference between the current value and the previous value 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.

Table 7. Numeric attribute options (continued)

Numeric aspects	Options and fields	Description
Scale	**Decimal adjustment**	Scale determines how many decimal places are in the number. Each decimal place reduces the range mentioned above 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 described above 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

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. To specify enumeration values, use the following 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).

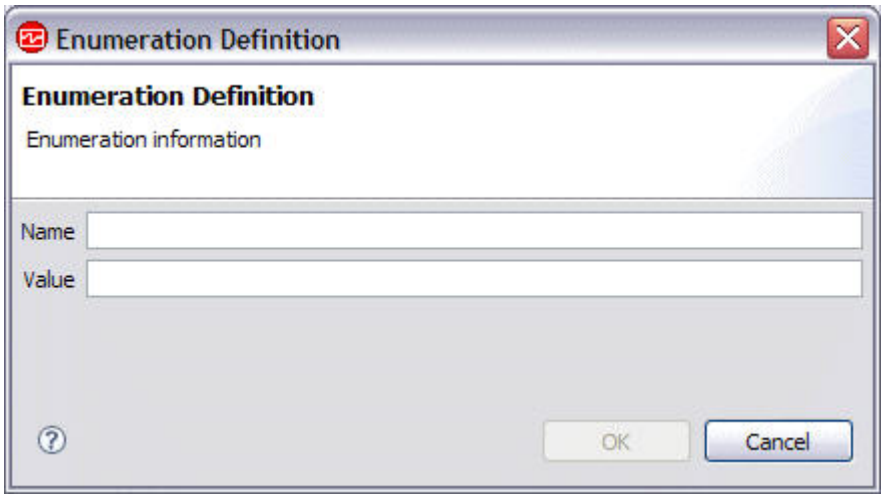


Figure 35. Defining enumeration for an attribute

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

Copying attributes

Perform the following steps to copy an attribute:

1. In the Agent Editor, Data Source Definition page, right-click the attribute you want to copy, and click **Copy Attribute**.
2. In the Copy Attribute window (Figure 36), type the name of the new attribute in the **Name** field, and click **OK**.

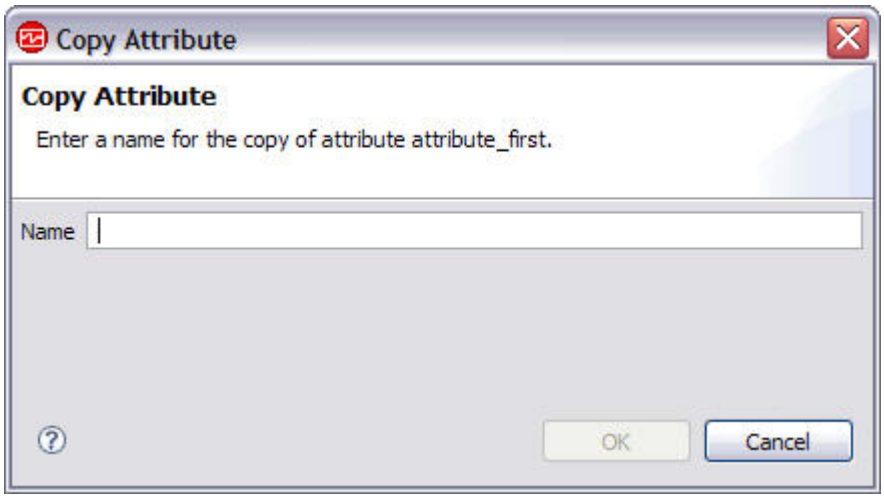


Figure 36. Copy Attribute window

Editing attributes

Edit an attribute using the following procedure:

1. Select the attribute on the Agent Information page.

   The lower part of the page is updated to display the properties for the selected attribute.
2. Specify your choices for the new attribute.

Note: If you are on the last page of the New Agent Wizard (the Data Source Information page), you can double-click the attribute to launch the Data Source Properties window that contains the same information as the lower part of the Agent Information page.

Removing attributes

To remove an attribute or attributes, right-click the attribute or attributes and select **Remove attribute(s)** from the menu that is displayed (Figure 37 on page 63).

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

The basic expression syntax used for derived expressions contains functions. These functions provide a more complicated manipulation of data including 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.

To add a new derived attribute, use the following procedure:

1. On the Data Source Definition page, right-click the data source and click **Add Attribute**.

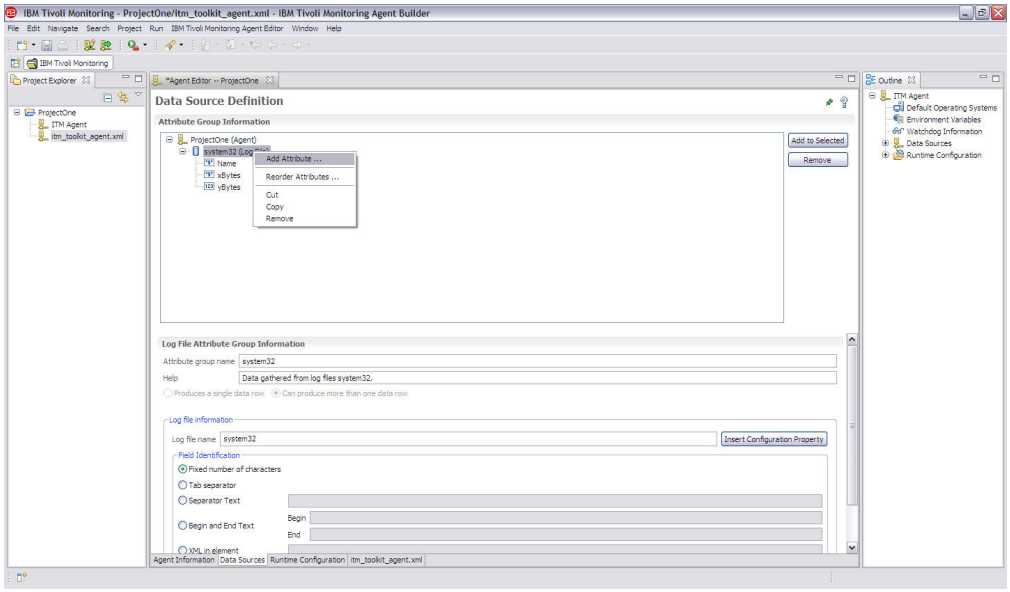


Figure 37. Adding an attribute

2. On the Attribute Information page, type an Attribute name and Help text (Figure 38 on page 64).

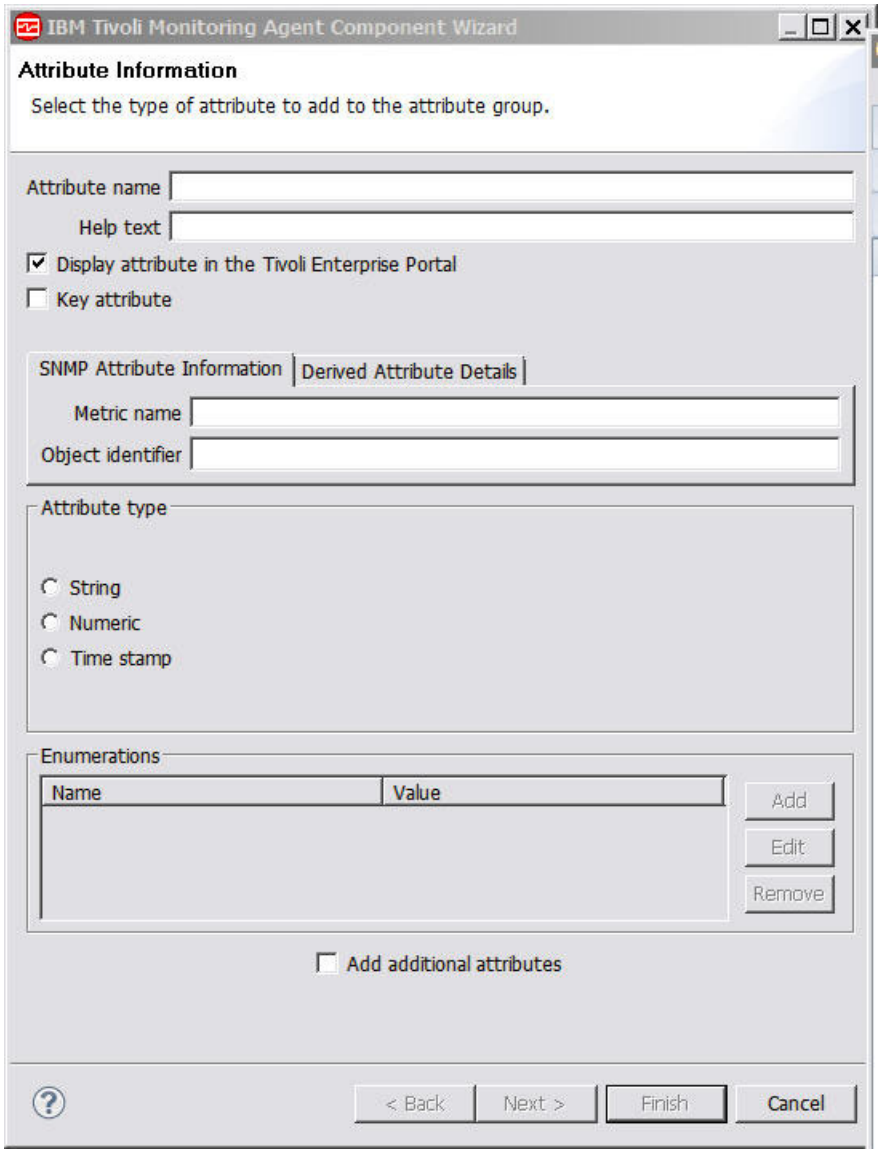


Figure 38. Attribute Information page

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.

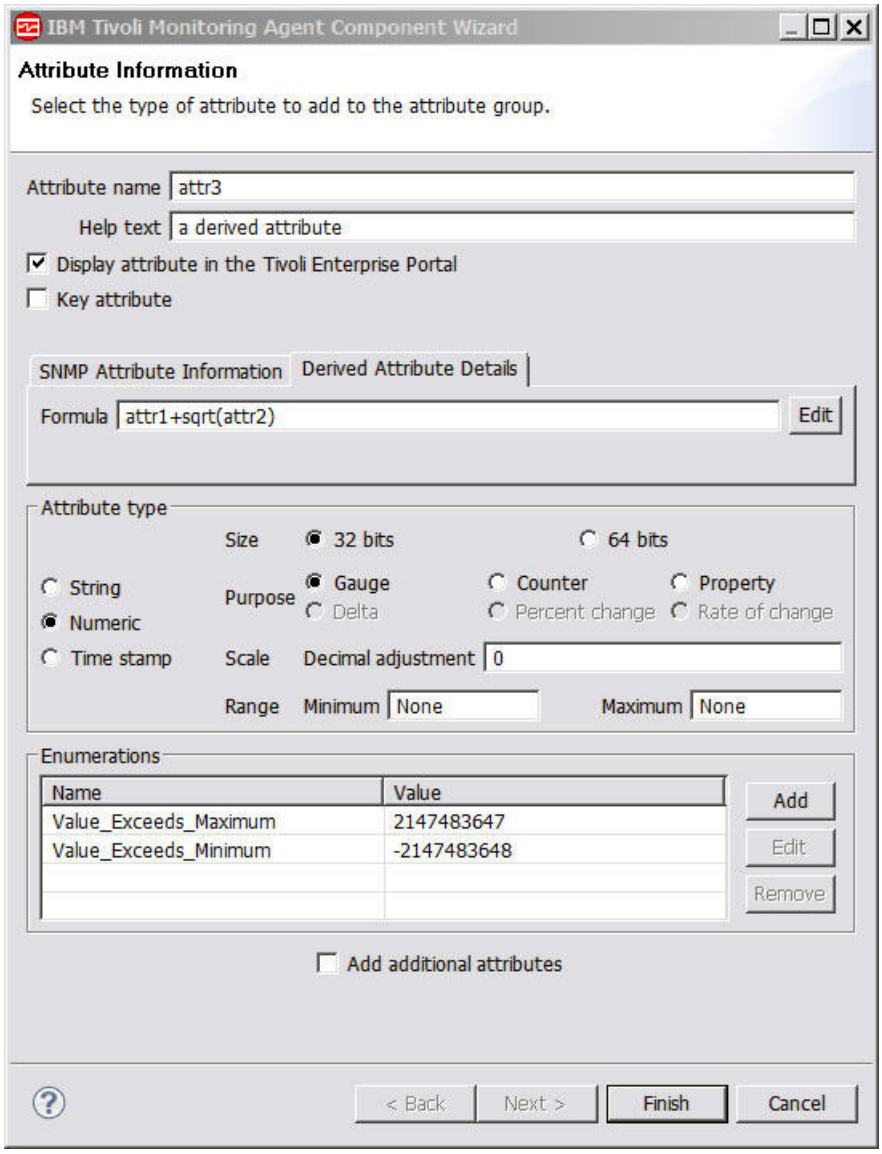


Figure 39. Attribute Information page with Derived Attribute Details tab open

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 81 for information about the operators and functions that can be used in the formula.

   Note: When you click **Edit**, the Formula Editor is displayed. See "Editing derived attributes" on page 66 for information about editing derived attributes.
5. In the **Attribute type** area, click the type of attribute.
6. Click **OK**. The Data Source Definition page is displayed again with the data source listed in it as before (see Figure 84 on page 118).
7. Click **Finish**.

Editing derived attributes

You use the Formula Editor to edit derived attributes. This editor is available from the **Derived Attribute Details** tab on the Attribute Information page as described in Step 3 of “Creating derived attributes” on page 62. For more information about the Formula Editor, see “Formula Editor”

Filtering attribute groups

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

To create an attribute group filter, use the following procedure:

1. Use one of the following steps to begin creating the filter
   a. If you are creating an attribute group, click **Advanced** in the initial data source information page.
   b. If the attribute group exists, select the attribute group in the Agent Editor and click **Advanced** in the Data Sources page
2. In the Advanced Data Source Properties page, enter a selection formula. The selection formula you enter must evaluate to a Boolean result, true, or false.

   Note: In the Advanced Data Source Properties page, you can click **Edit** to enter or modify the formula using the Formula Editor. For more information about the Formula Editor, see “Formula Editor”
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.

You can validate that the filter is performing the intended task using the test function for the attribute group. For more information about attribute group testing, see “Attribute group testing” on page 373

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 performing one of the following tasks:

1. Creating or editing derived attributes, see “Creating derived attributes” on page 62 and “Editing derived attributes”
2. Creating Filtered Attribute groups, see “Filtered Attribute Groups” on page 323
3. Filtering data from attribute groups, see “Filtering attribute groups”

Attention:

- When creating 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 creating Filtered attribute groups or when filtering 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 using 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 40 on page 68) 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 at the bottom of the Formula Editor window. Alternatively you can click **Insert** to begin entering a formula using the editor menu options. The editor contains two views of the formula in the default window, and an option for a third view:

- **Formula field (default view):** The complete formula is in the field at the bottom of the window. You can edit the formula by typing in this box.
- **Component view (default view):** The component of the formula being edited is in the upper part of the window. In Figure 40 on page 68, the + **Operator** is the current component. The operator and its two operands can be edited.
- **Formula hierarchy tree view:** The formula hierarchy tree is displayed on the left side of the window by selecting the **Show formula hierarchy** check box. See Figure 41 on page 69. The state of the check box is remembered in subsequent invocations of the Formula Editor.

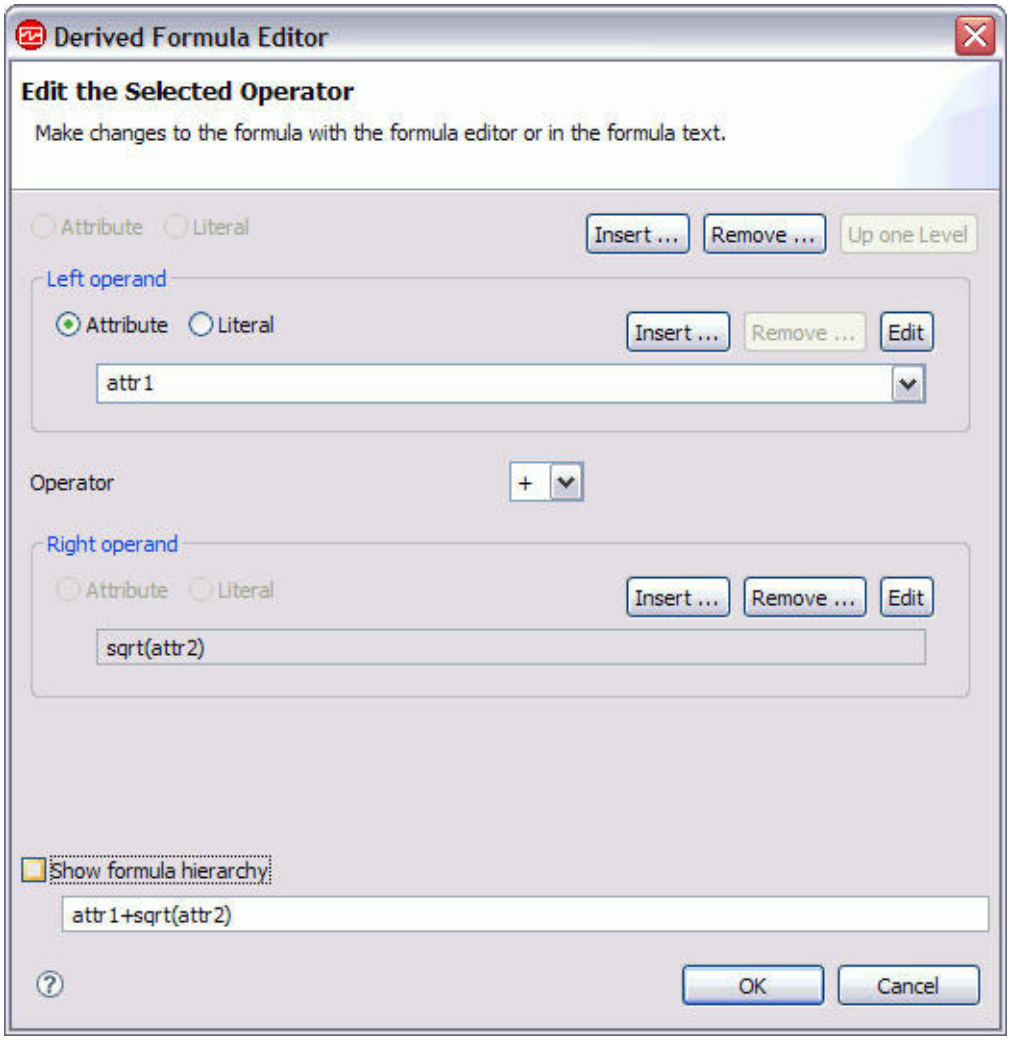


Figure 40. Derived Formula Editor window (default)

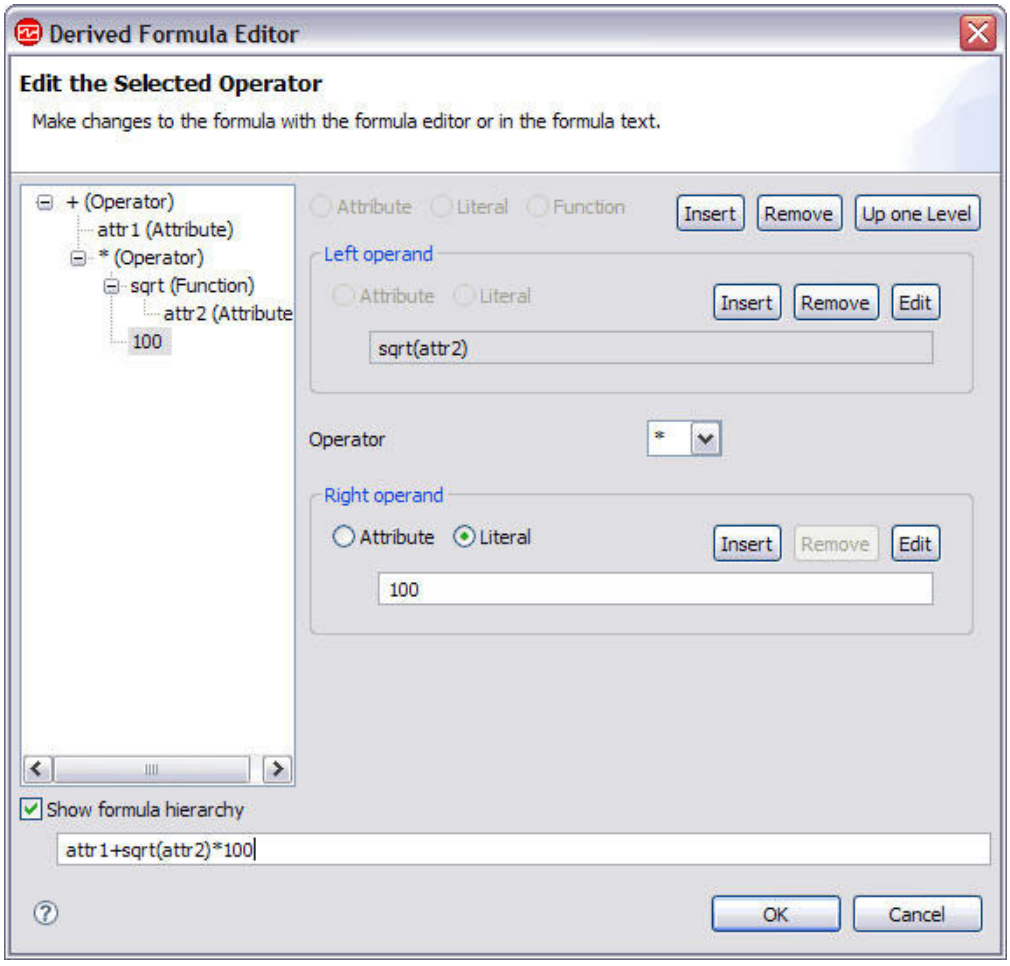


Figure 41. Derived Formula Editor view with formula hierarchy tree view

Changing the component view

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

- Move the cursor in the formula text.
- Select a different node in the formula hierarchy tree.
- Click **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 when selected:

- "Attribute component" on page 70
- "Literal component" on page 72
- "Operator component" on page 73
- "Conditional expression component" on page 74
- "Function component" on page 75

Attribute component

The attribute component shows an attribute name. You can select an attribute from a drop-down list of attributes in the attribute group.

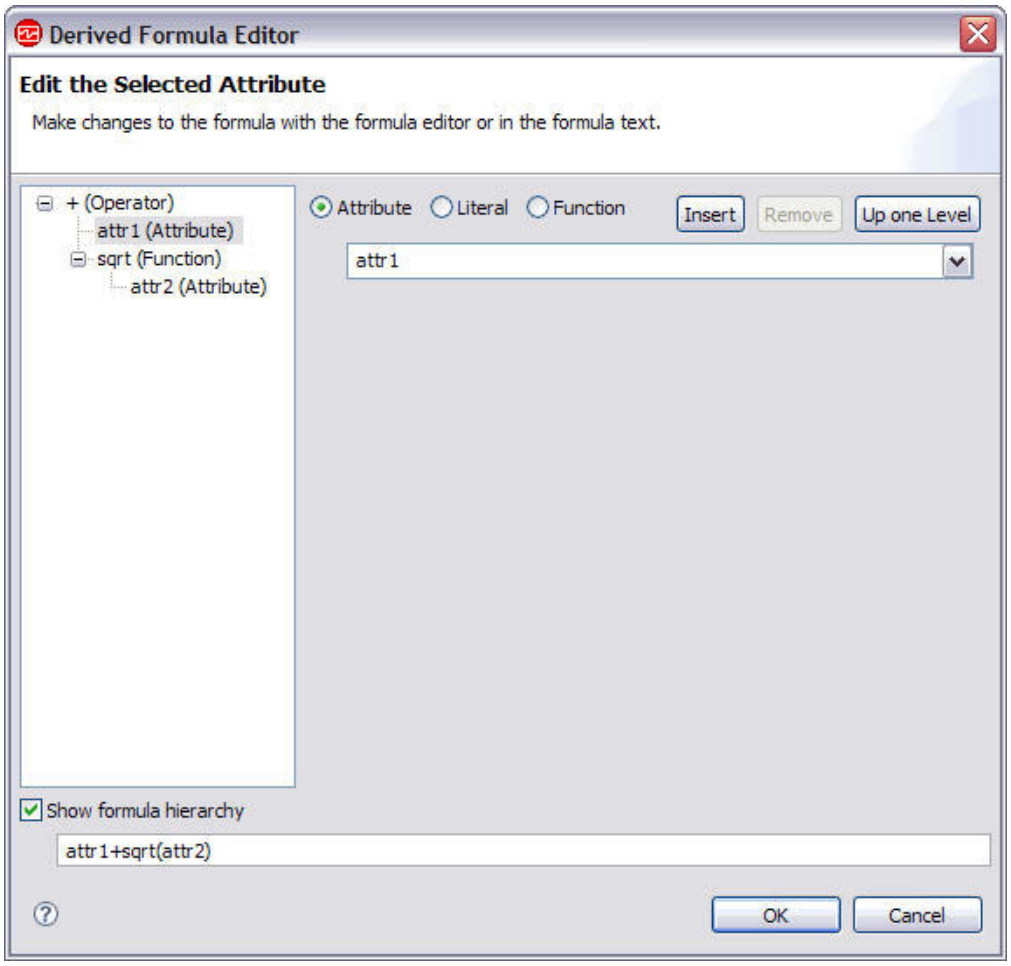


Figure 42. Edit the Selected Attribute page

- When an attribute name is selected, the other views are updated to reflect the newly selected name.
- You can replace the attribute with a literal by clicking **Literal**. The drop-down list is replaced by a standard text 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 drop-down list contains valid function names to choose from.
- You can type an attribute name instead of selecting one. This is useful if you have not yet defined 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 43 on page 71. **OK** is disabled until the error is corrected.
- When the attribute name is changed, the other views are changed to reflect the new attribute name.

- See "Common options" on page 77 for information about using the following options: **Insert**, **Remove**, and **Up one Level**.

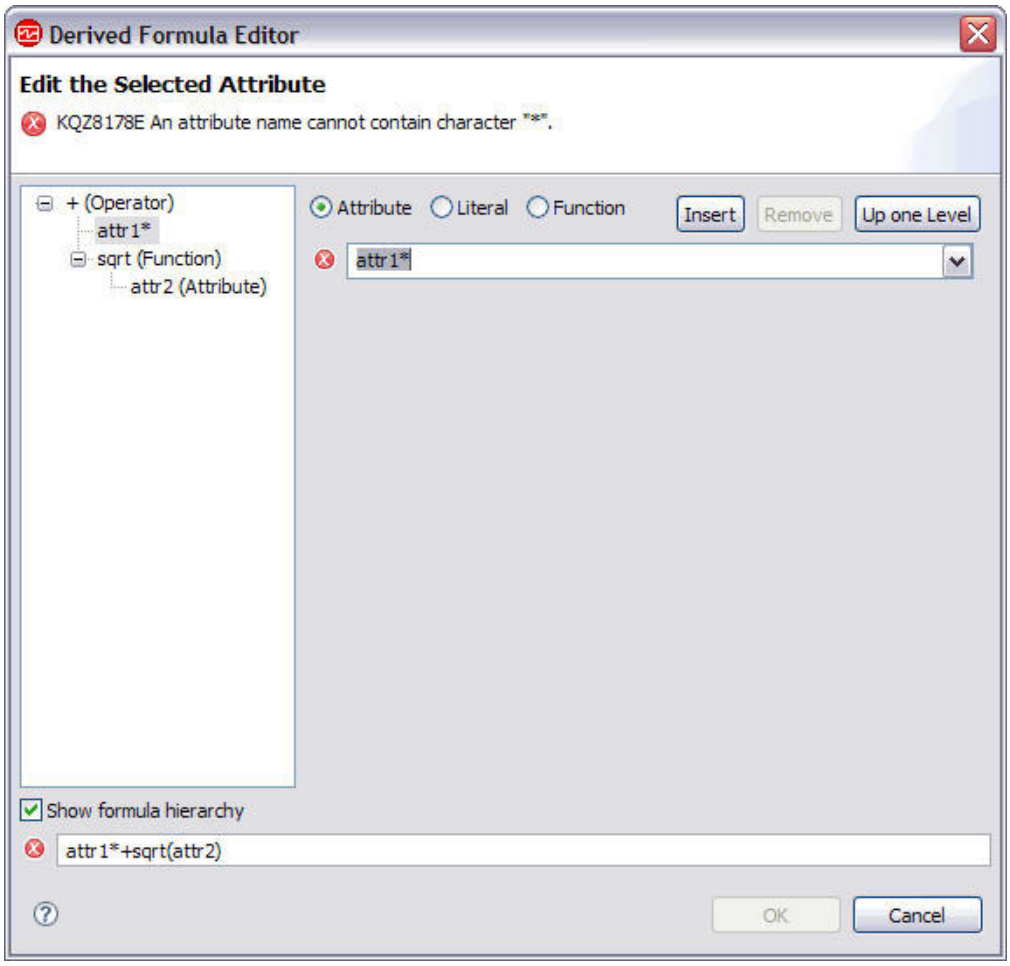


Figure 43. Edit the Selected Attribute page with error

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 44 on page 72 is displayed.

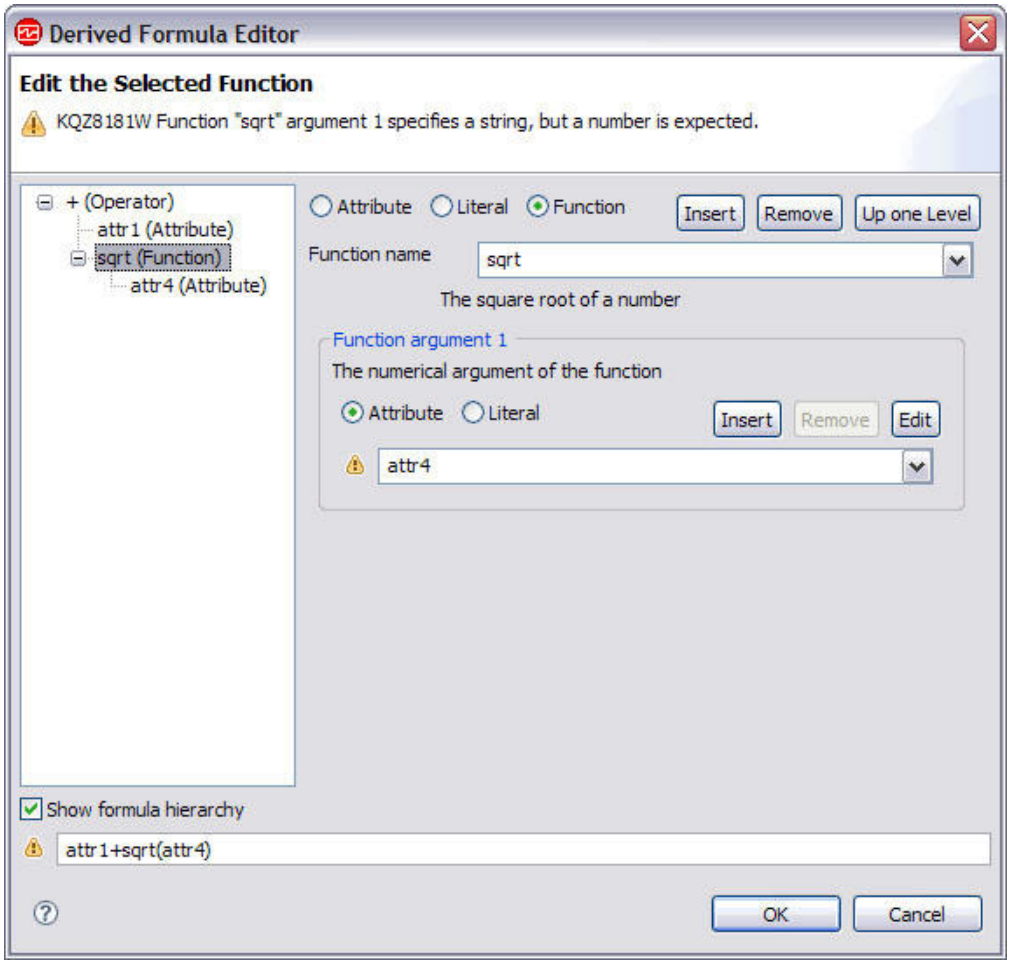


Figure 44. Edit the Selected Function page with warning

Literal component

A literal is any value that is entered directly in the formula that does not come from an attribute value or from a function. Literal string values are enclosed in double quotes. Literal numerical values are not enclosed in quotes.

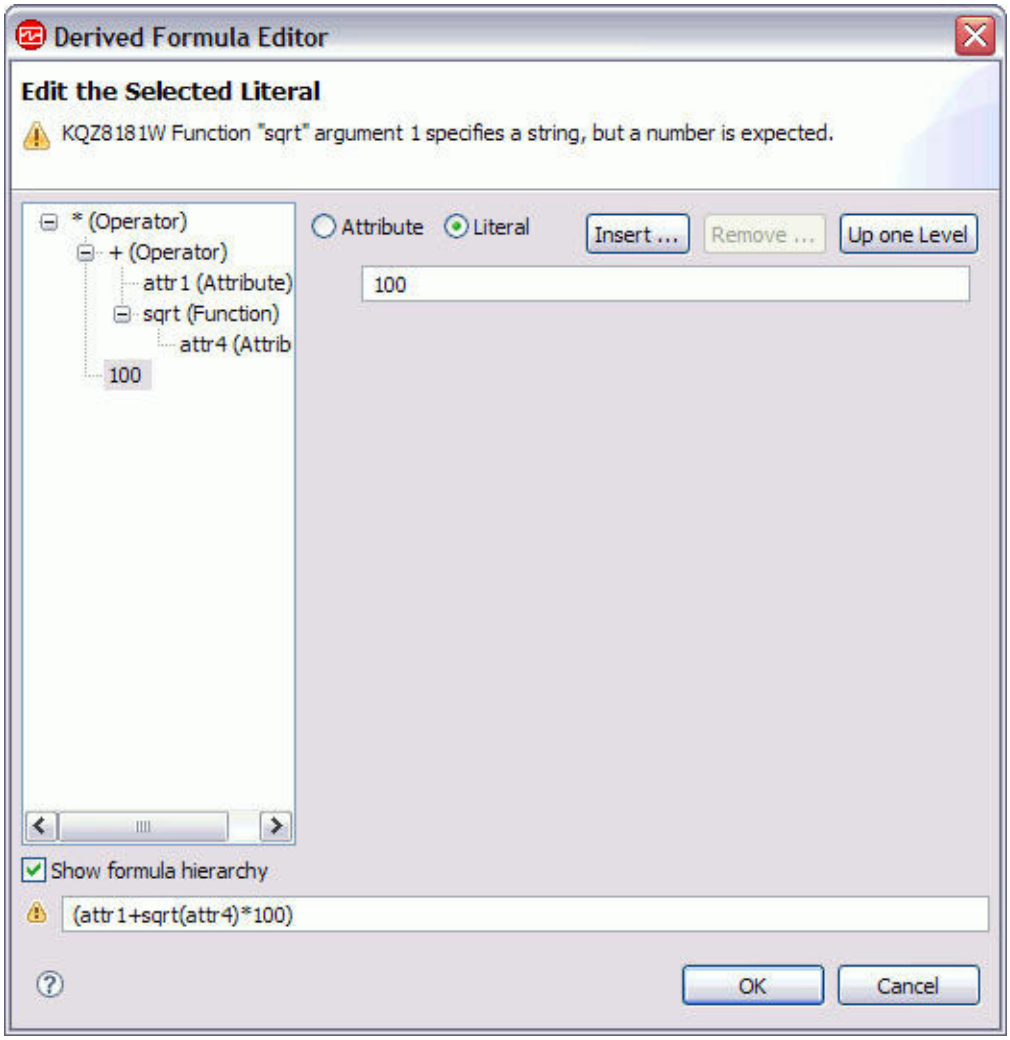


Figure 45. Edit the Selected Literal page

- You can replace a literal with an attribute by clicking **Attribute**. A valid attribute name must be selected or entered without quotes.
- You can replace a literal with a function by clicking **Function**. Parentheses are added after the name and the drop-down 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.
- An error is displayed if the field is not a number or a string literal. **OK** is disabled until the error is corrected.
- See "Common options" on page 77 for information about using the following options: **Insert**, **Remove**, and **Up one Level**.

Operator component

An operator component shows an operator and its operands.

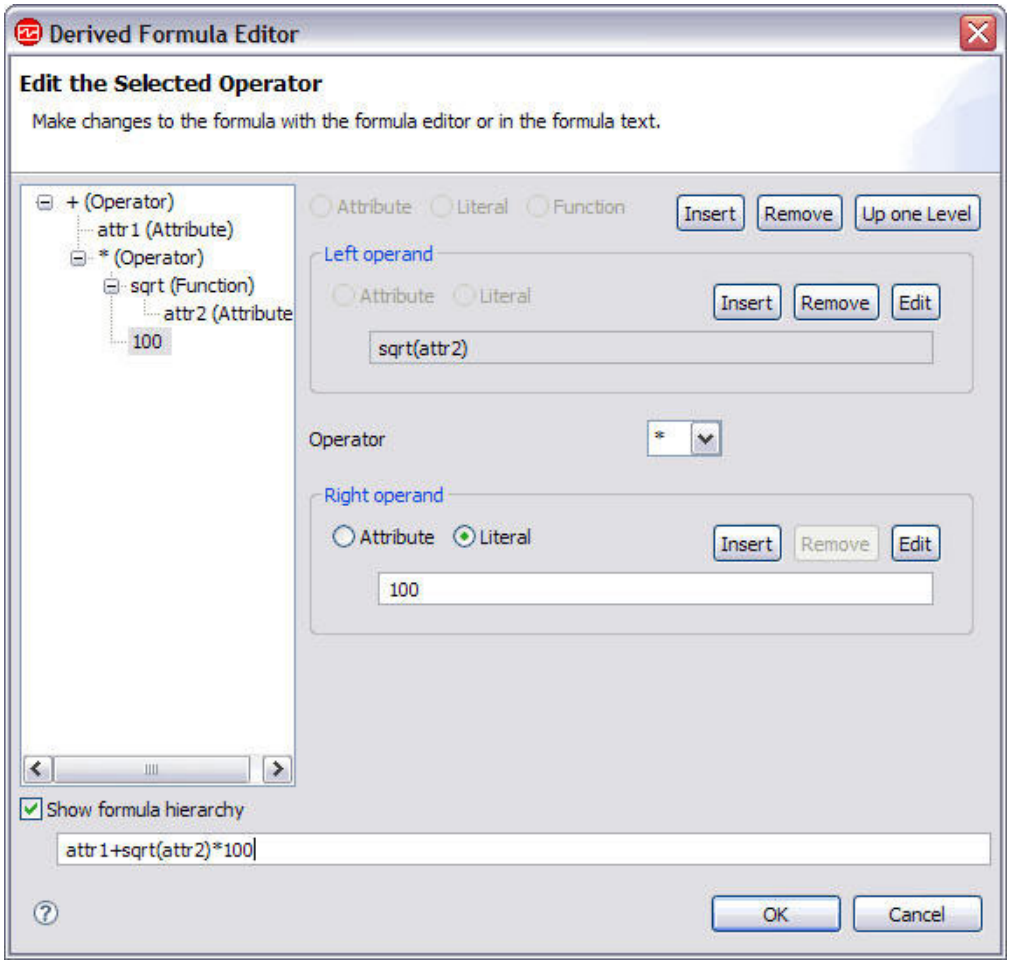


Figure 46. Edit the Selected Operator page

- In the **Operator** drop-down list, between the two operands, select the operator (+ - * / or %). The % operator multiplies the first operand by 100, and then divides by the second operand.
- The **Left operand** section of the page is above the operator,
- The **Right operand** section is below the operator.
- Simple operands (attributes and literals) can be edited without having to change the selected component to the operand as described in "Attribute component" on page 70 and "Literal component" on page 72.
- 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.
- See "Common options" on page 77 for information about using the following options: **Insert**, **Remove**, **Up one Level**, and **Edit**.

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.

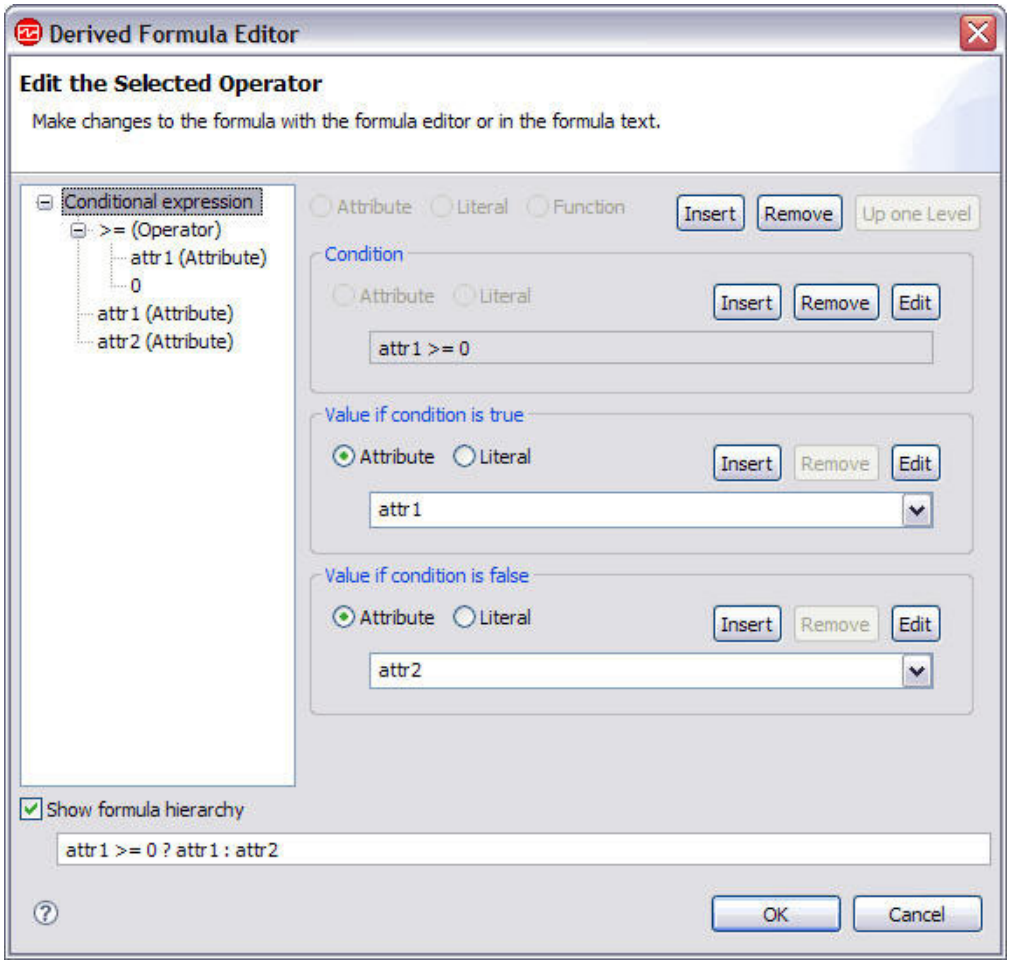


Figure 47. Edit the Selected Operator page (for a conditional operator)

- 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 "Attribute component" on page 70 and "Literal component" on page 72.
- 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 "Common options" on page 77 for information about using the following options: **Insert**, **Remove**, **Up one Level**, and **Edit**.

Function component

The function component shows the function and its arguments.

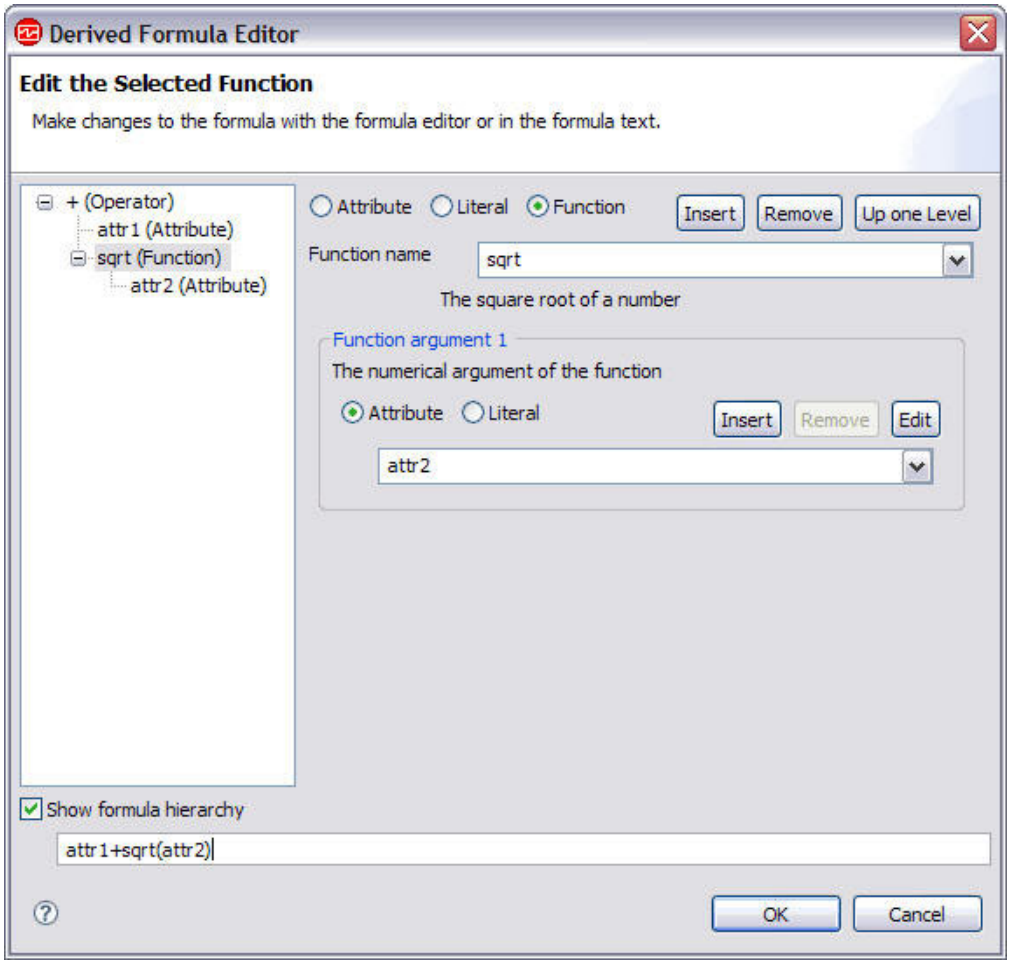


Figure 48. Edit the Selected Function page

- **Function name:** Select the function name from the drop-down list. The description of the selected function appears below the function.
- **Function argument** sections are shown below the function name. The appropriate number of arguments for the selected function are shown. The top line in the **Function argument** section is a description specific to the function selected.
- Simple arguments (attributes and literals) can be edited without having to change the selected component to the operand as described in "Attribute component" on page 70 and "Literal component" on page 72.
- 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, you can add arguments by clicking **Insert** or remove arguments by clicking **Remove** in addition to the actions described in "Common options" on page 77.
- For the getenv function, a configuration property can be chosen by clicking **Insert** as shown in Figure 50 on page 78. If you select the Configuration property choice, the Configuration Properties window is displayed.
- See "Common options" on page 77 for information about using the following options: **Insert**, **Remove**, **Up one Level**, and **Edit**.

Common options

This section describes the options that are common to the views in the Formula Editor:

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

Insert

Insert inserts an operator or a function above the component. The component is demoted to one of the operator operands or one of the function arguments. For example, if you click **Insert** above the sqrt function, you are asked what you want to insert and the following choices are displayed (Figure 49):

- 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

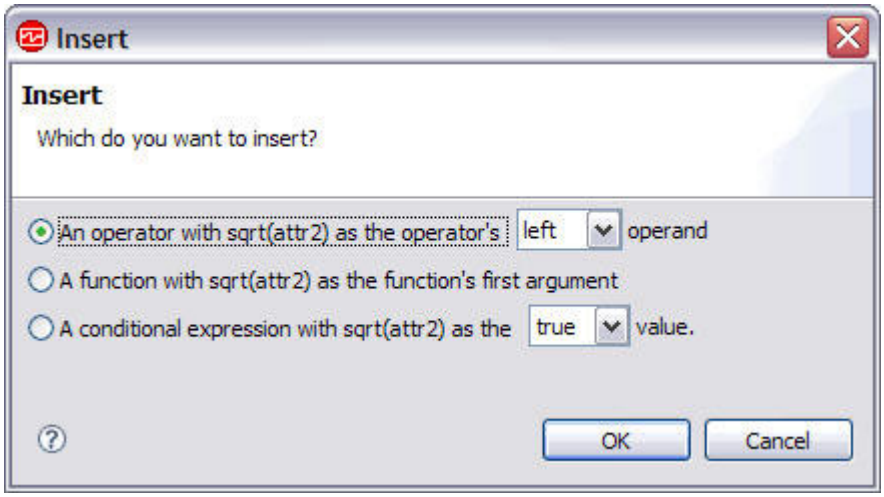


Figure 49. Insert window example

If you click **Insert** above the getenv function, you are asked what you want to insert and the following choices are displayed (Figure 50 on page 78):

- 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

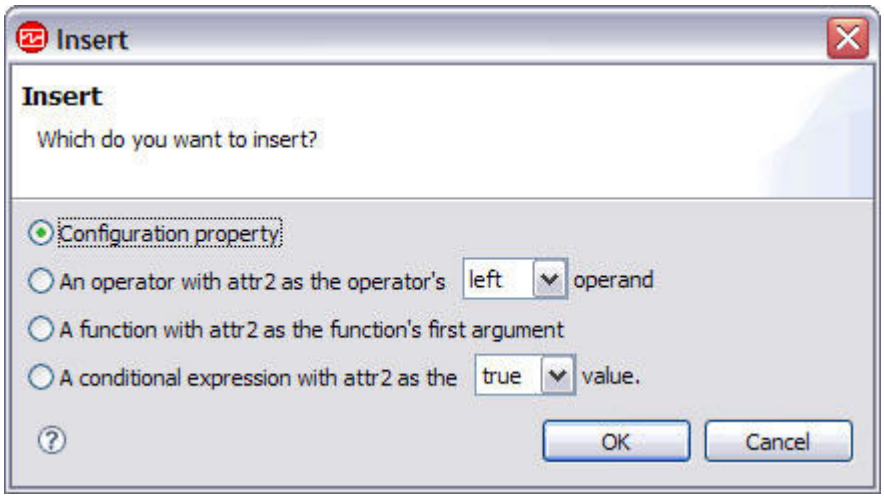


Figure 50. Insert window example for getenv function

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** above the sqrt function shows the following choices (Figure 51):

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

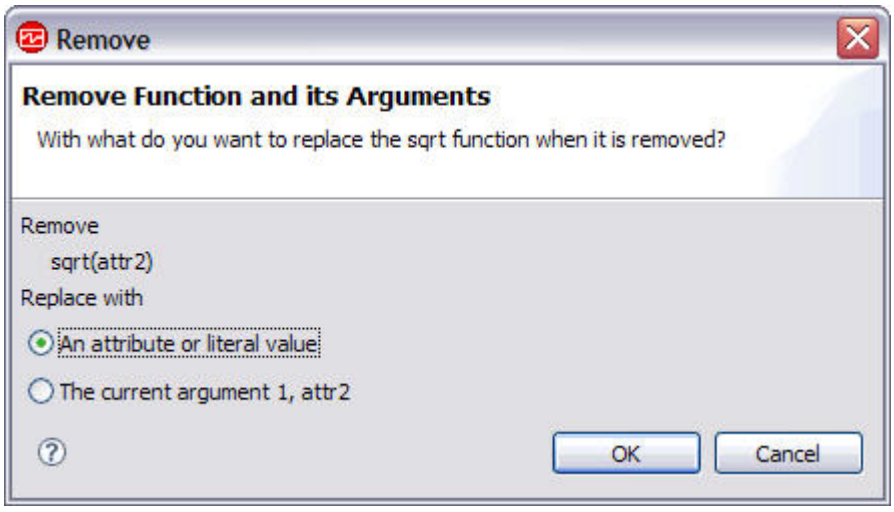


Figure 51. Remove function and its Arguments page

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

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

Up one Level

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

Edit

Click **Edit**, just above 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 errors

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

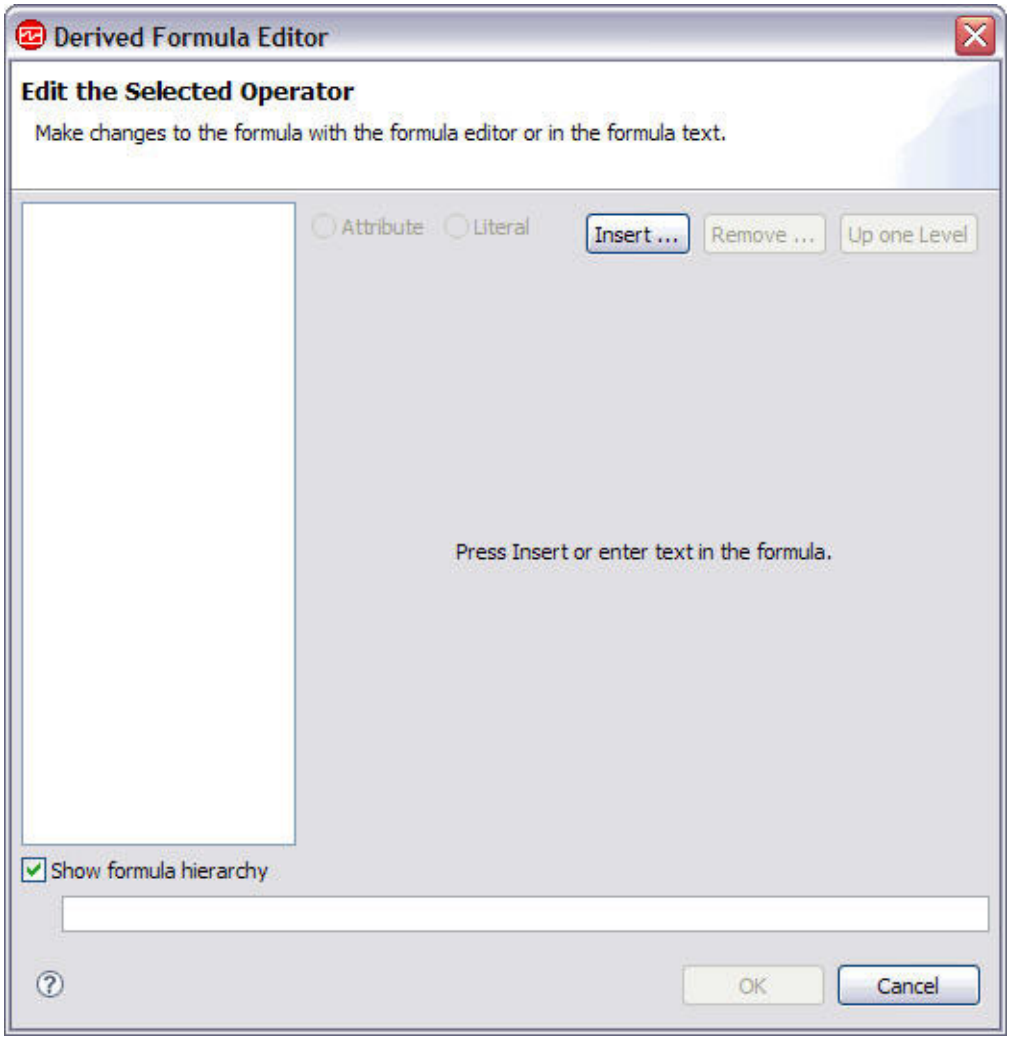


Figure 52. Empty formula example

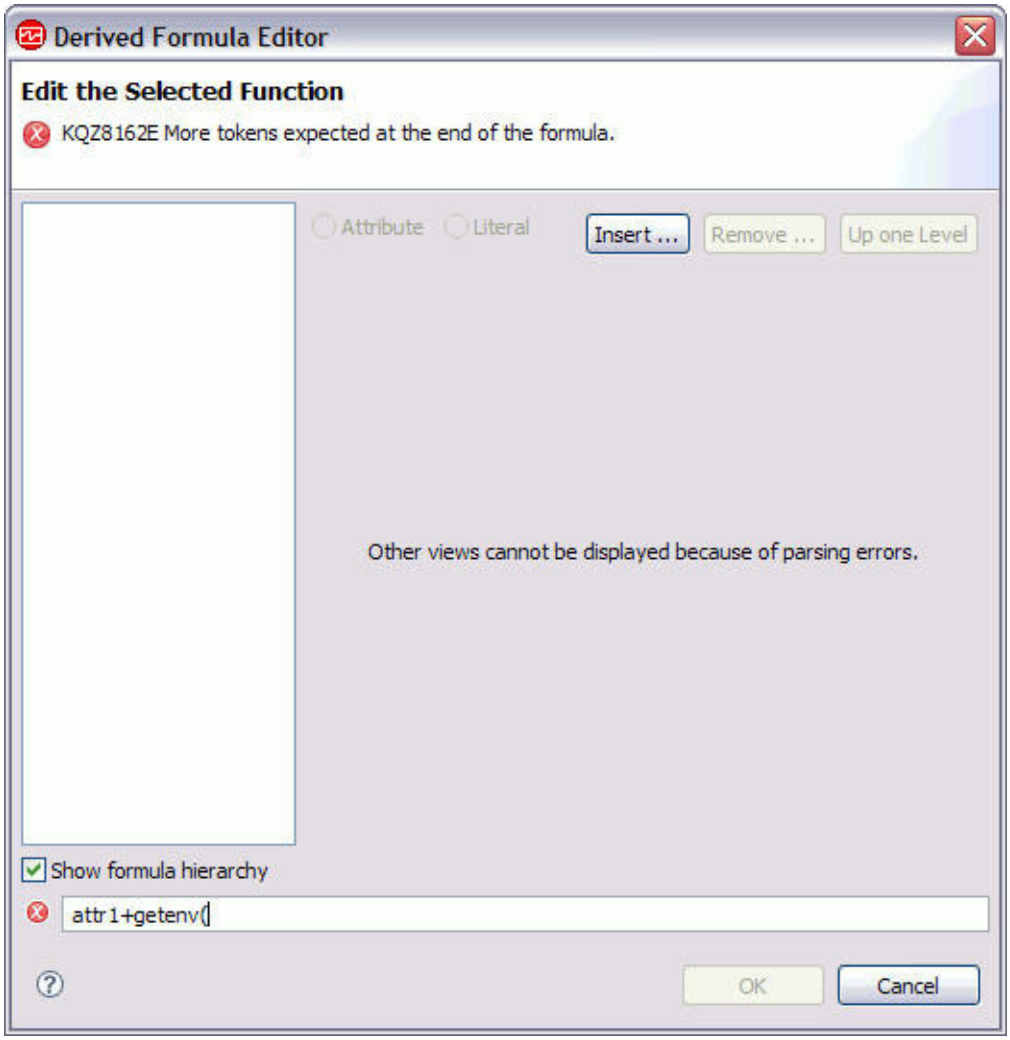


Figure 53. Parsing error example

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 54 on page 81):

- **An attribute**
- **A literal value**
- **An operator**
- **A conditional expression**
- **A function**

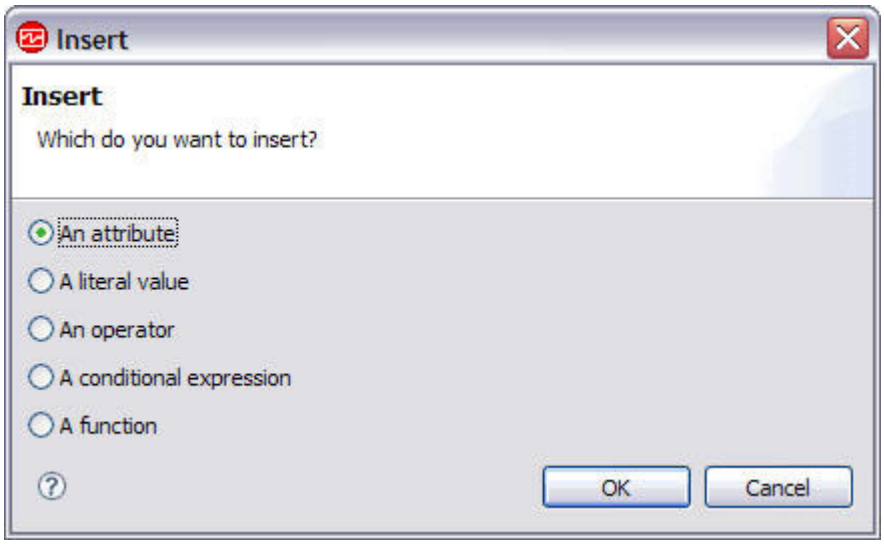


Figure 54. Insert window for fixing a formula manually

Formula operators and functions

The value for a derived attribute is the result of evaluating an expression based on constants and the values of other attributes 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 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 functions described below. 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 Source Information tree). Integer constants are specified as numbers. String constants are surrounded by quotation marks.

You can use the following functions in a formula:

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, or the "last" function can be the only argument to this function (to calculate the minimum of the most recent values of an attribute).

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, or the "last" function can be the only argument to this function (to calculate the maximum of the most recent values of an attribute).

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, or 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))
```

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, or the "last" function can be the only argument to this function (to calculate the standard deviation of the most recent values of an attribute).

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, or the "last" function can be the only argument to this function (to calculate the sum of the most recent values of an attribute).

last Returns a list of values for use by the min, max, average, and stddev 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 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 or not 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 attribute group filtering.

round Rounds the number to the nearest whole number.

abs Returns the absolute value of a number

sqrt Returns the square root 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.

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 only takes two string arguments.

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

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 may also be the string "LAST" to return the last token.

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.

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.

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 8 describes the valid format parameters. The attribute type is timestamp.

Table 8. 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
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 9 on page 84 provides examples of string representations of time stamps and the format strings used to parse them.

Table 9. 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

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.

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.

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.

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 615) for details on the regular expressions and substitution values 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 615) for details on the regular expressions and substitution values allowed in the replacement string.

The following functions take no arguments and just 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, because both are called every time, it matches the number of elements 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.

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
Returns the number of matching events represented by a flow control summary event (including the first event), or 1 if it is a single event from a

data source. This function applies only to event attribute groups with event filtering and summarization turned on.

summaryInterval
Returns the summary interval 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.

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

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

The number in parentheses is the raw value, but 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. This should be an integer > 0. This function applies only to event attribute groups with event filtering and summarization turned on.

cumulativeSum
Returns the sum of argument values of duplicate events 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.

Examples

Derived Attributes

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

- Name String
- xBytes Numeric
- 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 the percentage of the total bytes, which are yBytes, can be defined as 'yBytes % (xBytes + yBytes)' or 'yBytes % totalBytes'

Example: The following formula returns the maximum of the recently collected values for the Virtual_Size attribute. The number of samples 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: The following formula returns the square root of the sum of the squares of the xBytes and yBytes attribute values:

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

Example: The following formula returns the average of the xBytes attribute from the 20 most recent samples of the attribute group. If fewer than 20 samples have been collected since the agent was started, it returns the average of the xBytes attribute from all samples:

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

Filtering

Example: 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: 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 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 defining data sources that are not specific to the Windows operating system, you must specify the operating systems on which the data source provides data. By default, the data source provides data on all of the operating systems defined at the agent level (see “Operating system requirements” on page 15 and “Default operating systems” on page 39). But you can change the operating systems on which a particular data source functions by opening the Operating Systems section (click **Operating Systems**) and selecting the operating systems on which the data source is to operate. You can select individual operating systems, all operating systems, all operating systems of a specific type (Linux, UNIX, Windows), or the

agent default operating systems, as shown in the example in Figure 55.

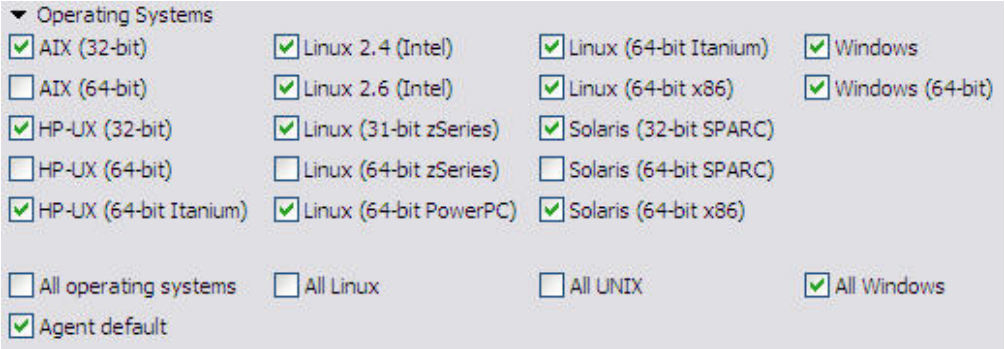


Figure 55. Operating systems where the agent is to run

Chapter 8. Monitoring a process

Use the following procedure to monitor a server process in your agent:

1. On the Agent Initial Data Source page (Figure 56) or the Data Source Location page, click **A process** in the **Monitoring Data Categories** area.

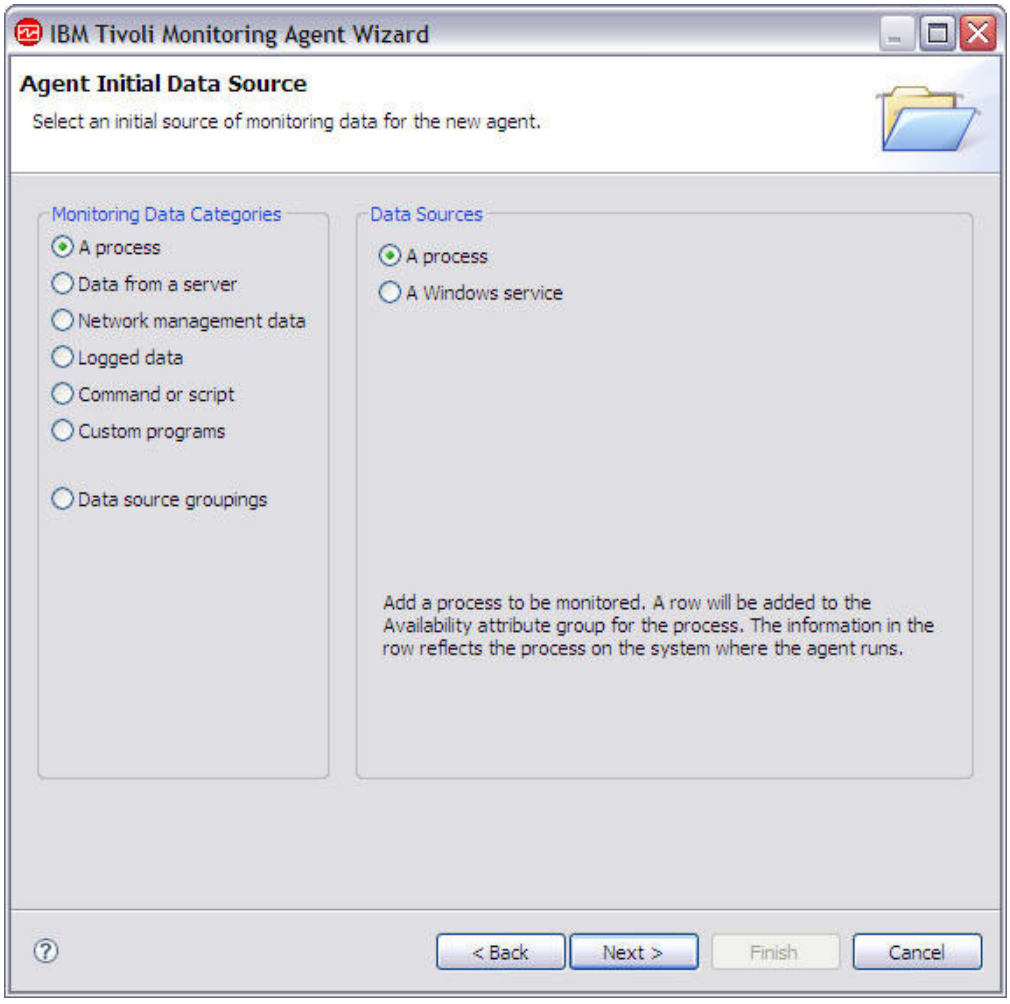


Figure 56. Adding a server process

2. In the **Data Sources** area, click **A process**.
3. Click **Next**.
4. On the Process Monitor page (Figure 57 on page 90), Process information area, provide the display name and process name, which you can type manually or obtain by clicking **Browse** to display a list of processes that are currently running on the local system or on a remote system. (Figure 58 on page 91).

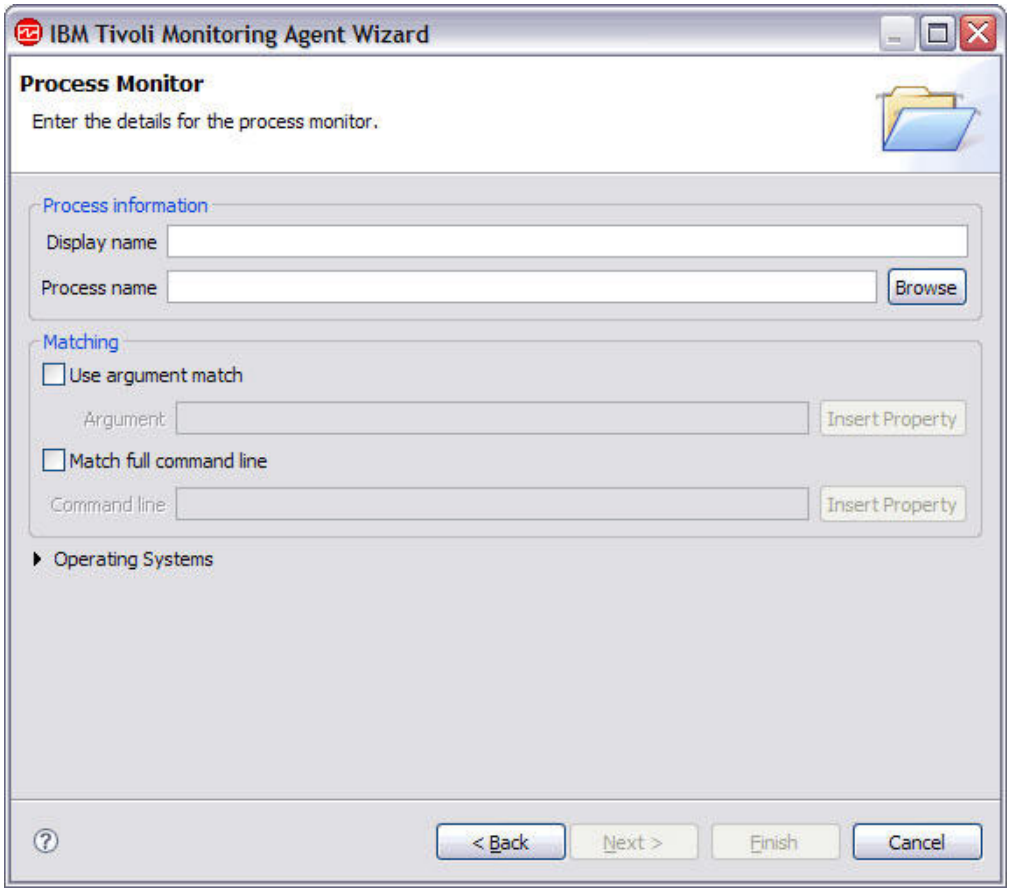


Figure 57. Process Monitor page

Table 10 contains information about the fields on the Process Monitor page.

Table 10. Fields on the Process Monitor page

Field name	Description	Acceptable values
Display name	Descriptive name for the component of the application implemented by the process as it is displayed 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
Argument	Arguments on which to match	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 full command line used to start the process	String
Operating systems	Select the operating systems on which this process runs	Any selection

5. If you clicked **Browse**, the Process Browser window is displayed (Figure 58). This window initially contains detailed information about each process on the Agent Builder system, including the ID, the process name, and the full command line for the process.

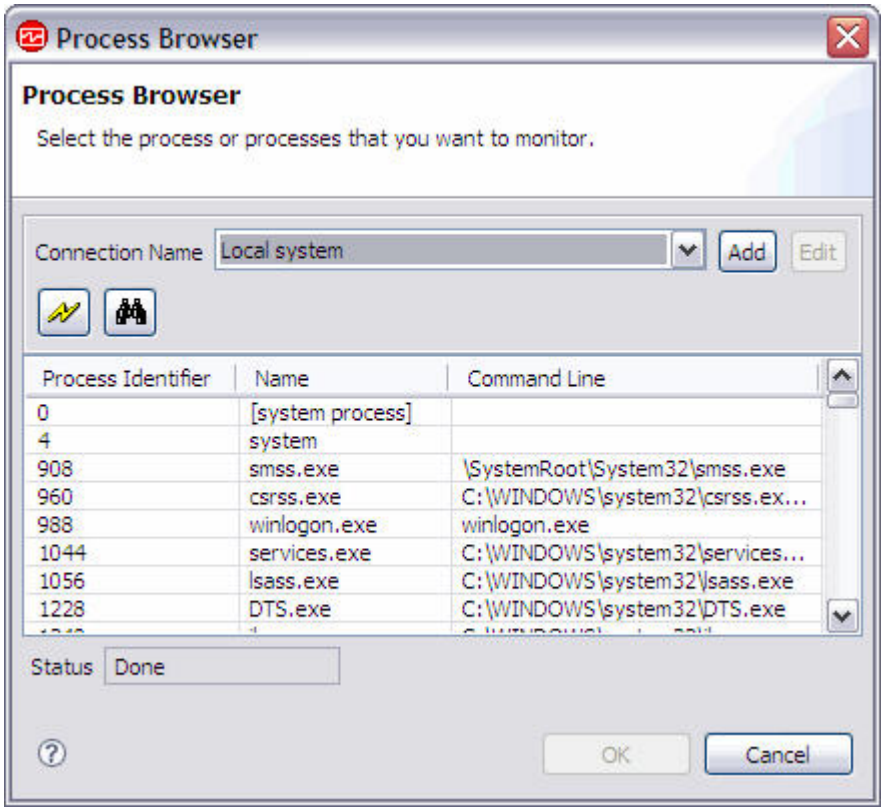


Figure 58. Browse the list of currently running processes

Select one or more processes or perform one or more of the following actions to work with the list in the Process Browser window:

- To sort the list of processes, click the column heading.
- To refresh the information in the window, click the **Refresh** (lightening bolt) icon.
- To search for specific processes, click the **Search** (binoculars) icon (Figure 59 on page 92).

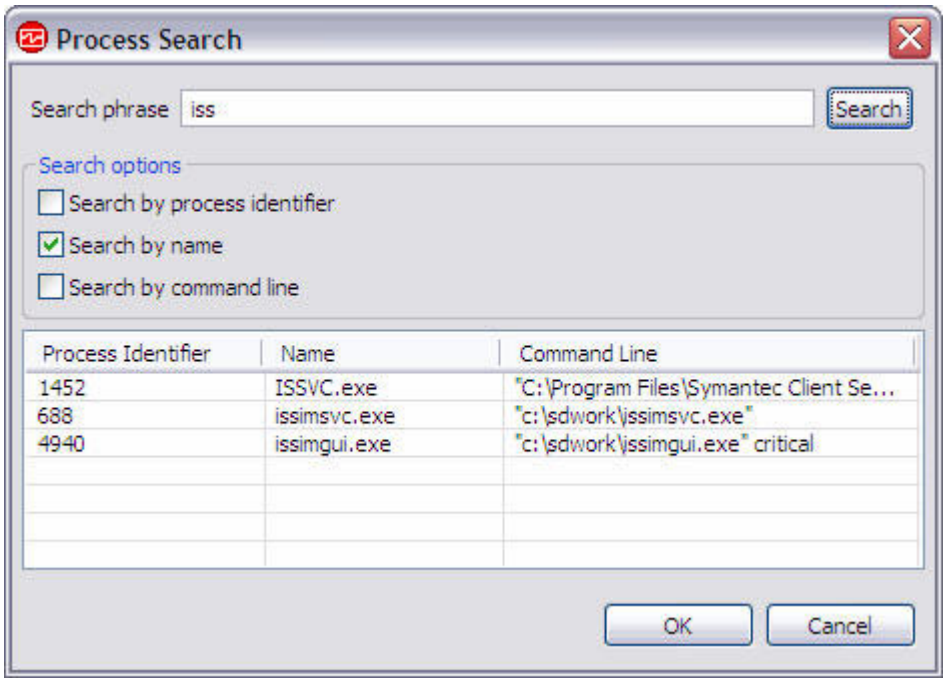


Figure 59. Search for a process

- 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. See "Defining connections for process browsing" on page 94 for more information. 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 displayed in the **Process name** field on the Process Monitor page (Figure 60 on page 93).

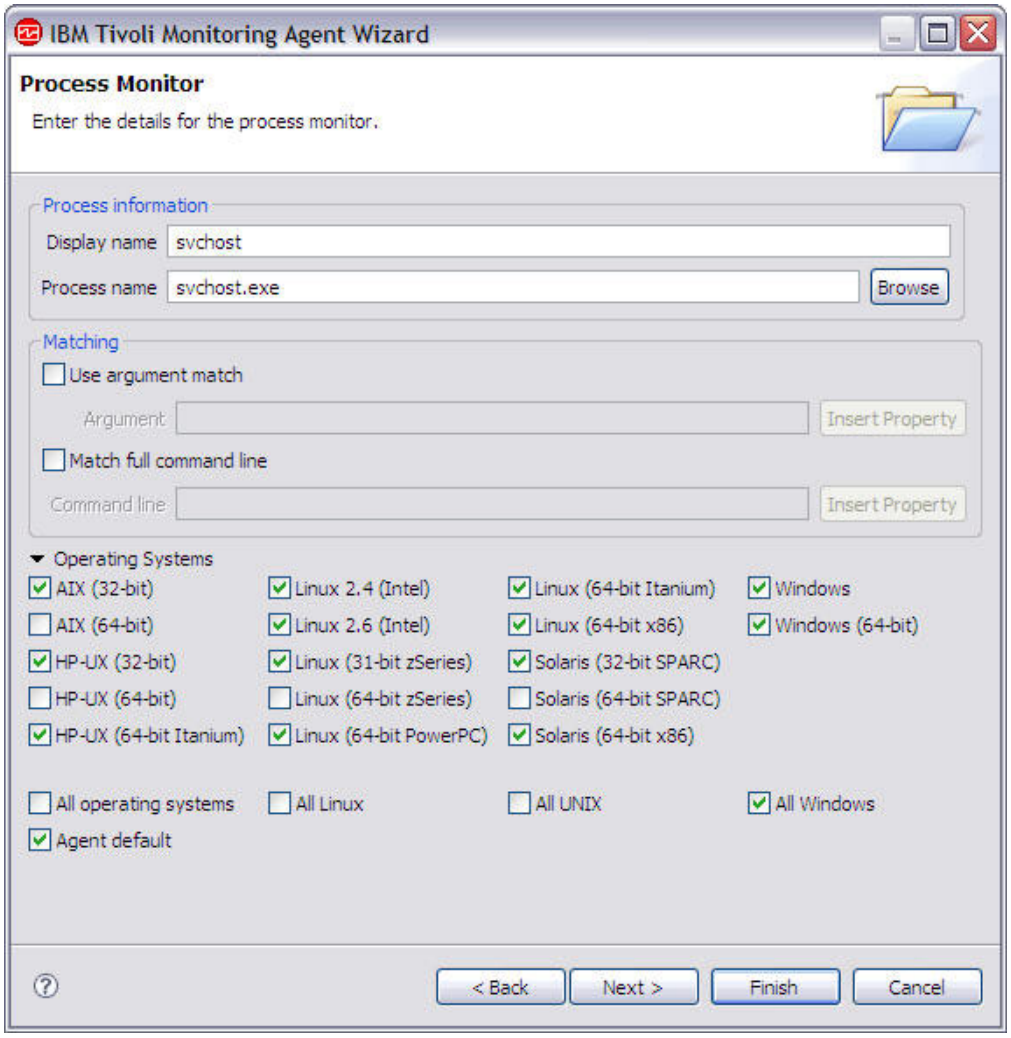


Figure 60. Process Monitor page example

6. Complete the Process Monitor page using the information in Table 10 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 additional process monitors with the same display name to cover the other operating systems. Add the process monitors one at a time, ensuring that the display name is the same for each monitor, but that the process name can be found on the operating systems that are checked.

7. Do one of the following steps:
   a. If you are using the New Agent Wizard, click **Next**.

      —OR—

   b. Click **Finish** to save the data source and open the Agent Editor.

Defining connections for process browsing

In addition to selecting processes from the system where Agent Builder is running, you can select processes from other systems for which you have credentials or that are being monitored by a Tivoli Monitoring operating system agent. You do this by defining a connection to the remote system.

To define a connection, click **Add** in the Process Browser window Figure 58 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 61.

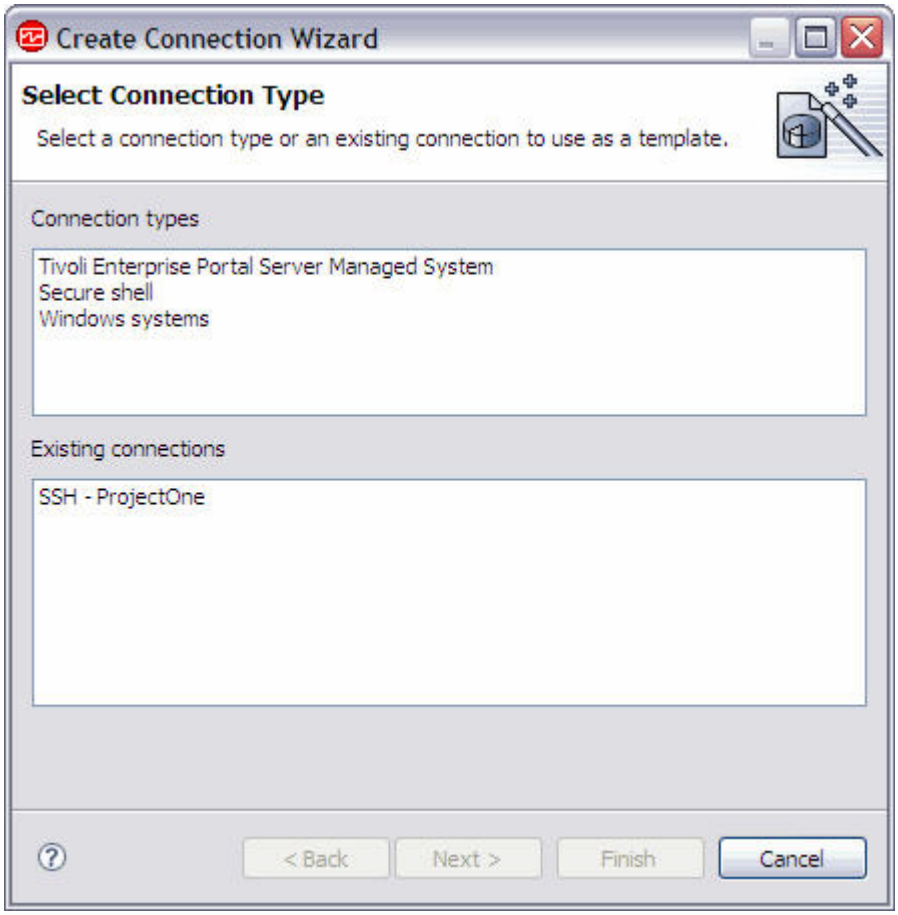


Figure 61. Select a connection type or connection template for process browsing

To add a Managed System connection, you require a Tivoli Enterprise Server host name, Tivoli Monitoring user name and password, and 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.

When you have added 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 have not been saved (for example, the password), the **Connection Properties** window for that connection opens and you can 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. Enter your user name and password, and then click the **Refresh** (lightening bolt) icon to connect before you select the managed system Figure 62.

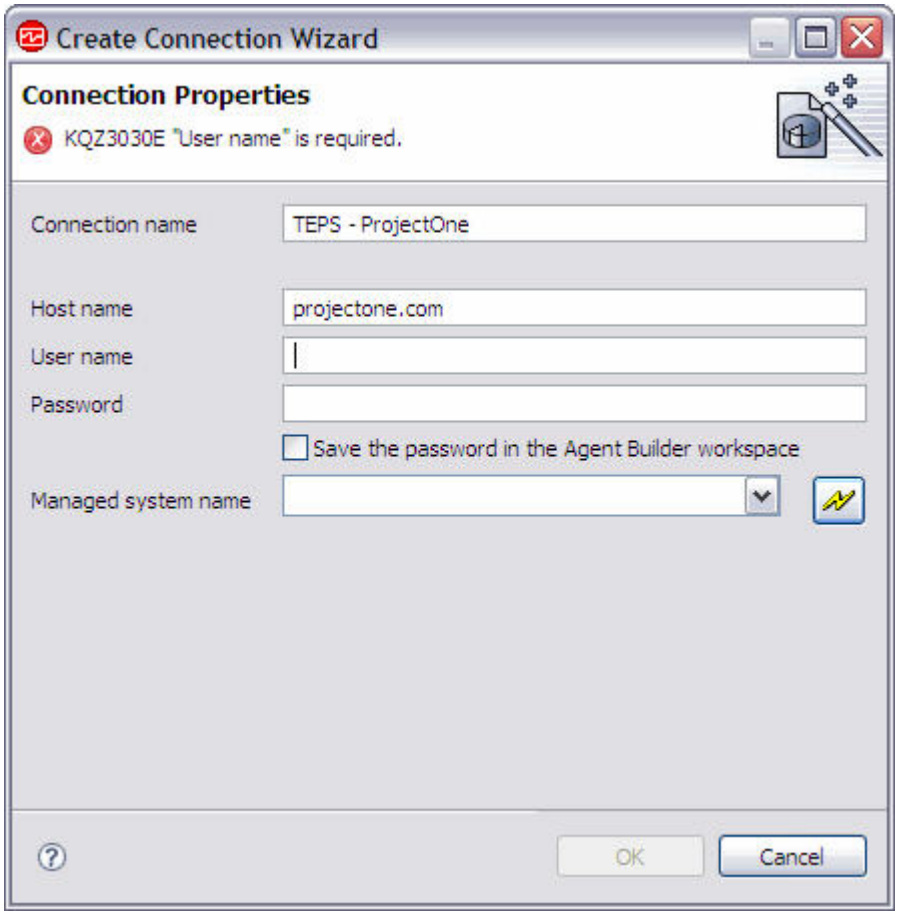


Figure 62. Edit connection properties for Tivoli Enterprise Portal Server Managed System

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

Use the following procedure to monitor a Windows service in your agent:

1. On the Agent Initial Data Source page (Figure 63) or the Data Source Location page, click **A process** in the **Monitoring Data Categories** area.

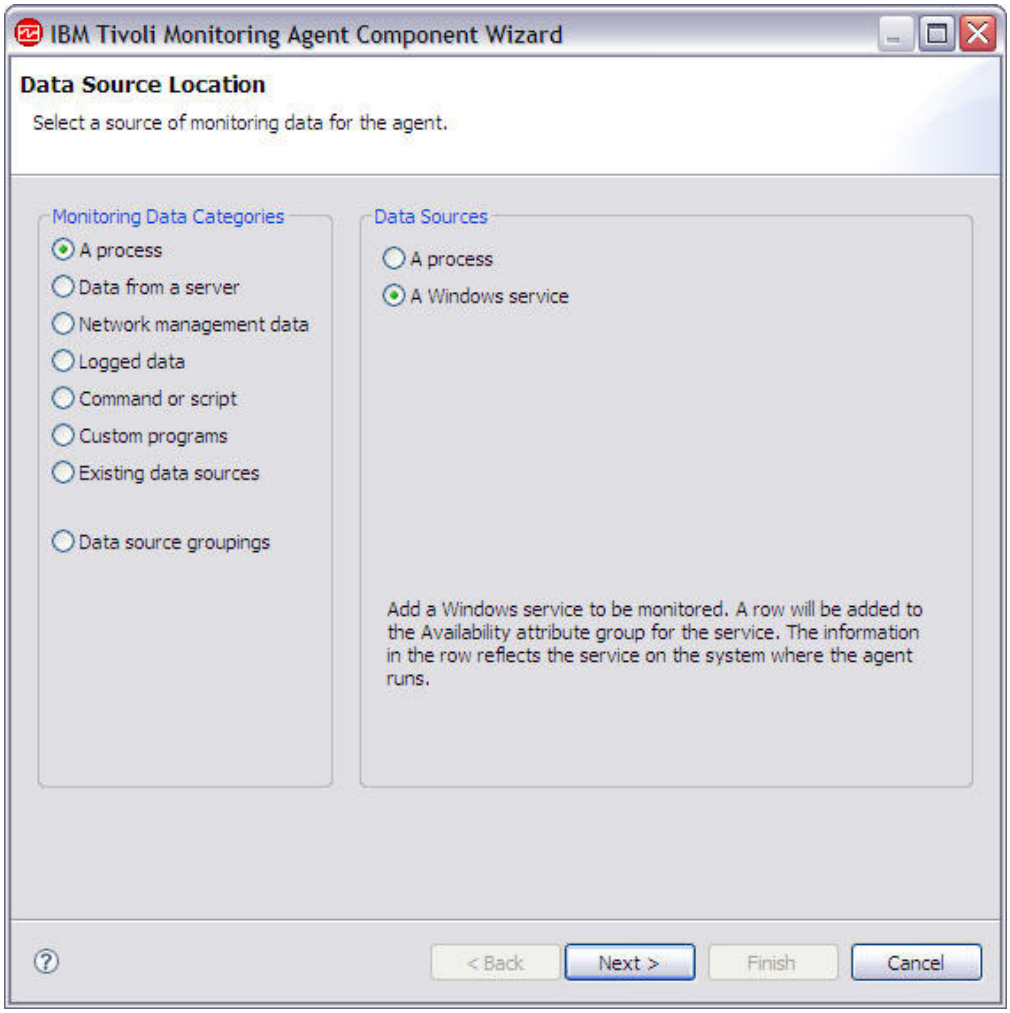


Figure 63. Adding a Windows service monitor

2. In the **Data Sources** area, click **A Windows service**.
3. Click **Next**.
4. On the Service Monitor page (Figure 64 on page 98), in the **Display name** field, provide a descriptive string for the name of the service application to be displayed in the Tivoli Enterprise Portal. You can type it manually or click **Browse** to display a list of services that are currently running on the local system or on a remote system.

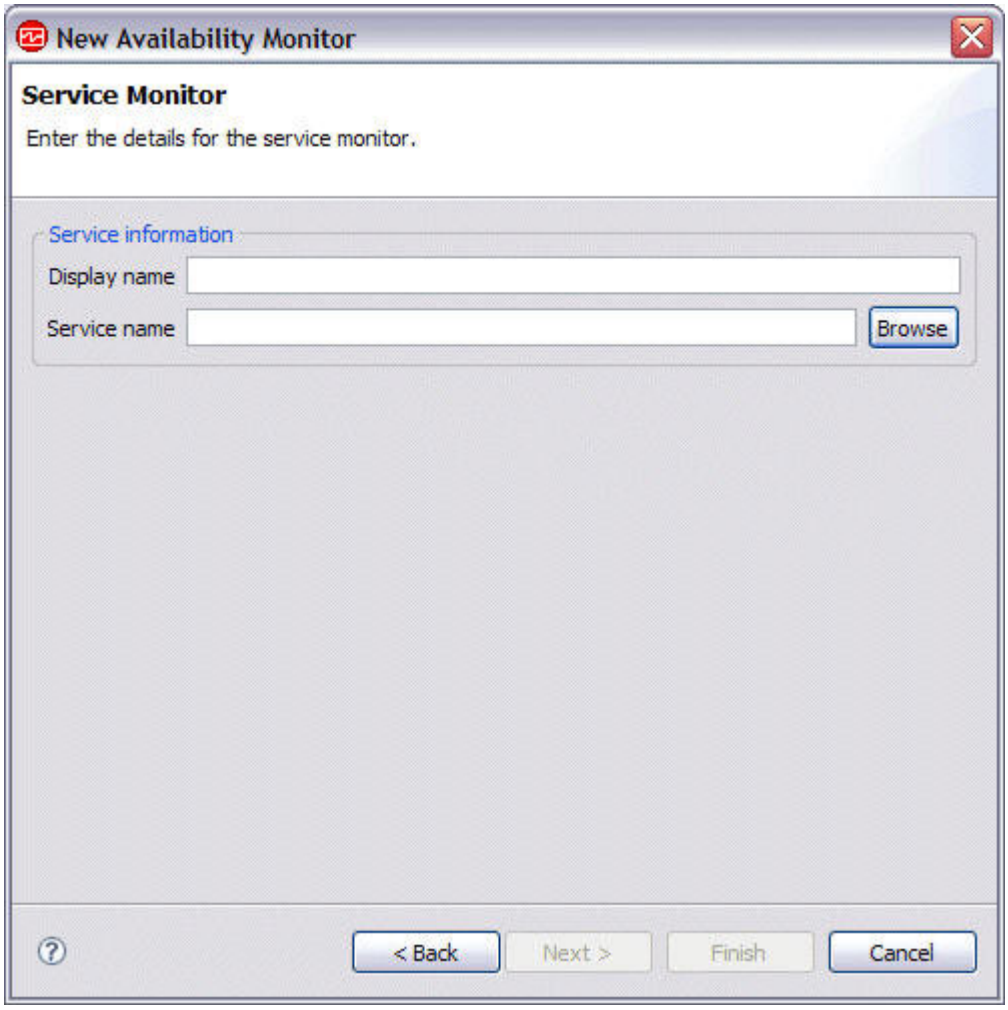


Figure 64. Service Monitor page

5. If you clicked **Browse**, the Service Browser window is displayed (Figure 65 on page 99). This window initially contains detailed information about each service on the Agent Builder system, including the ID, the process name, and the full command line for the process.

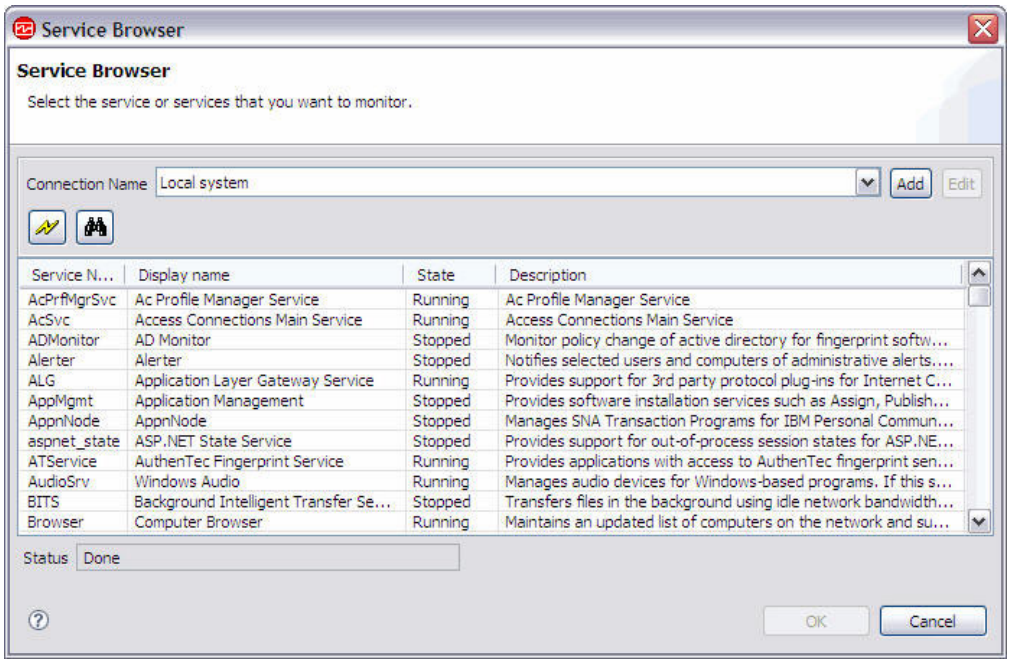


Figure 65. Browse the list of all services defined in the system

Note: There are no local services 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 browsing through the Tivoli Enterprise Portal Server or from a UNIX or Linux system.

Select one or more services or perform one or more of the following actions 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 display the Service Search window.

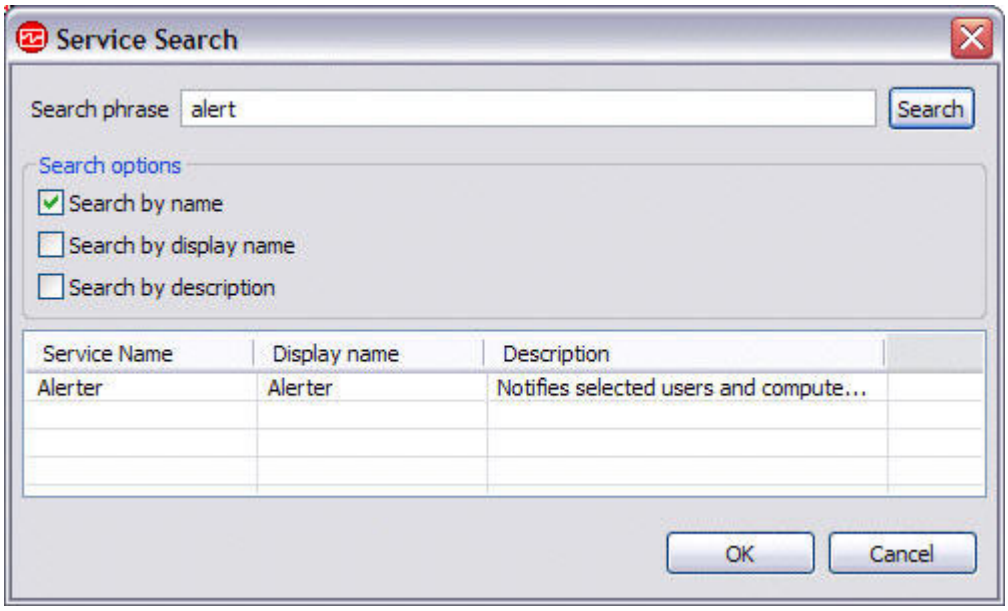


Figure 66. Search for a service

- 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. See "Defining connections for service browsing" on page 101, for more information. 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 displayed in the **Service name** field on the Service Monitor page (Figure 67 on page 101).

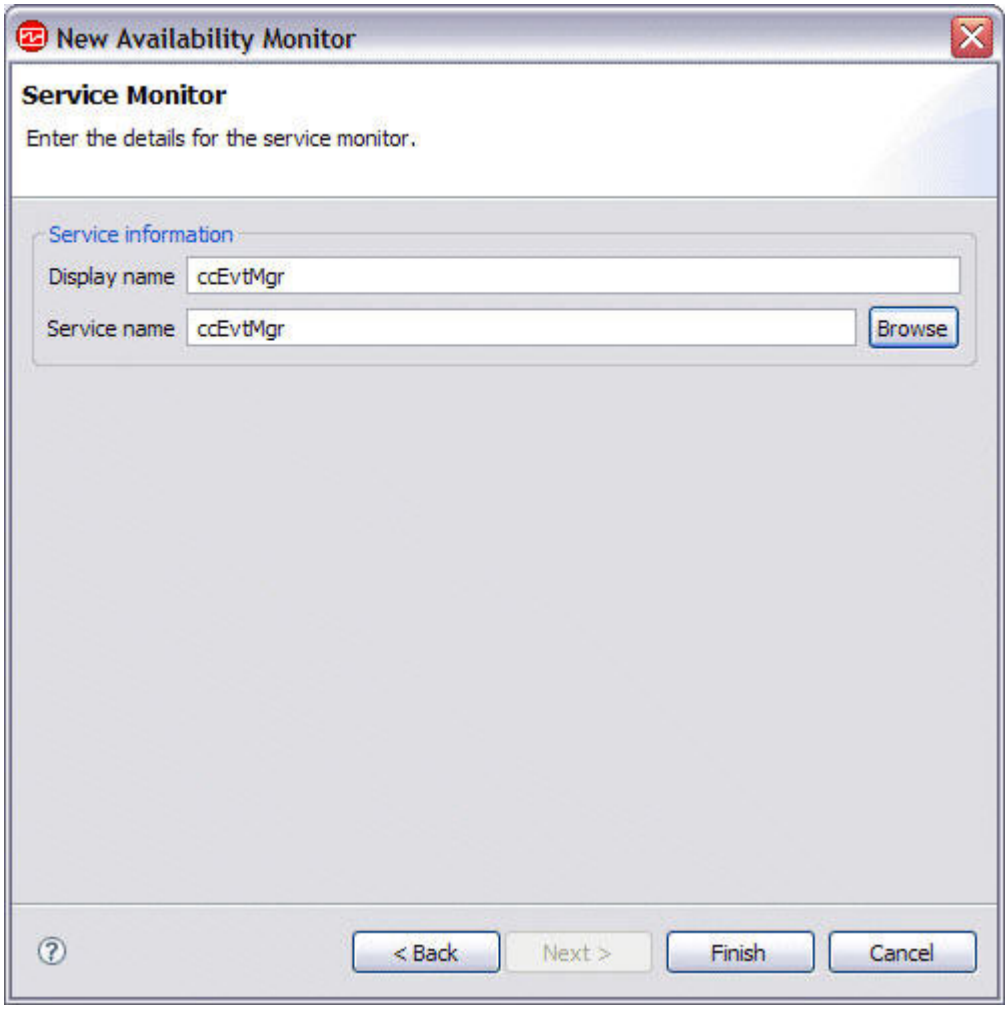


Figure 67. Service information

6. Do one of the following steps:
   a. If you are using the New Agent Wizard, click **Next**.
      —OR—
   b. 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 for which you have credentials or that are being monitored by a Tivoli Monitoring operating system agent. You do this by defining a connection to the remote system

To define a connection, click **Add** in the Service Browser window Figure 65 on page 99. You can select either a connection type (Windows, or Tivoli Enterprise Portal Server Managed System) or select an existing connection to use as a template Figure 68 on page 102.

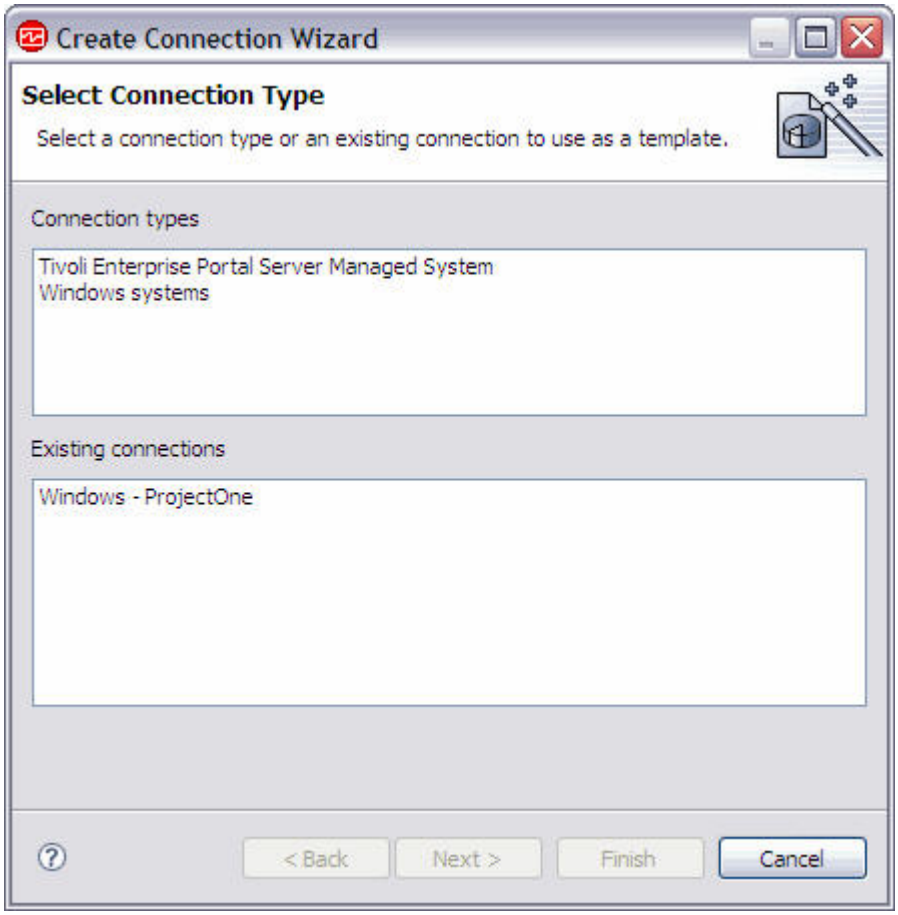


Figure 68. Select a connection type or connection template for service browsing

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 of the remote connection. 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.

When you have added a connection, you can select the connection from the **Connection Name** list in the Service Browser window. If all the fields necessary to make the connection have not been saved (for example, the password), the **Connection Properties** window for that connection opens and you can 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. Enter your user name and password, and then click the **Refresh** (lightening bolt) icon to connect before you select the managed system Figure 69 on page 103.

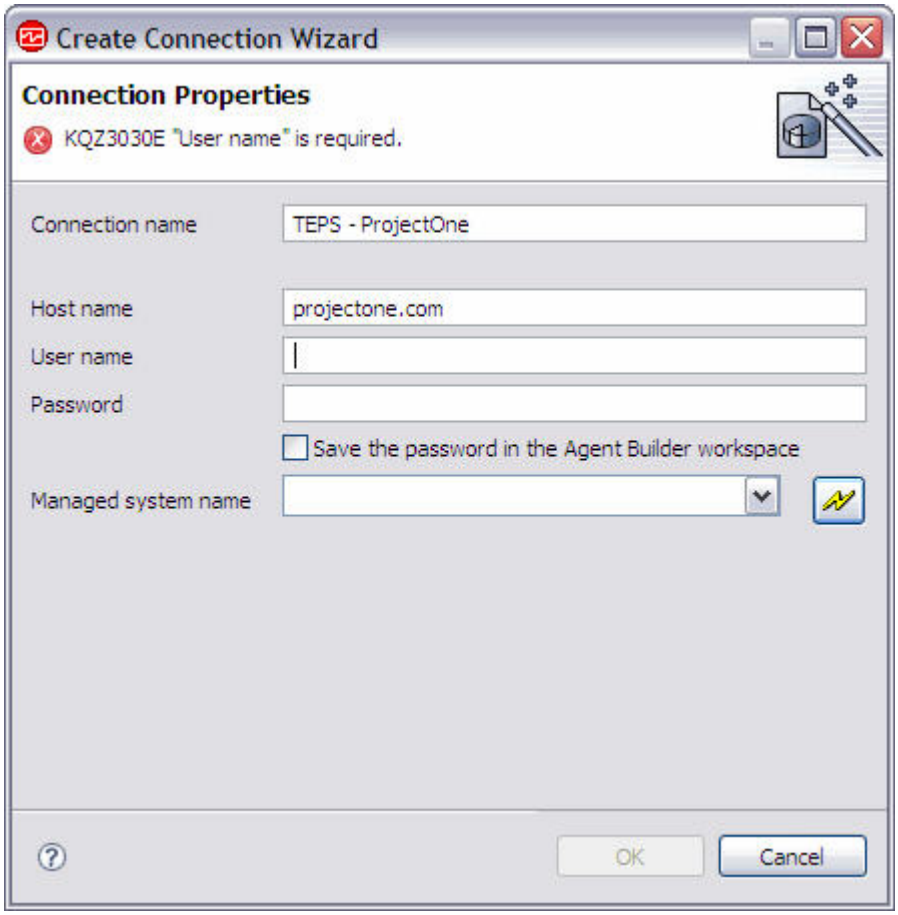


Figure 69. Edit connection properties for Tivoli Enterprise Portal Server Managed System

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 10. Monitoring data from Windows Management Instrumentation (WMI)

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

Use the following procedure to add WMI data:

1. On the Agent Initial Data Source page (Figure 70) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

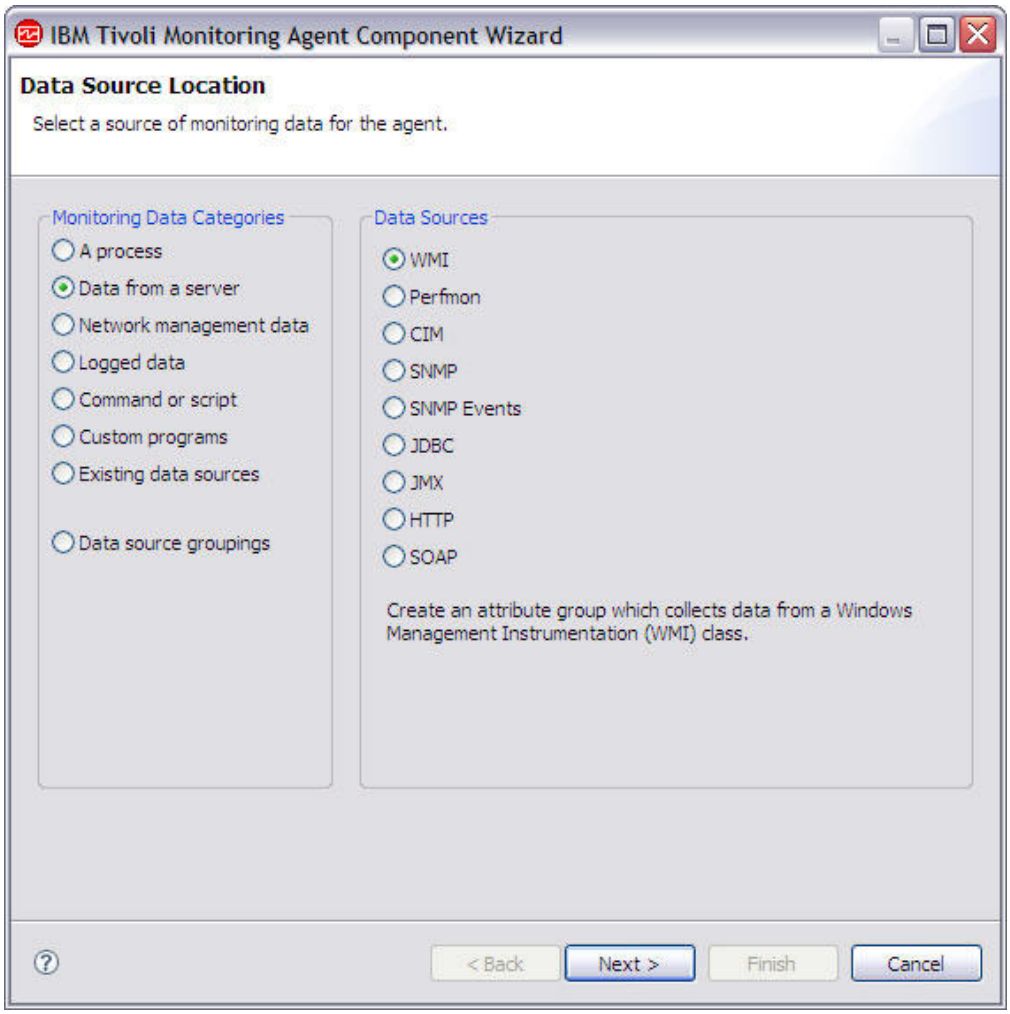


Figure 70. Adding WMI data

2. In the **Data Sources** area, click **WMI**.
3. Click **Next**.
4. On the Windows Management Instrumentation (WMI) Information page (Figure 71 on page 106), do one of the following steps:

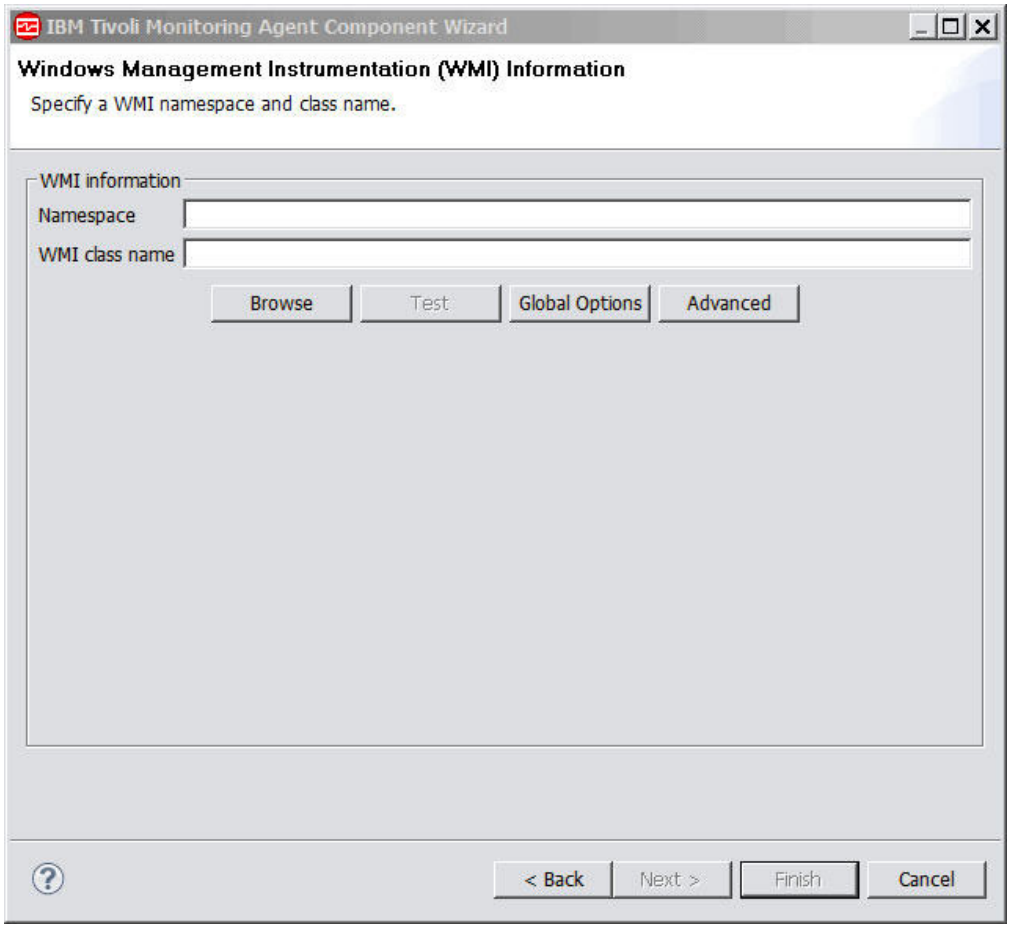


Figure 71. WMI Information page

a. Type a name for the WMI namespace and a name for the WMI class name in the fields.

   —OR—

b. Click **Browse** to see all of the WMI classes on the system (Figure 72 on page 107).

   To browse a remote system, select a system from the drop-down list (if one has been defined) or click **Add** to add the host name of a Windows system. Provide an Administrator ID and password. 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 only browse Windows systems. For instance, you cannot add the host name of a Linux or Solaris system to do a remote browse.

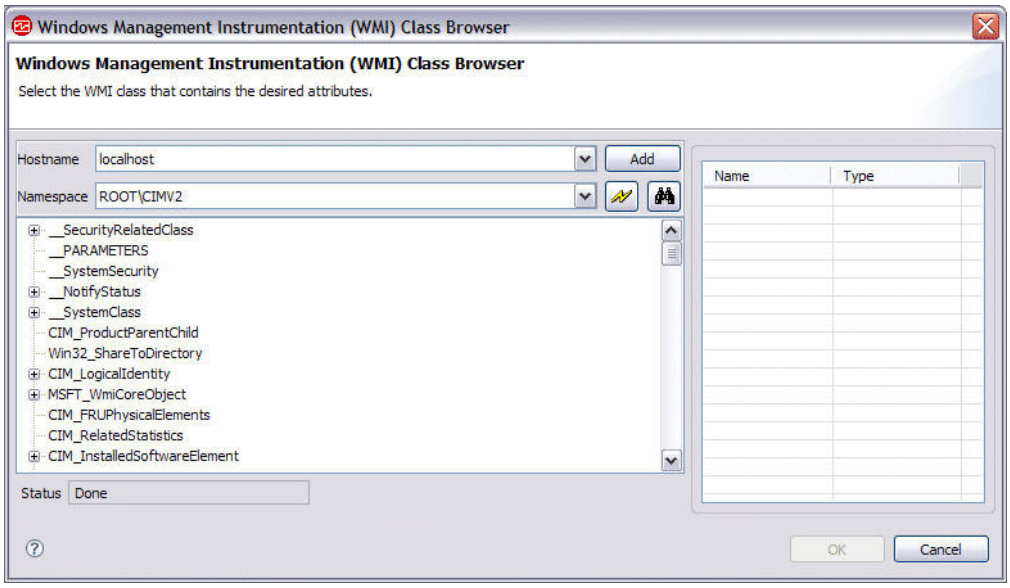


Figure 72. List of classes with their associated attributes

1) Click the plus sign (+) next to a class to expand the class and display the attributes.
2) 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 73). If you find the item for which you are searching, select it and click **OK**.

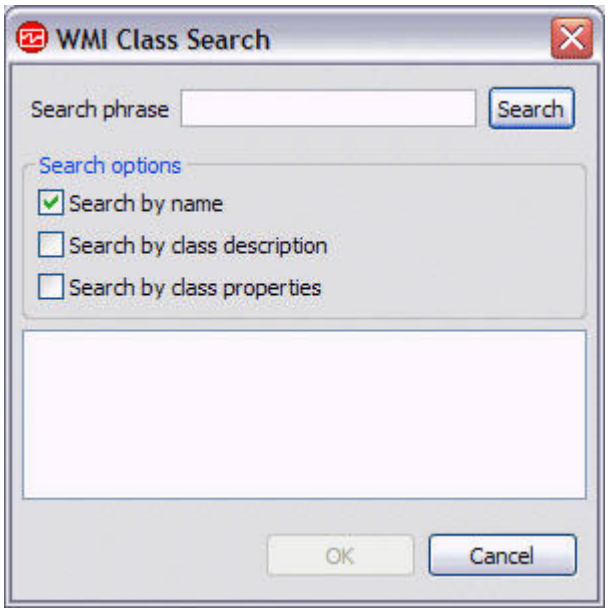


Figure 73. Optionally specifying WMI search options

3) The WMI Information page of the wizard is displayed again, showing the selected WMI class information (Figure 74).

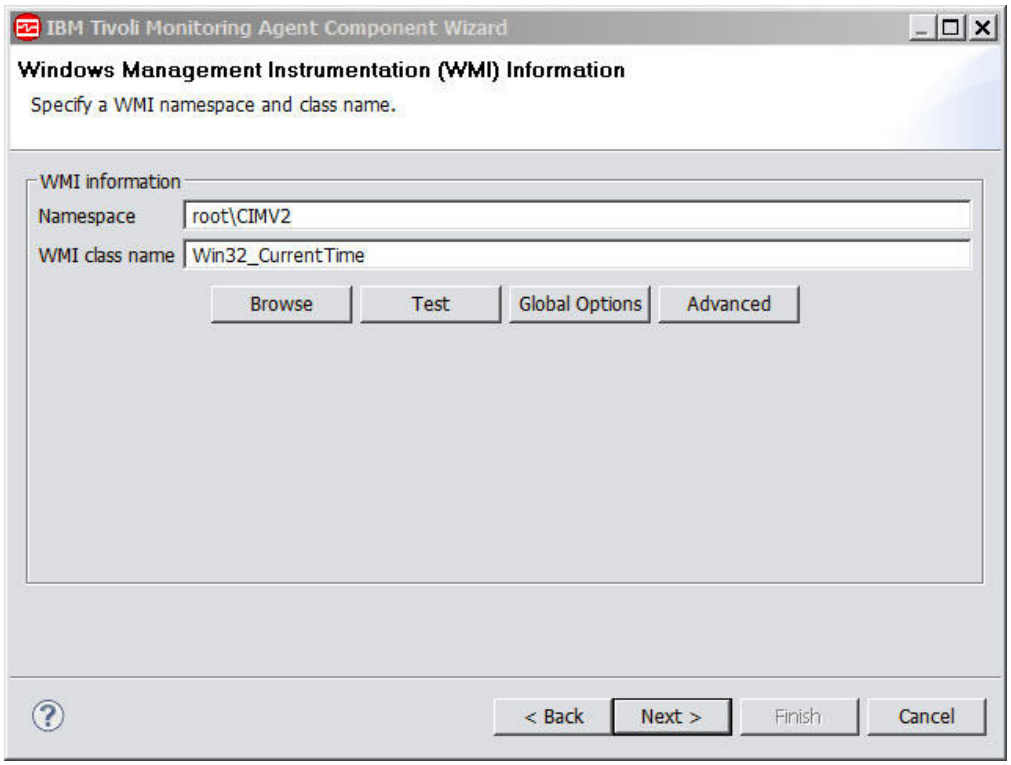


Figure 74. WMI Information page completed

4) *(Optional)* You can test this attribute group by clicking **Test**. For more information about testing, see "Testing" on page 110
5) *(Optional)* You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66
6) Click **Next**.

   Note: If you typed the WMI Namespace and WMI class name manually you will be 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 wish to add more attributes. Click **Finish** to complete.
7) On the Select key attributes page:
   a) Select key attributes or indicate that this data source produces only one data row. See "Selecting key attributes" on page 27 for more information.
8) Do one of the following steps:
   a) If you are using the New Agent Wizard, click **Next**.
      —OR—
   b) Click **Finish** to save the data source and open the Agent Editor.

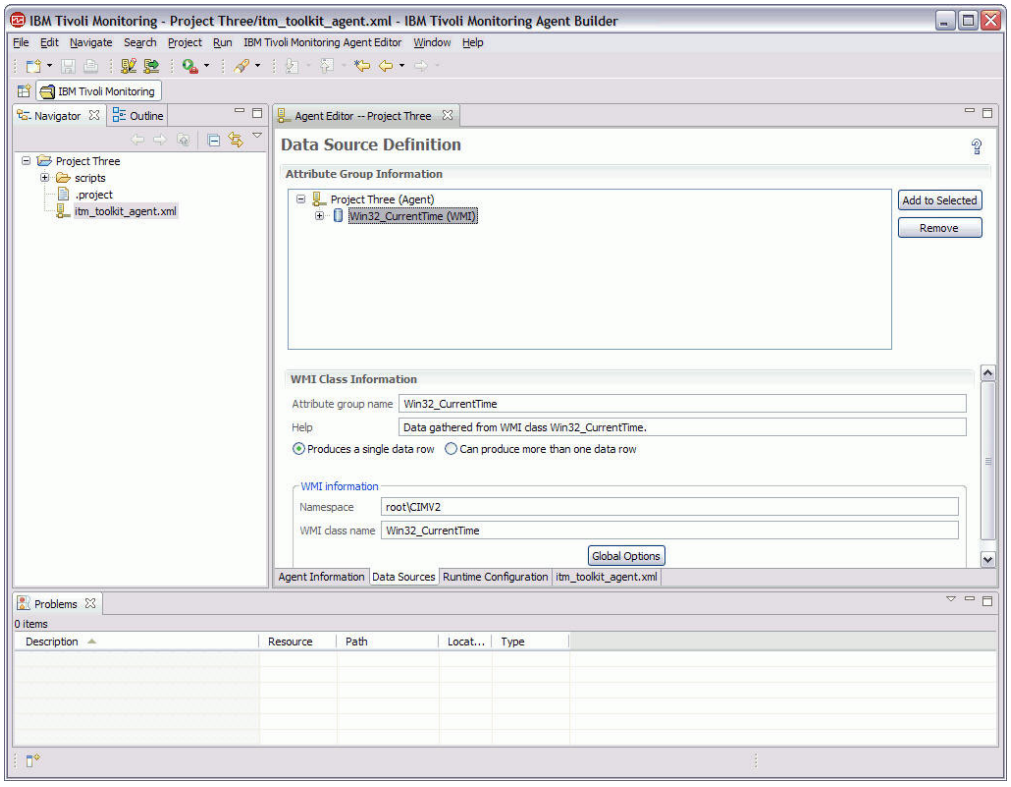


Figure 75. WMI shown on Data Source Definition page in the Agent Editor

9) You can add attributes and supply the information for them. See "Creating attributes" on page 54 for more information.

   In addition to the fields that are applicable to all of the data sources (Table 6 on page 56), the Attribute Information page for the WMI data source has the following field:

   Metric name
   Property name from the class you want to collect

c. If you want to set global options for the data source, click **Global Options** (Figure 76 on page 110).

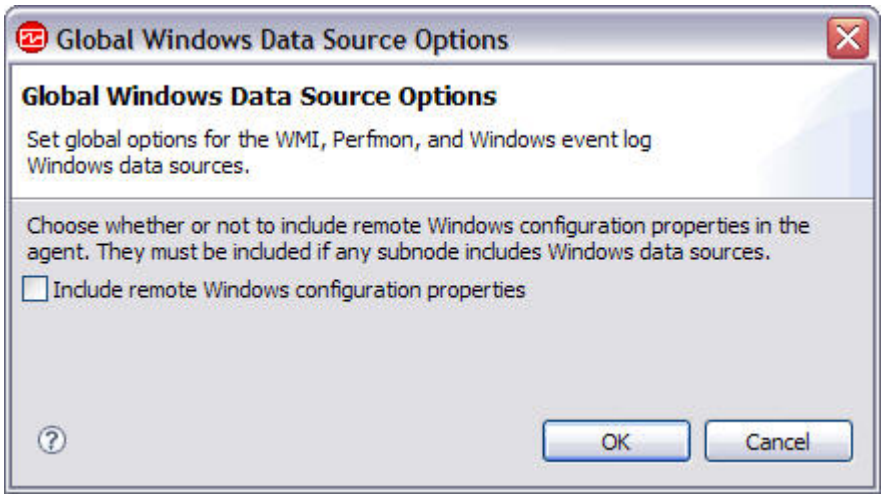


Figure 76. Global Windows Data Source Options window

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 370.

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the WMI Information page (Figure 74 on page 108).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37

After you click **Test** in one of the previous two steps, the WMI Test window Figure 77 on page 111 is displayed.

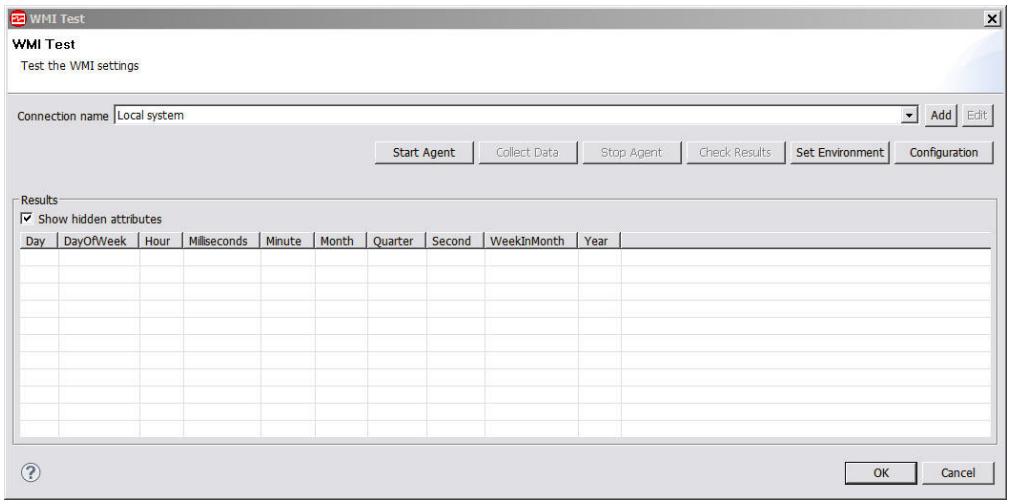


Figure 77. WMI Test window

To test the data source use the following procedure:

1. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see "Configuration prior to testing" on page 374.
2. Click **Start Agent**. A window indicates that the Agent is starting.
3. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries WMI for data.
4. The Test window collects and displays any data in the agent's cache since it was last started.
5. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534
6. The agent can be stopped by clicking **Stop Agent**
7. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 11. Monitoring a Windows Performance Monitor (Perfmon)

Use the following steps to add a Windows Performance Monitor data source:

1. On the Agent Initial Data Source page (Figure 78) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

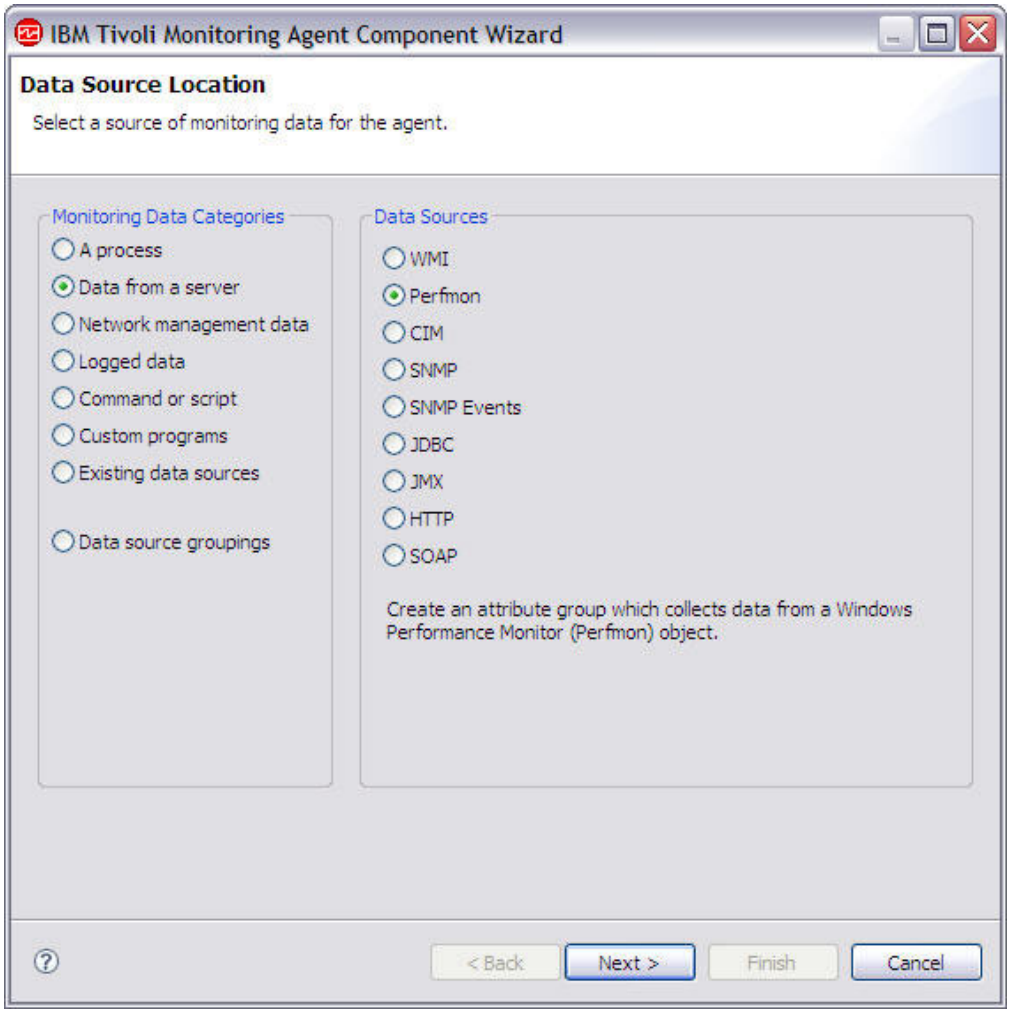


Figure 78. Adding Perfmon data

2. In the **Data Sources** area, click **Perfmon**.
3. Click **Next**.
4. On the Perfmon Information page (Figure 79 on page 114), do one of the following steps:

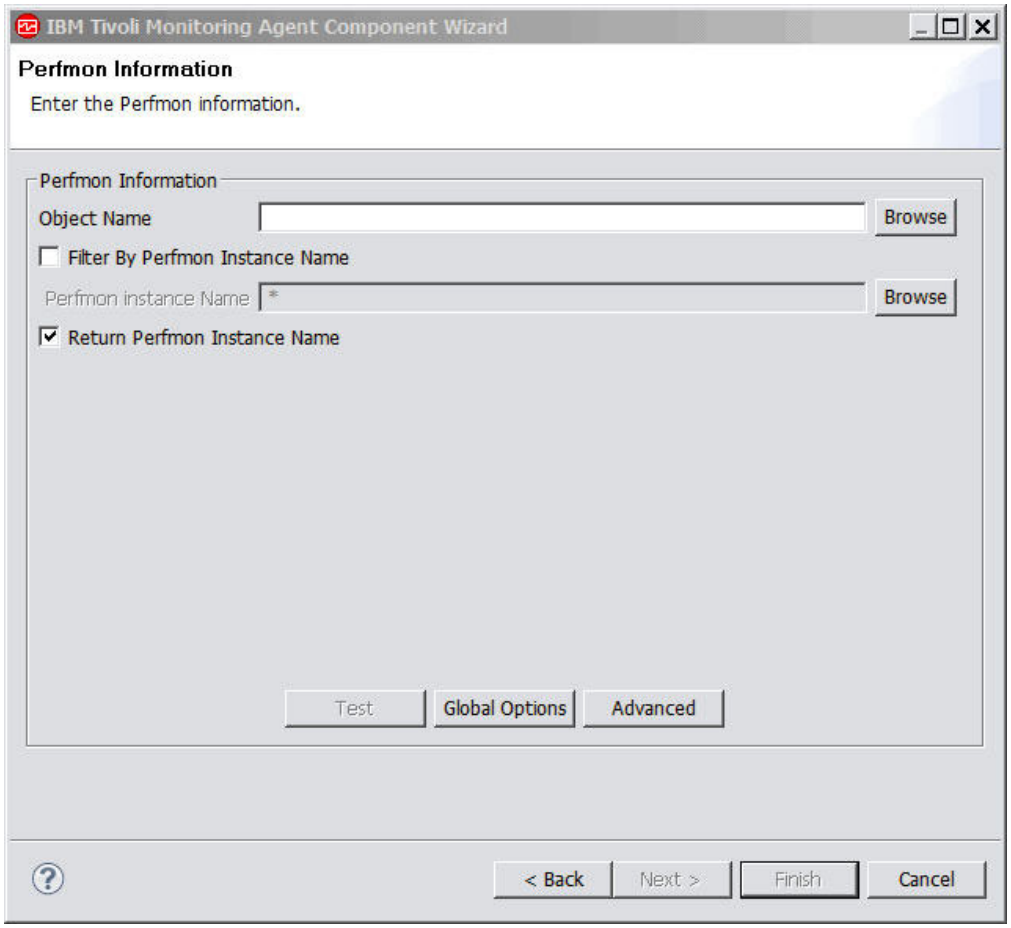


Figure 79. Perfmon Information page

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

   —OR—

b. Click **Browse** to display the list of Perfmon objects.

   Note: If you type the name for the Performance Monitor object, it must be the English name.

   When the Performance Monitor (Perfmon) Object Browser (Figure 80 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 drop-down list (if one has been 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 on an object name, the available counters in that object are displayed in the area on the right side of the window.

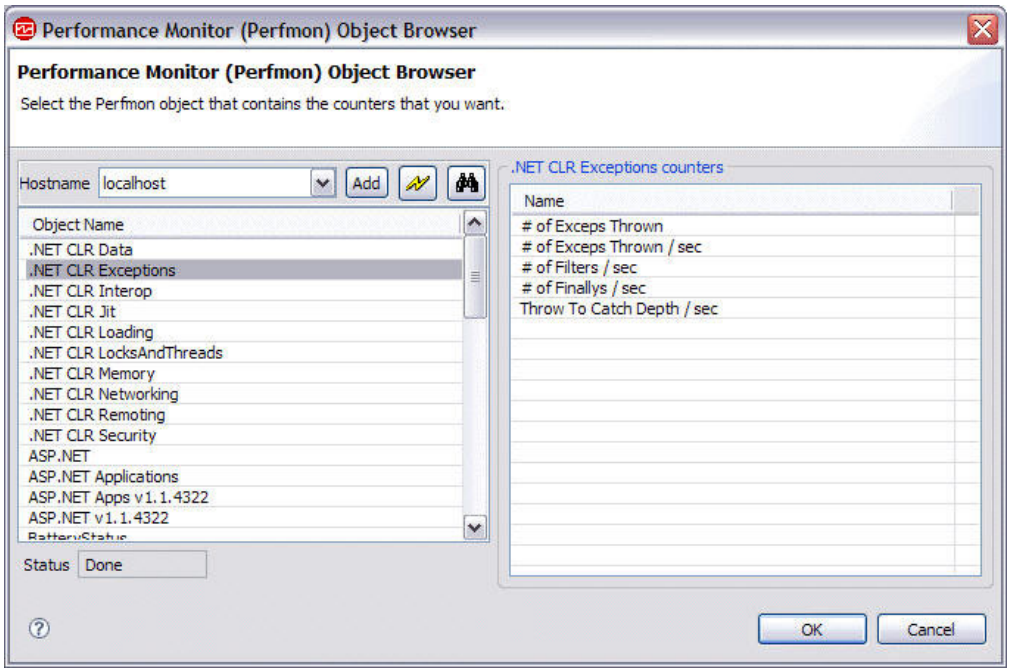


Figure 80. Browsing for Performance Monitor objects

- To sort the 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 display the Performance Monitor Search window (Figure 81). You can search only object names, only counter names, or both. The search operation performs a substring match and is not case-sensitive.

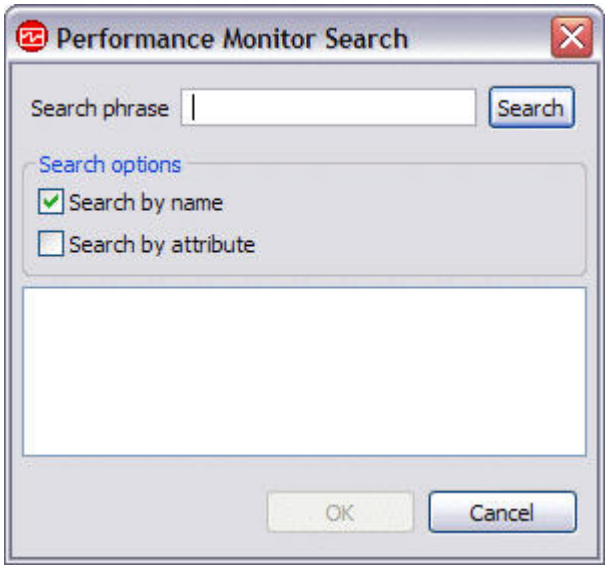


Figure 81. Performance Monitor Search window

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

– The Perfmon Information page is displayed with the name of the selected object in the **Object Name** field.

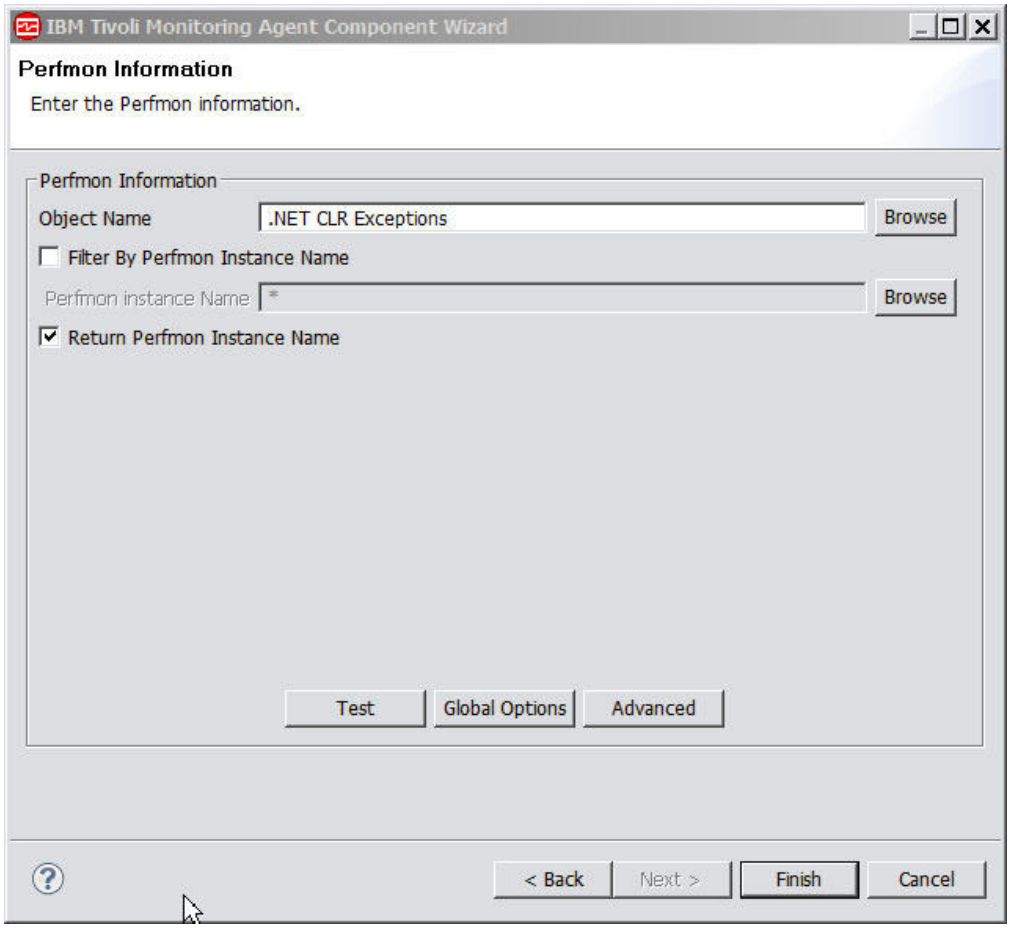


Figure 82. Perfmon Information page with Object Name

c. If you want to set global options for the data source, click **Global Options** (Figure 83).

Figure 83. 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 370.

5. If the Performance Monitor object selected returns multiple instances and you want to filter the results based on the instance name, select the **Filter by Perfmon Instance Name** check box on the Perfmon Information page. In the **Perfmon Instance Name** field, type the name of the instance to be filtered, or click **Browse** to list the instances available.

   To browse a remote system, either select one from the drop-down 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 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 (Figure 79 on page 114). Checking this option adds an additional attribute to the data source that is not displayed 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 is displayed. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. See "Selecting key attributes" on page 27 for more information.
8. *(Optional)* You can test this attribute group by clicking **Test**. For more information about testing, see "Testing" on page 118
9. *(Optional)* You can create a filter to limit the data 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 66

   Note: You can also filter by instance name, see 5

10. Do one of the following steps:
    a. If you are using the New Agent Wizard, click **Next**.

       —OR—

    b. Click **Finish** to save the data source and open the Agent Editor.

       The Agent Editor Data Source Definition page displays a list that contains the object (Figure 84 on page 118) and information about the object.

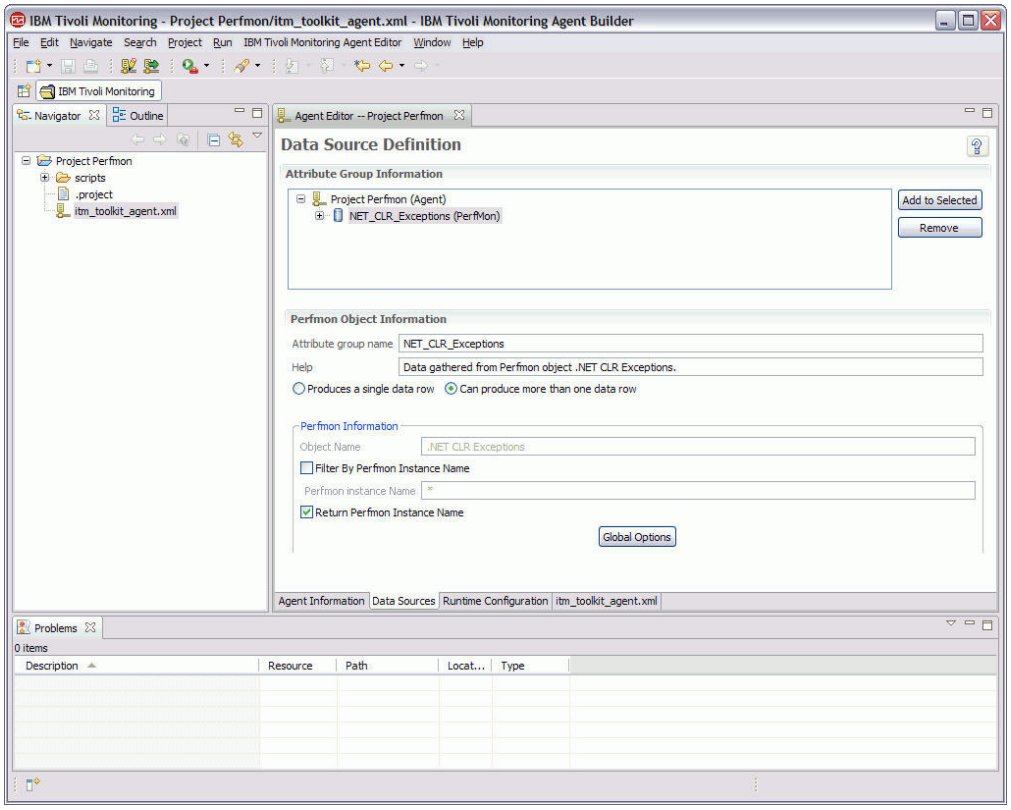


Figure 84. Perfmon Data Source Definition

11. You can add attributes and supply the information for them. See "Creating attributes" on page 54 for more information.

    In addition to the fields that are applicable to all of the data sources (Table 6 on page 56), the Attribute Information page for the Perfmon data source has the following field:

    Metric name
    Name of the counter for the specific object.

For information about Windows remote connection configuration for Perfmon data sources, see "Configuring a Windows remote connection" on page 370.

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the Perfmon Information page (Figure 82 on page 116).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37

After you click **Test** in one of the previous two steps, the Perfmon Test window Figure 85 on page 119 is displayed.

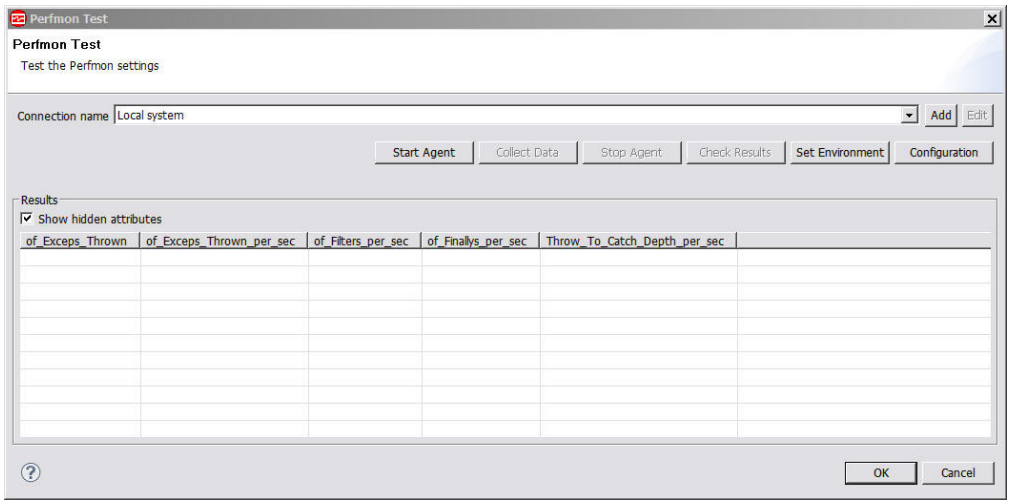


Figure 85. Perfmon Test window

To start and test the data source use the following procedure:

1. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see "Configuration prior to testing" on page 374.
2. Click **Start Agent**. A window indicates that the Agent is starting.
3. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries Performance Monitor for data.

   Note: You may not see useful data for all attributes until you click **Collect Data** a second time. This is because some Performance Monitor attributes return delta values, and a previous value is required to calculate a delta value.
4. The Test window collects and displays any data in the agent's cache since it was last started.
5. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534
6. The agent can be stopped by clicking **Stop Agent**
7. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

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

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

Use the following steps to collect data from a Simple Network Management Protocol:

1. On the Agent Initial Data Source page (Figure 86) 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**.

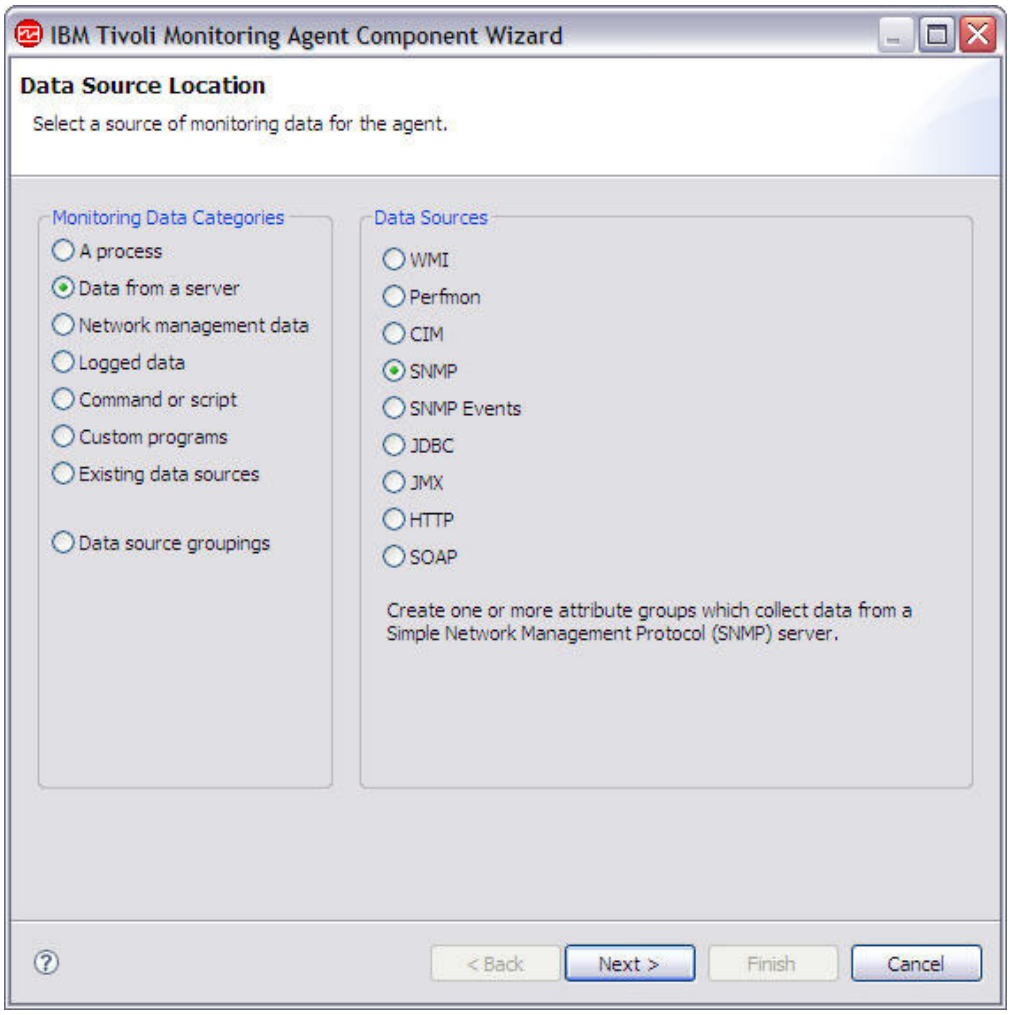


Figure 86. Adding SNMP data

3. Click **Next**.
4. On the Simple Network Management Protocol (SNMP) Information page (Figure 87 on page 122), do one of the following steps:

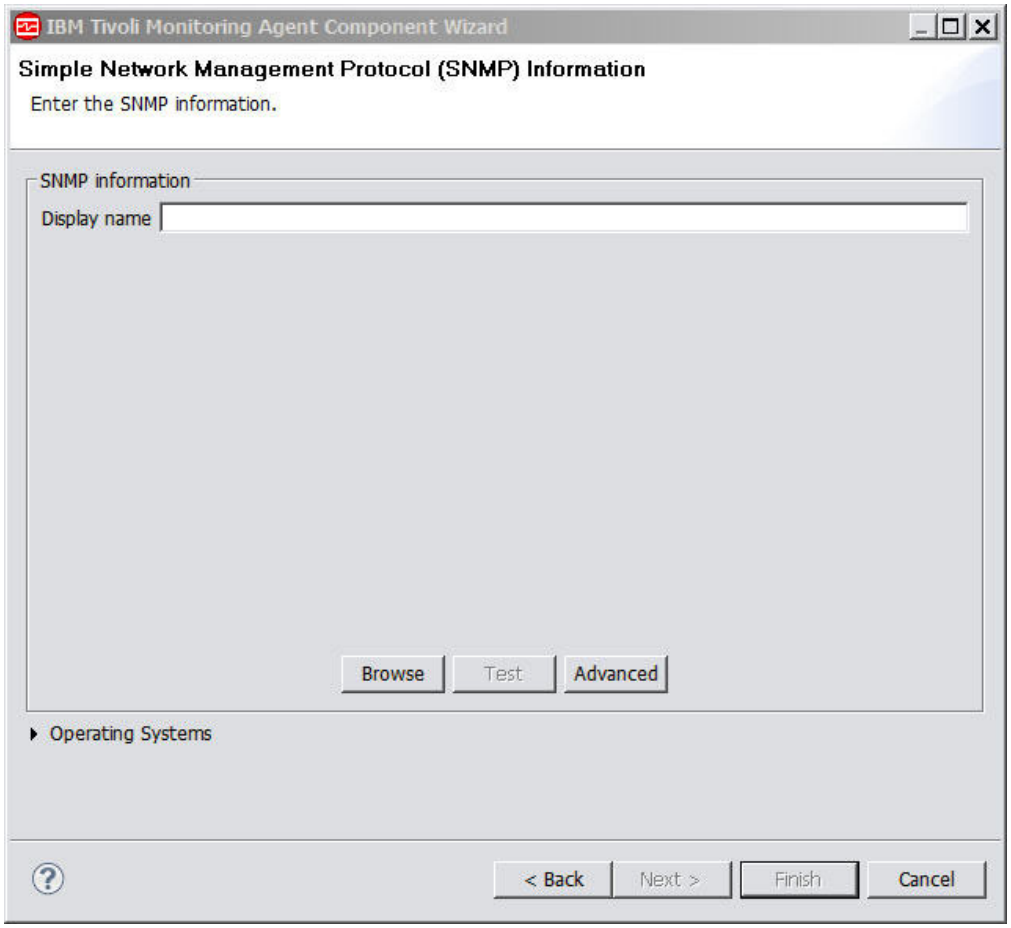


Figure 87. Simple Network Management Protocol Information page

a. Type the display name.
   —OR—
b. Click **Browse** to see all of the objects on the system (Figure 88 on page 123). After you have defined 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 you specified when you initially ran the Agent Builder, see ("Starting the Agent Builder" on page 13).

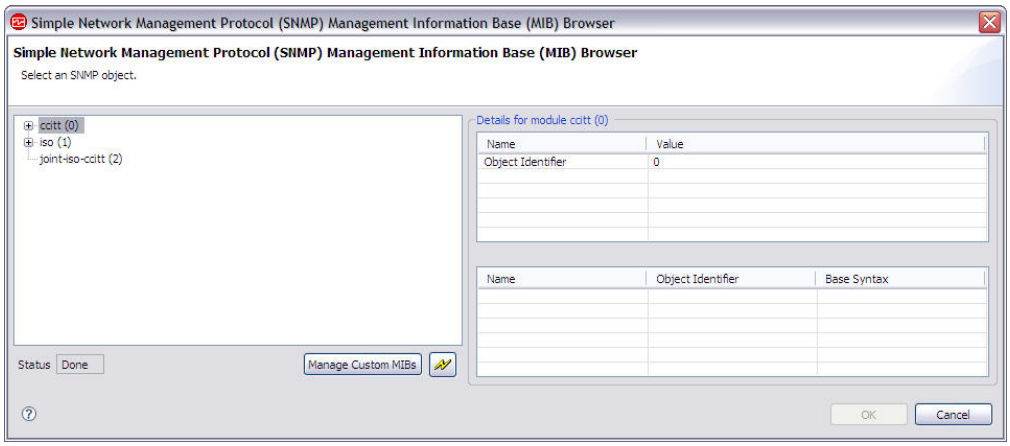


Figure 88. List of objects

c. If the MIB that defines the desired object is not loaded, click the **Manage Custom MIBs** button to open the Manage Custom MIBs dialog. Click **Add** to browse to the MIB file to add. To delete a MIB from the cache, select it and click **Remove**.

   Click **OK** to update the cache. If there were any errors parsing the MIBs, the Manage Custom MIBs dialog remains open to give you the opportunity to add or remove MIBs to eliminate the errors. Clicking the **Cancel** button returns the MIB cache to the state it was in when the dialog was launched.

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

d. Select an object from the list. (Figure 89)

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

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 91 on page 126).

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. Selecting a high-level object results in a large number of data sources being added.

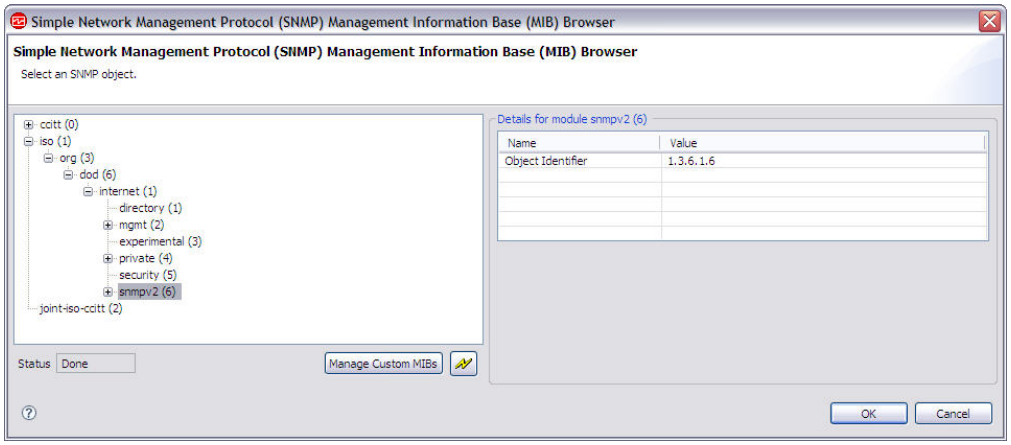


Figure 89. Selecting an object

5. On the Simple Network Management Protocol (SNMP) Information page (Figure 87 on page 122), 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 90 on page 125. For more information about testing, see "Testing" on page 128

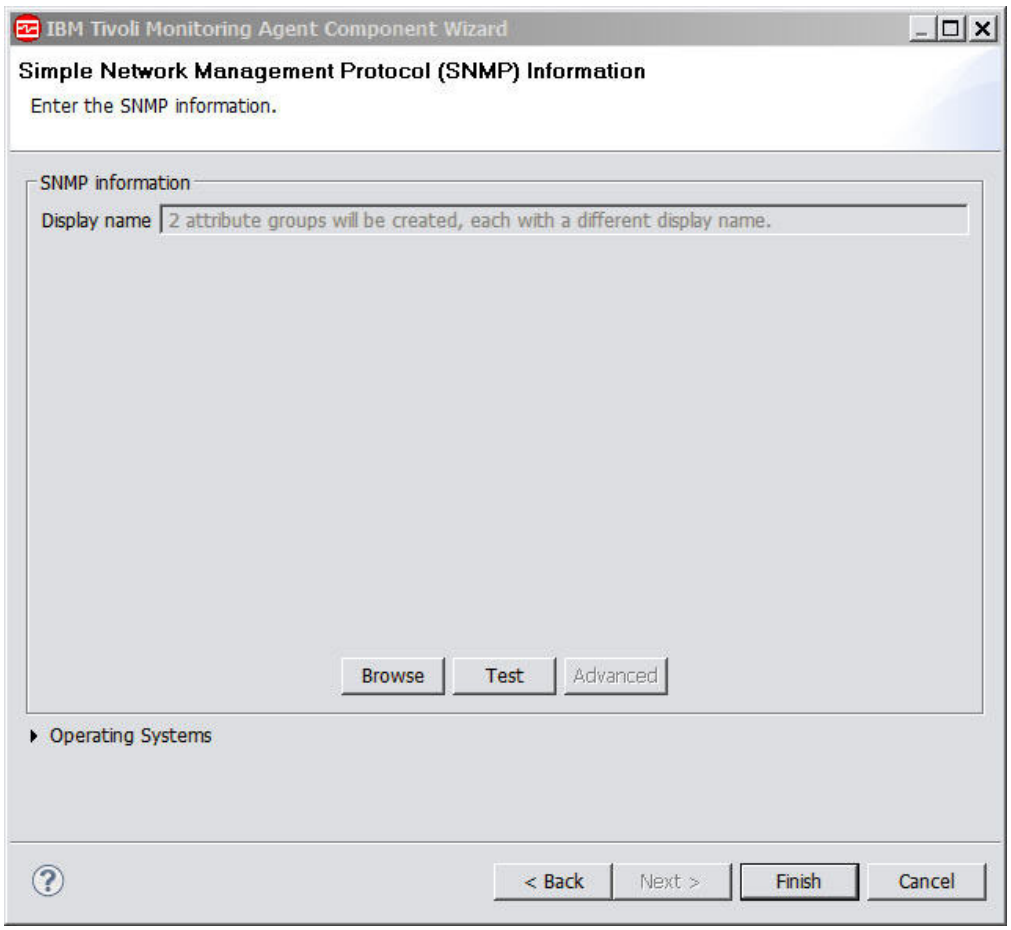


Figure 90. Simple Network Management Protocol Information page

(Optional) You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66

7. Click **Next**.
8. On the Attribute Information page, specify the information for the attribute.
9. Do one of the following steps:
   a. If you are using the New Agent Wizard, click **Next**.
      —OR—
   b. Click **Finish** to save the data source and open the Agent Editor.

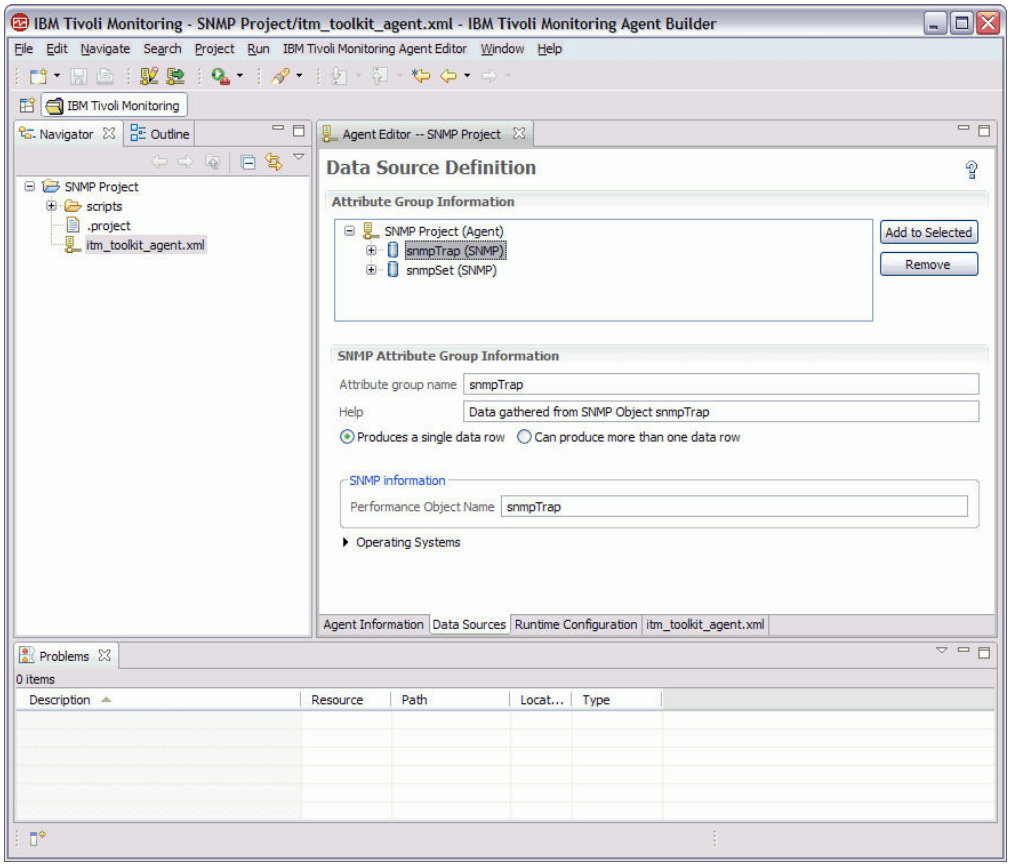


Figure 91. SNMP Data Source Definition

10. If you want to add attributes and supply the information for them, see "Creating attributes" on page 54 for more information.

    In addition to the fields that are applicable to all of the data sources (Table 6 on page 56), the Attribute Information page for the SNMP data source has the following fields:

    Metric name
    Arbitrary string

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

    To enable the Agent Builder to generate 64-bit data types in order to handle the maximum value for 32-bit unsigned MIB properties, see "SNMP MIB Parsing options" on page 127.

MIB errors

MIBs frequently have errors. Click the details for the error in the 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 at once to resolve this problem, or you can incrementally add MIBs until all of the missing definitions are resolved. With Agent Builder you can use any definitions that are completely resolved, so you can

choose to ignore an error that affects only 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. When these MIBs and errors are discovered, IBM Software Support, customers, and business partners add information to the AA&BSM (Application Availability and Business Service Management) Technical Exchange Wiki indicating what MIBs are discovered that have errors and which corrections need to be imported into the Agent Builder. This wiki can be found at:`http://www-03.ibm.com/developerworks/wikis/display/aabsmenbl/Home`

SNMP MIB Parsing options

Use the following steps to set the preferences for SNMP MIB parsing:

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 92 on page 127).

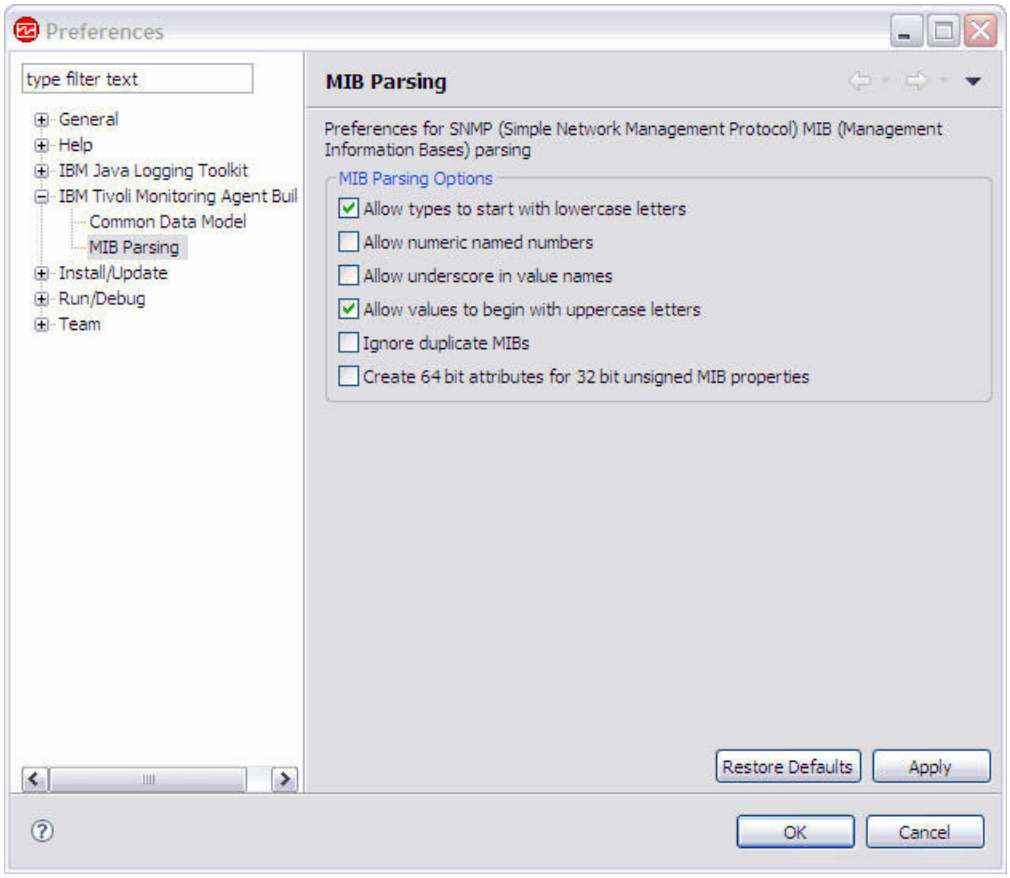


Figure 92. MIB Parsing window

4. The MIB parser used by the Agent Builder uses the grammar 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 lower case 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

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

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the Simple Network Management Protocol Information page (Figure 90 on page 125).

   Note:

   If the SNMP object selected contains more than one attribute group, you will be prompted to select the attribute group to test, on the Select Group window,Figure 93.

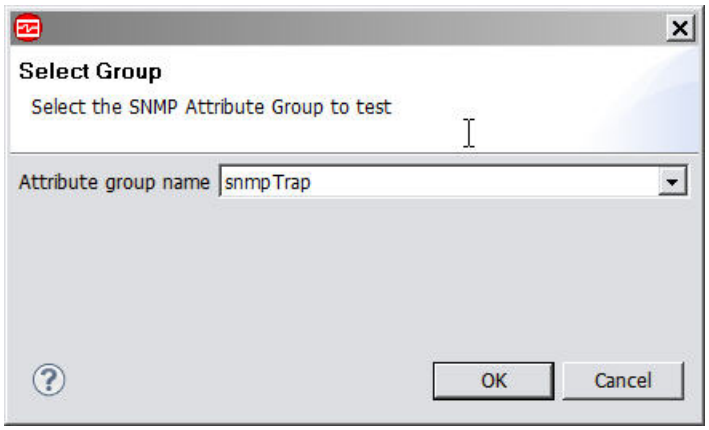


Figure 93. SNMP Attribute Group Test selection

2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test**. For more information about the Agent Editor, see “Tivoli Monitoring Agent Editor” on page 37

After you click **Test** in one of the previous two steps, the SNMP Test settings window Figure 94 is displayed.

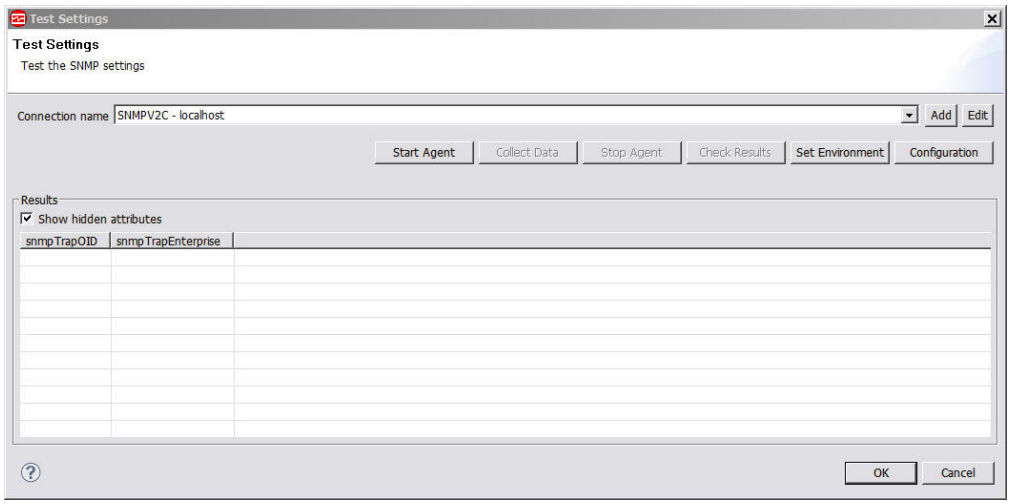


Figure 94. SNMP Test settings window

To start and test the data source use the following procedure

1. Select an existing connection from **Connection name** or click **Add** and you will be prompted to select a connection type, or select an existing connection to use as a template, using the Create Connection Wizard, Figure 95

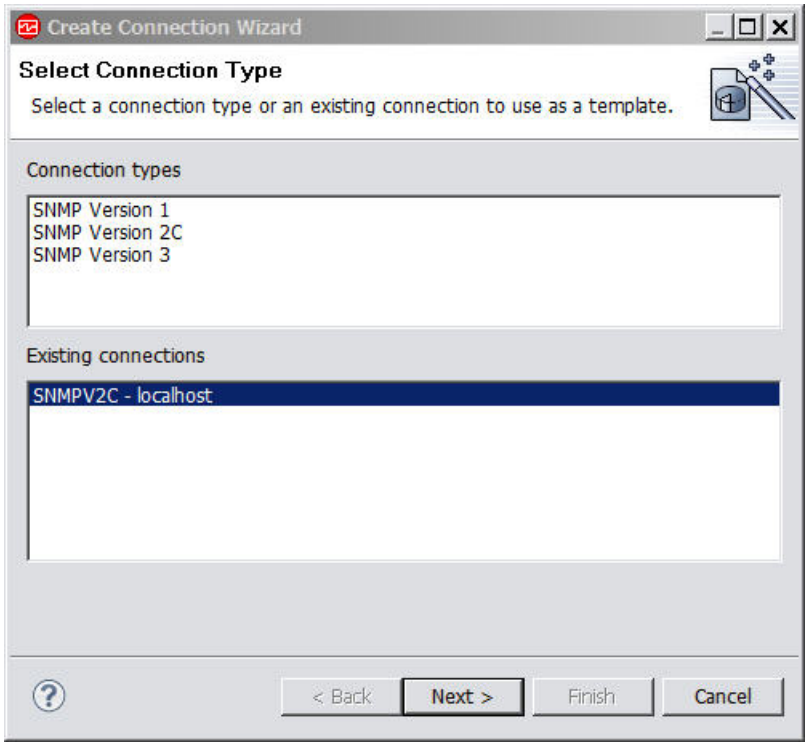


Figure 95. Create Connection Wizard

2. After selecting 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 94.

3. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see “Configuration prior to testing” on page 374.
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 queries the configured SNMP connection for data.
6. The Test Settings window collects and displays any data in the agent's cache since it was last started.
7. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in “Performance Object Status node” on page 534
8. The agent can be stopped by clicking **Stop Agent**
9. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, “Testing and debugging your agent,” on page 373

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

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 received by this provider is passed to Tivoli Monitoring as events.

Use the following steps to receive events from a Simple Network Management Protocol:

1. On the Agent Initial Data Source page or the Data Source Location page (Figure 96), click **Data from a server** in the **Monitoring Data Categories** area.

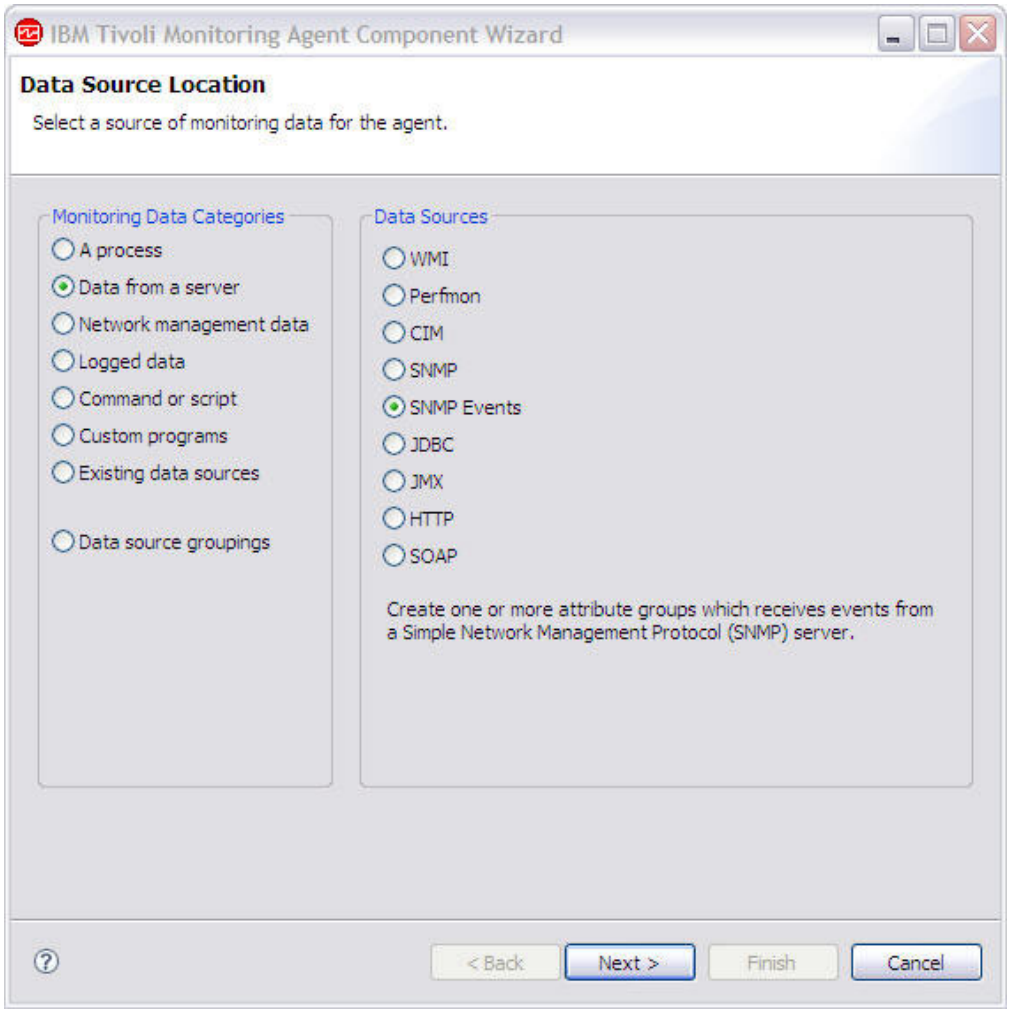


Figure 96. Adding SNMP events

2. In the **Data Sources** area, click **SNMP Events**.
3. Click **Next**.

4. In the Simple Network Management Protocol Event Information window (Figure 97), do one of the following steps:

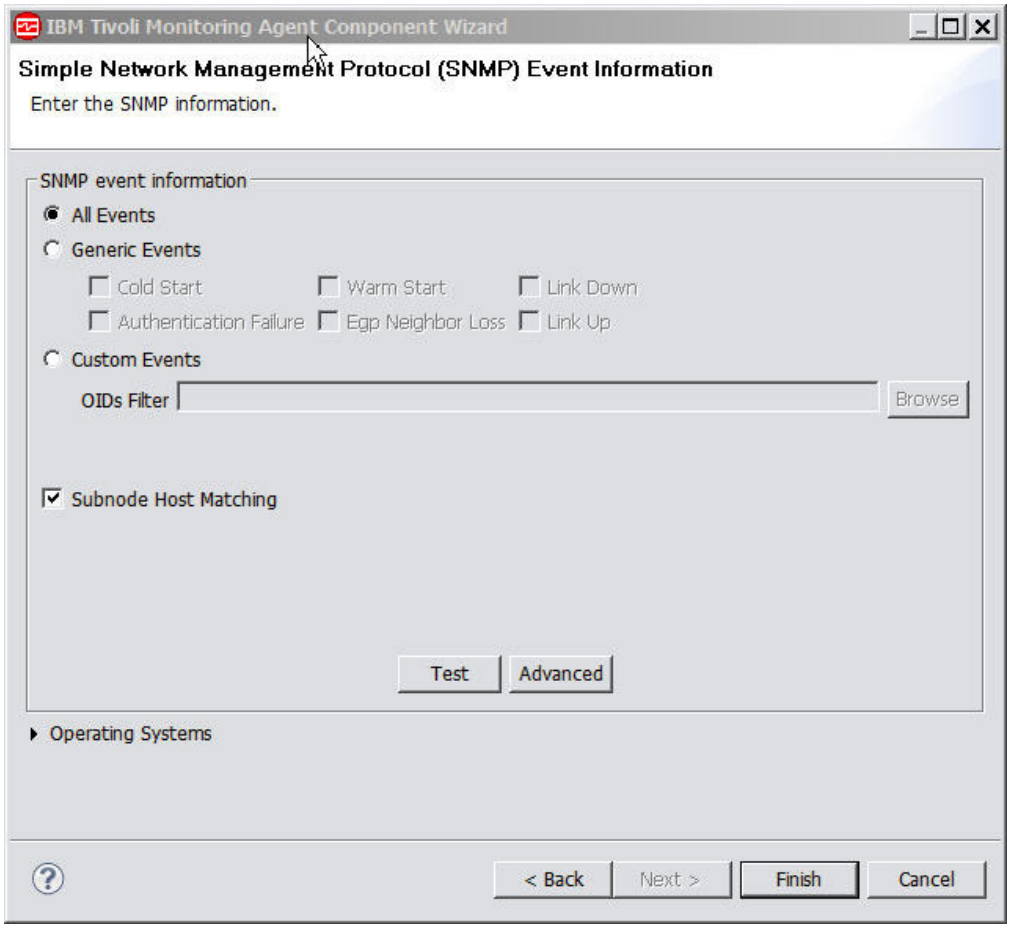


Figure 97. Simple Network Management Protocol Event Information window

a. Click **All Events** to create an attribute group that sends an event for any received SNMP event.

b. 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.

   —OR—

c. 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 (Figure 98 on page 133), the events on the left side of the window are organized by the MIB module in which they were defined. Expand an SNMP object to display 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 635).

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 received in the trap protocol data unit (PDU). For more information about this attribute, see the attribute definition 561.

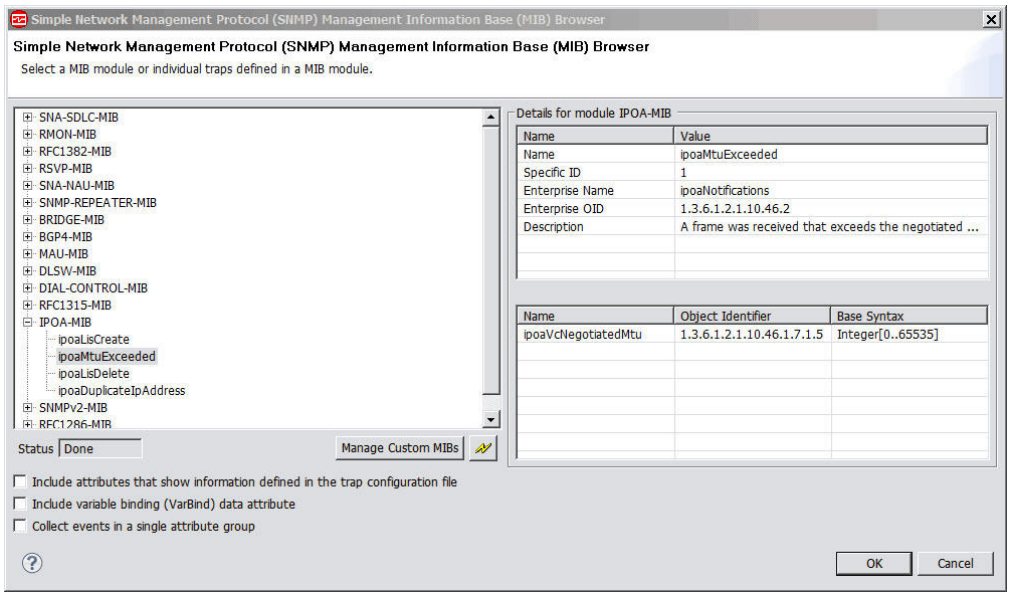


Figure 98. SNMP MIB Browser window

Notes:

1) 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.
2) 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 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 this flag is checked, one attribute is added for each of the variables defined in each of the events (duplicate variables are ignored) If you select a module, variable attributes are not added.
3) 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.

4) 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 97 on page 132, select the **Subnode Host matching** check box to match events to subnodes.

   For example: You have defined 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. Because each router instance includes the same data defined for the event filter, you need an additional way to make sure that events from your router appear in the attribute group for that router.

   When subnode host-matching is selected, an event 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 used by SNMP for that subnode instance is the host used for matching.

   If the **Subnode Host Matching** check box is clear, your subnode instances do not perform 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 97 on page 132, select the operating systems.

7. You can click **Test** in the SNMP Event Information window Figure 97 on page 132 to start and test your agent.. For more information, see “Testing” on page 137

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 405).

   a. When you finish selecting event filtering and summarization, return to the SNMP Event Information window Figure 97 on page 132. 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 99 on page 135), click one or more key attributes for the attribute group, or click **Produces a single data row**.

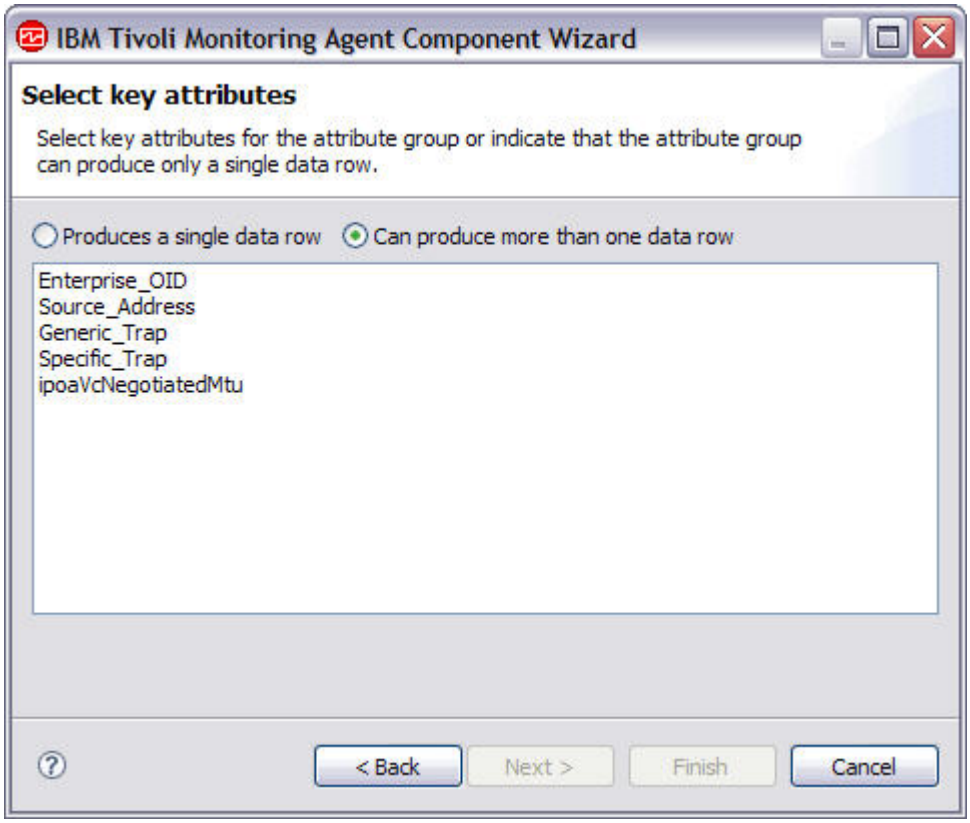


Figure 99. Select key attributes page

9. Click **Next**, or click **Finish** if you are using the new agent wizard to save the agent and open the Agent Editor.
10. For information about adding further attributes, see "Creating attributes" on page 54.

For more information about the attribute groups for SNMP events, see ("SNMP Event attribute groups" on page 560).

Configuration

After a data source is added, the configuration is displayed on the Runtime Configuration page of the Agent Editor. For example, Figure 100 on page 136 shows the configuration sections and some of the configuration properties that are automatically created when an SNMP Event attribute group is added to the agent.

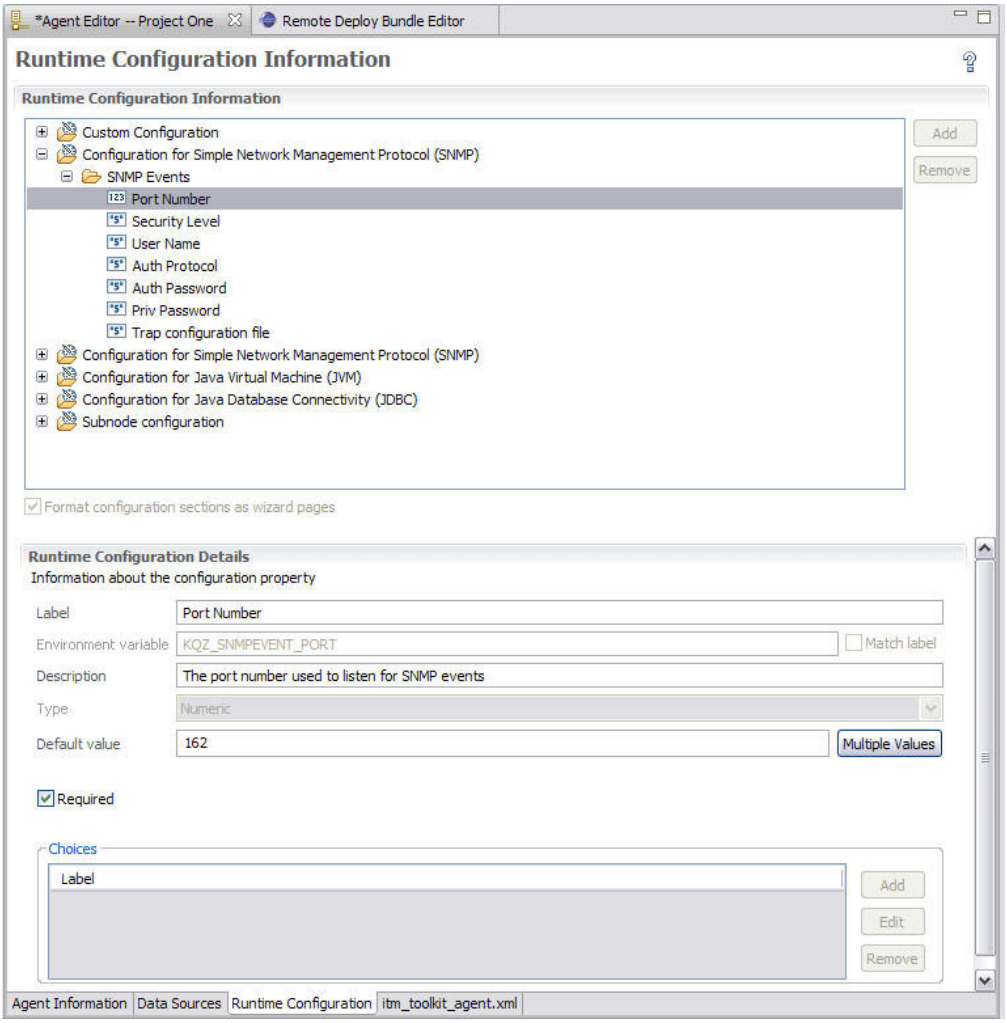


Figure 100. Runtime Configuration page

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 11. SNMP Events configuration properties

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

Table 11. SNMP Events configuration properties (continued)

Name	Valid values	Required	Description
Priv Password	String	No	SNMP V3 privacy password
Trap configuration file	File name including path	No	Location of the trap configuration file. If the file is not located 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 need to specify them, you must specify the appropriate values for the security level you select.

Testing

You can test the attribute group you have created within Agent Builder. To test the attribute group you will need to use a test program or application to generate SNMP events.

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** in the SNMP Event Information window (Figure 97 on page 132).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test**. For more information about the Agent Editor, see “Tivoli Monitoring Agent Editor” on page 37

After you click **Test** in one of the previous two steps, the Test Event Setting window Figure 101 on page 138 is displayed.

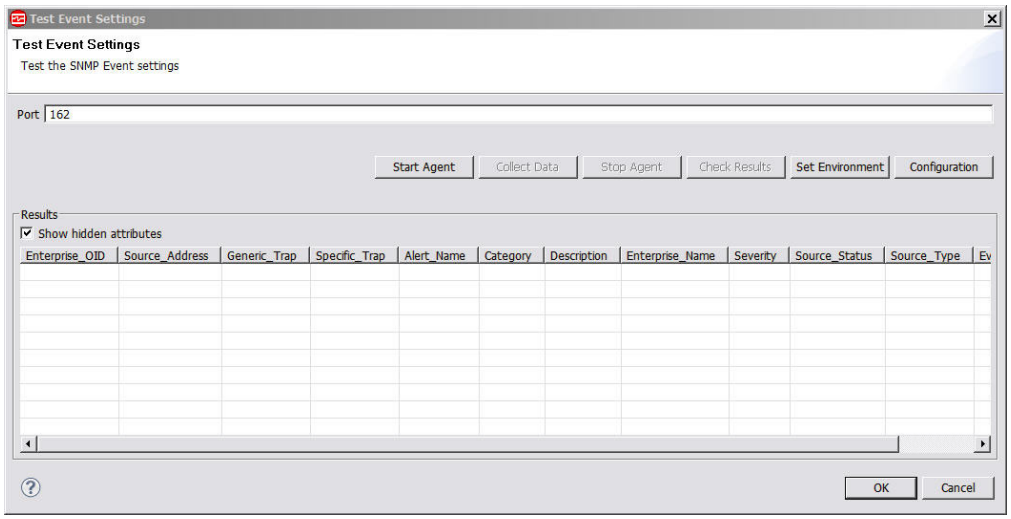


Figure 101. Test Event Settings window

To start and test your agent use the following procedure

1. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see "Configuration prior to testing" on page 374. For more about SNMP Event configuration properties, see "Configuration" on page 135.
2. Click **Start Agent**. A window indicates that the Agent is starting.
3. When the agent starts, it listens for SNMP events according to its configuration.
4. To test your agent's data collection you generate SNMP events matching the agents configuration. You can do this using an application or an event generator. When the agent receives SNMP events matching 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 Settings window collects and displays any events in the agent's cache since it was last started. An example data collection is shown in Figure 102

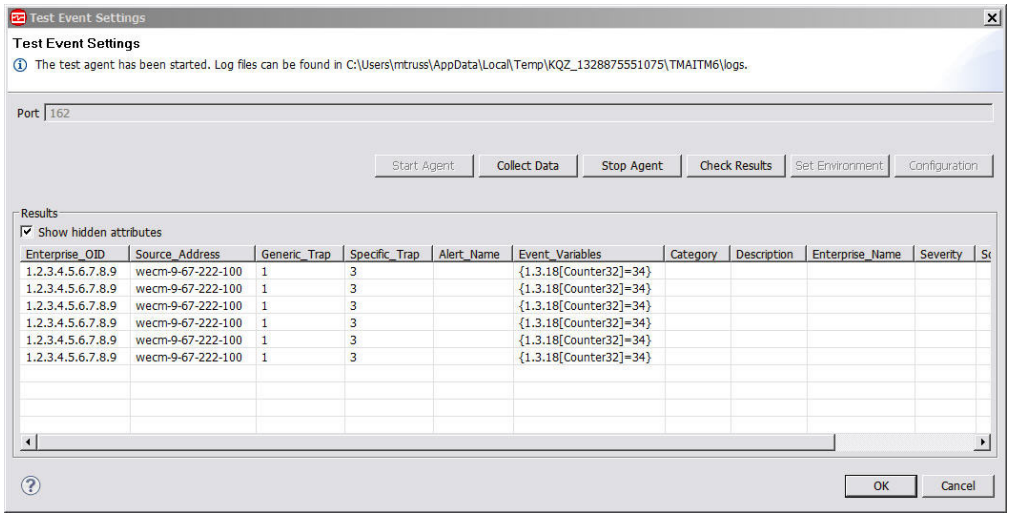


Figure 102. Test Event Settings window showing collected SNMP event data

6. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. This will show you more information about the data, in the Data Collection Status window. An example is shown in Figure 103. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534

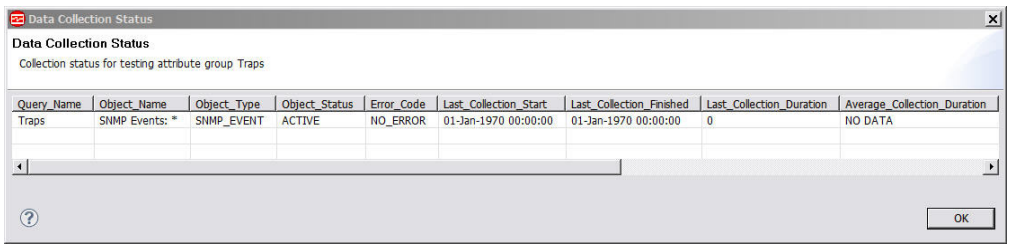


Figure 103. Data Collection Status window

7. The agent can be stopped by clicking **Stop Agent**.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 14. Monitoring Java Management Extensions (JMX) MBeans

With the JMX data provider you can collect data from JMX MBeans. Each JMX data source 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 type of MBean that contains the data you want to collect. Use an Object Name pattern to only identify a set of similar MBeans. The set of MBeans that match 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 you defined for the data source.

Use the following steps to add a JMX data source to collect data from Java Management Extension (JMX) MBeans:

1. On the Agent Initial Data Source page (Figure 104 on page 142) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

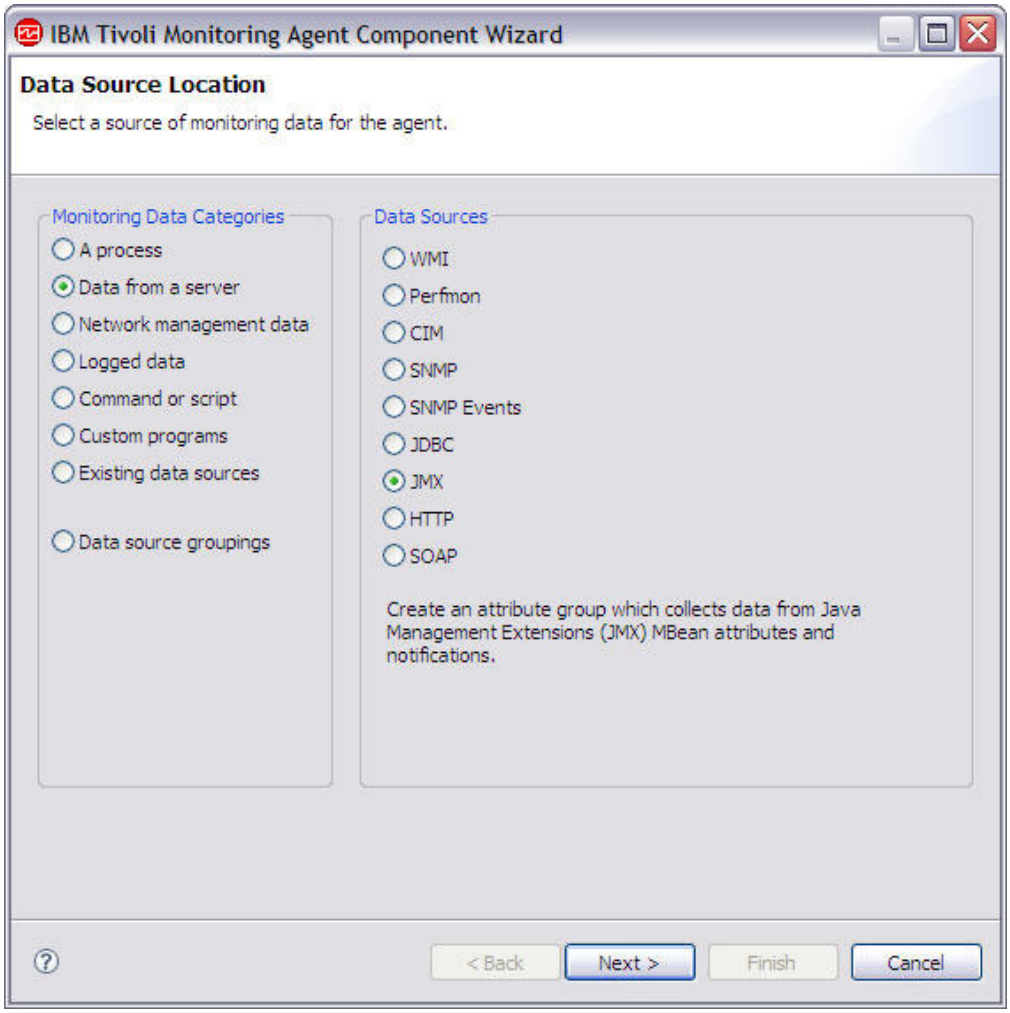


Figure 104. Adding JMX data

2. In the **Data Sources** area, click **JMX**
3. Click **Next**.
4. On the JMX Information page (Figure 105 on page 143), click **Browse** to see all of the JMX MBeans on the MBean server.

   After you have defined 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.

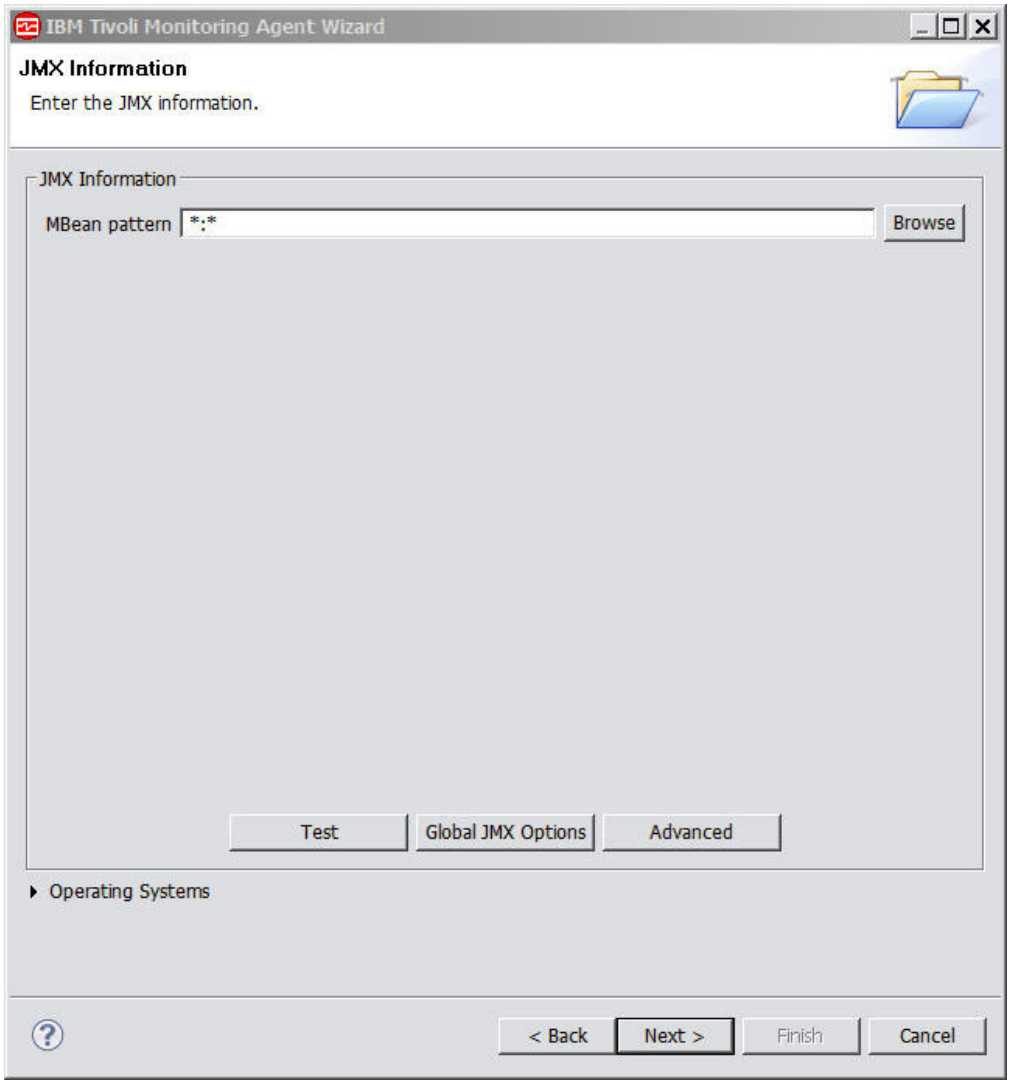


Figure 105. JMX Information page

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 containing 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
Displays the MBean pattern.

Global JMX Options
Displays the level of support (Figure 106 on page 144).

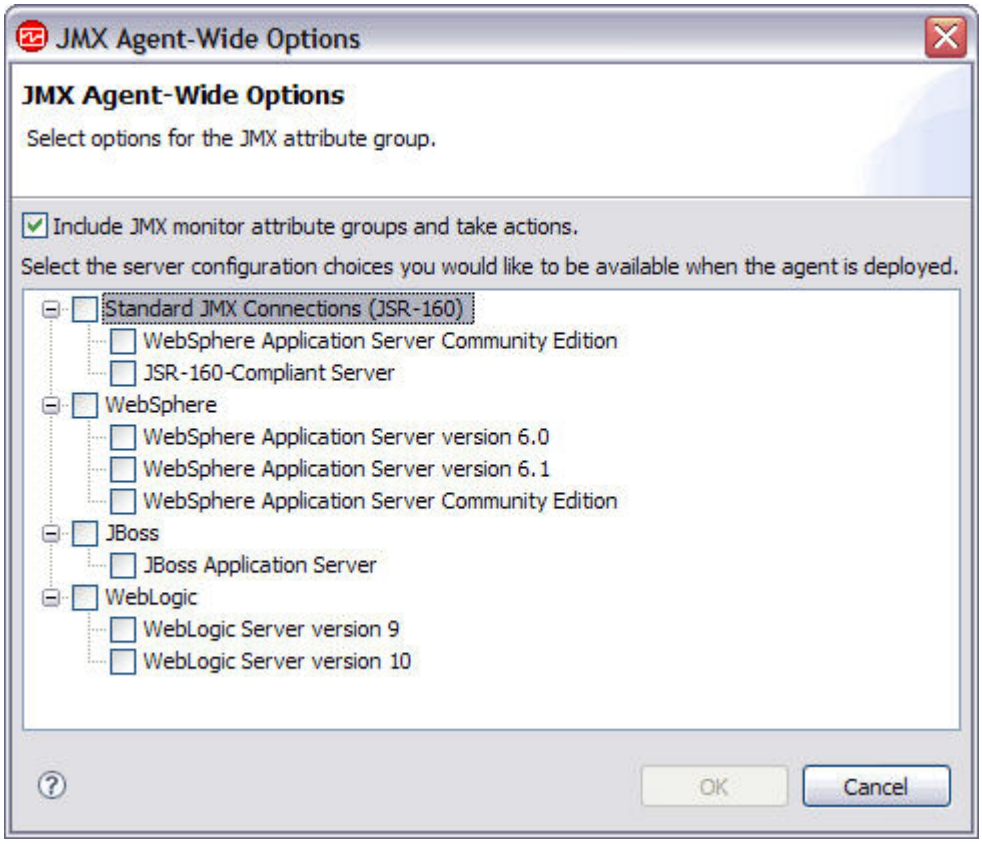


Figure 106. JMX Agent-Wide options window

Support is provided for the following JMX servers:

- Java 5 operating system MBean Server. Connection is made 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 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 you have 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 new connection to an MBean Server, click the plus (+) icon to add a new connection or to edit an existing connection. The following window (Figure 107 on page 145) is displayed when no connections have

been defined:

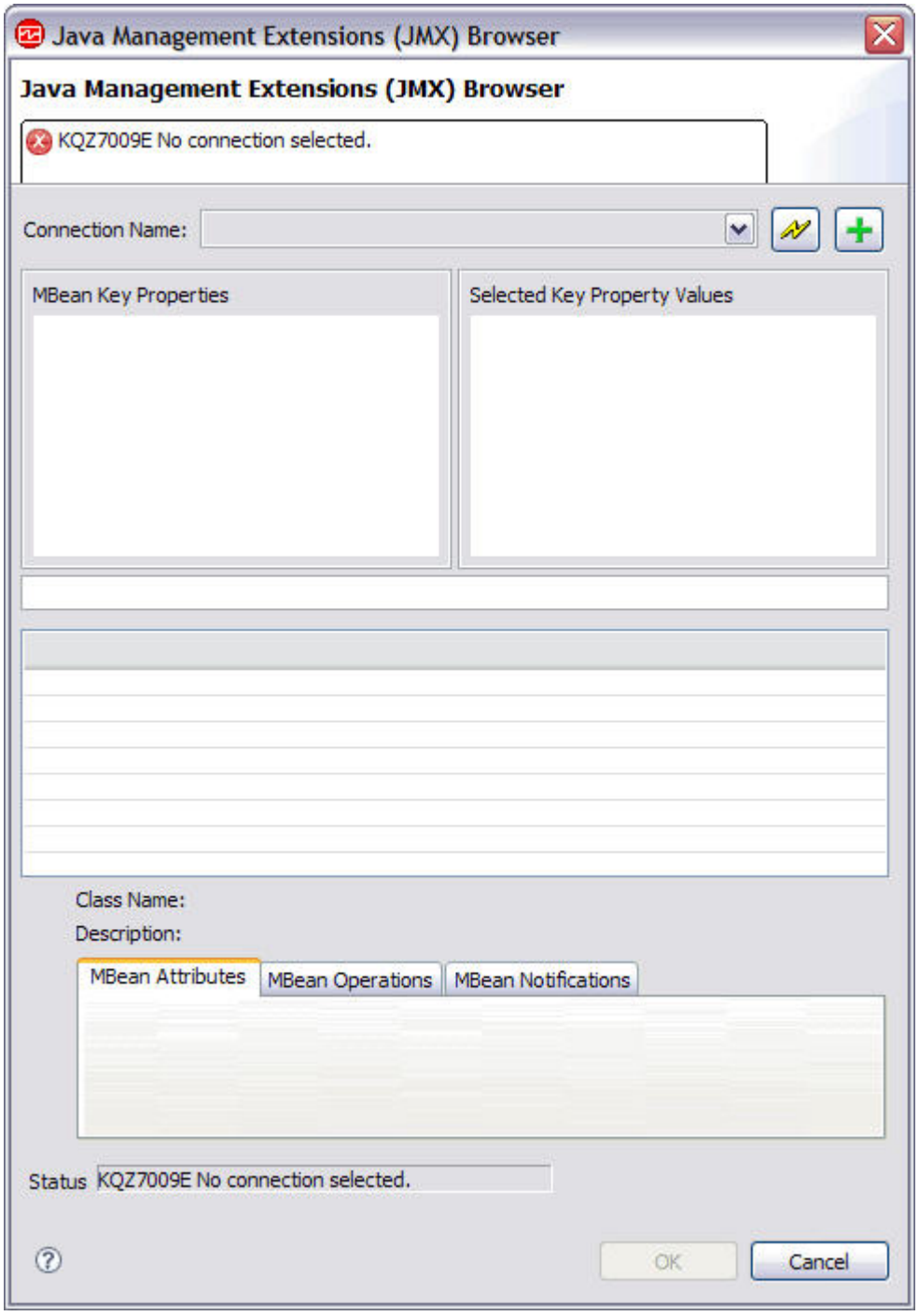


Figure 107. JMX Browser with no connection selected

b. After you click the plus (+) icon to add a connection, the JMX Connection Selection page (Figure 108 on page 146) is displayed:

Figure 108. JMX Connection Selection page

c. Use the MBean Server Connection Wizard to connect to an MBean server. The new connections listed on the page are selections you can make to create a new 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 new connection.

Figure 109. JMX connection templates

d. After selecting a connection type, you might be asked to select a more specific type of connection. On the page above, two templates 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**.

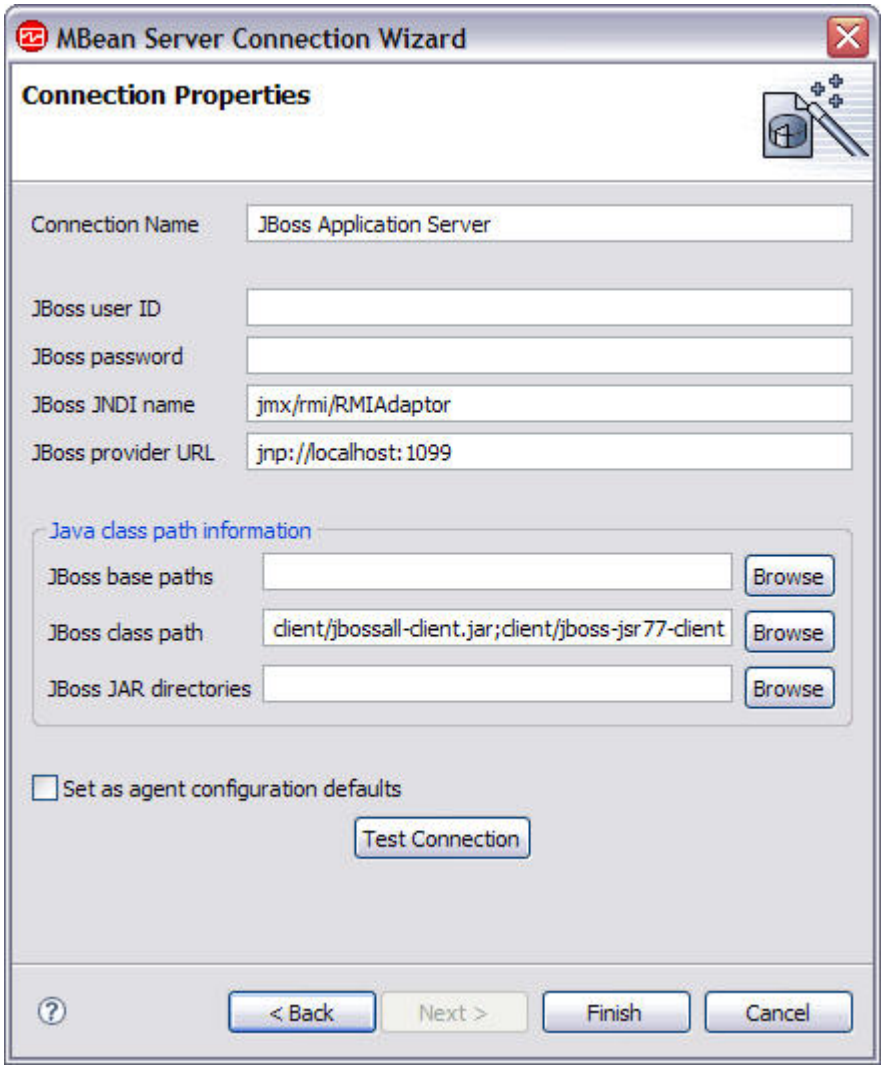


Figure 110. JMX connection properties

The Connection Properties page (Figure 110) contains the details on how to connect to an MBean server. This page needs to be completed with details about your MBean server.

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

After you have specified the properties needed to connect, click **Test Connection** to ensure that the connection can be established. If the connection is not successful, correct the necessary properties.

When the connection is successful, click **Finish** to return to the browser that will begin using the connection you just configured.

The Java class path information in the Connection Properties page contains three fields that should be completed as necessary to connect to an MBean server that requires java classes that are not included in the Java 5 runtime environment. Normally, the MBean server you need to connect to should be installed on the same system as the Agent Builder. In this case, specify the directory where the application containing the MBean server was installed as the Base Paths field. The Jar Directories field should then list the directories relative to the Base Paths directory that contain the jar files

needed to connect to the MBean server. The Class Path field can be used to include specific jar files. The jar files listed in the Jar Directories field do not need to be listed separately in the Class Path field.

Any of the fields can contain more than one reference; just separate the entries by a semicolon. These values are the same values that are needed when configuring the agent. See "JMX configuration" on page 157 for additional information about these fields.

6. After you have selected a connection, the JMX Browser downloads information about the MBeans from the JMX server. This information is displayed in the following four areas of the JMX Browser window (Figure 111 on page 150):

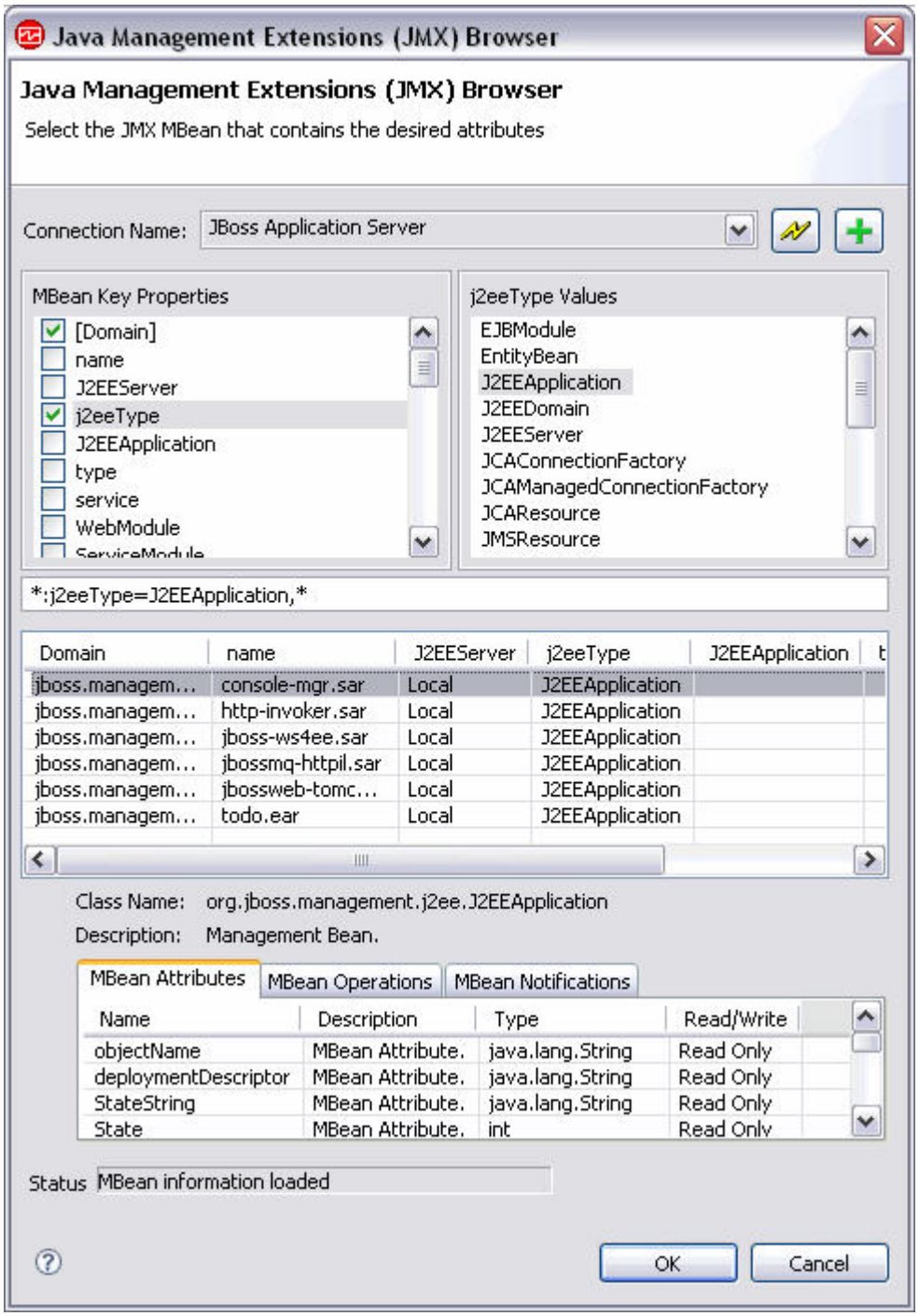


Figure 111. Java Management Extensions (JMX) Browser window

- **MBean Key Properties** area (upper left) - This is collection of every unique Object Name key found from all the MBeans on the server. The [Domain] entry is special because it is not really a key, but it is treated as an implied key for the value of the MBean's domain. When you select an item from this list, the JMX Browser finds all of the MBeans that contain that key property and displays the list of values of that key property in the list on the right. When you check a key property, it is included in the Object Name pattern for the data source.

- Selected Key Property Values area (upper right) - Shows the values of the currently-selected MBean Key Property from all MBeans. Selecting one of these values will check the MBean key property and update the Object Name Pattern that is displayed in the text field below with the MBean key property name and value.
- A table listing all MBeans matching the Object Name Pattern (center) - As you select Key Properties and Values from the MBean Key Properties and Selected Key Property Values lists, you will see the Object Name Pattern update and the list of MBeans in this table will change to reflect the list of MBeans that match the pattern you have currently selected. If you have a pattern that is not matching any MBeans, you can clear entries in the MBean Key Properties list by clicking the check box next to a key 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 context menu or by clicking on a column header). You can also show any key attribute in any column by selecting "Show Key Property" from the context 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 context menu.
- A table containing 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 showing 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 described above to build an Object Name Pattern that matches a set of MBeans that each contains the monitoring data you want to collect. For example, if you wanted to monitor data from all of the ThreadPool MBeans, you would do the following:

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 selecting **ThreadPool**, the type key property name is checked 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 until you have the Object Name Pattern that identifies the set of MBeans you want to monitor. You can also display a context 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 will be 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 will be warned that the data

source will be created with no attributes. If the selected MBean contains notifications, an "Event" data source will also be 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 and a warning is displayed, giving you a chance to select an MBean if you want to. 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 do the following:
   - Select whether JMX monitors are supported by this agent. See the next section for a description of JMX monitors. If you want JMX monitor attribute groups and Take Action commands to be created, check **Include JMX monitor attribute groups and take actions**.
   - Select the types of MBean servers your agent will connect to when it is deployed. There are several vendor-specific types of servers listed, along with a generic JSR-160-Compliant Server for standards-based servers. You can select as many as needed, but you should only select 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 have selected the desired option.

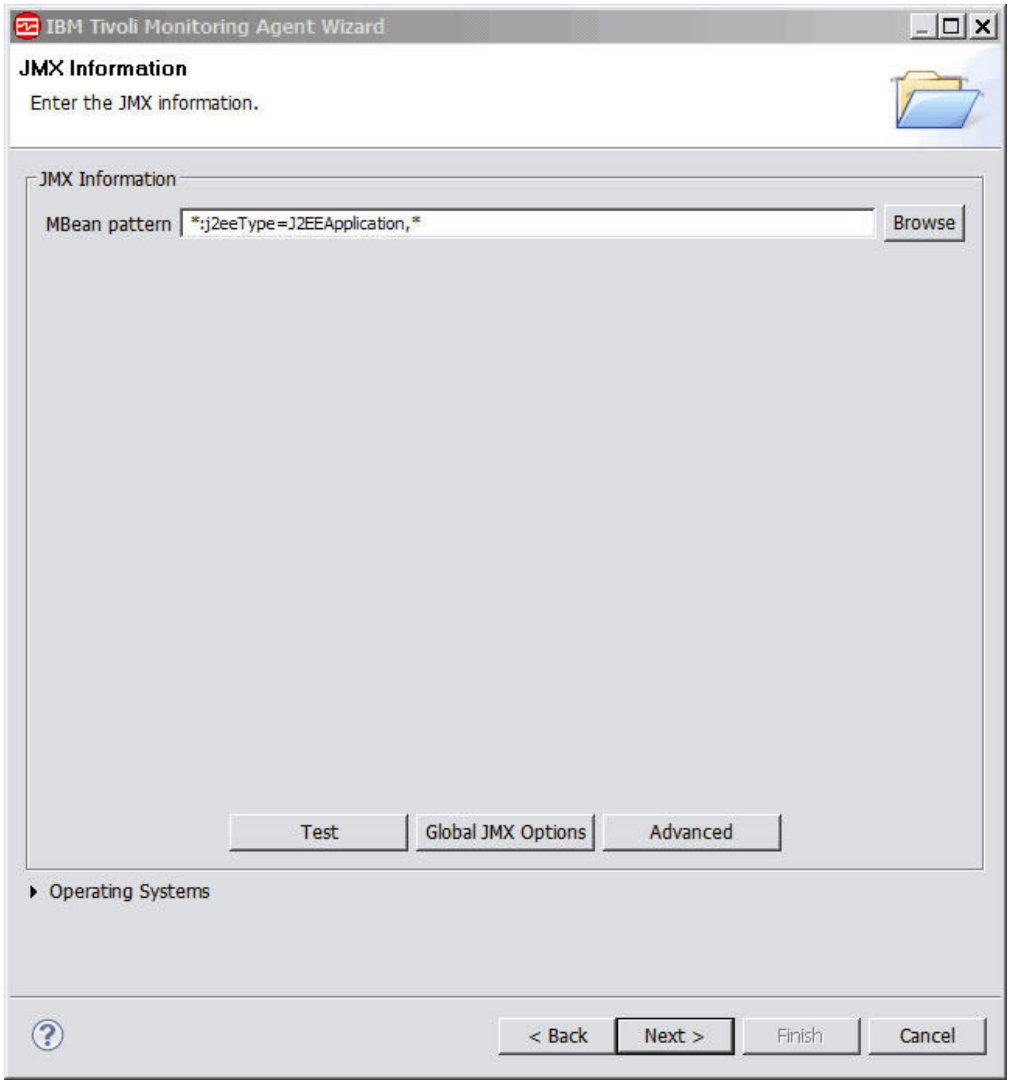


Figure 112. JMX Information page

11. *(Optional)* You can test this attribute group by clicking **Test**. For more information about testing, see "Testing" on page 170
12. *(Optional)* You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66
13. Click **Next**.
14. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. See "Selecting key attributes" on page 27 for more information.
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 113 on page 154), select any other types of application servers to which you want your agent to be able to connect.

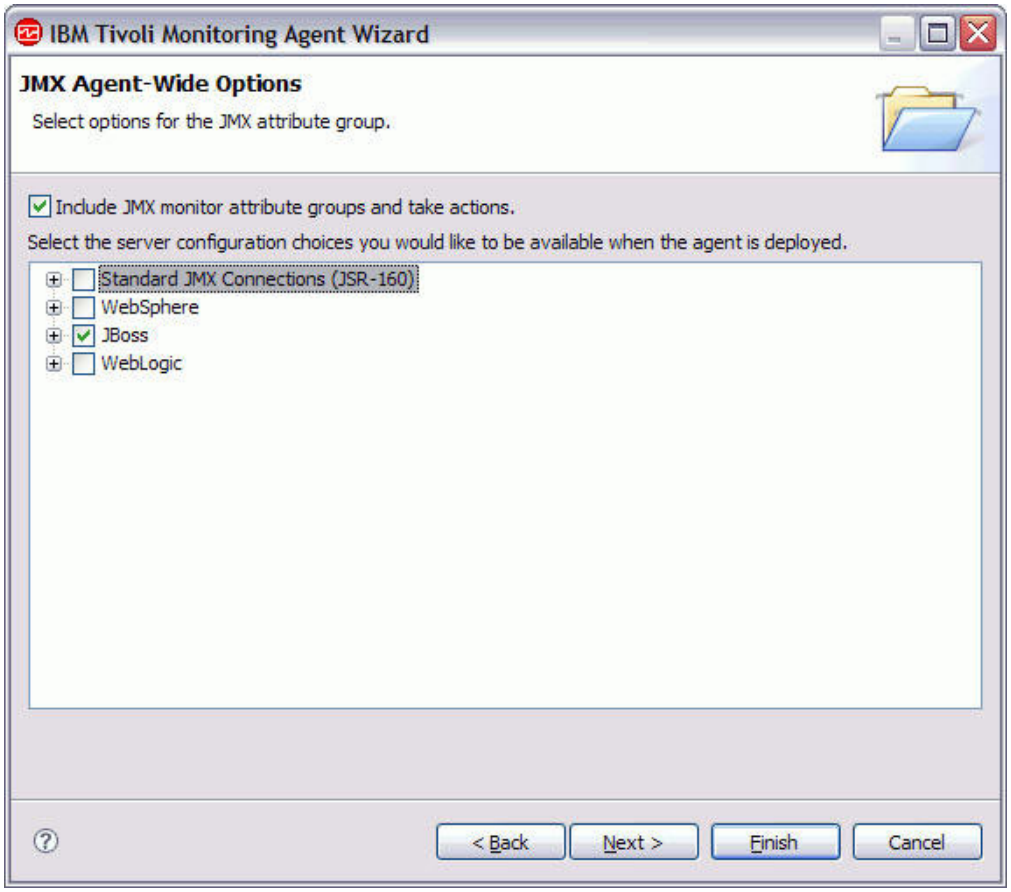


Figure 113. JMX Agent-Wide Options window

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

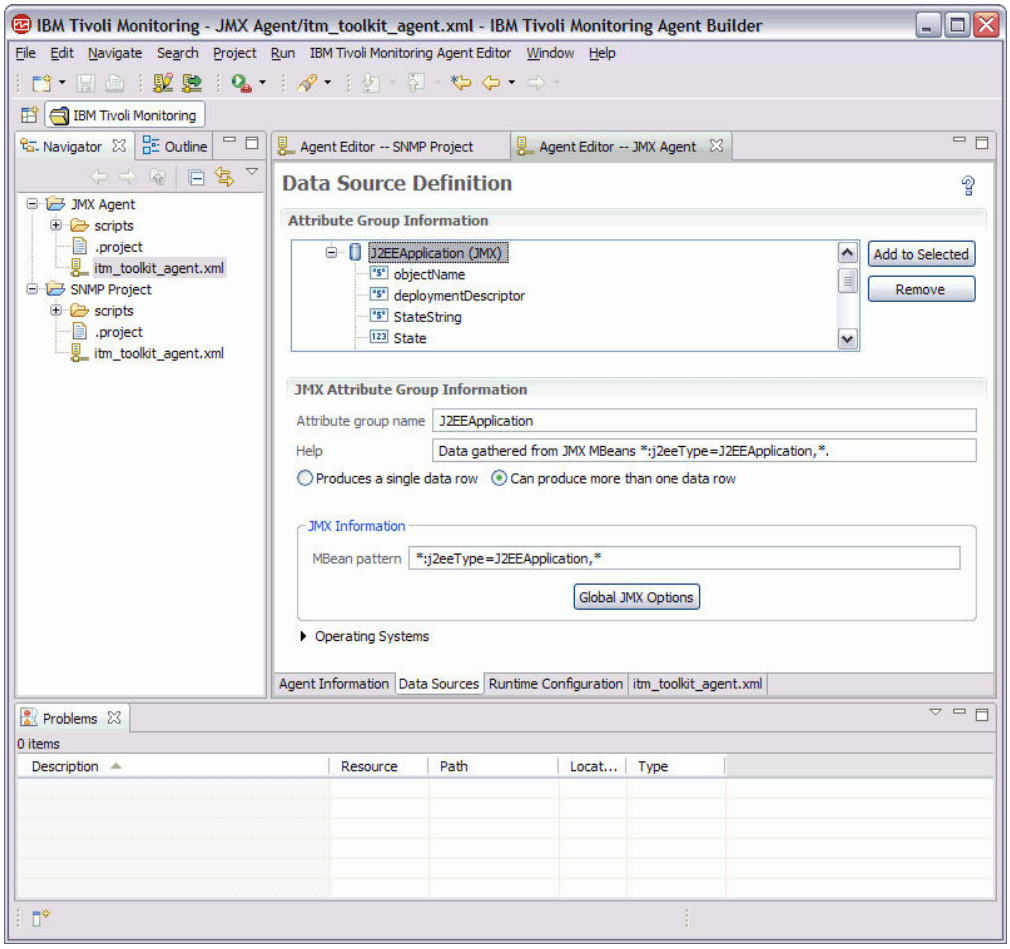


Figure 114. New attribute group in Agent Editor

18. If you want to change the types of application servers to which you can connect after the agent has been created, click **Global JMX Options** in the **JMX Data Source Information** area (Figure 114).
19. In the JMX Agent-Wide Options window (Figure 115 on page 156), change any selections that you want.

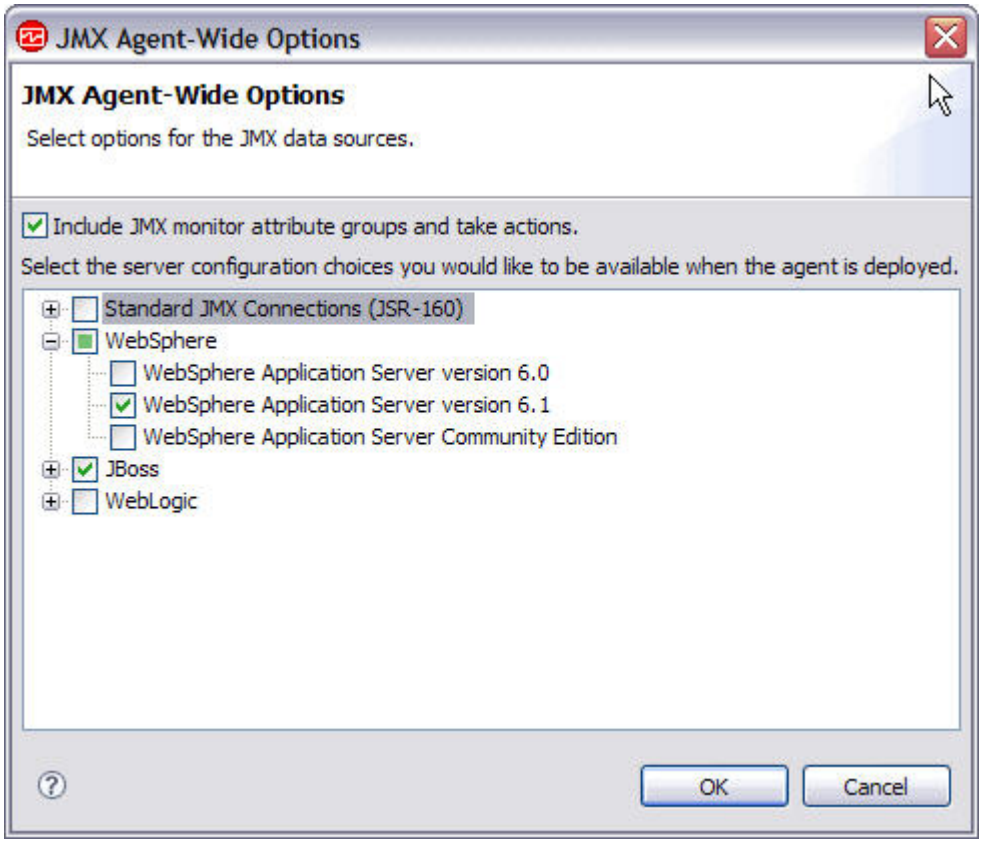


Figure 115. JMX Agent-Wide Options window

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 116 on page 157).

    The default value of the property, JBoss base paths, has the same value that was entered in the JMX browser in Step Figure 111 on page 150.

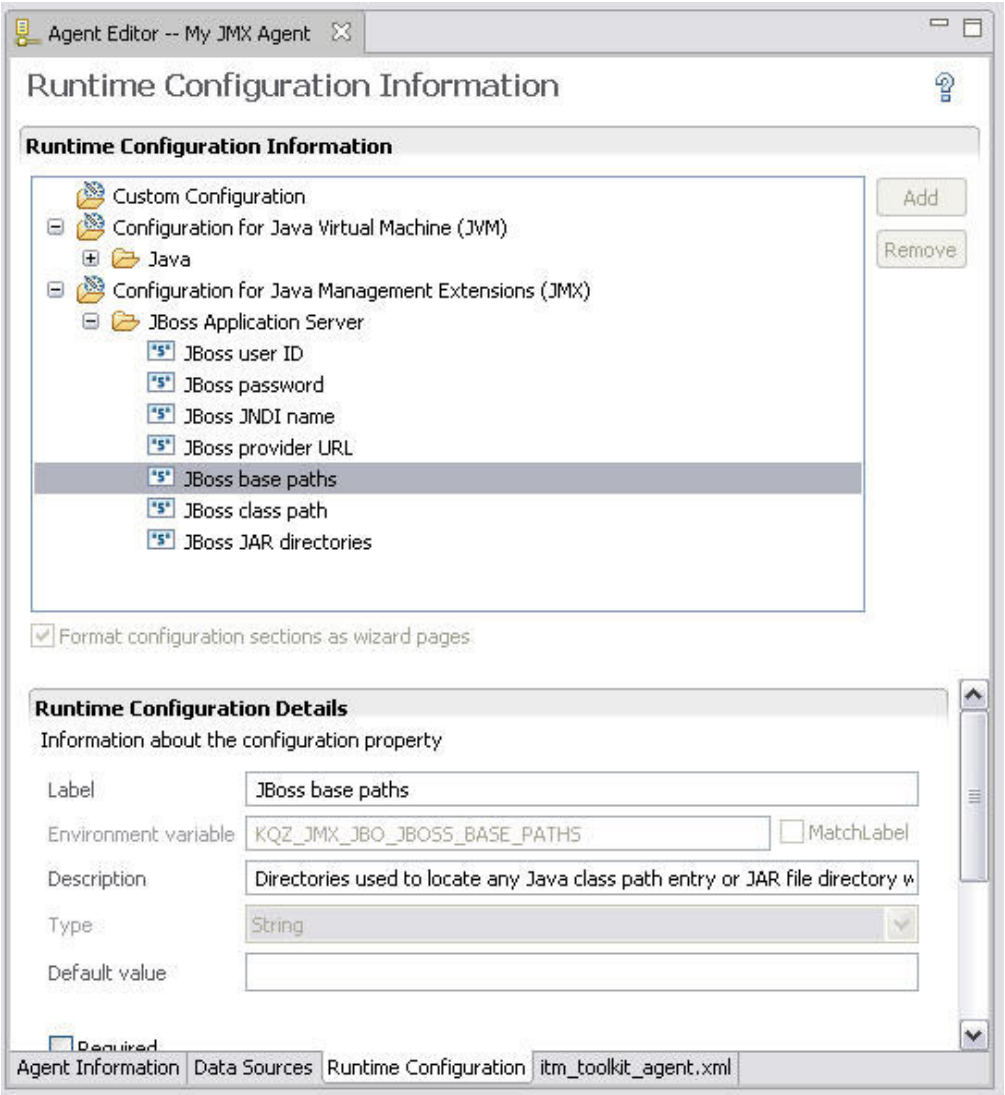


Figure 116. Runtime Configuration tab of the Agent Editor

For more information about the attribute groups for JMX events, see ("JMX Event attribute groups" on page 562),

JMX configuration

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, but it is usually 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 will not be included in the agent.

In most cases, an agent is designed to monitor one JMX application server type. When using the JMX Browser when creating the JMX data source, the JMX server configuration options used to browse the Mbean server are selected and added to your agent automatically. If you want to change the types of application servers to which you can connect after the agent has been created, click **Global JMX Options**

in the **JMX Data Source Information** area (Figure 114 on page 155). In the JMX Agent-Wide Options page (Figure 115 on page 156), 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 115 on page 156). When more than one type of JMX connection is supported, the runtime configuration prompts you for the connection type that will be 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 using the Agent Editor. See "Changing configuration properties using the Agent Editor" on page 367. If a JMX data source is defined in a subnode, you will also be 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 (The Agent Builder does not require these properties because it uses its own JVM and logging, which is configured through the JLog plugin.):

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 414.

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 name that is used to authenticate with the MBean server.

Password
Password for the User Name entered above.

Base paths
List of directories that are searched for jar files named in the Class Path, or directories 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 that are not fully qualified will be 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 need to be explicitly identified; they will be found because they reside in one of these directories. Subdirectories of these directories are not searched. Any directory names that are not fully qualified will be appended to each of the Base Paths until the directory is found.

Note: When monitoring remotely, the jar files and all of their dependent jar files that are required to connect to the application being monitored must be installed locally on the computer where the agent is running. These jar files must be configured in the JAR directories, in the base and class path fields. 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:

JMX Service URL
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 resides. If monitoring locally, this is the local system name. If monitoring remotely, this is the host name of the system where the application server resides.

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 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 including the JNDI name.

Note: If WebSphere administrative security is enabled you must make sure client login prompting is 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 your WebSphere application server profile's properties directory. 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 should be set to not contain a value.

You can view, add, and change the configuration properties using the Agent Editor. See "Changing configuration properties using the Agent Editor" on page 367. 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, monitors, and operations

In addition to providing monitoring data when requested, some MBeans also provide the following data:

- "JMX notifications"
- "JMX monitors" on page 161
- "JMX operations" on page 166

JMX notifications

In addition to providing monitoring data when requested, some MBeans also provide notifications. A notification is an object generated by an MBean that is passed to registered listeners when an event occurs.

Agents 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 attribute group's MBean pattern. The attribute group then displays one row per notification received. Each column contains one item of data from the

notification. The data desired 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

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 if you select **Include JMX monitor attribute groups and take actions** under Global JMX Options.

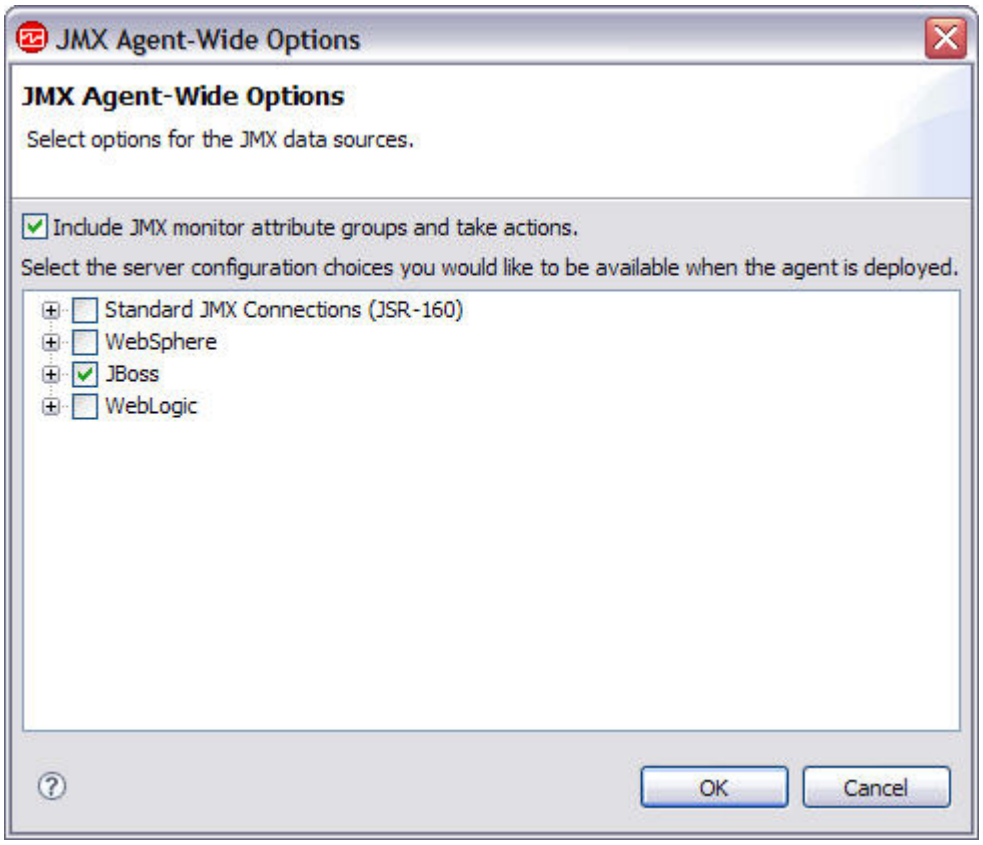


Figure 117. JMX Agent-Wide Options page

Any MBean that reports on an attribute of another MBean can be considered a monitor. In practice, JMX defines three concrete monitor classes, and those 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 have been added by the user.
- Counter Notifications

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

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

  This attribute group reports all notifications received from String Monitors.

Take Action commands for 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 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.

The following sections provide the parameters needed to create each type of monitor and examples:

- "JMX Add String Metric Watcher"
- "JMX Add Gauge Metric Watcher" on page 163
- "JMX Add Counter Metric Watcher" on page 164
- "JMX Delete Metric Watcher" on page 165

JMX Add String Metric Watcher: Use this Take Action command to create a monitor.

***Parameters**:*

MBean pattern
: All MBeans matching this pattern will be 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 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: This 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 will be sent.

JMX Add Gauge Metric Watcher: Use this Take Action command to create a monitor.

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 below.

Low threshold
Value that the observed attribute is expected to stay above.

Example: Request a notification when free memory goes below 10M:

```
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 will 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.

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 new threshold.

Modulus
Counter's maximum value, 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 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 3 errors, 4 will be added 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, etc. 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, but is exposed 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.

Starting and stopping monitors

Monitors are persistent across starts and stops of the agent and the JMX server. If the agent detects that the JMX server has been recycled, it reregisters the monitors. If the agent is recycled, monitors are reregistered. The monitor definitions are stored in a file called `default_`*`instanceName`*`.monitors` where *instanceName* is the agent instance name or default if it is a single instance agent. This file is located in the following directory (note that *xx* denotes the 2 character product code):

- Windows systems: `TMAITM6/k`*`xx`*`/config`
- UNIX/Linux systems: *`architecture`*`/`*`xx`*`/config` (see "New files on your system" on page 382 for information about determining the architecture value)

If the agent is restarted, it will use the monitor definitions file to restore the monitors.

JMX operations

Agents that have JMX data sources include the JMX_INVOKE Take 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 will run . If the pattern matches more than one MBean, the operation executes on each of the matched MBeans.

Operation name
: Name of the MBean operation to execute.

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 executed. If the operation returns a value, that value is written to the JMX data provider log file.

Example: Invoke 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: This is the operation that is executed 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: Invoke 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: This is the operation that is executed 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 end user to run the JMX_INVOKE Take Action command. Instead, additional 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.

Specific fields for Java Management Extensions (JMX) MBeans

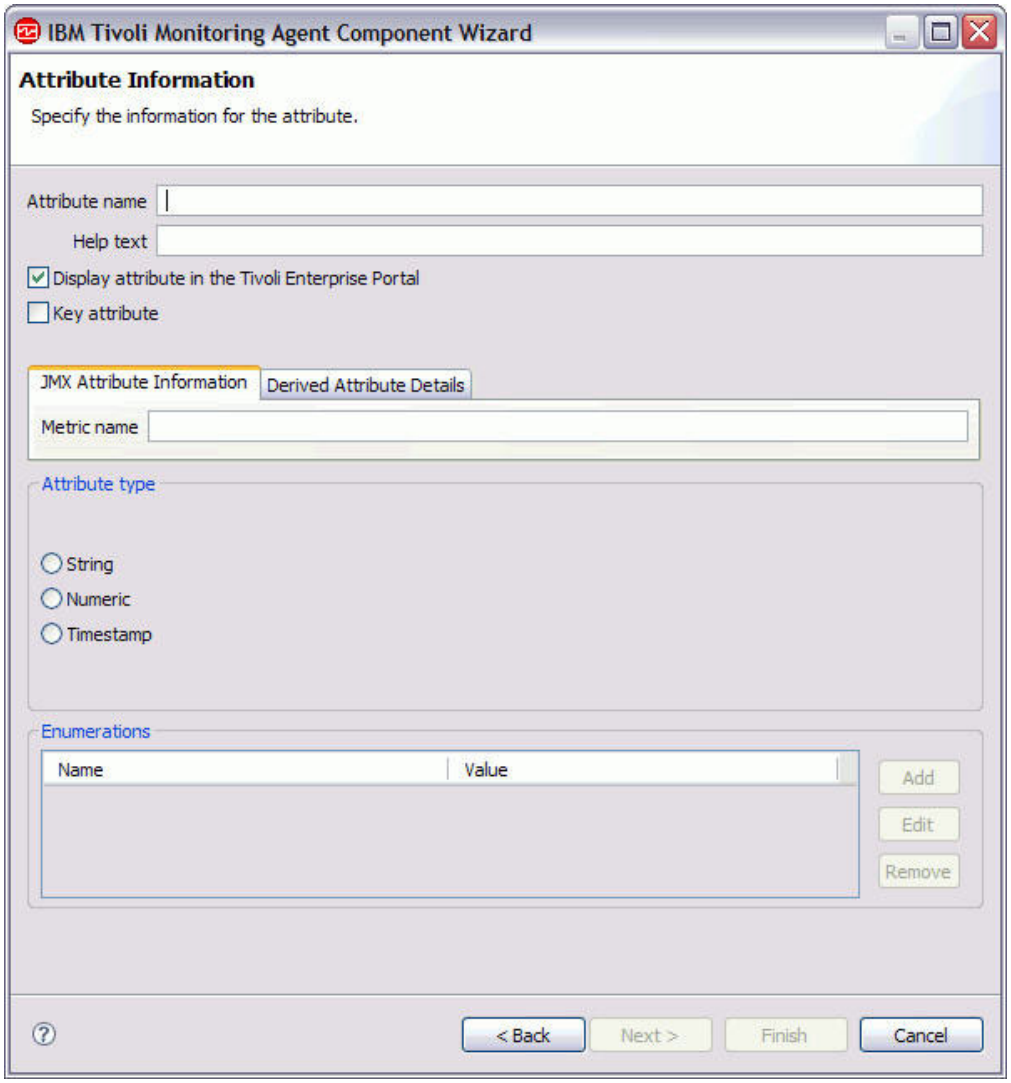


Figure 118. Attribute Information page

The syntax of the metric name for a JMX Attribute group consists of tokens separated by a period. The tokens form primary values and optionally secondary values:

- **Primary value**: a value obtained directly from the MBean or Notification in a given 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 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, or the result of a method call on that object. Such secondary values are obtained 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 also be the domain or 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 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 one of an MBean operations
- Value of a numeric or string public instance variable in a Notification object
- Numeric or string return value from one of a Notification's operations

By adding a secondary value to the metric's definition, you can drill down into a Java object's primary value, and invoke a public method or fetch a public instance variable.

By adding a secondary value to another secondary value in the metric's definition, you can drill down into a secondary value object. This 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.
Method.getHeapSize	MBean	The value 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 perform 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	This goes three 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 used is passed to the method as the argument. Only one 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.

Note that 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 perform operations on arrays or collections of data.

- **ElementCount** – returns the number of elements in an array.
- **Explode** – this explodes a row into several rows, one new row for each element of an array.

Examples of each of the above:

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 will contain the number of elements in the array.
Attribute.deployedObjects. Explode.MBean.Property. j2eeType	MBean	This causes the table to have one row for each element in deployed objects. This column contains the deployed Object's j2eeType.
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

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the JMX Information page (Figure 112 on page 153).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37.

After you click **Test** in one of the previous two steps, the JMX Test window Figure 119 on page 171 is displayed

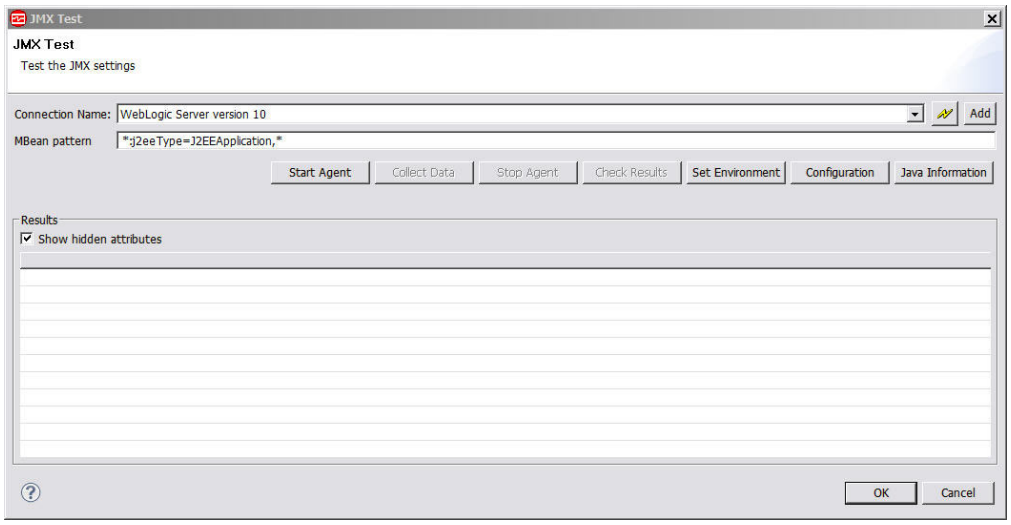


Figure 119. JMX Test window

To start and test the data source use the following procedure

1. Select a connection from the list available under **Connection Name** or alternatively click **Add** to add a new connection and follow the procedure detailed at step 5c on page 146.
2. *(Optional)* Before starting your test you can set environment variables, configuration properties and Java information. For more information, see "Configuration prior to testing" on page 374. For more about JMX configuration, see "JMX configuration" on page 157.
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 JMX Server for data.
5. The Test window collects and displays any data in the agent's cache since it was last started.
6. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534
7. The agent can be stopped by clicking **Stop Agent**
8. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 15. Monitoring data from a Common Information Model (CIM)

This chapter contains the following information:

- "Steps for monitoring data from a CIM"
- "CIM configuration" on page 176

Steps for monitoring data from a CIM

Use the following steps to collect data from a Common Information Model (CIM) repository:

1. On the Agent Initial Data Source page (Figure 120) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

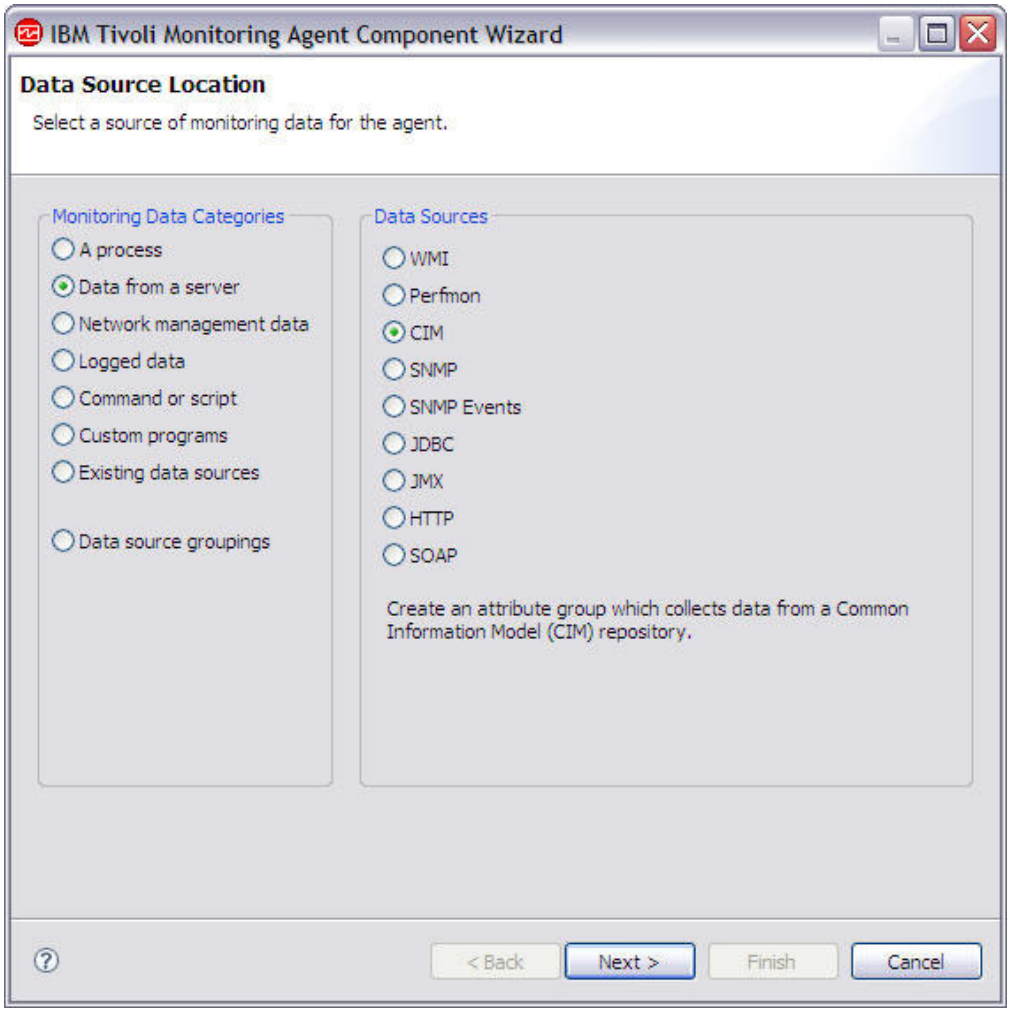


Figure 120. Adding CIM data

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

4. On the Common Information Model (CIM) Information page (Figure 121) **CIM information** area, do one of the following steps:

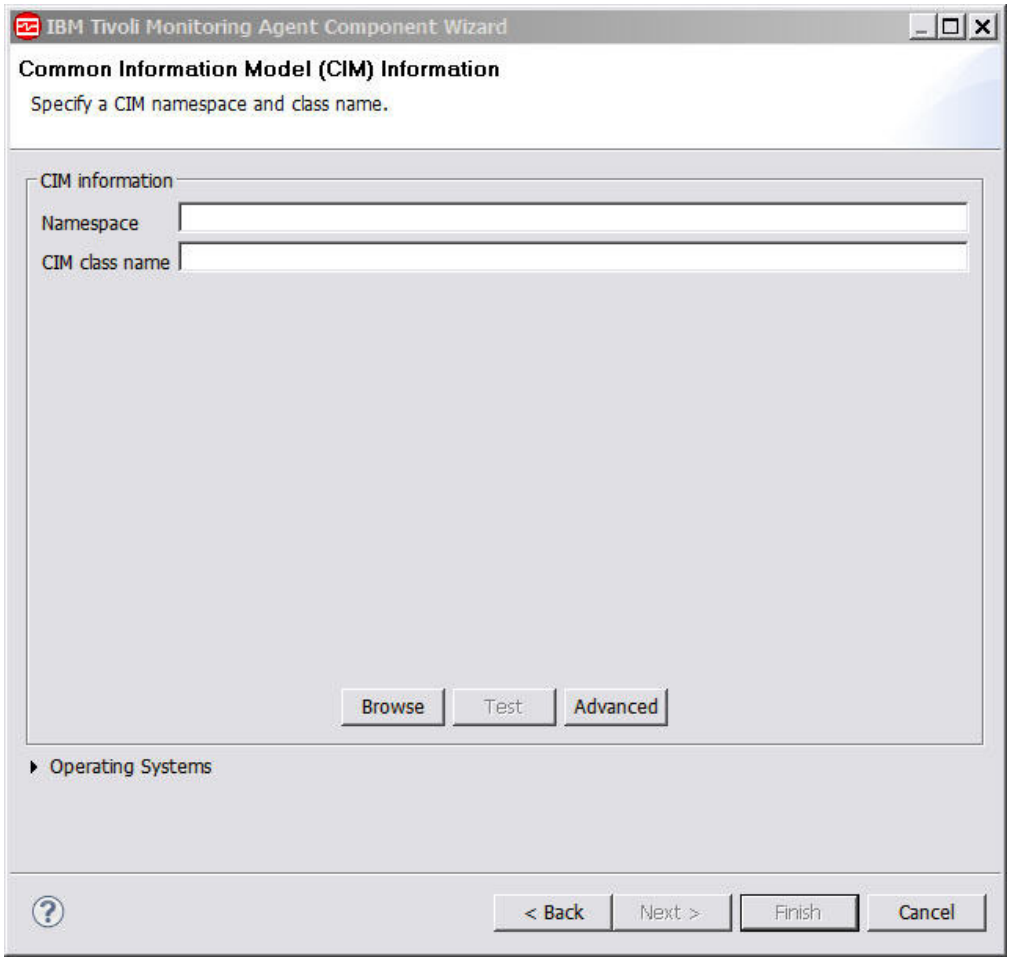


Figure 121. Common Information Model (CIM) Information page

a. Complete the **Namespace** and **CIM class name** fields for the data that you want to collect.

   —OR—

b. 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 122 on page 175).

   To browse a remote system, select a system from the **Hostname** drop-down list (if one has been defined), or 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 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 that has read permission to the objects in the namespace that you want to browse. The window is updated with the information for the remote system.

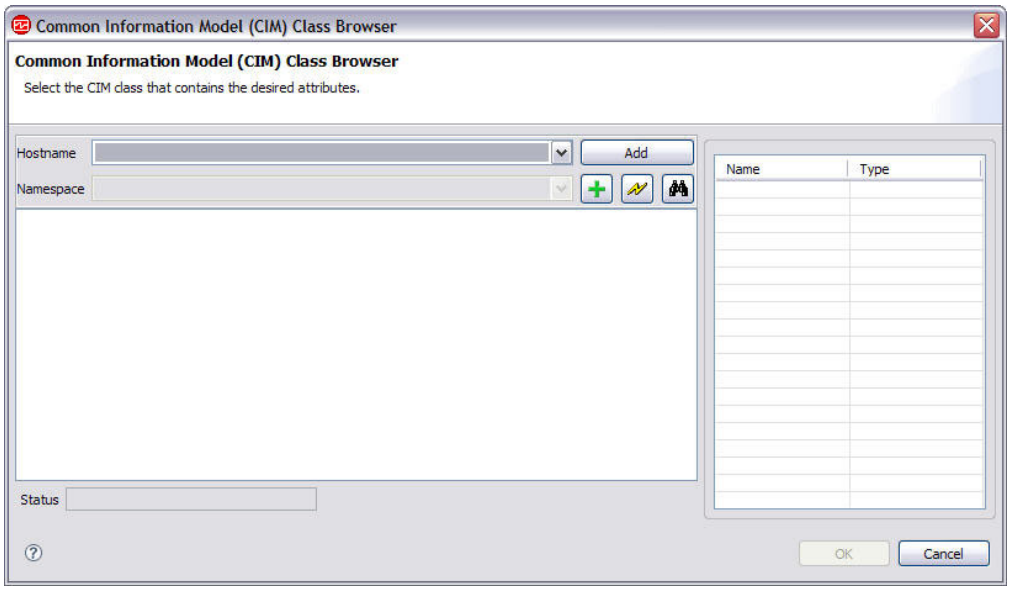


Figure 122. Common Information Model (CIM) Class Browser window

The Agent Builder attempts to discover the namespaces available on the CIM Server. The discovered namespaces are displayed in the **Namespace** drop-down 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 drop-down list, click the plus (+) icon next to the **Namespace** drop-down list. Enter the name of the namespace in the field and click **OK**. If the namespace is present on the CIM server, the classes defined in the namespace are listed. The namespaces you type are saved and put into the namespace drop-down list the next time you browse that particular CIM server.

When you select a namespace from the drop-down 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 the **Connect** to delete any cached information, and cause 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:
   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**.

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. See "Selecting key attributes" on page 27 for more information.
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" on page 177
10. *(Optional)* You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66
11. Do one of the following steps:
    a. If you are using the New Agent Wizard, click **Next**.
    —OR—
    b. Click **Finish** to save the data source and open the Agent Editor.

CIM configuration

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 using the Agent Editor. See "Changing configuration properties using the Agent Editor" on page 367. 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 used for communication that is not secure.

CIM SSL port number
The port number used for secure communication. The default value is 5989. (The default value for Solaris 8 is usually different.)

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the CIM Information page (Figure 121 on page 174).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see “Tivoli Monitoring Agent Editor” on page 37

After you click **Test** in one of the previous two steps, the CIM Test Settings window Figure 123 is displayed

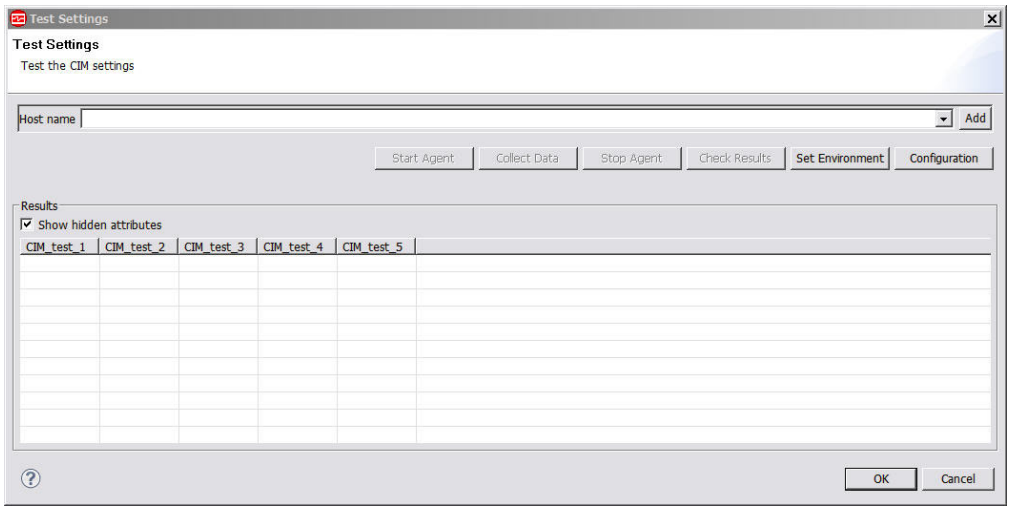


Figure 123. CIM Test Settings window

To start and test the data source use the following procedure

1. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see “Configuration prior to testing” on page 374.
2. Select or add a Host name, for more about adding a Host name, see 4b on page 174
3. Click **Start Agent**. A window opens indicating 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 CIM Server for data.
5. The Test Settings window collects and displays any data in the agent's cache since it was last started.
6. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in “Performance Object Status node” on page 534
7. The agent can be stopped by clicking **Stop Agent**
8. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, “Testing and debugging your agent,” on page 373

Chapter 16. Monitoring a log file

Use the following procedure to collect data from a log file:

1. On the Agent Initial Data Source page (Figure 124) or the Data Source Location page, click **Logged Data** in the **Monitoring Data Categories** area.

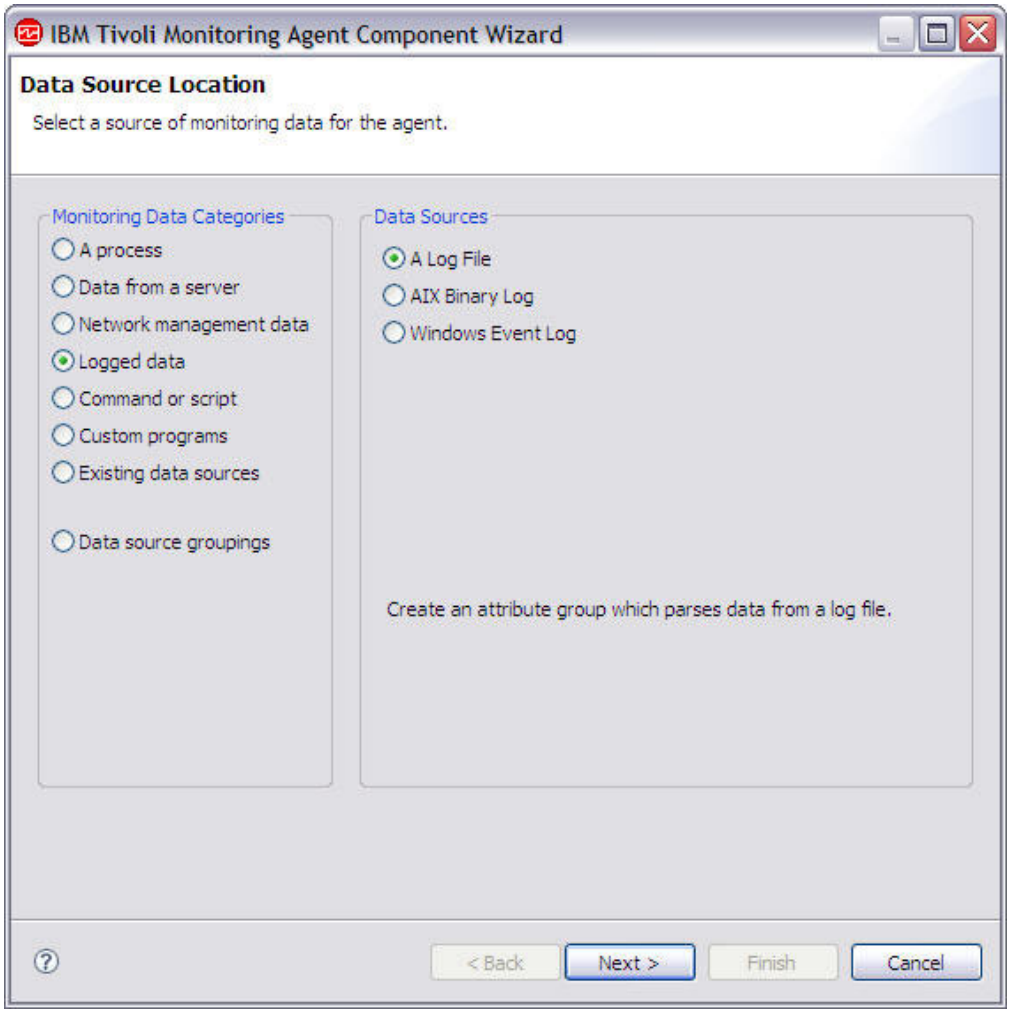


Figure 124. Adding a log file

2. In the **Data Sources** area, click **A Log File**.
3. Click **Next**.
4. On the Log File Information page (Figure 125 on page 180), type the name of the log file you want to monitor in the **Log File Information** area.

   The file name must be fully qualified. Optionally, 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 246 on page 360). The file can also be a dynamic file name. For more information, see Appendix I, "Dynamic file name support," on page 631.

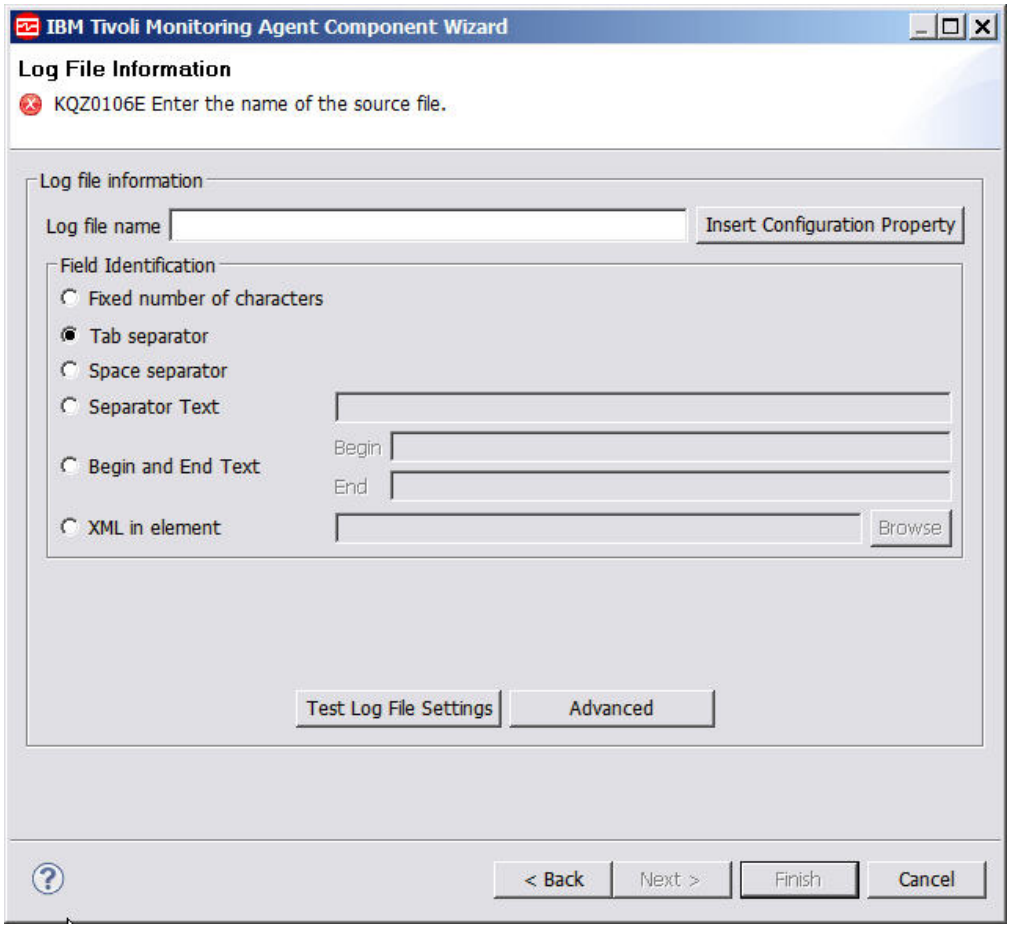


Figure 125. Log File Information page

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 126). If you use the browse functionality, the Agent Builder identifies all of the possible attributes of the record by looking at the child tags and their attributes.

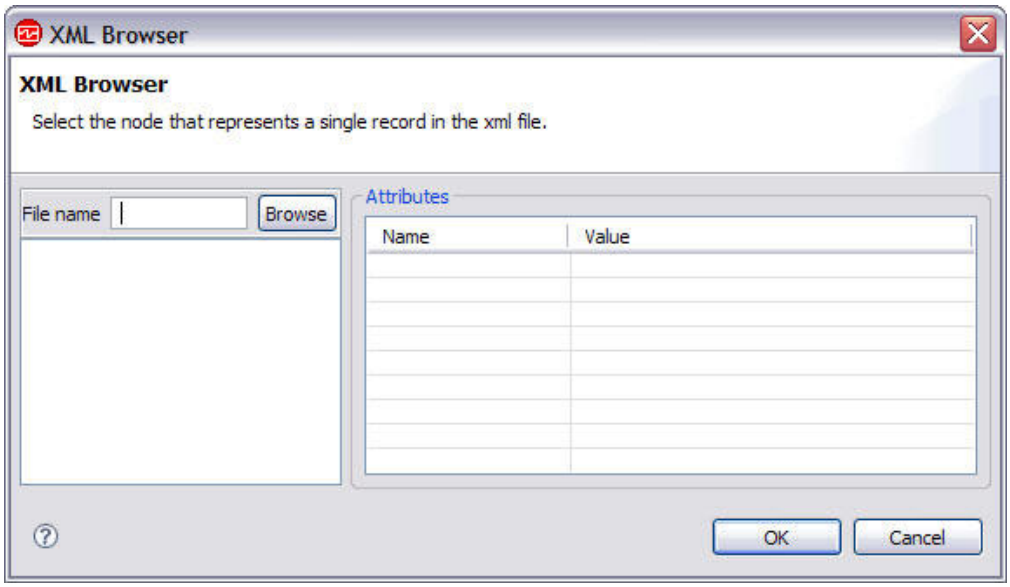


Figure 126. XML Browser window

Note: Unless you click **Advanced** and fill out the information in that window, the information that you fill out here assumes the following:

- Only one log file is monitored at a time.
- Each line of the log file contains all the fields necessary to fill the attributes to be defined.

6. *(Optional)* Click **Advanced** on the Log File Information page to do the following using the Advanced Data Source Properties page (Figure 127 on page 182):
   - Monitor more than one file at a time, or monitor files with different names on different operating systems or monitor file names 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 546. This function has been deprecated by the options available in the Event Information tab.

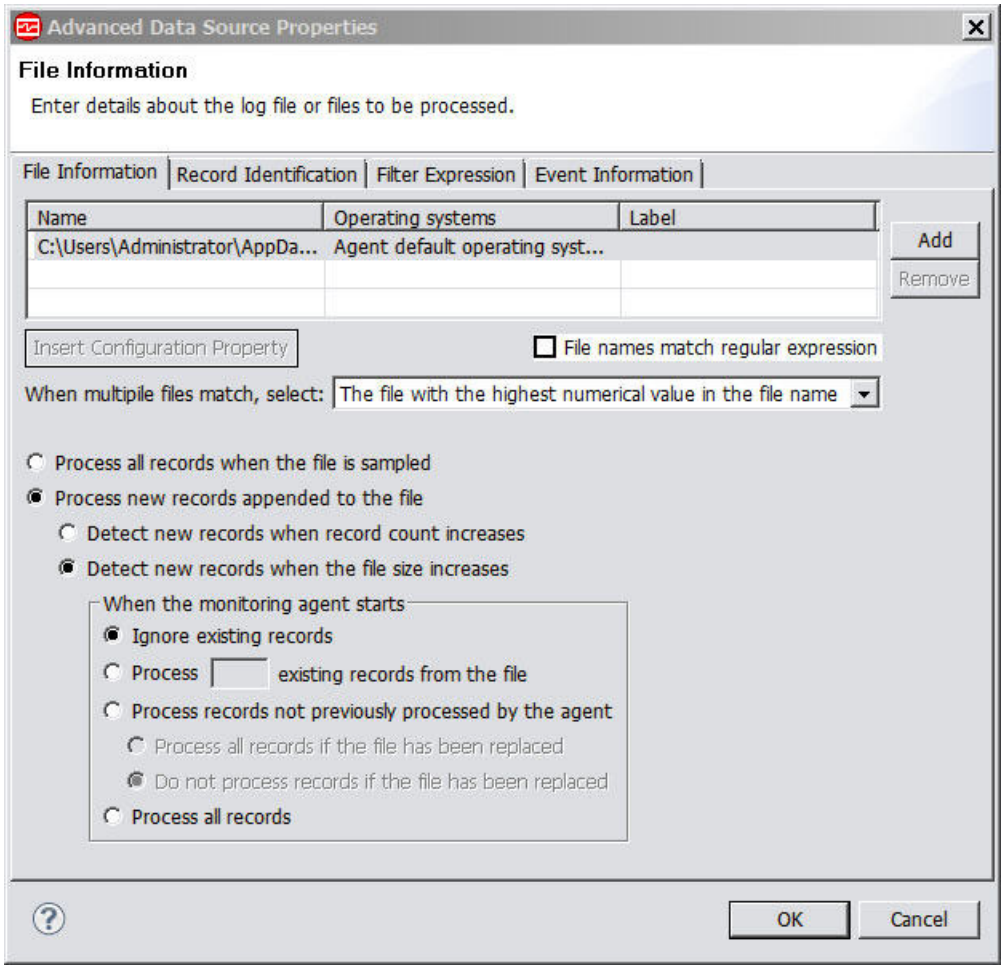


Figure 127. Advanced Data Source Properties page, File Information tab

a. To monitor more than one 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.

   (*Optional*) To select the operating systems on which each log file is to be monitored, do the following:

   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.

b. Select the **File names match regular expression** check box if the file name you are providing is a regular expression that will be used to find the file instead of being a file name. See Appendix G, "ICU regular expressions," on page 615 for more information.

   If you do not check this box, then the name needs to be an actual file name, or it must be a pattern following the rules for file-name patterns described in "Dynamic file name syntax" on page 631.

c. In the **When multiple files match** drop-down 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, and monitors updates to all of the files in parallel. Data from all files is intermingled during the data collection process. The best practice is 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 the best practice is to select **Entire record** in **Record Field Information** to define a single attribute. See Step 8 on page 187 for more information about specifying **Record Field Information** for attributes.

d. 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 a situation runs against the data source or a sample is taken. 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. A file must have at least two records to be processed correctly. With **Process new records appended to the file**, you can process new records 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 (the first record changes in any way), the entire file is processed and an event is produced for each record in the file.

Note: If you are appending records to an XML log file, the append records must contain a complete set of elements defined within the XML element you selected as Field Identification.

e. If you chose to process new records 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 or not the size of the file changes. This is useful when an entire log file is pre-allocated before any records are written to the file. Note that this 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 has been appended to a file in the typical way. There might be a brief delay recognizing that a monitored file has been replaced.

f. If you chose to detect new records based on the size of the file, 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 it ran. Events are produced for any records appended to the file since the last time the agent was running. Note that this involves a little extra overhead each time a record is added to the file.

g. If you chose to process records not previously processed by the agent, you can choose what to do when the agent starts and it appears that the existing file has been replaced. **Process all records if the file has been replaced** specifies the production of events for all records in the file if the current information about the monitored file and the information stored in the restart data do not match. Examples of mismatches include: the file name or file creation time has changed, the file size has decreased, the file last modification time is earlier than it was. **Do not process records if the file has been replaced** disables processing of any existing records in the file if the current information about the monitored file and the information stored in the restart data do not match.

h. Click the **Record Identification** tab (Figure 128 on page 185) to interpret multiple lines in the log file as a single logical record.

   Note: The **Record Identification** tab does not display if you select **XML in element** as the field identification on the Log File Information page.

   - **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** enables you to identify 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 either be **No text comparison** (the Maximum lines per record indicates a single logical record), **Identify the beginning of record** (which marks the start of the single logical record), or **Identify the end of record** (which 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** enables you to identify a pattern 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.

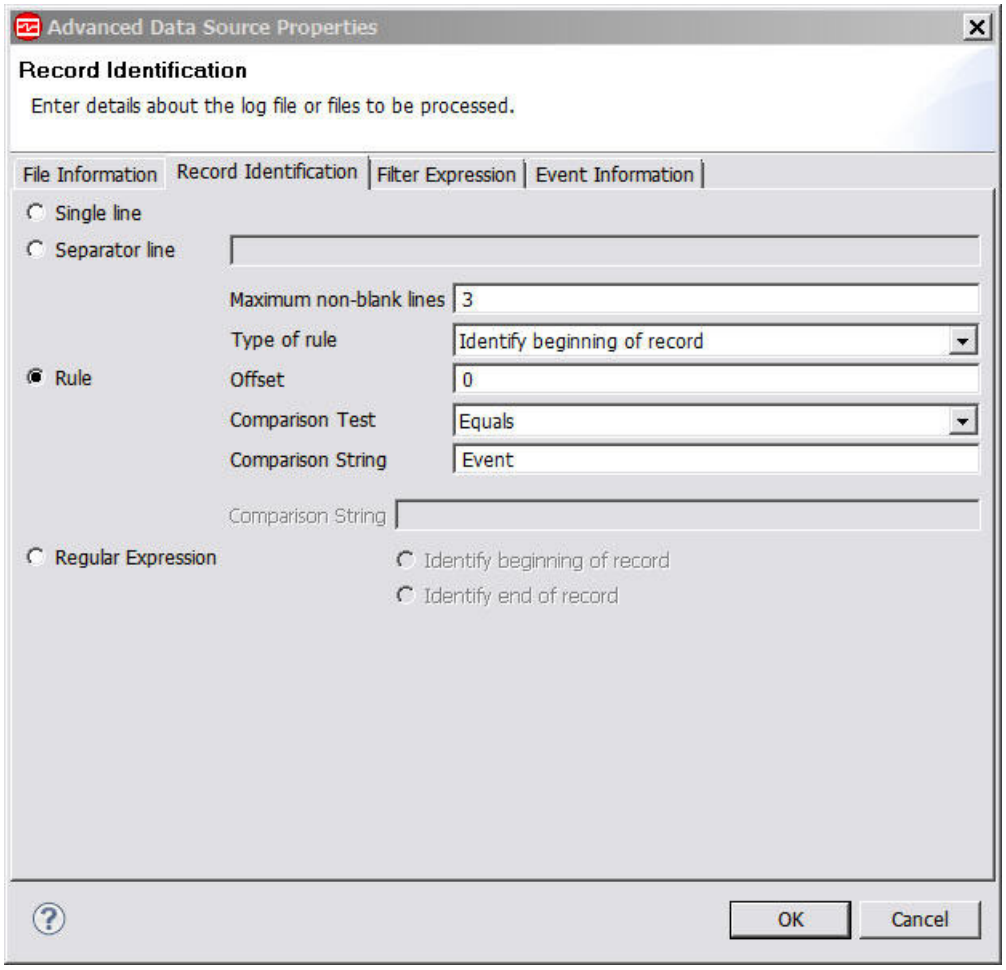


Figure 128. Advanced Data Source Properties page, Record Identification tab

i. If you did select **Process all records when the file is sampled** in Step 6d, click the **Filter Expression** tab (Figure 129 on page 186) to filter the data that will be returned as rows based on the values of one or more attributes and/or configuration variables. If you selected **Process new records appended to the file** in Step 6d you cannot create a filter expression. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66

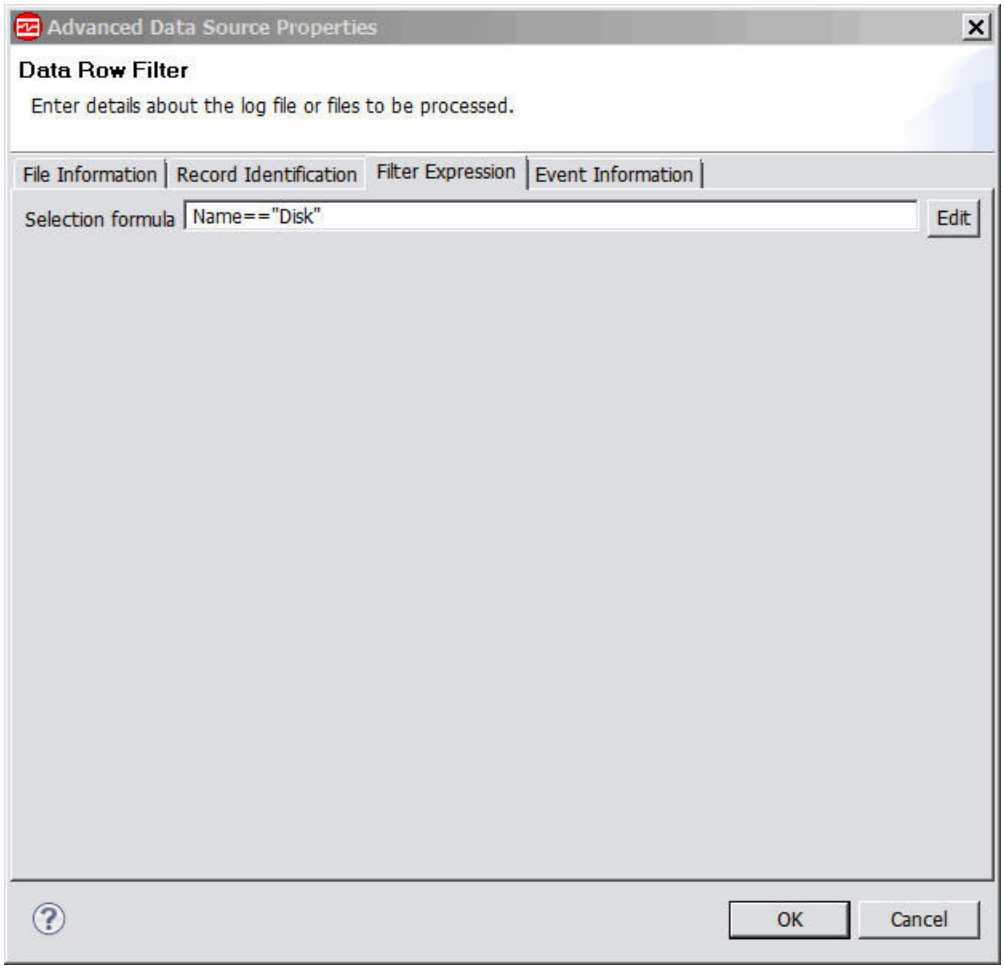


Figure 129. Advanced Data Source Properties page, Filter Expression tab

j. If you selected **Process new records appended to the file** in Step 6d, click the **Event Information** tab Figure 130 on page 187 to select event filtering and summarization options. For more information, see (Chapter 34, "Event filtering and summarization," on page 405).

Note: The Summary tab may be present if the agent was created with an earlier version of Agent Builder. The summary tab has now been deprecated by the Event Information tab

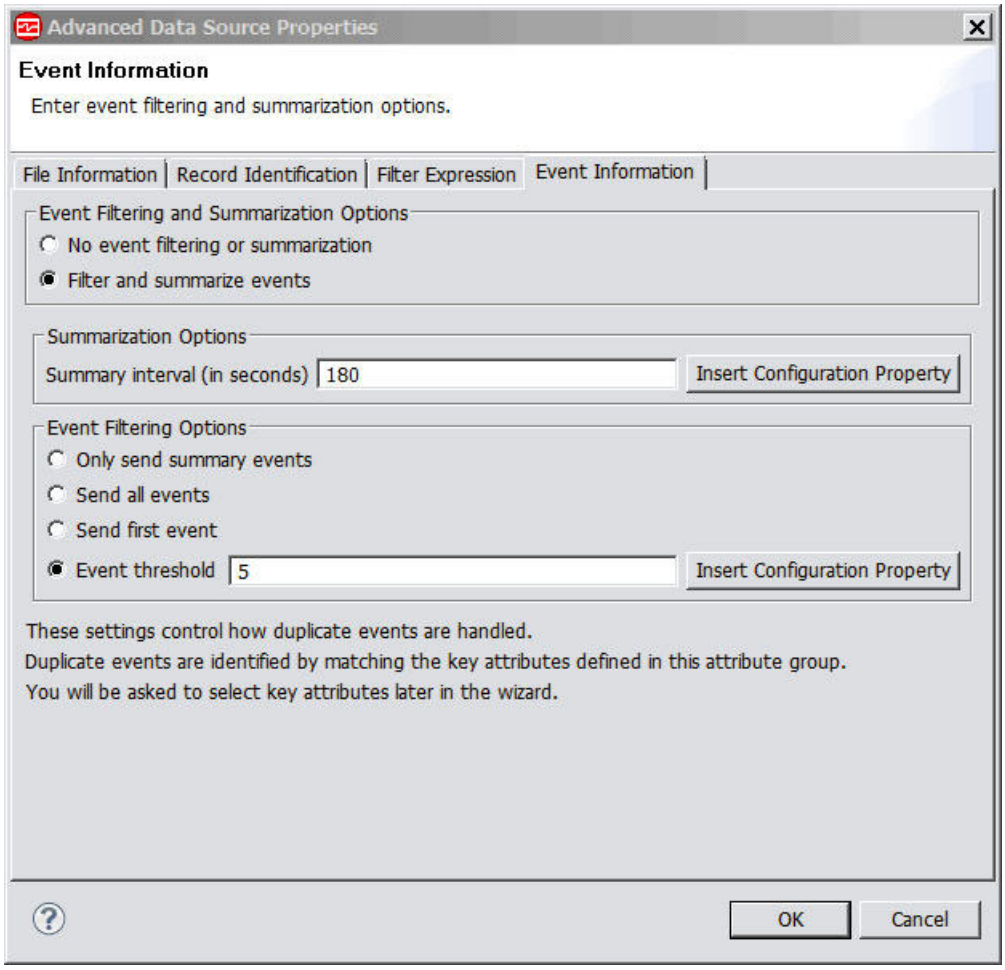


Figure 130. Advanced Data Source Properties page, Event Information tab

7. On the Log File Information page (Figure 125 on page 180), after you select the options for the log source, you can click **Test Log File Settings** to start and test the agent. For more information about testing, see "Testing" on page 195.
8. If you did not use the test functionality in Step (7) and you typed the log file name in the **Log File Information** area of the Log File Information page (Figure 125 on page 180), do the following:
   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 that is at the default minimum Tivoli Monitoring version of 6.2.1 or later, a Log File Status attribute group is automatically included. For more information about the Log File Status attribute group, see "Log File Status attribute group" on page 576.

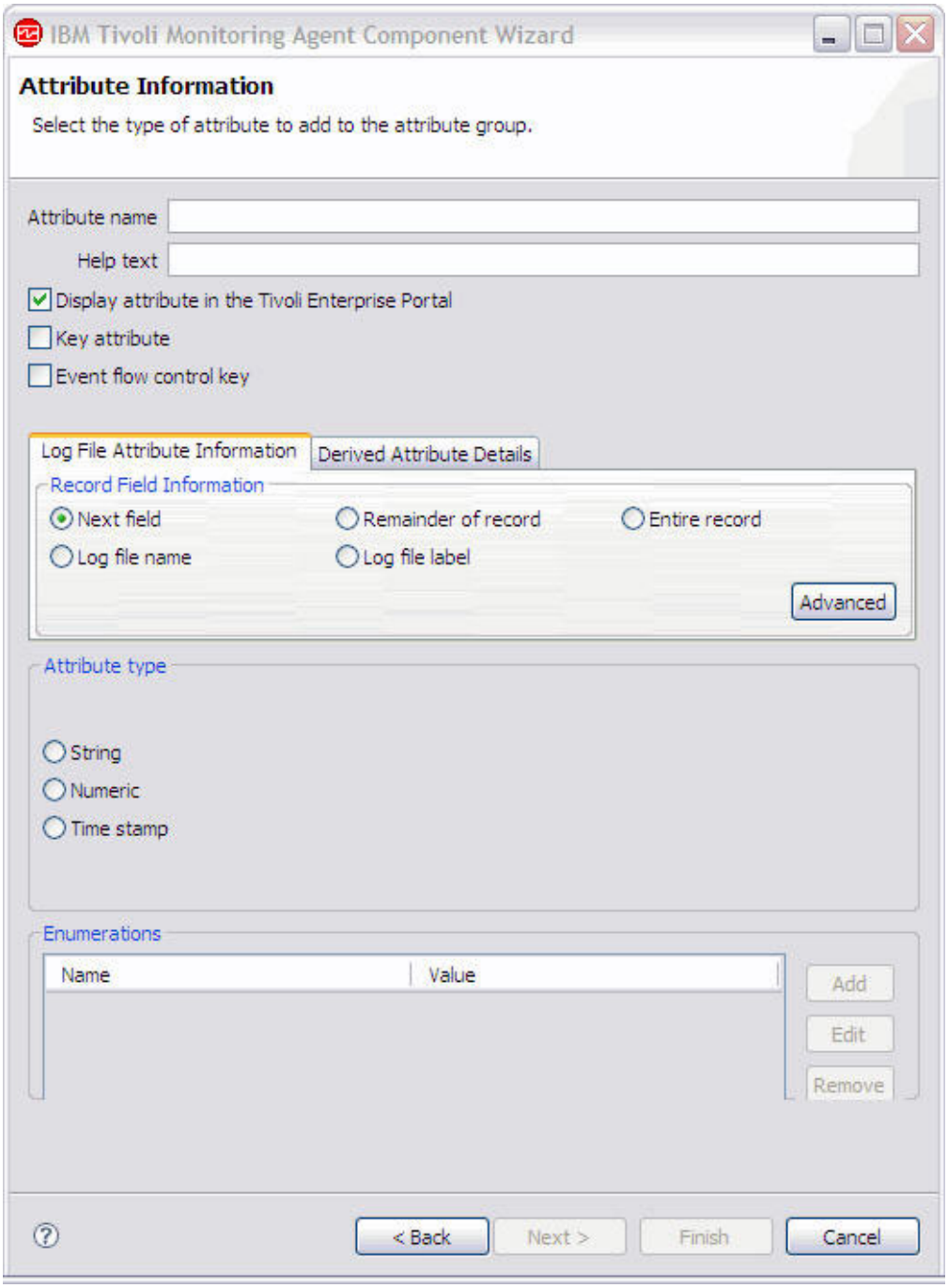


Figure 131. Attribute Information page

Along with the fields that are applicable to all of the data sources (Table 6 on page 56), 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, 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 have been parsed. This 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 assigned to the file on the advanced panel (above).

Note: Use the **Derived Attribute Details** tab only if you want a derived attribute, and not an attribute directly from the log file.

9. If you click **Advanced** in the **Record Field Information** area, the Advanced Log File Attribute Information page (Figure 132 on page 190) is displayed.

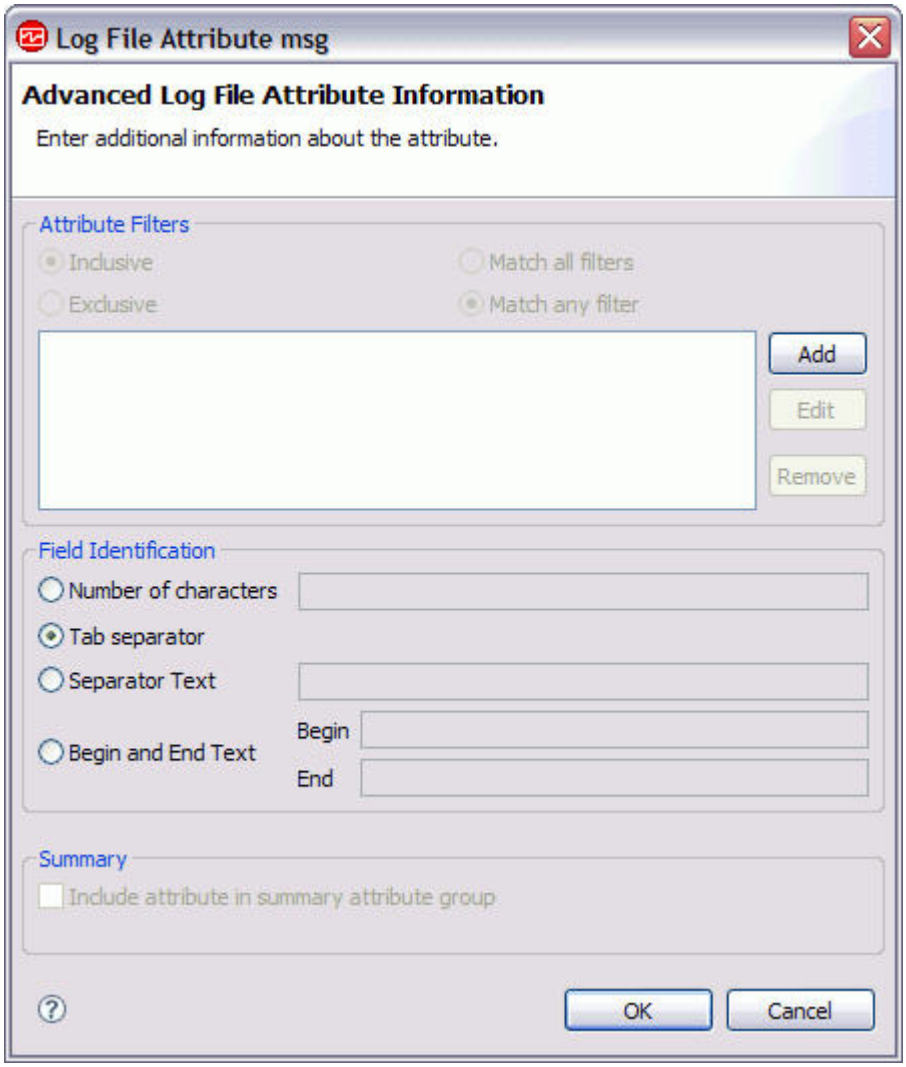


Figure 132. Advanced Log File Attribute Information page

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 set 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 set succeeds, the record is rejected and is not output.
   - **Match all filters** indicates that all filters defined to the filter set must match the attribute record in order for the filter set to succeed.
   - **Match any Filter** indicates that if any of the filters defined to the filter set match the attribute record, the filter set succeeds.

b. Use **Add**, **Edit**, and **Remove** to define the individual filters for an attribute filter set.

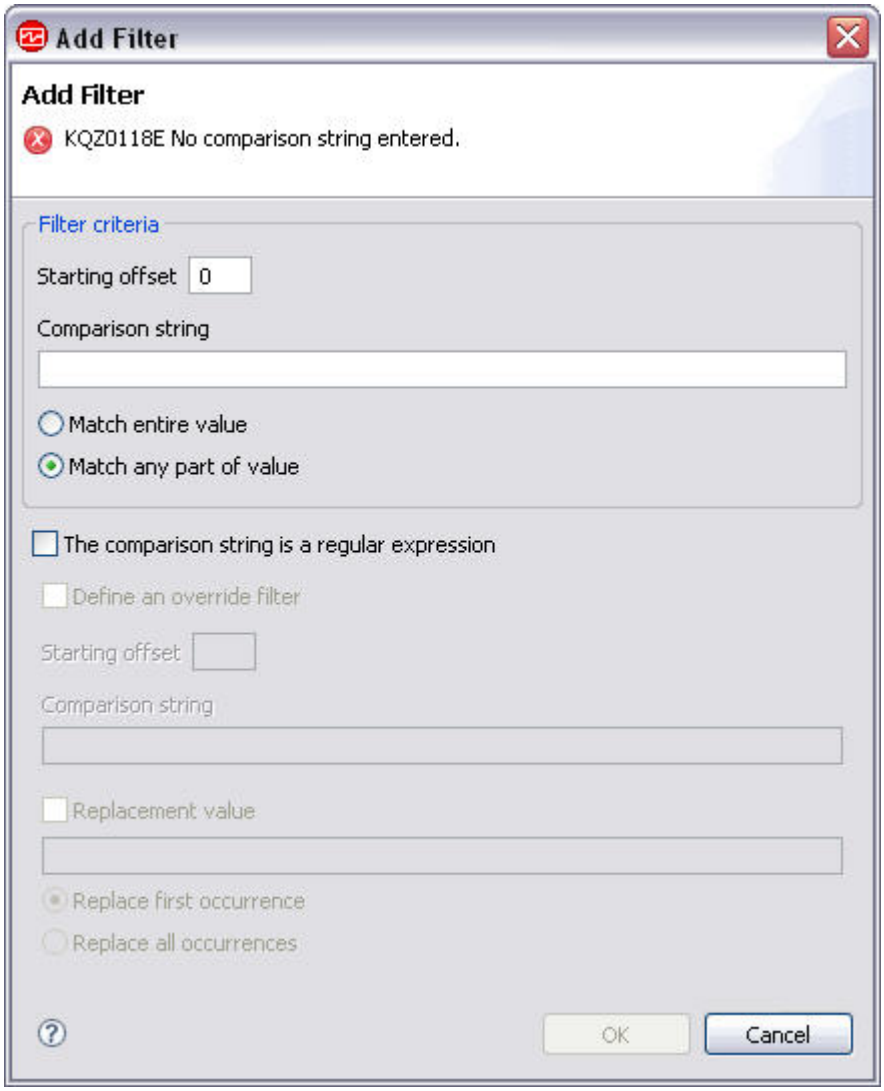


Figure 133. Add Filter window

To add a filter, perform the following steps:

1) Click **Add**, and complete the options in the Add Filter window (Figure 133) 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 will be used by the agent to filter the data read from the file. Depending on whether you choose for the filter to be inclusive or exclusive, the records that match the filter pattern will either be the only records returned to the Tivoli Enterprise Portal, subject to any override filters specified.

- **Match entire value** checks for an exact occurrence of the comparison string in the attribute string starting from the starting offset position.
- **Match any part of value** checks for the comparison string anywhere in the attribute string starting from the starting offset position.

b) **The comparison string is a regular expression** indicates 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 being examined matches the specified pattern.

To effectively use regular expression support, you must be familiar with the specifics of how ICU implements regular expressions, which is not identical to how regular expression support is implemented in Perl, grep, sed, Java regular expressions, and other implementations that you might have worked with in the past. See Appendix G, "ICU regular expressions," on page 615 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. (See Chapter 15, Step 9a for more details on Inclusive versus Exclusive, and the examples in Step 2 of this set of procedures). 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 will be used by the agent to filter the data read from the file. Depending on whether you choose for the filter to be inclusive or exclusive, the records that match the filter pattern are be eliminated from the records returned to the Tivoli Enterprise Portal, or they will be the only records returned to the Tivoli Enterprise Portal.

d) **Replacement value** can be used to alter the raw attribute string with a new value. See Appendix G, "ICU regular expressions," on page 615 for more details about special characters that can be used.

e) **Replace first occurrence** replaces the first occurrence matched by the comparison string with new text.

f) **Replace all occurrences** replaces all occurrences matched by the comparison string with new text.

2) Click **OK**.

Example 1

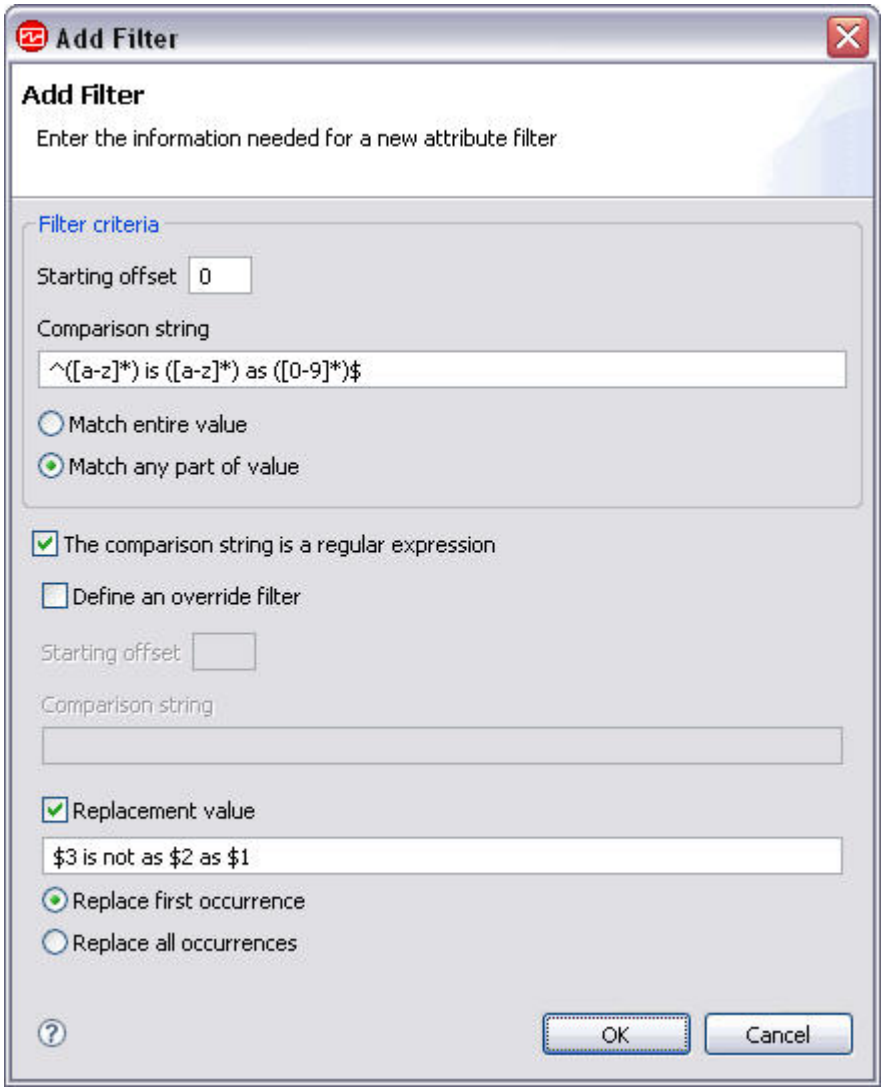


Figure 134. Add Filter example 1

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`.

Example 2

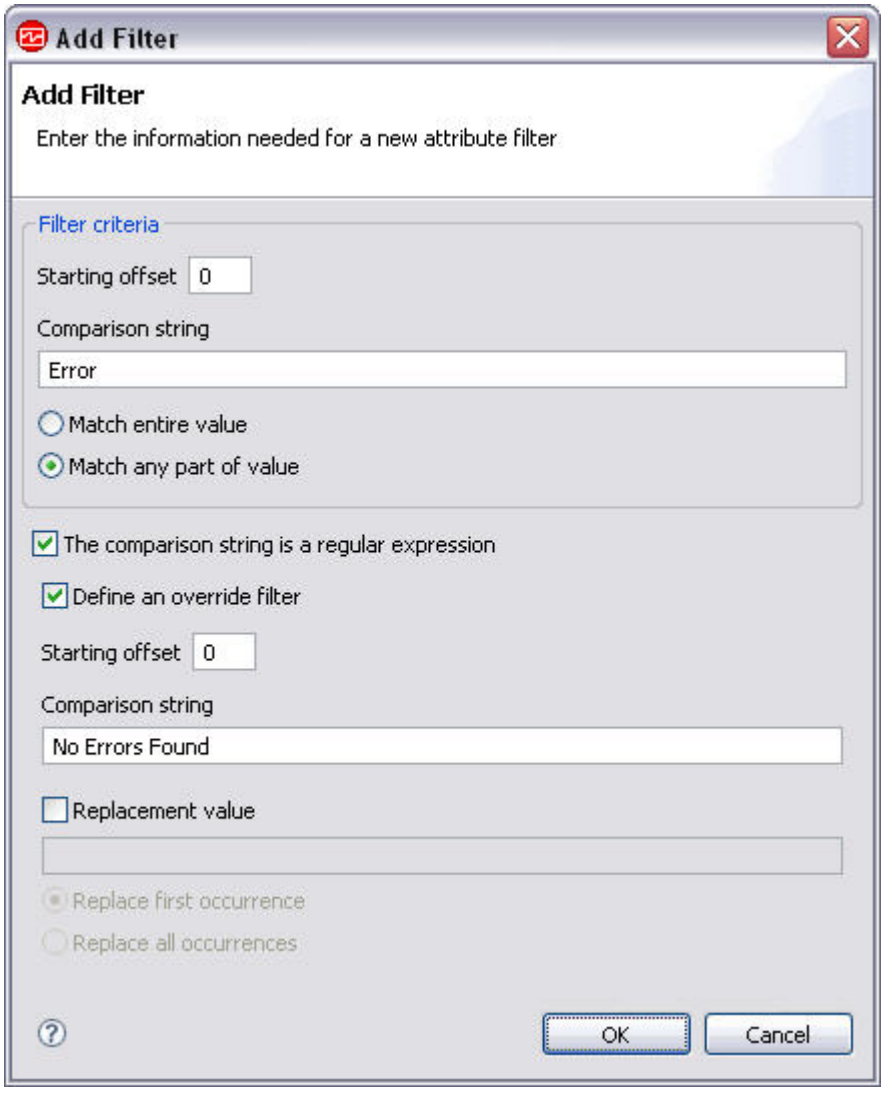


Figure 135. Add Filter example 2

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.

c. In the **Field Identification** section of the Advanced Log File Attribute Information page (Figure 132 on page 190), 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 you want to use.
   - **Begin and End Text** Enter both **Begin** text and **End** text.

d. In the **Summary** section of the Advanced Log File Attribute Information page (Figure 132 on page 190), 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.

e. Click **OK**.

10. If you used the test functionality in Step 7, 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. See "Selecting key attributes" on page 27 for more information.
11. Do one of the following steps:
    a. If you are using the New Agent Wizard, click **Next**.

    —OR—

    b. Click **Finish** to save the data source and open the Agent Editor.

Note: When a Log File attribute group is added to an agent that is at the default minimum Tivoli Monitoring version of 6.2.1 or later, a Log File Status attribute group is automatically included. For more information about the Log File Status attribute group, see "Log File Status attribute group" on page 576.

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test Log File Settings** on the Log File Information page (Figure 125 on page 180).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test Log File Settings** . For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37

After you click **Test Log File Settings** in one of the previous two steps, the Parse Log window Figure 136 is displayed

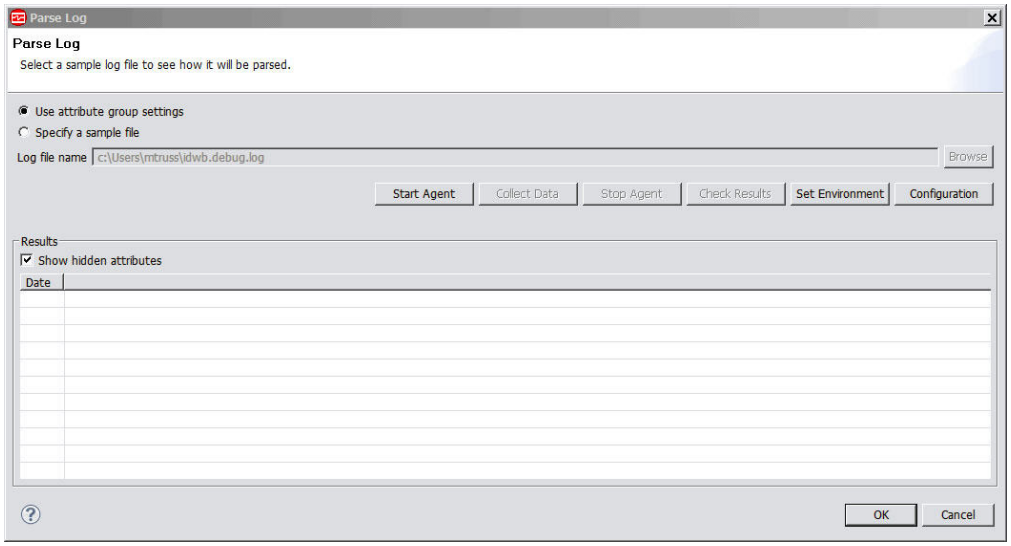


Figure 136. Parse Log window

To start and test your agent use the following procedure

1. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see "Configuration prior to testing" on page 374.
2. Click **Start Agent**. A window opens indicating that the Agent is starting.
3. When the agent starts, it monitors the configured logfile for new records
4. To test your agent's data collection, generate new records in the monitored logfile. When new records are added to the logfile the agent parses them according to its configuration and updates the corresponding attribute values in its internal cache.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The Parse Log window collects and displays any new attribute values in the agent's cache since it was last started. An example data collection is shown in Figure 137

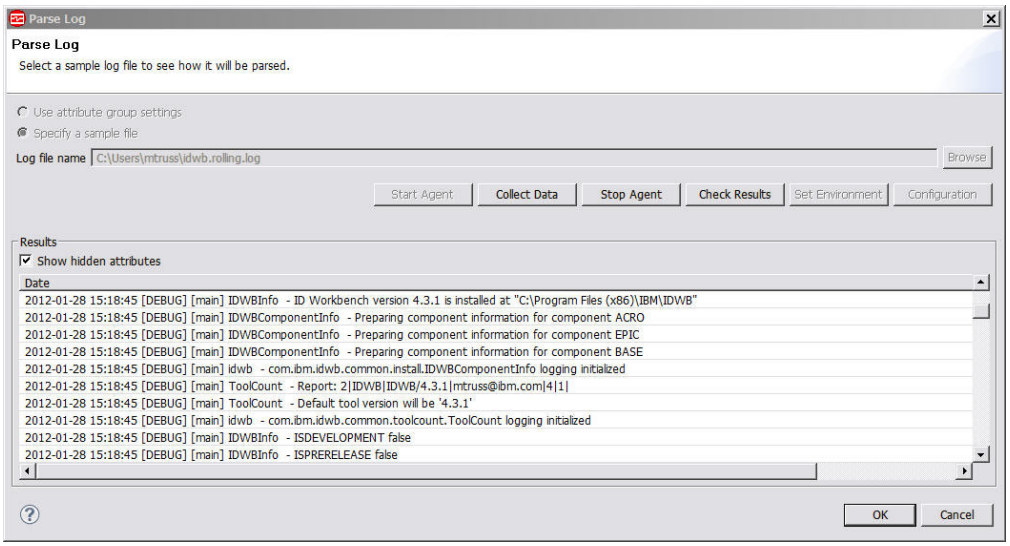


Figure 137. Parse Log window showing parsed logfile attribute values

6. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. An example is shown in Figure 138. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534

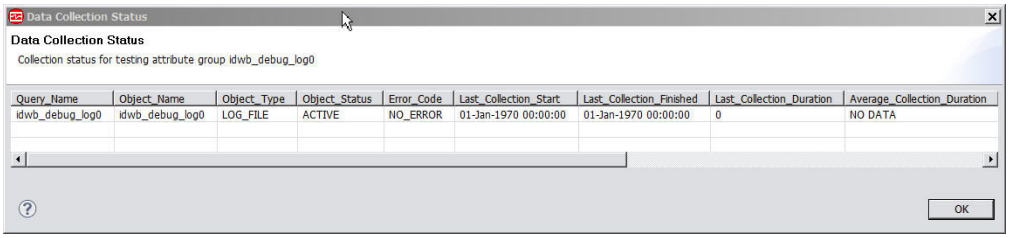


Figure 138. Data Collection Status window

7. The agent can be stopped by clicking **Stop Agent**.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 17. Monitoring an AIX Binary Log

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 ouput: '-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 548.

Use the following procedure to collect data from the AIX Binary log.

1. On the Agent Initial Data Source page (Figure 151 on page 212) or the Data Source Location page, click **Logged data** in the **Monitoring Data Categories** area.

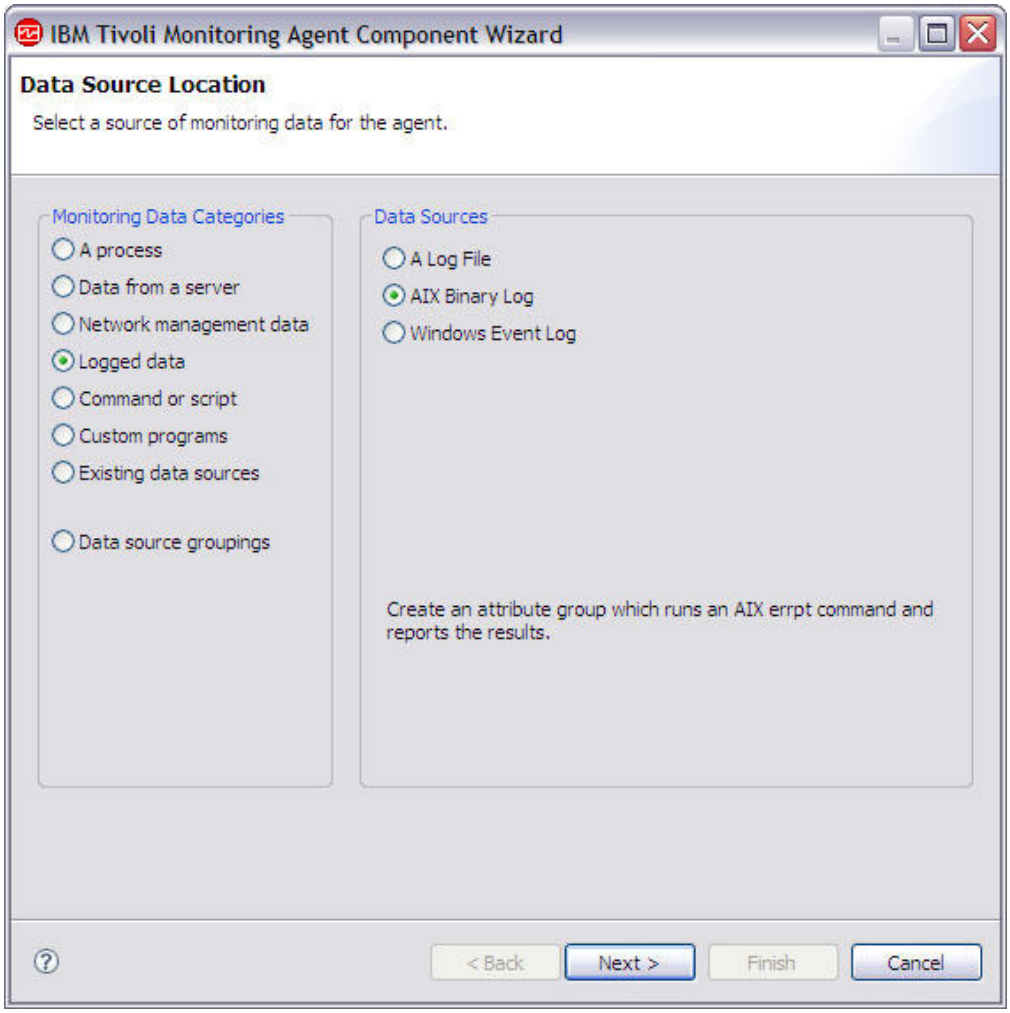


Figure 139. Adding an AIX Binary Log

2. In the **Data Sources** area, click **AIX Binary Log**.
3. Click **Next**.
4. On the Binary Log Information page Figure 140 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 ouput: '-a', '-A', or '-g'.

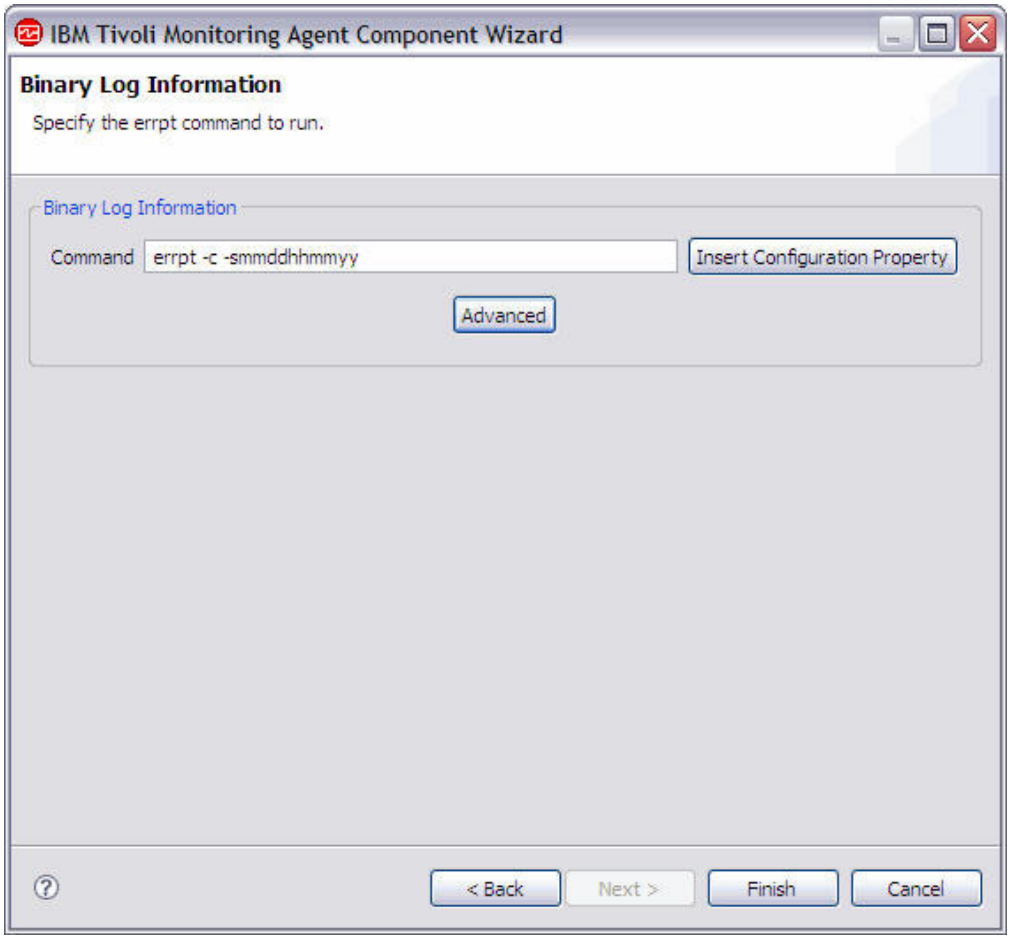


Figure 140. Binary Log Information page

5. (Optional) Click **Advanced** to select event filtering and summarization options. For more information, see "Controlling duplicate events" on page 406.
6. Do one of the following steps:
   a. If you are using the New Agent Wizard, click **Next**.
      —OR—
   b. Click **Finish** to save the data source and open the Agent Editor.

For more information about the attribute groups for AIX binary error logs, see "AIX Binary Log attribute group" on page 548.

Chapter 18. Monitoring a Windows Event Log

You can collect data from the Windows event log by using the type, source, or ID of events to filter the log events that the Windows system has 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, specifying **Error** as the event type, the agent matches all events logged to the Application log where the event type value is `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 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. See Step 6 on page 203 for more information about enabling the agent to process events generated while the agent is shut down.

Use the following procedure to collect data from the Windows event log:

1. On the Agent Initial Data Source page (Figure 141 on page 202) or the Data Source Location page, click **Logged Data** in the **Monitoring Data Categories** area.

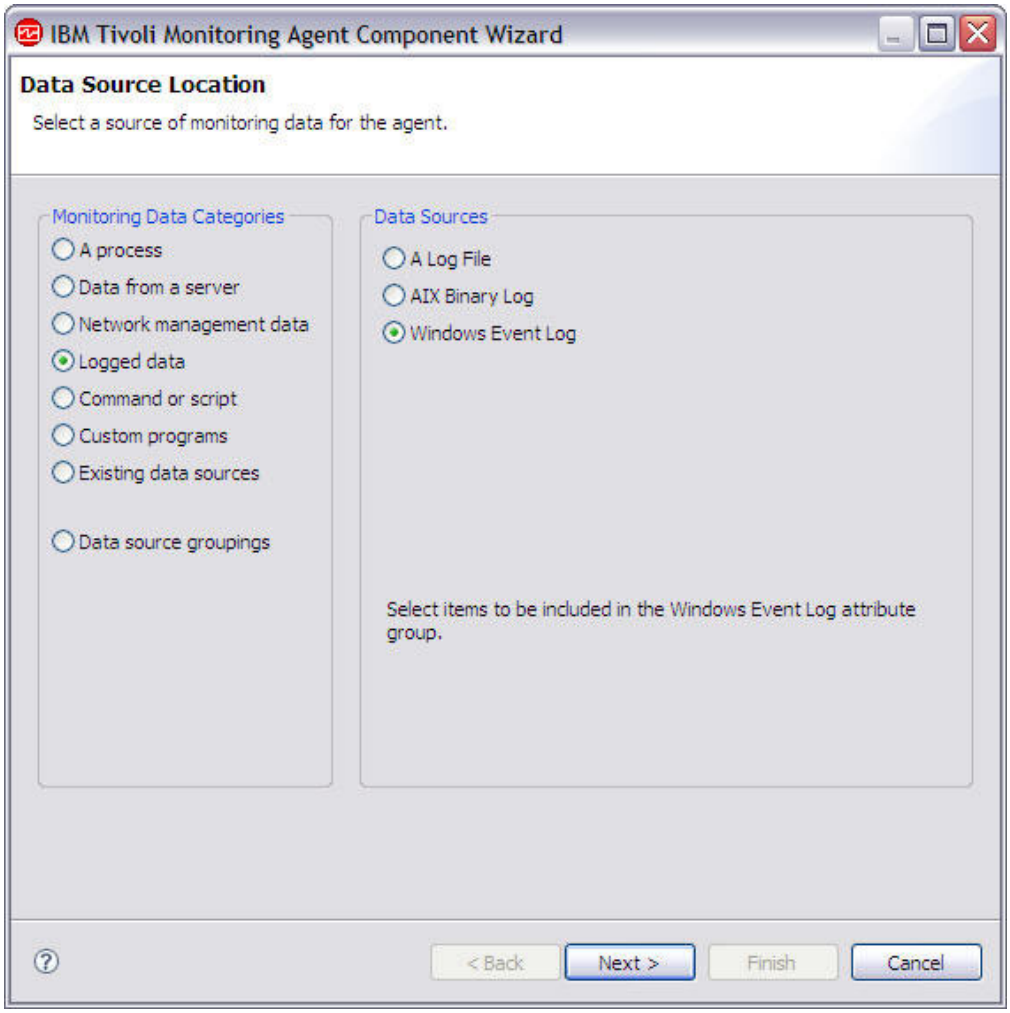


Figure 141. Adding a Windows Event log

2. In the **Data Sources** area, click **Windows Event Log**.
3. Click **Next**.
4. On the Windows Event Log page (Figure 142 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
   ```

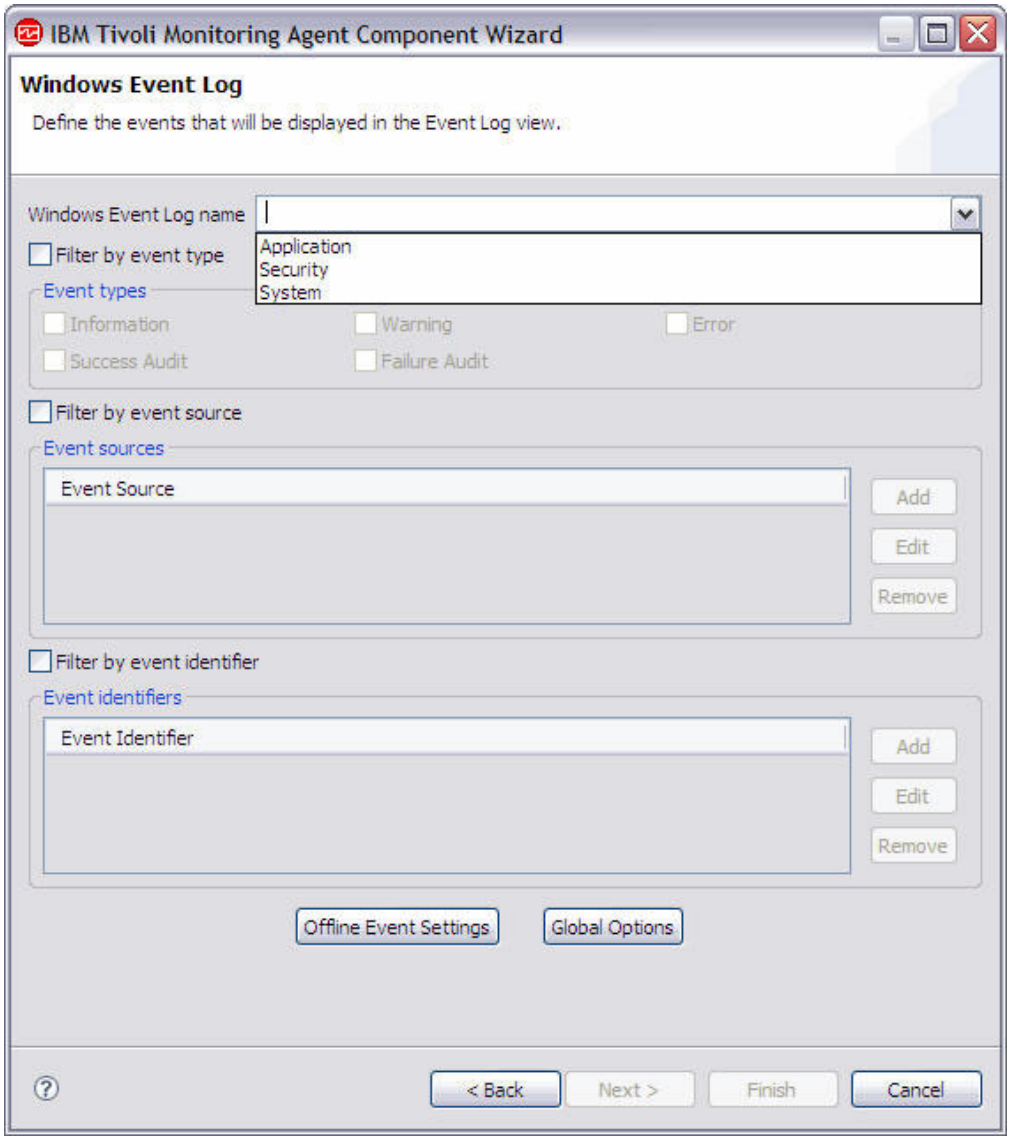


Figure 142. Windows Event Log page

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 206
   - "Filtering by event identifier" on page 208

   Note: You must select at least one of these filter criteria.
6. If you want the agent when it restarts to process log events that are generated while the agent is shut down, click **Offline Event Settings** (Figure 143 on page 204).

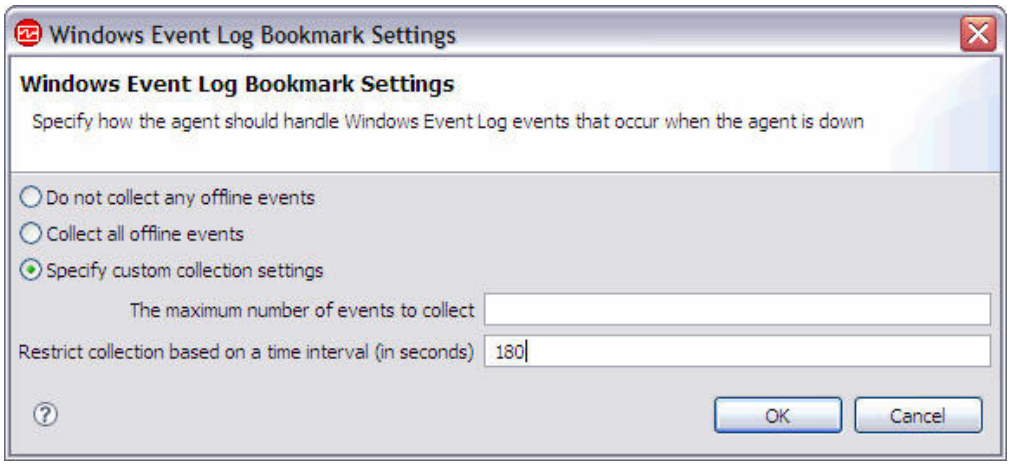


Figure 143. Windows Event Log Bookmark Settings window

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 generated while the agent is shut down are not processed. This option is the default option.
- **Collect all offline objects**: All events 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 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 processed is either the last 100 events generated before the agent starts, or any event generated within 30 seconds of the agent starting, depending on which variable 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 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 processed for the event log. This file is located in the `%CANDLE_HOME%\tmaitm6\logs` directory and is used when the agent is restarted to process old events generated while the agent was shut down.

7. If you want to set global options for the data source, click **Global Options** (Figure 144 on page 205).

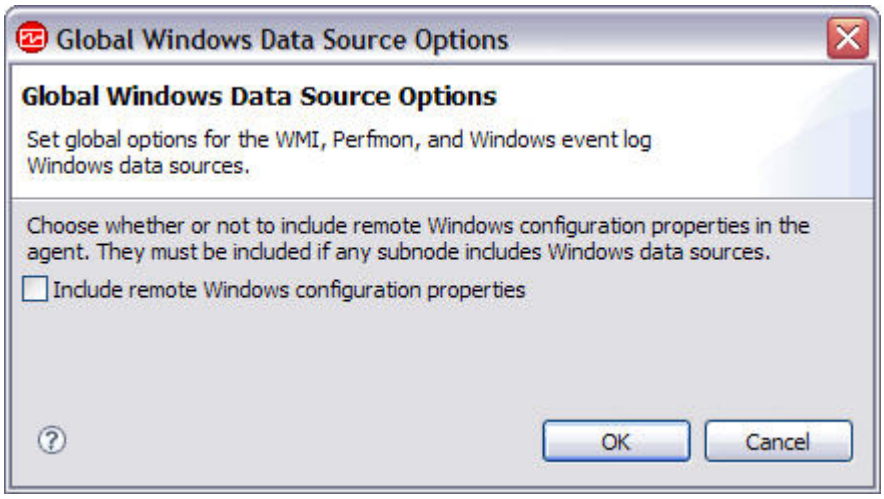


Figure 144. Global Windows Data Source Options window

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 370.

8. After specifying the filter and clicking **OK**, on the Windows Event Log page, do one of the following steps:
   a. If you are using the New Agent Wizard, click **Next**.

      —OR—

   b. Click **Finish** to save the data source and open the Agent Editor. The name of the new Windows Event Log is displayed on the Agent Editor Data Source Definition page.

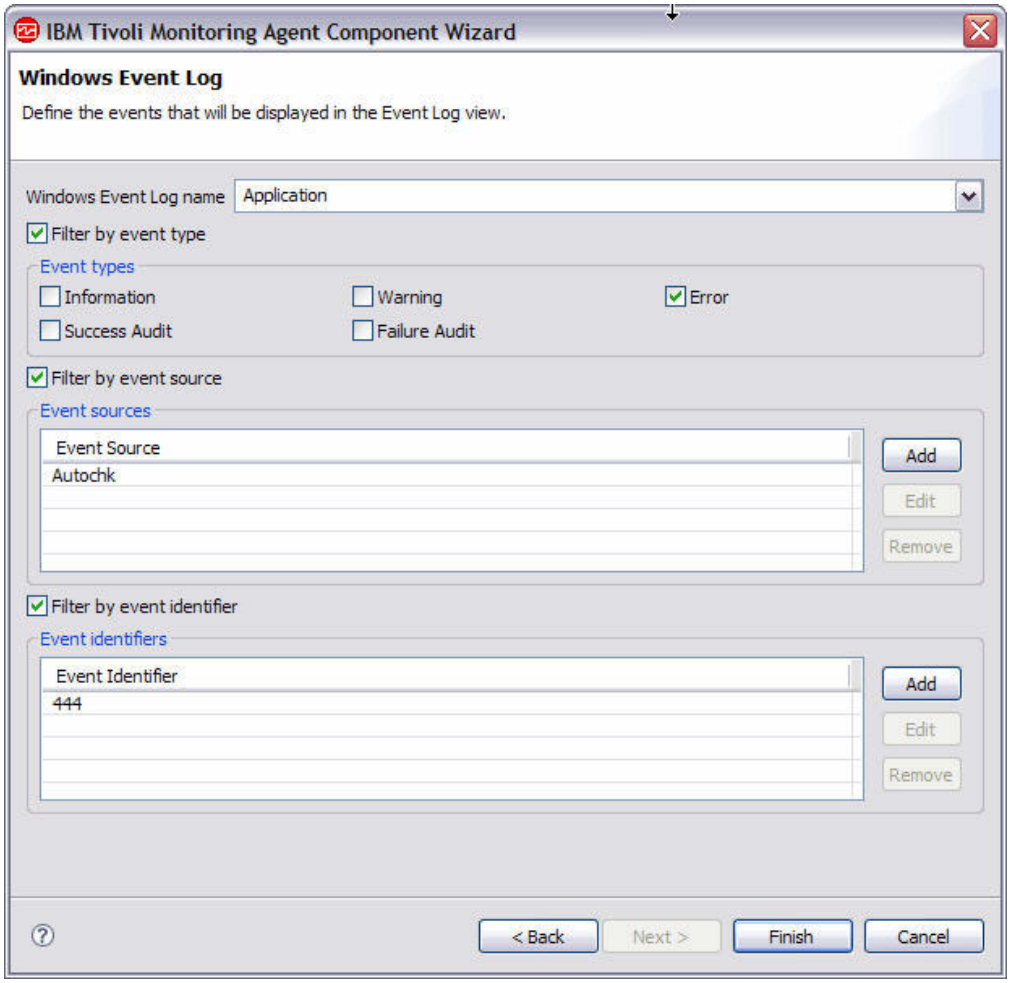


Figure 145. Windows Event Log page

For information about Windows remote connection configuration for Windows Event Log data sources, see "Configuring a Windows remote connection" on page 370.

Filtering by event type

To filter by event type, select **Filter by event type** and one or more type boxes. You can select any of the following types to act as filters:

- **Information**
- **Warning**
- **Error**
- **Success Audit**
- **Failure Audit**

Filtering by event source

To filter by event source, use the following procedure:

1. Select **Filter by event source** and click **Add** in the **Event sources** area of the Windows Event Log page (Figure 145). The Event Source window is displayed (Figure 146 on page 207).

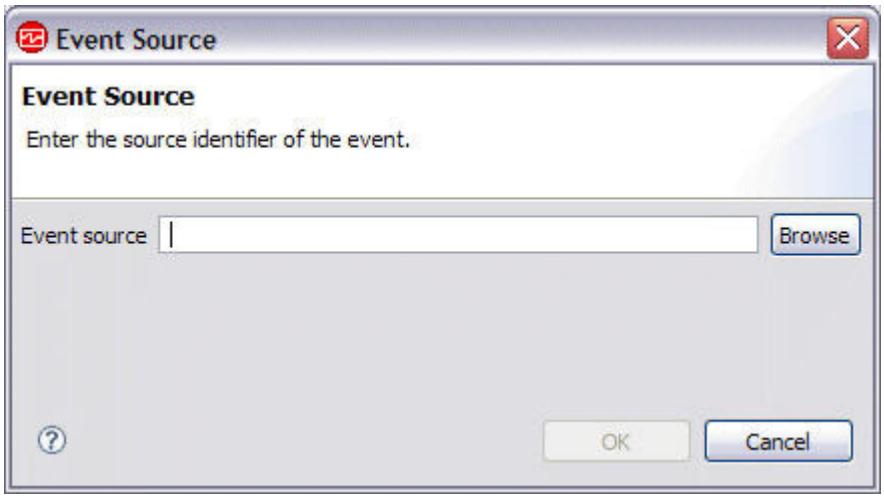


Figure 146. Event Source window

2. You can browse to select one or more names in the list, or type the event source name and click **OK**.

   If you prefer to browse to find an event source, select **Browse** to select one from the list (Figure 147) and click **OK**. The name you selected is displayed in the Event Source window.

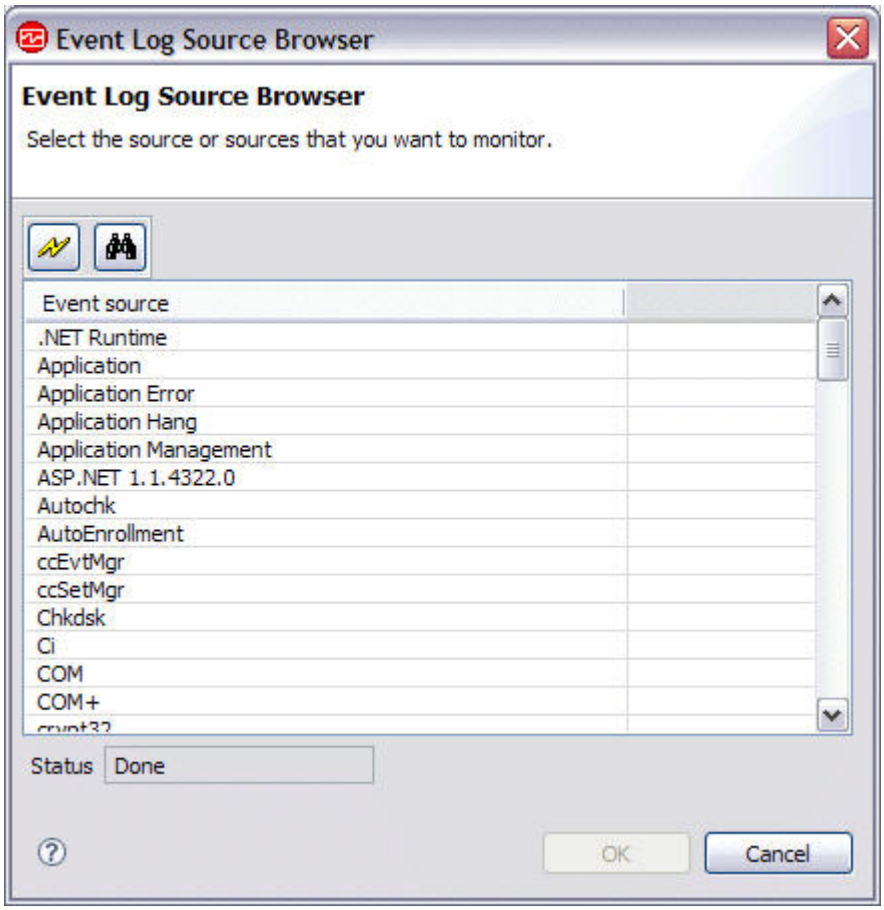


Figure 147. Event Log Source Browser window

Notes:

a. To sort the list of event sources, click on 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.

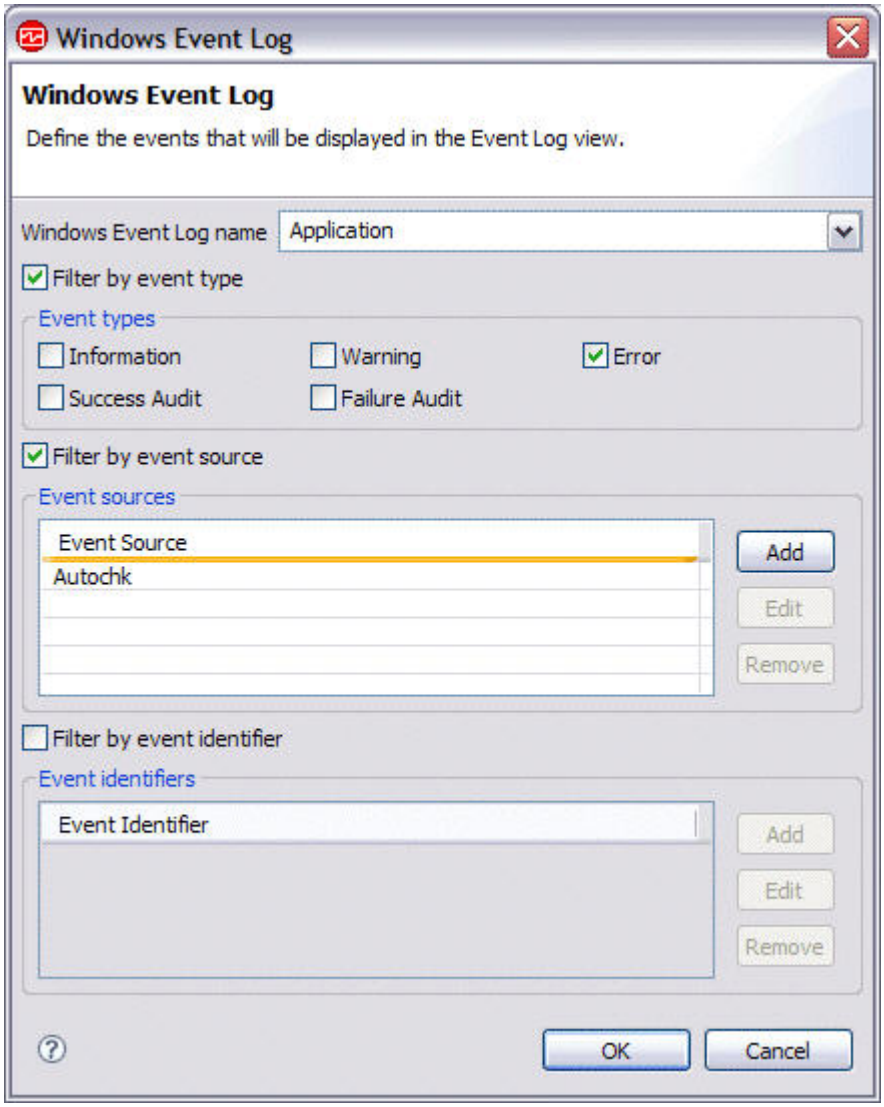


Figure 148. Windows Event Log window

3. Click **OK** to see the new event source filter displayed in the Event sources list in the Windows Event Log window (Figure 148).

Filtering by event identifier

To filter by event identifier, use the following 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 149 on page 209).

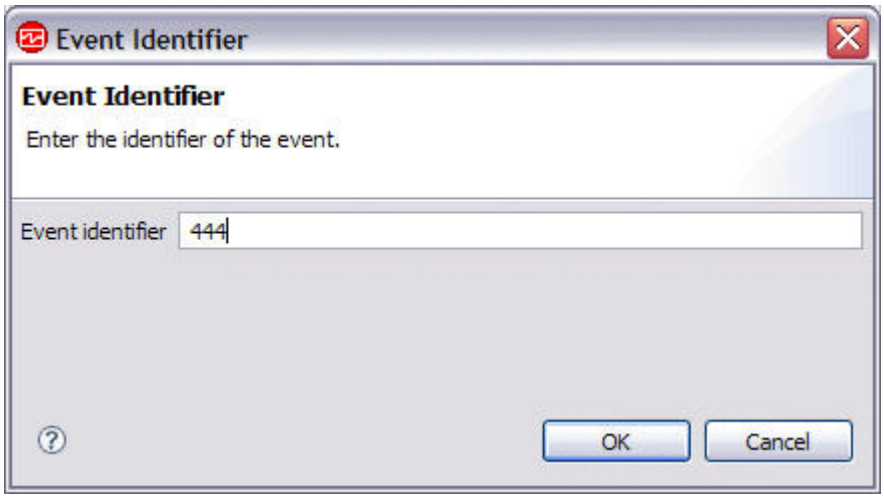


Figure 149. Event Identifier window

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 window (Figure 150 on page 210).

   Note: Each event identifier must be defined individually.

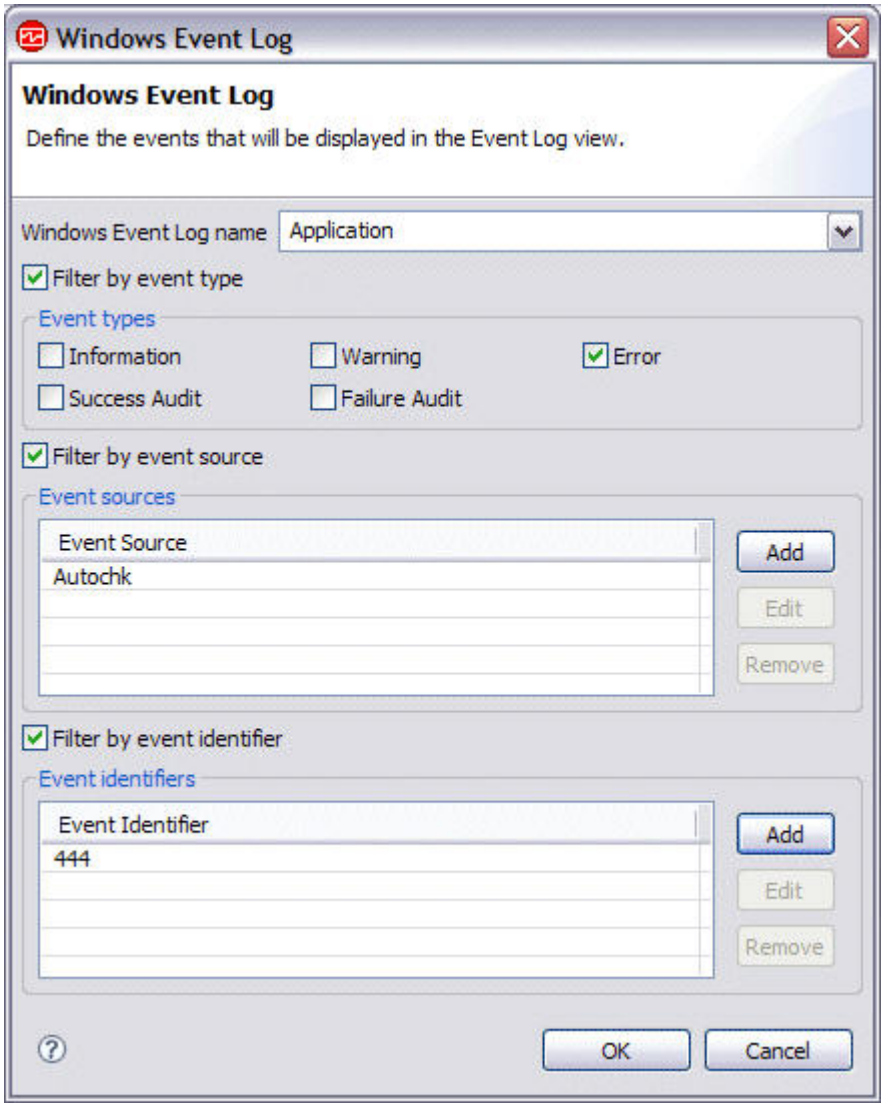


Figure 150. Windows Event Log window

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

You can test whether an application or system being monitored is available by using a *command return code*. A command return code is a user created script, executable file, query, or system command that is an application-specific mechanism for determining whether the application or system being monitored is available. The agent runs the specified command and determines the state of the application or system being monitored by examining the return code.

The command must present a unique return code for each descriptive state and define a message to be used by the agent for each of these return codes. The command may use environment and configuration variables within the user created script, executable file, query, or system command. The command may 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.

Use the following procedure to monitor a command return code:

1. On the Agent Initial Data Source page (Figure 151 on page 212) or the Data Source Location page, click **Command or script** in the **Monitoring Data Categories** area.

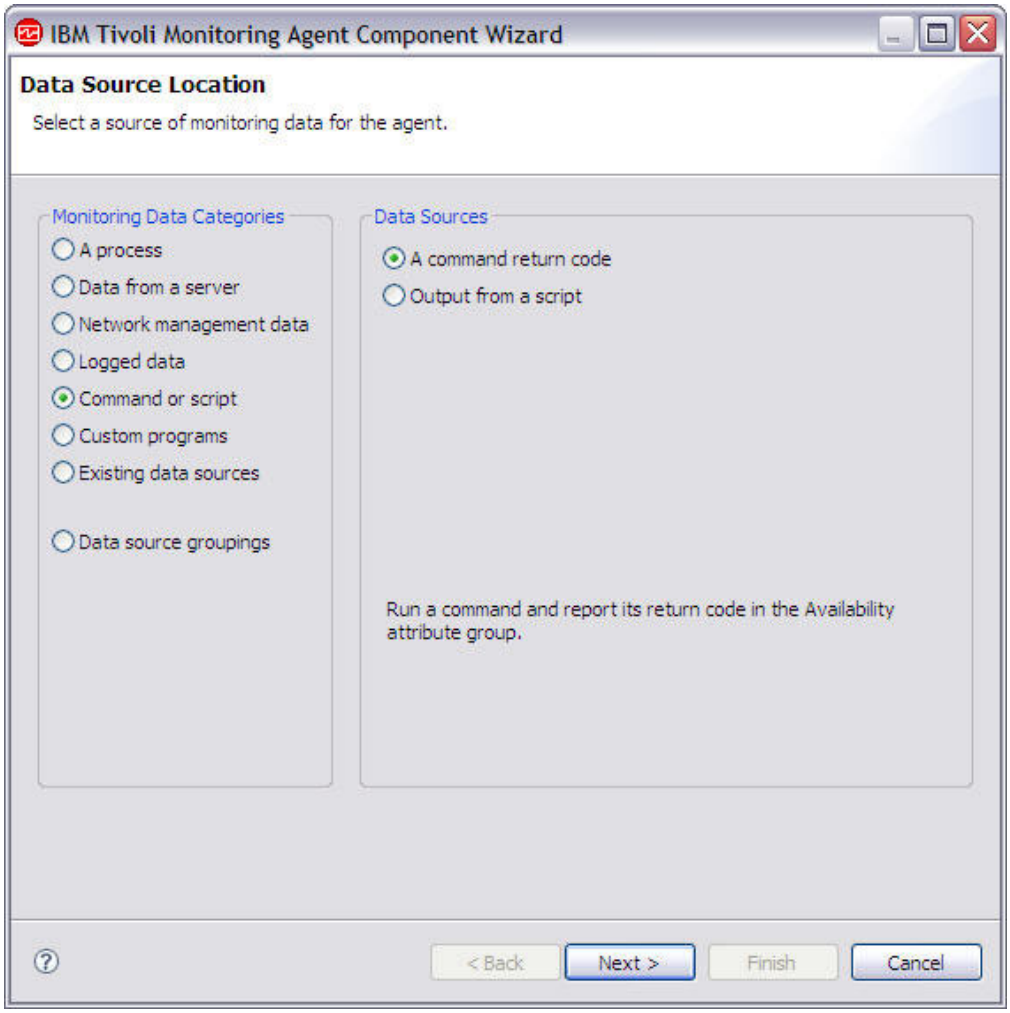


Figure 151. Adding data from a command

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.

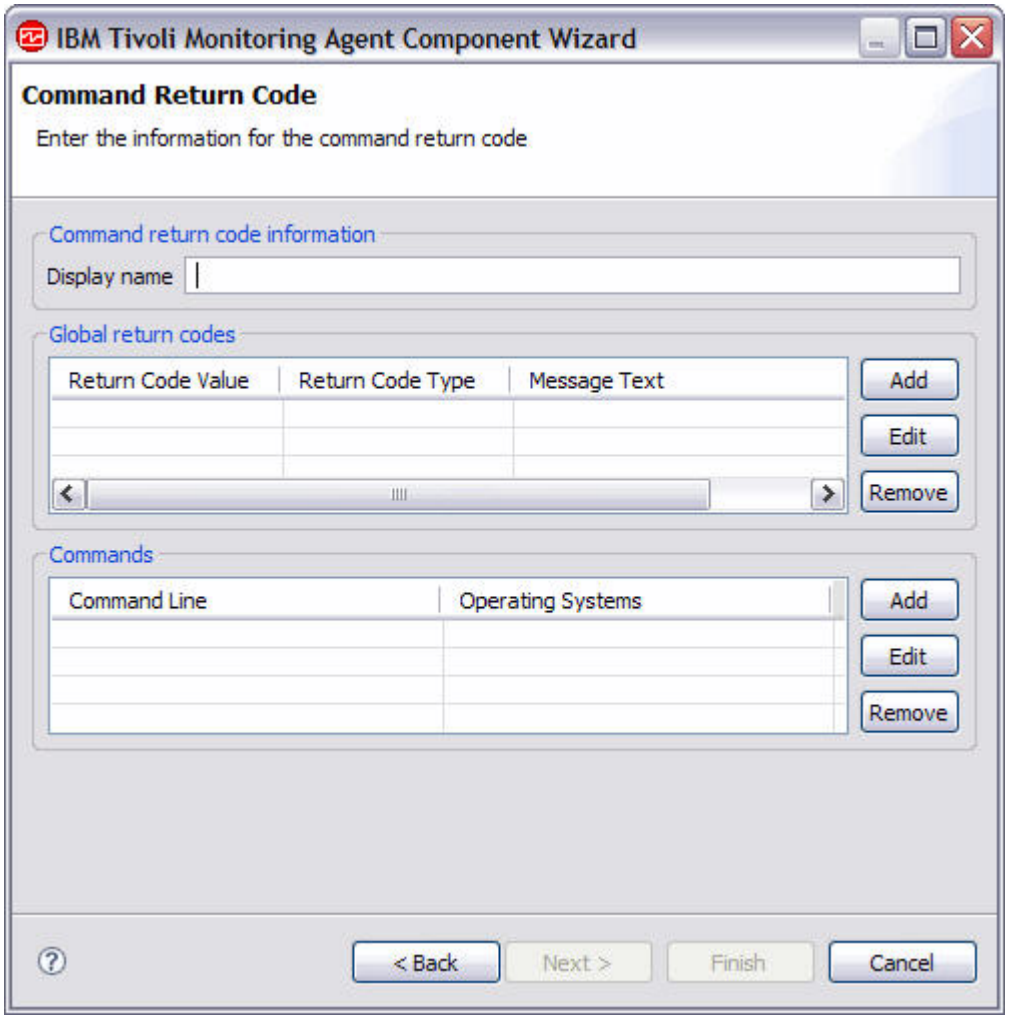


Figure 152. Command Return Code page

5. In the **Commands** area of the Command Return Code page, define and describe command lines that you want your command return code to use.

   Note: Define a command for each operating system supported by the agent. The commands can be shared, but the total set of operating systems picked for all of the commands must equal the set of operating systems the agent is supporting.

   a. Click **Add** in the **Commands** area of the Command Return Code window to display the Command Information window (Figure 153 on page 214).

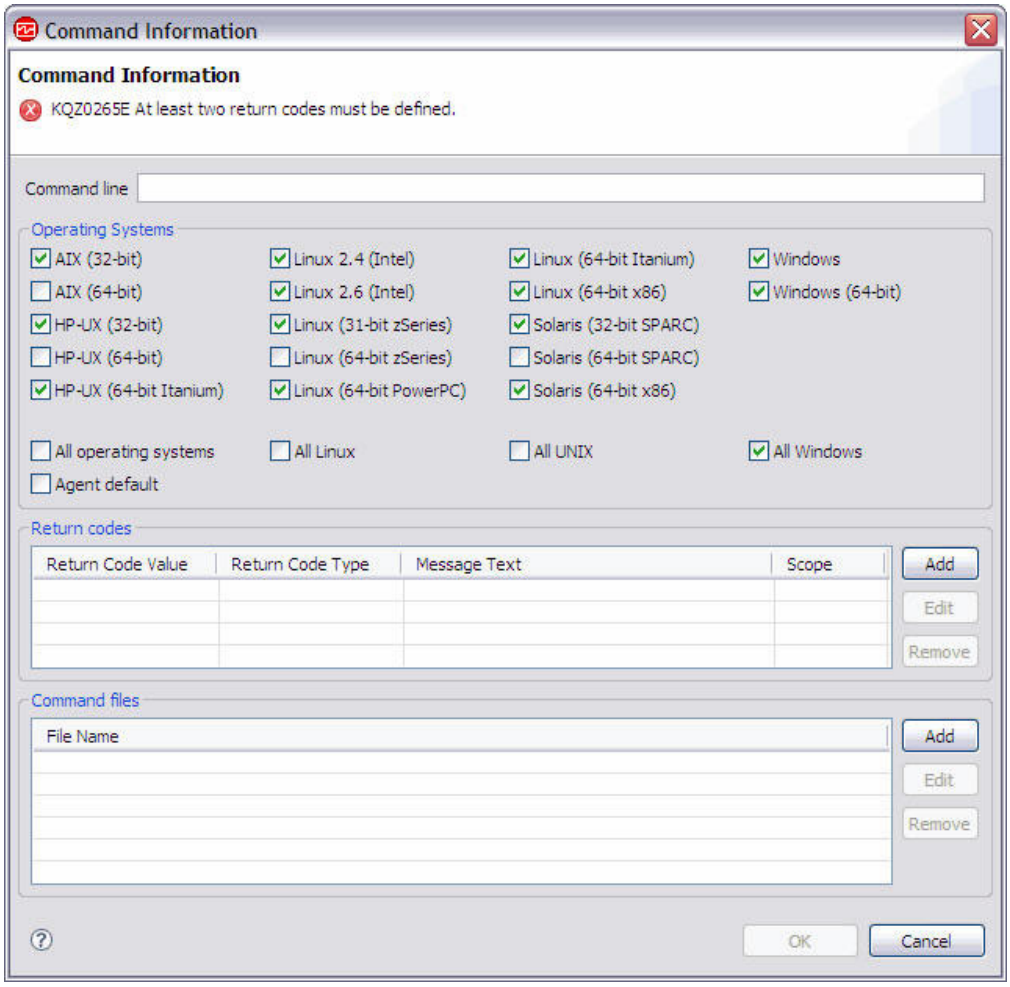


Figure 153. Command Information window

b. Type a command line and select an operating system from the list in the Operating Systems area of the Command Information window.

   Notes:

   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) "Quote" 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
      ```

   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 displayed in the Return Code Definition window (Figure 154 on page 215).

   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

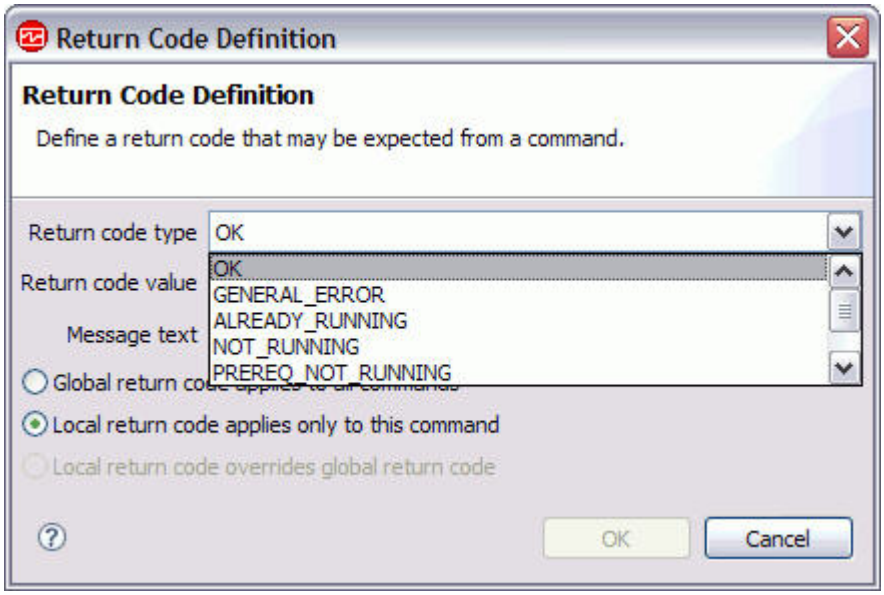


Figure 154. Return Code Definition window: Return code type

e. The return code value is an integer that specifies a defined return code for the command return code. Type a numeric value for the return code type that you selected (Figure 155). 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.

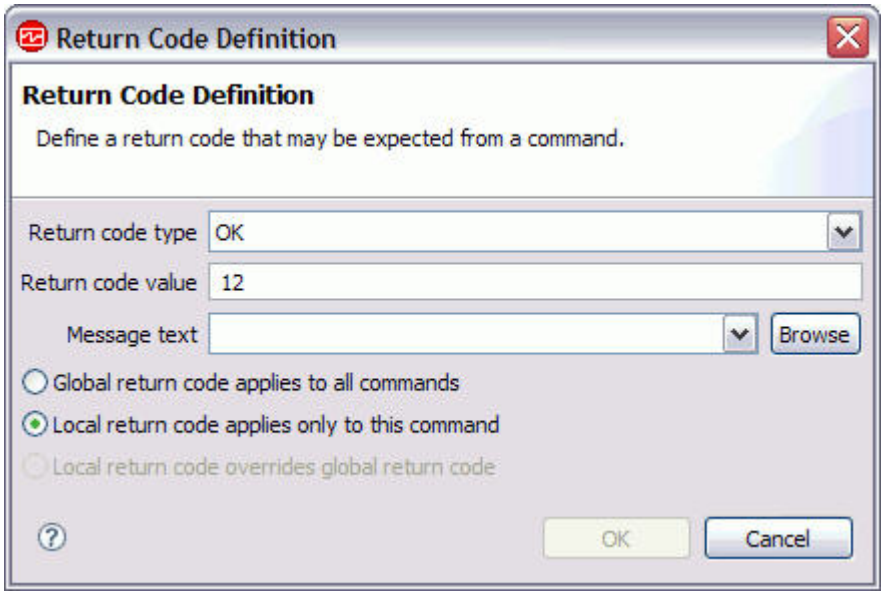


Figure 155. Return Code Definition window: Return code value

f. You must define a message for each return code so that the message can be displayed when the code is displayed.

Click **Browse** to set up the message text. The message window lists messages that are defined in the agent. The Messages (list) window is displayed (Figure 156).

Notes:

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 5j.
2) Until you have defined messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you have defined.

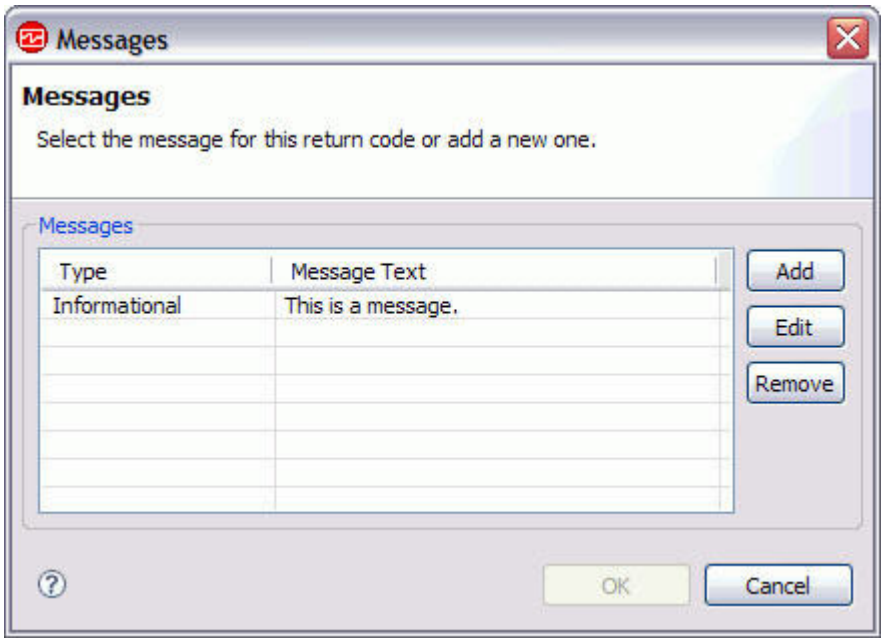


Figure 156. Messages window

g. In the Messages (list) window, click **Add** to see a Message Definition window (Figure 157 on page 217), where you can type the text that describes the meaning of the new message.

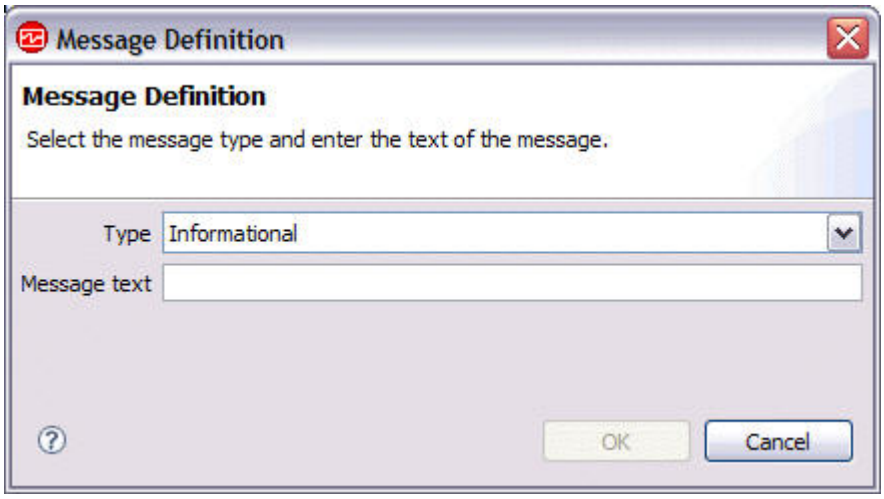


Figure 157. Message Definition window

Note: The message identifier is automatically generated for you.

h. Click **OK**.

i. The Messages (list) window is displayed with the new message (Figure 161 on page 220). To verify the message and make it permanent, select it in the list and click **OK**.

j. The new return code type, value, and text are displayed in the Return Code Definition window (Figure 158). 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.

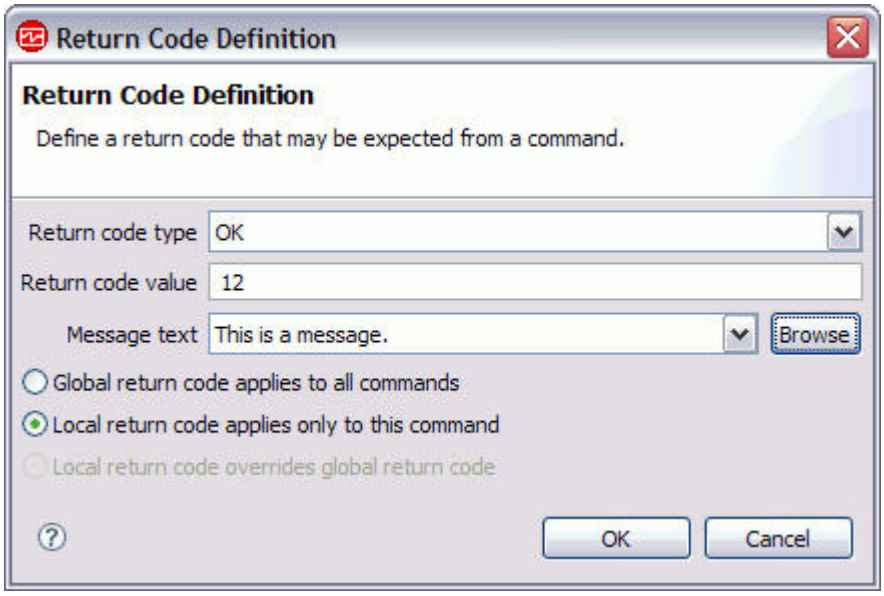


Figure 158. Return Code Definition window completed

k. Click **OK** in the Return Code Definition window.

l. Define at least two return codes (one to indicate no problems with the availability, another to indicate if a problem occurred) before leaving the Command Information window. Return to Step 5c if you need to add another return code.

m. (*Optional*) In the Command Information window (Figure 153 on page 214), **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 153 on page 214. These files are also packaged and distributed with the agent. If you want to edit the definition of a command file you have already added, or have made a change to the original command file since it was copied into the project, select the file and click **Edit**. See ("Editing a command file definition" on page 222).

n. Click **OK** in the Command Information window.

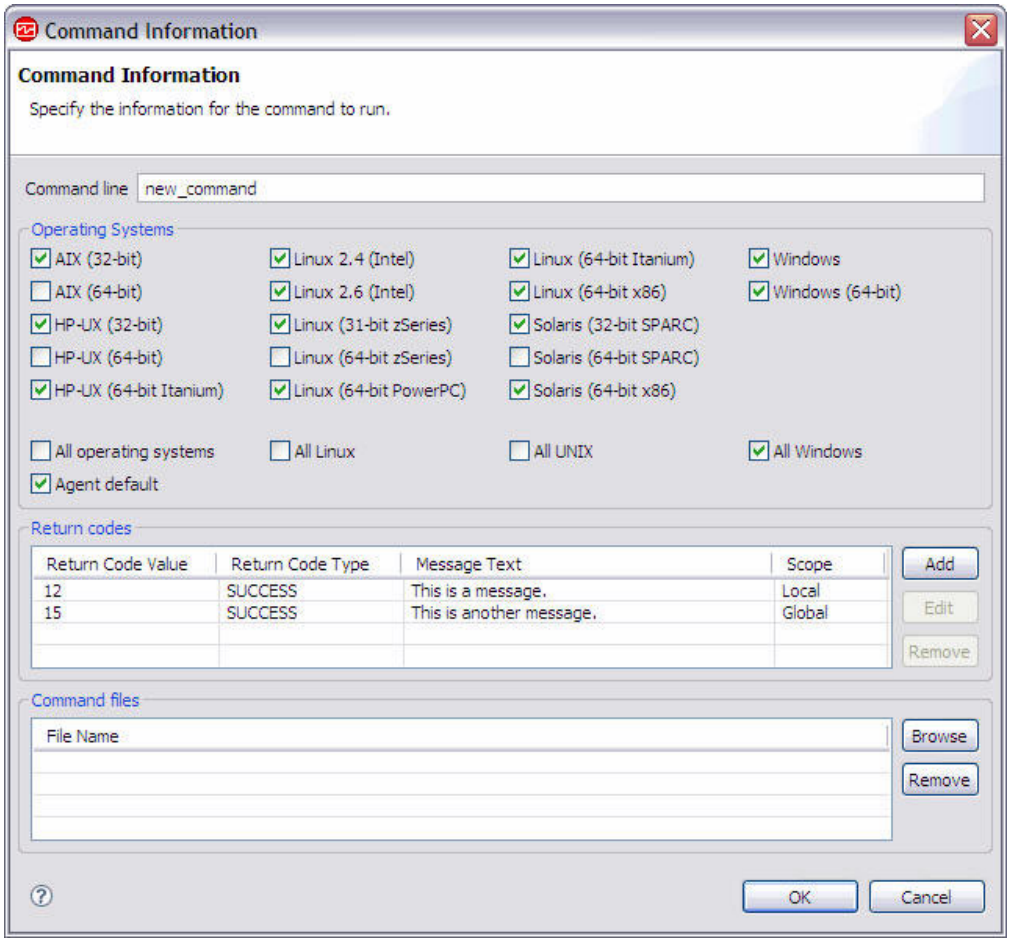


Figure 159. Command Information window completed

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 have not already been defined, define and describe global return codes that you want your command return code to use as follows:

a. Click **Add** in the Global return codes area of the Command Return Code page (Figure 152 on page 213).

   Note: The return codes defined here are global, which means they are appropriate for all of the commands defined for the command return code. (They are not shared between command return codes). In addition, you can define return codes while entering command information (Figure 153 on page 214). The return codes 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 example, if a return code is 0, this means that everything is functioning correctly. You can define it at the global level once, and then all defined commands interpret 0 in this way.) In addition, if the command on a Windows system returns a 5, but none of the other operating systems gives a return code of 5, you can define the return code of 5 only for the Windows command. In addition, if you define the same return code at the local command level that was previously defined at the global level, the one defined at the command level is used. You can use this 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 displayed in the Return Code Definition window (Figure 154 on page 215).

   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. The return code value is an integer that specifies a defined return code for the command return code. Type a numeric value for the return code type that you selected (Figure 160 on page 220).

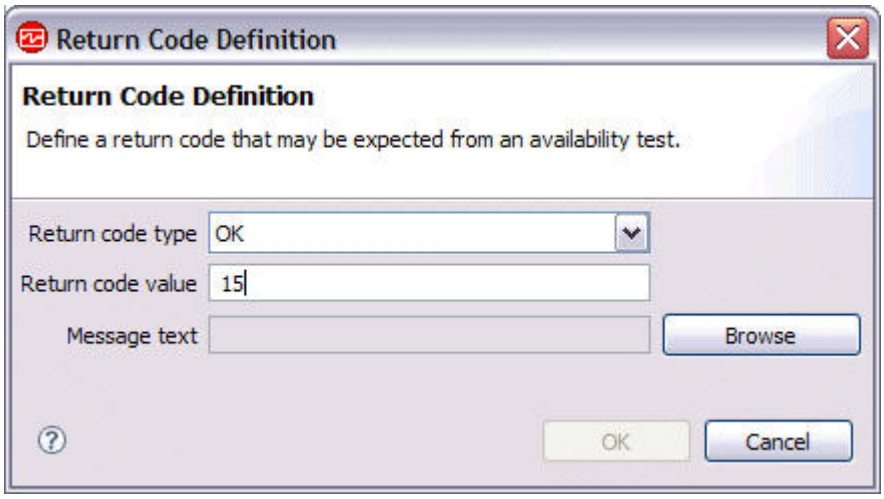


Figure 160. Return Code Definition window: Message text

d. You must define a message for each return code so that the message can be displayed when the code is displayed.

   Click **Browse** to set up the message text and its associated meaning. The Messages window lists messages that are defined in the agent. The Messages (list) window is displayed (Figure 161).

   Notes:

   1) Until you have defined messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages you have defined.
   2) You can select text that was entered previously by selecting it in the **Message text** drop-down list instead of clicking **Browse**. Then continue to Step 6g on page 221.

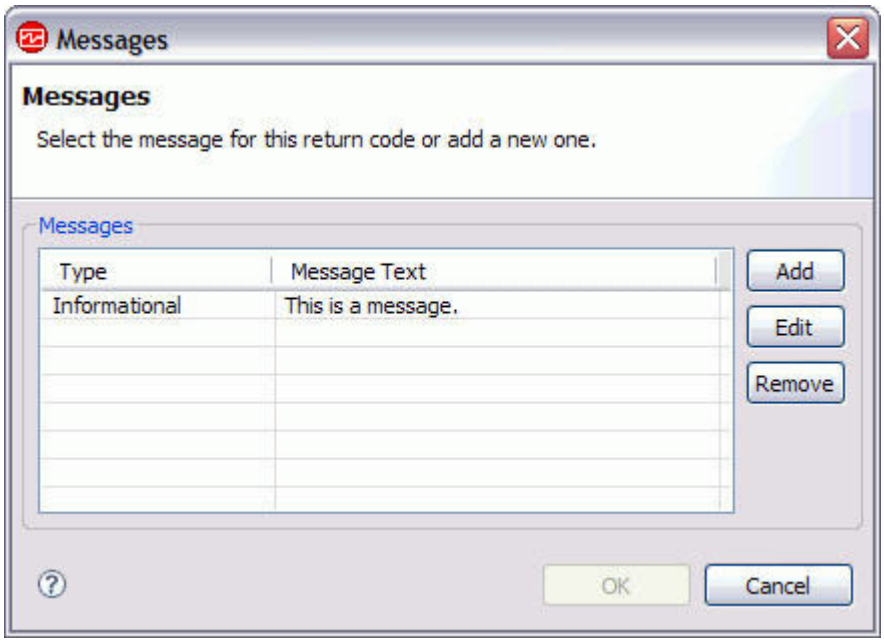


Figure 161. Messages window

e. In the Messages (list) window, click **Add** to see a Message Definition window (Figure 157 on page 217), where you can type text that describes the meaning of the new message. Click **OK**.

f. The Messages (list) window is displayed with the new message (Figure 156 on page 216). To verify the message and make it permanent, select it in the list and click **OK**.

g. When the new text, type, and value are displayed in the Return Code Definition window, (Figure 162), click **OK**.

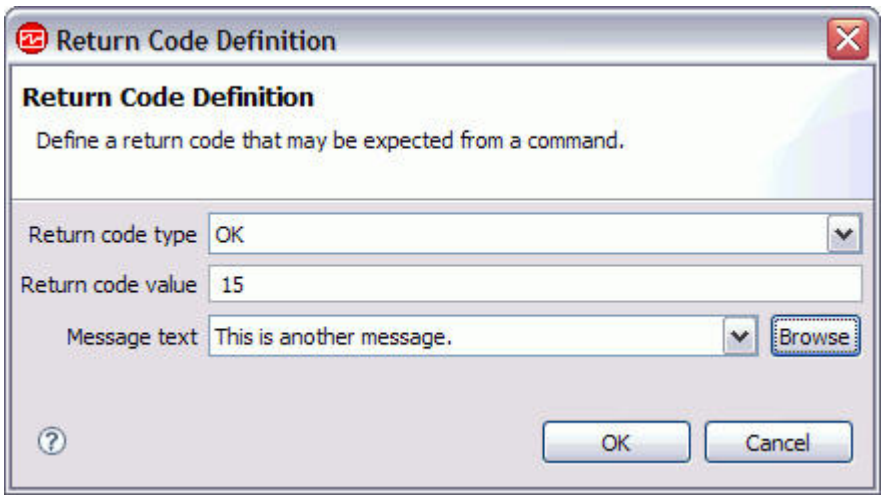


Figure 162. Return Code Definition window

h. On the Command Return code page, when you have completed 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.

  —OR—

- If you are using the New Agent Component Wizard, click **Finish** to return to the Agent Editor.

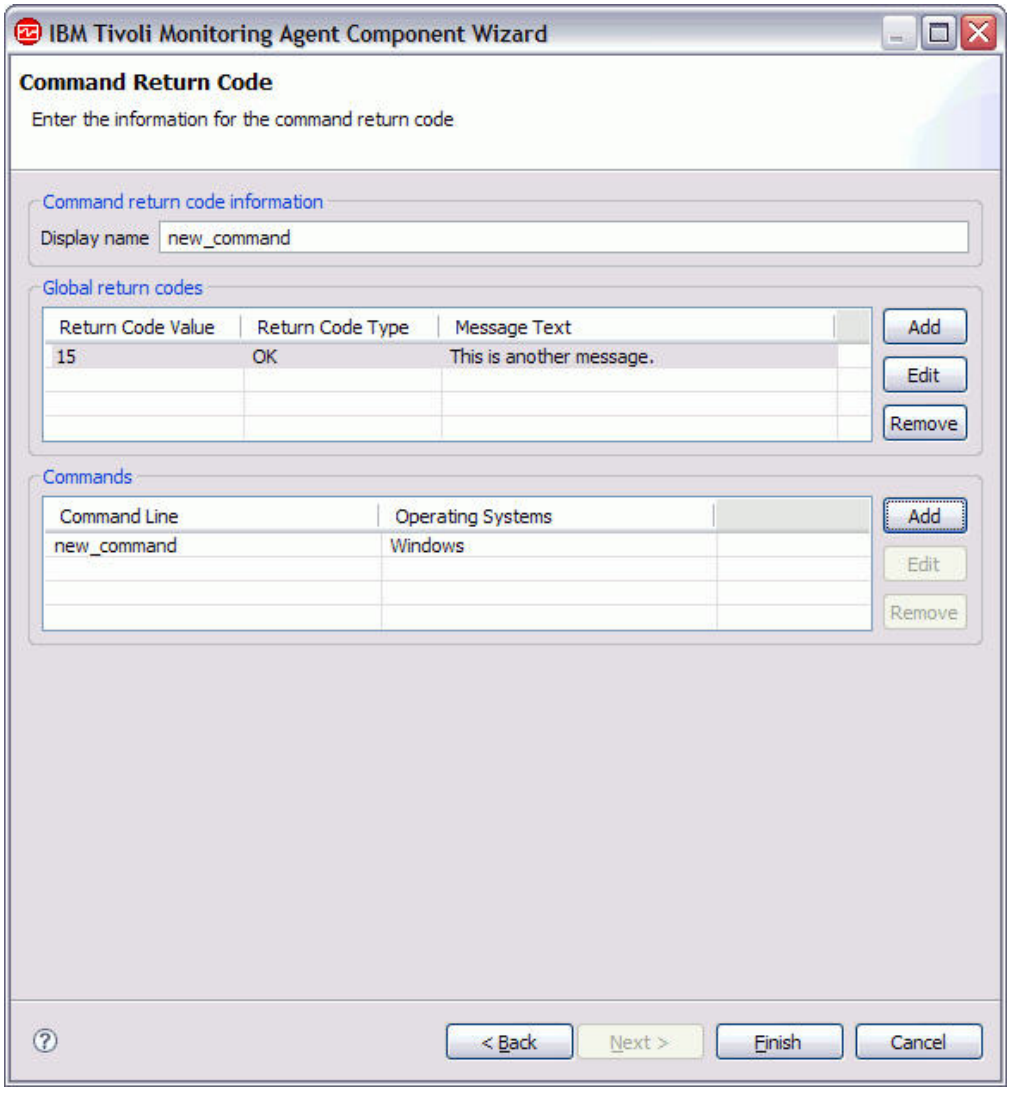


Figure 163. Command Return Code page completed

Editing a command file definition

You can change the command file imported into the project, or import changes to the existing command file into the project. Select the file in the Command files area of the Command Information window, and click **Edit** to open the Import Command File window Figure 164 on page 223.

From the Import Command File window, you can get the status of the command file, change the location of the original source file, and recopy the source file into the agent. Click **OK** to schedule a copy of the file to occur the next time the agent is saved, or click **Copy Immediately** to copy the file without first saving the agent.

Note: The **Copy Immediately** button is not available when accessing the Import Command File window from the New Agent Wizard.

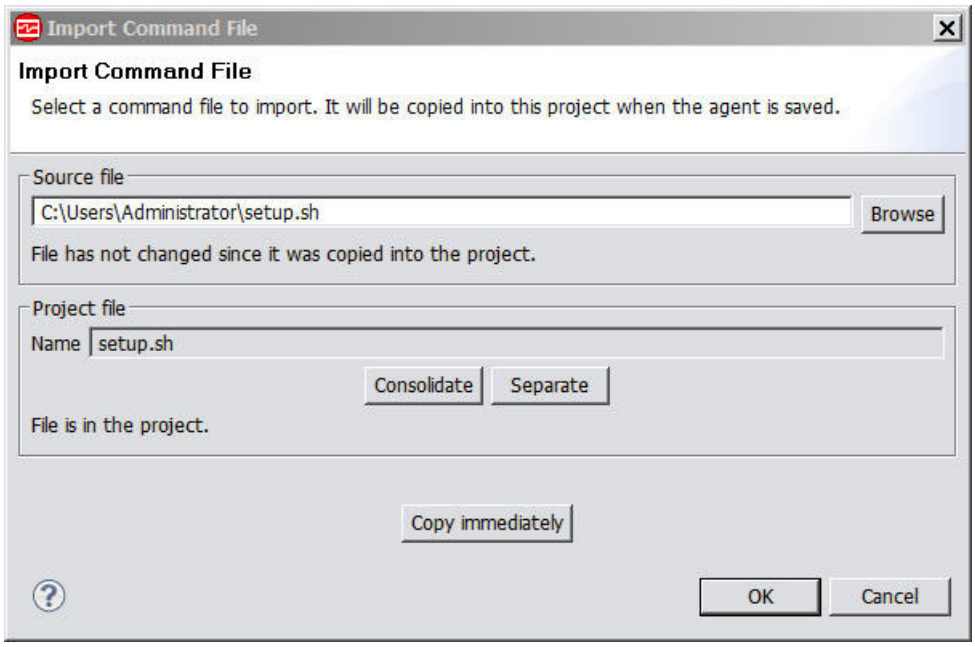


Figure 164. Import Command File window

File Separation & Consolidation

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 (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 press the Separate button to make separate copies of Java resource files for individual operating systems.

If you separated the files into operating-system-folders, or you created the agent in a version of the Agent Builder that did not support common folders, you can use the **Consolidate** button 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 others are discarded.

Chapter 20. Monitoring 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 from either a local or remote system. The output of the 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 separator(s) you define. For more information about separators, see "Script parsing and separators"

The command may use environment and configuration variables within the user-created script, executable file, query, or system command. The command may 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.

Collecting data from a remote system

To collect data from a remote system, Agent Builder creates a Secure Shell (SSH) session and invokes the script or external program on the remote system. The agent establishes and logs into an SSH session, uploads the scripts to the remote system, invokes 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 only supports use of 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 performed outside of the agent and Agent Builder.

To run a Take Action command written against a Secure Shell (SSH) enabled script data provider on the remote system, see "SSHEXEC action" on page 640.

Script parsing and separators

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.

Examples

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, for 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, so in this case when defining the attributes, change the default separator to use the "-" character.

Complex script output

Some scripts may output data rows that have irregular or changing separators, for example:

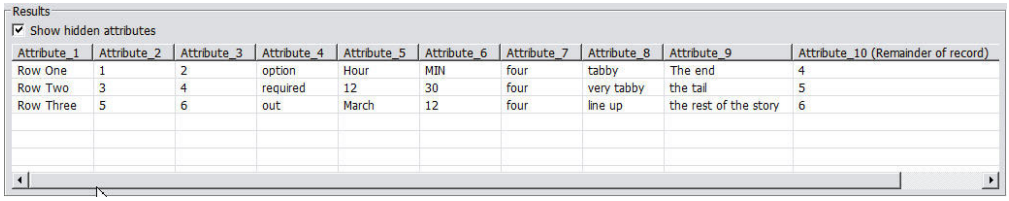
```
Row One;1;2;[option]Hour:MIN;fourtabby     The end;4
Row Two;3;4;[required]12:30;fourvery tabby the tail;5
Row Three;5;6;[out]March:12;fourline up    the rest of the story;6
```

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 defining 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 you want to extract the values between the "]" and ":" characters. In this case when defining 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

The effect of defining these separators on a script that outputs the data rows shown earlier in this example is demonstrated by the following Agent test 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 165. Example attribute value output when Agent parses complex script output.

The procedure to define the attribute separators is described under step 10 on page 233 of "Steps for monitoring output from a script."

Steps for monitoring output from a script

Use the following procedure to monitor output from a script:

1. On the Agent Initial Data Source page (Figure 166 on page 228) or the Data Source Location page, click **Command or script** in the **Monitoring Data Categories** area.

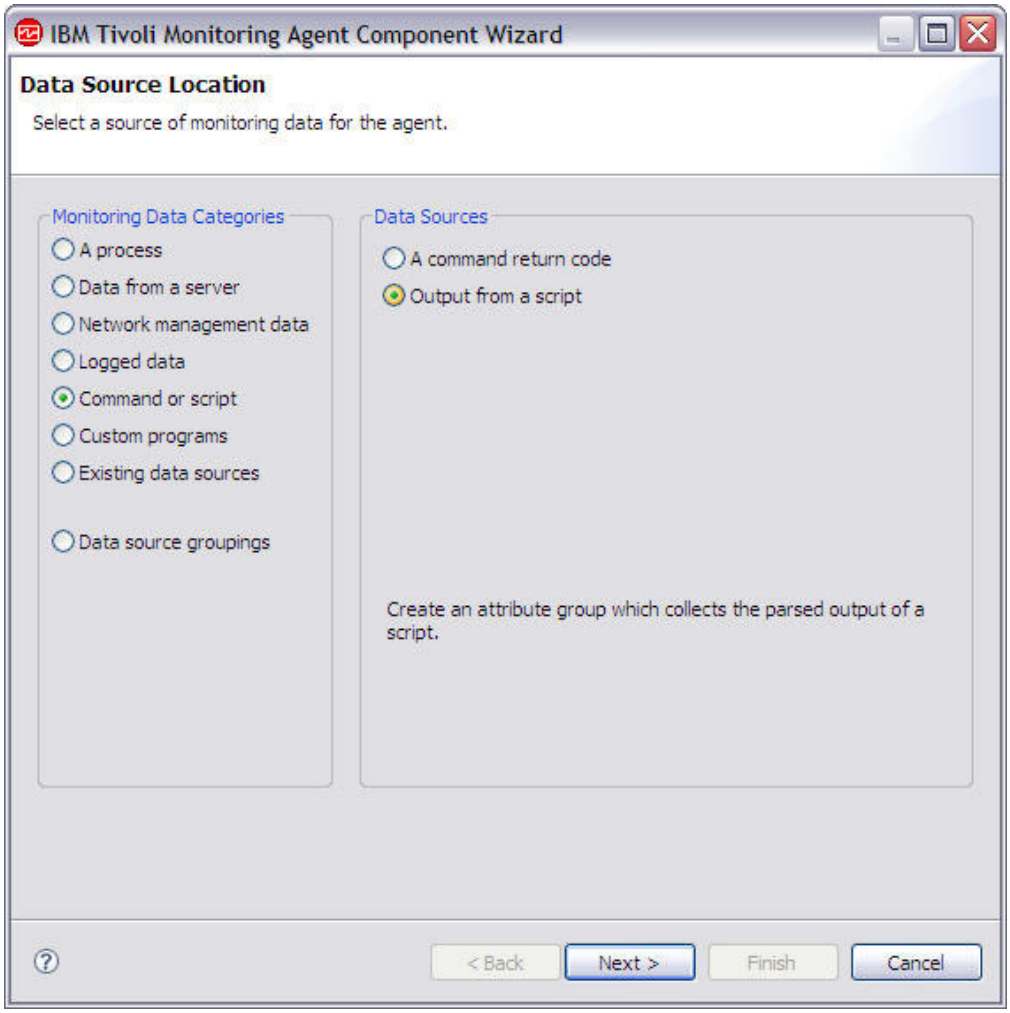


Figure 166. Adding data from a script

2. In the **Data Sources** area, click **Output from a script**.
3. Click **Next**.
4. On the Command List page (Figure 167 on page 229), click **Add** to display a Command Information window.

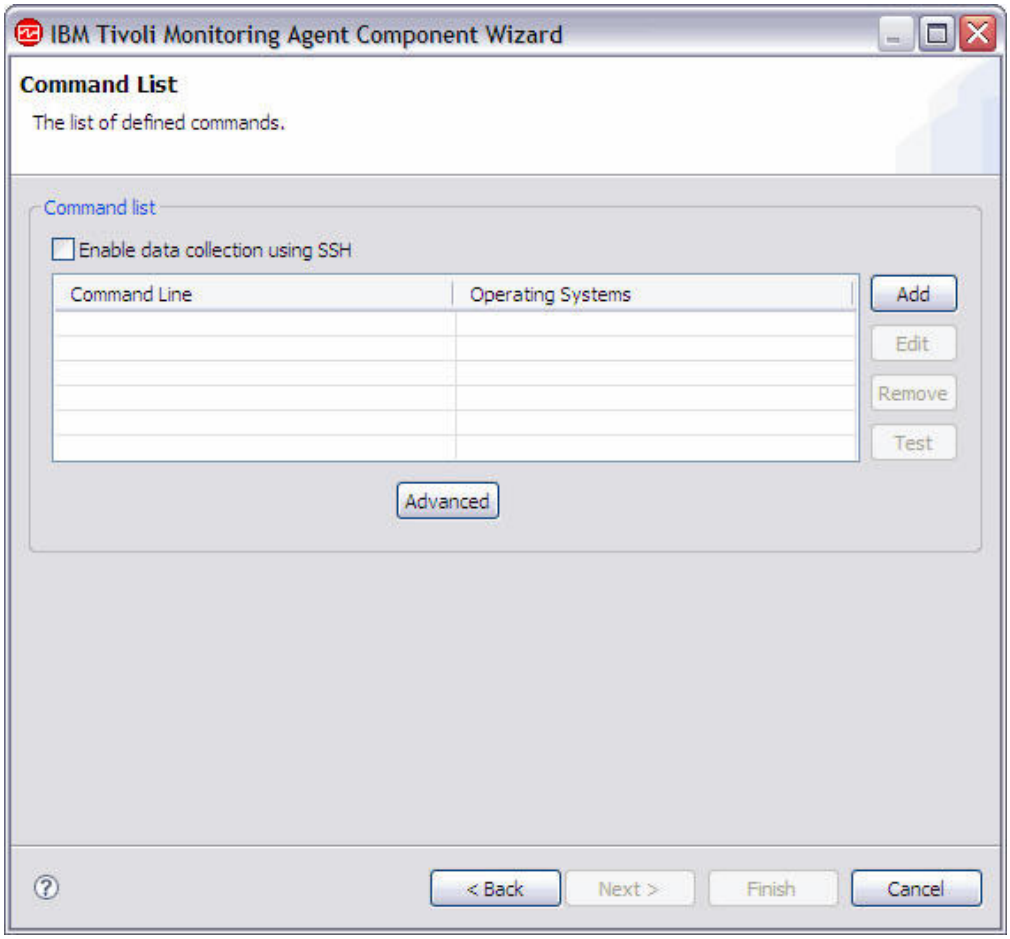


Figure 167. Command List page

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: A **Test** button is enabled if a command exists that can be run on the operating system on which the Agent Builder is currently running. You can use this button to test a command that you defined.

5. In the **Command Information** area in the Command Information window (Figure 168 on page 231), type a command name with the necessary arguments in the **Command** field, and a separator in the **Separator** field.

   Notes:

   a. Scripts in Windows are frequently invoked 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 have shell environments installed by default. To verify if a shell environment exists, SSH or log in 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 invokes the script. The following variables are exceptions to this:

   AGENT_BIN_DIR
   The directory where the agent will place binaries or scripts

   AGENT_ETC_DIR
   The directory where the agent will place configuration files

   AGENT_LIB_DIR
   The directory where the agent will place shared libraries or dlls

   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 executed 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, which 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`.

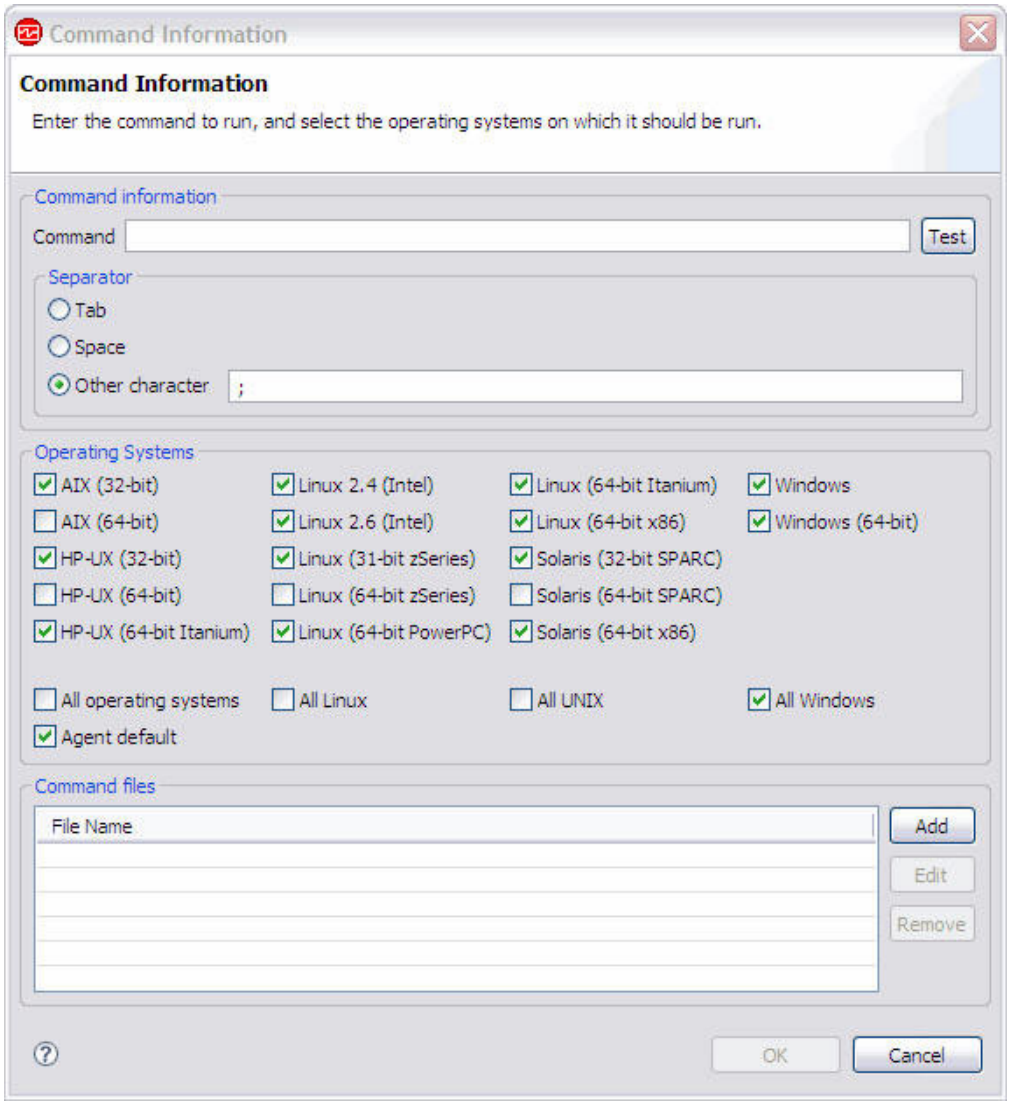


Figure 168. Command Information window for a script

6. In the **Operating Systems** area, select one or more operating systems.

   Note: When you collect data from a remote system using SSH, Operating Systems is a property of the system to which the agent is installed, not the remote system. It is recommended that you select the **All operating systems** check box when using 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 file or files from your system. 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 153 on page 214. These files are also packaged and distributed with the agent. If you want to edit the definition of a command file you have already added, or have changed the contents of the command file, select the file and click **Edit**. See ("Editing a command file definition" on page 222).
8. Click **OK**. The Command List page is displayed.
9. To test the command, perform the following steps:

a. Click **Test** to open the command information and display the Test Command window (Figure 169). To test the script on a remote system, select a system from the drop-down list (if one has been defined) or click **Add** to add the host name of a system.

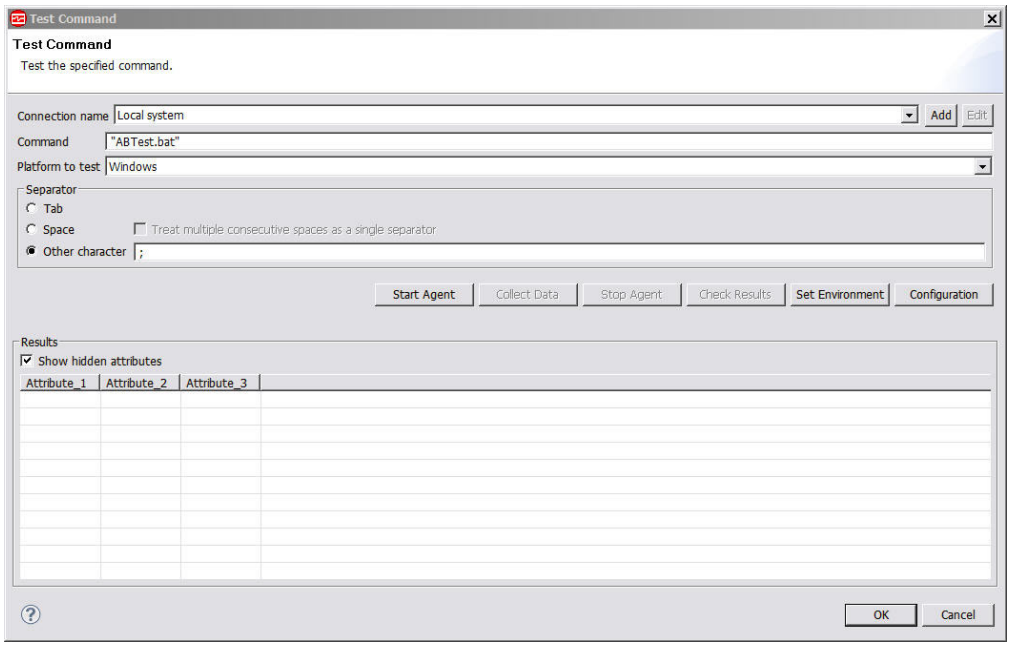


Figure 169. Test Command window

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 filled in.

   Note: You can specify other separators using the Attribute Information window Figure 170 on page 234 at attribute creation time or by using the Agent Editor to modify an existing attribute. For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37 and for more information about manipulating data source and attributes, see Chapter 7, "Editing data source and attribute properties," on page 53

2) *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see "Configuration prior to testing" on page 374.
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 had 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 made changes to the command or the separator, the appropriate command is updated to reflect those changes.

   If this window was launched when creating the script data source, the attributes were added to the new script data source.

   If this window was launched from an existing script data source, then any changes to the attributes are made to the script data source and any additional attributes are added, but any extra attributes are not removed. These options affect only the attributes parsed from the script output. Any derived attributes are not affected. You might need to update or remove derived attributes manually if any of these attributes become invalid based on the attributes they reference. The derived attribute formula is displayed and not the actual result value.

Note: If the attribute group already 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 9a on page 232

10. If you skipped testing the command in Step 9, perform the following steps:
    a. On the Command List page with the completed command information, click **Next**. See (Figure 167 on page 229).
    b. On the Attribute Information page Figure 170 on page 234, complete the attribute name and type information using Table 6 on page 56. Select **Add additional attributes** to add further attributes
    c. On the Attribute Information page Figure 170 on page 234, 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, a space, or 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.

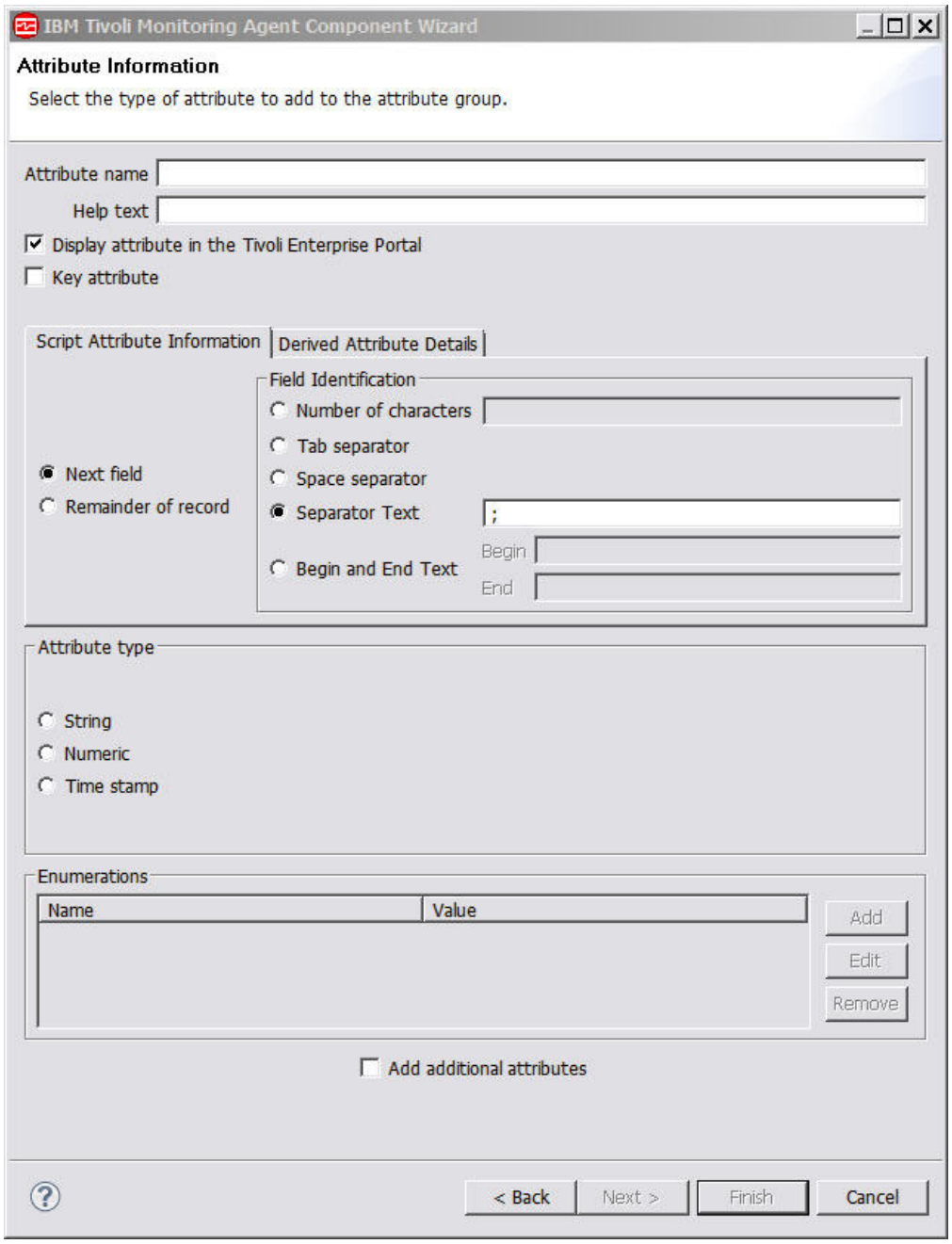


Figure 170. Attribute Information page

11. 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.

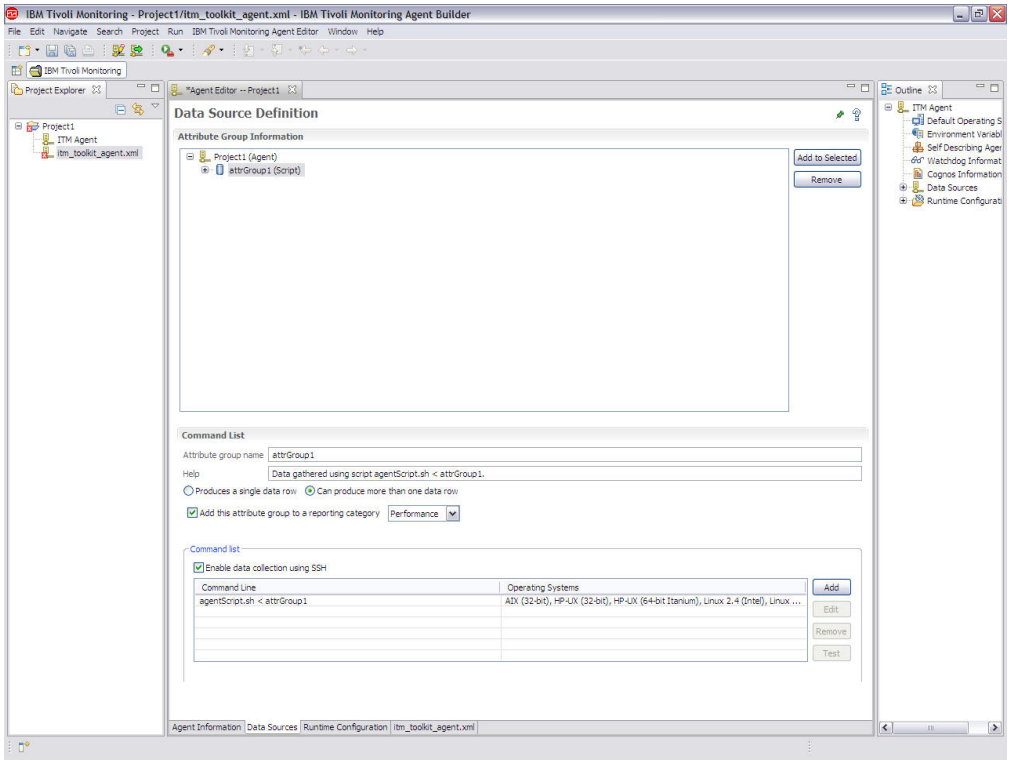


Figure 171. Agent Editor Data Source Definition page

12. You can add attributes and supply the information for them. See “Creating attributes” on page 54 for more information. In addition to the fields that are applicable to all the data sources (Table 6 on page 56), the Data Source Definition page for the Script data source Figure 171 has the following options:

    Command List
    Provides access to the commands and scripts to invoke during data collection.

    Add Allows the user to add a new command to be invoked 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 371).

Chapter 21. Monitoring data from Java Database Connectivity (JDBC)

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 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 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 Web site at www.microsoft.com
- Oracle Database: http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

Note: An important thing to remember is that the JDBC data provider can remotely monitor your database servers. A Java runtime environment and the JDBC driver jar files for the database server you are connecting to must already be located on the system where you will run the agent.

The following versions of Java are supported:

- Sun Microsystems Java Version 5 or later
- IBM Corporation Java Version 5 or later

Procedure

Use the following procedure to add a JDBC data source:

1. On the Agent Initial Data Source page (Figure 172 on page 238) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

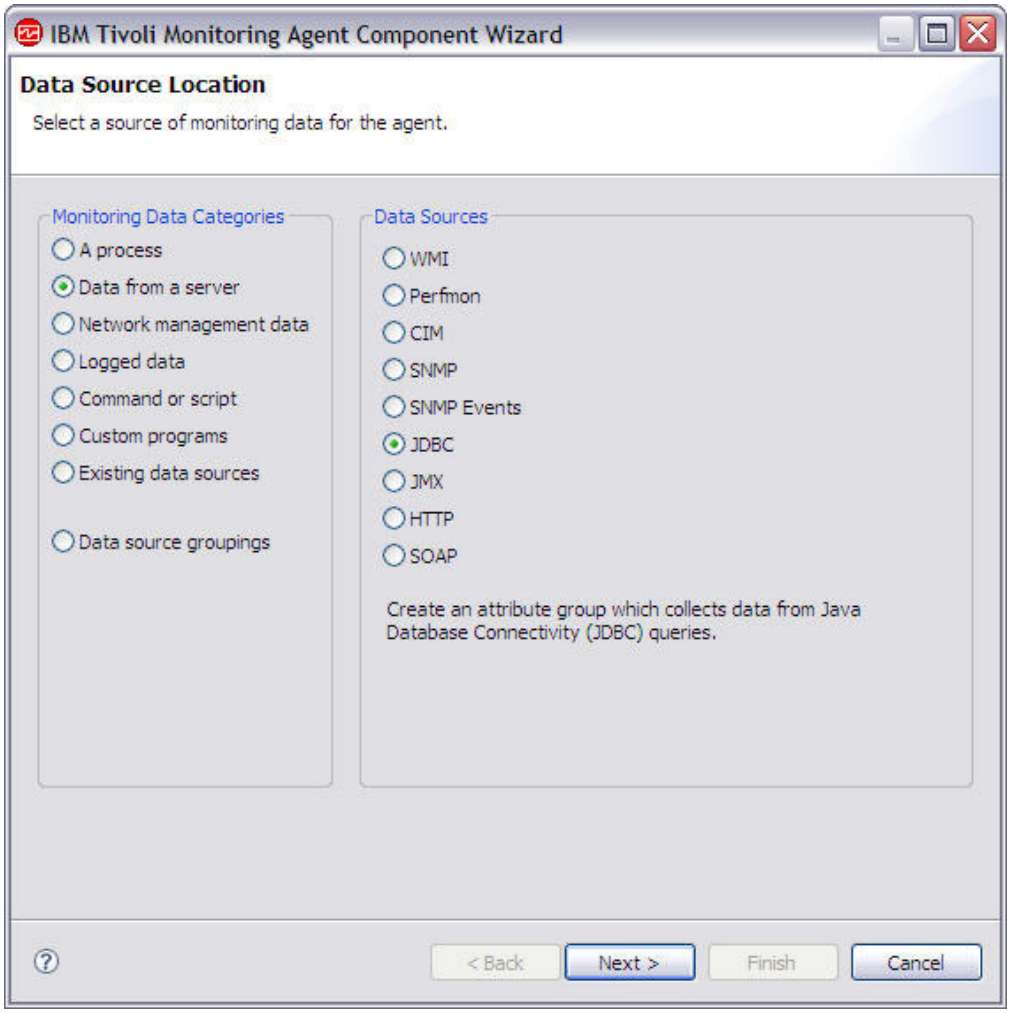


Figure 172. Adding JDBC data

2. In the **Data Sources** area, click **JDBC**.
3. Click **Next**.
4. On the **JDBC Information** area in the JDBC Information page (Figure 173 on page 239), click **Browse** to connect to a database and build your SQL Query.

   With the JDBC Browser you can connect to a database and view its tables so you can easily build an SQL query that collects the data that 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 right away 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.

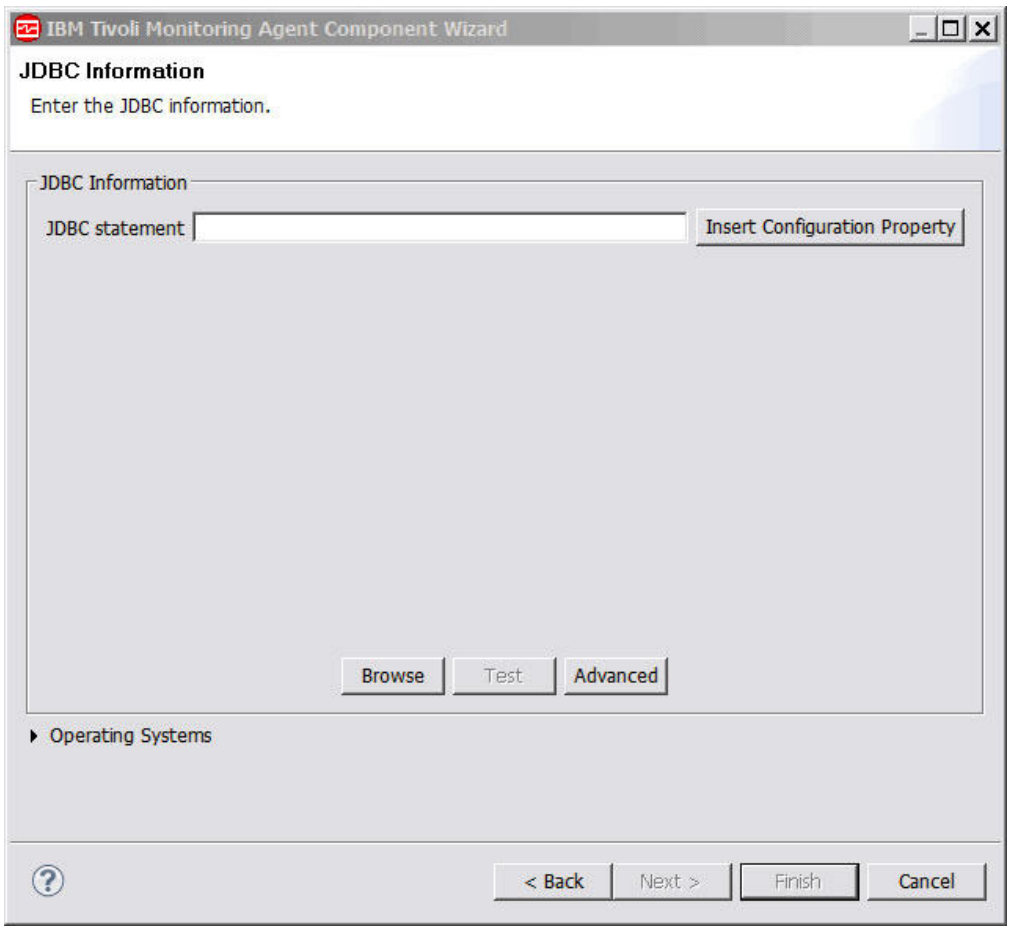


Figure 173. JDBC Information page

5. The first time the Browser is launched, the Java Database Connectivity (JDBC) Browser window (Figure 174 on page 240) indicates that no connections are selected. You must add a connection, so click **Add** and perform the Steps to add a connection. If you have a connection already defined that connection is used and you can proceed to Step 6.

   Note: The **Status** field shows the status of the current connection.

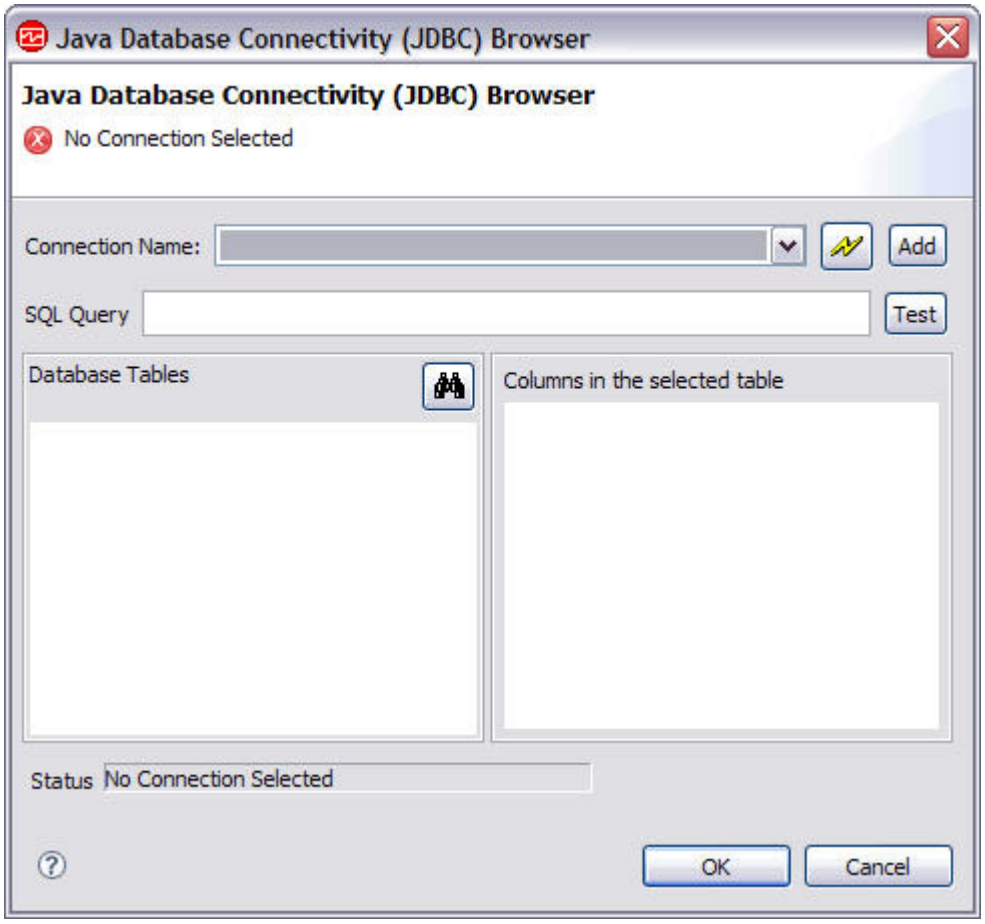


Figure 174. JDBC Browser window

Steps to add a connection:

a. On the JDBC Connections page (Figure 175 on page 241), click **New Connection**, and click **Next**.

Figure 175. Database Connection Wizard: JDBC Connections page

b. On the Connection Properties page (Figure 176 on page 242), complete the fields as follows:

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 need to be the database administrator

Password
Must be defined with at least read access to the database, but does not need to be the database administrator

Hostname
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 driver jar files needed to connect to the database. Type the path name, or click **Browse** to locate the directory.

(*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.

After completing the fields, click **Test Connection** to create a connection to the database using the configuration parameters you have specified. A message on the Connection Properties page indicates whether the connection succeeds. When you have a working connection, click **Finish**.

If you are building the agent on a system that is similar to your monitored systems, it is good to check this box. If you do not, the user configuring the agent is faced with an empty field and must determine the values for all of the information without default values.

Figure 176. Database Connections Wizard: Connection Properties page

6. In the Java Database Connectivity (JDBC) Browser window (Figure 177 on page 244), a connection is made to the configured database. The tables contained in the database are shown in the **Database Tables** area. Select a database table to see the columns contained in that table in the **Columns in the selected table** area.

   Notes:

   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 shown by selecting a different filter option. The available filter options are shown in Table 12.

Table 12. Filter options

Filter option	Description
All	Show all tables
User	Show only user tables
System	Show only system tables
View	Show ony database views

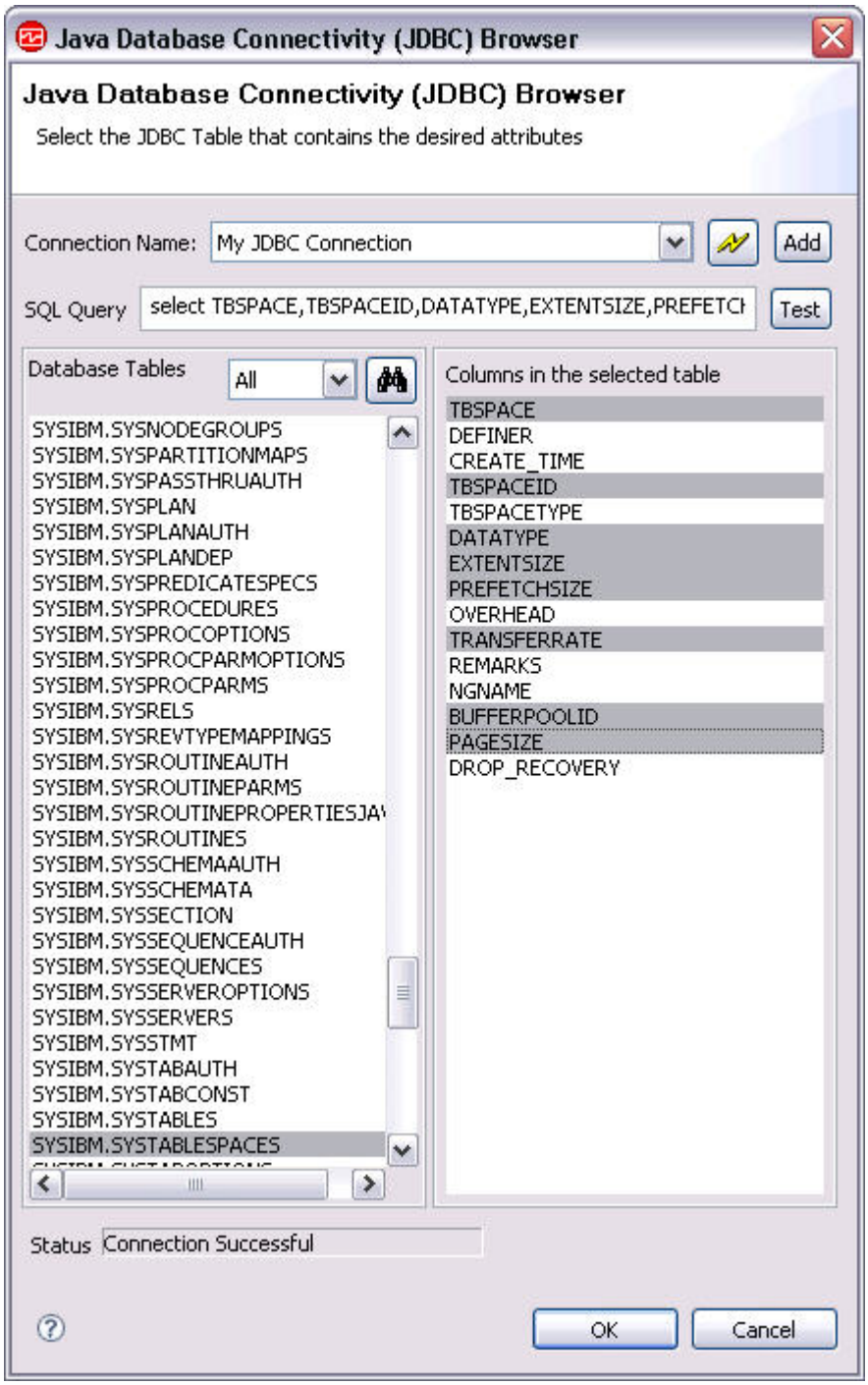


Figure 177. Java Database Connectivity (JDBC) Browser window

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)* You can test this attribute group by clicking **Test** on the JDBC Information window, Figure 178. For more information about testing, see "Testing" on page 252
8. *(Optional)* You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66
9. On the JDBC Information page (Figure 178), **Operating Systems** section, select the operating systems, and click **Next**.

   See "Specifying operating systems" on page 86 for information about which operating systems to select.

   Note: Click **Insert Configuration Property** to select a property to insert. For more information see "Customizing configuration" on page 359.

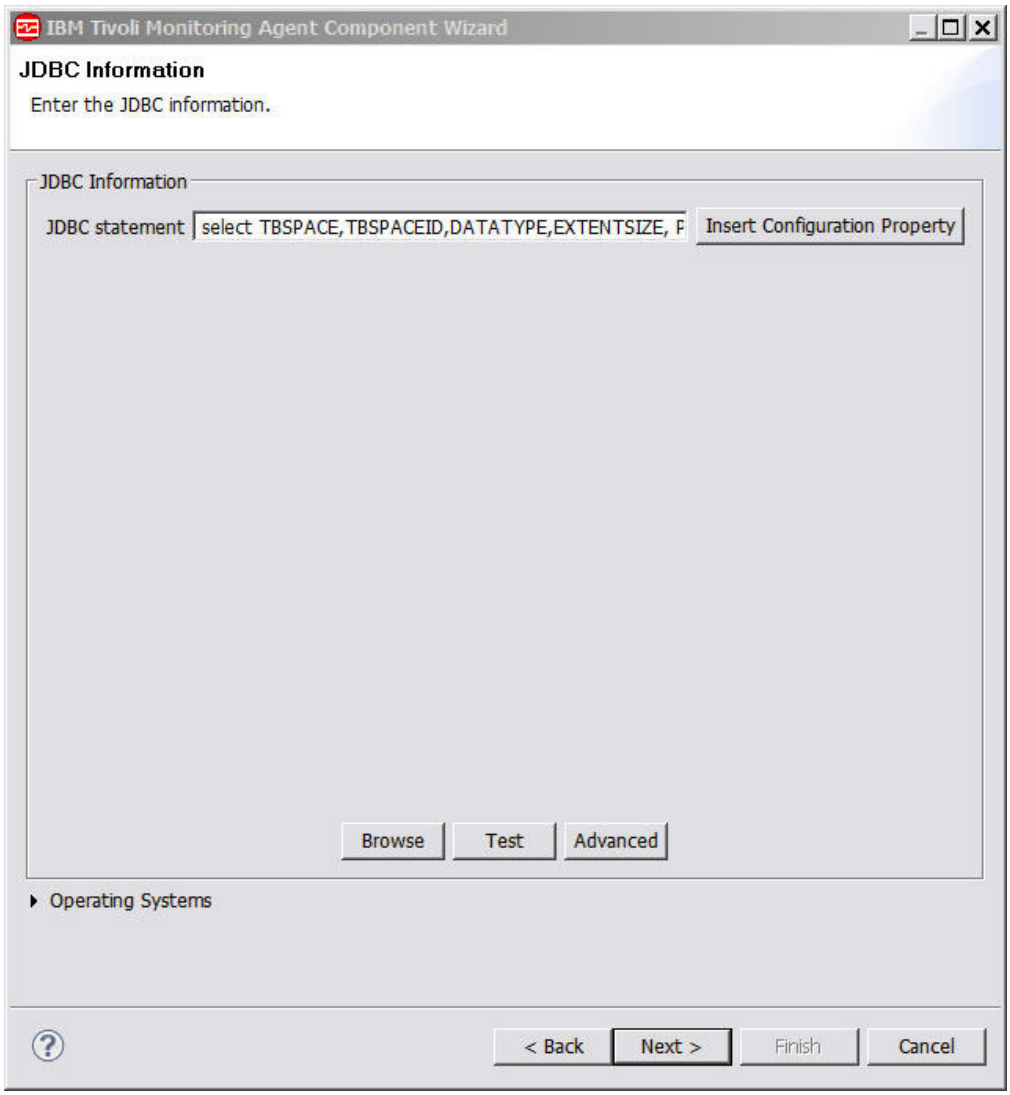


Figure 178. JDBC Information page

10. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. See "Selecting key attributes" on page 27 for more information.

11. If you want to test a data source that you previously defined, in the Agent Editor window (Figure 129), 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" on page 252.

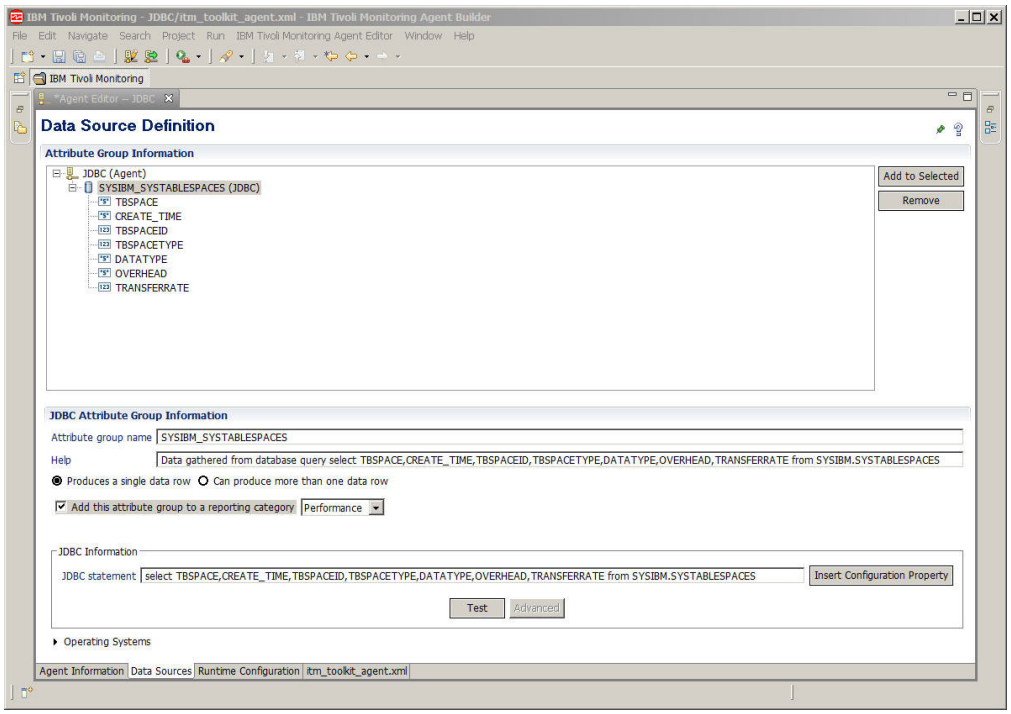


Figure 179. Data Source Definition page Attribute Group Information

12. If you want to view the configuration sections that were automatically generated, click the **Insert Configuration Property** tab of the Agent Editor (Figure 180 on page 247). You can change the labels or default values for these configuration properties to match the defaults you want a user to see when they initially configure the agent.

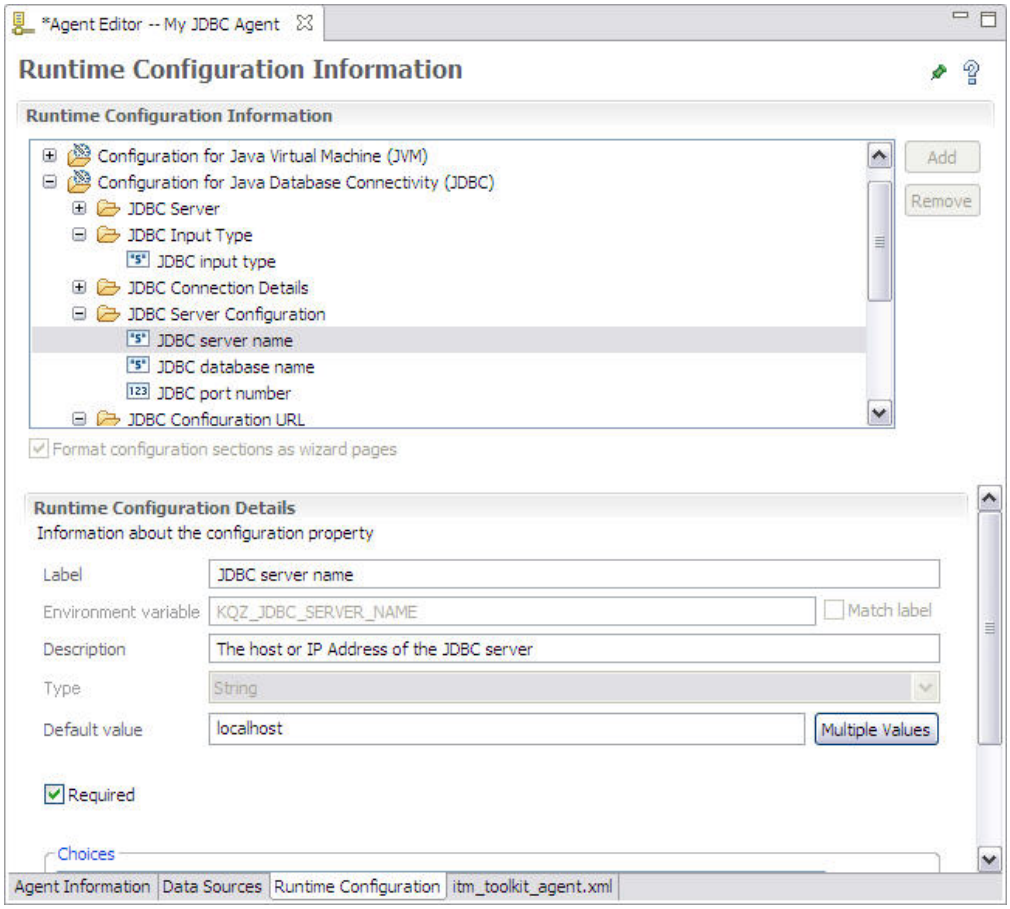


Figure 180. Runtime Configuration Information page, Configuration for Java Virtual Machine (JVM)

13. (*Optional*) If you chose to manually create the JDBC Data Source without clicking Browse in Step 4, complete the information on the Attribute Information page using Table 6 on page 56.

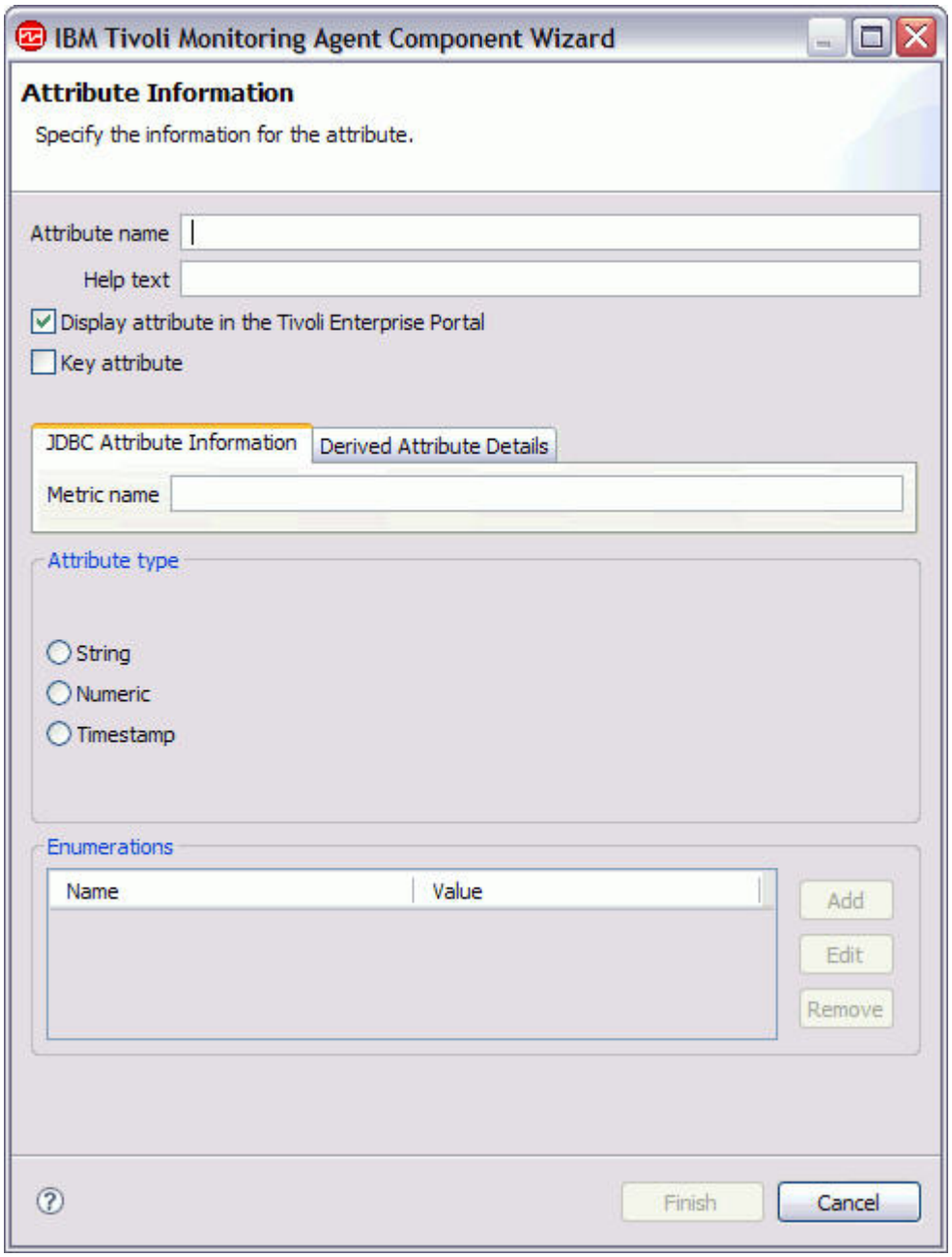


Figure 181. Attribute Information page

The Agent Builder JDBC Data Source supports collecting data from most SQL types. The information in Table 13 describes the type of attribute that is created by the JDBC Browser when it detects a column of one of these types. These are the supported data types for use with a monitoring agent.

Table 13. Supported SQL data types for use with a monitoring agent

SQL data type	IBM Tivoli Monitoring attribute created
BIGINT	This data type is a 64-bit gauge value in IBM Tivoli Monitoring. It will be a 32-bit gauge if you select IBM Tivoli Monitoring V6.2 compatibility.

Table 13. Supported SQL data types for use with a monitoring agent (continued)

SQL data type	IBM Tivoli Monitoring attribute created
DECIMAL DOUBLE FLOAT NUMERIC REAL	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. The attribute will be a 32-bit gauge if you select IBM Tivoli Monitoring V6.2 compatibility.
BIT INTEGER SMALLINT TINYINT	The following SQL types are created as 32-bit gauge attributes in IBM Tivoli Monitoring.
BOOLEAN	This is a 32-bit gauge value 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.
TIME DATE CHAR LONGVARCHAR VARCHAR	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

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 (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):

- 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 414.

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 named in the **Class Path** field, or directories 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 need to be explicitly identified; they will be found because they are located 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 on and the port number to which to connect. The URL format typically looks like this:

```
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 using subnodes. To do this 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, "Creating subnodes," on page 331 for more details about accessing the Subnode Configuration Overrides page. When configuring the agent at runtime, all of these properties must be configured by the user for each new subnode instance created.

In addition to the subnode overrides, your agent must also point to JDBC drivers for each database type to which you are going to connect in your

subnodes. The Jar directories parameter is the most convenient way to point to your JDBC drivers. List the directories containing the JDBC drivers with a semicolon separating each directory entry. For example, if you are connecting to DB2 and Oracle databases with the agent, you must specify a Jar directories value similar to the following: `C:\Program Files\IBM\SQLLIB\java;C:\oracle\jdbc`.

Stored procedures

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 only want to collect 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 quotes. If the argument can be parsed as an integer it is passed to the stored procedure as an integer argument. Any argument enclosed in double quotes is passed as a string argument.

SQL Server Samples

call sp_helpdb

Runs the procedure called sp_helpdb which requires no arguments. Data from all returned result sets are included in the data returned by the data provider.

call:2 sp_helpdb master

Runs the procedure called sp_helpdb with the master argument. This argument is a string input argument. Only the data from the second result set returned by the stored procedure is included in the data 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 expected additional result sets are expected to be compatible with the first one.

DB2 stored procedure

Here is a sample DB2 function written in SQL that 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 using the same syntax 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 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 using the same syntax 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

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the JDBC Information page (Figure 178 on page 245).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see “Tivoli Monitoring Agent Editor” on page 37

After you click **Test** in one of the previous two steps, the Test JDBC Statement window Figure 182 is displayed

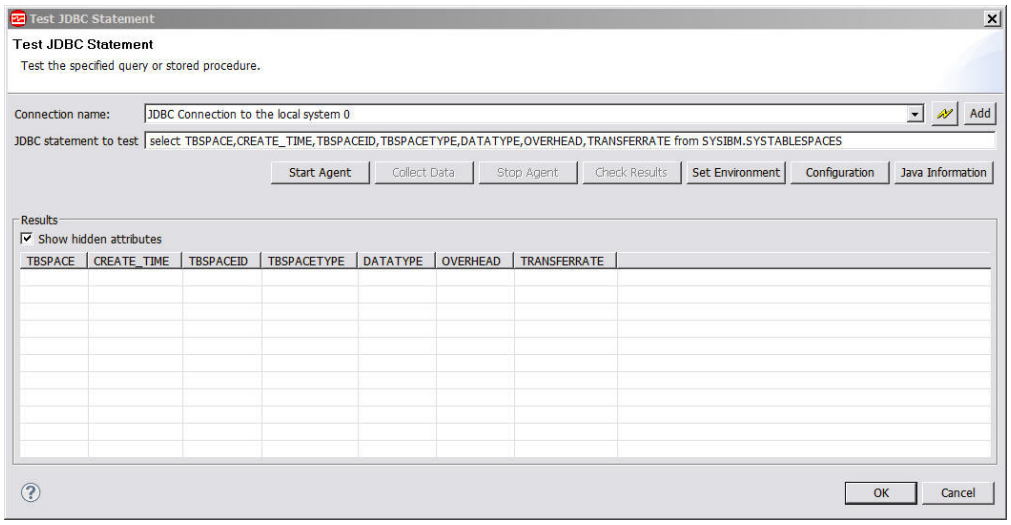


Figure 182. Test JDBC statement window

To start and test the data source use the following procedure

1. *(Optional)* Before starting your test you can set environment variables, configuration properties and Java information. For more information see "Configuration prior to testing" on page 374. For more about JDBC configuration properties, see "JDBC configuration" on page 249.
2. Click **Start Agent**. A window indicates that the Agent is starting.
3. 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.

   Note: The order of the returned data is significant; for example, the data value in the first returned column will always be assigned to the first attribute. If you reorder the attributes, you will have to rebuild your query to have a matching order.
4. The Test JDBC Statement window collects and displays any data in the agent's cache since it was last started.
5. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534
6. The agent can be stopped by clicking **Stop Agent**
7. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 22. Monitoring system availability using Ping

Part of network management involves the ability to determine if systems respond to an Internet Control Message Protocol (ICMP) ping. Using this data provider you can monitor basic online or offline status for a set of servers or other critical devices in your environment in a simple and low-overhead fashion. To monitor a list of systems, add the Ping data collector to your agent.

Ping configuration files

The agent must be provided with the list of systems to be monitored. The files 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 of the entries 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: The entries in the ping configuration file must either contain the fully qualified path to the device list file or be relative to the location of the ping configuration file.

A device list file is a text file containing 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 once 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 once. If multiple names resolve to the same IP address, each entry will be 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 and a device can be pinged by more than one of the device monitoring threads.

Procedure

Use the following procedure to monitor a list of systems:

1. On the Agent Initial Data Source page (Figure 183) or the Data Source Location page, click **Network management data** in the **Monitoring Data Categories** area.

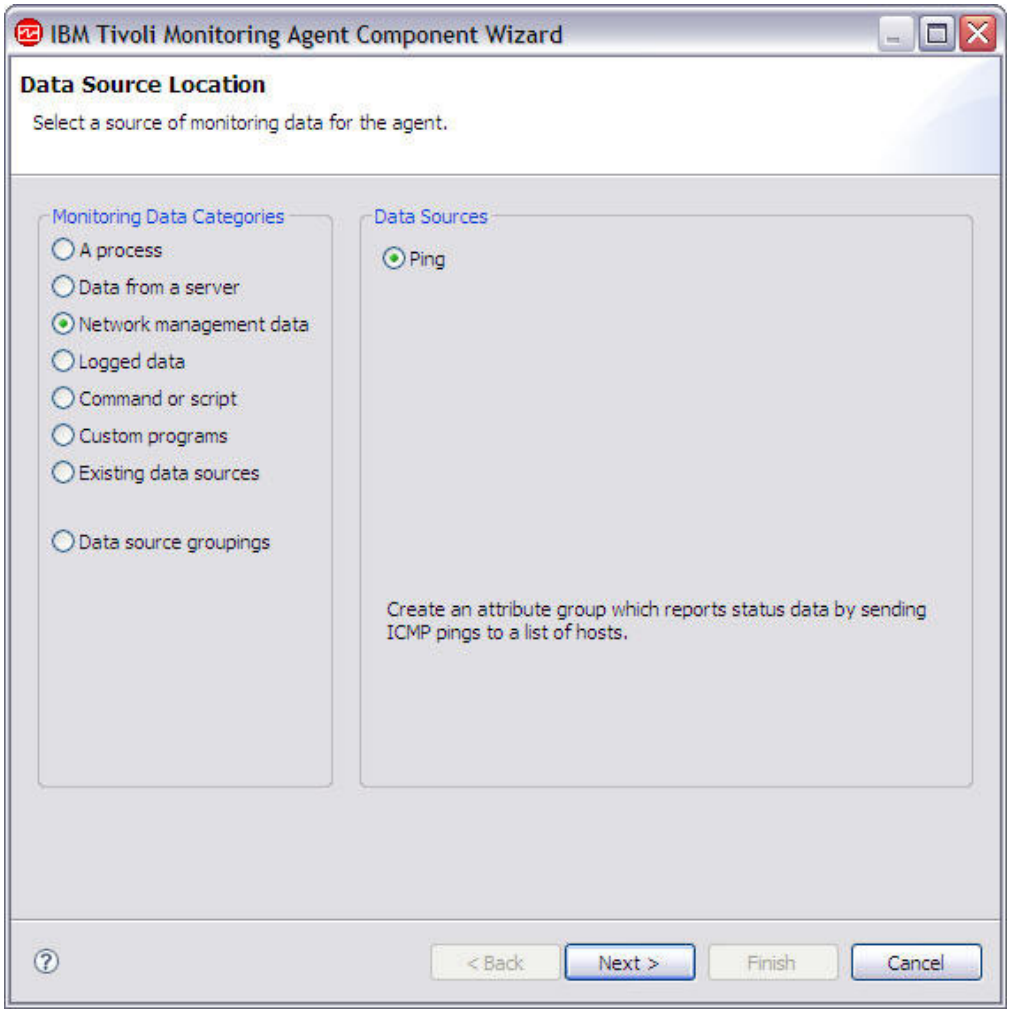


Figure 183. Adding Ping data

2. In the **Data Sources** area, click **Ping**.
3. Click **Next**.
4. In the **Operating Systems** area in the Ping Information window (Figure 184 on page 257), select the operating systems.

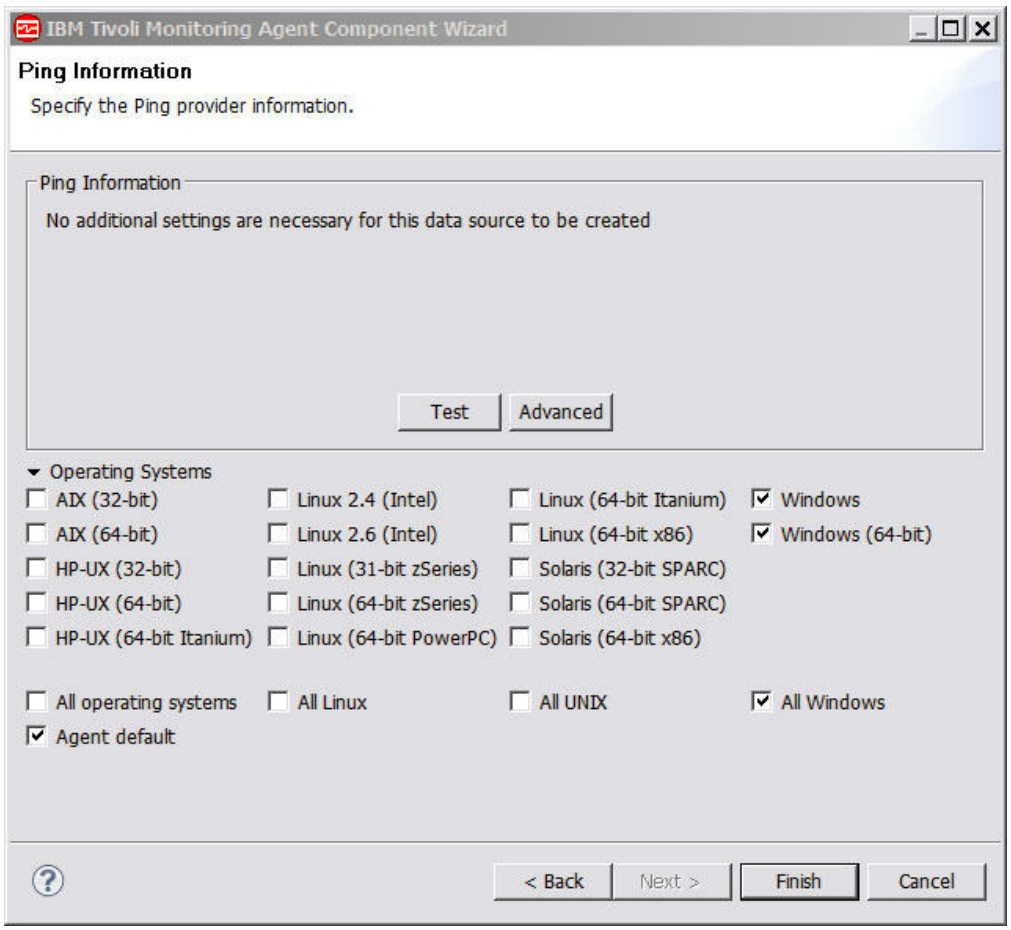


Figure 184. Ping Information window

5. *(Optional)* You can test this attribute group by clicking **Test**. For more information about testing, see "Testing" on page 258
6. *(Optional)* You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66
7. Do one of the following steps:
   a. If you are using the New Agent wizard, click **Next**.
      —OR—
   b. Click **Finish** to save the data source and open the Agent Editor.
8. If you want to add attributes and supply the information for them, see "Creating attributes" on page 54 for more information.

For more information about the attribute group for Ping, see ("Ping attribute group" on page 563).

Configuration

After a data source has been added, the configuration is displayed on the Runtime Configuration page of the Agent Editor.

The Network Management configuration section contains the following property:

Table 14. 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 containing a list of files, which each contain a list of hosts to monitor using ICMP pings.

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the Ping Information page (Figure 184 on page 257).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see “Tivoli Monitoring Agent Editor” on page 37

After you click **Test** in one of the previous two steps, the Test Settings window Figure 185 is displayed

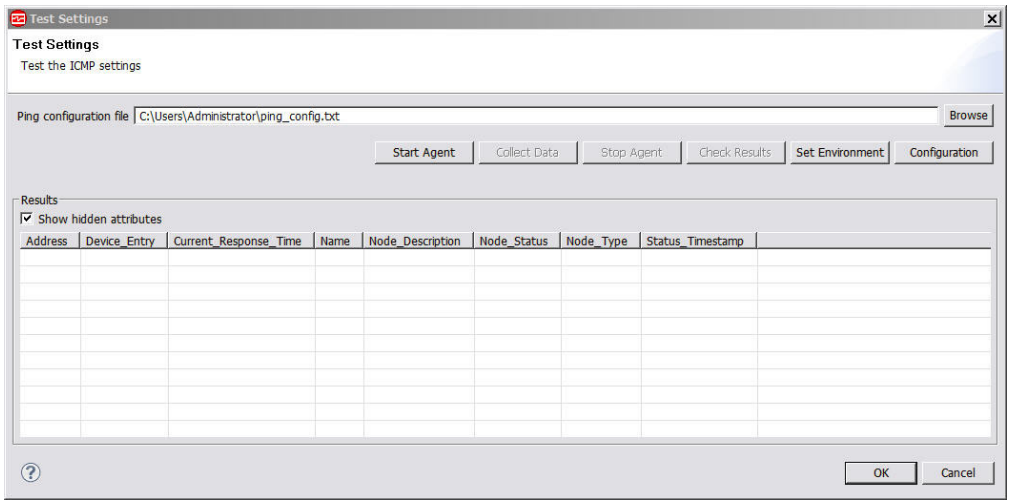


Figure 185. Ping Test Settings window

To start and test the data source use the following procedure

1. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see “Configuration prior to testing” on page 374.

2. Click **Browse** to select a Ping configuration file. For more about Ping configuration files, see “Ping configuration files” on page 255
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 pings the devices specified in the device list file which is referenced from the Ping configuration file.
5. The Test Settings window collects and displays any data in the agent's cache since it was last started.
6. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in “Performance Object Status node” on page 534
7. The agent can be stopped by clicking **Stop Agent**
8. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, “Testing and debugging your agent,” on page 373

Chapter 23. Monitoring HTTP availability and response time

Using the HTTP data source, you can monitor the availability and response time of selected URLs. For each URL, the results provide general information about the HTTP response to the HTTP request, including 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:

- Sun Microsystems 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:

1. On the Agent Initial Data Source page (Figure 186 on page 262) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

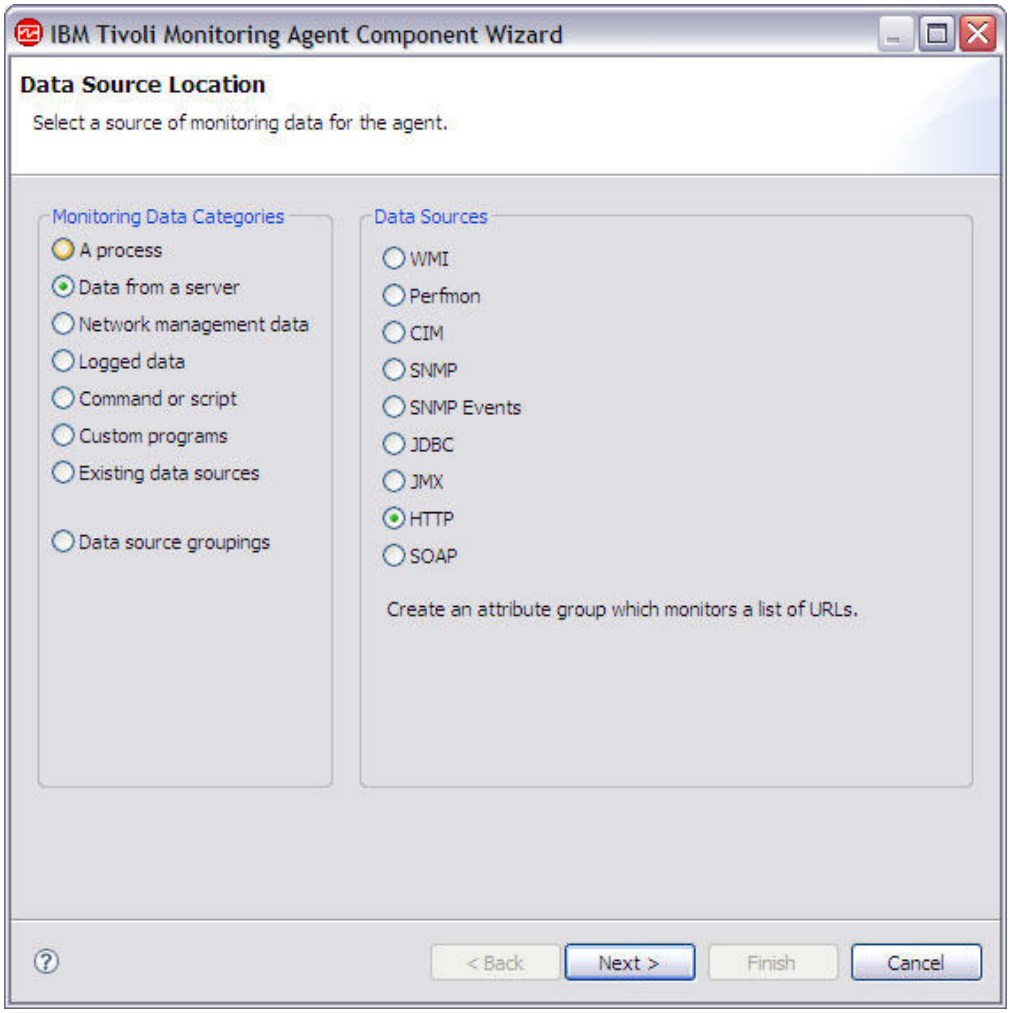


Figure 186. Adding HTTP data

2. In the **Data Sources** area, click **HTTP**.
3. Click **Next**.
4. On the HTTP Information page (Figure 187 on page 263), select one or more operating systems in the **Operating Systems** area.

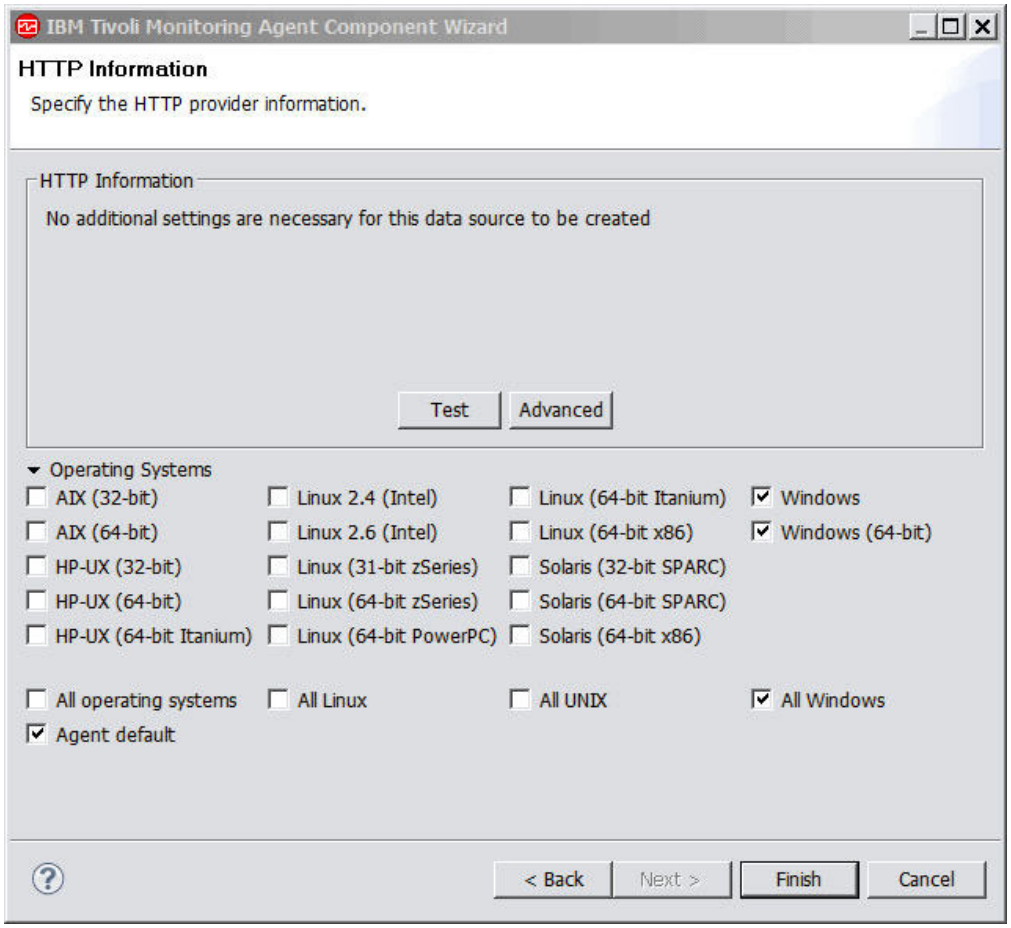


Figure 187. HTTP Information window

5. *(Optional)* You can test this attribute group by clicking **Test**. For more information about testing, see "Testing" on page 271
6. *(Optional)* You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66
7. Do one of the following steps:
   a. If you are using the New Agent Wizard, click **Next**.
      —OR—
   b. Click **Finish** to save the data source and open the Agent Editor.
8. The HTTP data source creates two attribute groups: Managed URLs and URL Objects. For information about these attribute groups, see "HTTP attribute groups" on page 566. You can add, modify, or delete attributes. For more information about creating attributes, see "Creating attributes" on page 54.

HTTP tables

The two attribute groups 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, such as the .gif and .jpg files that might be used in the website listed in the Managed URL report.

For information about the syntax used in the Managed URLs and URL Objects tables, see "Specific fields for HTTP attributes."

When you need 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 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 15. 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 monitored is the image 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 need to monitor objects found in a web page, in the URL Monitoring configuration section, set the **Page object collection** property to **No**.

Specific fields for HTTP attributes

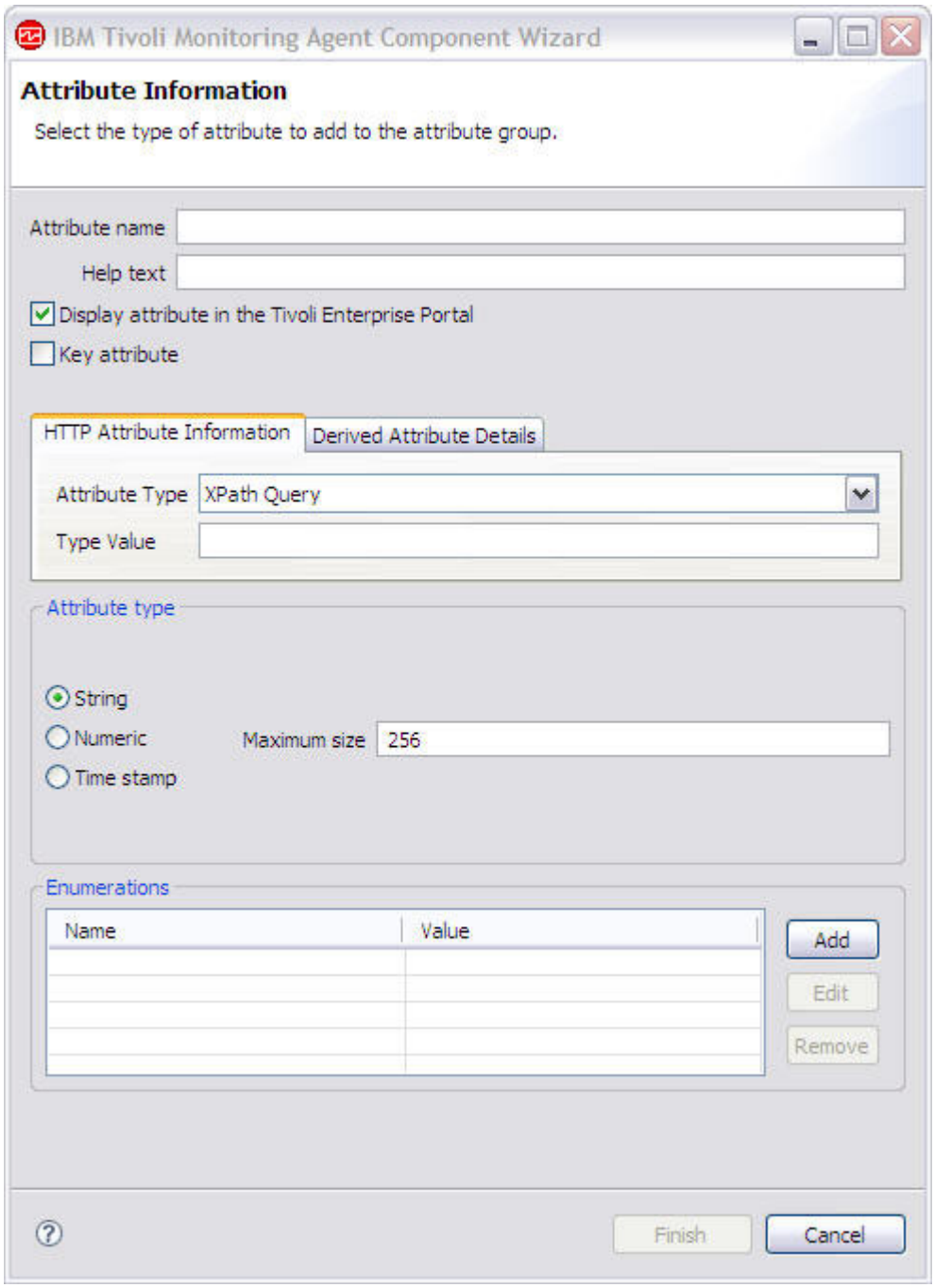


Figure 188. Attribute Information page

In the Attribute information page Figure 188, 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 16. HTTP Attribute Information - Managed URLs

Attribute type	Description	Type value	Data type returned	Differences with FTP and file protocols
XPath Query	Runs an XPath query on the content 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 obtained from a URL connection.	The data 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 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

Table 16. HTTP Attribute Information - Managed URLs (continued)

Attribute type	Description	Type value	Data type returned	Differences with FTP and file protocols
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.
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 obtained from the content returned by accessing this URL.	None	String	None
Page Objects	The number of objects 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 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 17. HTTP Attribute Information - URL Objects

Attribute type	Description	Type value	Data type 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 being monitored within the HTML page.	None	String	None
Object Size	The size in bytes of the content 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 located 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" on page 269. 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 appear in the URLs file depending on the method by which they were added.

Table 18. 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 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

Table 18. URLs file entries (continued)

URLs	Added by
`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 called URL Add. When this option is selected, a window is displayed where you can specify the following parameters:

URL A required parameter representing 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 permitted 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_Data defaults to INITCNFG.

After you complete the information and close the window, assign the HTTP URL Add action to the destination managed system 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, called HTTP URL Remove, permits immediate stopping of 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, and if you did not specify User_Data, you must specify INITCNFG as seen in the Tivoli Enterprise Portal.

If a URL was added manually to the URLs file, then you can delete it with the Take Action if you 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 seen in the Tivoli Enterprise Portal.

After completing the information and closing the window, assign the URL Remove action to the destination managed system associated with the agent.

Monitoring https:// URLs

The HTTP data source can only monitor 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

After an HTTP data source has been 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 19. URL Monitoring configuration properties

Name	Valid values	Required	Description
HTTP URLs file	Path to a file	Yes	The path to the file containing a list of URLs.
Page object collection	Yes, No The default value is Yes.	No	Whether to download objects found in a web page and collect data from them.

Proxy Server configuration

The Proxy Server configuration section contains the following properties:

Table 20. Proxy Server configuration properties

Name	Valid values	Required	Description
HTTP proxy hostname	String	No	The proxy host name to be used for HTTP connections.
HTTP proxy user name	String	No	The user name for the proxy server.
HTTP proxy port	Positive integer The default value is 80.	No	The HTTP port number of the proxy server.
HTTP proxy password	Password	No	The password for the proxy server.

Note: If the HTTP proxy hostname property is blank, no proxy is used.

Java configuration

If you define a 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 21. 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	Using this property you can specify the trace level used by the Java providers. For more information, see "Trace log format" on page 414.
JVM Arguments	String	No	Using this property you can specify an optional list of arguments to the Java virtual machine.

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the Http Information page (Figure 187 on page 263).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37

After you click **Test** in one of the previous two steps, the HTTP Test window Figure 189 is displayed

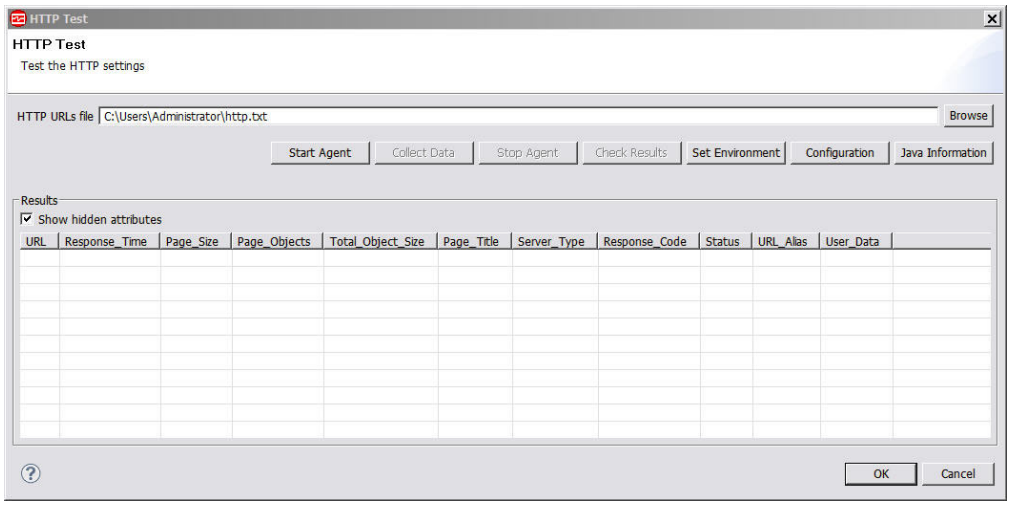


Figure 189. HTTP Test window

To start and test the data source use the following procedure

1. Click **Browse** to select the HTTP URLs file, for more information about URLs files, see "URLs file" on page 268
2. *(Optional)* Before starting your test you can set environment variables, configuration properties and Java information. For more information, see "Configuration prior to testing" on page 374. For more about HTTP configuration, see "HTTP configuration" on page 270.
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 URLs defined in the HTTP URLs file.
5. The HTTP Test window displays any data that is returned.
6. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534
7. The agent can be stopped by clicking **Stop Agent**
8. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 24. Monitoring data using SOAP

Using the Simple Object Access Protocol (SOAP) data source, you can specify an HTTP URL and send a GET, POST, or PUT request (with the associated POST data for POST or PUT requests). An XML response is retrieved and parsed, with the data being 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:

1. On the Agent Initial Data Source page (Figure 190) or the Data Source Location page, click **Data from a server** in the **Monitoring Data Categories** area.

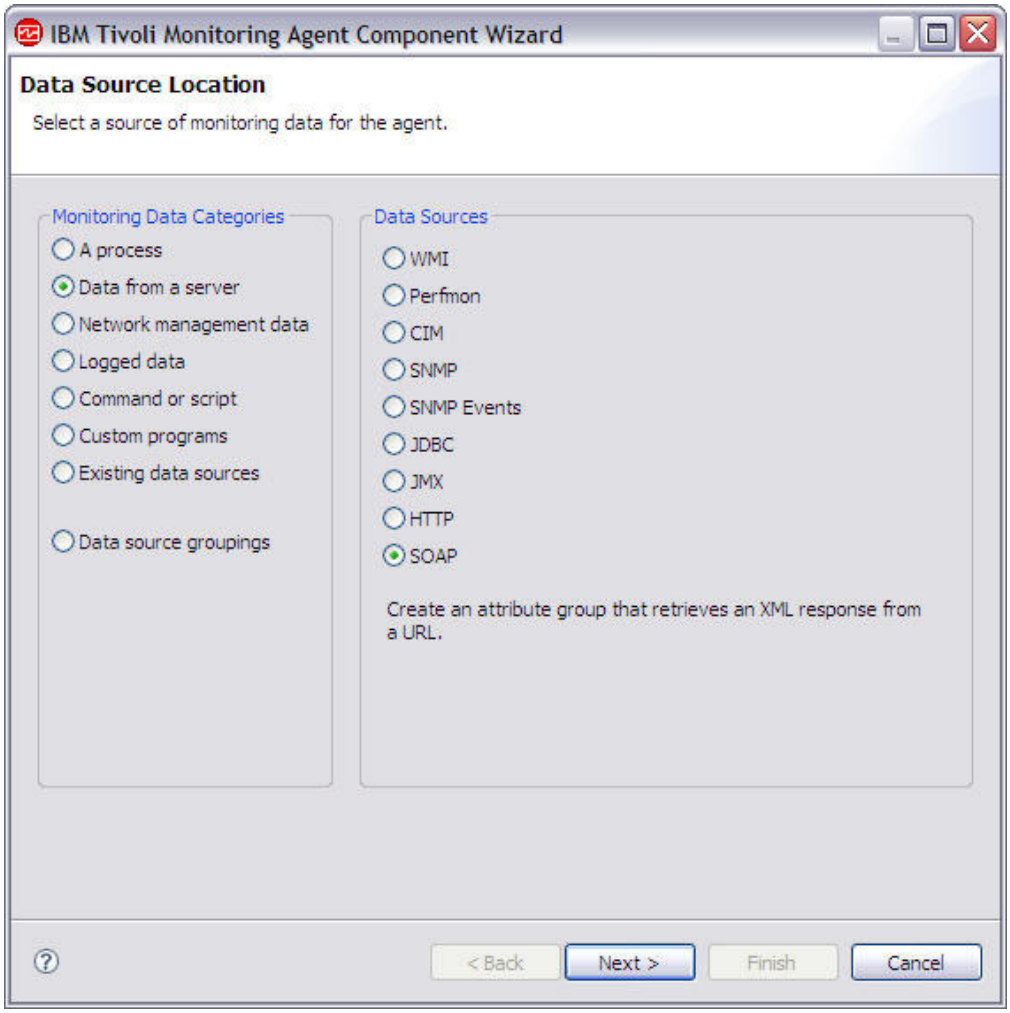


Figure 190. Adding HTTP data

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

4. On the SOAP Information page (Figure 191), 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 "Customizing configuration" on page 359.

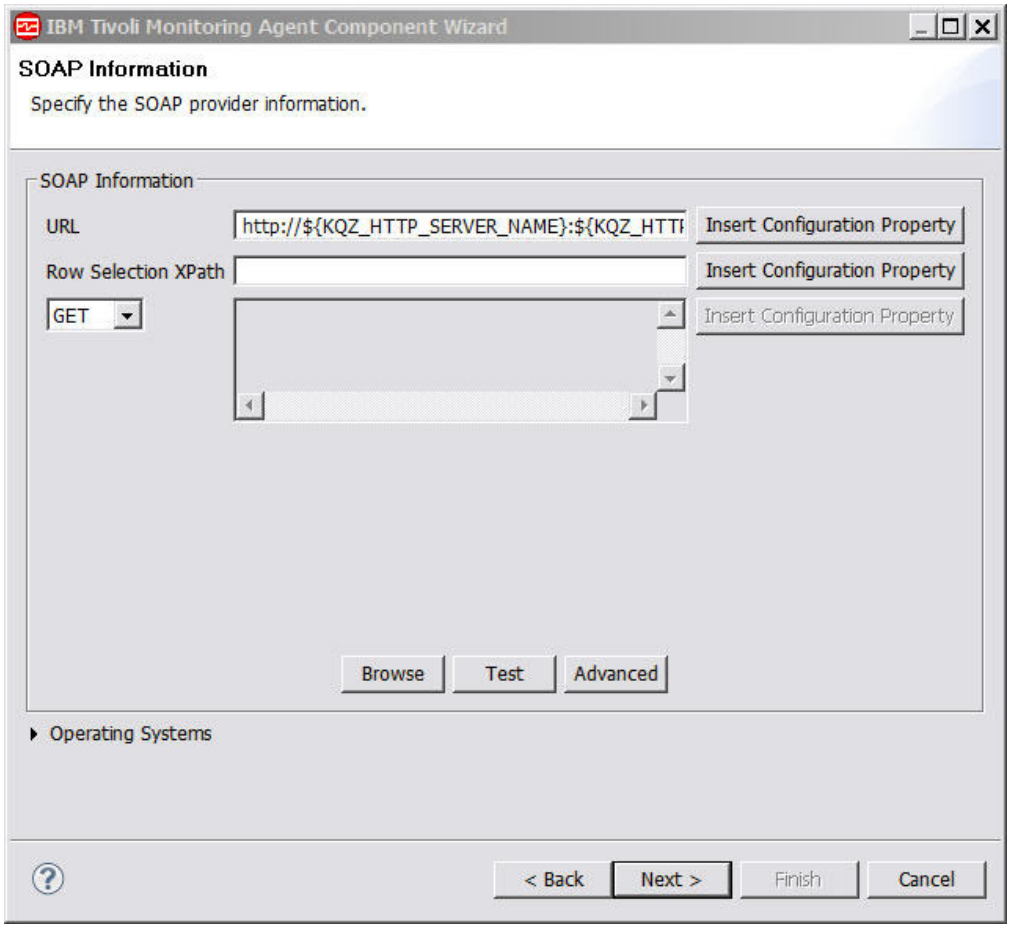


Figure 191. SOAP Information page

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 "Customizing configuration" on page 359.

6. Click **Browse**.

   Note: After having entered a URL and selected a request type, if you do not want to use the SOAP browser to build the definition, you can enter a Row Selection XPath in the SOAP Information window. You then define all of the attributes for the attribute group.

7. In the SOAP Browser window Figure 192 on page 275, do the following:

a. Enter a URL and select a request type if you have not already done so.
b. Use the **Configuration** button 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 displayed in a Document Object Model (DOM) tree. In the WebSphere Application Server example in Figure 192, the following URL was entered:

   ```
   http://nc053011.tivlab.raleigh.ibm.com:9080/wasPerfTool/servlet
   /perfservlet?module= \threadPoolModule
   ```

   The `PerformanceMonitor` XML element is displayed. This is the top-level XML element in the XML document returned by the SOAP provider.

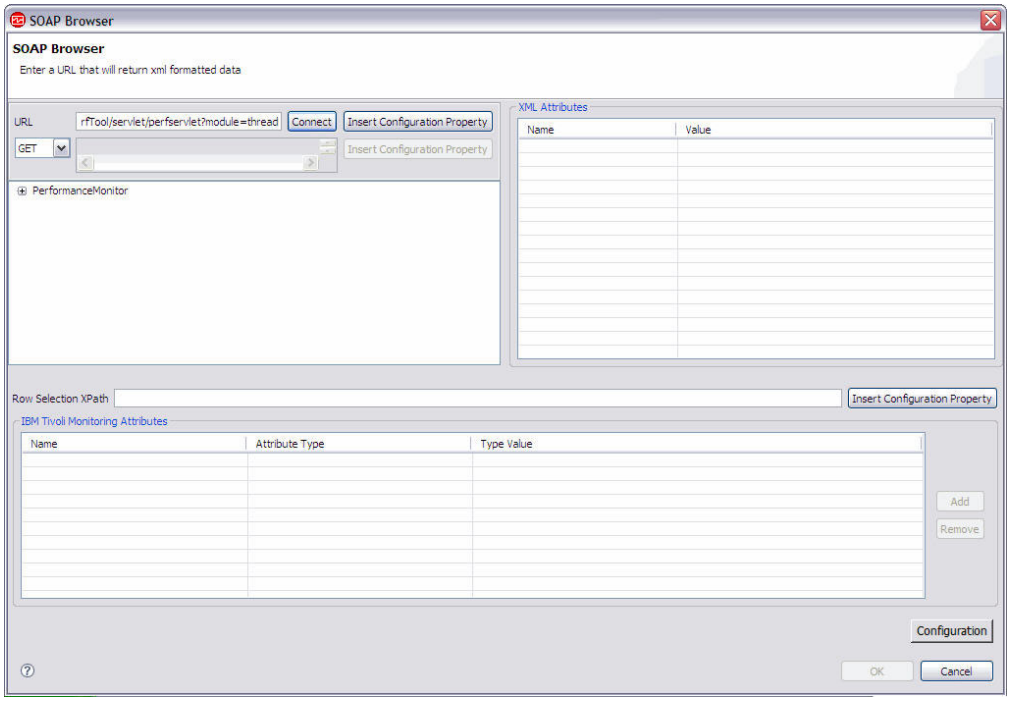


Figure 192. SOAP Browser window

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 193 on page 276, the `PerformanceMonitor/Node/Server/Stat/Stat/Stat` node has been selected. This node represents a row of data in the attribute group. When you select a node in the DOM tree and click **Add** in the **IBM Tivoli Monitoring Attributes** area, you get all of the attributes and elements defined on and within that node of the DOM tree.

   When a node is selected, the **XML Attributes** area displays 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 set with only one item, the attribute group contains only one row.

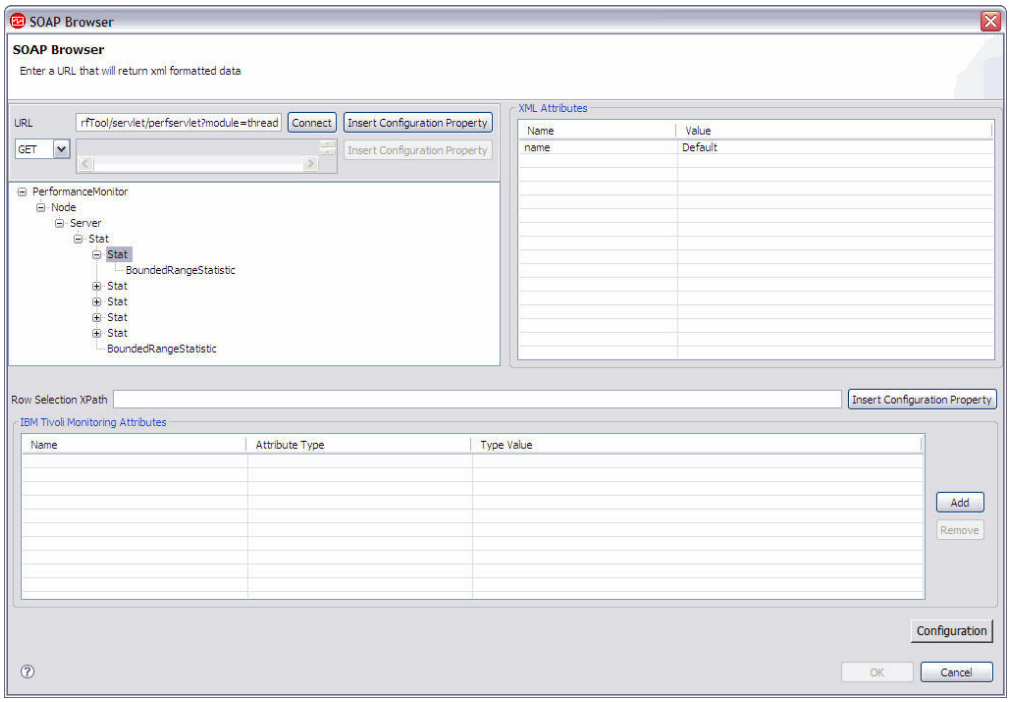


Figure 193. SOAP Browser window

e. Click **Add** in the IBM Tivoli Monitoring Attributes area. The list of Tivoli Monitoring attributes is displayed 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 Figure 194 on page 277, 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 280

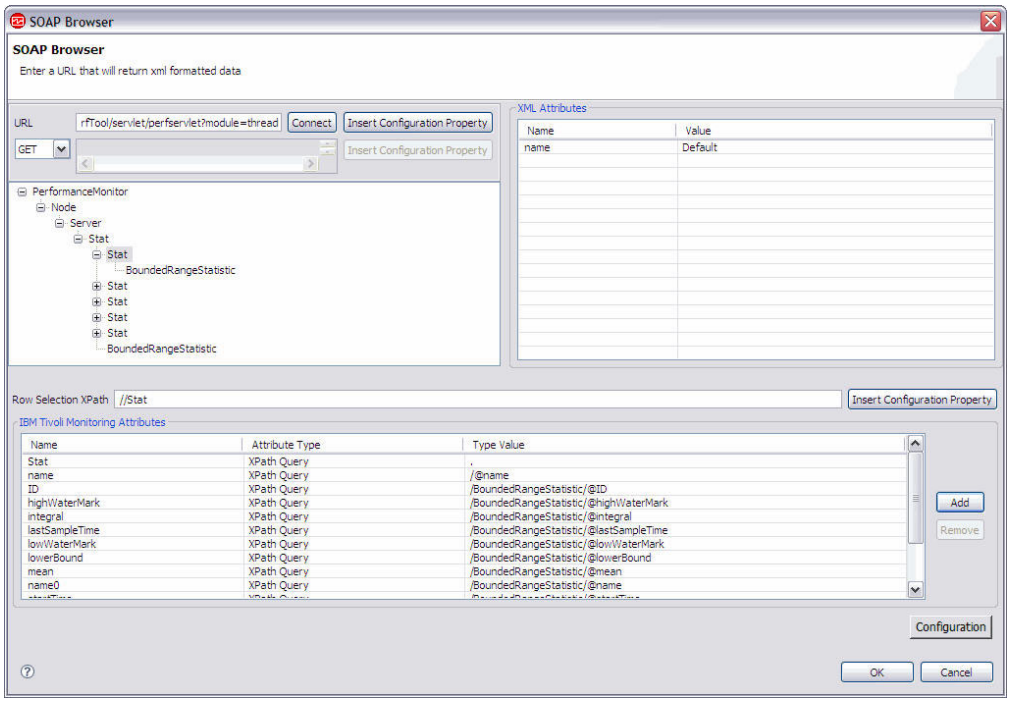


Figure 194. SOAP Browser window

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 the Browse functionality in Step 6 and you entered the URL and Row Selection XPath directly in the SOAP Information window, the Attribute Information page is displayed. Specify the information for the first attribute on the Attribute Information page, and click **Finish**. You can then specify additional attributes using the Agent Editor. For more information about creating attributes, see "Creating attributes" on page 54.
10. If you used the Browse functionality in Step 6, 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. See "Selecting key attributes" on page 27 for more information.
11. *(Optional)* You can test this attribute group by clicking **Test**. For more information about testing, see "Testing" on page 281
12. *(Optional)* You can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66
13. Do one of the following steps:
    a. If you are using the New Agent Wizard, click **Next**.
    —OR—
    b. Click **Finish** to save the data source and open the Agent Editor.

Specific fields for SOAP attributes

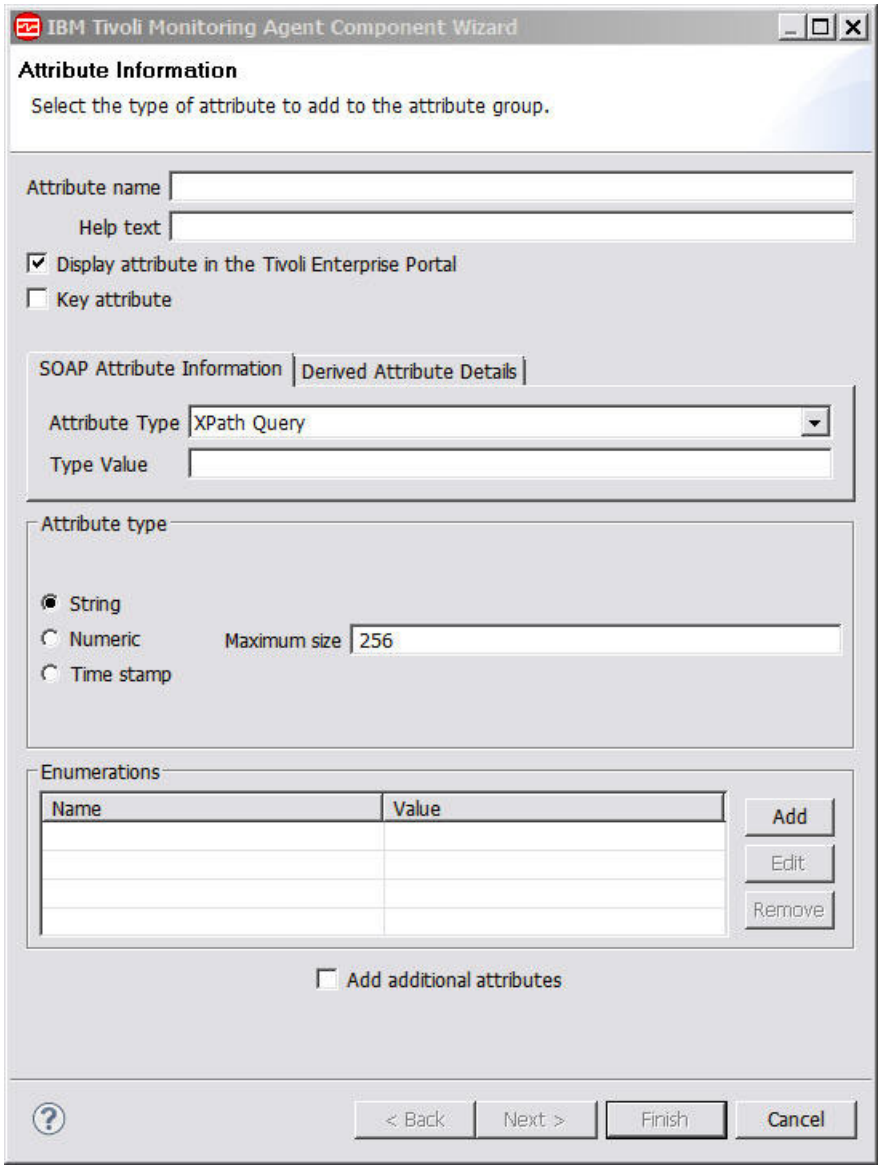


Figure 195. Attribute Information page

In the Attribute information window Figure 195, there are two fields for SOAP attributes that define how data is collected from the SOAP response. 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 22. SOAP Attribute Information

Attribute type	Description	Type value	Data type returned	Differences with FTP and file protocols
XPath Query	Runs an XPath query on the content 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 obtained from a URL connection. If a row selection query was defined, this XPath query should be relative to the row selection query.	The data 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 being browsed. If the data is in the 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

Table 22. SOAP Attribute Information (continued)

Attribute type	Description	Type value	Data type returned	Differences with FTP and file protocols
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.
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 obtained from the content 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, such as position(), first(), last(), and count() in the XPath, if a row contains multiple XML elements of the same type, and the position of an XML element in the node list determines the Tivoli Monitoring attribute the element maps to.

- 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
```

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 270.

SOAP configuration

After a SOAP data source has been 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 270. For information about Java configuration, see "Java configuration" on page 270.

HTTP Server

The HTTP Server configuration section contains the following properties:

Table 23. 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

Testing

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

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** on the SOAP Information page (Figure 191 on page 274).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see “Tivoli Monitoring Agent Editor” on page 37

After you click **Test** in one of the previous two steps, the SOAP Test window Figure 196 is displayed

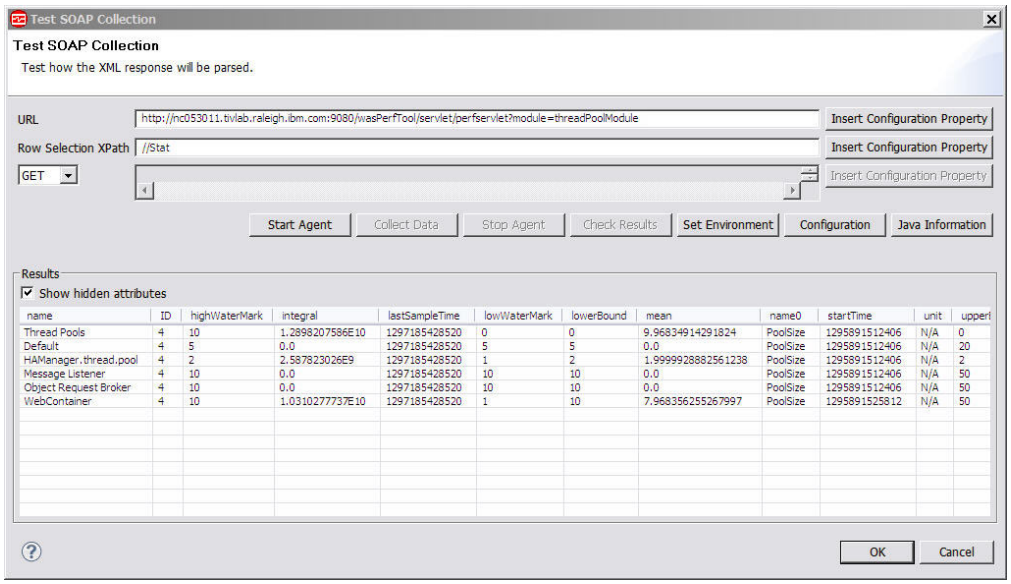


Figure 196. Test SOAP Collection window

To start and test the data source use the following procedure:

1. *(Optional)* Before starting your test you can set environment variables, configuration properties and Java information. For more information, see “Configuration prior to testing” on page 374. For more about SOAP configuration, see “SOAP configuration” on page 281.
2. Make any changes you require to the URL, Row Selection XPath, and request type.
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** to populate the Results table and preview how the data will be parsed and shown in columns in the Tivoli Enterprise Portal.

   In the Results area, you can make changes to the attribute definitions and then reload the data to see how your changes will affect what will be stored in the attribute group. Right-click in a column to display options to edit the attribute, hide the attribute, insert an attribute before the selected attribute, insert an attribute after the selected attribute, remove the attribute, remove all subsequent attributes (all attributes that appear to the right of the selected attribute in the Results table), and remove all attributes.
5. The agent can be stopped by clicking **Stop Agent**
6. Click **OK** or **Cancel** to return to the SOAP Browser window. Clicking **OK** saves any changes you have made.

For a general discussion about Agent Testing, see Chapter 31, “Testing and debugging your agent,” on page 373

Chapter 25. Monitoring data using Socket

The socket data source offers a convenient means for data to be provided 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 could 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. Note that the agent does not start or stop the application that is sending data to the socket; this is controlled by the user.

Use the following procedure to create an attribute group to collect data using a Transmission Control Protocol socket (TCP) socket.

1. On the Agent Initial Data Source page (Figure 197) or the Data Source Location page, click **Custom programs** in the **Monitoring Data Categories** area.

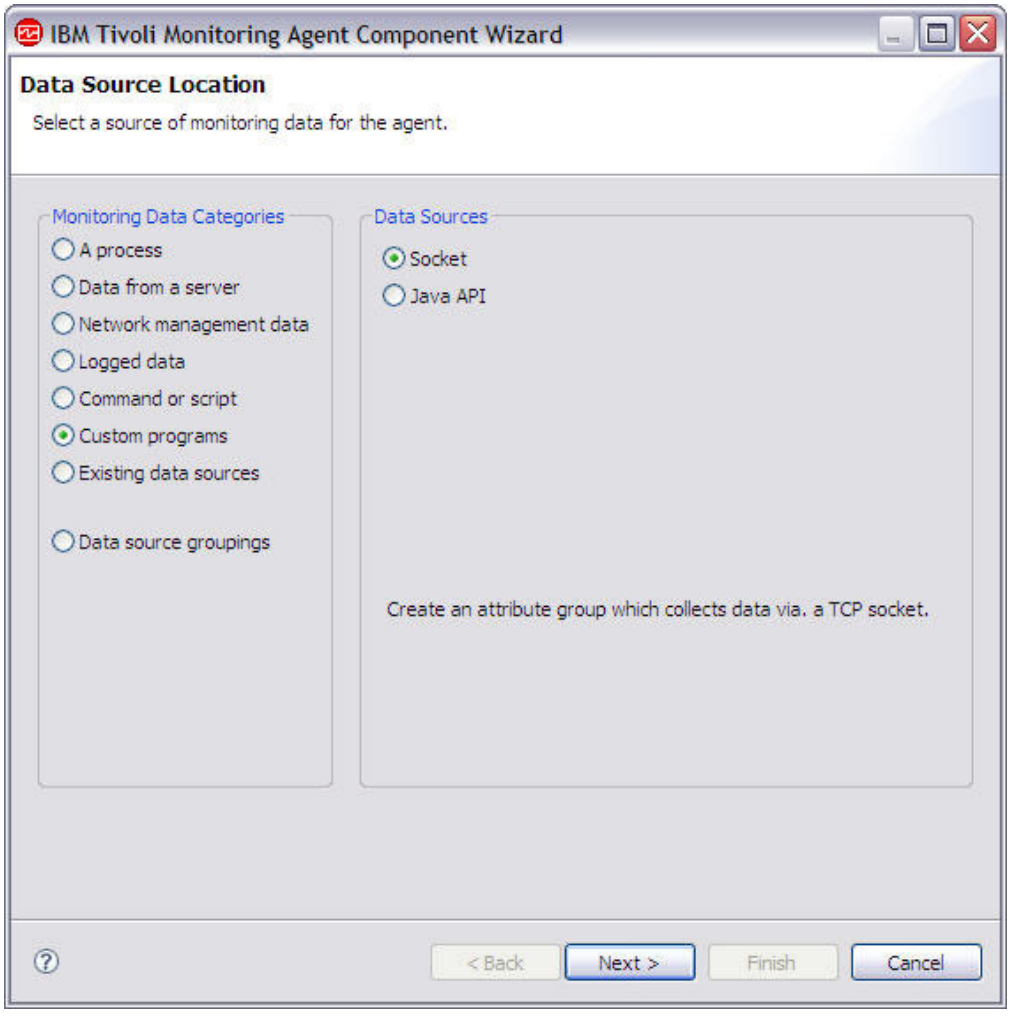


Figure 197. Monitoring data with Socket

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

4. On the Socket Information page, enter an Attribute group name.

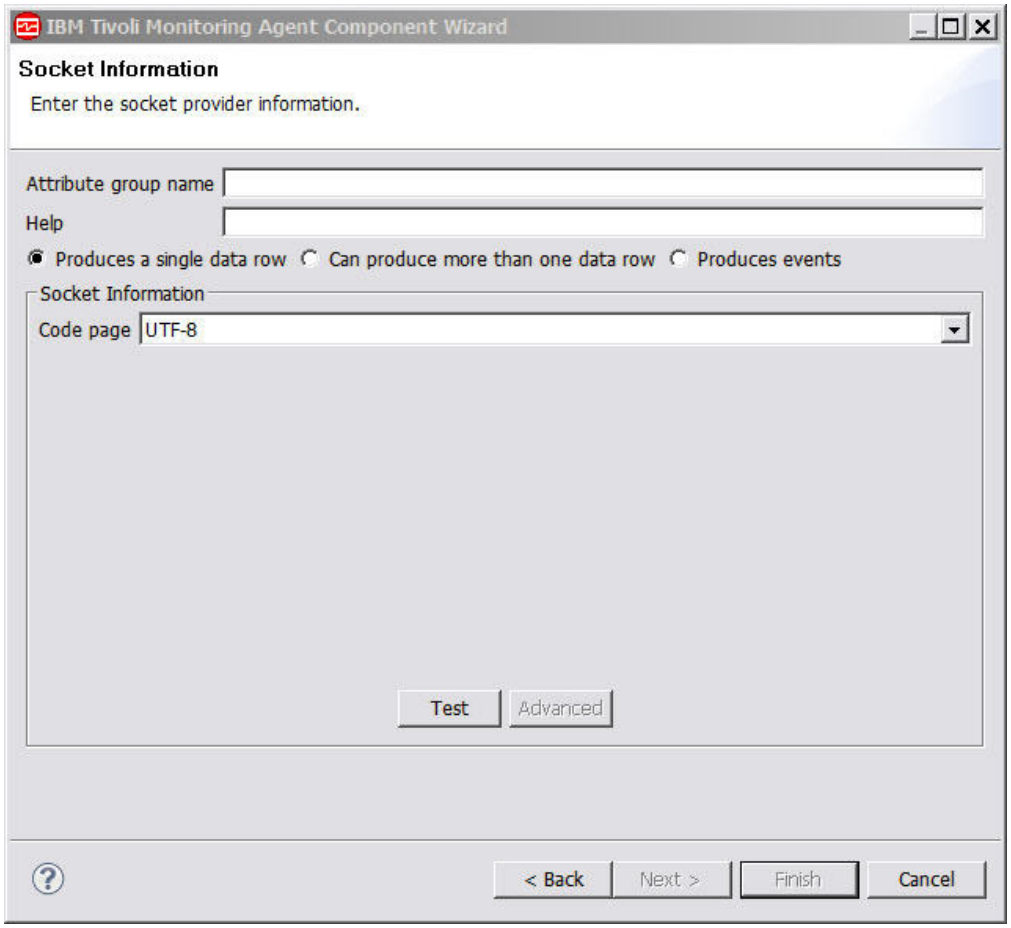


Figure 198. Socket Information window

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 288.
7. In the Socket Information section, select a Code page. For more information, see "Character sets" on page 292.
8. *(Optional)* Click **Advanced** to modify the advanced properties for the attribute group. The **Advanced** button is active when you have selected 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 54.
11. Click **Next**.
12. *(Optional)* On the Global Socket Data Source Information page, **Error Codes** section Figure 199 on page 285, you can define the error codes that the socket client can send if it cannot collect data. For more information, see "Sending errors instead of data" on page 290. To define an error code, do the following:

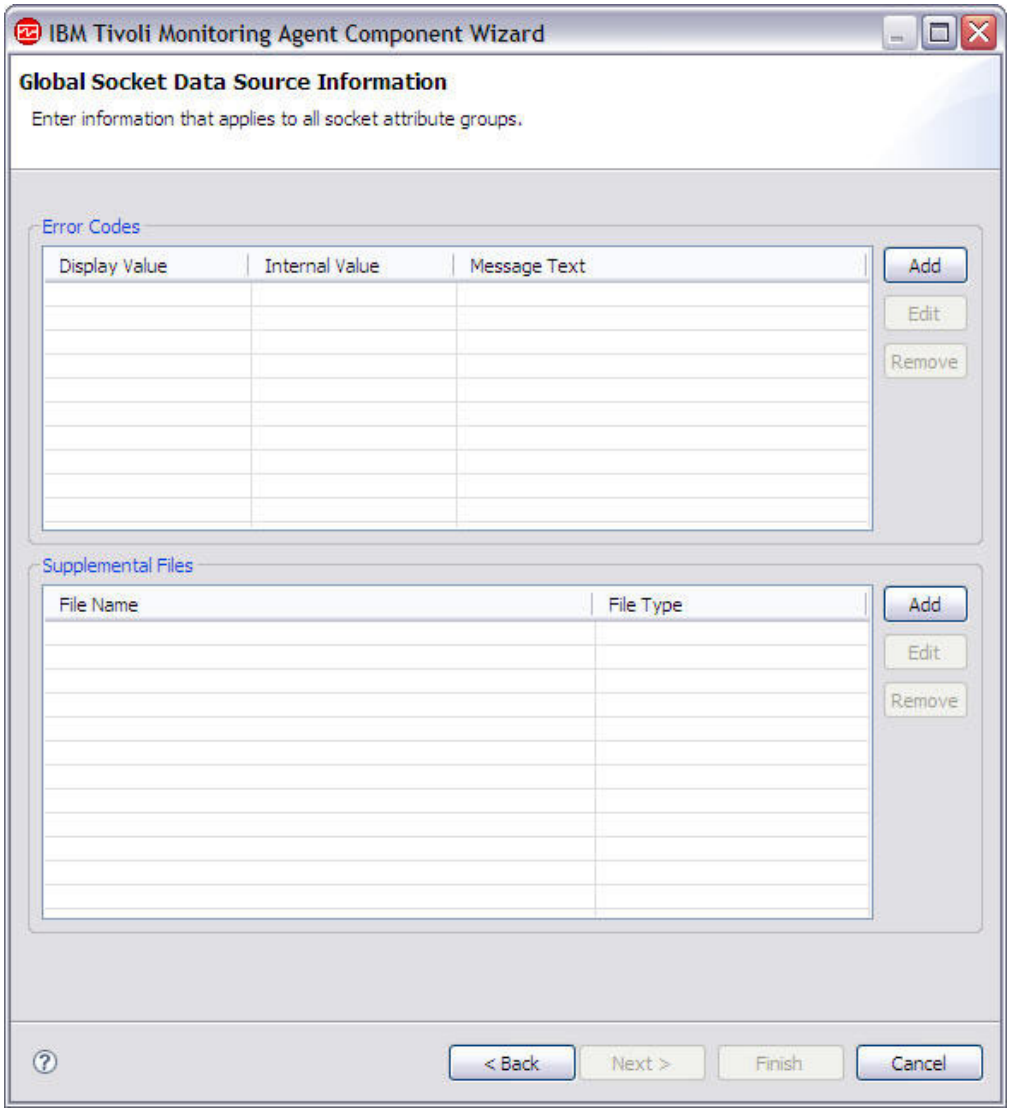


Figure 199. Global Socket Data Source Information page

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 permitted.

b. In the Socket Error Code Definition window Figure 200 on page 286, enter a display value that is shown in the Peformance Object Status attribute group.

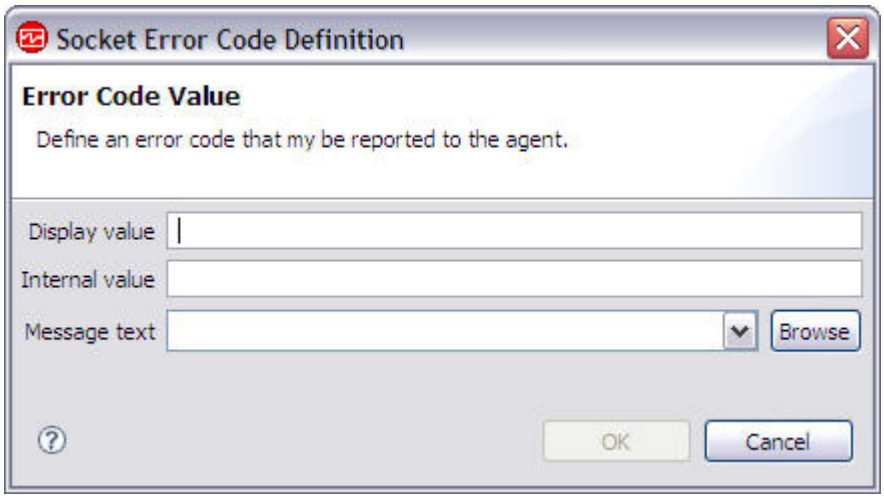


Figure 200. Socket Error Code Definition window,

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 drop-down list. Click **OK** to return to the Global Socket Data Source Information page. The message will be logged in the agent log file.

   If no suitable message text is available, click **Browse** to set up the message text. The Messages (list) window is displayed (Figure 201). The message window lists messages that are defined in the agent. Until you have defined messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you have defined.

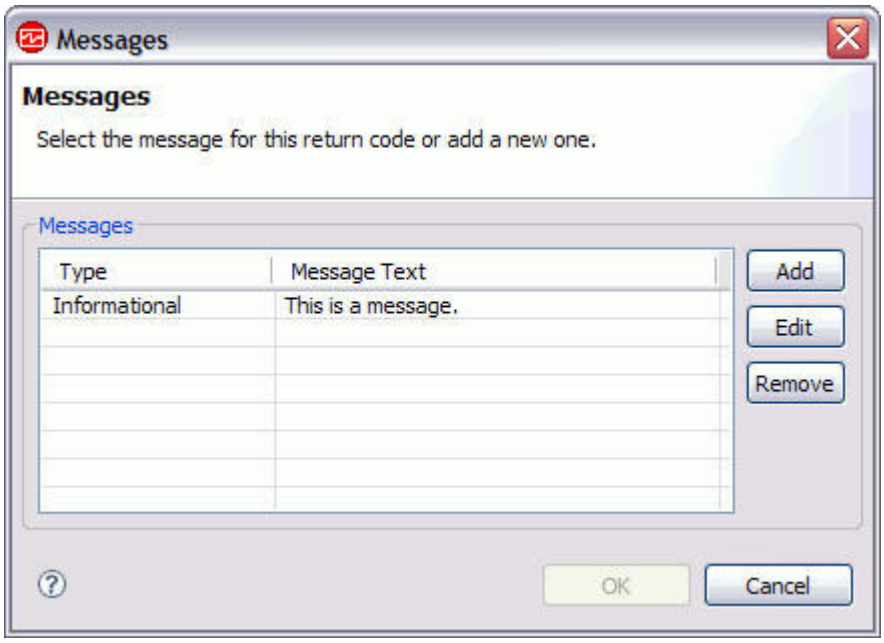


Figure 201. Messages window

e. In the Messages (list) window, click **Add** to see a Message Definition window (Figure 202), where you can type the text that describes the meaning of the new message and select the message type.

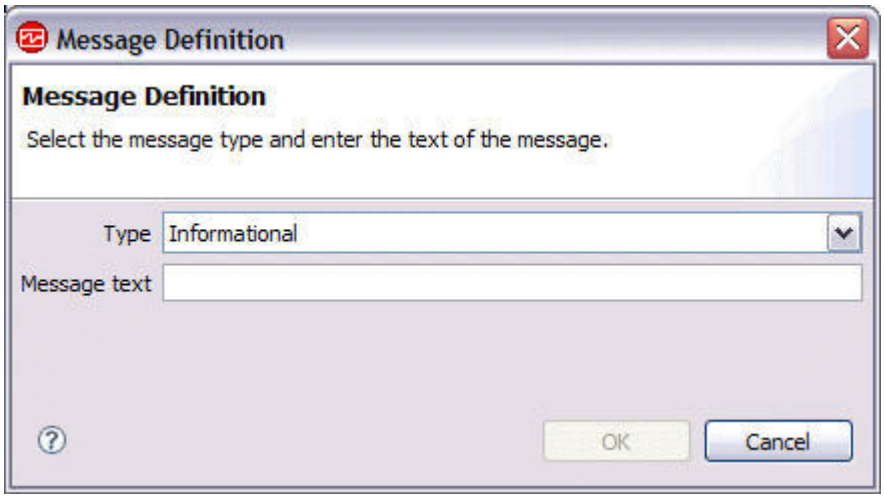


Figure 202. Message Definition window

Note: The message identifier is automatically generated for you.

f. Click **OK**.

g. The Messages (list) window is displayed 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 and 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 24. File types for supplemental files

File Type	Description
Executable	Select this option if you want to include an executable with the agent. The agent will not use these files.
Library	Select this option if you to include a library with the agent. The agent will not use these files.
Java resource	Select this option to include Java resources with the agent. The agent will not use these files.

For information about where the Supplemental Files are installed with your agent, see “New files on your system” on page 382.

Click **Edit** to make changes to the imported file. For more information, see “Editing a command file definition” on page 222.

14. *(Optional)* You can test this attribute group by clicking **Test**. For more information about testing, see “Testing” on page 295

15. *(Optional)* If the data source is sampled (you did not select "Produces events" on the Socket Information page), you can create a filter to limit the data

returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 66

16. Do one of the following steps:
    a. If you are using the New Agent Wizard, click **Next**.
       —OR—
    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 this 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 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 data received must follow a structured XML format. The following XML information flows are possible 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 that the attribute group is to produce one or more data rows, that is a sampled attribute group. If you select Produces events then your attribute group sends an event to Tivoli Monitoring each time a row is received.

When you view sampled data in the Tivoli Enterprise Portal, you see the latest set of collected rows. The data displayed for an event attribute group is the contents of a local cache 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 when you create your attribute group, 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, just 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 or 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 will 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 will show 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 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 defined in your attribute group. The only exception to this is that the derived attributes must be skipped, regardless of where they appear 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 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"`. Note that `"app1"` is not the managed system name, but the instance identifier specified when the subnode instance is configured.

The following XML elements make up the socket data:

socketData
: This is the root element. It has one optional attribute 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 may 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 that is updated with the Error Code Type that is defined for the error code you send.

For the example above, you must have defined the Error Code Value of 1000 in the agent. See the following sample error code definition:

Table 25. 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 the Performance Object Status query from the Tivoli Enterprise Portal, you will see the value `APP_NOT_RUNNING` in the **Error Code** column of the row for the **abc** attribute group 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 have previously been sent.

Handling take actions

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 the take action will use. 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 does not timeout 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 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 could receive:

```
<taskRequest id="1"><task command="K42 refresh" user="sysadmin"/></taskRequest>\n
```

The id is a unique identifier the agent uses to track requests that have been 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 it is shown in the Take Action Status query that is included with the agent. The dialog that is displayed on the Tivoli Enterprise Portal after running an action does not show the return code. That dialog simply 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 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 additional actions that start with the registered prefix, they will be passed to the client, which must be developed to handle those scenarios in an appropriate manner.

There is a timeout that controls how long the agent will wait for a response from the socket client. The setting is an environment variable that is defined in the agent called CDP_DP_ACTION_TIMEOUT and the default value is 20 seconds.

Note: The error code messages defined for socket data source attribute groups are not used for take actions. You can return the same return code values, but the agent will not log the message defined or impact the Error Code field in the Performance Object Status attribute group.

Data encoding

It is important to be aware of how your socket client is encoding data 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 using the table

below. You must also be careful to not break the XML syntax with your attribute values. The following table shows the characters that occur in your attribute values that you should encode:

Table 26. 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 will prevent data from being parsed correctly.

The agent does not contain a full-featured XML parser so you should not use special encoding for characters not in Table 26. For example, do not encode `¢` or `&cent` in place of a cent sign ¢.

Character sets

In addition to encoding special characters, the agent needs to know what code page was used to encode your data. Tivoli Monitoring stores and displays data in the UTF-8 character set. Each socket attribute group you define allows you to indicate whether you are sending the data to the agent as UTF-8 data or as the Local Code Page. Be aware of how your client is sending data. If you use a client written in Java, you should specify UTF-8 as the encoding on the writer you use to send data to the agent and UTF-8 as the Code Page for your attribute group.

Numeric Data

Be aware of how you are formatting your numeric attribute values. The numeric values you send to the agent must not contain any special characters. One example of this is the thousands separator character. Other examples are currency symbols or characters describing the units of the value. If the agent has trouble parsing numeric data, the agent logs an error indicating 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 58.

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 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 27. Performance Object Status values

Error Code	Description
NO_ERROR	No error has occurred. This indicates there are no problems with the attribute group. You should be aware that problems with a row of sampled data will not cause the state to change from NO_ERROR. You must validate the number of rows 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. This is not an error. It simply indicates there are no instances of the resources being monitored by this attribute group.
XML_PARSE_ERROR	The agent failed to parse data 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.

Limitations of the Socket data source

- Only connections to the local host (127.0.0.1) are possible as connections from remote systems are not secure.
- There is no mechanism in the socket API for the client to determine what subnodes are available. The client can send data for a given subnode, but it needs to already know the subnode name.

Socket configuration

After a Socket data source has been added, the configuration is displayed on the Runtime Configuration page of the Agent Editor. The Socket configuration section contains the following property:

Table 28. 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 will use to listen on for data from socket clients. A value of 0 indicates an ephemeral port is to be used.

The agent writes the value of the port being used to a file. Socket clients running on the agent computer can later read this file to know which port to connect to. The file that the port is written to is named <*kxx*>_<*instanceName*>_cps.properties where: *kxx* is the 3 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 filename is simply *kxx*_cp.properties.

On Windows, the file is written to the `%CANDLE_HOME%\TMAITM6` directory for 32-bit installations or `%CANDLE_HOME%\TMAITM6_x64` for 64-bit installations. On UNIX, the file is written to `/tmp`.

Sample script for socket

The samples script in this section demonstrate how a socket client could be written.

Perl sample

The following sample Perl script connects to a socket and sends data. This sample was written for an agent running 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

You can test the attribute group you have created within Agent Builder. To test the attribute group you will need a socket client to send data. An example socket client written with perl script can be seen at "Sample script for socket"

You can start the Testing procedure in the following ways:

1. During agent creation click **Test** in the Socket Information window Figure 198 on page 284).
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test**. For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37

After you click **Test** in one of the previous two steps, the Test Socket Client window Figure 203 is displayed

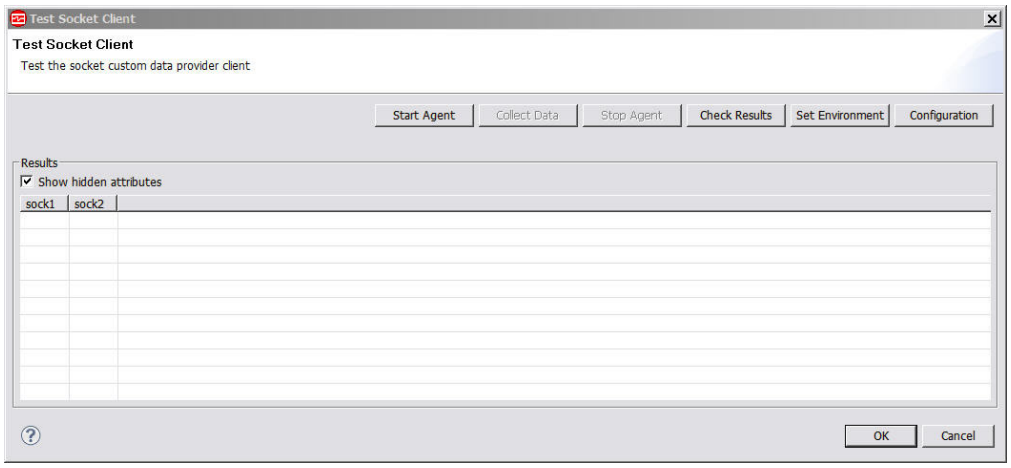


Figure 203. Test Socket Client window

To start and test your agent use the following procedure

1. *(Optional)* Before starting your test you can set environment variables and configuration properties. For more information, see "Configuration prior to testing" on page 374.
2. Click **Start Agent**. A window indicates that the Agent is starting.
3. When the agent starts, it listens for socket data according to its configuration.
4. To test your agent's data collection you generate socket data matching the agents configuration. You can do this using a socket client. When the agent receives socket data matching its configuration, it adds the data to its internal cache.
5. 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.
6. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. This will show you more information about the data, in the Data Collection Status window. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534
7. The agent can be stopped by clicking **Stop Agent**.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 26. Monitoring data using Java API

The agent developer can use the Java application programming interface (API) data source and the Java programming language to carry out data collection in a long-running process that is separate from the agent process. This provides the flexibility to the agent developer to collect data that cannot be collected with the other data sources supported in the Agent Builder. The agent starts the Java application and sends a shutdown request when it is time to shutdown. The Java application must only exit 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. The Java API provides the ability to:

- Connect to the agent process and register for attribute groups 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 and to create a sample Java application to use as a starting point for your Java application.

1. On the Agent Initial Data Source page (Figure 204 on page 298) or the Data Source Location page, click **Custom programs** in the **Monitoring Data Categories** area.

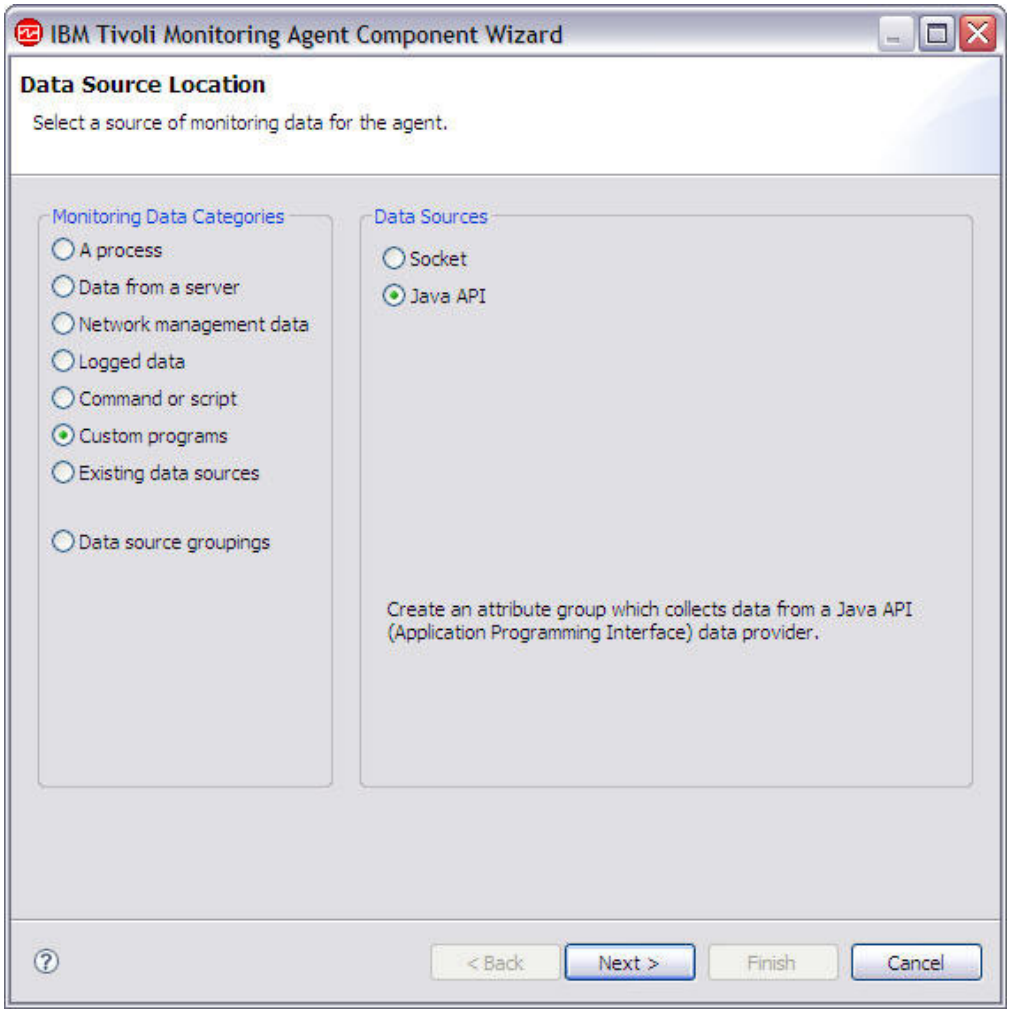


Figure 204. Monitoring data with Java API

2. In the **Data Sources** area, click **Java API**.
3. Click **Next**.
4. On the Java API Information page, enter an Attribute group name.

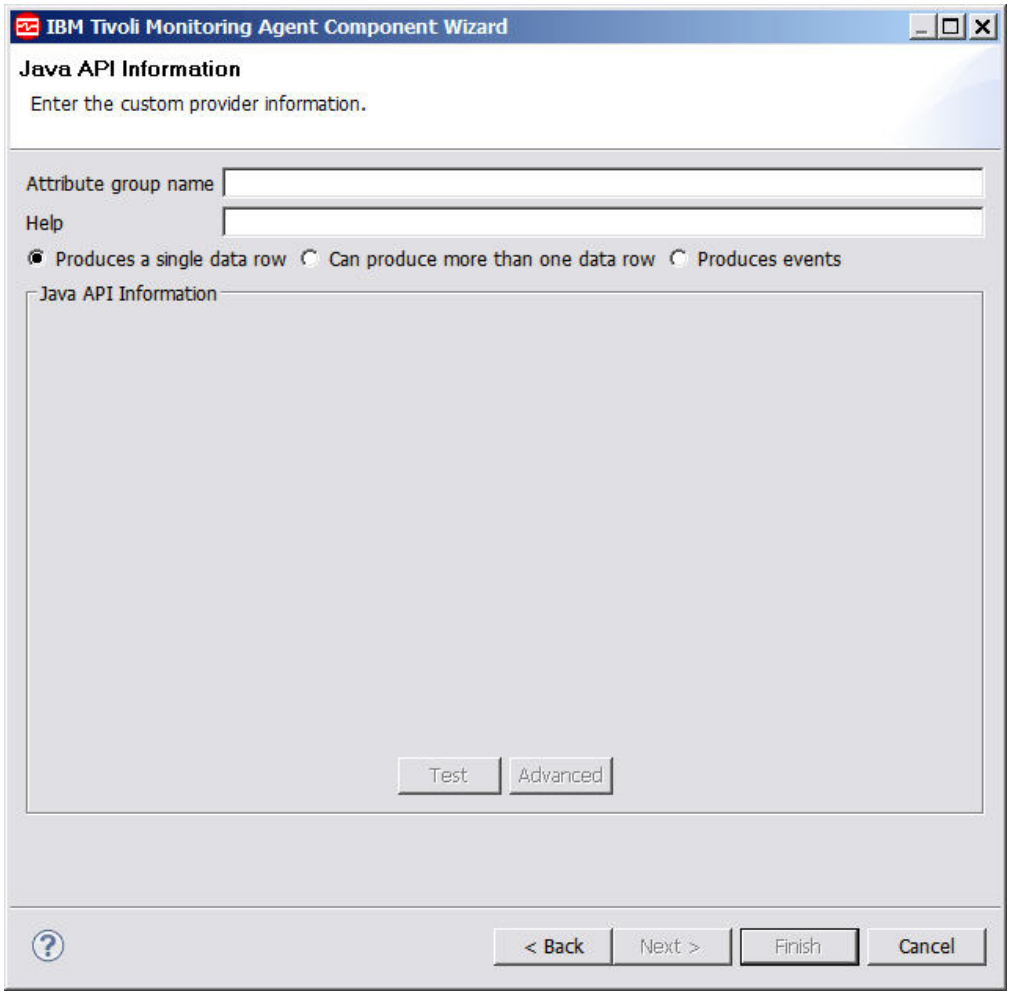


Figure 205. Java API Information window

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 will affect the sample Java application that is created at the end of the wizard. For more information, see "Sending data" on page 288.
7. *(Optional)* Click **Advanced** to modify the advanced properties for the attribute group. The **Advanced** button is active when you have selected 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 54.
10. Select **Add additional attributes** and click **Next** to add other attributes to the agent. References to the attributes will be incorporated into the sample Java application that will be created at the end of the wizard.
11. Click **Next**.
12. On the Global Java API Data Source Information page Figure 206 on page 300, 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 will be 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.

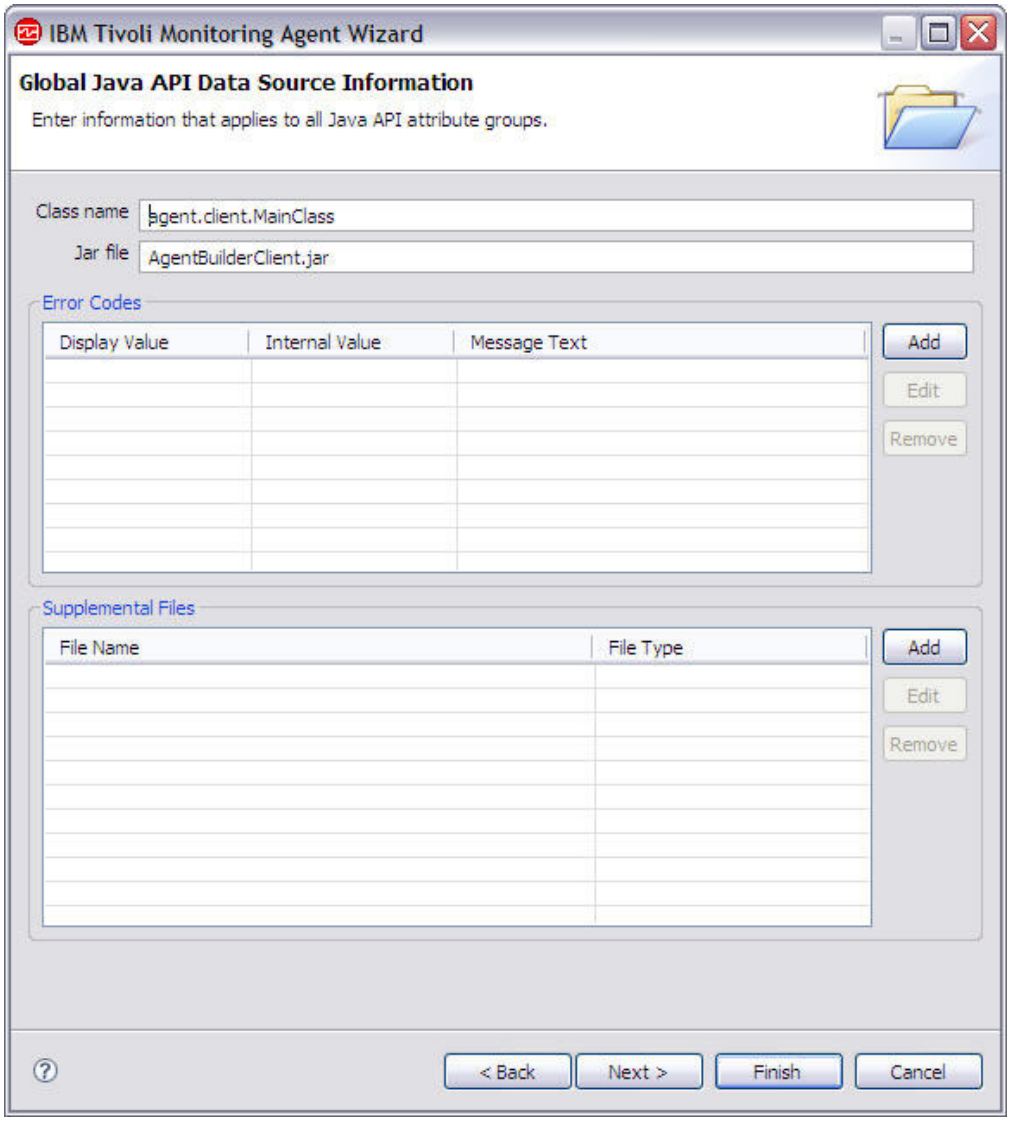


Figure 206. Global Java API Data Source Information page

13. *(Optional)* On the Global Java API Data Source Information page, **Error Codes** section, you can define the error codes that the Java application can send if it cannot collect data
    a. In the Error Codes section, click **Add**. An error code has a limit of 256 characters. Only ASCII letters, digits, and underscores are permitted. No spaces are permitted.
    b. In the Java API Error Code Definition window Figure 207 on page 301, enter a display value.

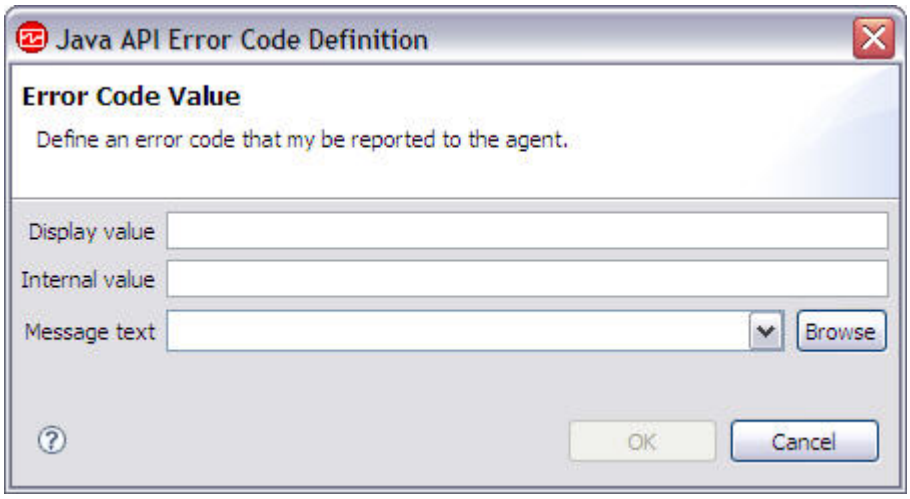


Figure 207. Java API Error Code Definition window,

c. Enter an internal value. The internal value must be an integer from 1,000 to 2,147,483,647.

d. You must define a message text for each error. You can use message text that was entered previously by selecting it from the drop-down list. Click **OK** to return to the Global Java API Data Source Information page. The message is logged in the agent log file.

   If no suitable message text is available, click **Browse** to set up the message text. The Messages (list) window is displayed (Figure 208). The message window lists messages that are defined in the agent. Until you have defined messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you have defined.

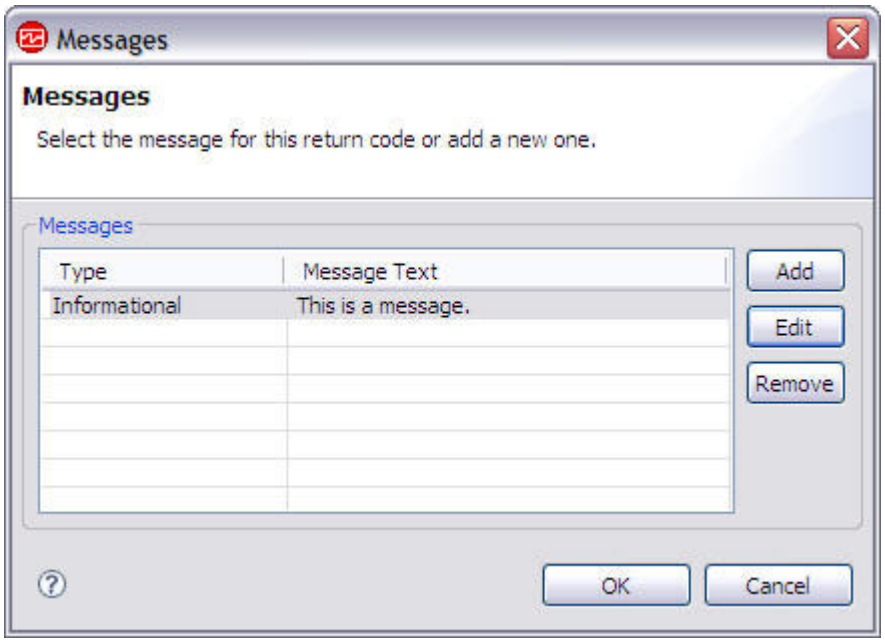


Figure 208. Messages window

e. In the Messages (list) window, click **Add** to see a Message Definition window (Figure 209), where you can type the text that describes the meaning of the new message and select the message type.

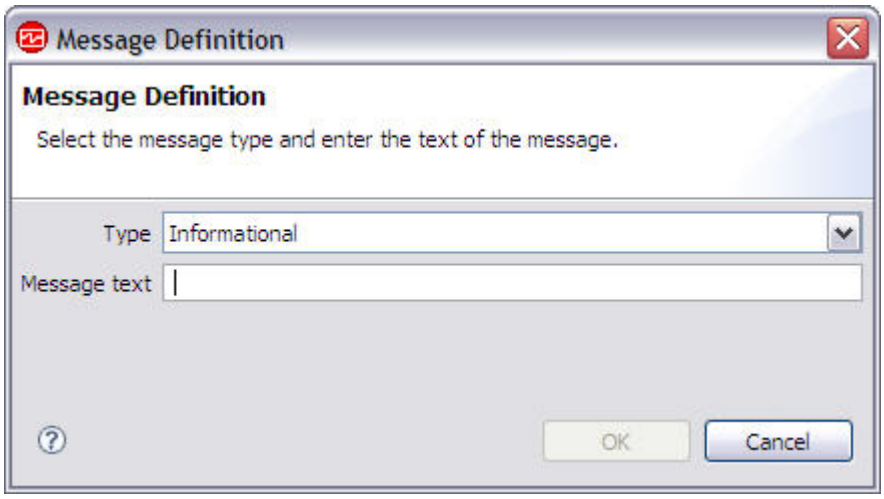


Figure 209. Message Definition window

Note: The message identifier is automatically generated for you.

f. Click **OK**.

g. 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 will be packaged with the agent and copied to the agent system when the agent is installed. The Java provider client API Jar file should not be listed here; it will automatically be 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 29. File types for supplemental files

File Type	Description
Executable	Select this option if you want to include an executable with the agent. The agent will not use this file, but it will be in the path for the Java application to use.
Library	Select this option if you to include a library with the agent. The agent will not use this file, but it will be in the library path for the Java application to use.
Java resource	Select this option to include Java resources with the agent. The agent will not use this file, but it will be in the classpath 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 382.

Click **Edit** to make changes to the imported file. For more information, see “Editing a command file definition” on page 222.

15. *(Optional)* If the data is sampled (you did not select "Produces events" on the Java API Information page), you can create a filter to limit the data returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see “Filtering attribute groups” on page 66
16. If you are adding this data source to a subnode, the Subnode Configuration Overrides page will be 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, since 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**.

       —OR—

    b. Click **Finish** to save the data source and open the Agent Editor.

    If this attribute group and the Java application are to run on operating systems different than the operating systems defined for the agent, select the correct operating systems on the Java API Settings page. To open this 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.

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 runtime is used to launch 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 314. 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 using API calls.

The Java application must be a long-running process. It should not terminate unless it receives a shutdown request from the API. If the Java application does terminate after it has registered with the agent, the agent will attempt to restart the Java application up to 3 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:

- Sun Microsystems 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 Java jar file containing the API used to communicate with the agent is automatically included with the agent runtime and included in the classpath of the Java virtual machine. 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 Java application's base classpath, along with the Java API Jar file.

Any Jar files that are not included with the agent, but are necessary for the runtime operation of the Java application, must be included in the "Classpath for external jars" configuration setting.

Generated sample Java application

When you create an agent with one or more Java API data sources, the Agent Builder generates Java application source code in the agent project that follows the structure of your agent. You will need to add your own Java code to the generated application to collect data for sampled attribute groups, handle events to be posted to event-based attribute groups, report errors if any problems are encountered, and perform tasks. The generated application does supply the agent with data, but it is sample data, to be replaced with data obtained from the resources you want to monitor.

This section describes the code that the Agent Builder generates and the code you need to add or replace for the resources you want to monitor.

In the discussion that follows, 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 210 on page 305:

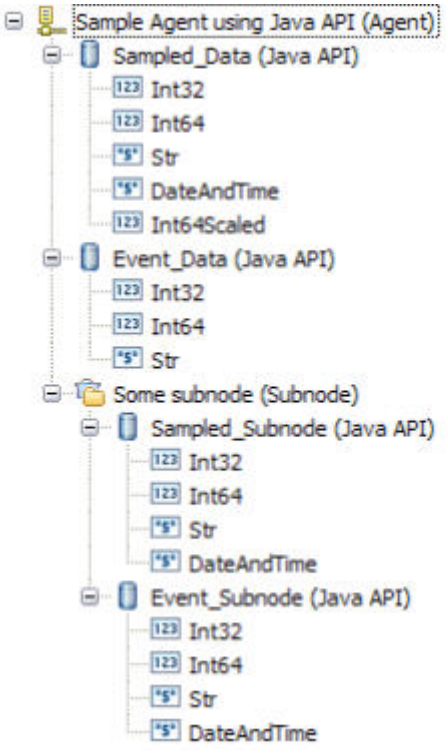


Figure 210. Sample agent structure

- *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 should modify, and files that you should not modify.

The following Java classes are created by the Agent Builder:

MainClass (agent.client package)
This is the class you specified on the Global Java API Data Source Information page. This class contains a main method and a method to handle take action requests. This class inherits from the helper class described next. You will need to modify this class to interface with resources you want to monitor and the actions you want to take.

MainClassBase (agent.client package)
This is a helper class which initializes the connection to the sever, 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 will need to 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)
These are helper classes, one for each Java API attribute group, which define the structure of the group's attributes in an internal class. Do not modify these classes.

ICustomAttributeGroup interface (agent.client.attributeGroups package)
This is an interface that defines public methods in each attribute group class. Do not modify this interface.

The classes which you can and should modify are never overwritten by the Agent Builder. The Agent Builder only creates them if they do not exist.

The helper classes and the interface are overwritten each time the Agent Builder is saved. As you make changes to the agent 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 reminding you not to modify the file.

Initialization and clean up

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 clean-up code should be added to MainClass. In the constructor, add initialization that is needed to create or access your resources. You may want to open connections to remote resources, create handles, or initialize data structures.

Before the agent terminates, the stopDataCollection method is called. If you need to close connections or perform any clean up 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's class. Similarly, if any clean up is needed only for a particular attribute group, that clean-up code can be added to the attribute group's stopDataCollection method.

When stopDataCollection is called, if you pass the clean-up work to another thread, be sure to wait for that thread to finish (or at least give it a reasonable amount of time to finish) before returning from the stopDataCollection method. Otherwise, the clean-up work could be abruptly terminated when the process ends because the main thread has completed.

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, and the attribute group helper objects create a protected reference to that logger in their constructors.) The logger object uses Java's trace logging utility. Errors and detailed trace information can be obtained from the trace log that is created by the logger. This is very important for troubleshooting problems with the provider.

One of the agent configuration settings is for the Java trace level. The following table shows the values you can set in the `JAVA_TRACE_LEVEL` configuration property, and if the API created the logger for you, the Level used by the logger.

Table 30. Java trace level options

Configured trace level	Java logging trace level	Description
Off	OFF	No logging is performed.
Error	SEVERE	Trace problems that have 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 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 will be lost. On UNIX and Linux systems, this data will be 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 attribute group's helper class defines an inner class called Attributes which has one field for each attribute defined in your attribute group. Derived attributes are not included since they are calculated by the agent. The field types are Java equivalents to the Tivoli Monitoring attribute types, as shown in Table 31.

Table 31. Attribute field types and their IBM Tivoli Monitoring attribute type equivalents

Tivoli Monitoring type	Attributes field type
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 needs to:

1. Collect the appropriate data from the resource 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 to 4 as necessary for each data row. (You may skip steps 1 to 4 altogether and return no rows. In this case the Error Code column of the Performance Object Status table will have a value of NO_INSTANCES_RETURNED. See ("Error codes" on page 311), for more information about error codes.)
6. Call AgentConnection.sendData to send the data to the agent, or call sendError to discard data copied from calls to setAttributeValues and send an error code instead.

You will need to collect the data from your resource (Step 1), replacing the sample data 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), or use the zero-argument constructor to create an Attributes object and then assign the fields of the attributes object to the attribute values you have collected. Fields have the same name as the attributes, though they start with a lower case 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), and you need to determine which resource to collect data from. There should 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 should 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 represented by the value in `K91_INSTANCE_KEY`.

These steps could be done in the collectData method, before the series of steps detailed in "Collecting sampled attribute group data" on page 307.

Alternatively, you may want to perform these steps in the attribute group class constructor (for example the `Sampled_Subnode` constructor) and establish a direct mapping from instance ID to resource. This would also give you the opportunity to create handles or open connections that could be used through the life of the agent, making 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 should remove the MonitoredEntity inner class, and replace the MonitoredEntity objects with objects that access your own resources. If you choose to perform the entire lookup procedure in the collectData method, you can remove the configurationLookup map from the class.

If you perform the steps above in the constructor, mapping 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 created in the constructor.
3. Perform the series of steps detailed in "Collecting sampled attribute group data" on page 307 to send data to the agent.

The Agent Builder chooses an arbitrary subnode property to use in its example, in this case `K91_INSTANCE_KEY`. If it chose the wrong property, or more than one property is needed to identify the correct resource, you should get the correct properties to find the desired resource.

Sending events

For attribute groups that generate events, there is no periodic call to a collectData method. 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 (the Event_Data class) creates and starts a thread which runs from an internal class named `SampleEventClass`. It periodically wakes up and sends an event. If you need 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 thread's run method, loop until the agent terminates.
3. Sleep for a period of time before checking for events. You may want to change the polling interval of 5000 milliseconds to a number that makes sense for your agent.
4. Determine if one or more events has occurred. The generated application does not check this, but always posts a single event.
5. For each event that needs to 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 could initialize that thread in the Event_Data constructor and register your own event handling object with your resource's event-handling mechanism. In your event handler, do the following:

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 would not need 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 needs to 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 this, the Java application performs 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 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 every subnode's attribute group. This is probably not the correct behavior for a production agent.

Take actions

Take actions are defined either in the Tivoli Enterprise Portal or by using the tacmd createaction command and 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 397).

The generated Java application registers for any actions that begin with the product code of the agent, for example, K91Refresh. This is done in the main helper class (MainClassBase) from the registerActionPrefix method. If you wish to register other prefixes, or not register for actions at all, override the registerActionPrefix in MainClass.

When the agent needs to run an action which starts with a prefix your agent has registered, the MainClass.takeAction method is called. You add code to call Request.getAction(), perform the appropriate action, and then call AgentConnection.sendActionReturnCode to send the return code from your action. A return code of 0 means the action succeeded, any other return code means the action failed.

Handling exceptions

The collectData and takeAction methods can throw any Java exception, so you may allow your collection code to throw exceptions without catching them. The handleException method (for collectData) or handleActionException method (for takeAction) will be called when the helper class gets the exception.

For collectData exceptions, you should call AgentConnection.sendError when an exception occurs or when there is any problem collecting data. The generated application passes an error code of GENERAL_ERROR, but you should replace this with an error code defined by your agent that best describes the problem that was encountered. For more information about adding error codes, see Step 13 on page 300.

For takeAction exceptions, you should 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 will not be called if a CpciException is thrown during the collecting of data as the helper class will handle the exception on its own.

Note: If you choose to catch your own exceptions inside the collectData method rather than making use of the handleException method, make sure any 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 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. Note that if you reply to a collect data request with a sendData call but you have included no data rows, you will see 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 32. Internal error codes for the agent

Error Code	Description
NO_ERROR	There are no problems with the attribute group at this time.
NO_INSTANCES_RETURNED	The Java application responded to a data collection request but provided no data. This is not an error. It generally indicates there are no instances of the resource being monitored by the attribute group.
OBJECT_NOT_FOUND	The agent tried to collect data for an attribute group that has not been registered through the client API. This may mean the application failed to start or had not initiated the attribute group registration when the agent tried to collect data.
OBJECT_CURRENTLY_UNAVAILABLE	The application sent the agent an error code that was not defined in the global list of error codes.

Table 32. Internal error codes for the agent (continued)

Error Code	Description
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 will require you to make corresponding changes to the Java application. If the structural changes are complex, or if you want to start over without customizations you have made, you can delete any or all of the Java source files before saving the agent.

The following table describes required modifications to the Java application source files after certain agent changes are made in the Agent Builder when the agent is saved.

Table 33. Changes to an agent requiring modifications to the Java source

Agent change	What the Agent Builder does	Manual changes 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	Migrate 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	Migrate 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 migrate customized logic to some other class Remove the attribute group helper class

Table 33. Changes to an agent requiring modifications to the Java source (continued)

Agent change	What the Agent Builder does	Manual changes needed in the Java source
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	Migrate 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

Note that some of the changes mentioned above can be streamlined if you use the Eclipse Refactor - Rename action on all the affected names (including helper class names) before saving the changed agent.

Using 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 will be to modify a parameter of an existing method call, such as changing a posted error code from GENERAL_ERROR to an error code defined in your agent.

If you need to do more extensive coding with the Java API, you can view Javadoc from the Eclipse text editor while editing the Java code by doing the following:

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 is automatically added to the project's Java Build Path when an agent which contains a Java API attribute group is created or imported, or 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

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 34. 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	Yes	Use this property to specify the trace level used by the Java providers.
JVM Arguments	String	No	Use this property to specify an optional list of arguments to the Java virtual machine.
Classpath 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.

Testing

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

To start the Testing procedure , select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see “Tivoli Monitoring Agent Editor” on page 37

After you click **Test** in one of the previous two steps, the Test Java Client window Figure 211 on page 315 is displayed.

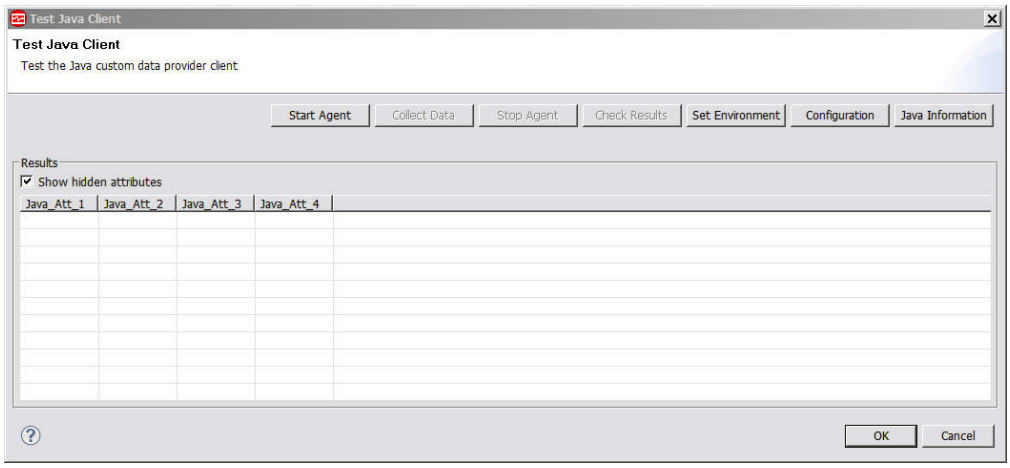


Figure 211. Test Java Client window

To start and test the data source use the following procedure

1. *(Optional)* Before starting your test you can set environment variables, configuration properties and Java information. For more information, see "Configuration prior to testing" on page 374. For more about default Java runtime configuration properties, see "Java API configuration" on page 314.
2. Click **Start Agent**. A window indicates that the Agent is starting.
3. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent monitors the Java Client for data.
4. The Test Java Client window displays any data that is returned.
5. Optionally, for example if something does not seem to be working as expected, you can click **Check Results**. The Data Collection Status window opens and shows you more information about the data. The data collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534
6. The agent can be stopped by clicking **Stop Agent**
7. Click **OK** to close the Test window.

For a general discussion about Agent Testing, see Chapter 31, "Testing and debugging your agent," on page 373

Chapter 27. Creating attribute groups from existing sources

When at least one attribute group exists the option to create an attribute group using an existing attribute group or groups is available to you on the Agent editor Data Source Location page (Figure 212 on page 318). You can create an attribute group using existing data sources in the following ways:

1. Joining data from two existing attribute groups
2. Filtering data from an existing attribute group

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.

Use the following procedure to create an attribute group using existing data sources:

1. On the Agent Initial Data Source page or the Data Source Location page (Figure 212 on page 318), click **Existing data sources** in the **Monitoring Data Categories** area.

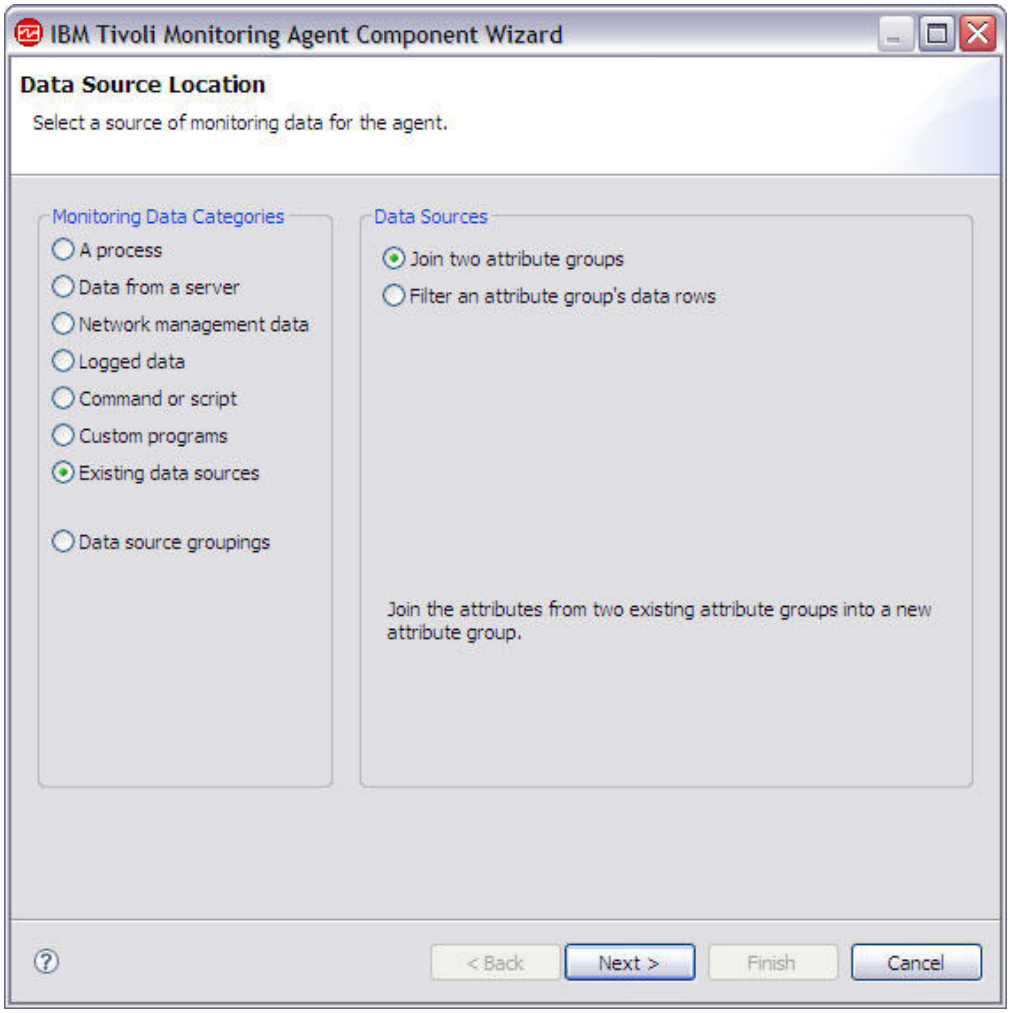


Figure 212. Adding existing data sources

2. In the **Data Sources** area, select one of:
   a. **Join two attribute groups**, click **Next**, and see about "Joining two attribute groups"
   b. **Filter an attribute group's data rows**, click **Next** and see about "Filtered Attribute Groups" on page 323

Joining two attribute groups

With Agent Builder, you can construct an attribute group from two other attribute groups. This is most useful when the agent collects data from two different types of data sources (WMI and PerfMon, for example, or SNMP and script) where each set of attributes is more useful when used together in one Tivoli Enterprise Portal view.

For example, 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 very useful to you by itself, but if you can combine them both using the index to match the appropriate rows from each attribute group, you have a more useful attribute group that you can use to display the name, type, and metrics together.

This same mechanism can be used to add tags to information collected through normal attribute groups so that it can be more easily correlated in an event system when a problem is detected. For example, a company wants to manage all of its servers by collecting common data and using common situations to monitor the health of the servers. It also wants to be able to identify the servers with additional information that tells it what application is running on a particular server. The company wants to have control of the values used on each server, but it does not want to create different agents for each application. It can accomplish this by creating an additional attribute group in its single agent as follows:

```
Application_Information
 application_type  integer
 application_name string
 application_group string
```

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, and 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 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 containing all the attributes from both source attribute groups.

Table 35. source attribute group one (single row)

Attribute1	**Attribute2**	**Attribute3**
16	some text	35

Table 36. source attribute group two (single row)

Attribute4	**Attribute5**	**Attribute6**	**Attribute7**
5001	more data	56	35

Table 37. Resulting join

Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**	**Attribute6**	**Attribute7**
16	some text	35	5001	more data	56	35

If one source attribute group is defined to return only one row (a single-row attribute group) while the other can return more than one row (a multi-row attribute group), then 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 38. source attribute group one (single row)

Attribute1	Attribute2	Attribute3
16	some text	35

Table 39. 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 40. 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, if 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 perform the join. The resulting attribute group contains rows of data where the value for the attribute in the first attribute group matches the value of the attribute from the second attribute group.

Table 41. source attribute group one (more than one row)

Attribute1	Attribute2	Attribute3
16	some text	35
27	more text	54
39	another string	66

Table 42. 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 43. 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. See "Availability node" on page 529 for more information about the data contained in the Availability attribute group.

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**. 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.

A window is displayed so you can identify the two attribute groups to join. The window also displays the cardinality of the selected attribute groups. If both attribute groups selected are defined as multiple data row, then you must select an attribute in each attribute group that identifies the column on which to join. When you confirm the choices, a new attribute group is displayed in the data sources tree.

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 only join an event attribute group to a single row non-event attribute group.

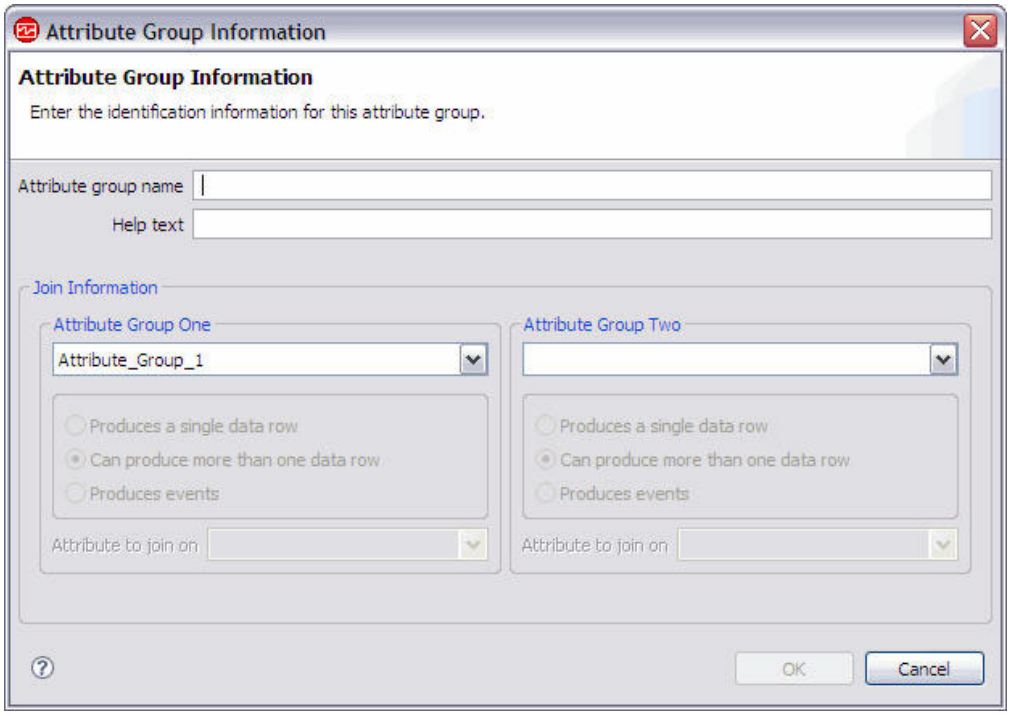


Figure 213. Attribute Group Information window

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.

  —OR—

- The attribute is used in any derived attribute in the joined attribute group.

Joined attributes cannot be deleted. Only derived attributes, if any have been added, can be deleted from the joined attribute group.

Adding an attribute

New joined attributes cannot be explicitly added. Only derived attributes can be explicitly created.

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 has been 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. See ("Committing a version of the agent" on page 49) for more information about how to commit agents.

Removing availability filters

The last availability filter cannot be removed if the Availability attribute group is referenced in a joined attribute group.

Joined attributes

The attribute name and help text of the joined attribute can be changed so that they are different from the attribute name and help texts for the source attribute.

The joined attribute can be displayed or not displayed on the Tivoli Enterprise Portal by selecting or clearing the **Display attribute in the Tivoli Enterprise Portal** check box. This choice is irrespective of whether the source attribute is displayed on the Tivoli Enterprise Portal.

Any attribute or combination of attributes (that are displayed on the Tivoli Enterprise Portal) can be marked as key attributes by selecting the **Key attribute** check box. This is independent of whether the attributes are key attributes in the source attribute groups, or even whether the source attributes are displayed on the Tivoli Enterprise Portal.

Attribute type information for joined attributes is taken from the source attributes and cannot be changed in the joined attribute. In the Join Attribute Information section of editor (Figure 214 on page 323), click **Locate source attribute** to navigate to the source attribute.

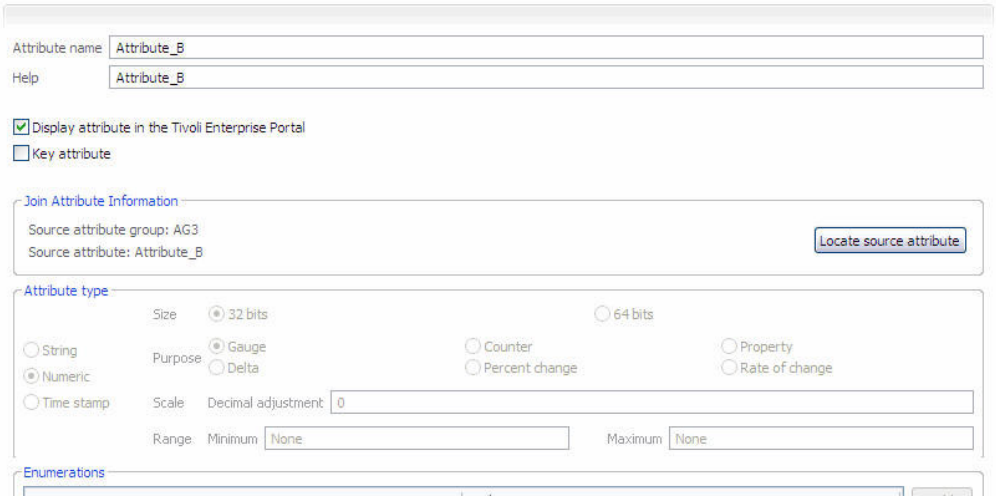


Figure 214. Locating source attribute information

Any changes to the source attribute groups are reflected in the displayed joined attributes. If the source attribute groups change or a different attribute group is set as the source attribute group, those attributes are automatically updated under the joined 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, when the name or help text are manually changed in a joined attribute, changes to the source attribute name or help text are no longer copied to the joined attribute.

Filtered Attribute Groups

A Filtered attribute group filters rows of data from an existing attribute group to create the filtered attribute group data rows. A filtered attribute group is most useful when a base data source (for example SNMP, WMI, Perfmon) query, returns data that you would prefer is divided into separate groups. For example, 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 the table down into memory and disk as separate tables. You can break down the table by creating two base attribute groups that collect the same data and filter out the rows you do not want in each case. 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 you 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 data returned is:

```
Name    Type  Size   Used  Free
Memory  MEM      8      4     4
```

and where `Type=="DISK"`, the data returned 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.

Creating a Filtered Attribute Group

Use the following steps to create a filtered Attribute Group

1. Select **Filter an attribute group's data rows** on the Agent Editor Data Source Location page
(Figure 215),

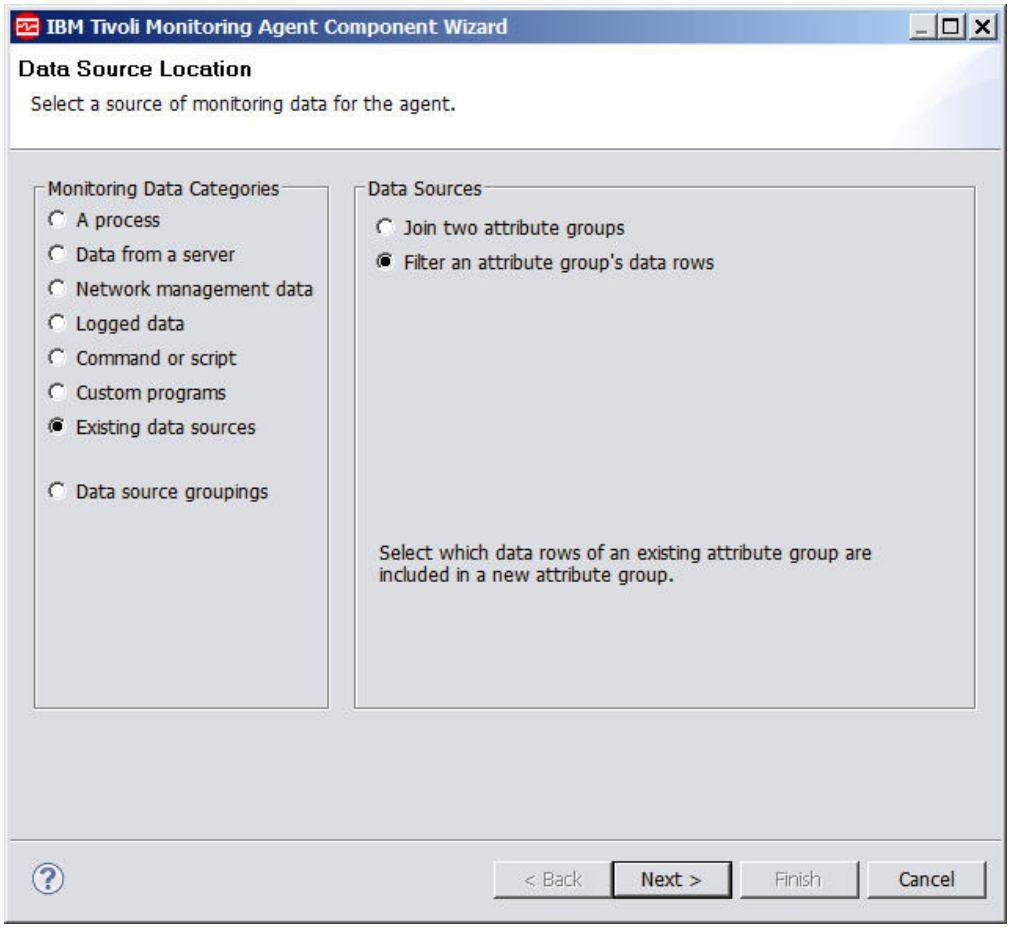


Figure 215. Adding existing data sources

2. Click **Next** and the Filter Information page Figure 216 on page 325 is displayed.

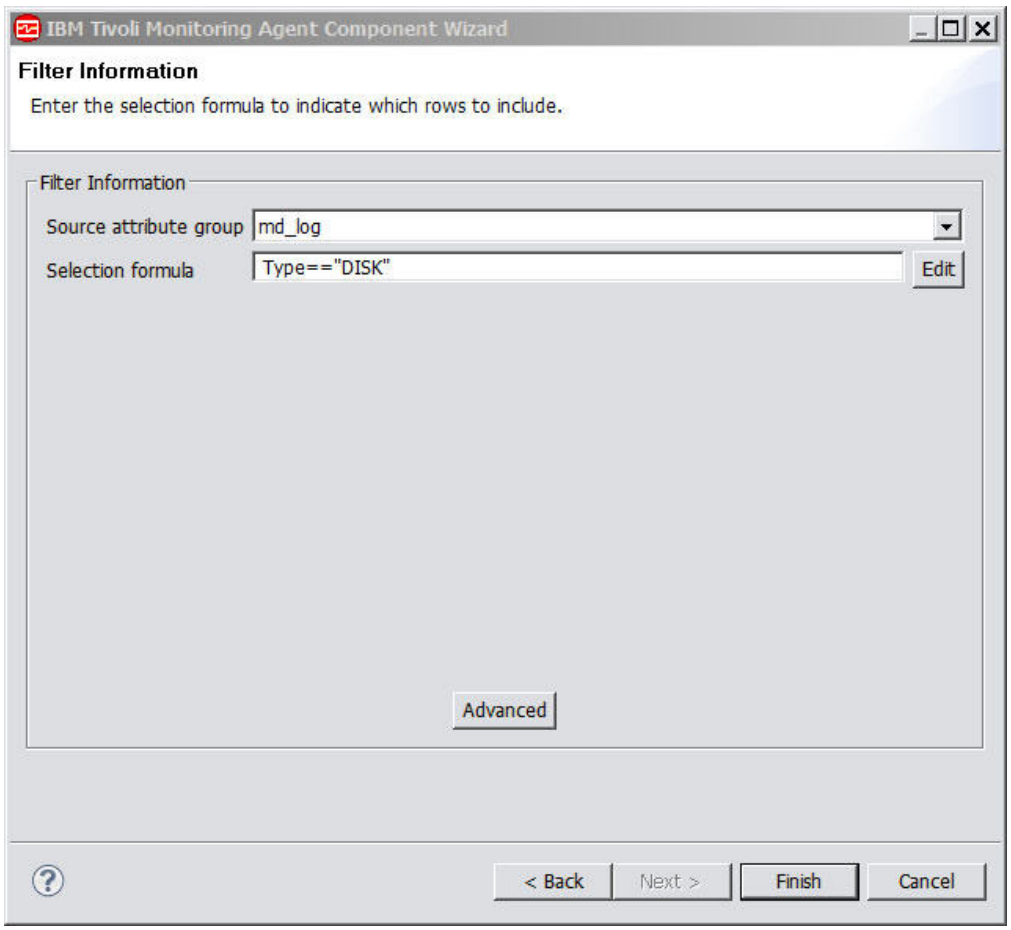


Figure 216. Filter Information window

3. Select a **Source attribute group** from the drop-down list.
4. Enter a **Selection formula** to filter the data from the attribute group you selected. For example in the Filter Information page Figure 216 shown, the selection formula filters data rows where the **Type** attribute is equal to "DISK". Data rows whose Type attribute do not match "DISK" are discarded. The selection formula you enter must evaluate to a Boolean result, true, or false.

   Note: In the Filter Information page Figure 216, you can click **Edit** to enter or modify the formula using the Formula Editor. For more information about the Formula Editor, see “Formula Editor” on page 66

Chapter 28. Creating a navigator group

This chapter provides information about how to create navigator groups. See "Navigator groups" on page 31 for an explanation of how navigator groups are used.

You can create a navigator group while creating an agent using the IBM Tivoli Monitoring Agent Wizard at the base agent level, or while defining a subnode. Use the following steps to organize the data sources for your agent:

1. On the Initial Agent Data Source page or the Data Source location page, click **Data source groupings** in the **Monitoring Data Categories** area.

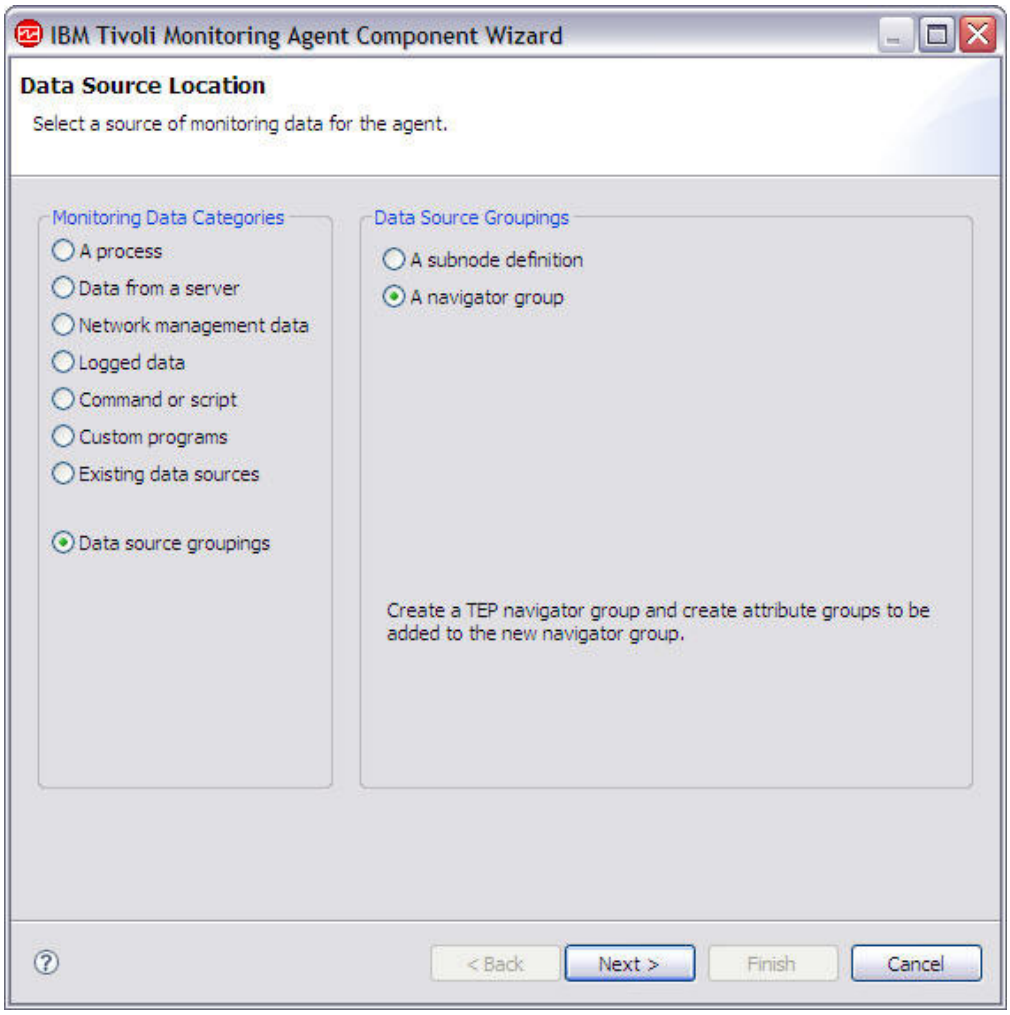


Figure 217. Selecting the initial source of monitoring data for a new agent

2. In the **Data Sources** area, click **A navigator group**.
3. Click **Next**.
4. On the Navigator Group Information page (Figure 218 on page 328), 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

.

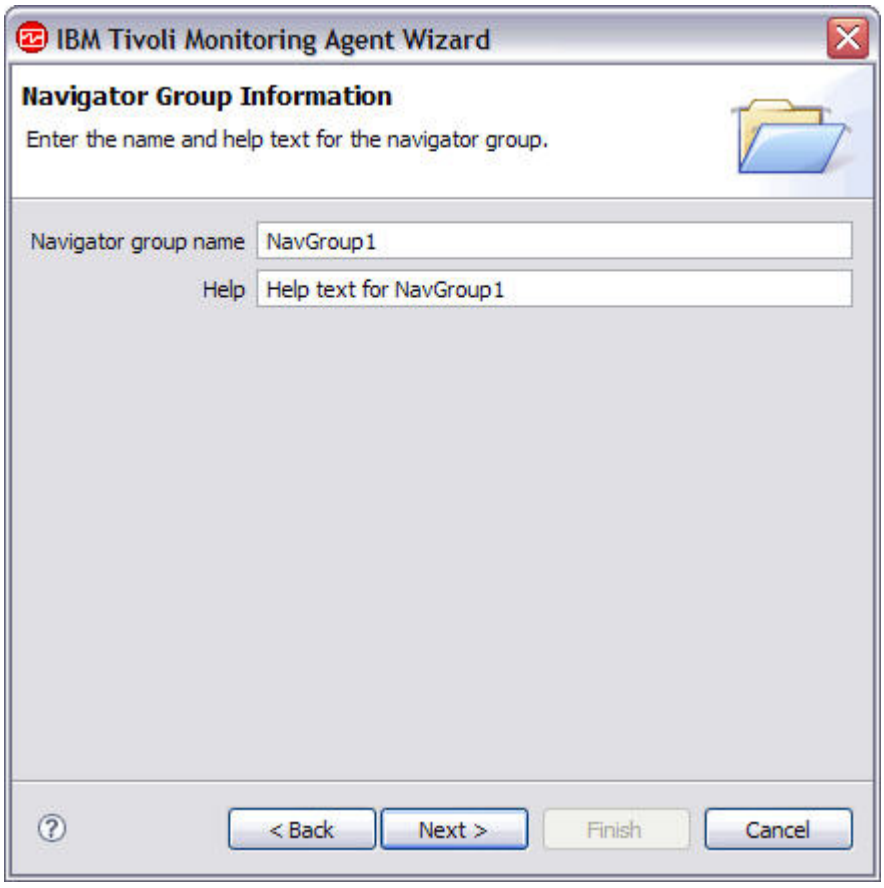


Figure 218. Providing the navigator group name and help text

5. On the First Navigator Group Data Source page (Figure 219 on page 329), select the first source of monitoring data for the new navigator group by clicking a category in the **Monitoring Data Categories** list and a data source in the **Data Sources** list. Then, click **Next**.

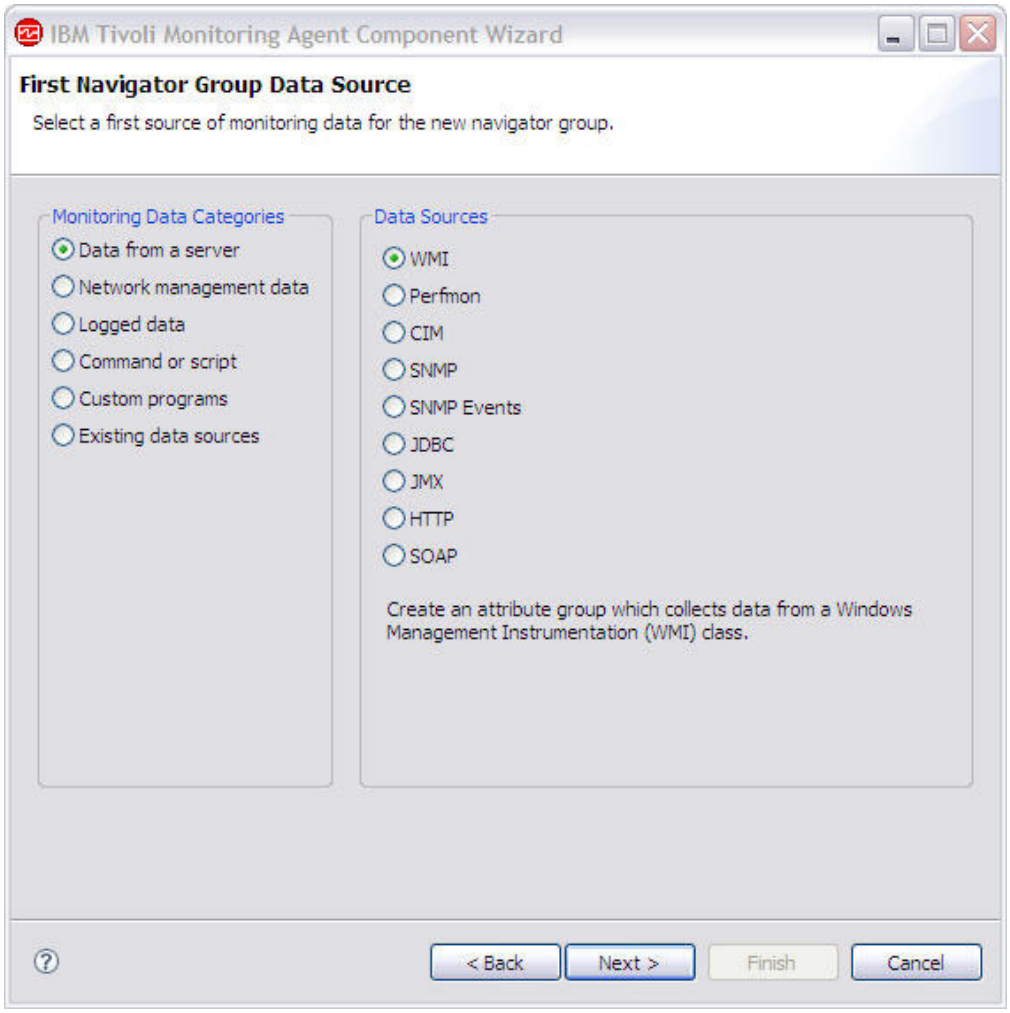


Figure 219. Selecting the first source of monitoring data

Note: The data source can be created as usual or you can click **Existing data sources** and chose to move one or more data sources that you have already created into the navigator group.

6. If you want to create a new 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 top level of the Data Sources tree

     —OR—
   - 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**.

     —OR—
   - Select a subnode and click **Add to Selected**.

Chapter 29. Creating subnodes

This chapter provides information about how to create subnodes. See "Subnodes" on page 31 for an explanation of how subnodes are used.

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 the same type of monitored entity, 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 220 shows the monitoring of different systems.

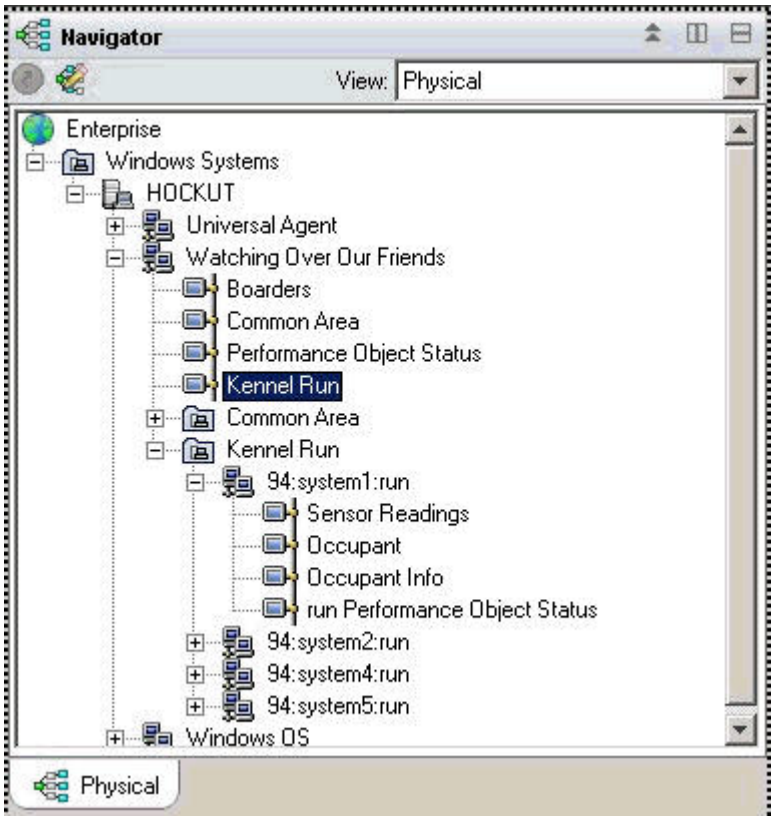


Figure 220. Subnodes monitoring different systems

Monitoring multiple types of information

When one agent monitors multiple types of monitored entities, each type of entity is displayed in a separate subnode in the Tivoli Enterprise Portal physical Navigation tree. Each subnode includes the information 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:

- Common Area
- Kennel Run

The agent in Figure 221 has one copy of each subnode type running. A particular agent might create any subset of the defined agents. Subnodes can be used to mimic Tivoli Monitoring V5 profiles.

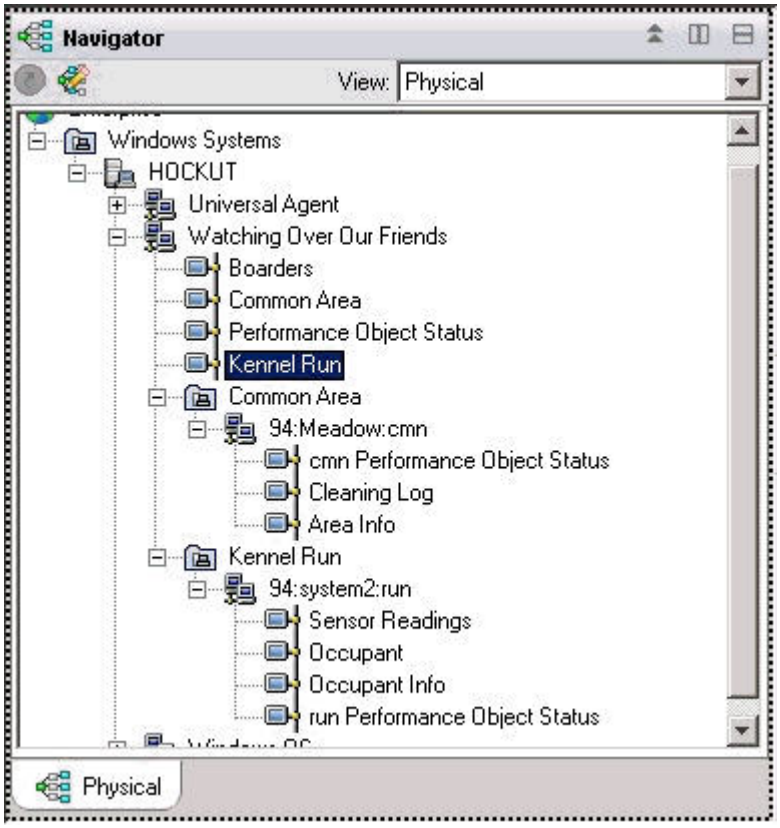


Figure 221. Subnode types in Navigator tree

Both ways of using subnodes (different subnodes of the same type and subnodes of different types) can be used in the same agent, where each type can have more than one subnode instance.

Figure 221 shows two types of subnodes that monitor two types of entities: Common Areas and Kennel Runs. In addition, there are several subnodes 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 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 48 on page 495.

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 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 and 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 222 on page 334, four subnodes are running to collect data.

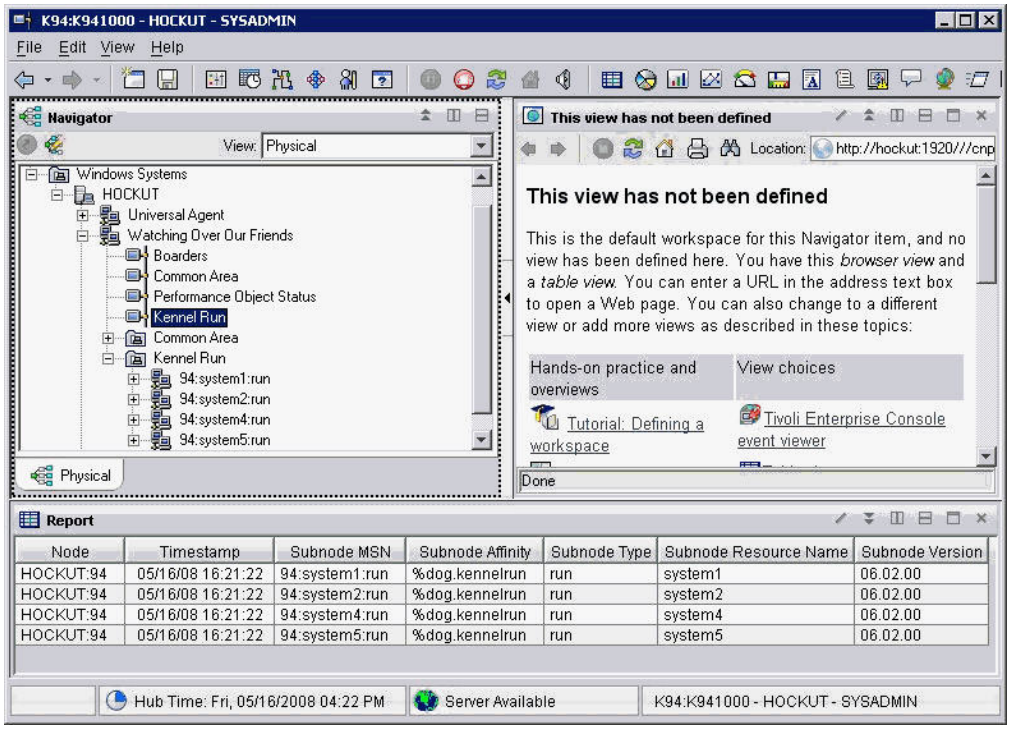


Figure 222. Monitoring multiple subnode instances of the same subnode type

The Performance Object Status subnode contains data that is visible in the Navigator tree and can have situations that monitor the status of the other data collections.

The example in Figure 223 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 defined in the subnode, there is one row in the table.

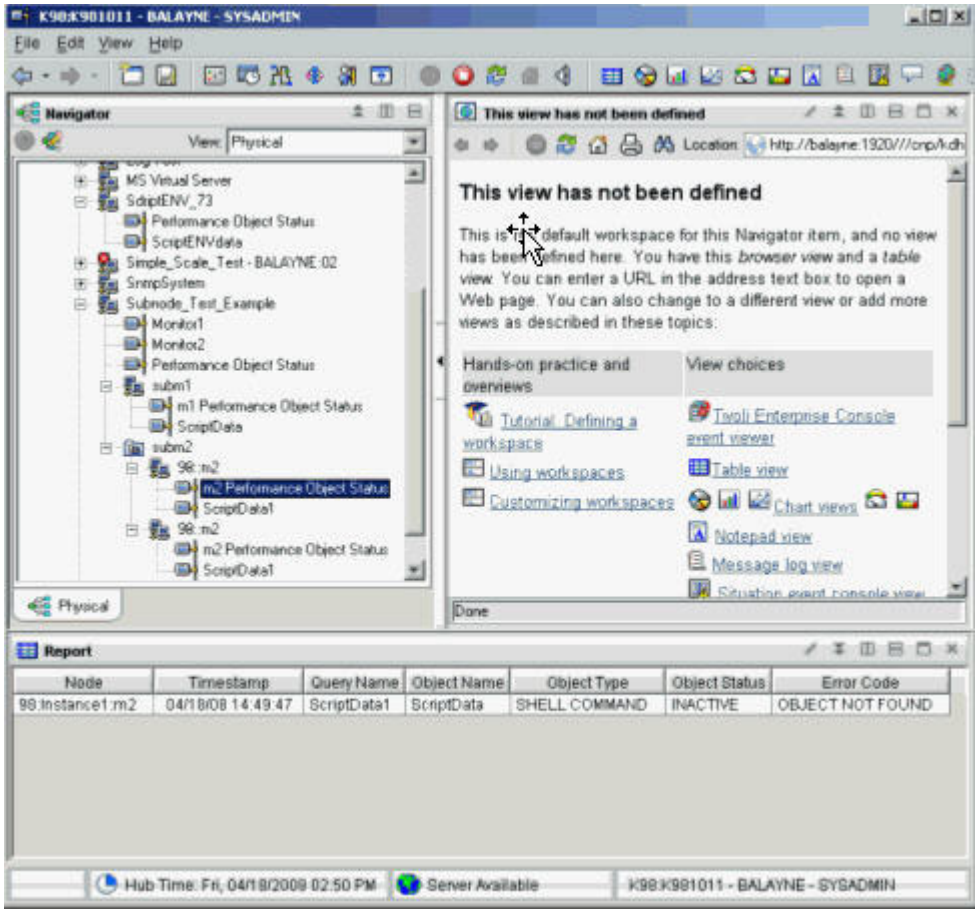


Figure 223. Example: data collection in a subnode

Creating subnodes when creating an agent

You can begin creating a subnode from one of the following windows:

- Data Source Location page in the New Agent Wizard (as the first component of an agent)
- Data Sources tree in the New Agent Wizard
- Agent Editor

Creating a subnode as the first component of an agent from a Data Source Location page in the New Agent Wizard

The following procedure describes how to create a subnode when creating an agent in the New Agent Wizard.

1. On the Data Source Location page (Figure 224 on page 336), click **Data source groupings** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **A Subnode Definition**.

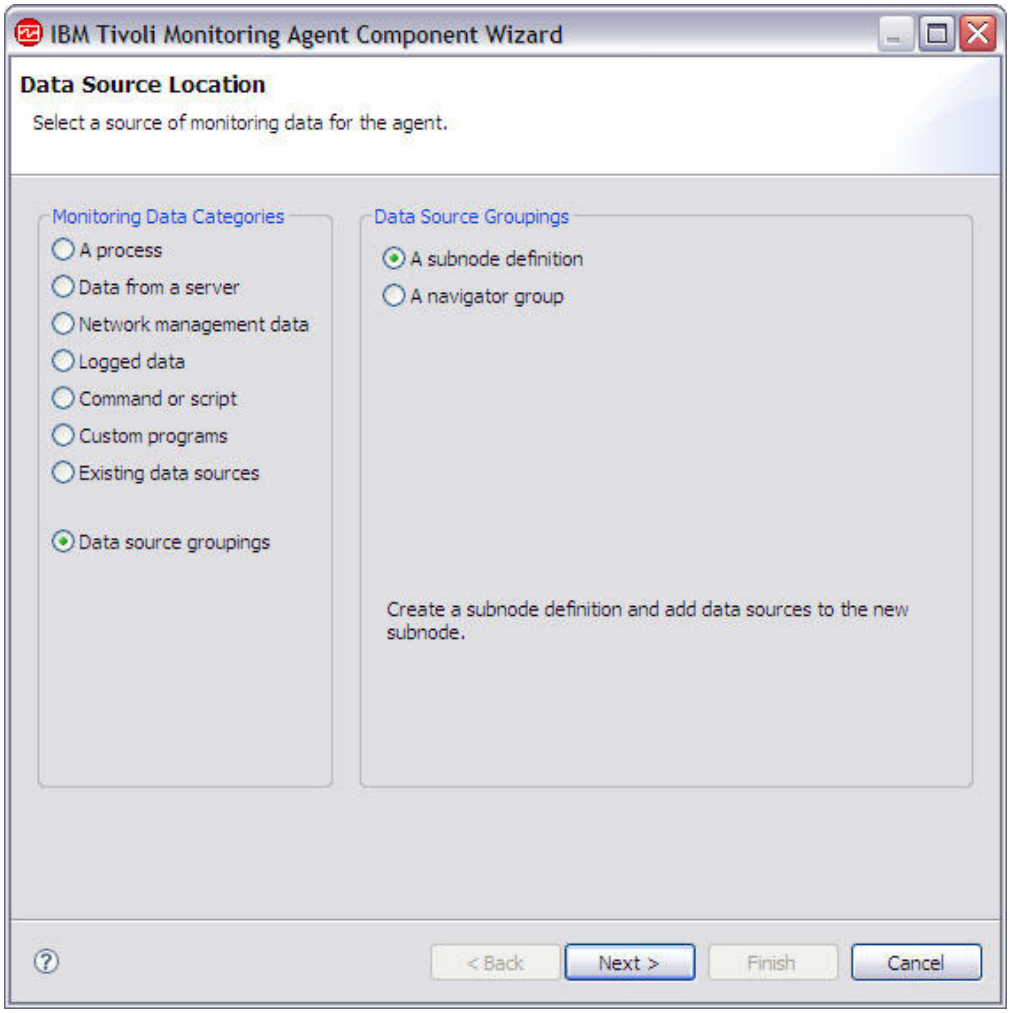


Figure 224. Data Source Location page - Creating a subnode

3. Click **Next**.
4. Complete the Subnode Information page (Figure 225 on page 337) 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, type 1-3 characters (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 529.

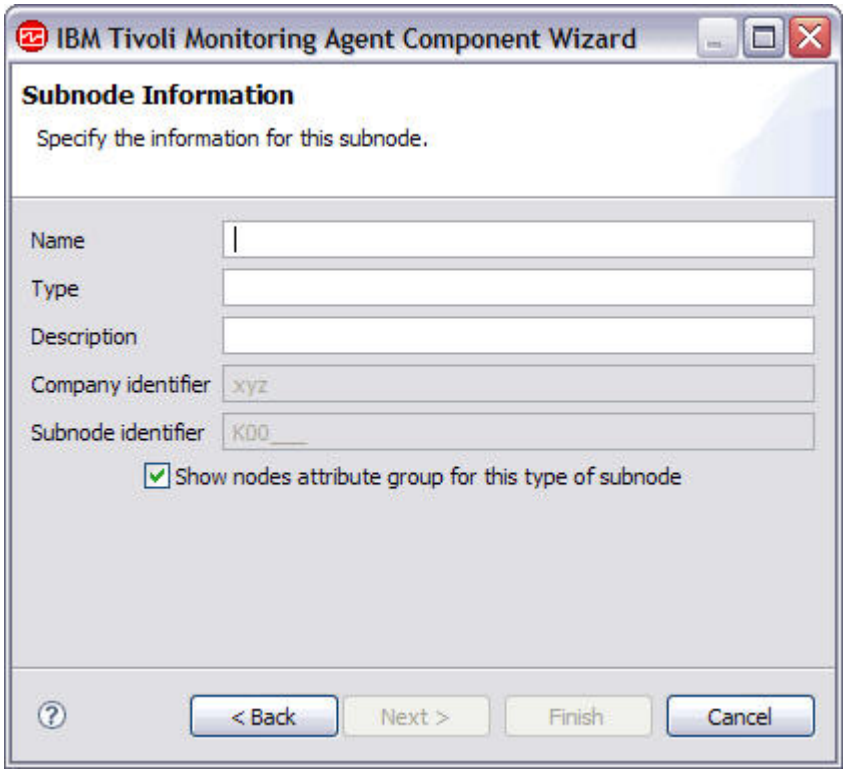


Figure 225. Subnode Information page

e. Click **Next**.

5. Complete the Initial Subnode Data Source page (Figure 226 on page 338) 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**.

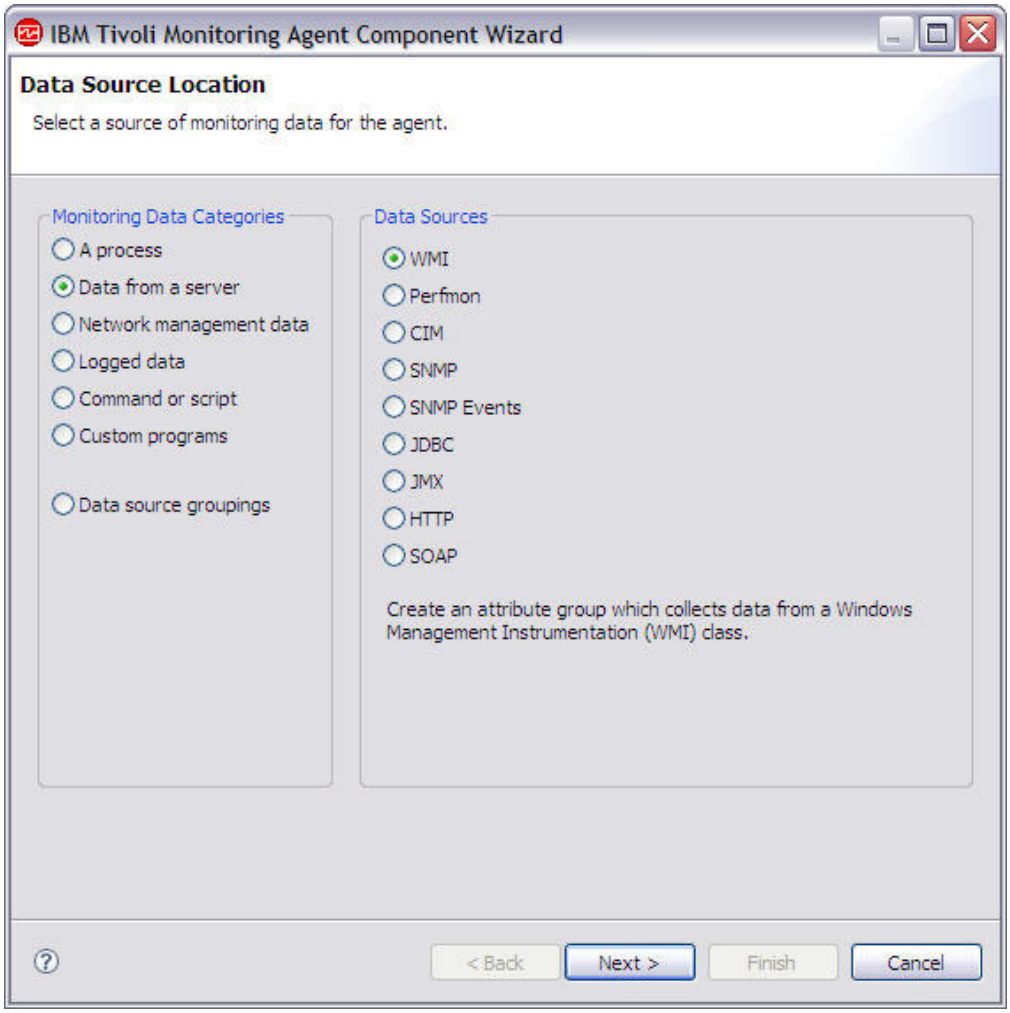


Figure 226. Initial Subnode Data Source page

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 you selected requires configuration, you can use the Subnode Configuration Overrides page to choose the configuration properties. In the Subnode Configuration Overrides page (Figure 227 on page 339), choose the configuration properties that you want for the subnode at the agent level, and the ones 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.

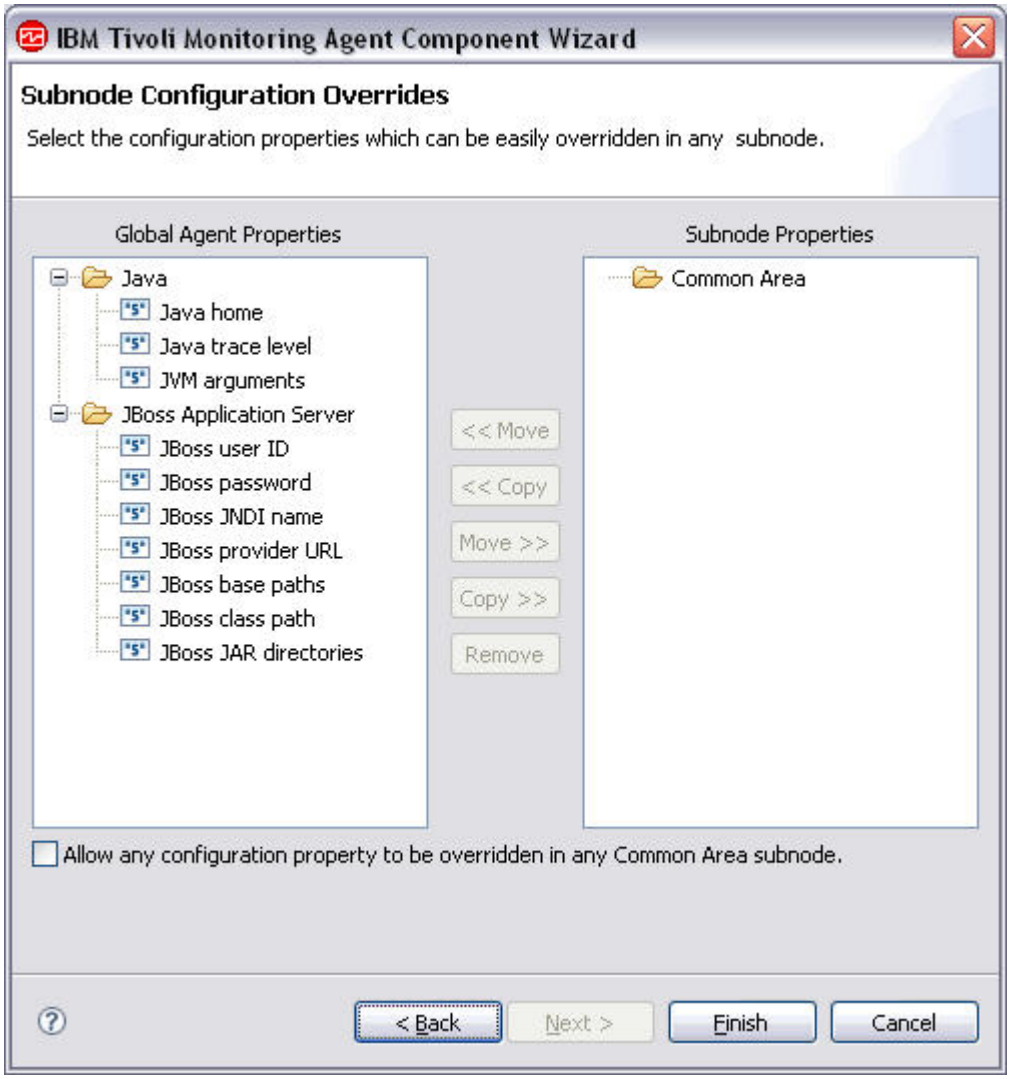


Figure 227. Subnode Configuration Overrides page

8. Click **Next**.
   The Data Source Definition page is displayed.

Creating a subnode from the Data Sources tree in the New Agent Wizard

1. On the Data Source Definition page (Figure 228 on page 340), right-click an agent name to create a new subnode definition from the **Data Sources** tree and click **New Subnode Definition**.

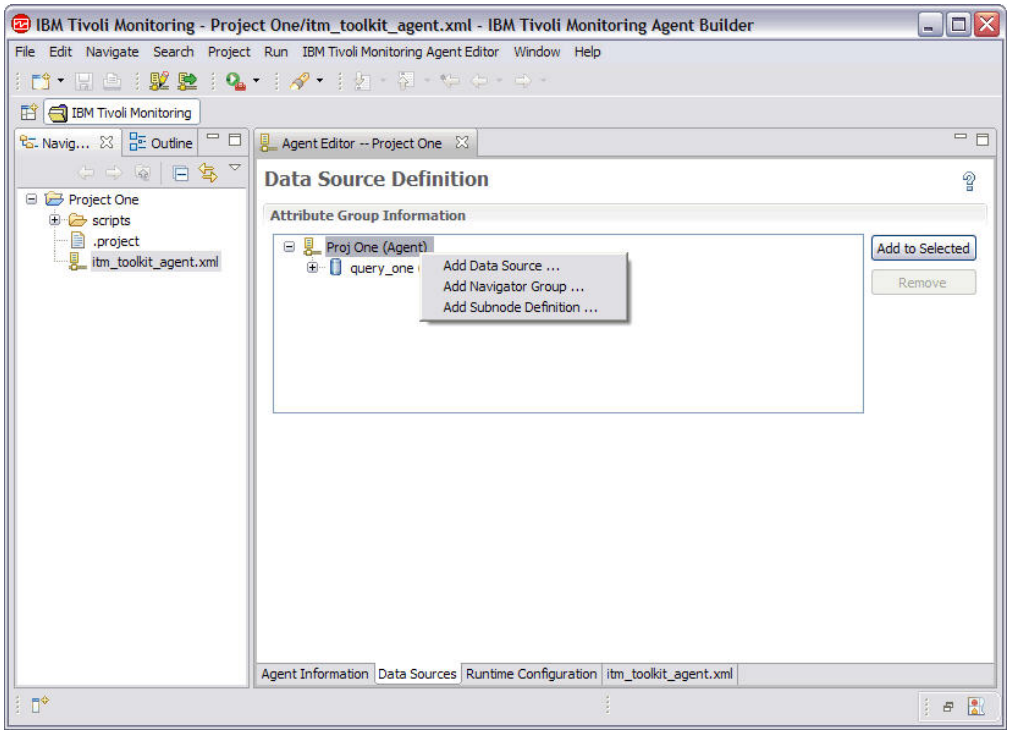


Figure 228. Data Source Definition page

2. Enter information about the new subnode definition to create a new subnode as shown in Figure 229 on page 341, and then click **Next**.

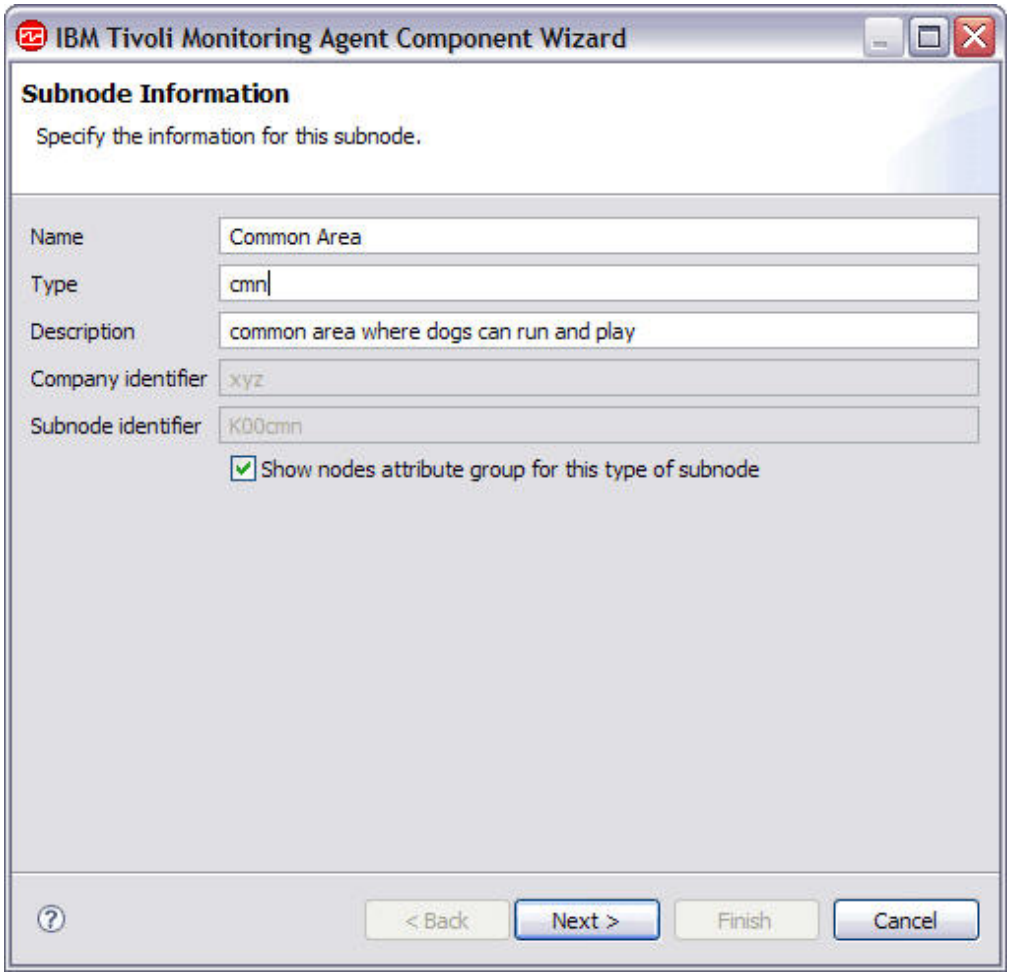


Figure 229. Defining a new subnode

3. On the Initial Subnode Data Source page (Figure 230 on page 342), select a type of data source for the new subnode, and then click **Next**.

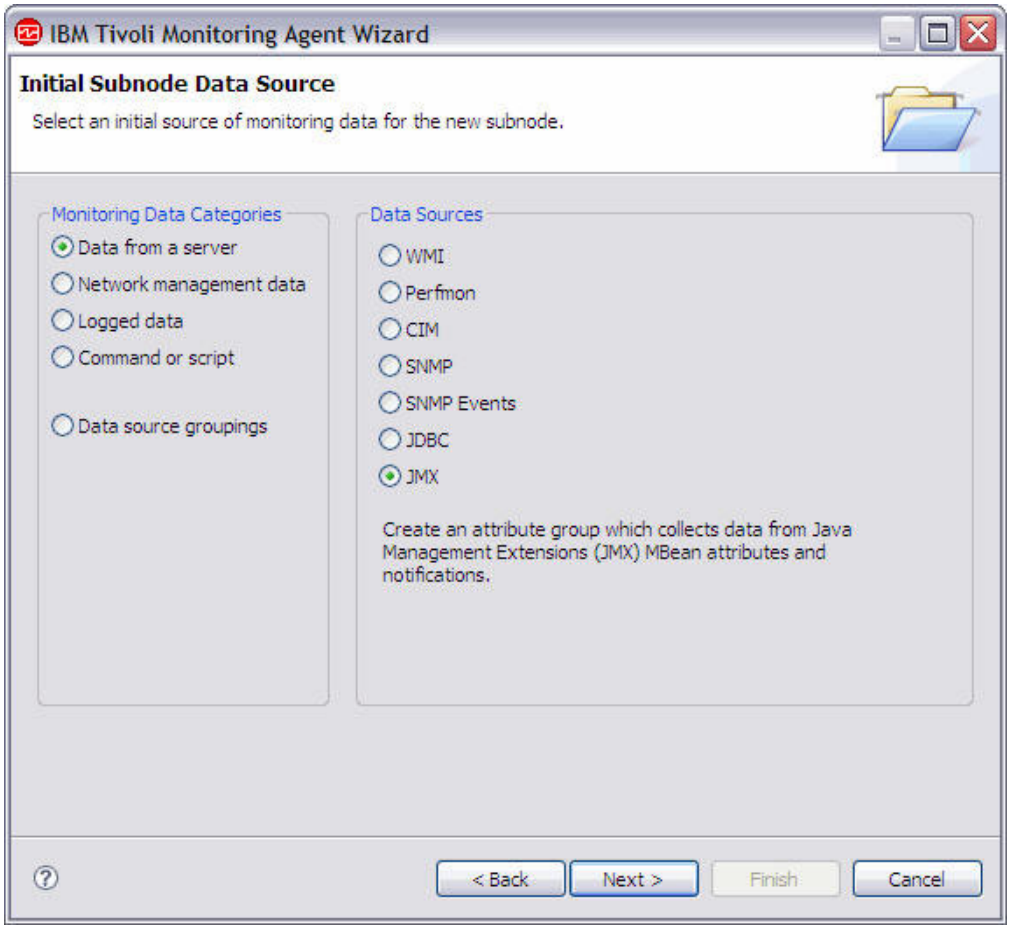


Figure 230. Initial Subnode Data Source page

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 completing the final data source-specific page.
5. On the Subnode Configuration Overrides page (Figure 231 on page 343), 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, in which case the subnode property, if it is set, 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.

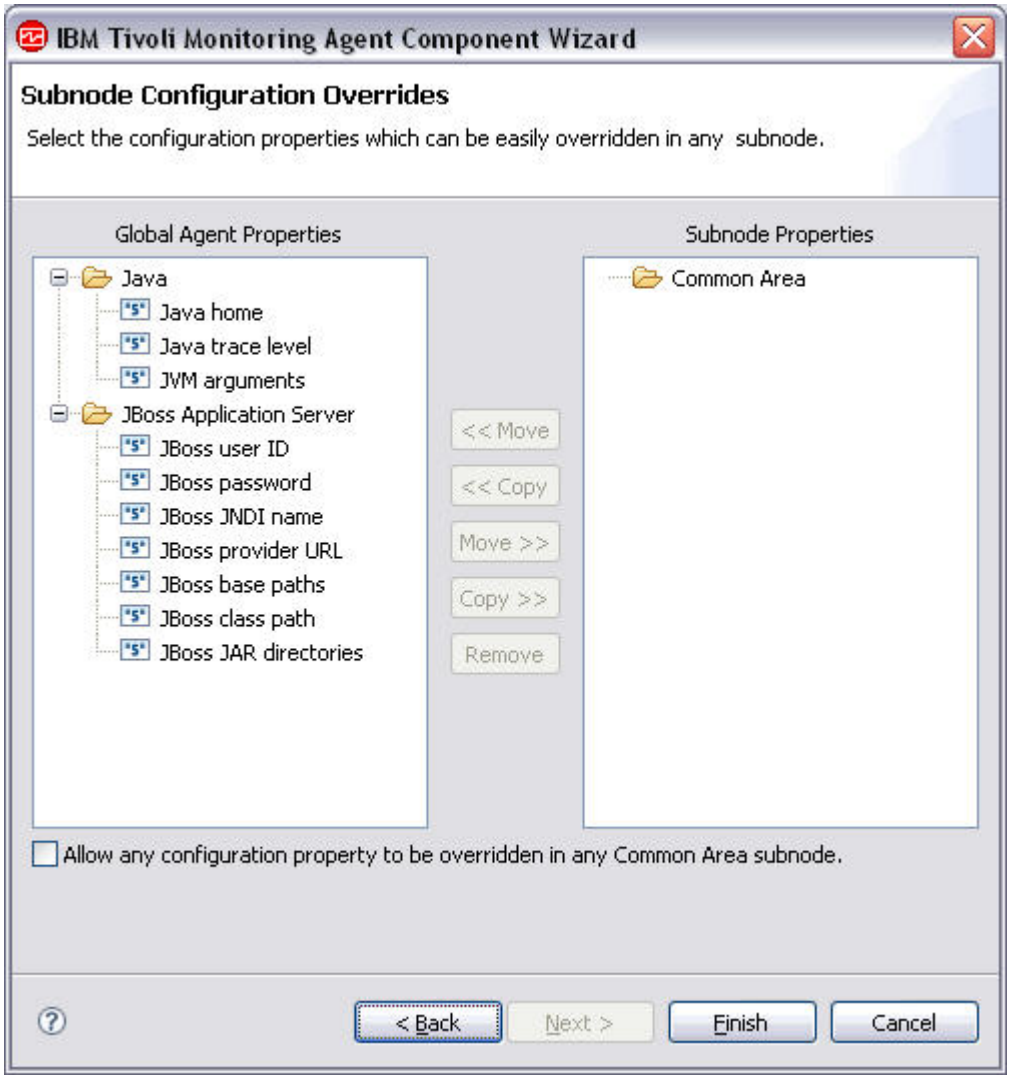


Figure 231. Subnode Configuration Overrides page

6. When finished identifying the properties, click **Finish**. The Data Source Definition widow is displayed.

Creating a subnode in the Agent Editor

1. Start the Agent Editor (Figure 232 on page 344) 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 context menu.

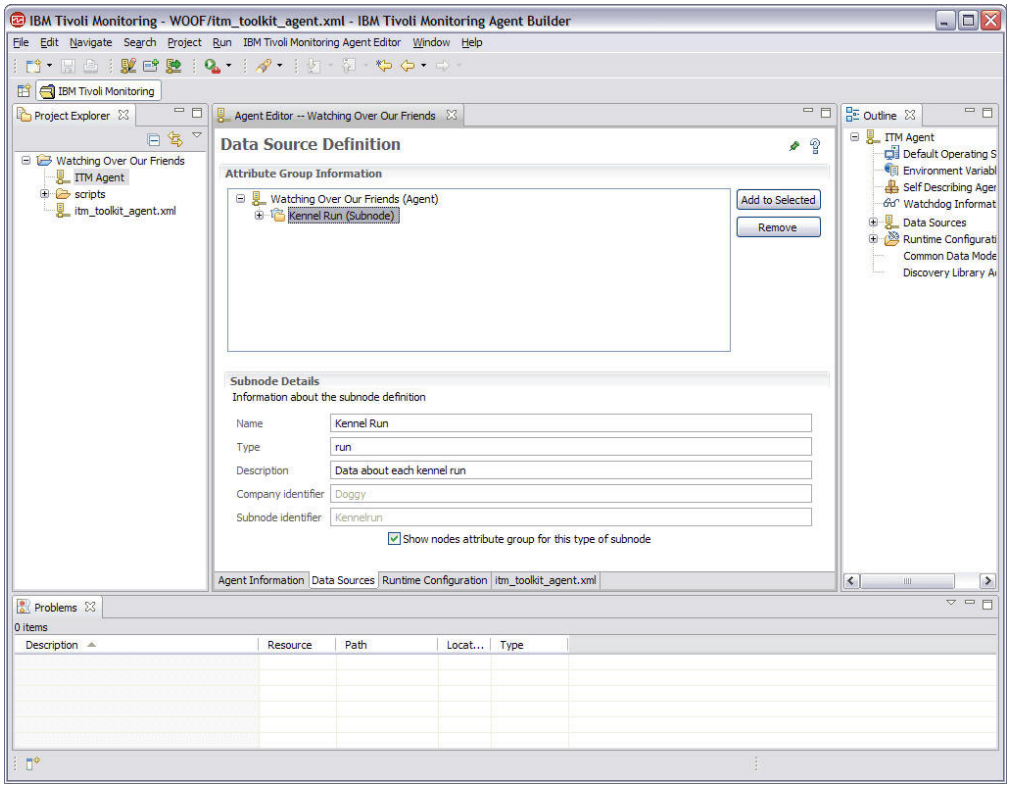


Figure 232. Agent Editor Data Source Definition page

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 is different 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 once during runtime configuration. They do not form subsections and cannot be duplicated or removed.

  See “Configuring a subnode from the command line” on page 350 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 this happens, 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 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 “Configuring a subnode from the command line” on page 350 for GUI and command line examples of configuring subnodes.

Configuring a subnode

Use the steps in “Creating subnodes when creating an agent” on page 335 to create a subnode. Then, use the steps in this section to configure the subnode.

When adding a data source to a subnode, the Subnode Configuration Overrides page is presented if the data source requires configuration, such as the one in Figure 233. It shows custom configuration properties and any other configuration properties that are applicable to the subnode type.

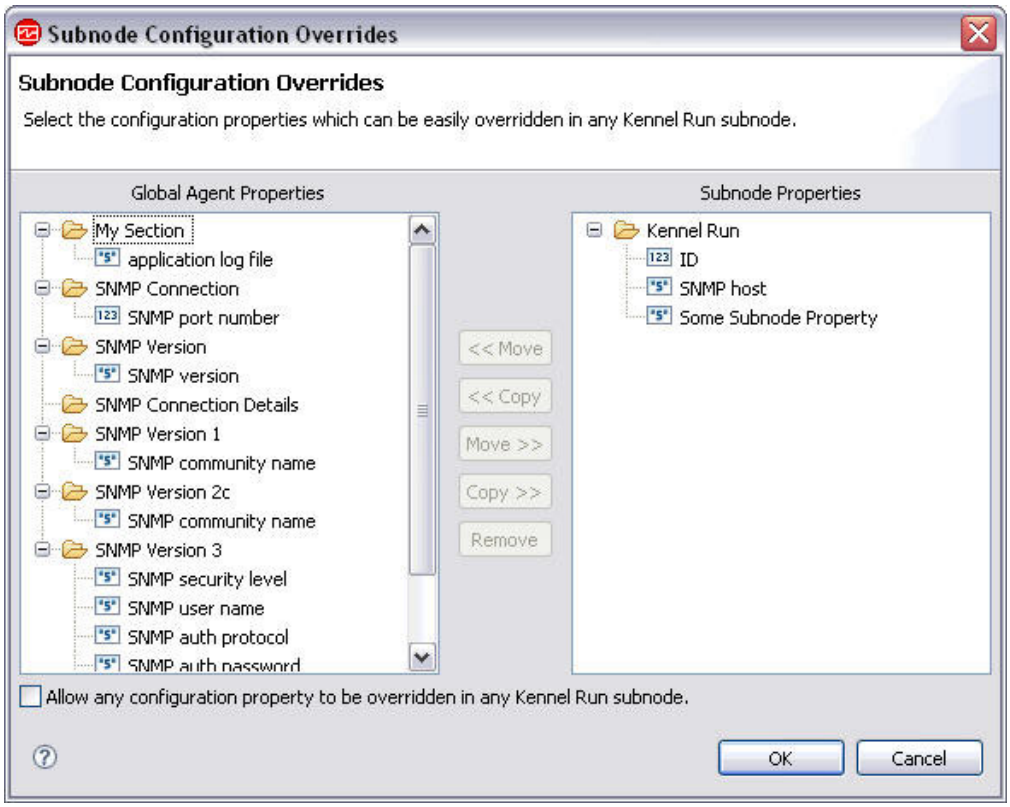


Figure 233. Subnode Configuration Overrides window

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, and the ones you want to vary for each subnode.

- Use **Copy >>** to copy configuration properties so that they are both located 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 that is located at both levels is a required property, it is required only at the agent level, and is always optional at the subnode level.

- Use **Move >>** to move properties from the agent level to the subnode level. **Move** is not permitted 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 only be removed 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.

You can change the configuration for an existing subnode using the Agent Editor.

Overriding custom configuration

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 234).

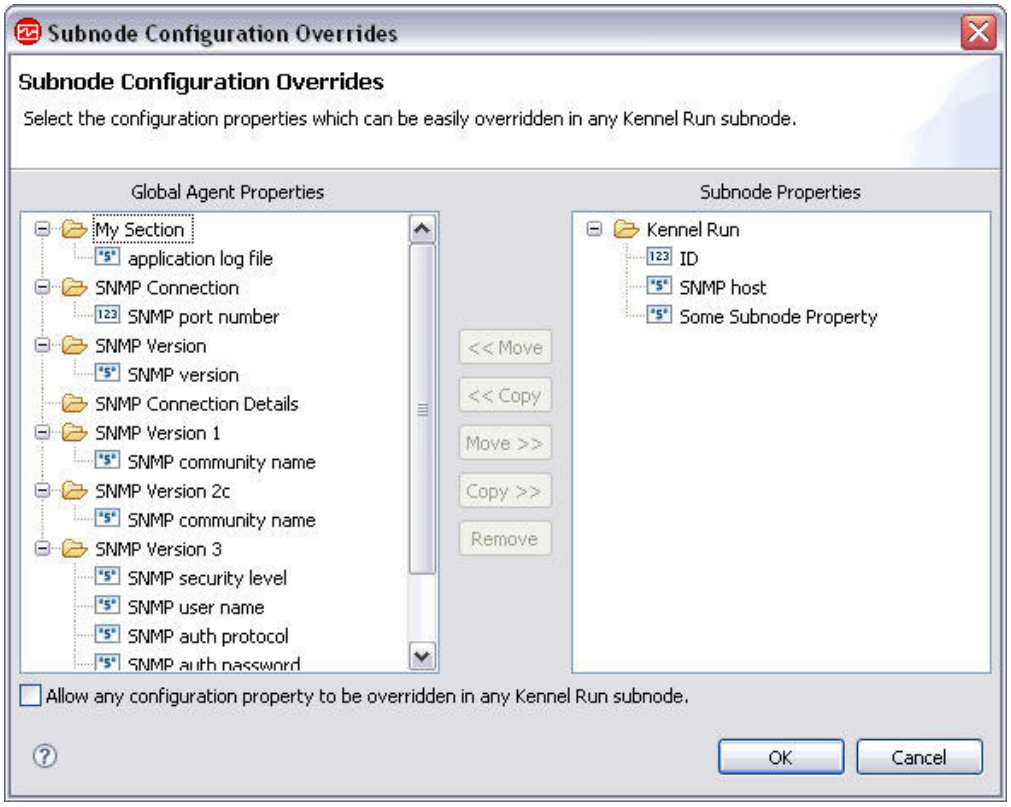


Figure 234. Overriding custom configuration

When copying or moving a custom property from the subnode level to the agent-level, you are prompted for the section in which to place the property , as shown in Figure 235 on page 347. You can select an existing custom section, or enter the name of a new custom section.

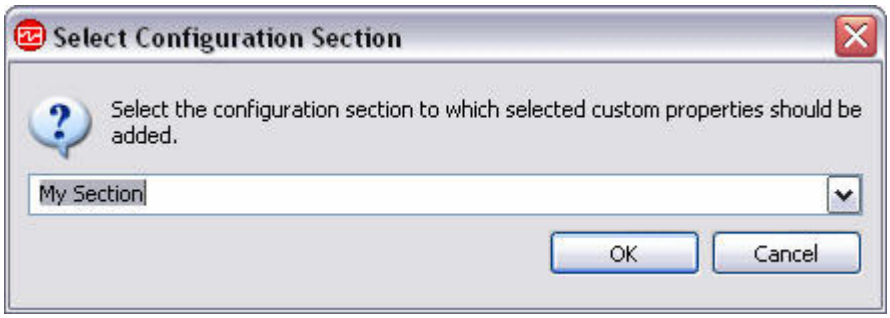


Figure 235. Select Configuration Section window

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 using the same community name. With subnodes, each instance of a subnode is capable of connecting 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 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 functionality might be limited when someone wants to vary a property from one subnode to the next that the agent developer did not think necessary to vary.

Also, the following properties cannot be copied to the subnode level, so 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 jars
- Socket data source port numbe
- JMX or JDBC class path settings

Advanced subnode configuration

There is an option in IBM Tivoli Monitoring V6.2.1 and later agents that can help overcome these limitations. On the Subnode Configuration Overrides page, there is a check box labeled **Allow any configuration property to be overridden in any subnode** (see Figure 234 on page 346). For this option to be enabled, you must select **6.2.1** as the **Minimum ITM version** when naming your agent (Step 12 “Naming your agent” on page 20.). If you choose this option, each subnode instance can override any property from any agent-level configuration section. But this override can only be performed from the GUI and not from the itmcmd command line.

This option causes an **Advanced** field that contains a drop-down list to be displayed on each subnode configuration panel (Figure 236 on page 348). The

initial selection in the **Advanced** field provides the brief directions: **Select a section to override values**.

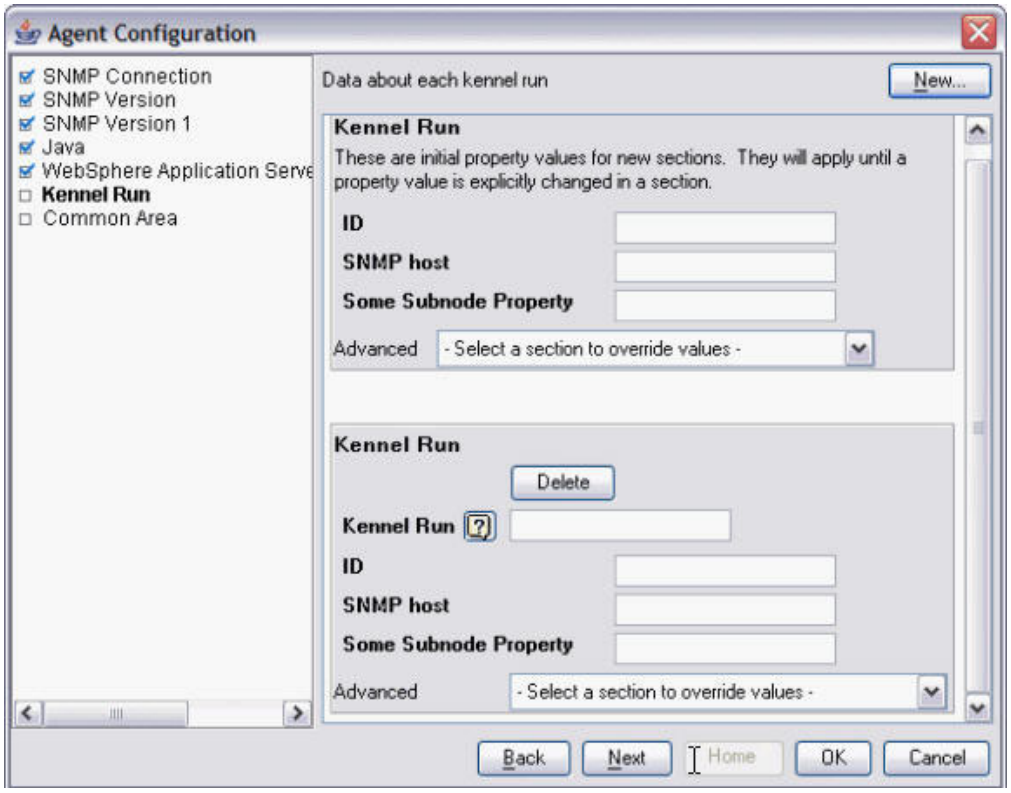


Figure 236. Override configuration drop-down lists

When you click on the drop-down list, you see a list of all the non-subnode sections that contain configuration properties (Figure 237 on page 349).

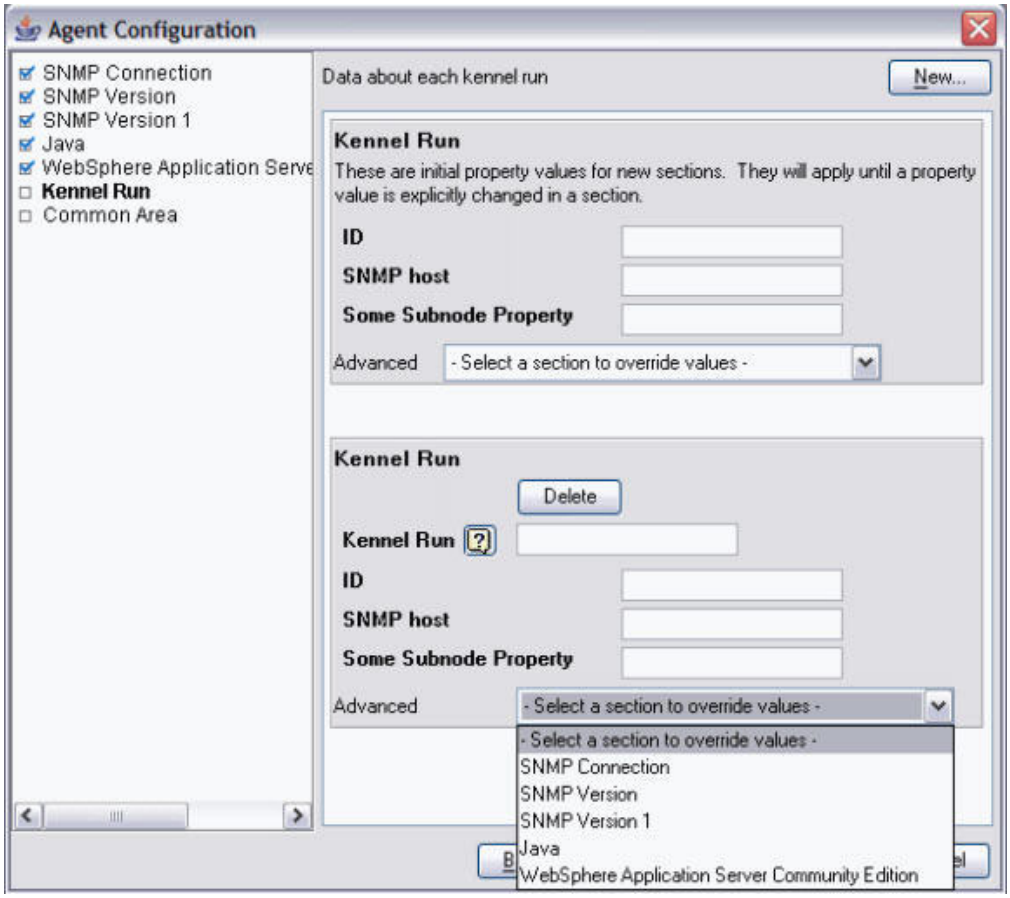


Figure 237. Sections to override

When you 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 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 238 on page 350.

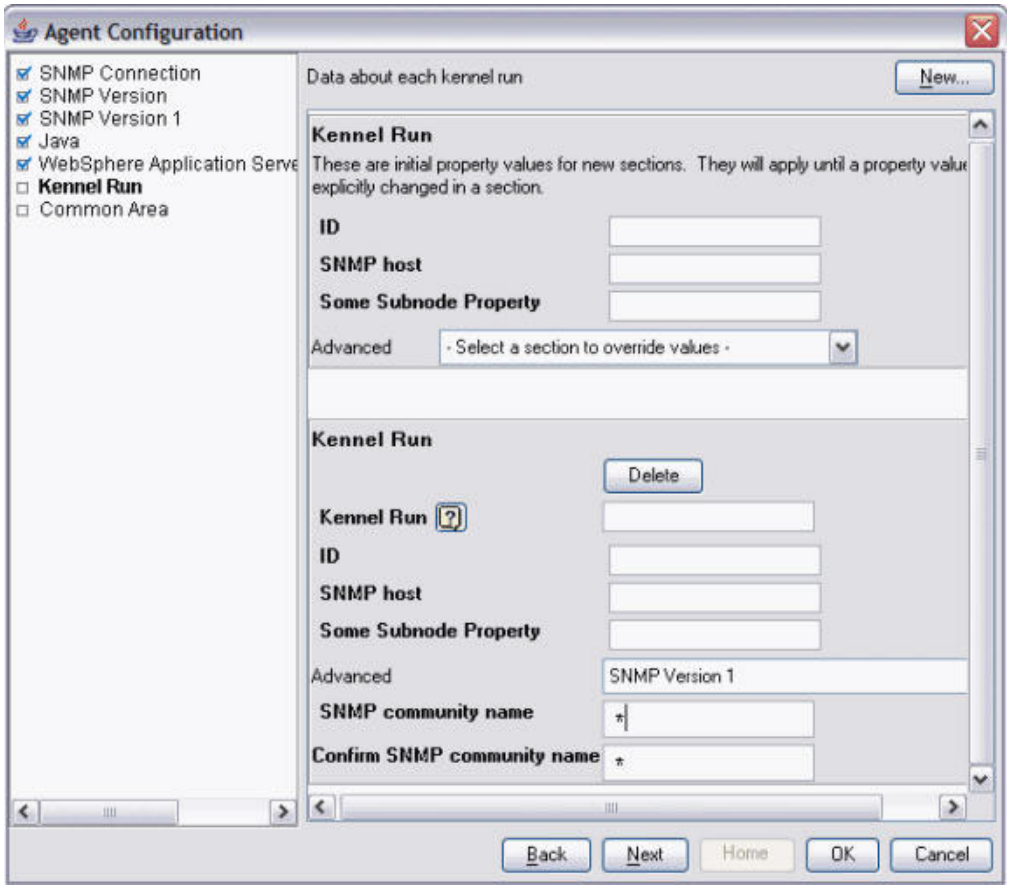


Figure 238. SNMP Version 1 Properties expanded

The following additional 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** drop-down list. For example, in Figure 238, SNMP host is not displayed below the **Advanced** drop-down list because it was copied to the subnode properties and is already displayed above.
- Sections that contain no properties to override do not have a selection in the **Advanced** drop-down list.
- Overridden values that you enter for one section are retained even if you select a different section to display different properties.
- To 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 will be changed to 6.2.1, although you are be prompted before the change is made so you can cancel the setting if you do not want to change the minimum version. 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 23).

Configuring a subnode from the command line

You can also configure a subnode using the command line. 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 defined during configuration.

property_name
: Name of the configuration property

value
: Value for the property

Example:

This example shows how to configure a sample agent that has defined 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 239 shows these configuration properties on the Runtime Configuration Information page of the Agent Editor.

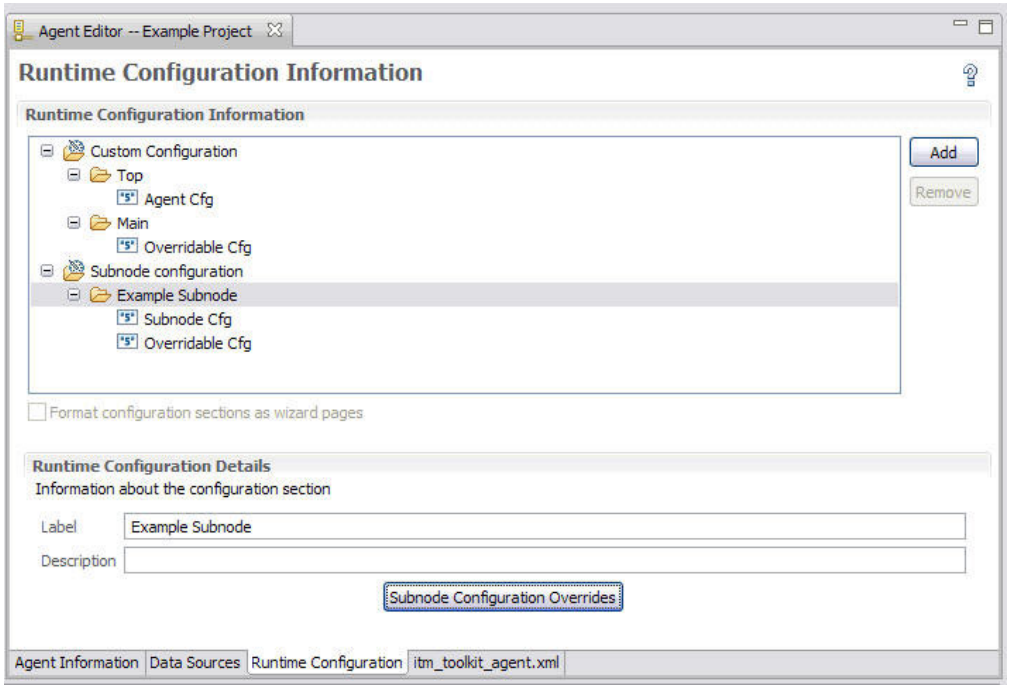


Figure 239. Configuration property definitions in the Agent Builder

When configuring this example agent, the first page that is displayed is the **Top** section, which contains the **Agent Cfg** property as shown in Figure 240 on page 352. Because this is an agent-level property, it is shown once during agent configuration. Any instance of the Example Subnode can see this property value,

but all instances see the same value.

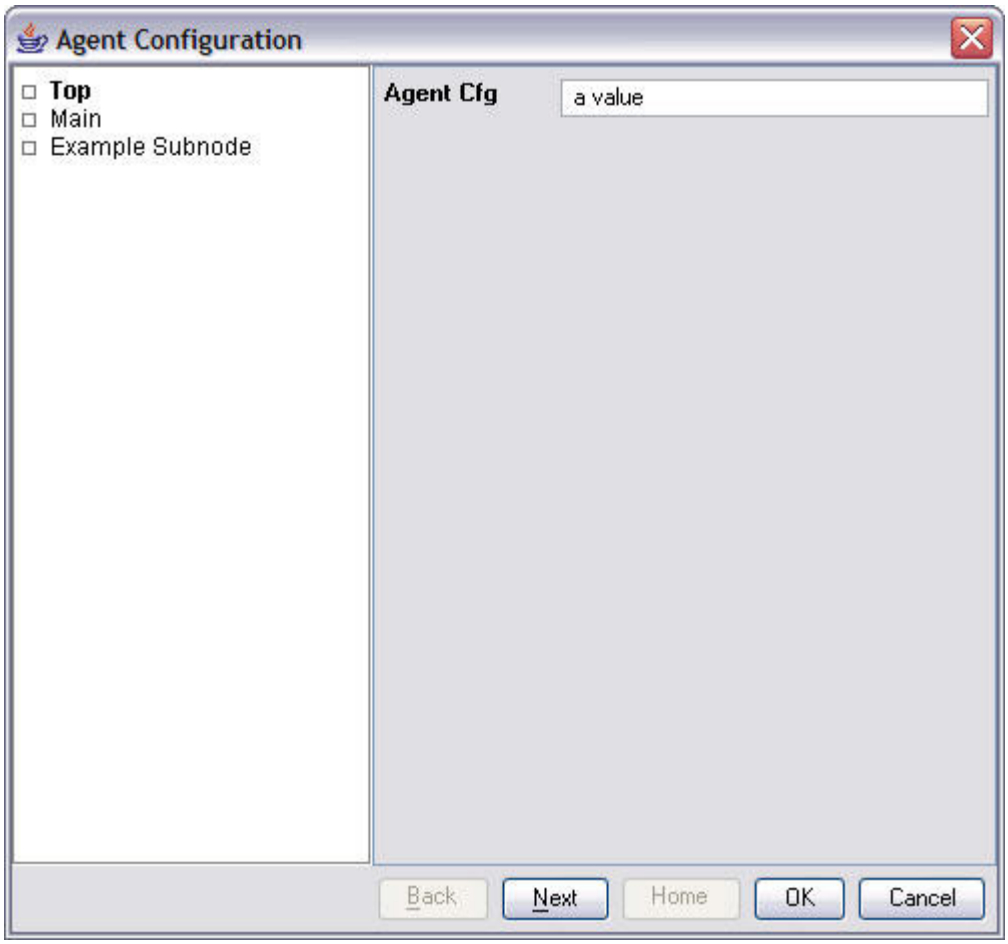


Figure 240. **Top** *section with agent-level configuration for the* **Agent Cfg** *property*

If you are configuring from the Tivoli Enterprise Monitoring Server command line, the Agent Cfg property can be set 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 241 on page 353. 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, but any Example Subnode instance can override the value entered here with a different value.

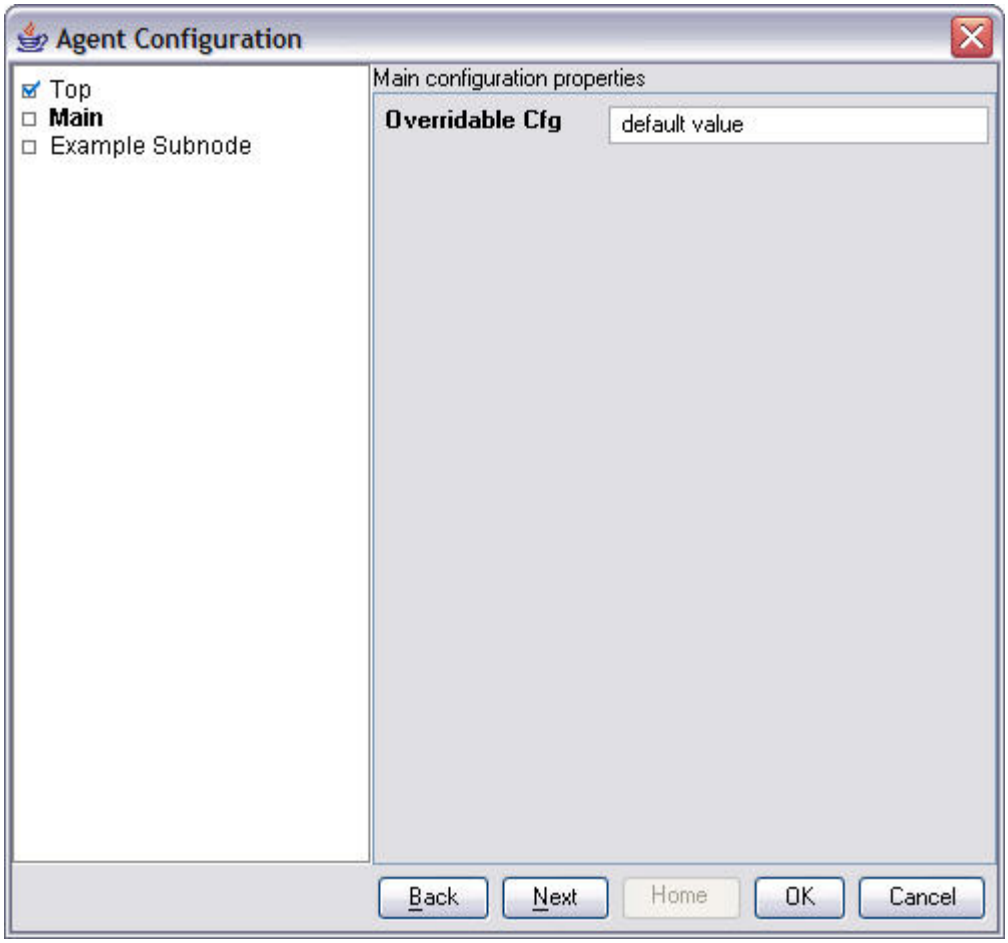


Figure 241. **Main** *section with the agent-wide default value for the* **Overridable Cfg** *property*

If you are configuring from the Tivoli Enterprise Monitoring Server command line, this property can be set 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 242 on page 354. Because this is the first time to configure this agent, there are no subnode instances defined 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.

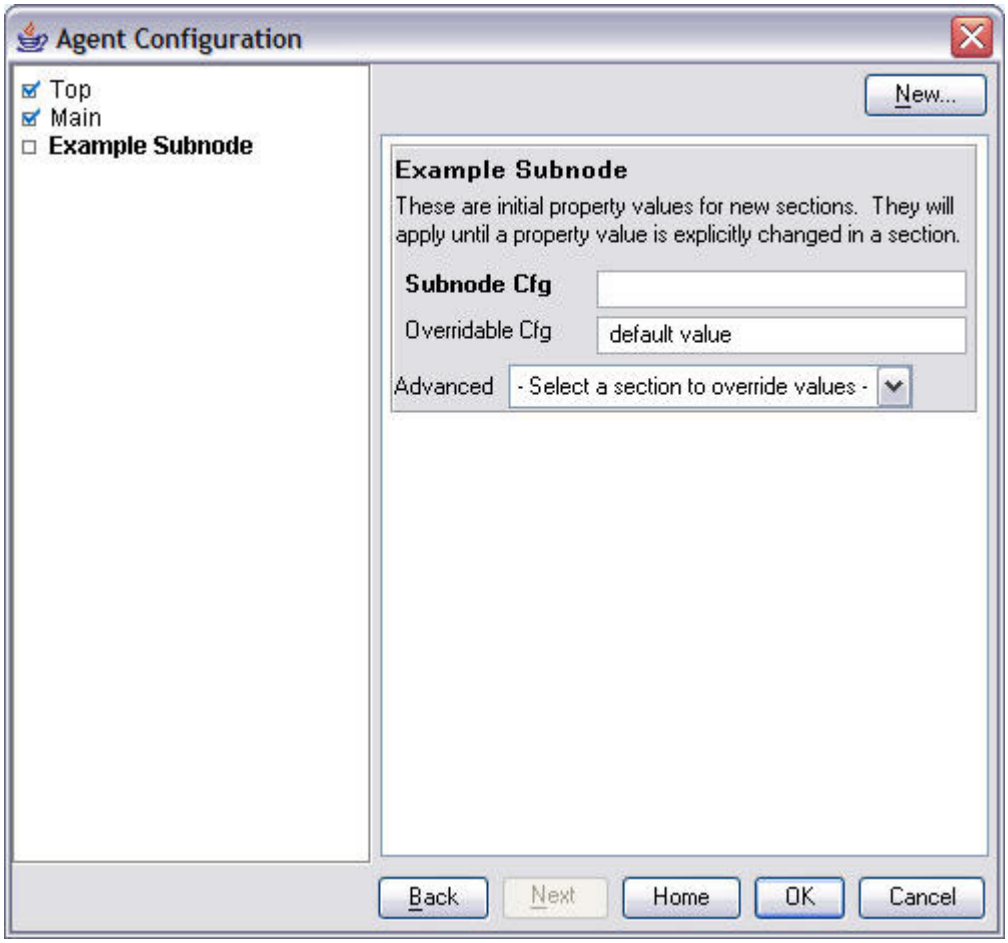


Figure 242. **Example Subnode** *section page with no subnode*

Subnode instances are defined by performing the following actions on the empty **Example Subnode** section page (Figure 243 on page 355):

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**. The first **Example Subnode** subsection is displayed below 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 appears below 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`.

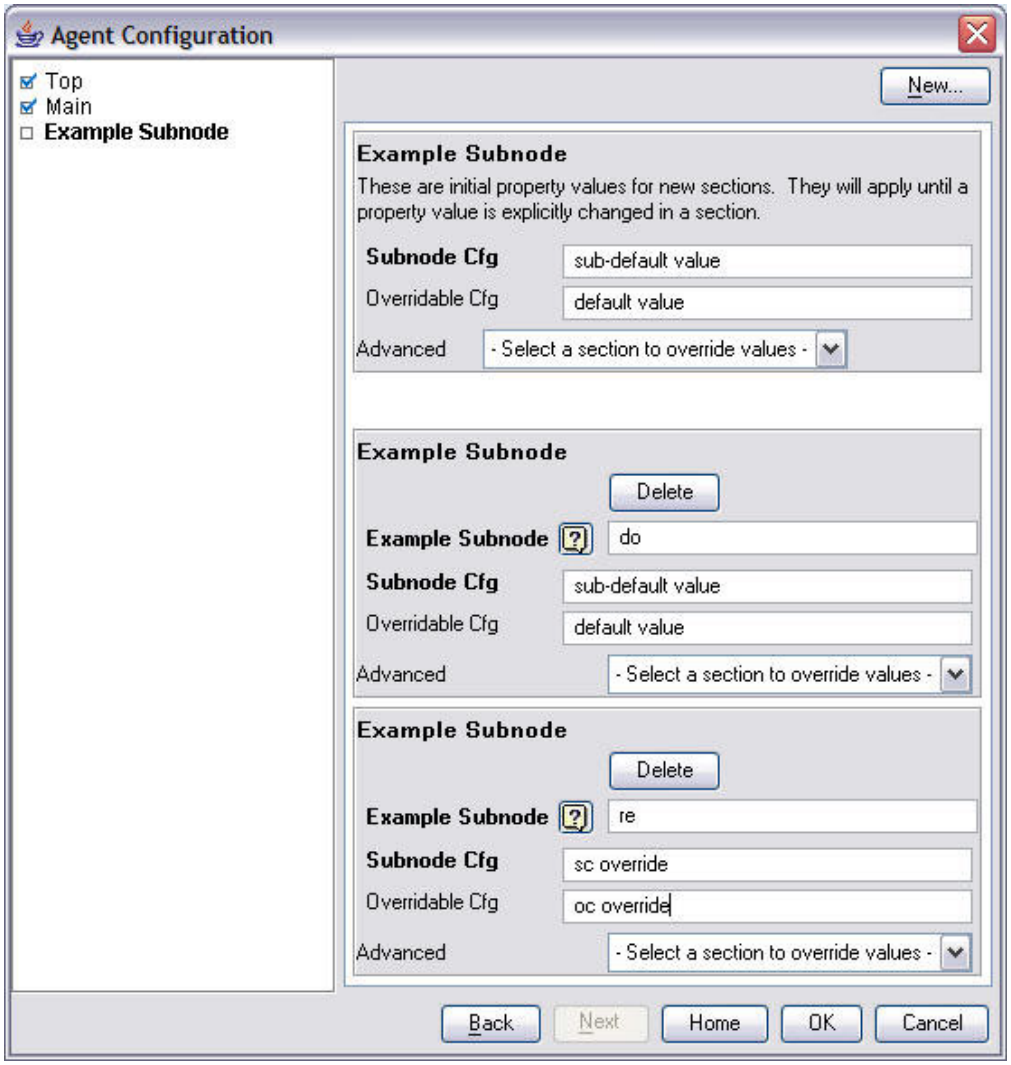


Figure 243. **Example Subnode** *section page with two subnode instances defined*

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 using the command line to create a subnode instance, at least one property

must be specified, even if all the properties use default values. Other than that, default values do not need to be specified on the command line when defining 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"
```

Windows data sources

If an agent has Windows data sources (Windows Event Log, WMI, Perfmon) at only the agent level and not in subnodes, including Windows Remote Connection configuration properties in the agent is optional. If they 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:

1. On the Windows Management Instrumentation (WMI) Information page, click **Global Options** when displaying the data source properties, either while creating the data source or from the Agent Editor Data Sources page (Figure 244).

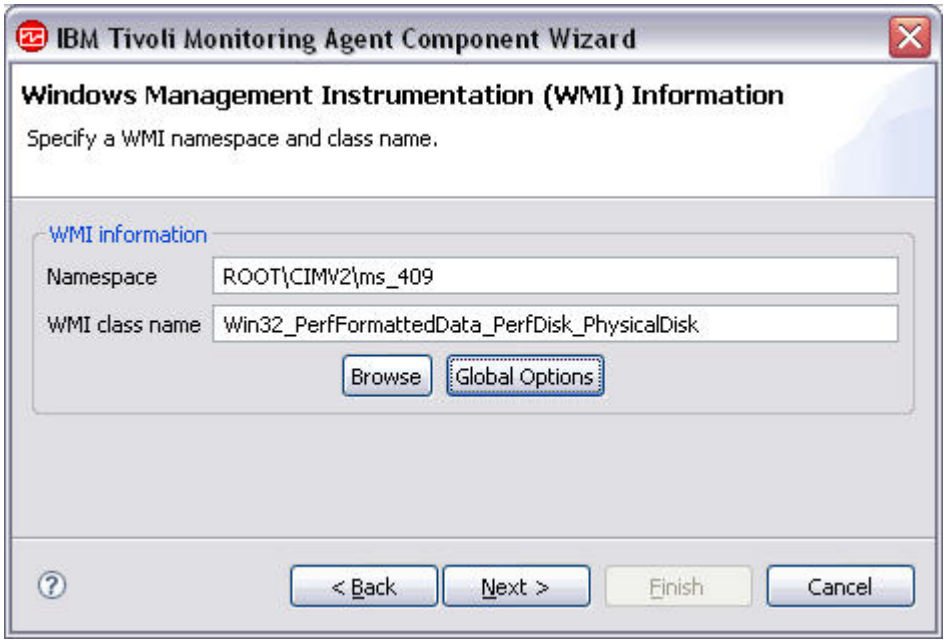


Figure 244. Windows Management Instrumentation (WMI) Information page

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 245 on page 357).

Figure 245. Global Windows Data Source Options

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. Configuring an agent

If you are adding SNMP, JMX, CIM, JDBC, HTTP, and SOAP data sources to your agent, you configure these data sources while defining the data source 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 141
- Chapter 15, "Monitoring data from a Common Information Model (CIM)," on page 173
- Chapter 21, "Monitoring data from Java Database Connectivity (JDBC)," on page 237
- Chapter 23, "Monitoring HTTP availability and response time," on page 261
- Chapter 24, "Monitoring data using SOAP," on page 273

This chapter provides information about the following topics:

- Customizing the configuration of process, log file, and script data sources so an agent can access the application it is monitoring
- Changing configuration properties using the Agent Editor
- Configuring a Windows remote connection
- Configuring an SSH remote connection

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 additional 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, or the user ID and password to access an interface, you can add configuration properties to prompt for this information.

Customizing configuration

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 your agent requires system-specific information for one of the areas of data collection (except script environment variables), either the **Insert Property** or the **Insert Configuration Property** button is displayed when you are defining the data source.

For example (Figure 246), when creating an attribute group that monitors a log file, an **Insert Configuration Property** button is located just below the **Log file name** field.

Figure 246. **Insert Configuration Property** *button on the Log File Information page*

1. Click **Insert Configuration Property** to display the Configuration Properties window (Figure 247 on page 361).

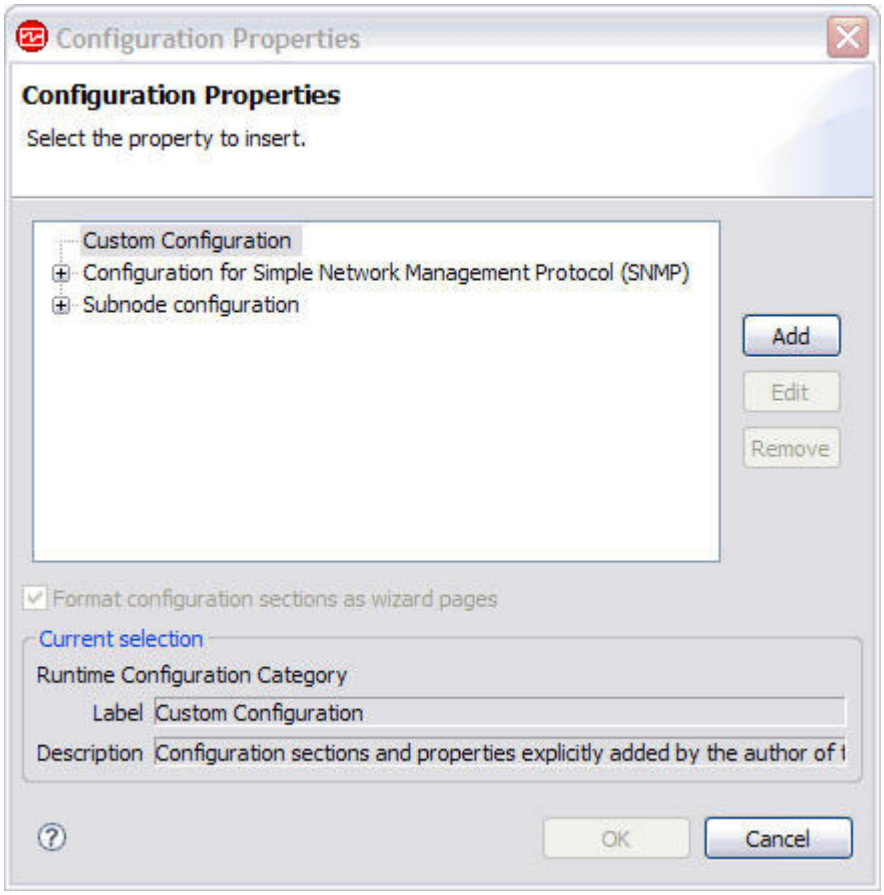


Figure 247. Initial Configuration Properties window

2. In the Configuration Properties window, click a property, and click **Add**.

   Note: Initially there are no configuration properties defined for the agent.
3. In the Runtime Configuration Property window (Figure 248 on page 362), complete the fields as described in this step.

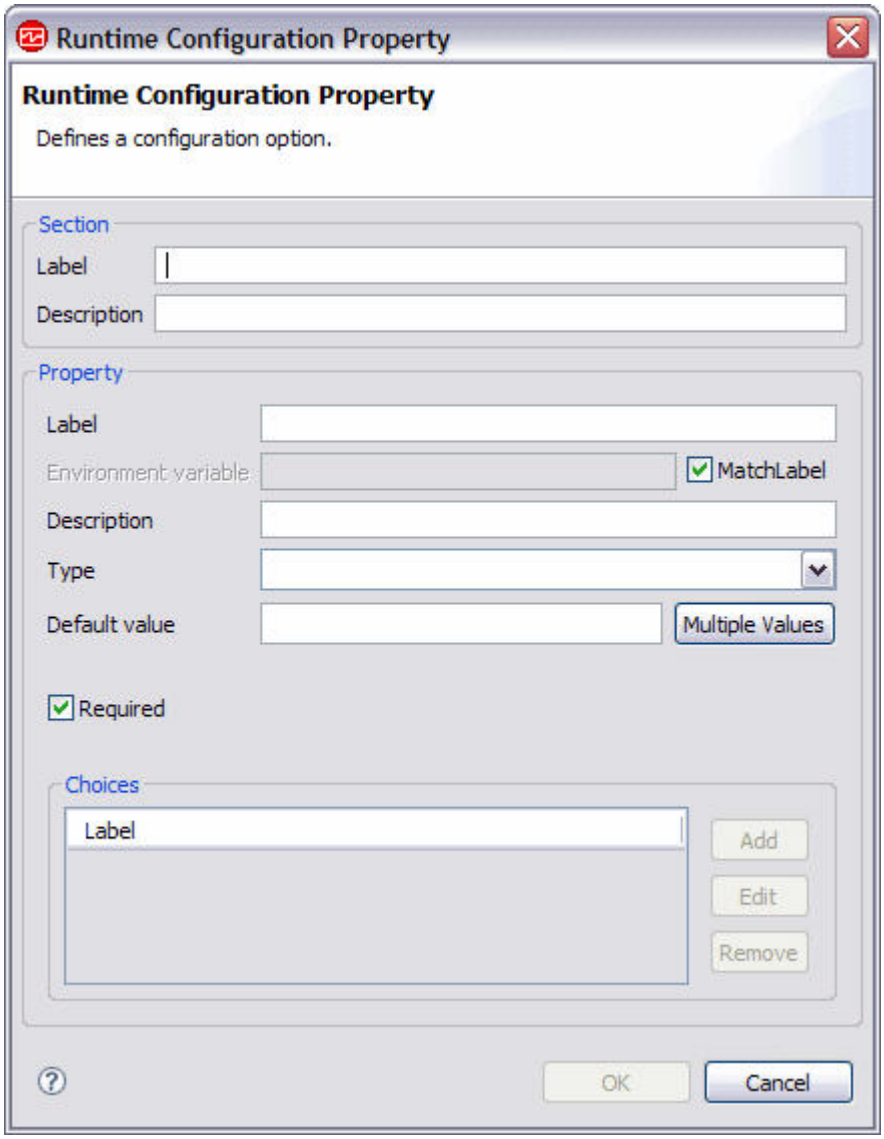


Figure 248. Runtime Configuration Property window

a. In the **Section** area, complete the following fields:

   Label Text that describes the properties in this section

   Description
   (optional) Description of the properties in this section

b. In the **Property** area, complete the following fields:

   Label Text that is displayed in the agent configuration panel that identifies the information you should 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. You can clear the **Match Label** check box if you want to change the environment variable independently from the label.

Description
(optional) Description of the property that is being defined.

Type Type of information being collected, one of the following options:

String For any alphabetic information that needs to 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 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 entered are displayed in the agent configuration panel as a group of radio buttons, you can make only one selection from the group of buttons.

Read Only Text
Displays text when configuring 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. A **Browse** button enables you to browse the file system for the desired file.

Default value
(Optional) Specify the value that will appear in the configuration panel at runtime 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 249 on page 364), specify the default values you want for Windows and for UNIX and Linux.

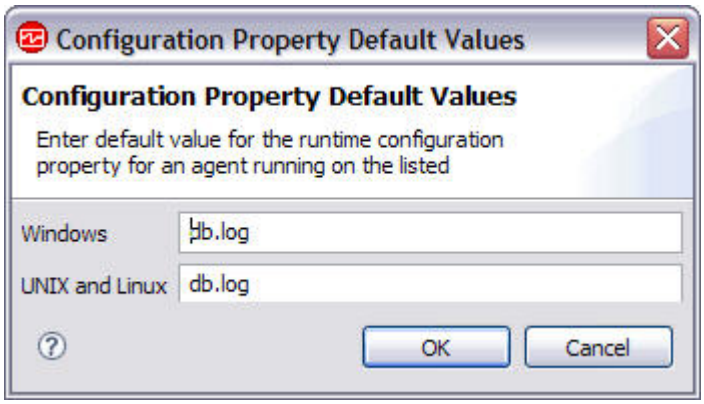


Figure 249. Configuration Property Default Values window

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 this operation or continue with V6.2.1 compatibility enabled.

Required
Check this field if the user must enter a value when the agent is configured, or clear this field if entering a value is optional.

c. To add a choice, click **Add**

4. In the Configuration Property Value window (Figure 250), fill in the **Label** and **Value** fields. The Label is displayed as one of the choices, and the Value becomes the property value if this choice is chosen.

Figure 250. 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 251 on page 365).

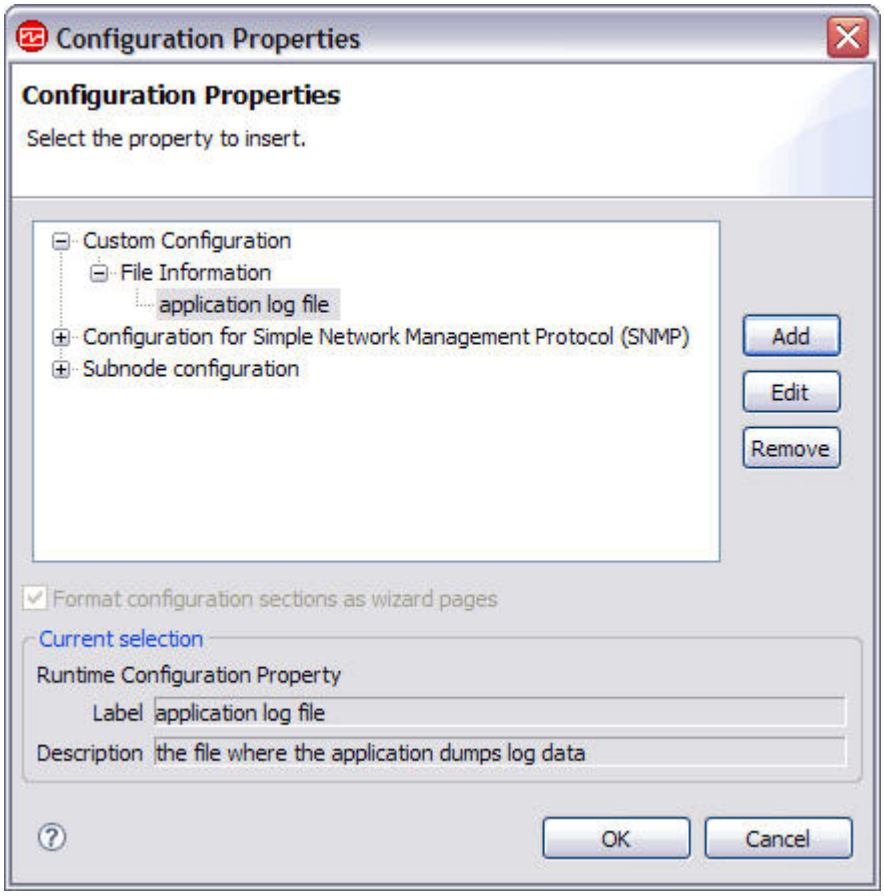


Figure 251. New configuration property displayed

6. *(Optional)* To add another property to an existing section, select the section or an existing property in the section in the runtime configuration tree of the Configuration Properties window, and click **Add**.

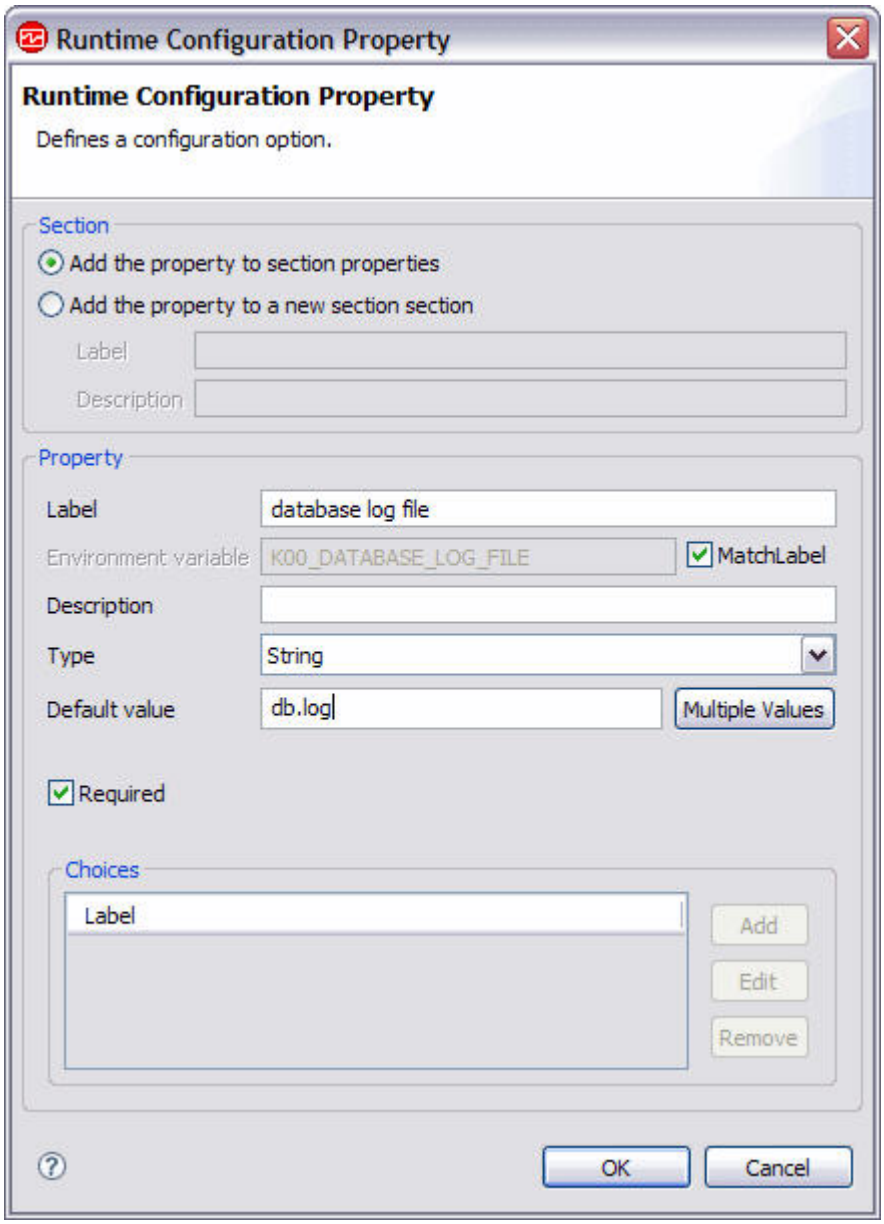


Figure 252. Adding a second property to an existing section

7. Complete the fields for the new property. (Same fields as in Step 3).
8. Click **OK**. The property you most recently added is selected.
9. Keep the selection or select the property you want to insert into the log file name.
10. Click **OK**. The property is inserted into the log file name.

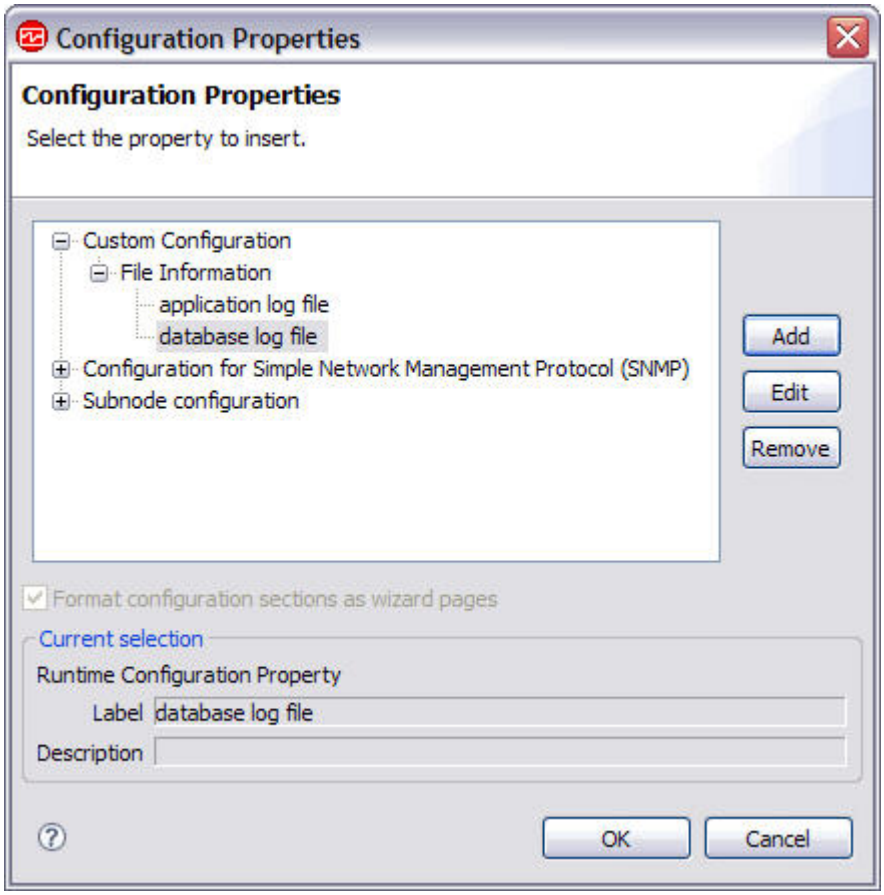


Figure 253. Property inserted into the log file name

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 also be used in other locations that accept a configuration property, including a script data source. This means that you can access the value for the configuration element File Information with the script variable *$K00_APPLICATION_LOG_FILE* or the Windows batch file variable *%K00_APPLICATION_LOG_FILE%* if the product code is K00.

Changing configuration properties using the Agent Editor

You can view, add, and change the configuration properties using the Agent Editor.

1. Click the **Runtime Configuration** tab.

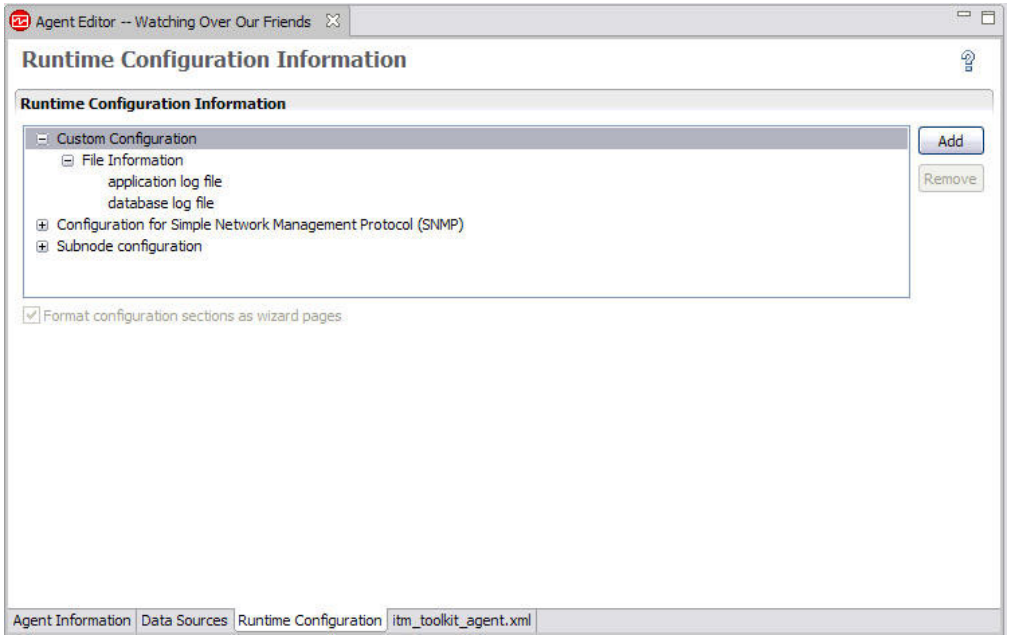


Figure 254. **Runtime Configuration** *tab in the Agent Editor*

2. Select a configuration section, and click **Add**.

   The **Add** button works just like it did in "Customizing configuration" on page 359. There is no **Edit** button because a configuration section or property is edited on this page when it is selected.

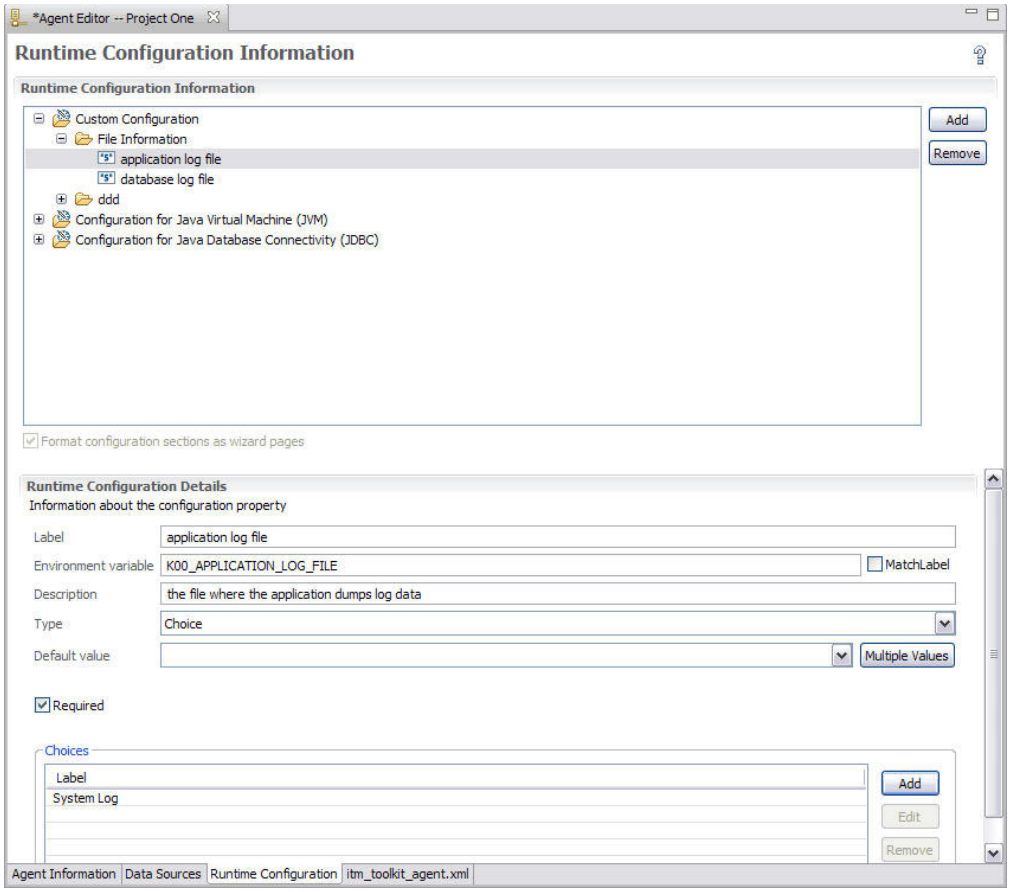


Figure 255. Editing a configuration section in the Agent Editor

3. Select a configuration property to display the **Runtime Configuration Details** area.

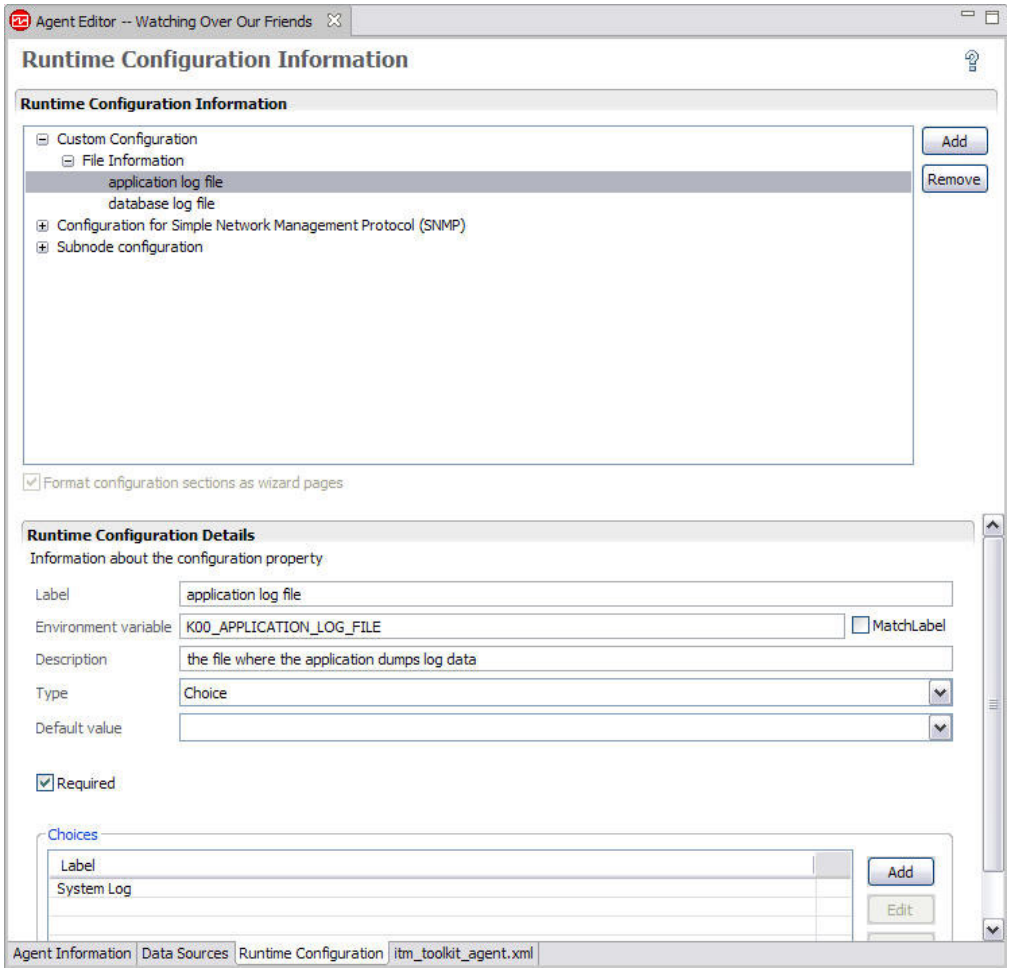


Figure 256. Editing a configuration property in the Agent Editor

4. In the **Runtime Configuration Details** area, edit the fields to configure the property.

Configuring a Windows remote connection

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, and also on remote Windows systems. These three data source types re referred to as "Windows data sources." If these Windows data sources are monitoring data remotely, they all share the same 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 to maintain compatibility with earlier versions of agents that might have used 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 want the base agent to remotely monitor a single remote Windows system, select the **Global Options** button on the Windows Data Source window to open the Global Windows Data Source Options window where you can select **Include Windows Remote Connection configuration**.

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 allow Windows Remote Connection if it is in a subnode and 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 will share the same Windows Remote Connection configuration properties.

You can view, add, and change the configuration properties using the Agent Editor. See "Changing configuration properties using the Agent Editor" on page 367. If a Windows data source is defined in a subnode, you can also specify Subnode Configuration Overrides. See "Subnode configuration" on page 344.

The following connection specific configuration properties are on the Windows Remote Connection configuration page:

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

Configuring a Secure Shell (SSH) remote connection

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 the same SSH remote connection configuration properties for the agent level where they are defined. To maintain compatibility with earlier versions of agents that might have used the script data provider before remote monitoring was enabled, 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 want the agent to remotely monitor a remote system, click **Global Options** on the Script Data Source window to open the Global Script Data Source Options window. Here you can select the **Include SSH Remote connection configuration** check box. If you define a Script data source in a subnode in your agent and you choose to enable the SSH remote connection configuration properties, you can configure each instance of a subnode to monitor a different remote system. All script data sources in the subnode share the same SSH remote connection configuration properties.

You can view, add, and change the configuration properties using the Agent Editor. See ("Changing configuration properties using the Agent Editor" on page 367). 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).

The following connection-specific configuration properties are on the SSH remote connection configuration page:

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 logging in 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 if the script data provider drops the login session to the remote system after collecting 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 are enabled on the Password Configuration page:

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 are enabled on the Public Key Configuration page:

Username
User name associated with the public key file

Public Keyfile
Public key file associated with the user

Private Keyfile
Private key file associated with the user

Password
Password used to unlock the private key file

Chapter 31. Testing and debugging your agent

After you have used the Tivoli Monitoring Agent Builder to create an 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 will learn to modify or tweak settings in the agent to ensure that the data displayed to you is beneficial and accurate.

To begin use the Agent Builder test function to test your agents attribute groups. Later you can install the agent and perform further agent testing with the Tivoli enterprise Portal.

Attribute group testing

Agent Builder supports an attribute group test function for most data sources.

You start the Testing procedure in the following ways:

1. During agent creation click **Test** on the relevant data source Information page.
2. After agent creation, select an attribute group on the Agent Editor Data Sources Definition page and click **Test** . For more information about the Agent Editor, see "Tivoli Monitoring Agent Editor" on page 37.

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 257.

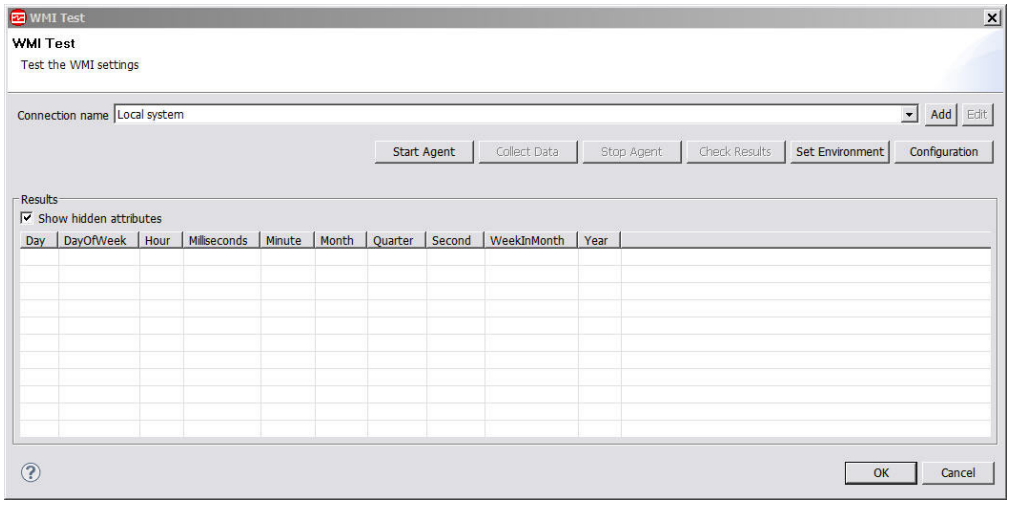


Figure 257. WMI Test window

(Optional) Before starting your test you can set environment variables and configuration properties. For more information, see "Configuration prior to testing" on page 374.

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" on page 110

- Windows Performance Monitor (Perfmon), for more about the Perfmon test procedure, see "Testing" on page 118
- Simple Network Management Protocol (SNMP), for more about the SNMP testing, see "Testing" on page 128
- Simple Network Management Protocol (SNMP) event sender, for more about the SNMP event test procedure, see "Testing" on page 137
- Java Management Extensions (JMX), for more about the JMX test procedure, see "Testing" on page 170
- Common Information Model (CIM), for more about the CIM test procedure, see "Testing" on page 177
- Log file, for more about the log file test procedure, see "Testing" on page 195
- Script, for more about the script test procedure, see step 9 on page 231 in "Steps for monitoring output from a script" on page 227
- Java Database Connectivity (JDBC), for more about the JDBC test procedure, see "Testing" on page 252
- Internet Control Message Protocol (ICMP) ping, for more about the ICMP test procedure, see "Testing" on page 258
- Hypertext Transfer Protocol (HTTP) Availability, for more about the HTTP test procedure, see "Testing" on page 271
- Simple Object Access Protocol (SOAP), for more about the SOAP test procedure, see "Testing" on page 281
- Transmission Control Protocol socket (TCP) socket, for more about the socket test procedure, see "Testing" on page 295
- Java Application Programming Interface (API), for more about the Java API test procedure, see "Testing" on page 314

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), the services that are installed on the system (windows services) and the Windows Event Logs that are present on the system.
- There is little or no customization that you can do in the agent (AIX Binary Log, command return code)

Note: When testing data sources, after you click the **Collect Data** button, data may not be displayed at all or may not be current after the first click, In this case, click **Collect Data** a second time to display current data.

Configuration prior to testing

Before starting your test you can optionally set environment variables and configuration properties from the data source Test window

1. To set environment variables, click **Set Environment** , This displays the Environment Variables window where you can set the variables and the values. Once 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 will contain any existing environment variables created by you for your agent. It will also contain any environment variables that you added from previous tests of this agent.

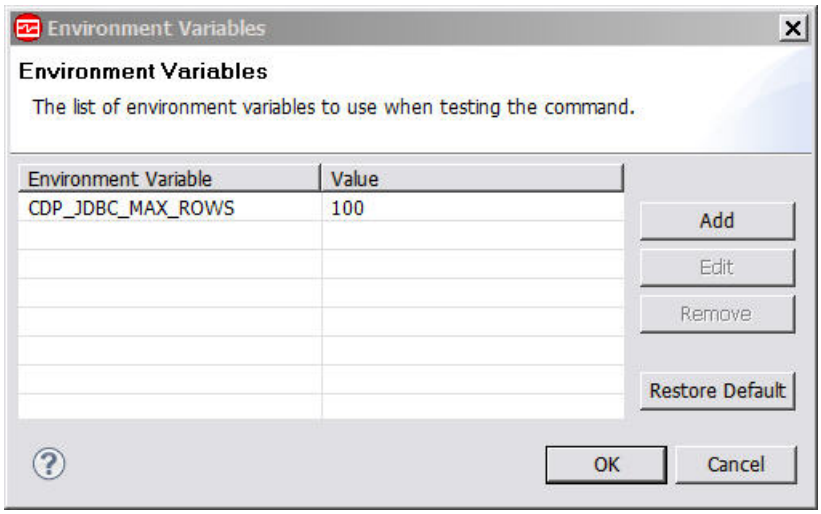


Figure 258. Environment Variables window

a. To add or edit individual variables, click **Add** or **Edit**
b. Click **Remove** to remove individual variables, or **Restore Default** to restore default variables and remove all others.
c. Click **OK** to implement your changes.

2. To set configuration properties, click **Configuration** which displays the Runtime Configuration window where you can add configuration properties.

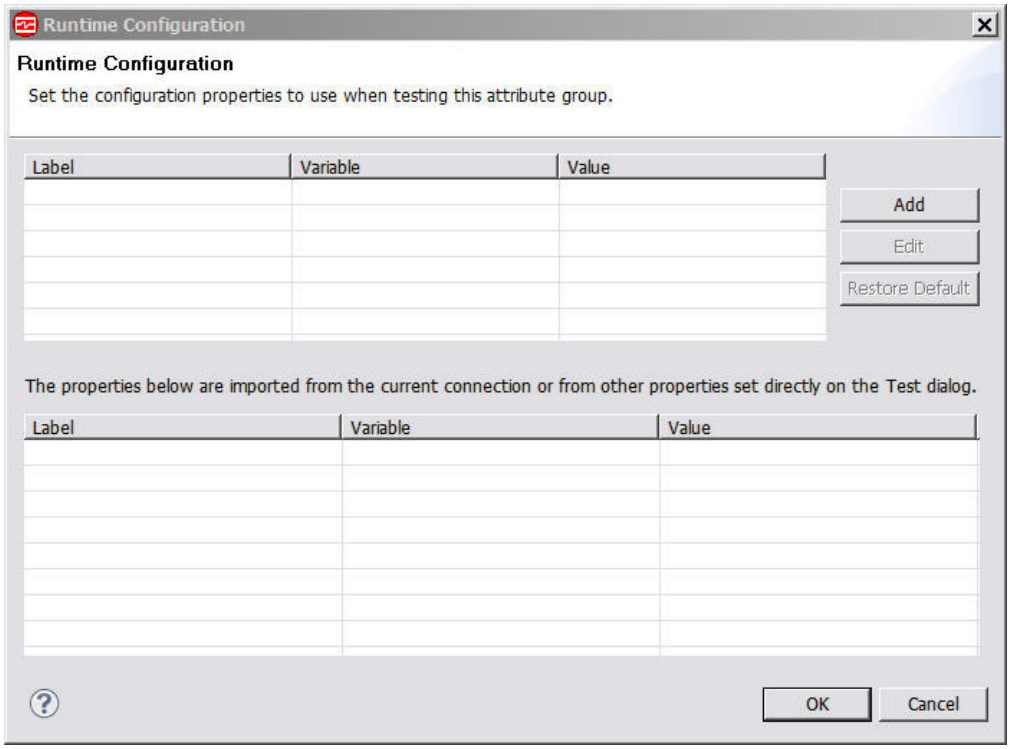


Figure 259. Runtime Configuration window

3. Click **OK** to return to the Test window.

Setting Java Information

For certain attribute groups you can additionally set Java information such as debug level using the **Java Information** button.

You can set Java information from the Test window of the following types of attribute groups:

- Java Management Extensions (JMX)
- Java Database Connectivity (JDBC)
- Hypertext Transfer Protocol (HTTP) Availability
- Simple Object Access Protocol (SOAP)
- Java Application Programming Interface (API)

From the attribute group Test window, you set Java information as follows:

1. Click the **Java Information** button.

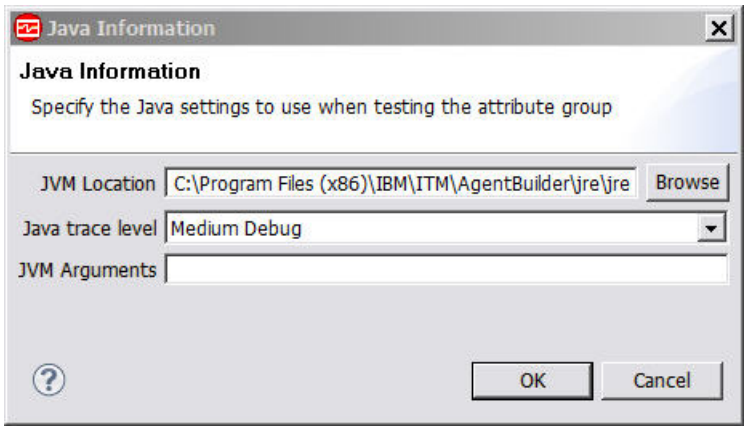


Figure 260. Java Information window

2. Enter Java Information, for example, Browse to or type the location of a Java virtual machine (JVM), select a Java tracel level, or enter JVM arguments
3. Click **OK** to return to the Test Settings window.

Debugging

Each data source that is tested has a test directory created for it by Agent Builder. This directory is used for the data source's test runtime environment. Log files relating to tests run on the data source are stored under this directory. The log files can be useful to help debug issues found during testing.

Notes:

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 261 on page 377

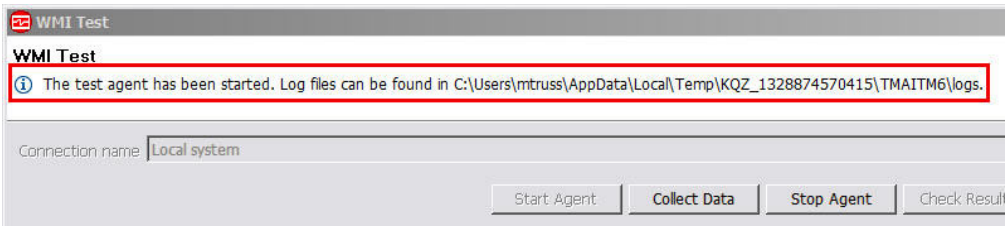


Figure 261. WMI Test window showing test log location.

2. All test data source directories are deleted when the Agent Builder is shut down.

Installing and testing an agent

This section describes the two methods for installing and testing the agents you created 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.

Notes:

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 397 in this guide.
2. After you have installed 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 386. If after viewing the data in the Tivoli Enterprise Portal, you determine you need to make modifications to the agent, see Chapter 6, "Modifying your agent by using the Tivoli Monitoring Agent Editor," on page 37.
3. For an agent that supports UNIX, generate the installer image on a UNIX system because this creates the files with the appropriate permissions.

Installing the agent locally

Use the following procedure when you have finished 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 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.

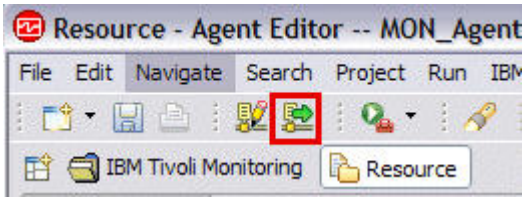


Figure 262. Generate agent icon

c. Double-click the `itm_toolkit_agent.xml` file and select IBM Tivoli Monitoring Agent Editor > **Generate Agent** (Figure 263).

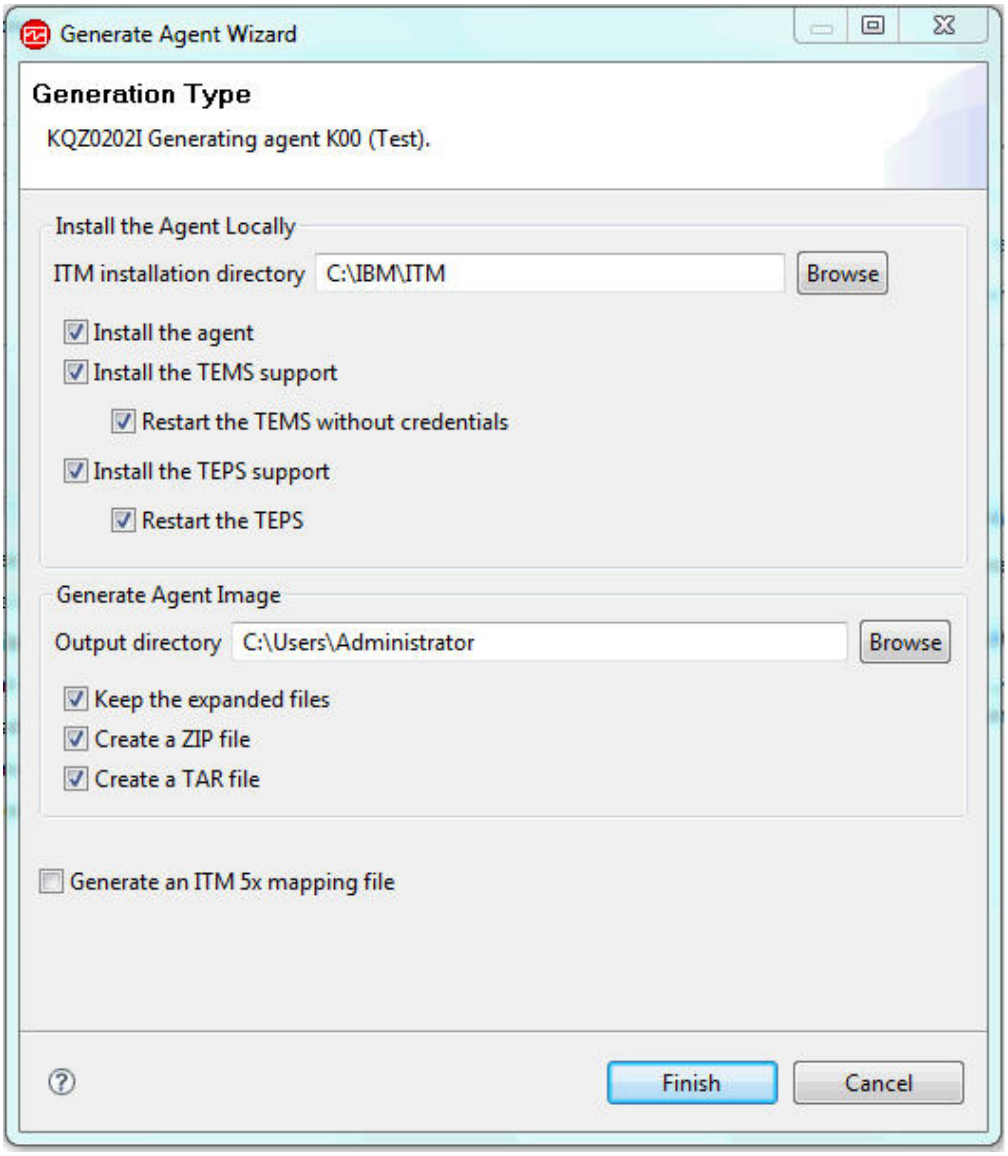


Figure 263. Generate agent menu option

2. In the **Install the Agent Locally** section, type the installation directory for IBM Tivoli Monitoring. The Agent Builder fills in 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 checkboxes are enabled as follows:

Install the Agent
Is enabled if the Agent Builder detects an appropriate TEMA in the location specified (appropriate means 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, TEMS support, TEPS support).
4. Click **Finish**.
5. Configure and start the agent (see "Configuring and starting the agent" on page 382).

   If you are running Tivoli Monitoring v6.2 Fix Pack 1 or later, you can install the TEMS and TEPS 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 will cause the Tivoli Enterprise Monitoring Server to be recycled).

   Note: The Tivoli Enterprise Monitoring Server must be running in order to be able to install support files without recycling the Tivoli Enterprise Monitoring Server.

Creating the agent image

Use the following procedure to create the agent package:

1. Select the `itm_toolkit_agent.xml` file 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 263 on page 378).
2. In the Generate Agent Image section Figure 264 on page 380, enter the name of the directory where you want to keep the expanded files or put the compressed file or files.

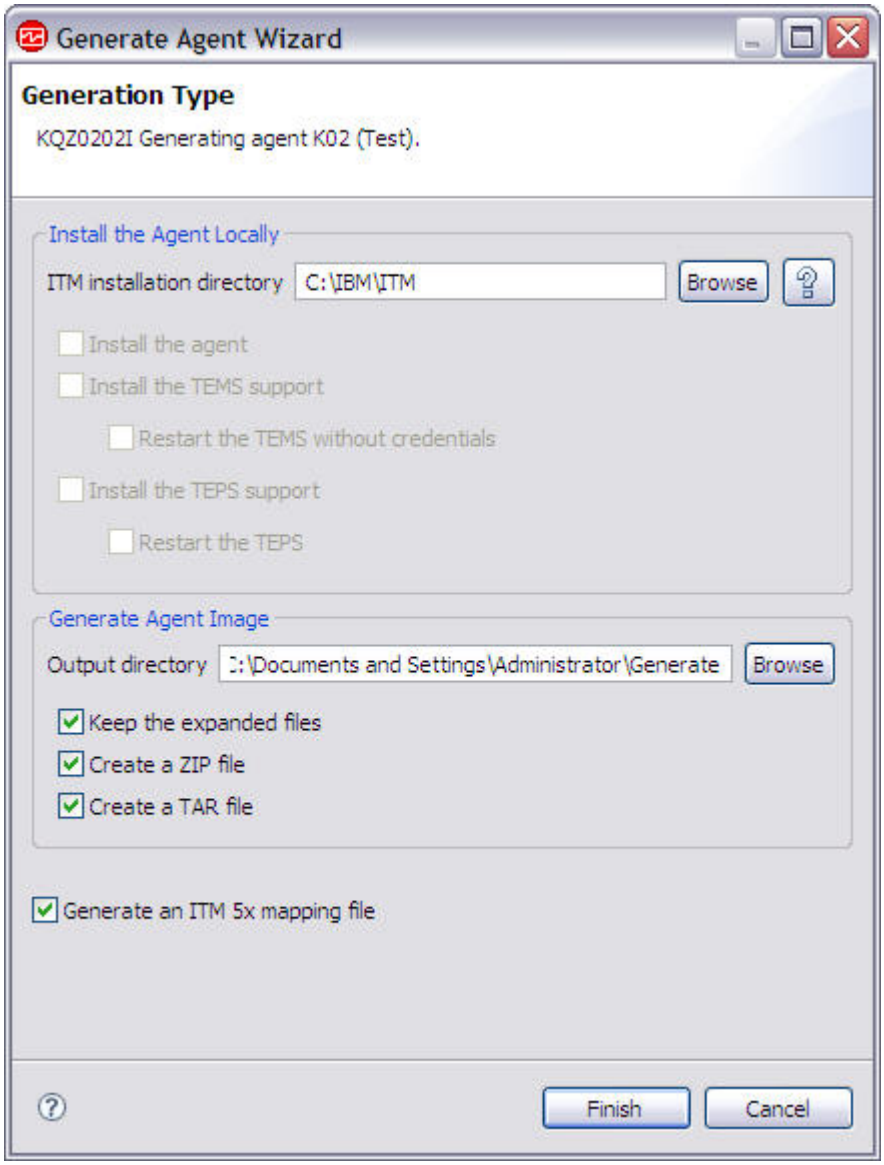


Figure 264. Create an agent image

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 ZIP 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" on page 380.

Installing the package

Use the following procedure to install the package on other systems:

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 lower case.

   You can install remotely 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.
   - InstallIra.bat/.sh *itm_install_location* [[-h *Hub_TEMS_hostname*] -u *HUB_TEMS_username* -p *Hub_TEMS_password*]-, installs the agent, Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal support all at once.
   - installIraAgent.bat/.sh *itm_install_location* - for the agent-only installation.
   - installIraAgentTEMS.bat/.sh *itm_install_location* [[-h *Hub_TEMS_hostname*] -u *HUB_TEMS_username* -p *Hub_TEMS_password*] - installs the Tivoli Enterprise Monitoring Server support.
   - installIraAgentTEPS.bat/.sh *itm_install_location* - installs the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal support.

   The install 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 a Tivoli Monitoring v6.2 Fix Pack 1 system and do not provide a userid to the installIra or installIraAgentTEMS, then the Tivoli Enterprise Monitoring Server will be recycled after the support files are installed. The Tivoli Enterprise Monitoring Server will restart 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 permitted, because the new function that uses them is not available. On a Tivoli Monitoring v6.2 Fix Pack 1 system, if you provide the new arguments, then Tivoli Enterprise Monitoring Server support will be loaded without causing the Tivoli Enterprise Monitoring Server to restart.

   Note: The agent, Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal support must be installed before the agent is displayed correctly in the Tivoli Enterprise Portal.
4. Configure and start the agent (see "Configuring and starting the agent" on page 382).

After you install the agent

After installing the agent, if you have made any changes to the layout of your agent that cause Navigator items to be moved or removed, restart the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal for the changes to be correctly recognized.

Configuring and starting the agent

This section describes the procedure for installing agents that you created with the Tivoli Monitoring Agent Builder.

1. Open **Manage Tivoli Monitoring Service**. The new entry `Monitoring Agent for agent_name` is displayed.

   Note: If the **Manage Tivoli Monitoring Service** window is already displayed, 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.

   Notes:

   a. On UNIX systems, the option to select is **Configure**.

   b. In the case of multi-instance agents, when you are configuring, you are prompted for an instance name.
3. If you added runtime configuration elements to your agent, or if you selected a data source, then you are presented with configuration panels to collect the required information for your agent.
4. Start the agent.
5. Open the Tivoli Enterprise Portal and navigate to the new agent.

Results after generating and installing the agent with the Agent Builder

This section describes changes that you see after generating and installing the new agent created with the Tivoli Monitoring Agent Builder.

New files on your system

After you generate and install the new 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\kxxagent.exe`
: Agent binary

`TMAITM6\KxxENV`
: Environment variable settings

`TMAITM6\Kxx.ref`
: Agent provider configuration

`TMAITM6\SQLLIB\kxx.his`
: SQL description of agent attribute information

`TMAITM6\SQLLIB\kxx.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/`*`xx`*`architecture`*`.ver`
: Internal versions and prereqs 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 running an agent with product code 19, here's 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 left of the colon, "sol286" is the architecture in use for this agent. It will be different for different combinations of operating system and computer hardware type. Note that the agent must already be installed in order for this 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 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 has been 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 new 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 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 265 on page 386). 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 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 or not your agent supports multiple instances.

Note: The following screens were taken from a Windows system. UNIX systems have similar screens.

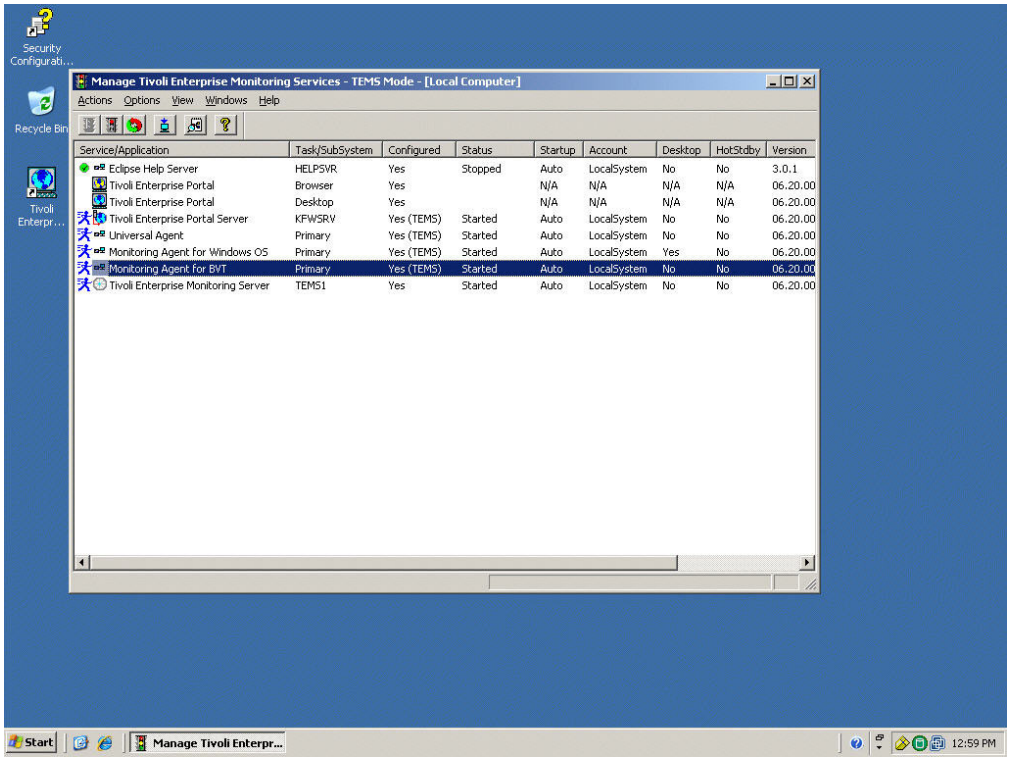


Figure 265. Manage Tivoli Enterprise Monitoring Services window

Changes in the Tivoli Enterprise Portal

After you generate, install, and start the agent, and click the green **Refresh** button, your new agent that you created with the Tivoli Monitoring Agent Builder can be viewed in the Tivoli Enterprise Portal, and 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 using the Agent Builder (Figure 266 on page 387)

  Note: For each navigator item you must define a default query.

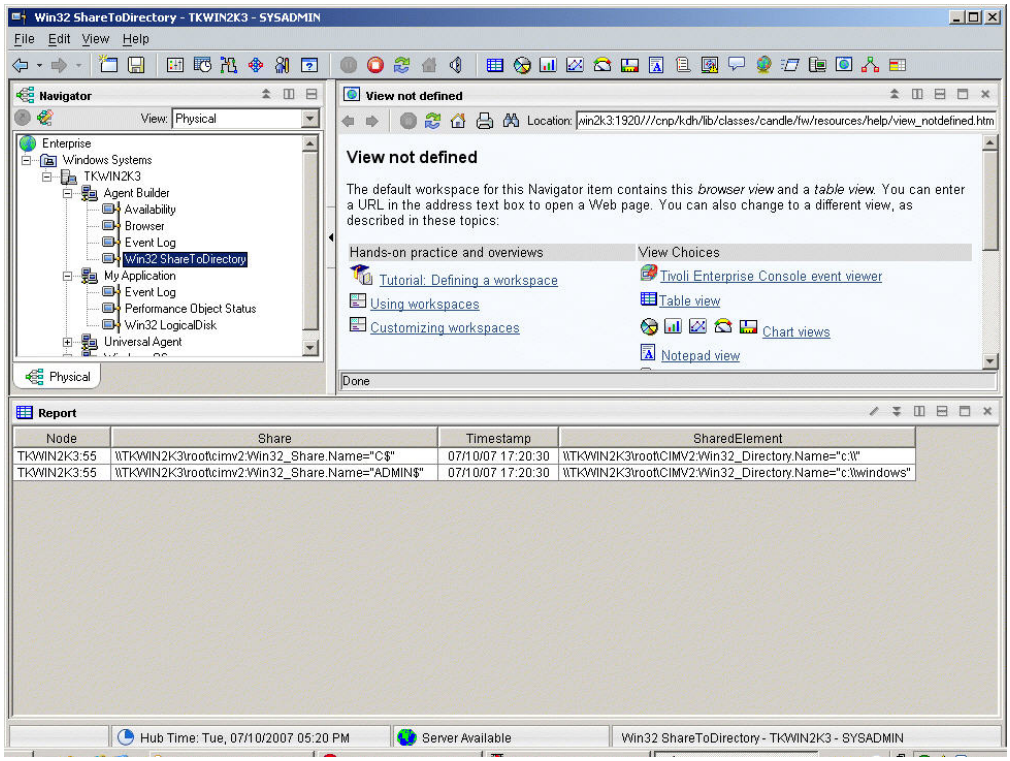


Figure 266. Nodes for attribute groups you defined

- If your agent contains subnodes, an expandable node is present for each subnode that is defined in your agent. The following nodes appear under the expandable node:
  - xxx performance object status, where xxx is the 3-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 additional nodes:
  - An availability node if your agent contains an availability data source (Figure 267 on page 388)

    Note: This node behaves differently depending on the contents of the agent. If the agent only monitors 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 data source and the performance object status data source.

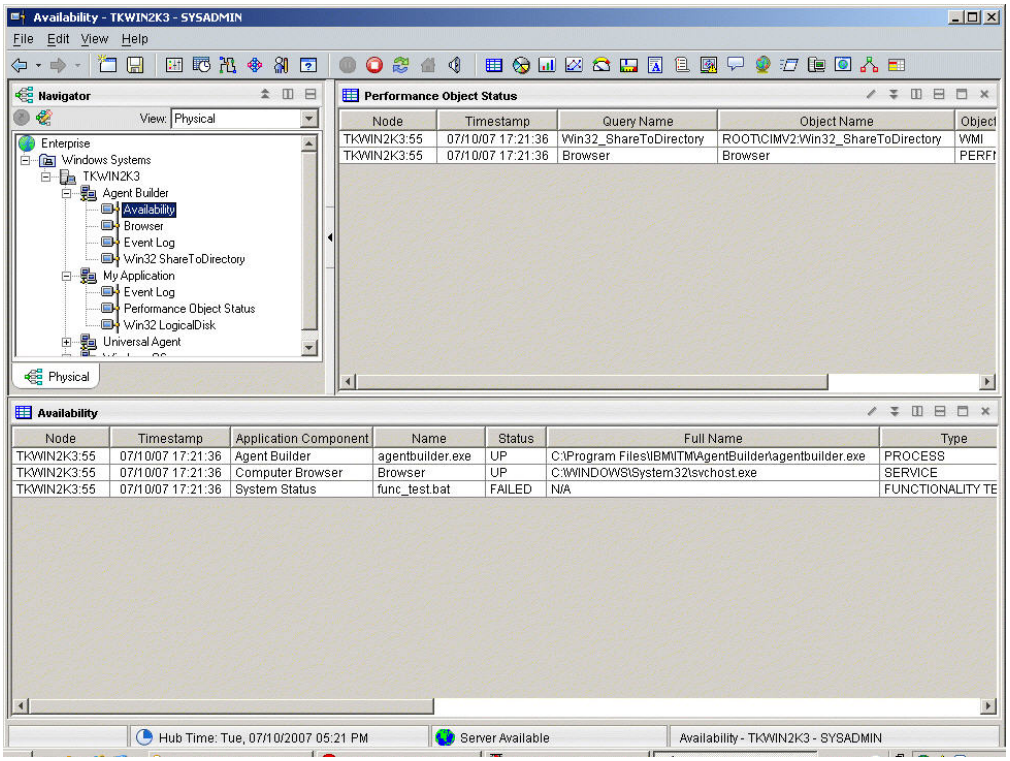


Figure 267. Availability node

– Performance Object Status (if you have specified that you want the agent to perform performance monitoring and not availability) (Figure 268)

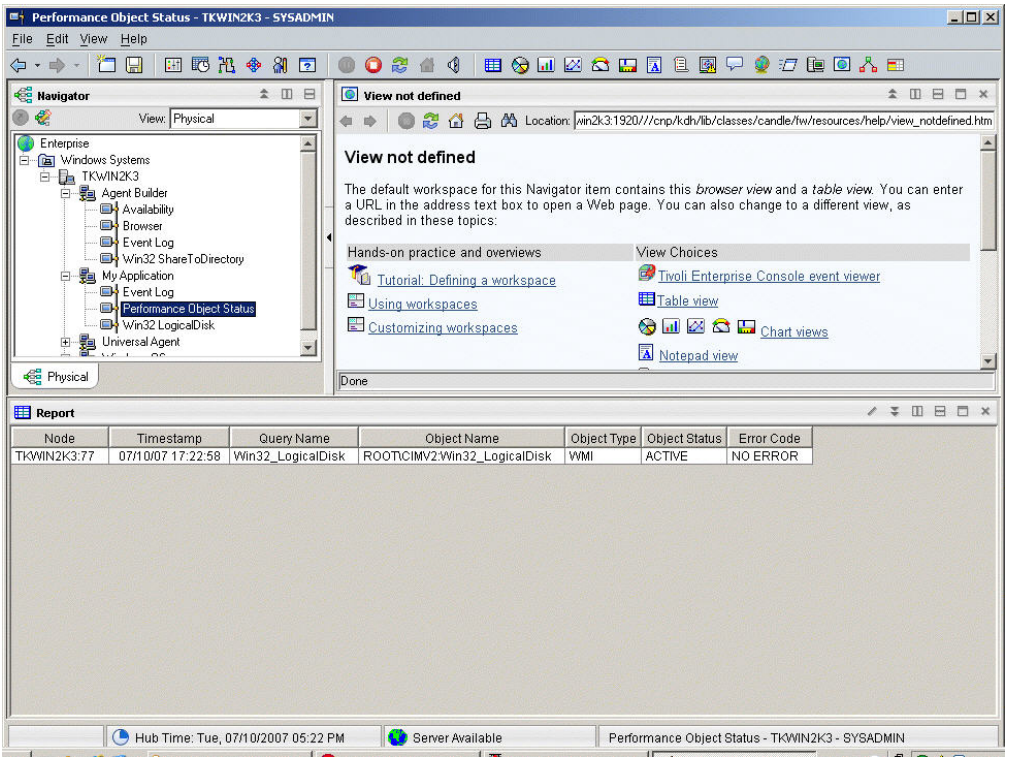


Figure 268. Performance Object Status node

– Event log (if you have specified that filter for your agent) (Figure 269)

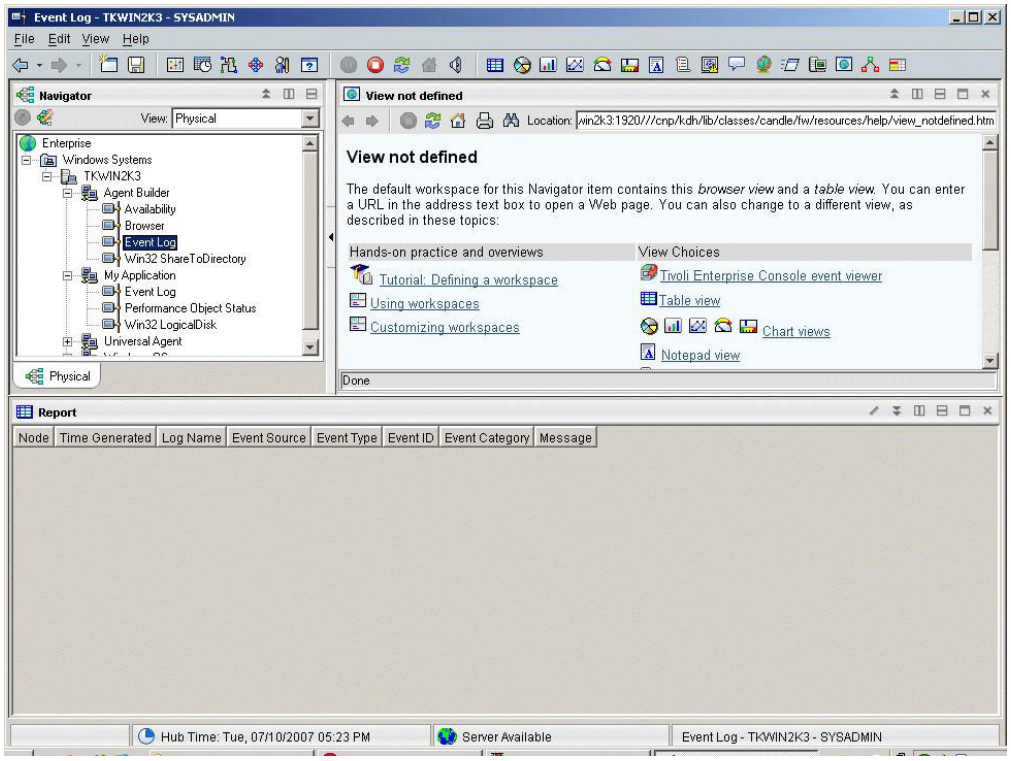


Figure 269. Event log node

See Appendix C, “Attributes reference,” on page 529 for descriptions of the attribute groups and attributes for Agent Builder.

Uninstalling an agent

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 using the Tivoli Enterprise Portal”
- “Removing an agent without using Tivoli Enterprise Portal” on page 390

Removing an agent using the Tivoli Enterprise Portal

To use the Tivoli Enterprise Portal to remove an agent that the Agent Builder generated, perform the following step:

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 Tivoli Enterprise Portal

When you want to remove an agent that the Agent Builder generated from the target system, but you do not have a Tivoli Enterprise Portal, you can use the following commands:

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 found in *ITM_INSTALL*/bin:

```
uninstall.sh [-f] [-i] [-h ITM_INSTALL]
[product platformCode]
```

Clearing the agent from the Tivoli Enterprise Portal after removing the agent

After removing the agent using the Tivoli Enterprise Portal, or not using the Tivoli Enterprise Portal, perform the following steps to clear the agent from the portal:

1. Ensure your Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server are up and running.
2. Log in 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

After you have created, installed, and tested 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 a single installation image for both situations and workspaces, as well as 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 397.

Creating 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, use the online IBM Tivoli Monitoring documentation or 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 639).

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

Build the workspaces in the environment from which they will be used. When building the workspaces, you must change the display settings on your computer to build workspaces at the lowest resolution normally used in your environment environment. Building workspaces at a higher resolution can create views that are too cluttered to be used reasonably at lower 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, perform the following steps:

1. Go to the *ITM_INSTALL*/CNP directory and open the cnp.bat file. If you used the default installation, this is the C:\IBM\ITM\CNP directory. In the cnp.bat file, you must update the **set _CMD= %_JAVA_CMD%** line to include option **-Dcnp.candle.mode="$_KCJ_$"**.

   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.

2. The updated **set _CMD= %_JAVA_CMD%** looks similar to the following:

```
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: This statement needs to be one line, unlike how it is shown here.

3. Open a new Tivoli Enterprise Portal Client, and log in with the "sysadmin" user ID.
4. Now you can 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**. On the right side of the window, select the Workspace Administration Mode check box. If you have correctly done this, *ADMIN MODE* is displayed in the desktop title bar.

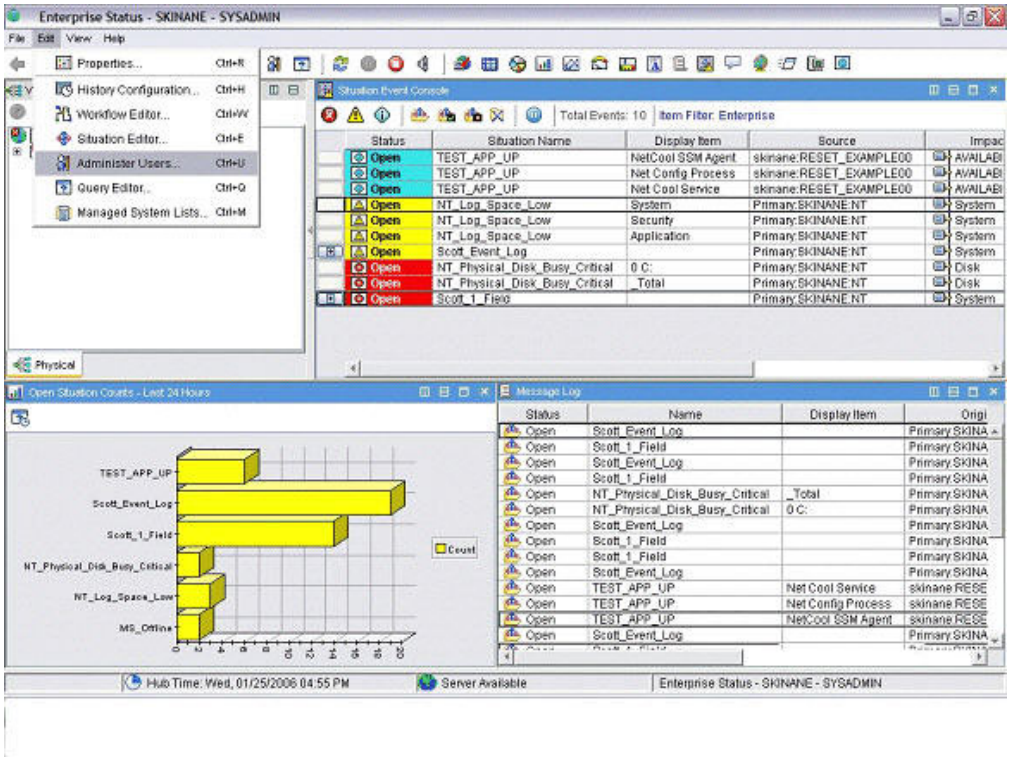


Figure 270. Setting the sysadmin user ID

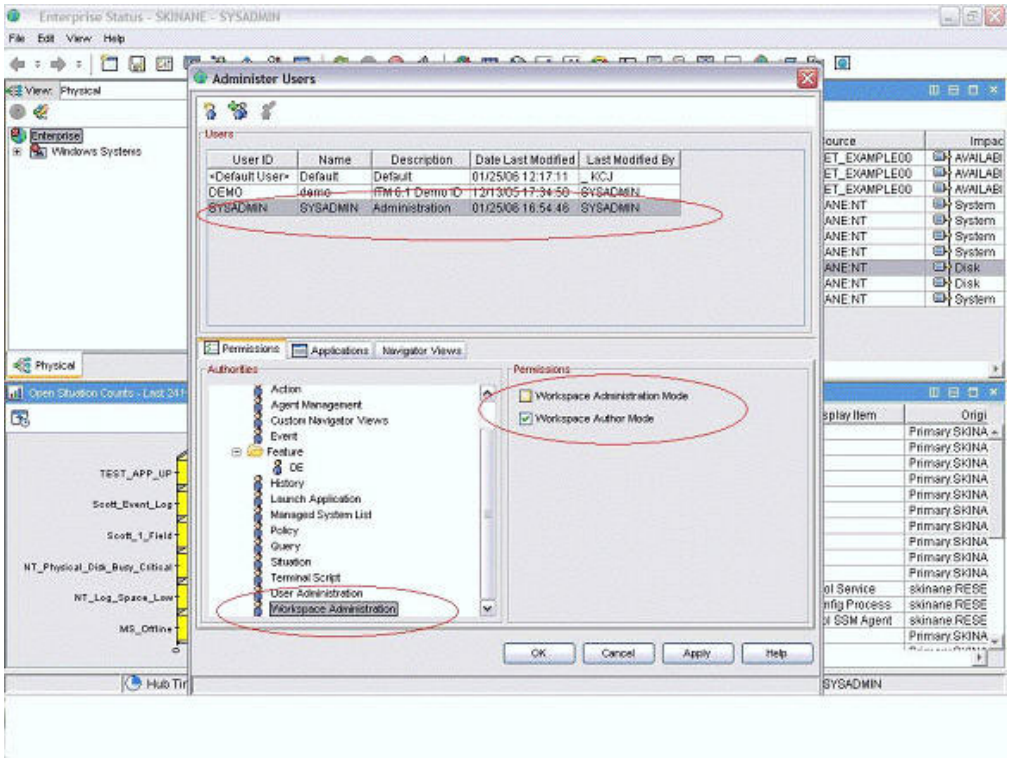


Figure 271. Setting the sysadmin user ID (continued)

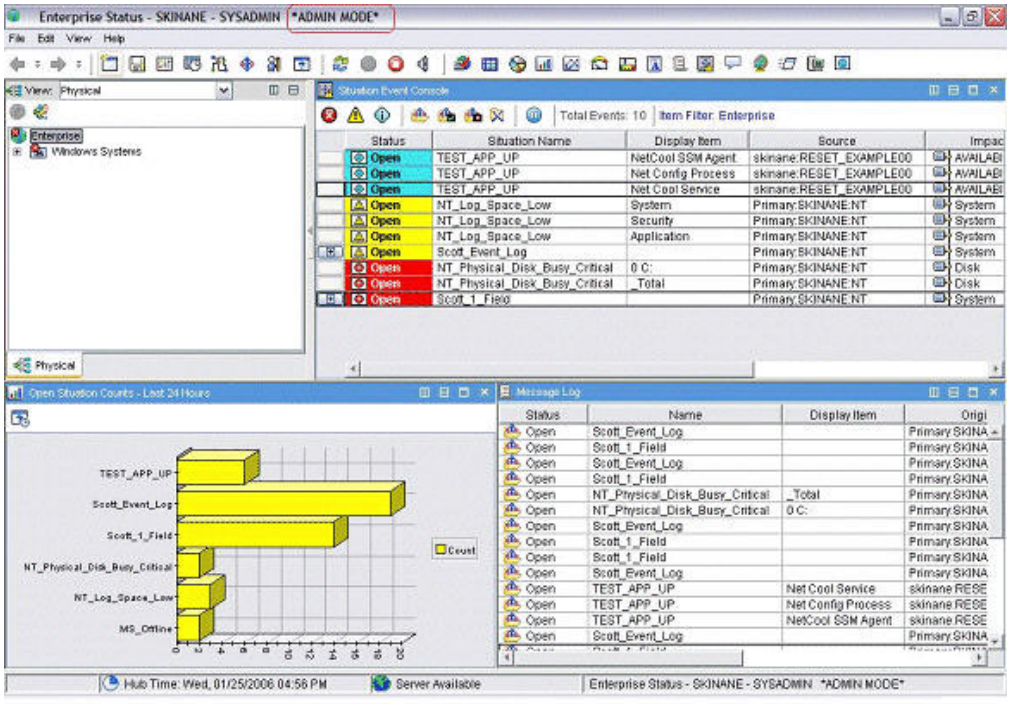


Figure 272. Setting the sysadmin user ID (continued)

5. After you are in "Administrator" mode as depicted in Figure 272, you are now ready to create workspaces for your application. For detailed information about

how to customize and create workspaces, see the online IBM Tivoli Monitoring documentation or 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 navigate to the properties by either viewing a workspace or clicking the icon with the controls on it, or by going to one of the view property pages and then navigating to the workspace level in the properties tree on the left pane. 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:

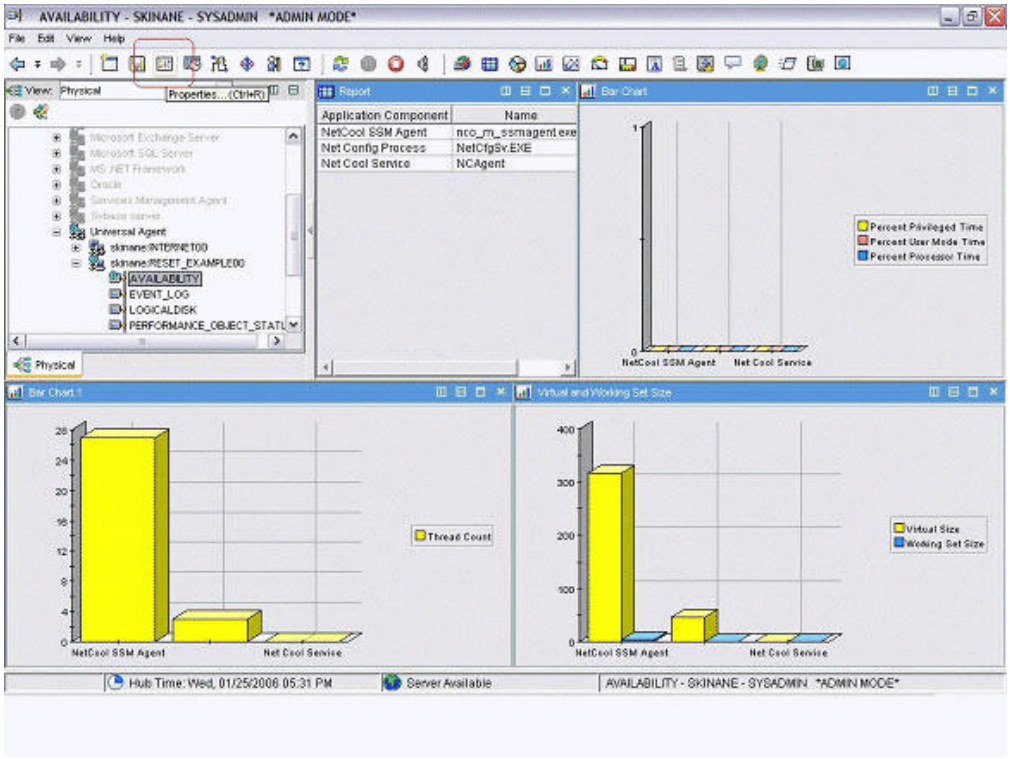


Figure 273. Setting workspace properties

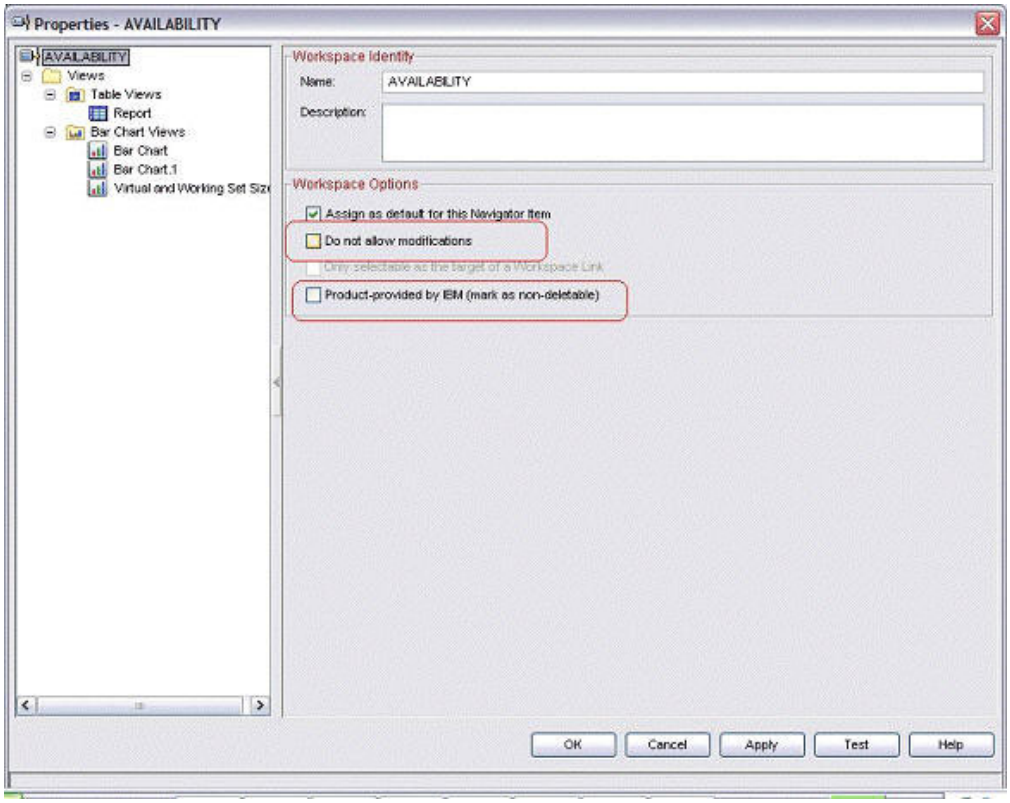


Figure 274. Setting workspace properties (continued)

Chapter 33. Importing application support files

Custom situations, workspaces, Take Action commands, and queries can also be included in the installation package. To have a single installation image for both situations and workspaces, as well as 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.

Exporting and importing server definitions

Definitions associated with an agent can also be included in the installation package. The content of these definitions is different for an agent 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 using the process described in the "Agent Autonomy" chapter in the *IBM Tivoli Monitoring Administrator's Guide*, and the resulting files are copied into the root of the Eclipse project for the agent.

Tivoli Enterprise Monitoring Agents:

After you have created your 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 391. To extract the situations workspaces, Take Action commands, and queries, perform the following steps:

1. From Eclipse, right-click the agent project folder.
2. Select **IBM Tivoli > Import Application Support Files**.

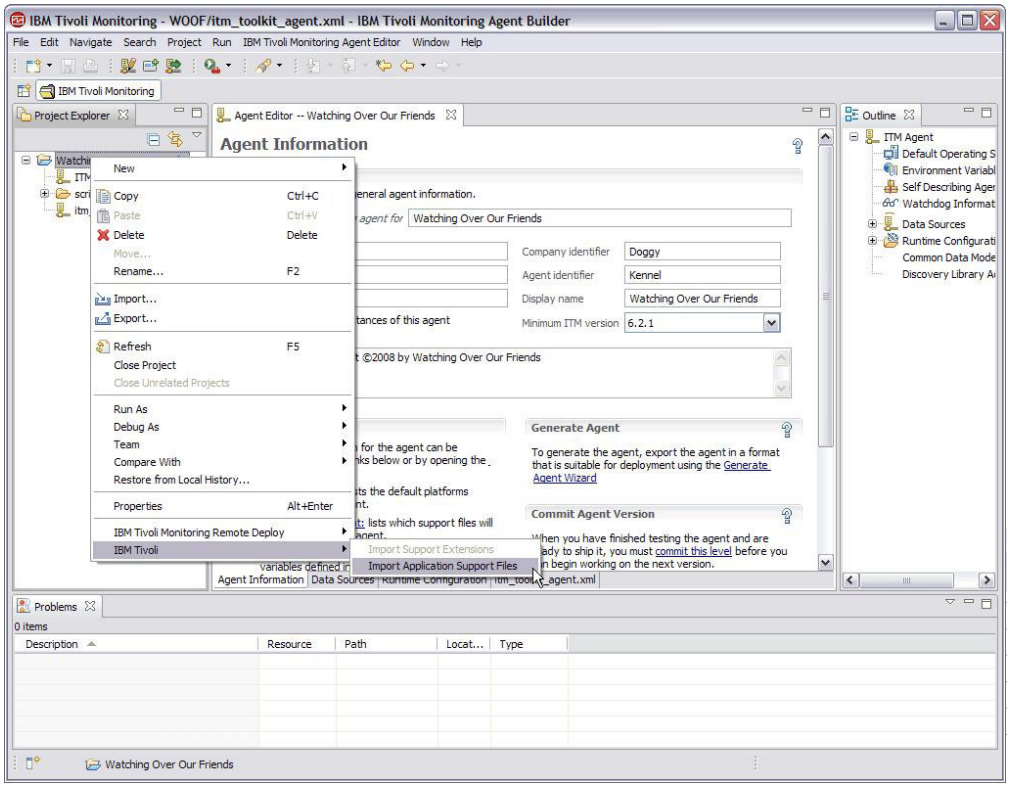


Figure 275. Importing application support files

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 defined for the agent.

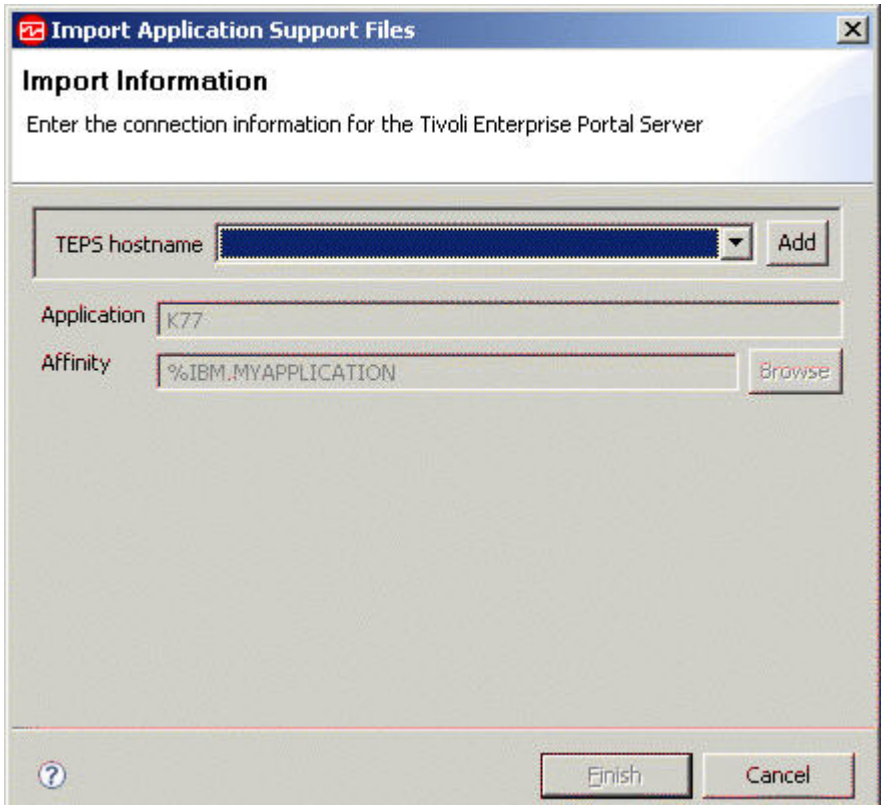


Figure 276. Importing situations

6. Select the situations you want to export from the list and click the button to add them to the selected situations table and click **OK**.

   The import might take a few moments. When the task completes, you will see the SQL files in the appropriate folders in the agent project.

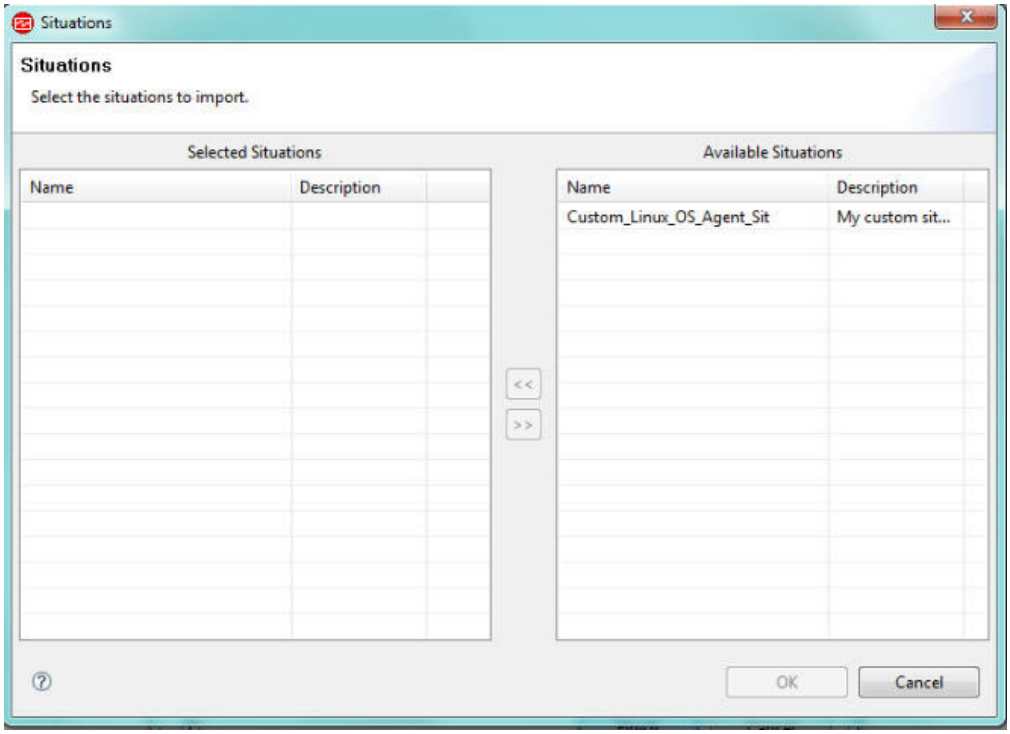


Figure 277. Selecting situations

7. If you defined Take Action commands for your agent, a dialog presents the Take Action commands defined.
8. Select the Take Action commands you want to export from the list and click the button 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.

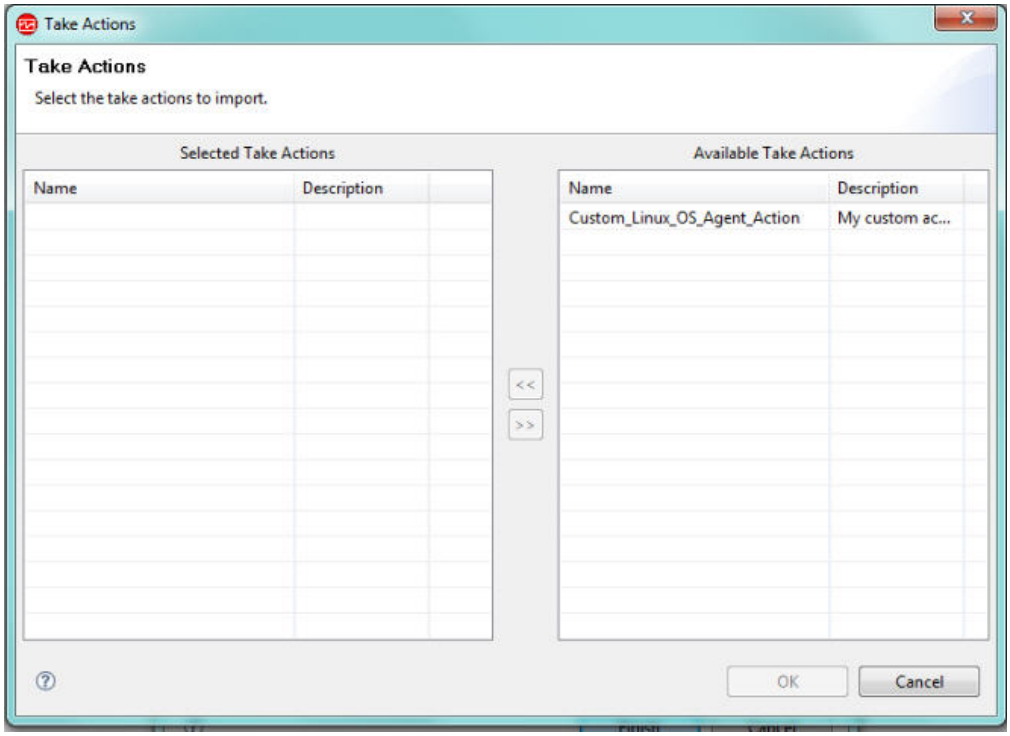


Figure 278. Select Take Action commands

9. If you defined custom queries for your agent, a dialog presents the Take Action commands defined.
10. Select the queries you want to export from the list and click the button to add them to the Selected Queries table and click **OK**.

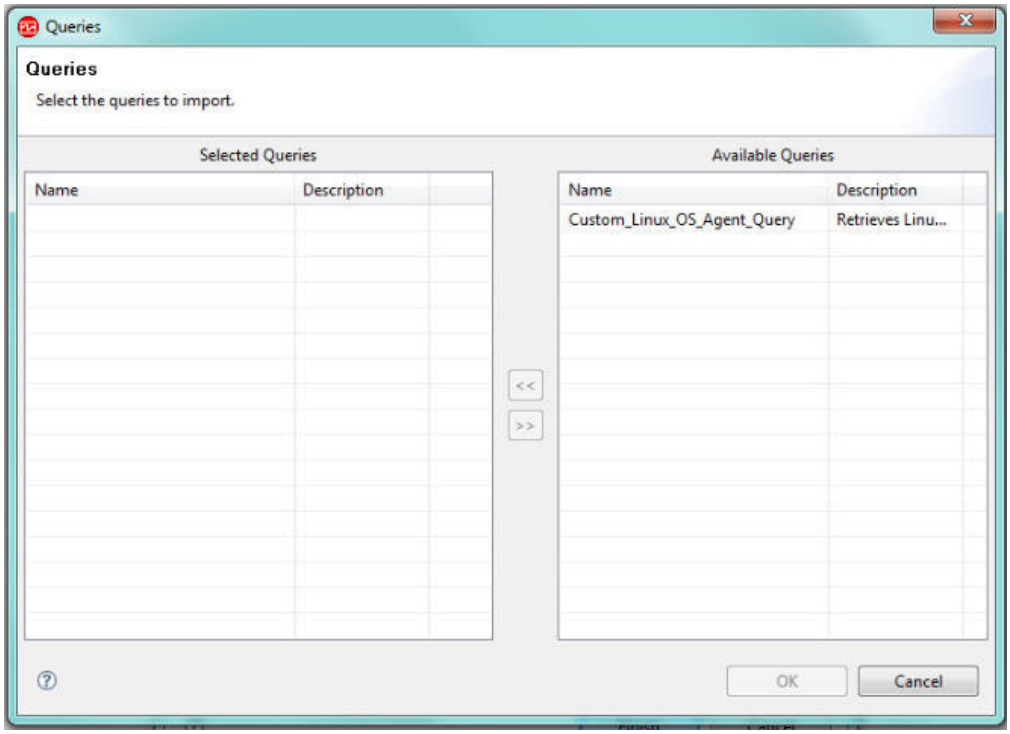


Figure 279. Importing queries

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.

Tivoli System Monitor Agents:

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
- *(Optional)* 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 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.

Create the files using the process 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 will be included in the agent image and installed by the installer. When the agent is installed the installation will do the following:

- Copy the included files into the appropriate locations.
- Any private situations defined in the pc_situations.xml file will run on the agent.

- The trap definitions defined in the pc_trapcnfg.xml will be used to forward traps based on the situations.
- The agent will be 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 will start 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 "SnmpEvents" 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 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 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 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 lower. A situation that monitors for an `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, this is not possible for pure event attribute groups.

Similarly, for historical data, you can configure how frequently sampled data is collected, but when you turn collection on for pure event data, you get each row when it happens.

The data displayed in the Tivoli Enterprise Portal for sampled data is the latest set of collected rows. The data displayed for pure event attribute groups is the contents of a local cache 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

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 280 on page 407.

Whether an event is treated as a duplicate of other events is determined by the key attributes you define in the attribute group. An event is considered a duplicate only if 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.

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**: Use this option to create a summary record for each event with duplicates and each unique event based on the key attributes, and 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 received in the summary interval specified and no duplicate events. This option also sends the summary records.

  – **Event threshold**: Sends an event to Tivoli Monitoring for processing by situations and historical collection when the number of duplicate events 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 5 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, 1 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.

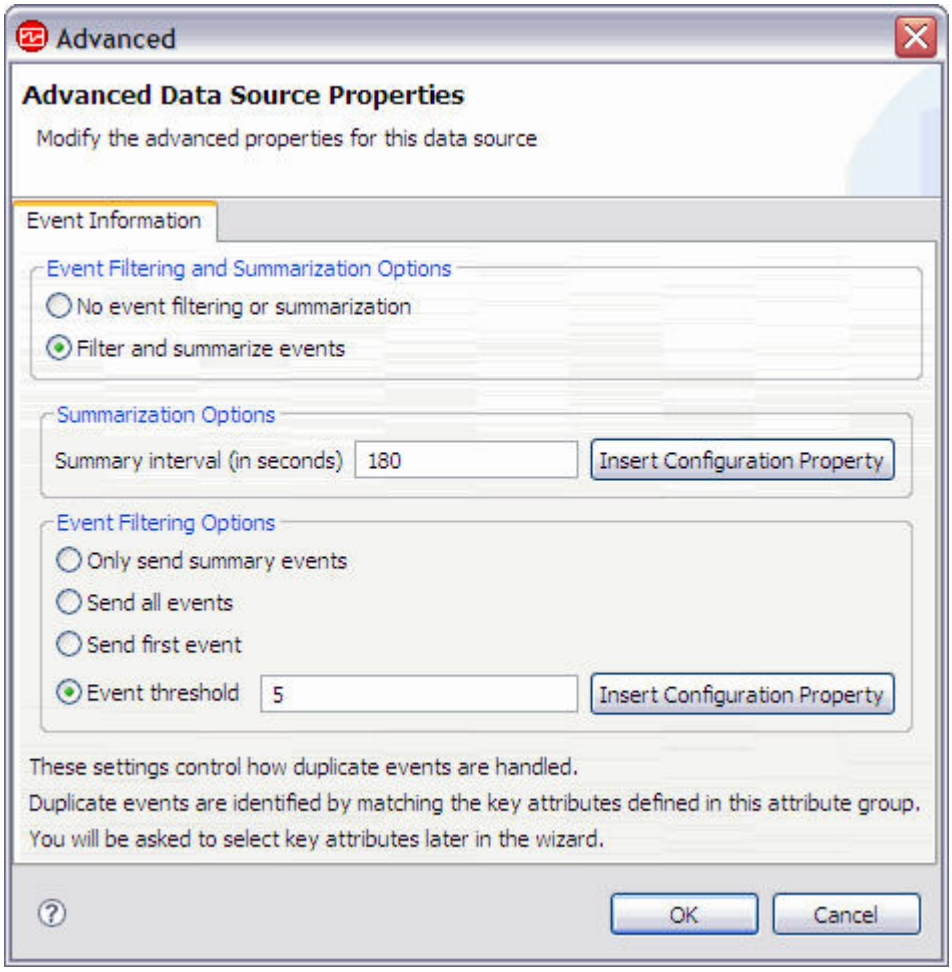


Figure 280. Advanced Log File Attribute Group Information page, Event Information tab

Viewing event filtering and summarization in the Tivoli Enterprise Portal

For pure event attribute groups, the agent maintains a cache of the last events received. By default, this cache is 100 in size. If you have enabled event filtering and summarization for an attribute group, there can be differences between the number of events in the cache and the number of events that are sent to Tivoli Monitoring. There might be more events in the cache because events which occurred but did not reach the designated threshold to be sent to Tivoli Monitoring are visible in the cache. Or there might be fewer events in the cache if you selected the **Send all events** option. With the **Send all events** option, an event is sent each time a duplicate occurs, but only one copy is kept in the cache, with the occurrence count being incremented each time the event occurs. To view the actual events that are sent to 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 this 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 is on top and the real-time (cache) view is displayed on the bottom. The names of the nodes in the Tivoli Enterprise Portal reflect the settings selected. By default, the historical view displays the newest events in the bottom rows, while the default real-time view of the cache displays the newest events in the top rows. 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 281 displays 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 has been scrolled to the bottom and the real time (cache) view to the top.

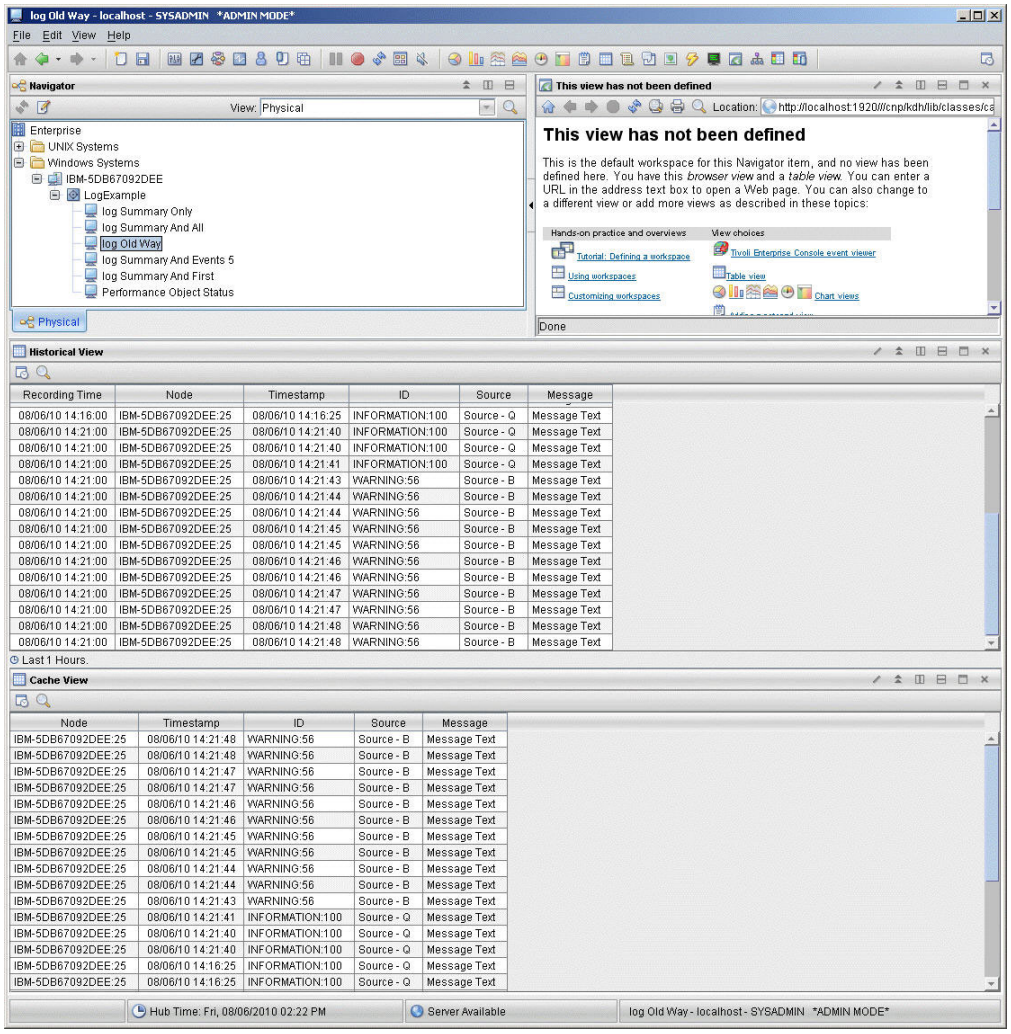


Figure 281. Historical view and cache view when event filtering or summarization is not enabled

Figure 282 on page 409 displays 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.

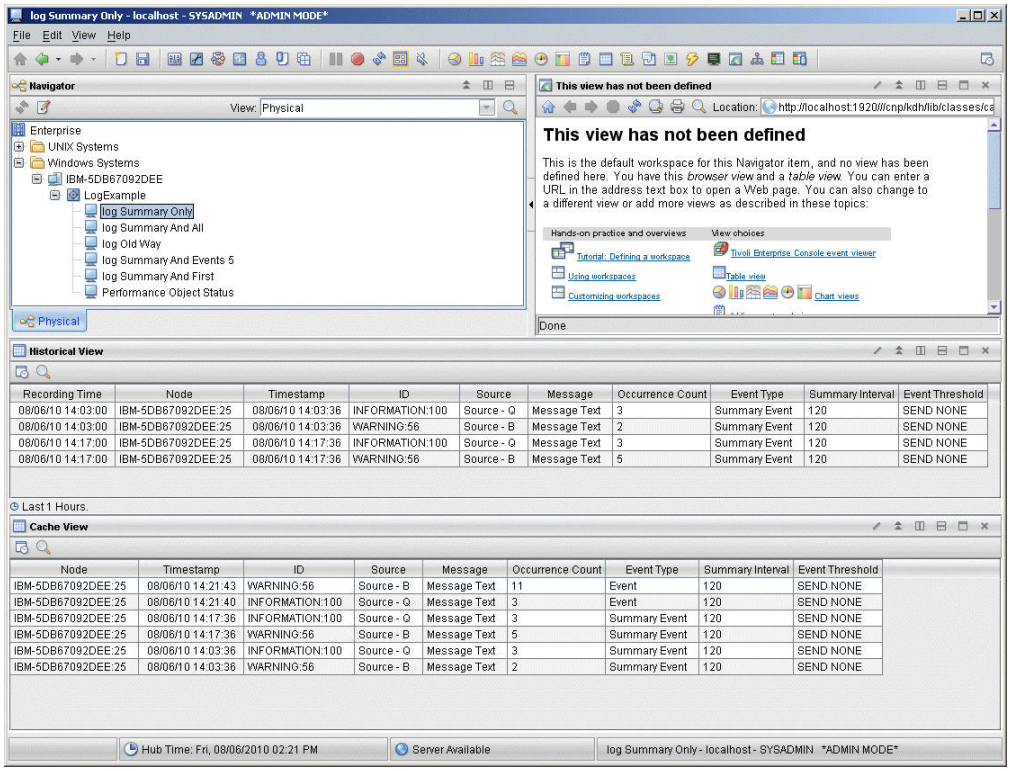


Figure 282. Historical view and cache view when Only send summary events is selected

Figure 283 on page 410 displays the historical view and cache view if you selected the **Send all events** option in the **Event Information** tab. All of the events appear 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. Note also 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.

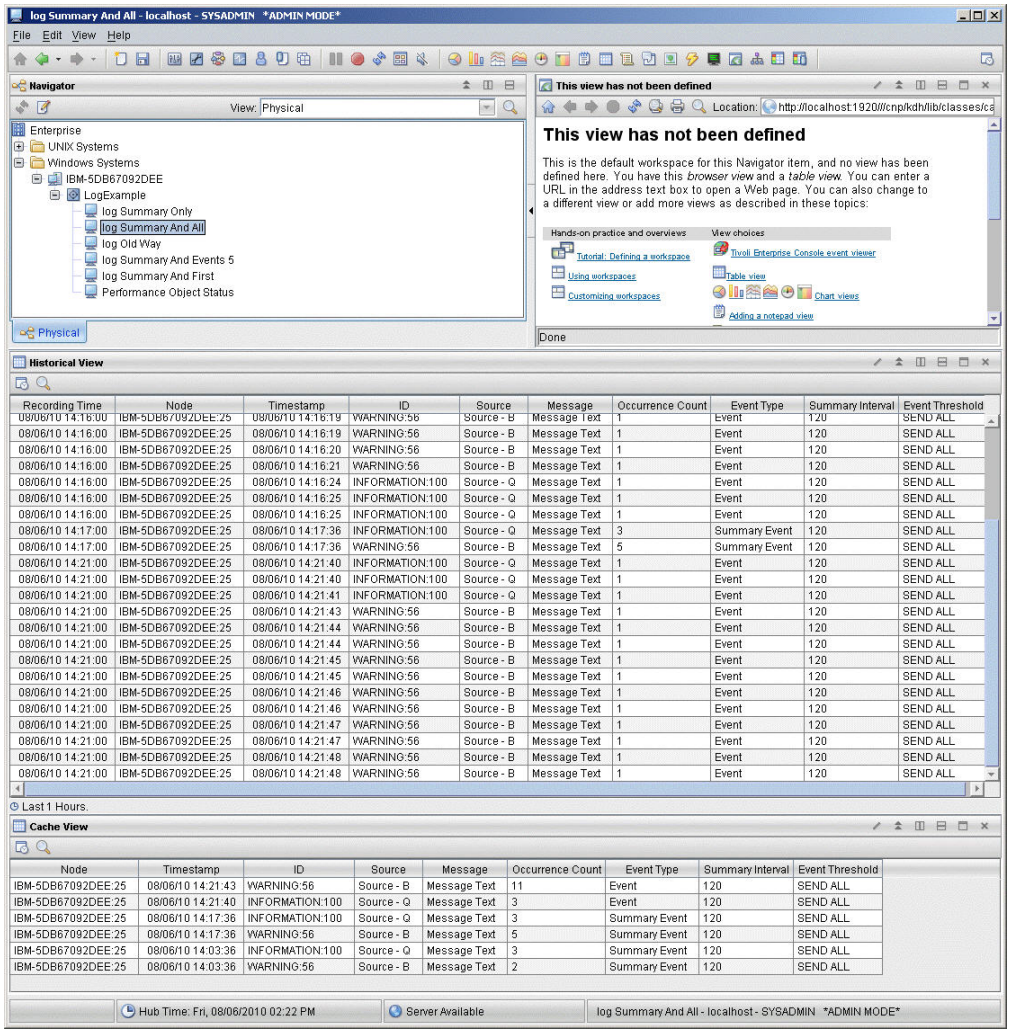


Figure 283. Historical view and cache view when Send all events is selected

Figure 284 on page 411 displays 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 received in the interval and no duplicate events.

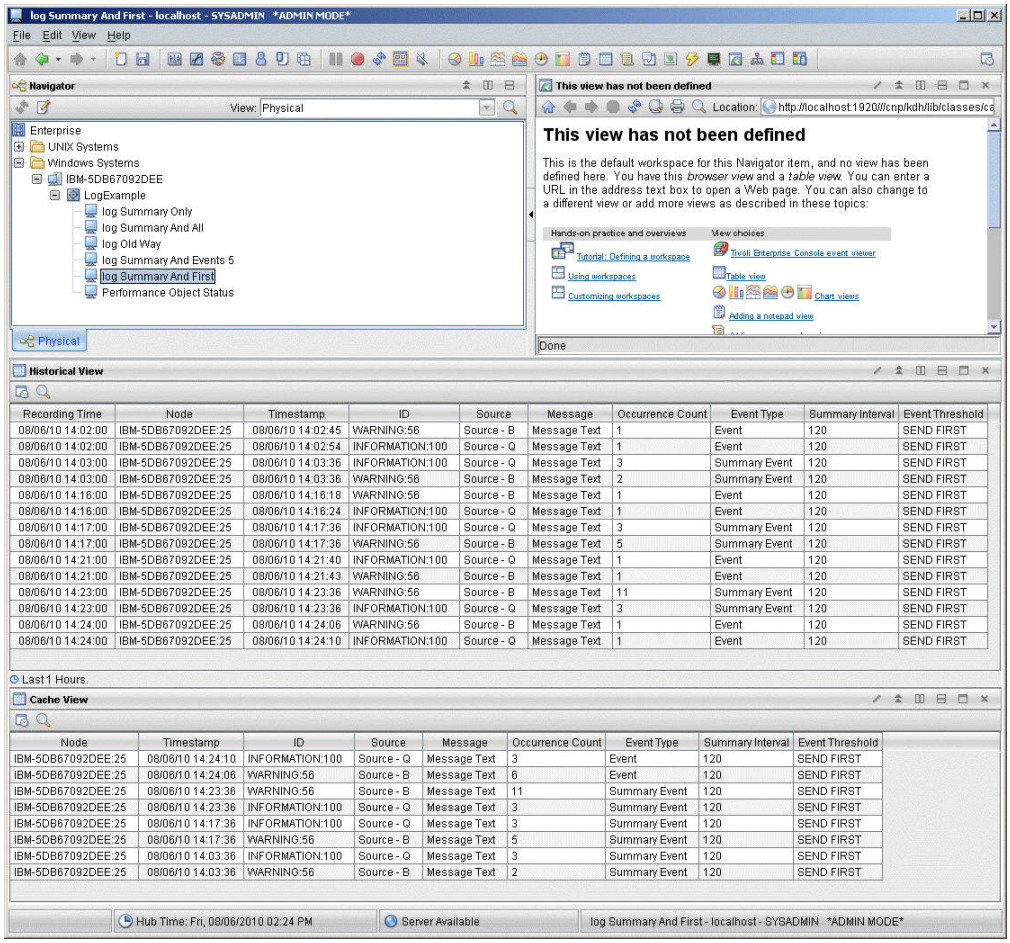


Figure 284. Historical view and cache view when Send first event is selected

Figure 285 on page 412 displays 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 where a threshold of 5 has been specified, the historical view displays an event only when 5 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, 1 event is displayed. If 10 duplicates are received, 2 events are displayed.

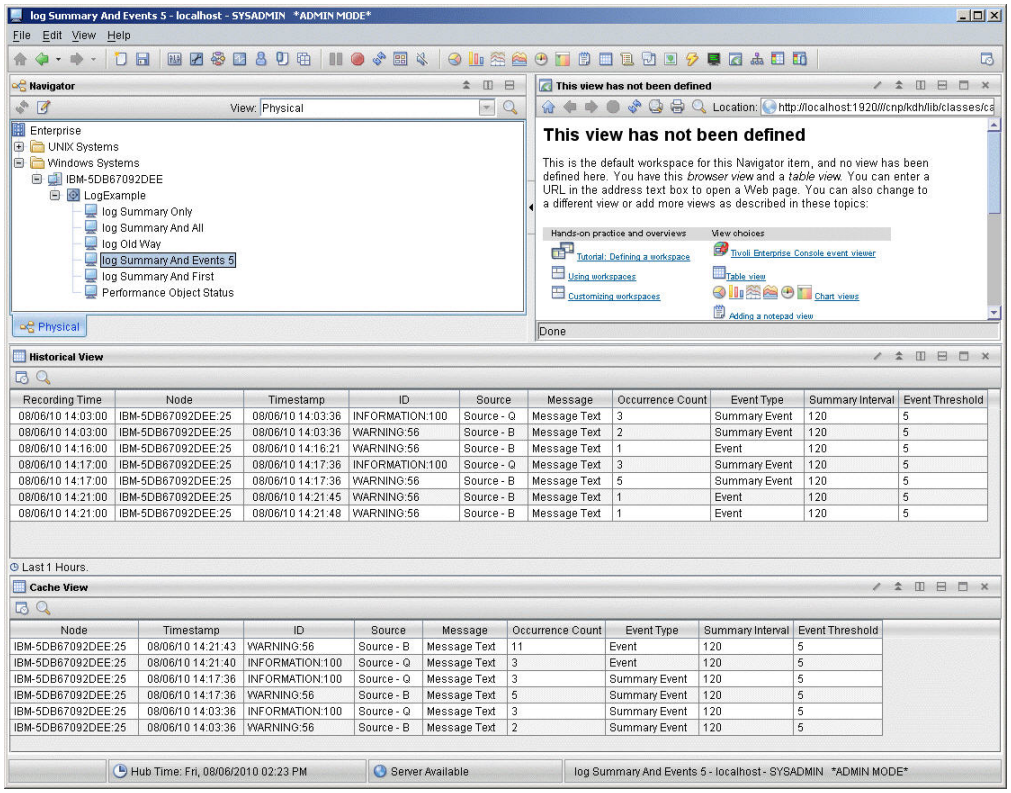


Figure 285. Historical view and cache view when Event threshold is selected

123

1.

2.

3.

Chapter 35. Troubleshooting

This appendix 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 listed in Chapter 3, "Installing the Tivoli Monitoring Agent Builder," on page 7.

Gathering product information for IBM Software Support

Before contacting IBM Software Support, gather the information in Table 44 that relates to the problem.

Table 44. Information to gather before contacting IBM Software Support

Information type	Description
Log files	Collect trace log files from failing systems. Most logs are located in a logs subdirectory on the host computer. See "Agent Builder trace logging" on page 413 for lists of all trace log files and their locations.
Operating system	Operating system version number and patch level
Messages	Messages and other information 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	For an agent that exhibits a problem, send a small non-confidential agent .xml file of a special case or sample test scenario that illustrates the problem.

Agent Builder trace logging

Note: Trace logs contain detailed internal information and require an internal understanding. These logs are for use only by IBM Software Support.

Trace logging is built into the software to assist IBM Software Support in gathering information to determine why a problem is occurring. If you encounter an issue, IBM Software Support might ask you to obtain the trace logs to review the information. This chapter introduces trace logging and explains 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 416.

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 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 you specified when you initially ran the Agent Builder, see ("Starting the Agent Builder" on page 13).

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 "Output window" on page 417.

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 impact 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 level tracing as well.

See "IBM Java Logging Toolkit" on page 416 for more information about setting these levels.

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 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 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 GMT 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

Tivoli Monitoring Agent Builder logging options can be configured after installation. If you are experiencing problems, IBM Software Support might ask you to configure logging and tracing information. This section explains how to configure the product to provide logging data.

Configuration options

You can configure trace logging by using the IBM Java Logging Toolkit to set the trace level for the appropriate component of the Tivoli Monitoring Agent Builder.

IBM Java Logging Toolkit: You can perform all logging configuration in the IBM Java Logging Toolkit. Access the Eclipse preference window by performing the following steps:

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 impact 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.

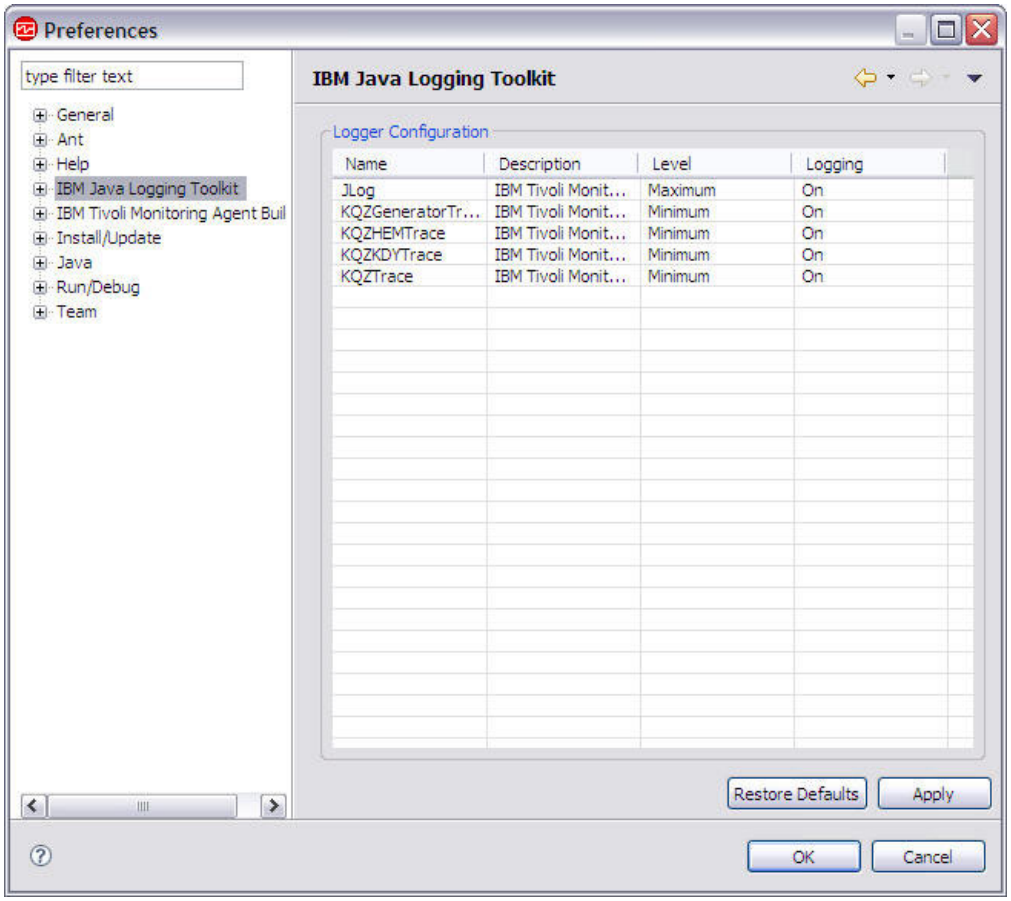


Figure 286. IBM Java Logging Toolkit

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

Output window: Use the appropriate output windows to customize the trace file settings. Access the Eclipse output page by doing the following:

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 will be 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
(*Optional*) Check this box to store the trace logging data.

Enable logging to console
(*Optional*) 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:

1. Close the Tivoli Monitoring Agent Builder.
2. 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.
3. 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 information, performance metrics, status, and other information where a set of data collected at one time 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 being written to a log file. For simplicity, these types of data are grouped together 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, a situation 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, 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 by each situation seeing the same data, you get more consistent results and you minimize 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 performs 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 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 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 relatively 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 additional 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 at once. 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 the Tivoli Enterprise Portal. To avoid "freezing" the agent, the

agent builder automatically defines all subnode agents to run using "scheduled" collection. The agent developer tunes the number of threads and refresh interval to collect the data at a reasonable interval for the type of data being collected. For example, the refresh interval can be once a minute, or once 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 using `CDP_DP_THREAD_POOL_SIZE`. The right 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 using SNMP, it is best to have `CDP_DP_THREAD_POOL_SIZE` similar to the number of remote systems being monitored. By setting the pool size similar to the number of remote systems being monitored, the agent can collect 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 (for example, if the script runs for a long time, or a JDBC query takes a long time) and the target resource can handle it, use enough threads so that long data collections can run in parallel, and probably a few more for the rest of the attribute groups.

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 CPU 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 so 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 additional 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 (or slightly larger if data collection can take a long time), so that 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 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 firing and clearing 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 529. When these attribute groups are used to tune data collection performance, the most interesting 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 CPU usage or time 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 of them 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 423.
- **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. Due to the way this data is collected, even an idle system indicates 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 and 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/Informs, JMX notifications and events from the Java API and socket providers are received asynchronously and forwarded to Tivoli Monitoring immediately. There is no need to tune these collectors. The agent subscribes to receive Windows Event Log entries from the operating system using the Windows Event Log API. If the agent is using the older Event Logging API, it polls the system for new events 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 it will 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 using patterns for the file name or a static name. As the set of files that match 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 needs to 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

Environment variables used for more advanced tuning are defined at the agent level. You set the following variables once 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

Example: 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 you need to 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 and then set CDP_<attribute group name>_REFRESH_INTERVAL for the attribute groups which need to 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 lower value than a refresh interval, unexpected collections may occur.

Environment variables

These environment variables control the behavior of the agent at runtime. They can be built into the agent using the Environment Variables page, for more see "Environment variables" on page 40. On Windows systems, environment variables are defined in the agent *KXX*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 running C Shell. Use an alternative shell if you need to make use of environment variables.

Table 45. 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 upper case, even if the actual attribute group name is not.
CDP_DP_CACHE_TTL	60	Any integer greater than or equal to 1.	Data 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 to happen before timing out and returning cached data, even if the cached data is stale (older than CDP_DP_CACHE_TTL seconds). If this variable is not set, the agent waits as long as the data collection takes, which at times can make the Tivoli Enterprise Portal time out and give up on it. If no thread pool is configured, this variable is ignored and data collection is performed 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. This is logically equivalent to a thread pool size of 1.

Table 45. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_THREAD_POOL_SIZE	15 if subnodes are defined, otherwise not set	Any non-negative integer	The number of threads created to perform background data collections at an interval 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 performing, and helps you 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. If you have not included it in a custom workspace for the agent, 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.
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.	The 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 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 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 launched by a script based attribute group to complete.

Table 45. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_PING_TIMEOUT	30 seconds	Any positive integer greater than or equal to 10.	The number of seconds to wait for the program launched 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 retry sending the SNMP request. The total number of requests 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 time out. Each row in an attribute group is a separate request. Each request can be retried based on the value defined by CDP_SNMP_MAX_RETRIES. This timeout value is the number of seconds for which a response is waited before a retry is attempted. 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.
CDP_DP_HOSTNAME	Name of the first installed network interface	An IP address or hostname	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. This is used by the SNMP data provider.
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_SAMPLE_FACTOR	5	Any non-negative integer	For a Log File data provider, sets the sampling factor when the **Process all records when the file is sampled** option is selected on the Agent Builder. This wait time ensures that patterns that span multiple records are completely written before logging scans 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	Specifies 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 being used.

Table 45. Environment variables (continued)

Environment variable	Default value	Valid values	Description
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 briefly so that previous updates can be processed. Use this environment variable only if the monitored file receives high-volume bursts of new records and there is a concern that some record updates might be lost.
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 launched 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 which to upload the script files provided with the agent. A relative location is relative to the user's home directory. The default of . (period) refers to the user's home directory itself.
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 invoked to delete the uploaded script files 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 being monitored has trouble responding when multiple requests are received at once.
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 consume too much time, there is always a delay of at least CDP_PING_MIN_INTERVAL_DELAY seconds before beginning the pings again. Data is refreshed no more frequently than this setting, but 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 pinging the devices in a device list file, 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 will be sent before pausing for the amount of time specified by the CDP_ICMP_PING_BURST_DELAY variable. A value of 0 disables this function.

Table 45. Environment variables (continued)

Environment variable	Default value	Valid values	Description
CDP_ICMP_PING_BURST_DELAY	10	Any integer greater than or equal to 0	The amount of time in milliseconds to wait after sending a set number of pings as defined by the CDP_ICMP_PING_BURST variable. A value of 0 disables this function.
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. Three ping attempts are made 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. You should 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 should not be set in the environment of the remote system.

Table 45. 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 the CDP_DP_EVENT_LOG_MAX_BACKLOG_ EVENTS variable has not been set to 1 or any integer greater than 1, does not process events generated while the agent is shutdown. 0 is the default. If set to 1, and the CDP_DP_EVENT_LOG_MAX_BACKLOG _EVENTS variable has not been set to an integer greater than 1, processes all events generated while the agent is shutdown. If set to a value greater than 1, and the CDP_DP_EVENT_LOG_MAX_BACKLOG _EVENTS variable has not been set to an integer greater than 1, processes events 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 generated within 300 seconds of the current computer time. Where a value greater than 1 is entered for both the CDP_DP_EVENT_LOG_MAX_BACKLOG_ TIME and the CDP_DP_EVENT_LOG_MAX_BACKLOG_ EVENTS variables, either that time interval of events or that number of events is processed, depending on which is matched first.

Table 45. 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 the CDP_DP_EVENT_LOG_MAX_BACKLOG_ TIME variable has not been set to 1 or any integer greater than 1, does not process events generated while the agent is shutdown. 0 is the default. If set to 1, and the CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME variable has not been set to an integer greater than 1, processes all events generated while the agent is shutdown. If set to a value greater than 1, and the CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME variable has not been set to an integer greater than 1, processes a maximum of that number of events generated while the agent is shutdown. For example, if the value is set to 200, then at startup of the agent the 200 events generated directly before startup are processed. Where a value greater than 1 is entered for both the CDP_DP_EVENT_LOG_MAX_BACKLOG_ EVENTS and CDP_DP_EVENT_LOG_MAX_BACKLOG_ TIME variables, either that number of events or that time interval of events is processed, depending 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 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.
CD_DP_INITIAL_COLLECTION _DELAY	varies	Any positive integer	The number of seconds attribute group data collection can be initially delayed while waiting for the custom provider process to register with the agent.

Understanding informational, warning, and error messages

Messages relay information about system or application performance, 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.

The Tivoli Message Standard requires unique message identification numbers and help content fields for messages issued from a Tivoli component or application. This standard provides a consistent and meaningful way to identify messages across the entire Tivoli product suite.

If you receive a warning or error message, you can do one of the following steps:

- Follow the instructions listed in the Detail window of the message, if this is included in the message.
- Consult the message details listed in this chapter to see what action you can take to correct the problem.

Product messages

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:

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 in this guide.

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.

Agent Builder messages

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.

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 JDBC server from the connection wizard was made successfully.

Operator response: Press "Finish" to save the connection definition for browsing the JDBC server.

KQZ0262E The server connection was not successful: *exception_message***.**

Explanation: The connection to the JDBC 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.

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.

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 executing this query. Check the query for mistakes. The database returned the following problem information: *exception_message***.**

Explanation: The database server returned an error when it attempted to execute the query.

Operator response: Modify the query and attempt the test again. The message returned by the database 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.

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.

KQZ7007I The server connection was successful.

Explanation: The connection to the MBean server from the connection wizard was made successfully.

Operator response: Press "Finish" to save the connection definition for browsing MBeans.

KQZ7008E The server connection was not successful: *exception_message***.**

Explanation: The connection to the MBean 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.

KQZ7009E No connection selected.

Explanation: A server connection must be selected in order to connect.

Operator response: Select the server to want to connect to, or press "Edit" to define a new connection. Then press "Connect" to the server you have selected.

KQZ7010W An MBean was not selected. Continuing will create an attribute group containing no attributes.

Explanation: An MBean must be selected to determine the attributes that will be created.

Operator response: Select "OK" to continue without selecting an MBean. 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.

KQZ8183E **Expression "*literal_token*" is not a valid literal value.**

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.

Agent trace logging

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 located in a `logs` subdirectory on the host computer. See the following sections to learn how to configure and use trace logging:

- "Principal trace log files" on page 477
- "Example for using trace logs" on page 479
- "Setting RAS trace parameters" on page 480

Note: The documentation refers to the RAS facility in IBM Tivoli Monitoring as "RAS1".

IBM Software Support uses the information captured by trace logging to trace a problem to its source or to determine why an error occurred. Although the default configuration for trace logging is to be enabled, it can be disabled.

Overview of log file management

Table 46 on page 478 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 2 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 8-character hexadecimal time stamp representing the time at which the program started.
- *nn* is a rolling log suffix. See "Examples of trace logging" on page 477 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 running 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 running on balayne, with product code 13:

```
inst1 BALAYNE_13_k13agent_46c49270-01.log
```

Examples of trace logging

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 as follows:

```
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 reach 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 46 on page 478 contains locations, file names, and descriptions of trace logs that can help determine the source of problems with agents.

Table 46. 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 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 located in the following path on Windows: *install_dir*\InstallITM. On UNIX, the file is located 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 Portal 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 located in the following path on Windows:*install_dir*\InstallITM path. On UNIX, the file is located 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 **Notes:** 1. The files are located 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 46. 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 K*XX*_*JMX*|*JDBC*|*JAVA*_trace.log • For a multiple instance agent K*XX*_*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 located 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. Ranges from 1-5, by default, though the first is always retained, because it includes configuration parameters.		

See the *IBM Tivoli Monitoring Installation and Setup Guide* for more information about the complete set of trace logs that are maintained on the monitoring server.

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 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 the data source starting 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 alternate method to view trace logs:

1. In the Windows **Start** menu, choose **Program Files > IBM Tivoli Monitoring > Manage Tivoli Enterprise Monitoring Services**. The Manage Tivoli Enterprise Monitoring Services window is displayed.
2. Right-click a component and select **Advanced > View Trace Log** in the pop-up menu. For example, if you want to view the trace log of an agent, right-click the name of the that 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 ras1log tool, see the *IBM Tivoli Monitoring Troubleshooting Guide*.

Setting RAS trace parameters

Objective

Pinpoint a problem by setting detailed tracing of individual components of the monitoring agent and modules.

Background Information

The Agent Builder agents and agent runtime use RAS1 tracing and generate the logs described in Table 46 on page 478. The default RAS1 trace level is ERROR.

Before you begin

See "Overview of log file management" on page 476 to ensure that you understand log rolling and can reference the correct log files when you manage log file generation.

After you finish

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 46 on page 478 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 TEMS. 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 tracing setting can generate a large amount of trace logging. Use them only temporarily, while you are troubleshooting problems. Otherwise, the logs can occupy excessive amounts of hard disk space.

Procedure

Specify RAS1 trace options in the *agent_name*`ENV` file on Windows systems, and in the *agentname*`.ini` file on UNIX systems.

Use the following procedure to manually edit the configuration file to set trace logging:

1. Open the trace options file. This file is located in the following folder:
   - **On Windows:** *install_dir*`\tmaitm6\`*productcode*`ENV`

- **On UNIX:** *install_dir*\config

2. Edit the line that begins with KBB_RAS1= to set trace logging preferences. For example, if you want detailed trace logging, 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 startups of a given program. After this value is exceeded, the oldest log files are discarded. Default value is 9.
   - LIMIT: the maximum size, in megabytes (MB) of a 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.

(*Windows only*) Alternate method to edit trace logging 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 pull-down 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 startups 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.

   Note: The **KDC_DEBUG** setting and the Maximum error tracing setting can generate a large amount of trace logging. 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 reporting a restart of the monitoring agent so that your changes take effect.

Problem classification

This section provides 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 47 on page 484 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 on help links within the Agent Builder
- 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
- [4]
- JMX browser fails to connect to WebLogic 10.x

Table 47. Problems and solutions for installation and use of the Agent Builder

Problem	Solution
Maximum data size exceeded When saving 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 cause unnecessary overhead 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.

Receive error when installing 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 native 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 have installed Cygwin (a set of UNIX tools for Windows), and it 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 clicking **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, and it showed me an odd set of processes and it automatically selected UNIX operating systems for my process instead of Windows.	You selected the wrong agent. The Demo CD contains a lot of agents which would not normally be found together on a system. Stop the other OS agents and browse using the real Monitoring Agent for Windows, or browse using a system that has a normal Tivoli Monitoring installation.

4. Using the Parse function on the Parse Log window, returns incorrect results, such as no rows appear

Table 47. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
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 install does not support the -console option. It does, however, support the -silent option. See "Silent installation" on page 9 for more information.
No help is displayed when clicking on help links within the Agent Builder	On some AIX and Linux systems, Eclipse is unable to determine the default system browser. To correct this, from within the Agent Builder, perform the following steps: 1. Select **Window > Preferences**. 2. Under General, select **Web Browser**. 3. Ensure "Use external Web browser" is selected. 4. Click **New...**. 5. Enter a descriptive name for the browser in the Name field. 6. For location, enter the full path to the browser executable. 7. Click **OK**. 8. Ensure the browser you previously defined is checked and click **OK**. Now, when you click help links in the Agent Builder, the browser specified launches to display the content.

Problems launching the Agent Builder on a non-English system
 	
When running the Agent Builder on a non-English system, the JVM might have problems determining the locale of the system. This prevents the Agent Builder from launching.
	To get around this, edit *install_location*/agentbuilder.ini and add the following two lines:
	`-Dfile.encoding` *LOCALE_PREFIX*
	
	Where *LOCALE_PREFIX* is the prefix for the locale the system is running (i.e. FR for French). Also, note that the -vm line and the line immediately after must remain the first two lines in the file, so they should 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 while trying to browse WMI or services remotely.	This is not supported. You will 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 drop-down list instead of entering the host/ip address.

Table 47. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
Key attribute missing When saving 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 key(s). If the data source can only return one row of data, set the **Produces a single data row** radio button for the data source. This has 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 When changing a command return code or a script data provider executable, newly defined return codes do not show up.	If you make a change to the executable, then you must copy the new executable into the agent's project directory under the scripts subdirectory. This will ensure that the new executable is picked up when you regenerate the agent.
Error when attempting to launch Eclipse: Agent Builder fails to start and displays an error pointing to a log file in the configuration directory. The log file contains: `No application id has been Found.`	Uninstall Agent Builder, and re-install it in a fresh directory.
Browsing processes or services on a remote system do not work	Both the Tivoli Enterprise Monitoring Server and the Tivoli Enterprise Portal Server are required to be running as well as the OS agent if you are going to use the browse remote system function. Restarting the Tivoli Enterprise Monitoring Server or starting it after the OS agent will have the OS agent offline until its heartbeat is done with the Tivoli Enterprise Monitoring Server or the OS agent is 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 invoked 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 47. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution

Parsing MIBs
 	
When loading my MIB into the Agent Builder, it did not parse. All I can see is something such as the following:
 	
`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.NullPointerException`
`at 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 so on
 	
`MibParser.java:1198` and `MibParser.java:1081` might point to other areas of the parser.
	To get the MIB to load:
	
	The MIB parser used by the Agent Builder uses the grammar 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:
	1. On the top menu bar, click **Window** > **Preferences**.
	2. Click **IBM Tivoli Monitoring Agent Generator** in the menu on the left.
	3. Select one of the following options in the **MIB Parsing Options** list:
	
	Allow types to start with lower case 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) illegal 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 these MIBs and errors are discovered, IBM, customers, and business partners will add information to the A&BSM Technical Exchange Wiki (http://www.ibm.com/developerworks/wikis/display/aabsmenbl/Home) indicating what MIBs are discovered that have errors and what the corrections are that need to be imported into the Agent Builder. See 487.
Agent Builder version on UNIX The Agent Builder is installed on a UNIX system, and you cannot launch 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 47. 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 in the order in which fields are encountered in the line 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 both the new and old log file formats in order to correctly support current and old versions of the application. The best way to handle this in an agent is to add a new 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 so that only the old or new attribute group is actually used in a particular 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 creating a problem that occurs if you try to 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 using the name you want to have displayed 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 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 using 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 need to collect.	See Negative or wrong attribute value for a technique to use derived attributes to display large numeric values using signed 32-bit attributes.
Parse Log window characters corrupted Characters 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 47. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
Browse button not active when adding data source While adding a data source, the **Browse** button 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 creating WMI, Perfmon, or Windows Event Log data sources. The Windows APIs must be available on the Agent Builder system in order to Browse these data sources. You can still define these types of data sources while running 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 has been applied to the IBM Pegasus CIM Server.
	This issue has been resolved in IBM Pegasus CIM Server V2.6.1.35 available at https://www.ibm.com/services/forms/preLogin.do?lang=en_US&source=aixpegcim
	
	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 the IBM Pegasus CIM Server to 2.6.1.35 or later
	
	For current details, refer to AIX Information - Common Information Model Guide at http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.cim/doc/cim/About.htm?tocNode=int_187407 .
	
	—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 47. 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)
	Re-apply 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 has been 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 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 47. 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 need 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 have created 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 install 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 that, 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 below**. 3. Select the project containing Agent A and click **OK**. The agent will be re-validated and the error marker will be removed.

Table 47. Problems and solutions for installation and use of the Agent Builder (continued)

Problem	Solution
Agent Builder script argument parameterization does not work. When creating 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 generating the agent and installing locally and specifying a password with special characters, the login to the monitoring server fails with error message: KQZ0208E The specified username 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 iFix to the Agent Builder, when you uninstall the Agent Builder, the iFix is not removed. When uninstalling, an installer message informs you that it was unable to remove all directories.	Remove the iFix directories manually before reinstalling 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 launching Agent Builder, the buttons are greyed out on the workspace screen and you are unable to perform any action.	This is an Eclipse issue with the Xming and Cygwin tools and not an issue with the Agent Builder. There is currently no fix available.
Using the Parse function on the Parse Log window, returns incorrect results, such as no rows appear When using the Parse function on the Parse Log window, incorrect results are returned, such as no rows appear.	The Test Log File parser only reads the last 1,000 lines of the log file in tail mode. Make sure the data you are trying to test for occurs in the last 1000 lines of your sample log file.

Table 47. 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 show that a CORBA.MARSHAL error occurred.	Copy the wlclient.jar and the wljmxclient.jar files from your WebLogic server installation into the Agent Builder installation's jre/jre/lib/ext directory and restart Agent Builder. This loads the WebLogic jars in the system class loader which makes sure the correct classes are loaded to create the JMX connection. After you have created your attribute groups using the JMX browser, you must remove these two jars 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 may not have detected and processed the file yet. Ping may report all zeros since the ping has not been performed yet Click **Collect Data** a second time If this 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 collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 534 You can attempt further debugging by looking at the test log files, see "Debugging" on page 376.

Troubleshooting: Agents

This section provides 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
- Agent support files still exist after uninstalling
- JDBC connections are failing to find my JDBC driver
- 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
- 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

Table 48. Problems and solutions for agents

Problem	Solution
Program fails No data for a script provider or a command return code is unusable. You will 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 execute 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 48. Problems and solutions for agents (continued)

Problem	Solution
Negative or wrong attribute value I created an agent that includes a number that should be a large positive number, but I see a negative number or a number that is completely wrong.	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 you are trying to display has overflowed the 32-bit signed number. In many cases, the values are traced to the log file and replaced with enumerations indicating that 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. To do this: 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 deselecting **Display attribute in the Tivoli Enterprise Portal**. Hide the original attribute if it is likely that it will 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.

Table 48. Problems and solutions for agents (continued)

Problem	Solution
JMX Notifications I do not have the ability to create a data source to receive notifications.	When using the browser, take note of the MBean information displayed at the bottom of the browser panel for the MBean you are working with. If the MBean contains items in the Attributes tab, then you will get an attribute group containing those metrics. If the MBean contains items in the Notifications tab, you will get an event attribute group containing a standard set of metrics for a Notification object. If both tabs contain values, then you get attribute groups for each. Without there being notifications 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, just manually type in the Object Name pattern and click **Finish**. This creates the same event attribute group that you would have gotten 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.
Locally configuring an agent fails on Windows An Agent Builder created agent containing configuration properties requires Tivoli Monitoring to install the Java Runtime Environment. If the Monitoring Agent for Windows OS was installed using the tacmd createnode command to create a Tivoli Monitoring v6.2 Fix Pack 1: Windows OS Agent, then the Java Runtime Environment will not be installed. On this system, you cannot locally configure any Agent Builder created agents that contain configuration properties.	The agent can still be configured remotely using the Tivoli Enterprise Portal. The following solutions can be used to install the Java Runtime needed by Tivoli Monitoring: • Install an IBM provided agent that requires the Java runtime. • Install the Monitoring Agent for Windows OS locally instead of using the tacmd createnode command. • Install the Tivoli Monitoring V6.2: Windows OS Agent 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 containing JMX, SNMP, or JDBC attribute groups.	You must have Tivoli Monitoring v6.2 Fix Pack 1 to configure an agent containing JMX, SNMP, or JDBC attribute groups using the **itmcmd config** (or CandleConfig) commands. You can use the GUI to configure the agent if upgrading 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 will be different for your server. If the JBoss server is running, it must be restarted after copying this file.
JMX browser connecting to WebSphere 6.1 The JMX MBean browser cannot connect to WebSphere 6.1 with security enabled, using the SOAP connector protocol.	The browser can collect the MBean names using the rmi connector protocol.

Table 48. Problems and solutions for agents (continued)

Problem	Solution
Error code for attribute group, but data 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 could not be collected. This 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 could not locate a certain class or the attribute object could not be serialized. When you see this error, the attributes have to be collected individually from the MBean Server instead of collecting all attributes in one remote call. This can significantly impact the performance of the agent.
JMX data provider connecting to WebSphere 7.0 JMX data provider fails to connect to WebSphere 7.0 with security enabled, using the RMI connector.	This is an issue with the WebSphere Application Server. Upgrade to WebSphere 7.0.0.1 or later to resolve this connection issue.
JMX data provider connecting to Oracle WebLogic Server on AIX The JMX data provider fails to connect to the Oracle WebLogic Server when running on AIX.	This connection problem is a limitation 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 looking 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 will display 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 will now be displayed in the title bar. Refer to “Creating workspaces” on page 391 for more information.
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 will not be 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 created by Agent Builder are not translated when shown on the Tivoli Enterprise Portal.	The Agent Builder does not build language packs, so the text that is displayed in the Tivoli Enterprise Portal is in the language you use when building the agent.

Table 48. Problems and solutions for agents (continued)

Problem	Solution

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 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 will not be one you are actually monitoring. If all of the data in your Availability table is filled in, then this does not represent a problem. You can verify this by checking the PID printed in the trace against the PIDs for the processes you are monitoring.

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 a trace entries like these: (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 filled in 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 could be parsed from the log record, but others could 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 The situations should be true (or my node should display the little red circle indicating that the situations are true), but they are not displayed as true. I checked and the data has exceeded the threshold. Why can't I 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 A popup error on the Tivoli Enterprise Portal says that the "C:\IBM\ITM\InstallITM\Batch\kincli -startagent -akxx" command failed.	After installing 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 system's PATH to include some DLLs that need to be found for those functions to work, but the services do not pick up those changes until the machine reboots.

Table 48. Problems and solutions for agents (continued)

Problem	Solution
When running command return code, the return code is not accurately shown When running 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 when 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 executes "exit %ERRORLEVEL%".
Attribute group's data missing from Tivoli Enterprise Portal I built an attribute group 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 separated by a colon). It was attempting to parse the string contained within the first <> pair. Fix the script to return data in the correct format. You can test this by editing the script in the agent directory because it will call 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 have to be installed again, even if you changed nothing but the version number. This is due to certain configuration files having the version number in them. 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 have created workspaces or situations.	All the situations and workspaces need to 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 is due to the previous data point being lost.
Core dump after editing the ref file Core 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 48. 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 at the bottom of the view is displayed: `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 defining 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 need to assign a query to a workspace. After you do this, you can see the data.
Bad string values collected from SNMP are displayed in attribute groups	This 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 trace setting to: ERROR (UNIT:shell ALL)
	
	If you see a trace like the one below 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 trying to remove an agent, you see that the agent is still listed in the desktop navigation view, but it is greyed-out.	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, which is the top 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 48. Problems and solutions for agents (continued)

Problem	Solution

Service or process monitoring returns zeros for metrics
 	
When creating 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 will create 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 script for the command return code 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 did 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 displayed when trying to monitor a service You try to monitor a service but the message `Status=UNKNOWN` is displayed, even though you have 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` should be filtered out to prevent confusion, especially 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 48. Problems and solutions for agents (continued)

Problem	Solution
No data 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 could 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 have checked, and the situations are associated with the right nodes.	Restart the Tivoli Enterprise Portal.
Data being sent by the agent does not look like it should in the Tivoli Enterprise Portal The data looks like the attributes are not being parsed in the right places.	Re-installing Tivoli Monitoring is the safest way to ensure that you have not introduced 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, so you won't have this type of problem as you build and deploy updates to an agent.
The status for a re-installed agent indicates that the agent is configured in Managed Tivoli Monitoring Services While developing an agent, you will likely generate, install, test, update the agent and then generate, install and test again. When you do this, the agent status indicates that the agent is configured in Managed Tivoli Monitoring Services after the agent is re-installed.	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 appear in the default list.)	When you created the situation on a live system, it was automatically distributed to the agent using the host name. This host name will not generally be available in every environment, so you should 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 re-import the situations. Then rebuild the Solution Installer image, and re-install the Tivoli Enterprise Monitoring Server support for the agent.

Agent crashes
 	
One reason this could occur is that the ICCRTE_DIR might not be set in the ENV file on Windows systems.
	You might have cygwin installed, so 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 48. 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 prepends "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, so by using the same Subnode ID in 2 instances of the Agent, even Agent Instances hosted on separate systems, the Managed System Names will collide and the subnodes will 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 (Windows and UNIX) so that they are all in a consistent place and easily accessed by the agent and each other.
Agent Configuration stops When deploying or configuring 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 required 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 will not have User Configurable Initial Values, although default values might still be assigned when building the agent. You also cannot override configuration parameters that are not explicitly listed in the subnode configuration overrides section. This can be addressed in two 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 need to be overridden.

Table 48. 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 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 retries** The default number of SNMP retries 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 424.
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 will be 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 removed from the snmpTrapOID, varbind does not match the enterprise OID in the mib. To fix, you need to 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 will now look 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 48. 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 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 used to test script execution). Since these are not programs, they cannot be started with createProcess(). To invoke these commands, create a .bat or .cmd file that contains the commands. It is possible to invoke 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 has been resolved in IBM Pegasus CIM Server V2.6.1.35 available at https://www.ibm.com/services/forms/preLogin.do?lang=en_US&source=aixpegcim 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, refer to AIX Information - Common Information Model Guide at http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.cim/doc/cim/About.htm?tocNode=int_187407
Collecting metrics through the Windows APIs	To collect metrics through the Windows APIs, the agent must be hosted on a Windows operating system, and remote registry administration must be enabled on the remote systems.

Table 48. 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 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)

Table 48. Problems and solutions for agents (continued)

Problem	Solution
CIM data provider intermittently fails to collect data from Solaris 9 WBEM Server after all patches have been applied.	This issue has been reported to Sun Microsystems and is a vendor limitation. If you are running a typical multi-threaded agent against a small number of 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 ini file: `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 impact on the scale of the agent. Use this flag only when it is absolutely necessary.

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 has been 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 applying Solaris Patches: • 121308 Console Patch (Sparc) • 121309 Console Patch (X86)	If you are running Solaris 10 6/06 or earlier, you must modify the your WBEM configuration file to allow remote connections after installing patch 121308-XX. Refer to the following Sunsolve document for detailed information: Document ID: 211275 Title: Solaris[TM] 10 WBEM only listens to port 898 on localhost

Table 48. Problems and solutions for agents (continued)

Problem	Solution
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 using tacmd createnode, so no Java was installed on the local system. The agent has been configured enough to allow it to connect to the Tivoli Enterprise Monitoring Server 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 using the Tivoli Enterprise Portal. Optionally, you can install a supported JRE locally to configure the agent using the Manage Tivoli Enterprise Monitoring Services interface. This problem is known to occur on 6.2.2 releases. Also, perform the steps in the following problem: Password is not stored when locally configuring the agent on a Windows system.
Password not stored when locally configuring the agent on a Windows system	When Tivoli Monitoring installs the Java Runtime Environment it applies a special patch to encrypt the password. When the JRE was manually installed, this patch was not applied. If you manually installed the JRE, perform the following steps so that passwords are correctly encrypted: 1. Copy the following three 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 48. Problems and solutions for agents (continued)

Problem	Solution
Connecting to Microsoft SQL Server using integrated authentication is failing	You can connect to a Microsoft SQL Server without a user ID or password by using Microsoft's integrated authentication. 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 runtime. • To connect using the JDBC browser, make sure the JDBC driver authentication dll is present somewhere in the system path of the computer. Start Agent Builder after the dll is located 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 using integrated authentication. This uses the current user's account to perform the authentication. • The agent runtime can connect to a Microsoft SQL Server using integrated authentication by performing the following steps: – Make sure 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 that comes with the JDBC driver must be located in the system's path or when you configure the agent, include a JVM argument in Java properties that specifies the location of the user authentication dll. 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.

Table 48. Problems and solutions for agents (continued)

Problem	Solution
Agent support files remain after uninstalling After uninstalling 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 using the following procedure: a. Ensure that your Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server are up and running. b. Log in 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.
JDBC connections are failing to find my JDBC driver JDBC connections are failing to find my JDBC driver that was added to the classpath.	Make sure 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. Note that 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.
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 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 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 called TECLIB and copy the renamed kXX.baroc file into the newly created TECLIB directory. 3. Re-try the installation or deployment of the agent that failed.

Table 48. Problems and solutions for agents (continued)

Problem	Solution
Installing the agent or application support fails with error message: KQZ0208E When installing 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 username or password is incorrect.	The password cannot include special characters.
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 have 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 using entries in your local host file.

Authorization failures when using SSH public key authentication
 	
Authorization failures when using SSH public key authentication.
	Check the agent log file. An error message similar to the one below indicates that the agent process can not open the private key file. This 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 a an example of an error message when he agent process can not 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 lower case.

Table 48. 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, it appears 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 if a shell environment exists, SSH or log in 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 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. In the case of 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 runtime on Solaris or HP-UX systems. You will know 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 two 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 runtime that was not provided by IBM if you have one available on the system.

Troubleshooting: Report Creation

This section provides a table that shows problems that might occur while creating 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 49. Problems and solutions for Report Creation

Problem	Solution
Cognos data model for an agent loads into Framework Manager but testing a table causes a table or view not found error The Cognos data model for an agent loads into Framework Manager, but the testing a table causes a table or view not found error.	This problem could have two 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 used in the data model's data source does not match the schema used in the Tivoli Data Warehouse database. Change the schema value in the data source to match the schema used.
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>

Table 49. Problems and solutions for Report Creation (continued)

Problem	Solution

When running on Windows, testing an Oracle data source in Tivoli Common Reporting fails
 	
When running on Windows, testing an Oracle data source in Tivoli Common Reporting fails with an error similar to the following:
`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 can not be utilized when creating reports for an agent I am not able to utilize the ManagedSystem dimension when creating reports for my agent.	Perform the following checks: • Make sure you have created or altered the ManagedSystem table in the Tivoli Data Warehouse. For more information, see “Create or alter the ManagedSystem Table and Stored Procedure in the Tivoli Data Warehouse” on page 597. • Make sure you have created the stored procedure that populates the ManagedSystem table for Agent Builder agents. For more information, see “Create or alter the ManagedSystem Table and Stored Procedure in the Tivoli Data Warehouse” on page 597. • Make sure you have run the stored procedure using your agent's product code. For more information, see “Running the stored procedure” on page 604. • Examine the ManagedSystem table to ensure that entries for your agent type exist in the table.

Table 49. 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 appear: **Issue 1: Facts identified by cardinality** The query subject is identified as a fact because of the cardinality of its relationships. It is recommended that you review the implications of this 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 recommended that you evaluate this query subject in the context of the model to ensure queries will not be 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 have defined. This ambiguity can cause a query to be split unnecessarily. It is recommended that you examine and resolve this 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 should be an exact match to the keys of a uniquely identified determinant. Please review the cardinality of the relationship and the determinant information to determine if adjustments are required. (continued on the next page)	These warnings are expected and do not signify a problem with the Cognos data model.

Table 49. 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 will override the Minimized SQL setting** A query subject has attributes that will 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 will override the Minimized SQL Generation type. **Problem Description:** There may be problems generating minimized SQL for the object Time Dimension. The object is set to use minimized SQL. **Problem Description:** There may be problems generating minimized SQL for the object ManagedSystem. The object is set to use minimized SQL. **Problem Description:** There may be problems generating 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 sites contain troubleshooting information:

- Go to the IBM Software Support site at http://www.ibm.com/software/support/probsub.html and follow the instructions.
- Go to the IBM Tivoli Distributed Monitoring and Application Management Wiki at http://www.ibm.com/developerworks/wikis/display/tivolimonitoring/Home. 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 http://www.ibm.com/software/support/isa.

Appendix A. Sharing project files

If you want to share an IBM Tivoli Monitoring agent project with someone, complete the following steps:

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 need to make the directory `c:\Documents and Settings\User1\workspace\TestProject` and make all of its contents accessible to your system
2. Select **File > Import**.

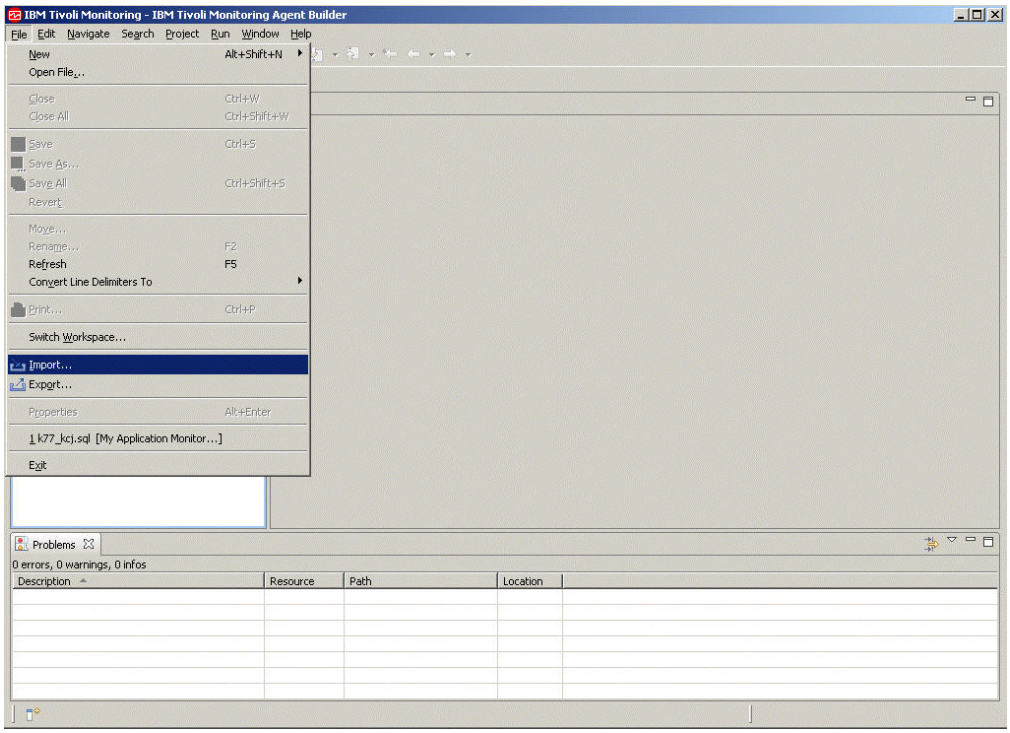


Figure 287. Importing files

3. Open **IBM Tivoli Monitoring**.
4. Select **IBM Tivoli Monitoring Agent** and click **Next**.

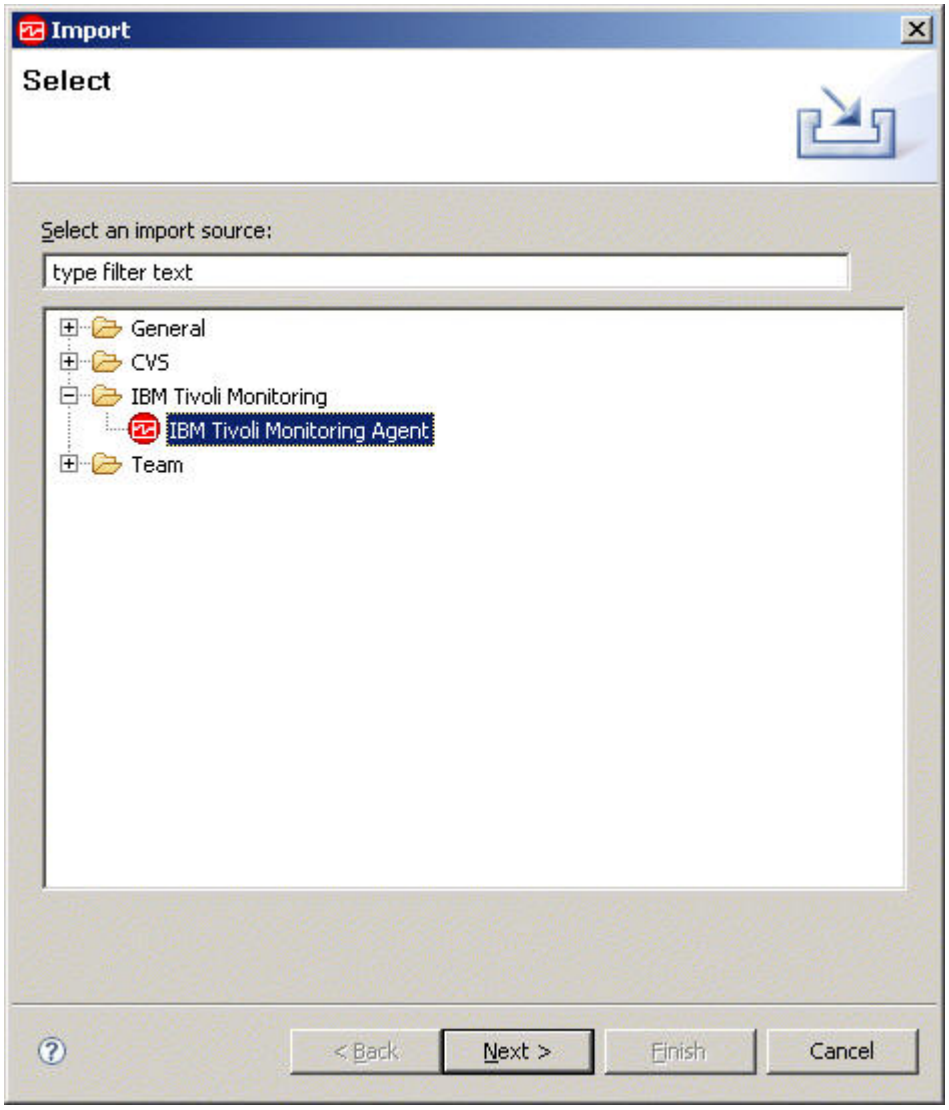


Figure 288. Importing files (continued)

5. Either type the full path to the agent xml file or click **Browse** to navigate to the file.
6. Click **Finish**.

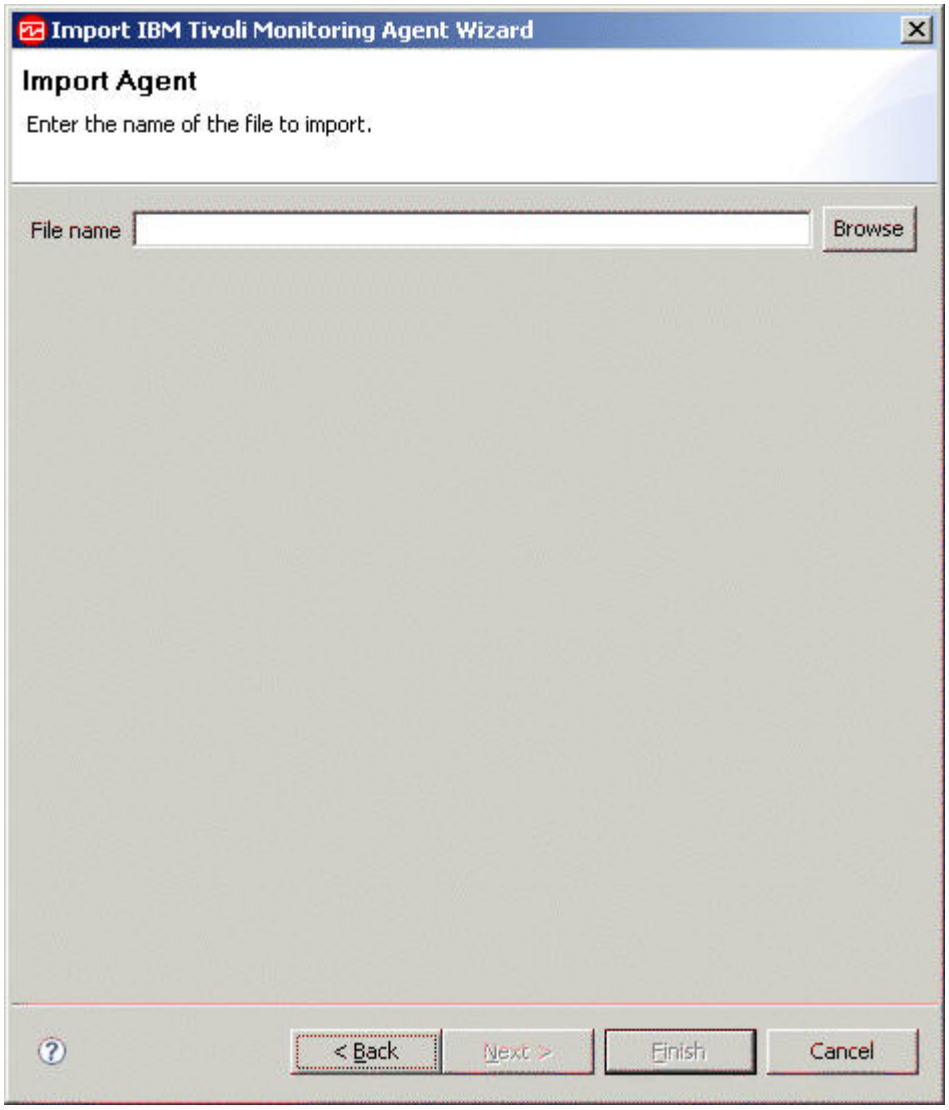


Figure 289. Importing files (continued)

When the wizard completes, you will see the new IBM Tivoli Monitoring agent project in your workspace.

Appendix B. Command-line options

This chapter lists and describes the Tivoli Monitoring Agent Builder commands. Table 50 summarizes these command-line options. After the table, a section for each command describes how to run the command 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.

Comments
Provides additional commands or text that can give you more information.

List of commands

The Tivoli Monitoring Agent Builder contains a command-line interface (CLI) that you can use to generate the Tivoli Monitoring Agent without launching 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

Table 50 lists the name and purpose statement for each command option for the **text** command:

Table 50. 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. Also installs into a local Tivoli Monitoring environment.
generatemappingfile	Creates the mapping file for migrating 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 commands in this appendix 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.

generatelocal

Purpose

Loads and validates the `itm_toolkit_agent.xml` file and generates the files for running the Tivoli Monitoring Agent. Also installs 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 containing 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` will be validated and the files that are required to run ABCAgent will be generated in `C:\IBM\ITM`:

```
install_location\agenttoolkit.bat C:\ABCAgent -generatelocal C:\IBM\ITM
```

generatemappingfile

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 613.

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 containing `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
```

generatezip

Purpose

Loads and validates the `itm_toolkit_agent.xml` file and generates a compressed file named *productcode*`.zip` or *productcode*`.tgz` so that the agent can be installed on another system. Depending on your environment, both files can be generated.

Format

For Windows systems:

install_location`\agenttoolkit.bat` *project_dir* `-generatezip` *output_dir*

where:

project_dir
: Name of a directory containing 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` will be validated and a compressed file that contains the required files for running ABCAgent will be generated in `C:\Output`:

```
install_location\agenttoolkit.bat\ C:\ABCAgent -generatezip C:\Output
```

Appendix C. Attributes reference

This appendix 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: processes, services (Windows only) 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

Description
The name of the process, service or functional test. This will match the executable name of the process, the service short name or the name of the process 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 this is 'UP', 'DOWN' 'WARNING' or 'PROCESS_DATA_NOT_AVAILABLE'. 'PROCESS_DATA_NOT_AVAILABLE' will be displayed for a process when the matching process is running but the resource use information cannot be collected for that process.
- For services this is 'UP', 'DOWN' or 'UNKNOWN'. 'UNKNOWN' will be displayed when the service is not installed.
- For command return codes this is 'PASSED' or 'FAILED'

Type String

Names

Attribute name
Status

Column name
STATUS

Full Name attribute

Description
The full name of the process including information that is process dependent. It might include the full path if the process was started that way, or a partial path or even a path that was 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 measured in faults per second. This will only contain 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 will only contain valid data for processes.

Type Integer (Gauge)

Names

Attribute name
Working_Set_Size

Column name
WORKSET

Thread Count attribute

Description
The number of threads currently allocated by this process. This will only contain valid data for processes.

Type Integer (Gauge)

Names

Attribute name
Thread_Count

Column name
THREADS

PID attribute

Description
The process id associated with the process. This will only contain valid data for processes.

Type Integer (Gauge)

Names

Attribute name
PID

Column name
PID

Percent Privileged Time attribute

Description
The percentage of the available CPU 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 CPU 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 execute instructions

Type Integer (Gauge)

Names

Attribute name
Percent_Processor_Time

Column name
PERCPROC

Command Line attribute

Description

The program name and any arguments specified on the command line when the process was started. This has the value N/A if this is 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 displayed, 'NOT_RUNNING' when it is not running correctly or 'N/A' 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 will 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 will 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 is valid only for command return codes.

Type String

Names

Attribute name
Functionality_Test_Message

Column name
FUNCMSG

Performance Object Status node

The Performance Object Status attribute group contains the status of the performance objects collected for this application. When data cannot be collected for a table, the application can be degraded, the corresponding part of the application might not be installed, configured or running, or the monitoring for the application can be impaired. This attribute group is grouped in the availability navigator group item.

For example, it is possible to recognize when a counter will not work and to remove it from the collected set. Errors are logged in the trace log, and the status in the Performance_Object_Status table shows that a problem occurred:

- If one or more counters cannot be collected (but not all), the Object Status is `ACTIVE`, and the error code is `COUNTER NOT FOUND`.
- If all counters fail, the Object Status is `INACTIVE`, and the error code is `COUNTER NOT FOUND`.
- If all counters succeed, the Object Status is `ACTIVE`, and the error code is `NO ERROR`.
- If the call to gethostbyname indicated in the RAS1log could not be resolved to an address, the Object Status is `CANNOT_RESOLVE_HOST_NAME`, and the error code is `32`.

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

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The value is the time 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

Query Name attribute

Description
The name of the attribute group

Type String

Names

Attribute name
Query_Name

Column name
ATTRGRP

Object Name attribute

Description
The name of the performance object

Type String

Names

Attribute name
Object_Name

Column name
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 will return the numbers. The defined values are WMI(0), PERFMON(1), (4), SHELL COMMAND(5) or LOG_FILE (11). Any other values show the numeric value in the Tivoli Enterprise Portal.

Names

Attribute name
Object_Type

Column name
OBJTYPE

Object Status attribute

Description
The status of the performance object

Type Integer with enumerated values. The strings are displayed in the Tidvoli Enterprise Portal, the warehouse and queries will return the numbers. The defined values are: ACTIVE (0) or INACTIVE (1). Any other values will display the numeric value in the Tivoli Enterprise Portal.

Names

Attribute name
Object_Status

Column name
OBJSTTS

Error Code attribute

Description

The error code associated with the status. When the group is ACTIVE this will be NO ERROR. Other values allow you to verify the operation of the agent.

Type Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal, the warehouse and queries will return the numbers. The defined values are: when the

NO ERROR (0) group is ACTIVE this will be NO ERROR (0). Other values allow you to verify the operation of the agent. They are:

- 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)

- APPLICATION_HUNG (1000)
- DISK_NOT_READABLE (1001)
- 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)
- NVALID_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)

Any other values will display the numeric value in the Tivoli Enterprise Portal.

Names

Attribute name
Error_Code

Column name
ERRCODE

Last Collection Start attribute

Description
The most recent time a data collection for this group started

Type

Timestamp with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values shown in parentheses. The following values are defined:

- NOT COLLECTED (0691231190000000)

Any other values will display the actual value returned by the agent in the Tivoli Enterprise Portal.

Names

Attribute name
Last_Collection_Start

Column name
COLSTRT

Last Collection Finished attribute

Description
The most recent time a data collection for this group finished

Type

Timestamp with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values shown in parentheses. The following values are defined:

- NOT COLLECTED (0691231190000000)

Any other values will display the actual value returned by the agent in the Tivoli Enterprise Portal.

Names

Attribute name
Last_Collection_Finished

Column name
COLFINI

Last Collection Duration attribute

Description
The duration of the most recently completed data collection for this group in seconds.

Type

Real number (32 bit counter) with 2 decimal places of precision

Names

Attribute name

Column name

Average Collection Duration attribute

Description
The average duration of all data collections for this group in seconds.

Type

Real number (32 bit counter) with 2 decimal places of precision with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values shown in parentheses. The following values are defined:

- NO DATA (-100)

Any other values will display the actual value returned by the agent in the Tivoli Enterprise Portal.

Names

Attribute name
Average_Collection_Duration

Column name
COLAVGD

Refresh Interval attribute

Description
The interval at which this group is refreshed in seconds.

Type Integer (32 bit counter)

Names

Attribute name
Refresh_Interval

Column name
REFRINT

Number of Collections attribute

Description
The number of times this group has been collected since the agent started.

Type Integer (32 bit counter)

Names

Attribute name
Number_of_Collections

Column name
NUMCOLL

Cache Hits attribute

Description
The number of times a data request for this group was satisfied from the cache.

Type Integer (32 bit counter)

Names

Attribute name
Cache_Hits

Column name
CACHEHT

Cache Misses attribute

Description
The number of times a data request for this group was not available in the cache.

Type Integer (32 bit counter)

Names

Attribute name
Cache_Misses

Column name
CACHEMS

Cache Hit Percent attribute

Description
The percentage of data requests for this group that were satisfied from the cache.

Type Real number (32-bit counter) with 2 decimal places of precision

Names

Attribute name
Cache_Hit_Percent

Column name
CACHPCT

Intervals Skipped attribute

Description
The number of times a background data collection for this group was skipped because the previous collection was still running when the next one was due to start.

Type Integer

Names

Attribute name
Intervals_Skipped

Column name
INTSKIP

Thread Pool Status attribute group

The Thread Pool Status attribute group contains information that reflects the status of the internal thread pool 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 value is the time 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 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 has 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 has 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 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 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

A Summary node is created for each Log File data source where **Include summary attribute group** was selected in the advanced properties of the data source. Its name 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 additional attributes (see Step 9d on page 194 in Chapter 16, “Monitoring a log file,” on page 179), then the summary attribute group for that log attribute group will also contain each of the attributes you selected. The values will be 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 will include one row per unique set of keys. The row indicates how many log records were 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 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 coming back from an `errpt` command 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 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 was recorded as reported by errpt.

Note: This attribute is hidden at run time. This attribute contains a raw value. Other attributes 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 the **Display attribute in the Tivoli Enterprise Portal** check box.

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 are the enumerated values. The raw values for use with situations are H, S, O, and U respectively

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 describing 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 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 the **Display attribute in the Tivoli Enterprise Portal** check box.

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 containing 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 was 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

The Monitor and Notification attribute groups are listed below. 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 as to 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 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 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. This describes how the MBean's observed attribute 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 being 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 attribute's maximum value. 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 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. This describes how the MBean's observed attribute 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 being 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 dip below

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 rise above

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 have been 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 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 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. This describes how the MBean's observed attribute 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 being monitored in the Observed MBean

Type String

Names

Attribute name
Observed_Attribute

Column name
OBSERVED_A

Compare String attribute

Description
The string 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 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 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 sending it 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 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 additional information. In the SNMP MIB Browser window (Figure 98 on page 133), select the **Include static attributes** 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 value is the time 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 being 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 will return 0.0.0.0 for this enumeration. Any other hostname 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 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 will 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 returned by the HTTP request.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse and queries will 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 additional objects associated with the monitored page.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse and queries will 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 returned by the HTTP request.

Type Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse and queries will 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 used at the target URL web site.

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 will 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 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

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 value is the time 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

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
This is 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
This is 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 this agent has processed. This is an event based attribute group.

The following list 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 value is the time 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 launched.

Type Integer

Names

Attribute name
Action_App_Return_Code

Column name
TSKAPRC

Action Message attribute

Description
The message associated with the return code of the action.

Type String

Names

Attribute name
Action_Message

Column name
TSKMSGE

Action Instance attribute

Description
The instance 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 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 will 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 2 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 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 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 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 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 has been processed, data after it has not been 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 based on the granularity of the data exposed by the operating system. This can result in values of "0.00" when a regular expression takes a very small time to evaluate. The CPU times should be used 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 the default minimum Tivoli Monitoring version of 6.2.1 or later. The minimum ITM Version is selected on the Agent Information page, for more information see Step 12 “Naming your agent” on page 20. 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 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, respectively. 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. The defined values are INCLUDE(1) or EXCLUDE(2) if the filter is used to filter records and are BEGIN(3) or END(4) if the filter is used to define records.

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. This is 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 has never been used or if each of the filter processing took less than one hundredth of a second.

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 one of the filter processing took less than one hundredth of a second.

Type Integer (Gauge)

Names

Attribute name
Min_Processor_Time

Column name
CPUTMIN

Filter Count attribute

Description
The number of times the filter has been invoked. 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 has been invoked 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 has been invoked 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 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

Attention:: This 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 397.

With Agent Builder you can build an installable package to easily distribute custom workspaces, situations, queries, and Take Action commands that you have created as an application support extension for an existing agent.

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 391.

Creating a new Application Support Extension project

To create an Application Support Extension project:

1. From the Agent Builder, select **File > New > Other**.
2. Select **IBM Tivoli Application Support Extension** under IBM Tivoli Monitoring Wizards.

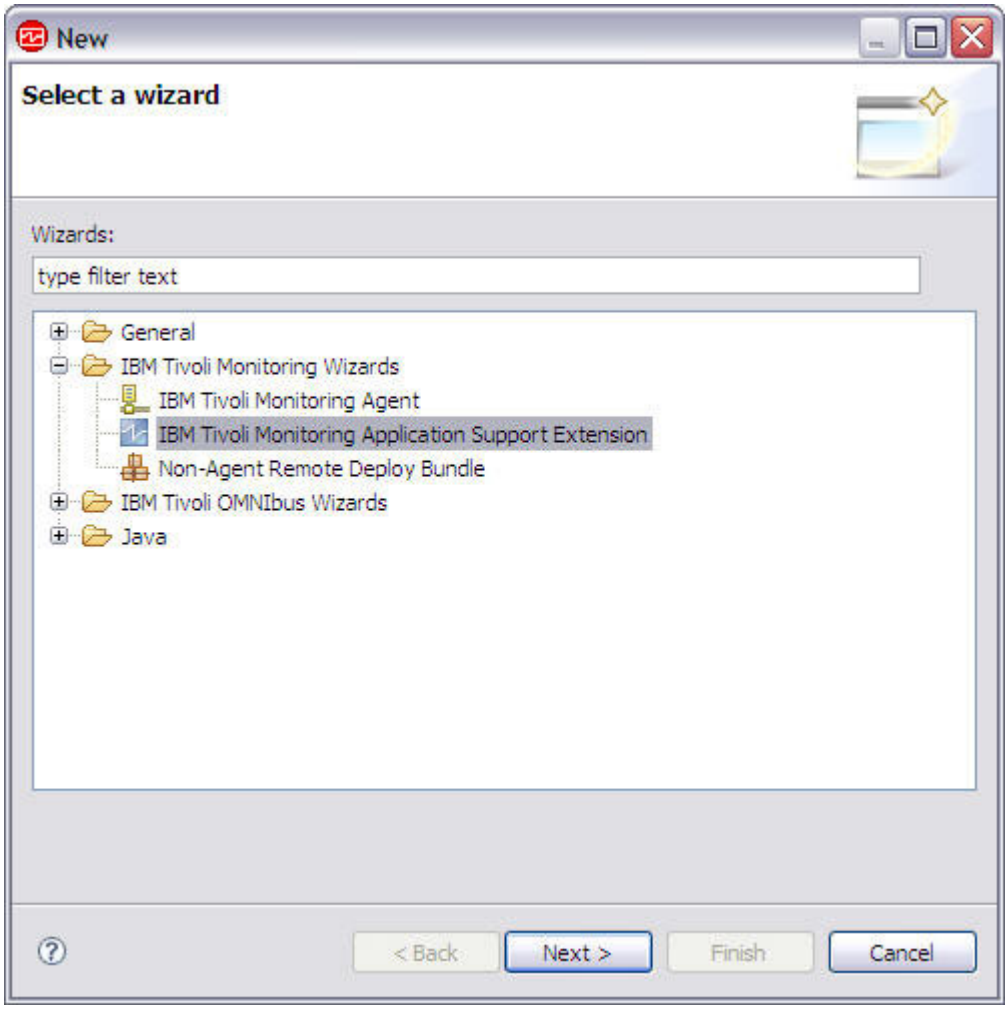


Figure 290. IBM Tivoli Monitoring Application Support Extension

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 291 on page 587.

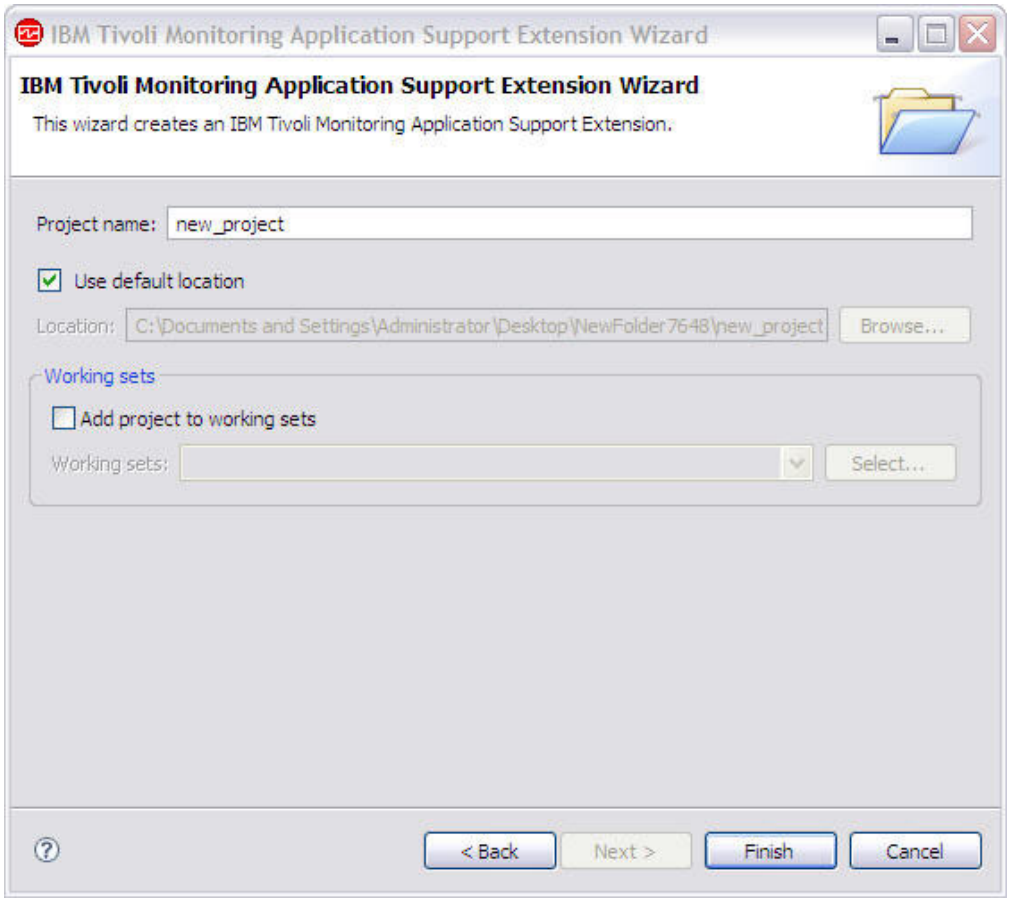


Figure 291. Naming the project

Adding your support files to an Application Support Extension project

To add your support files, complete the following steps:

1. Right-click the newly created project and select **IBM Tivoli > Import Application Support Extensions**.

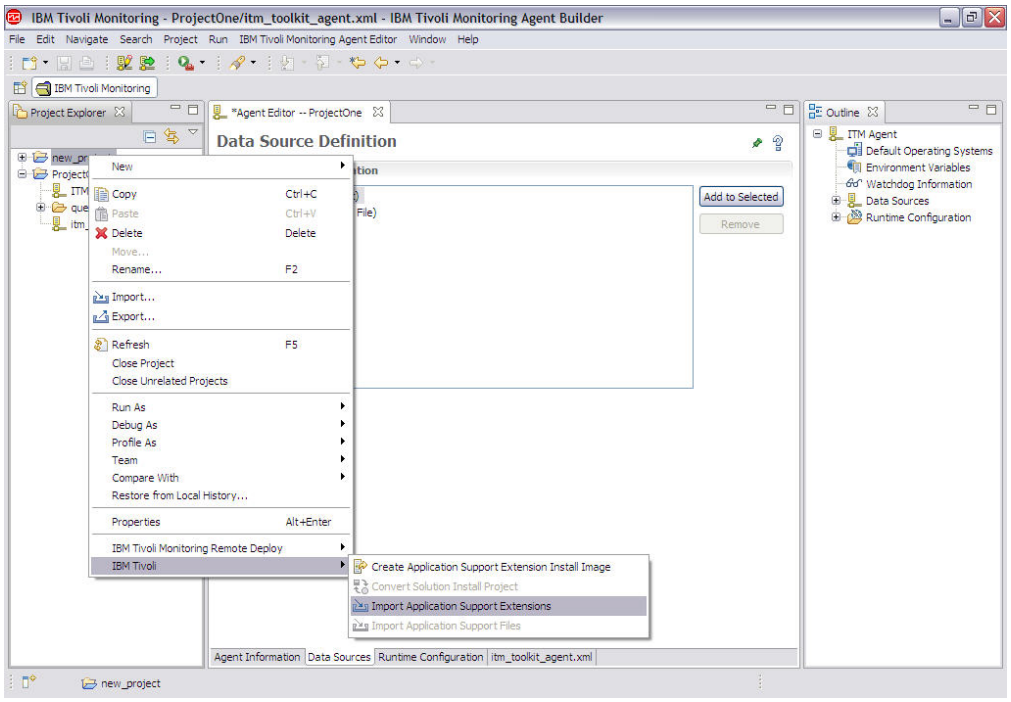


Figure 292. Importing support files

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 a new one.

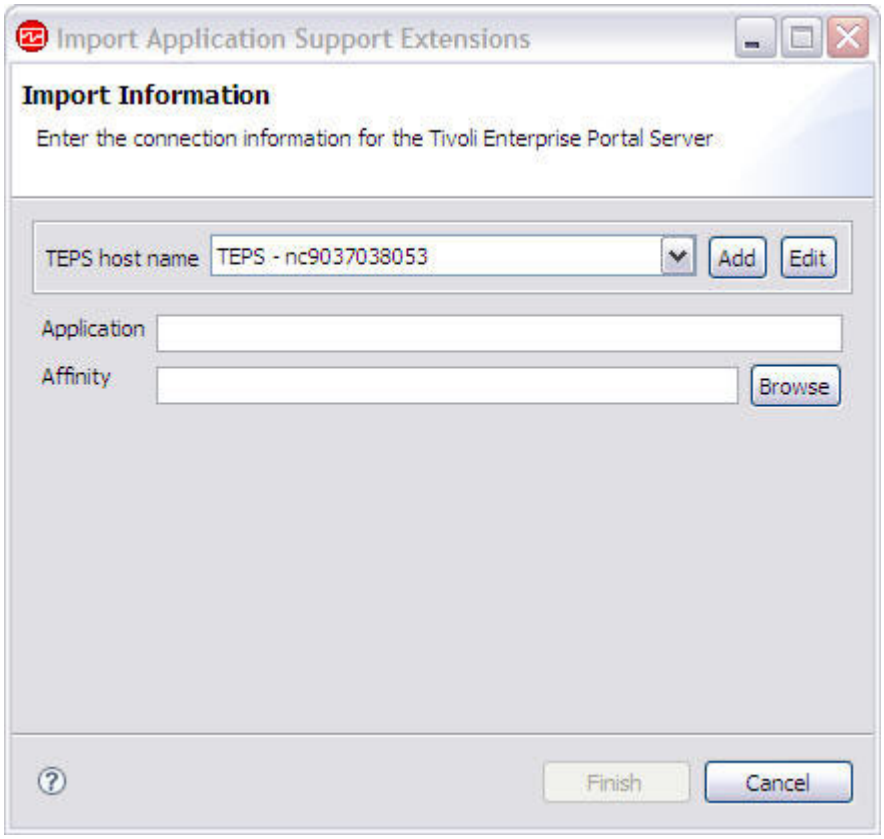


Figure 293. Import Information

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 so on, 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.

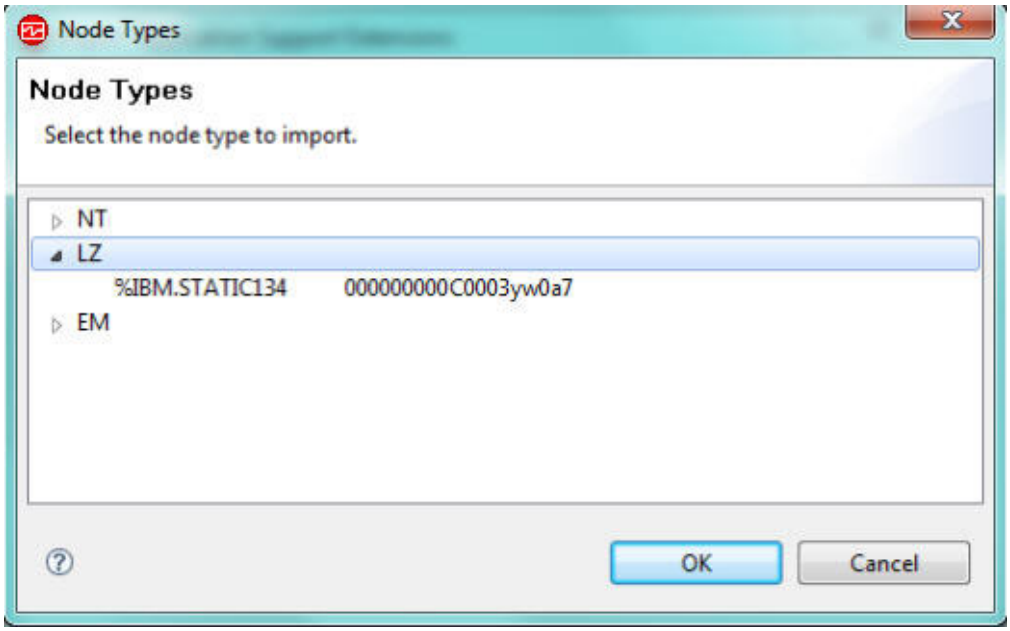


Figure 294. Affinities list

5. When you are satisfied with the import information, click **Finish**.

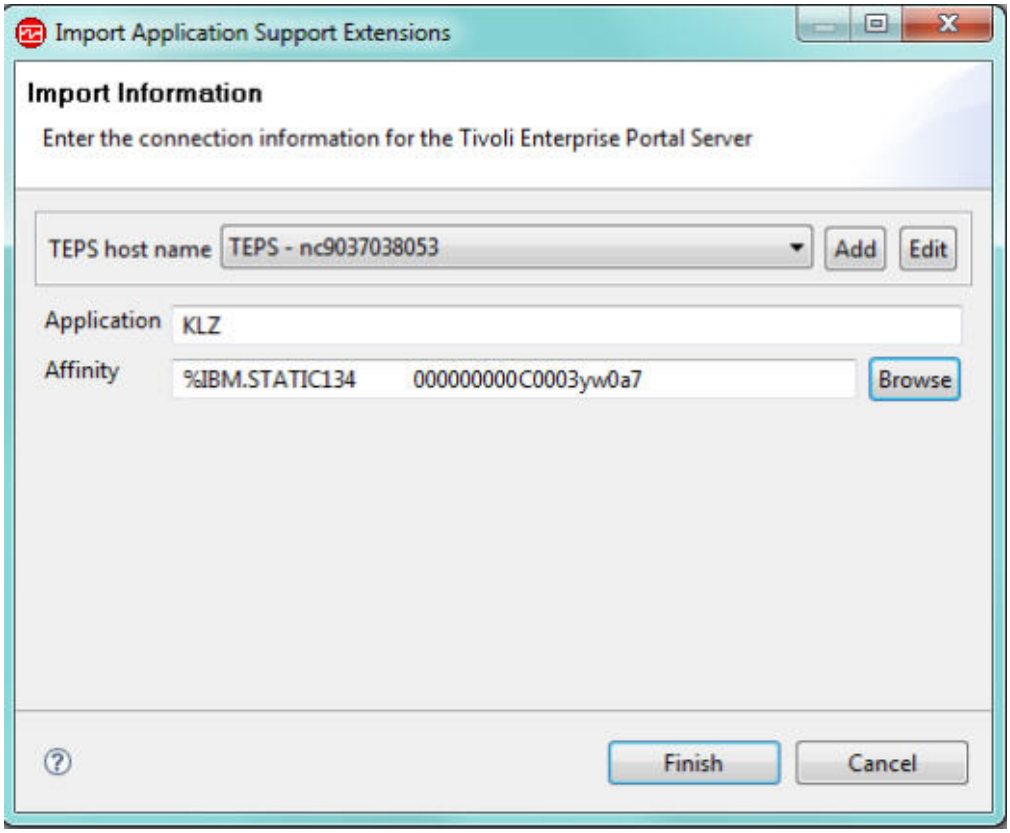


Figure 295. Import Information complete

6. In the Situations window Figure 296 on page 591, select the situations you want to import from the Available Situations list. Click the button 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.

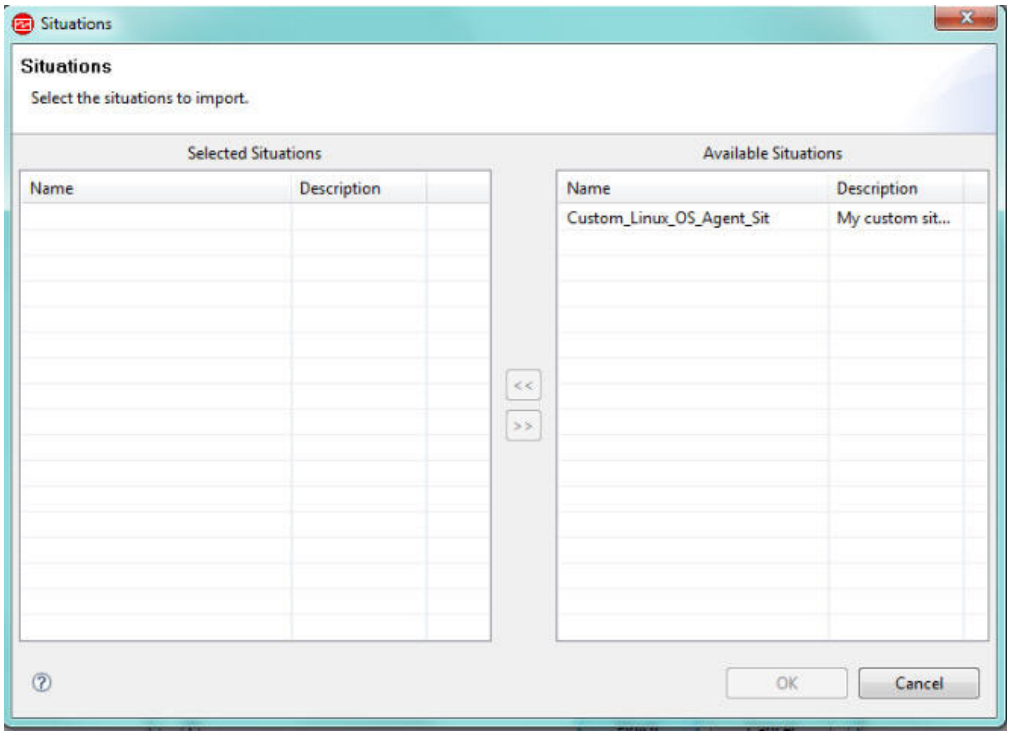


Figure 296. Importing situations

7. In the Queries window Figure 297 on page 592, select the queries you want to import from the Available Queries list. Click the button to add them to the Selected Queries list and click **OK**.

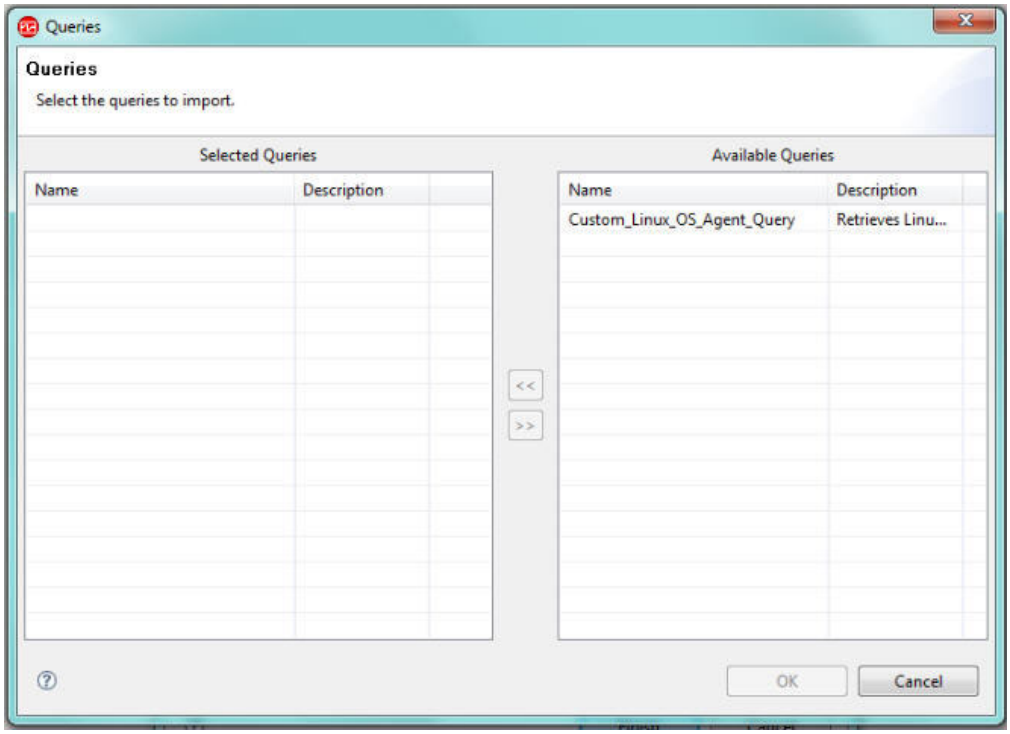


Figure 297. Importing queries

8. In the Take Actions window Figure 298, select the Take Action commands you want to import from the Available Take Actions list. Click the button to add them to the Selected Take Actions list and click **OK**.

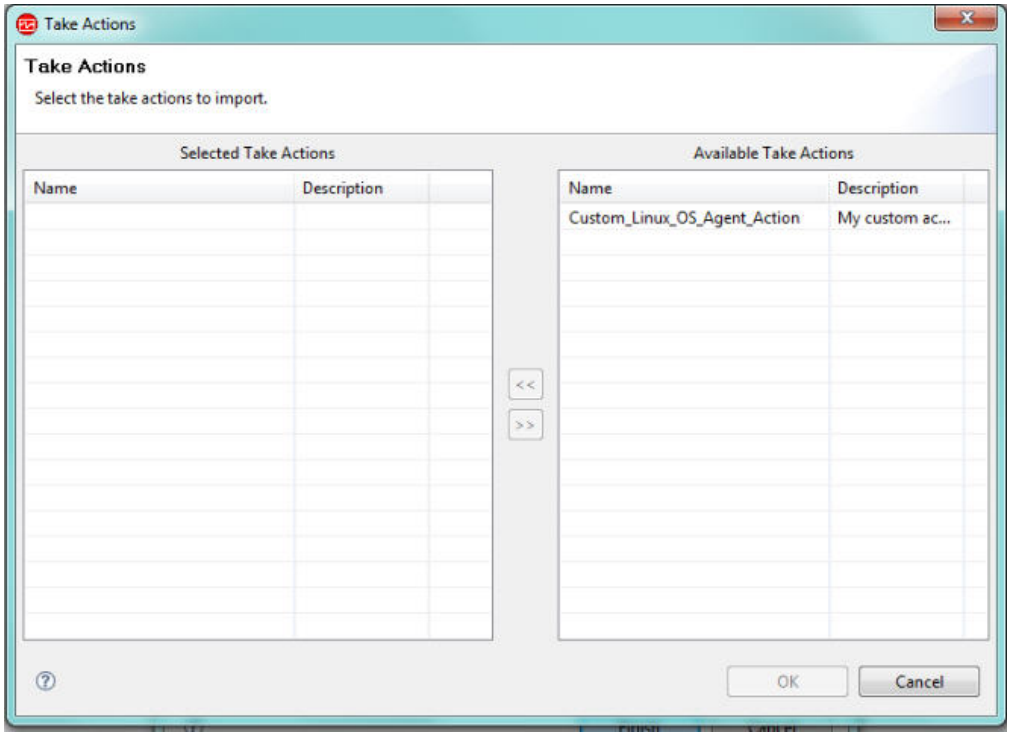


Figure 298. Import Take Action commands

9. The support files for the agent are put in the project under the appropriate folder.

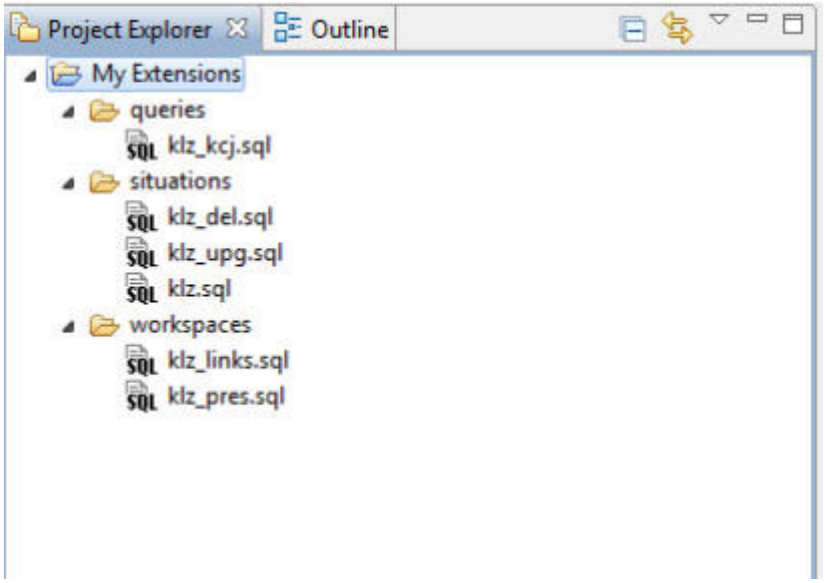


Figure 299. Agent support files

You can repeat this process for as many different agents as you wish. The Agent Builder will create a single install image from all of the support files in the Application Support Extension project.

Generating the Application Support Extension install image

To generate the install image for your Application Support Extension, perform the following steps:

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 Install Image window Figure 300 on page 594, enter the directory where the image is to be placed.

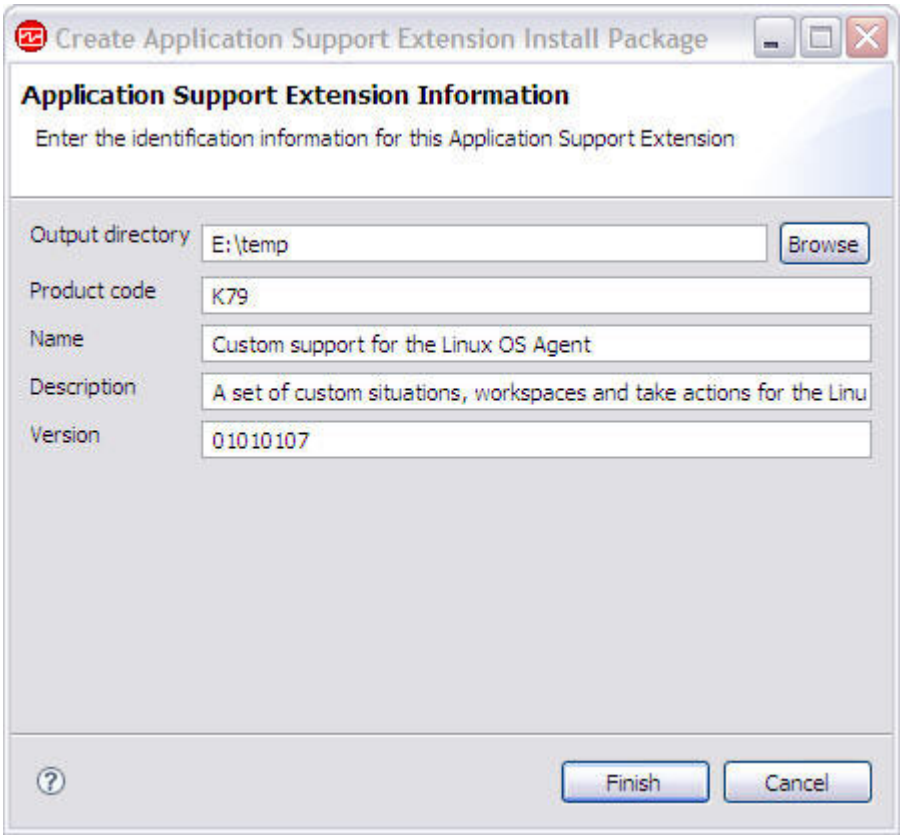


Figure 300. Application Support Extension Install Image window

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 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 3-letter product code, you will be 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

To install your Application Support Extension, perform the following steps:

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:

a. Run the install*KXX*TEMSSupport.bat (on Windows) or install*KXX*TEMSSupport.sh (on Unix).
b. 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:
   a. Run the install*KXX*TEPSSupport.bat (on Windows) or install*KXX*TEM\PSSupport.sh (on Unix).
   b. The format for the command is as follows:

```
installKXXTEPSSupport[.bat | .sh] <ITM Install Directory> [-r]
```

   where `-r` indicates that the Tivoli Enterprise Portal Server should be restarted after installation

Converting an existing Solution Install Project to an Application Support Extension project

If you have existing Solution Install projects that you want to convert to Application Support Extension projects, perform the following steps:

Note: Only support files in the Solution Install project will be migrated.

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, which allows you to import agent information into the Cognos Framework Manager for report creation. This data model can be opened and viewed in the Framework Manager, which will build a model package to be published into Tivoli Common Reporting. The data model can also be customized or modified within the Framework Manager prior to publication.

Once a report is created, Agent Builder also allows for a final report package to be imported into the Agent Builder project. This enables future agent projects to be generated with the reports already part of the agent package. The reports that are packaged as part of the agent install image can be imported into Tivoli Common reporting in your production environment.

Note: In this appendix, note the following:

- k*xx* or K*xx* refers to the product code given to the agent, for example, k99.
- *dbType* refers to the database being used by the Tivoli Data Warehouse, for example, DB2.

Prerequisites

To use this functionality, you need to perform the following prerequisite steps:

Note: These steps only need to be completed once, as all future data models generated with Agent Builder will use this environment.

Note: It is advisable to create an isolated development environment for agent testing and report creation.

1. Install and configure a Tivoli Data Warehouse.
2. Create tables and Procedures in the Tivoli Data Warehouse.
   - Create or alter the ManagedSystem table in the Tivoli Data Warehouse, and create the stored procedure to populate the ManagedSystem table.
   - Install the Tivoli Reporting and Analytics Model into the Tivoli Data Warehouse.
3. Install and configure Tivoli Common Reporting.
4. Install and configure the Framework Manager.

Tivoli Data Warehouse

To create reports, you need a Tivoli Data Warehouse, a Warehouse Proxy agent, and a Summarization and Pruning agent 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

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 given 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 already exist in the Tivoli Data Warehouse.
- Edit the ManagedSystem table. Use this script if the table already exists in the Tivoli Data Warehouse. Other reporting products may create the ManagedSystem table, but they do not create it with all of the required columns.
- Create a stored procedure that will populate the ManagedSystem table from tables in the Tivoli Data Warehouse.

These scripts only need to be run once.

DB2: The scripts for DB2 are located in the following directory:

`reports/db2/K`*`xx`*`/reports/cognos_reports/itmk`*`xx`*`/db_scripts`

For DB2, perform the following steps:

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 this 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.
   a. If the query returns `1`, the table exists. Run the alter script:

      ```
      db2 -tvf alter_table.sql
      ```

      —OR—

   b. 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
   ```

Oracle: The scripts for Oracle are located in the following directory:

`reports/oracle/K`*`xx`*`/reports/cognos_reports/itmk`*`xx`*`/db_scripts`

For Oracle, perform the following steps:

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 this 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.
   a. If the query returns 1, the table exists. Run the alter script:
      ```
      @<path to alter_table.sql>;
      ```
      —OR—
   b. 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>;
   ```

SQL Server 2005 and 2008: The scripts for SQL Server are located in the following directory:

`reports/mssql/K`*xx*`/reports/cognos_reports/itmk`*xx*`/db_scripts`

For SQL Server 2005 and 2008, perform the following steps:

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 this 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.
   a. 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>
      ```
      —OR—
   b. 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>
   ```

Tivoli Reporting and Analytics Model

Tivoli Reporting and Analytics Model (TRAM) contains the base set of knowledge that is common to all reporting packages, and 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.

Populating the Tivoli Data Warehouse with the Tivoli Reporting and Analytics Model: Perform the following steps to create Tivoli Reporting and Analytics Model Common Dimensions in Tivoli Data Warehouse:

1. Browse to the Tivoli Reporting and Analytics Model database scripts.
   a. Extract the agent package.
      - On Windows systems, agent package is k*xx*`.zip`.
      - On Linux and UNIX systems, agent package is k*xx*`.tgz`.
   b. Navigate to the appropriate database scripts.
      - DB2 scripts located in the Agent package at:

reports/db2/K*xx*/reports/cognos_reports/itmk*xx*/db_scripts

- Oracle scripts located in the Agent package at:
  reports/oracle/K*xx*/reports/cognos_reports/itmk*xx*/db_scripts
- Microsoft SQL Server scripts located in the Agent package at:
  reports/mssql/K*xx*/reports/cognos_reports/itmk*xx*/db_scripts

2. Run the database scripts to generate the common dimensions within the Tivoli Data Warehouse. Each script set provides a README for usage instructions.
3. 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. This application needs to be installed and configured with a data source connecting 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 on the appropriate database type. Note the name 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 43.

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

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 will open the data model for modification, customization, or publication.

Prerequisites

Once the agent has been 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 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 once 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 may 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: Once you have configured historical collection and summarization you need to wait a sufficient amount of time for data to end up in the summary tables.

   Note: By default, the Summarization and Pruning agent is configured to run once a day at 2 a.m. You may 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*.

Opening the Agent Data Model in Framework Manager

Note: The generated data model for the agent will contain all of the summary time dimensions for each attribute group: hourly, daily, weekly, monthly, quarterly, and yearly. The dimensions will only exist in the Tivoli Data Warehouse for the agent if summarization and pruning is configured for the agent and the dimensions are selected, and the Summarization and Pruning agent has created and populated the tables. Reports can be defined and published into Tivoli Common Reporting that use dimensions that do not exist, but they will not function until the summary tables are created by the Summarization and Pruning agent.

1. Open the Framework Manager.

2. Click the **Open a project...** link.
3. Navigate 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/`
4. Select the Agent project file, `Kxx.cpf`.

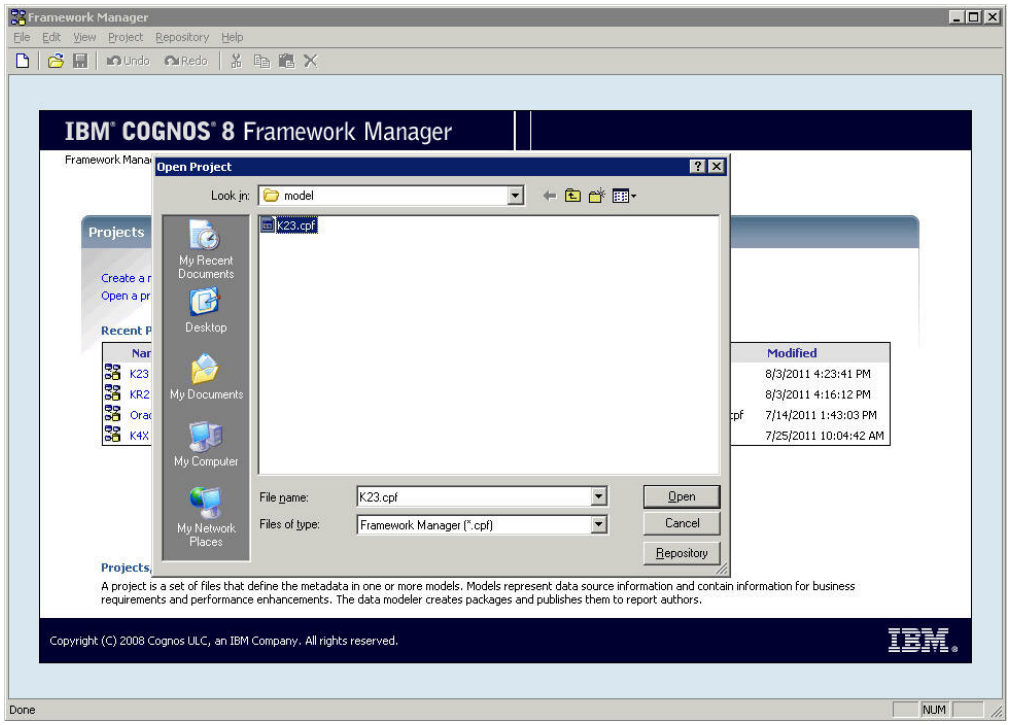


Figure 301. Selecting agent project file

Note: Once an agent project is opened in Framework Manager, the agent name is listed under the Recent Projects.

Populating the ManagedSystem Table

The MangedSystem table is populated using the `kqz_populate_msn` stored procedure. For more information, see "Running the stored procedure" on page 604. This procedure needs to 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 will 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 wish to use the scripts, the list of attribute groups is listed in the script's comment header block.

Sample scripts are created that show which tables need to 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 51. 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 should be started. More than one Tivoli Enterprise Monitoring Server can be specified using a space separated list. If you specify more than one Tivoli Enterprise Monitoring Server, put the list in quotes. For example, -n “tems1 tems2”
-m *managed_system_group_or_manged_system*	The managed system group or managed system name against which the collection should be started. More than one managed system group or managed system can be specified using a space separated list. If you specify more than one managed system group or managed system, put the list in quotes. For example, -m “msg1 msg2”

The following table describes the optional arguments:

Table 52. Optional arguments

Argument	Description
-s *teps_host*	The host name or IP address of the Tivoli Enterprise Portal Server. If this is not specified, the default is localhost.
-p *teps_password*	The password for the Tivoli Enterprise Portal Server user specified with the -u option. If not specified, the script will prompt for the password
-c *historical_collection_interval*	The historical collection interval to use when starting the historical collections. If not specified, the default is 1h (1 hour). The valid values are: 15m, 30m, 1h, 12h or 1d, where m is minutes, h is hours and d is days.

Table 52. Optional arguments (continued)

Argument	Description
-r *pruning_interval*	The pruning interval to use for the historical data. The historical data should 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 has been started, the kqz_populate_msn stored procedure should be run periodically so that the ManagedSystem table contains the most current list of managed systems in the Tivoli Monitoring environment.

Running the stored procedure

Perform the following steps to run the stored procedure:

DB2:

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>')"
   ```

Oracle:

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>');
   ```

SQL Server 2005 and 2008:

1. 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>'"
   ```

Publishing the Agent Data Model to Tivoli Common Reporting

1. Open the Framework Manager.
2. Open the Agent project.
3. Expand **Packages** in the navigation tree.
4. Right-click the agent package and select **Publish Packages**.

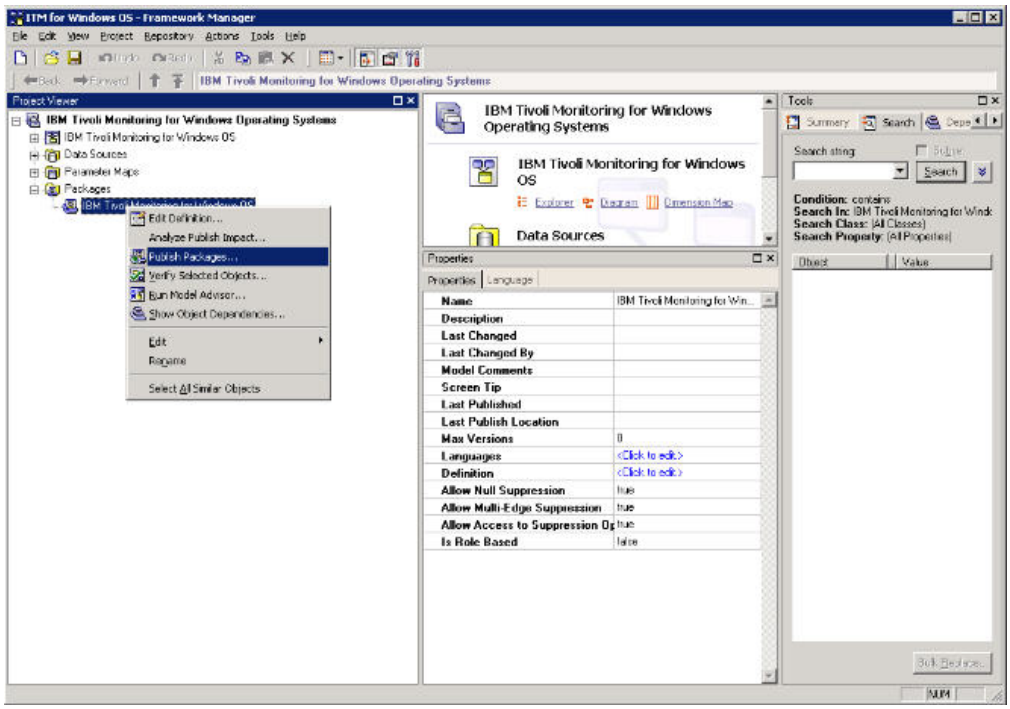


Figure 302. Selecting Publish Packages

Creating reports in Tivoli Common Reporting

1. Log in to Tivoli Common Reporting.
2. Navigate to Public Folders, and under **Reporting** in the left panel select **Common Reporting**.

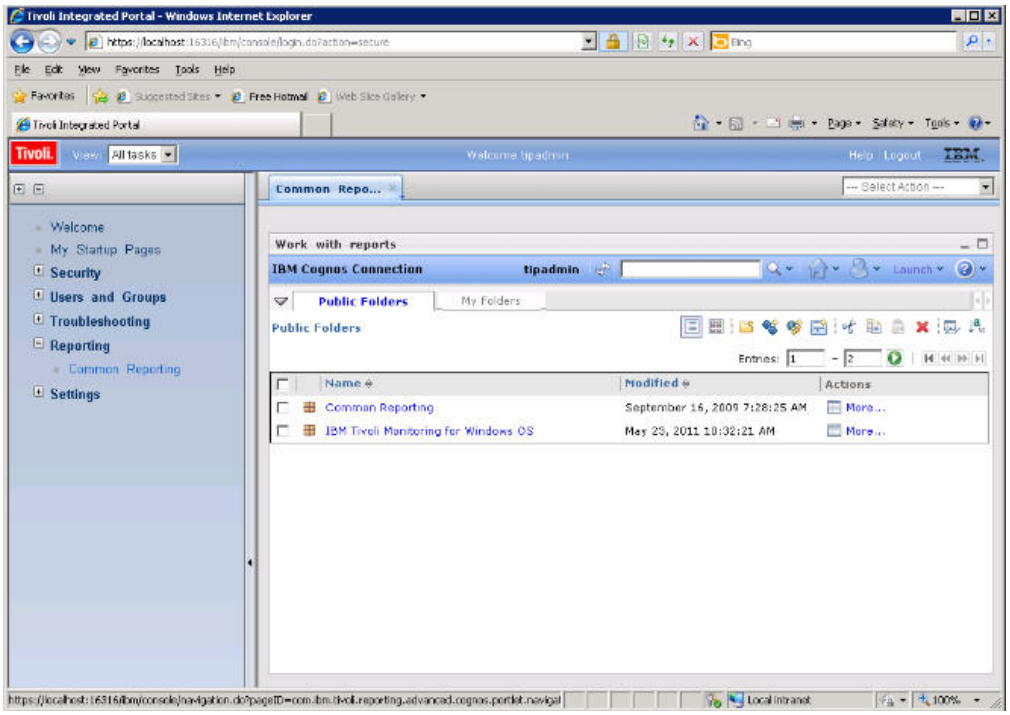


Figure 303. Selecting Common Reporting

3. Select your Tivoli Monitoring agent from the list provided.
4. Open the report creation tool, by clicking on the Launch drop-down menu and selecting **Report Studio** or **Query Studio**.

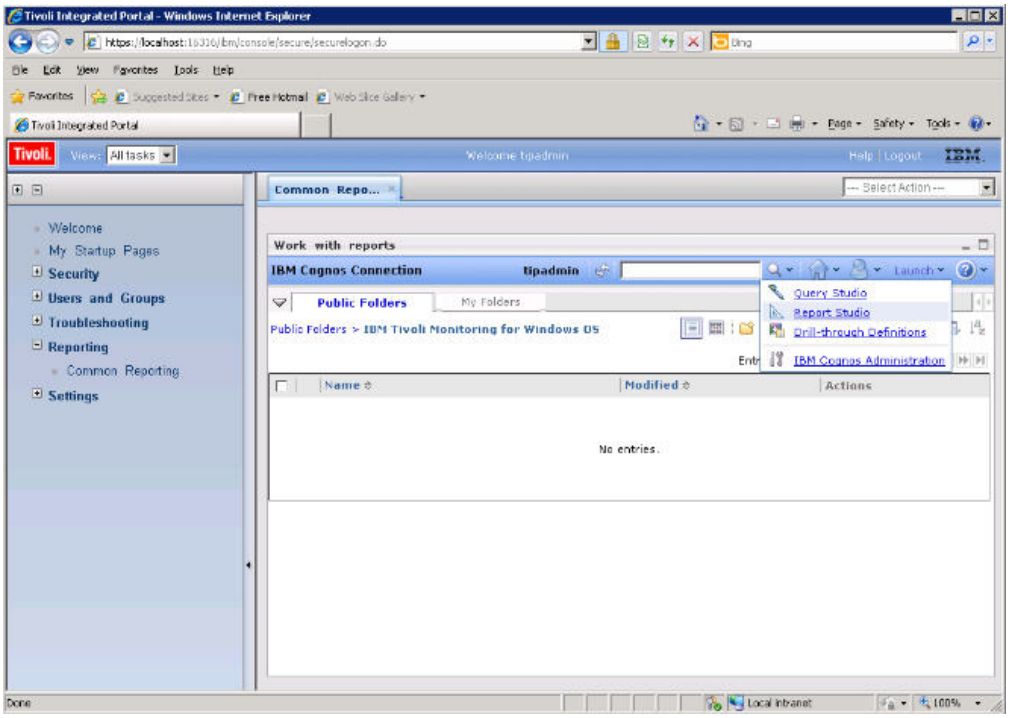


Figure 304. Selecting Report Studio

You can use the Report Studio to create new reports or templates, or you can modify an existing report or template.

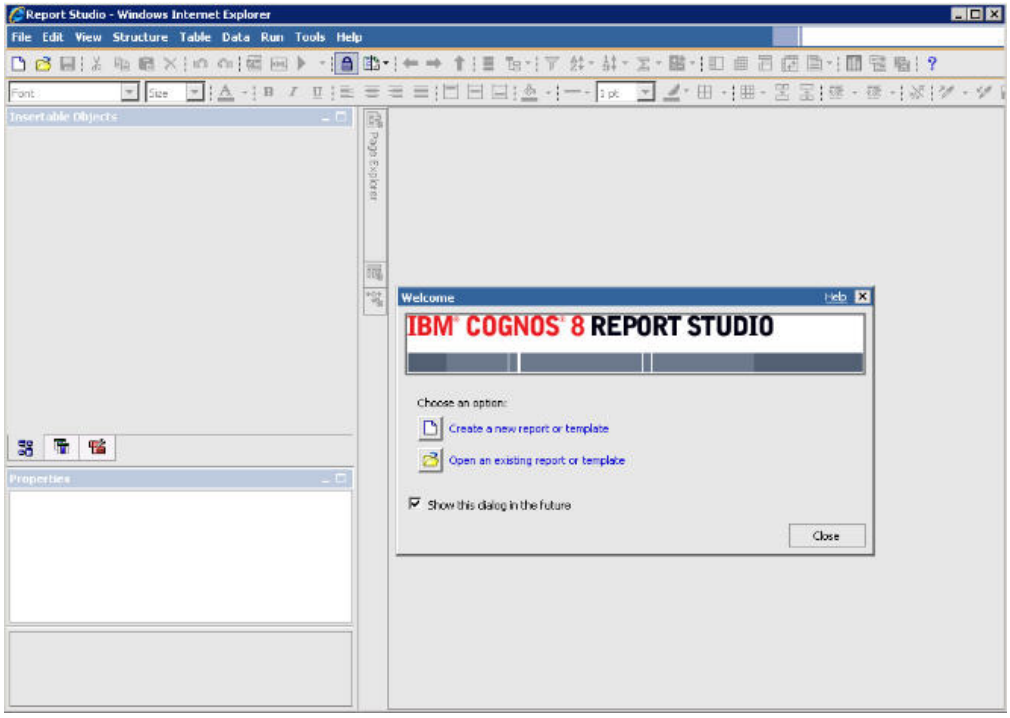


Figure 305. Report Studio

For more information, see the IBM Tivoli Common Reporting Information Center.

Exporting reports and data models from Tivoli Common Reporting

1. Log in to the Tivoli Common Reporting.
2. Navigate to Public Folders, and under **Reporting** in the left panel select **Common Reporting**.
3. In the Work with reports section, click on the Launch drop-down menu and select **IBM Cognos Administration**.
4. Click the **Configuration** tab.
5. Click **Content Administration**.

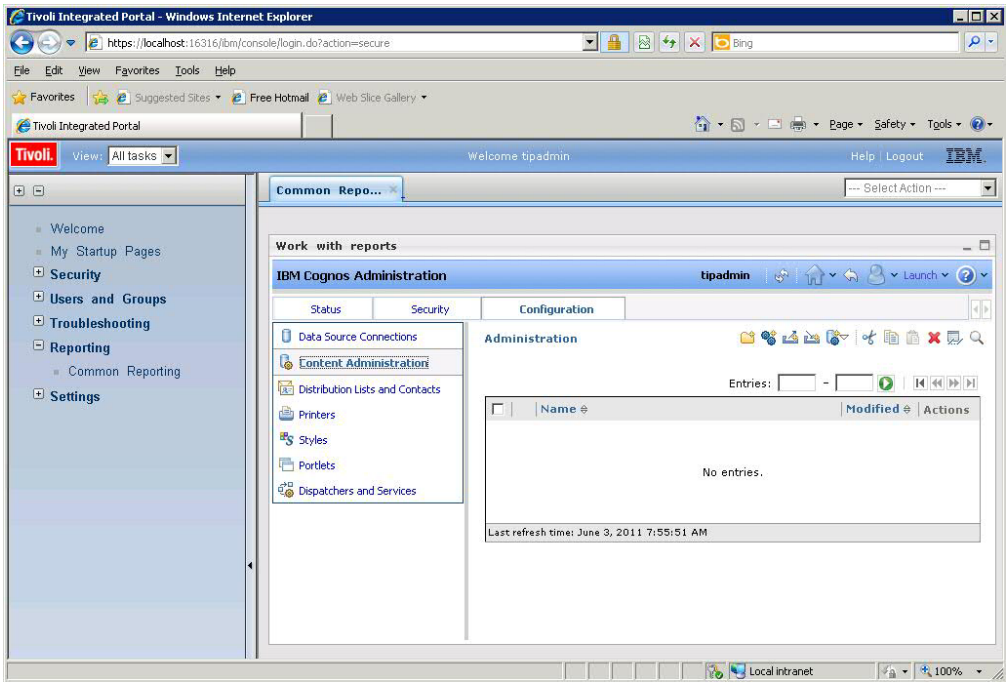


Figure 306. Content Administration tab

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 the **Select public folders and directory content** radio button.
9. In the Public Folders dialogue, click the **Add...** link.
10. Move your agent package to **Selected entries**.
11. On the last page of the wizard, select **Save Only**. Once the wizard completes, the report package is listed on the Content Administration tab.
12. On the Content Administration tab Figure 307 on page 609, click the Run button (green arrow) to create the ZIP file.

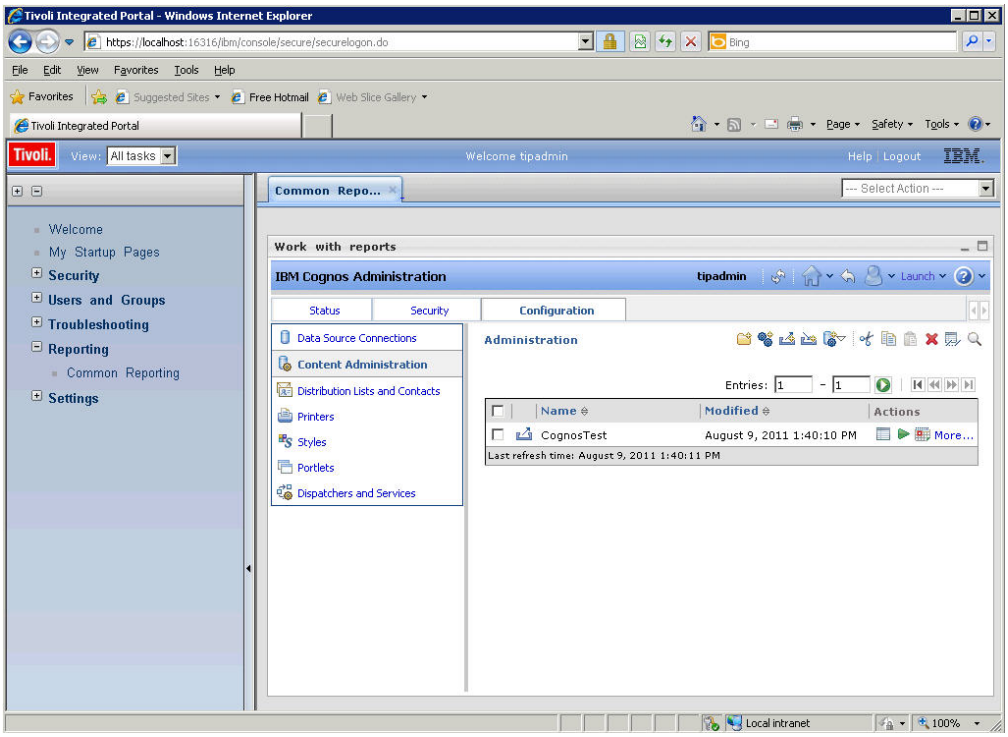


Figure 307. Content Administration tab with agent package listed

13. The ZIP file 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`

For more information about exporting reports, see Exporting Cognos report packages in the *Tivoli Common Reporting User's Guide*.

Importing reports into Agent Builder

Once the report package has been exported from Tivoli Common Reporting, it can be imported into the Agent Builder project. The report package can then be included in the agent install image.

1. Right-click on the agent project in the Agent Builder.
2. Select **IBM Tivoli** > **Import Report Package**.

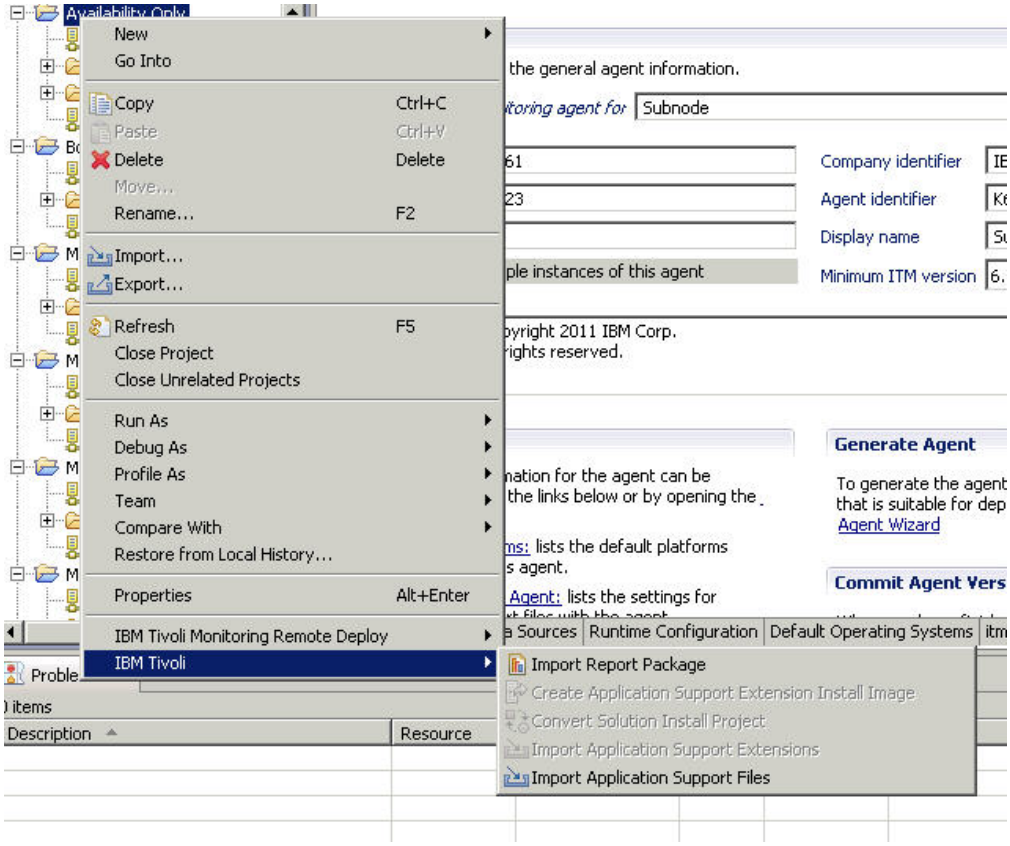


Figure 308. Importing report packages

3. In the Import Report Package window (Figure 309), select the Database Type on which the report package was created.

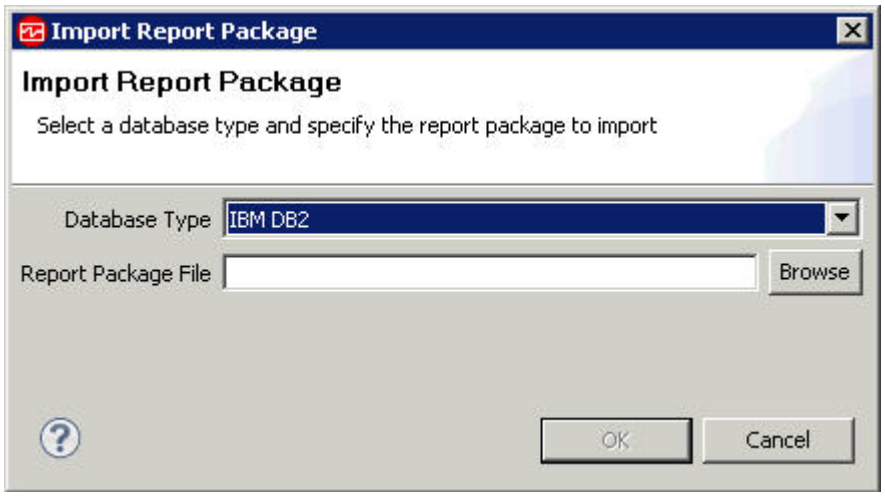


Figure 309. Import Report Package window

4. Enter the fully qualified path to the report package, or click **Browse** to select it.
5. Click **OK**.
6. The report package will now appear in the agent project under the `reports/`*`dbtype`* directory.

Note: If you create report packages that are database specific you need to import each package into the Agent Builder.

Installing reports from an agent package into Tivoli Common Reporting

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/Kxx/reports/cognos_reports/itmkxx/packages`
2. Copy the reports zip file into the TCR 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. Navigate to Public Folders, and under **Reporting** in the left panel select **Common Reporting**.
5. In the Work with reports section, click on the Launch drop-down menu and select **IBM Cognos Administration**.
6. Navigate to the **Configuration** tab, and open the **Content Administration** section.
7. Click **New Import** to create a new package import.
8. Select the agent's reports package.
9. Select the public folders you want to import.
10. Select save.
11. Click the Run button (green arrow) to perform the import.

For additional 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

If you determine that you need to create a new 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 15. When generating the agent, ensure that you have selected the **Generate an ITM 5x mapping file** option in the Generate Agent Wizard. For more information, see "Installing the agent locally" on page 377.

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 just these possible respects from that for a standard resource model:

- When performing the upgrade, the resource model mapping file used by the assess command must be created, either 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 this mapping file you might need to create a Java resource model plug-in class, which you can use to add your own coding to the standard evaluation used by the Agent Builder
- If the resource model data collection is different from that used by the standard resource models, you might also have to create a new agent to collect the data, using the agent builder. For this agent you will also need to create an agent mapping file, to enable the migration to work correctly.
- If the data collection in the agent cannot be provided by standard operating system facilities, you might need to create a Java agent plug-in class to do the collection

When these items have been developed and tested, the migration follows exactly the same steps as the migration of standard resource models.

Appendix G. ICU regular expressions

The following appendix is extracted from the *ICU User Guide*. This appendix 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 53. 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 control-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.

Table 53. Regular expression metacharacters (continued)

Character	Description
\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.
\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 six hex digits may be supplied.
\xhh	Match the character with two 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 one character from the set. See UnicodeSet for a full description of what may 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 54. 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 one times. Prefer one.
{n}	Match exactly n times

Table 54. Regular expression operators (continued)

Operator	Description
{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 one times. 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)
++	Match 1 or more times. Possessive match.
?+	Match zero or one times. 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. Somewhat 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.

Table 54. Regular expression operators (continued)

Operator	Description
(?<= ...)	Look-behind assertion. True if the parenthesized pattern matches text preceding the current input position, with the last character of the match being the input character just before the current position. Does not alter the input position. The length of possible strings 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 preceding the current input position, with the last character of the match being the input character just before the current position. Does not alter the input position. The length of possible strings 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 following the setting. For example, (?i) changes to a case insensitive 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 55. 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 $@.

Table 55. Replacement text characters (continued)

Character	Description
$#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 may be specified at the time that an expression is compiled into a RegexPattern object, or they may be specified within the pattern itself using the (?ismx-ismx) pattern options.

Table 56. 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. Note that 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 "$" in a pattern. By default these match only at the start and end, respectively, of the input text. If this flag is set, "^" and "$" also match at the start and end of each line within the input text.

Appendix H. Non-agent bundles

Starting with Tivoli Monitoring V6.2.2, 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. For example, 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 desired systems.

Creating a file bundle

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Under **IBM Tivoli Monitoring Wizards**, select **Non-Agent Remote Deploy Bundle** (Figure 310).

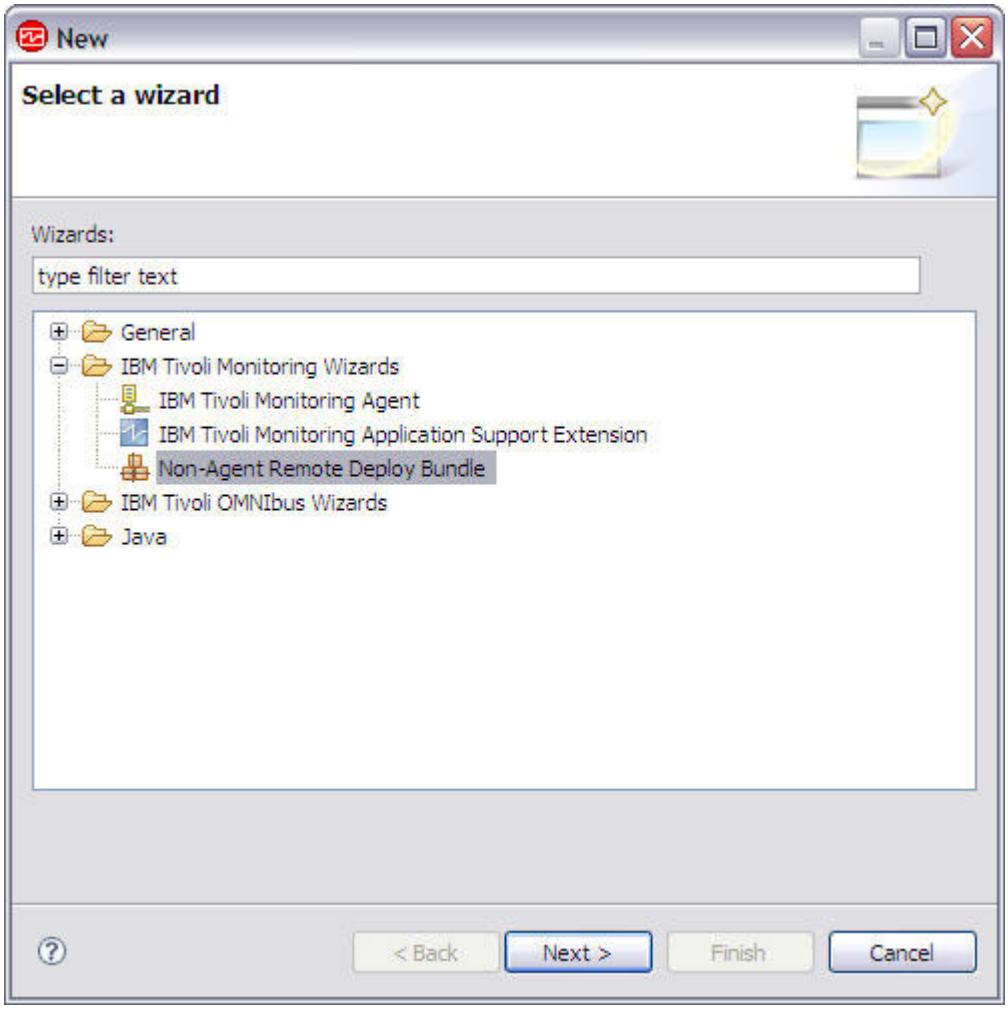


Figure 310. Select a wizard window

3. Click **Next**
4. In the **Project name** field, enter a name for your project.

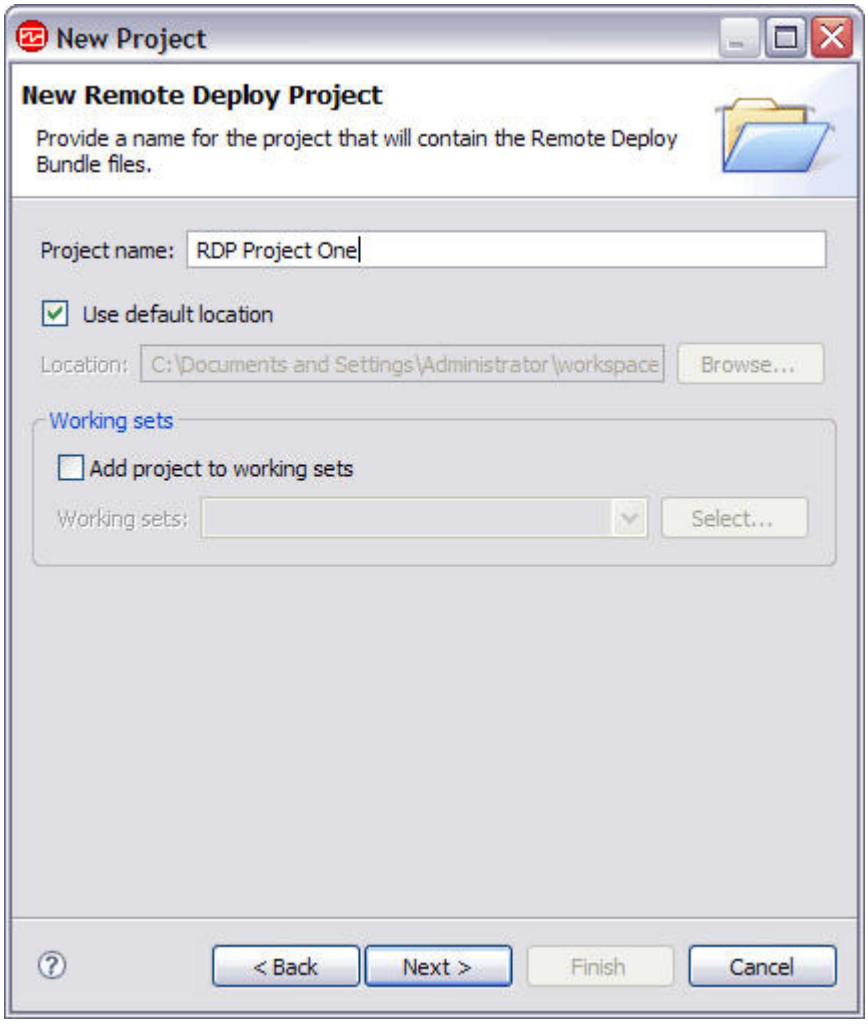


Figure 311. New Remote Deploy Project window

5. Click **Next**.
6. Complete the information in the Remote Deploy Bundle Information window (Figure 312 on page 623) as follows:

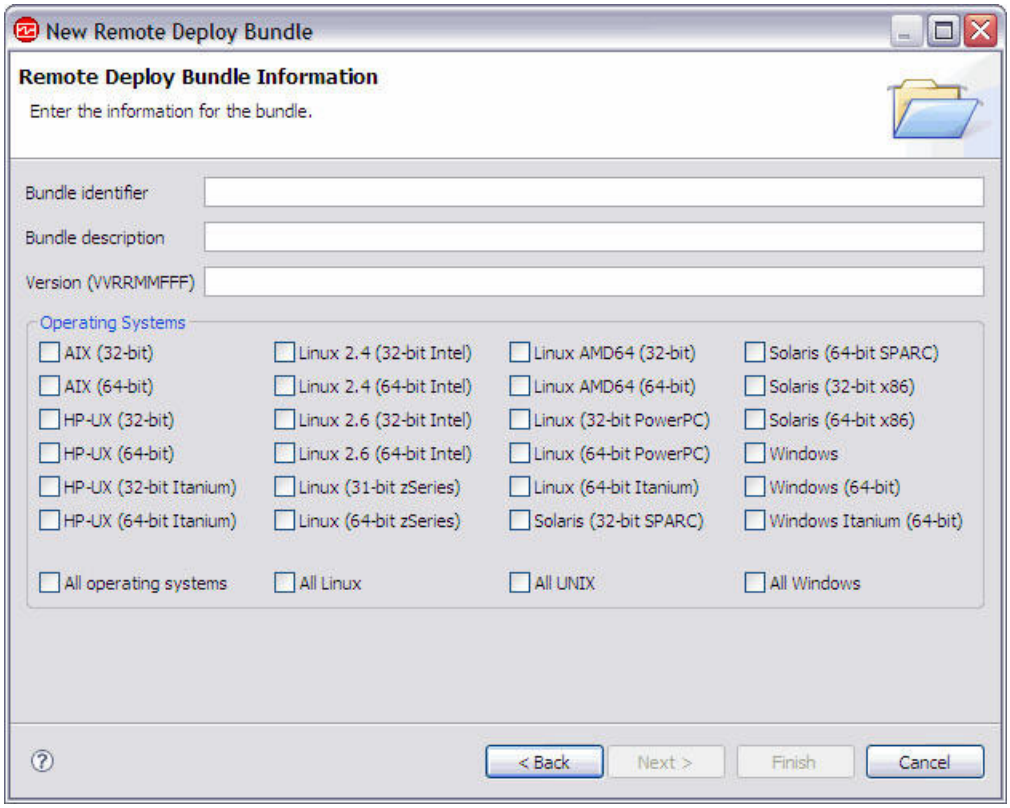


Figure 312. Remote Deploy Bundle Information window

a. In the **Bundle identifier** field, type an identifier that is a unique alphanumeric string between 3 and 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 new project in the workspace and open the Remote Deploy Bundle Editor.

Remote Deploy Bundle Editor

The Remote Deploy Bundle Editor (Figure 313 on page 624) provides information about the bundle for a project.

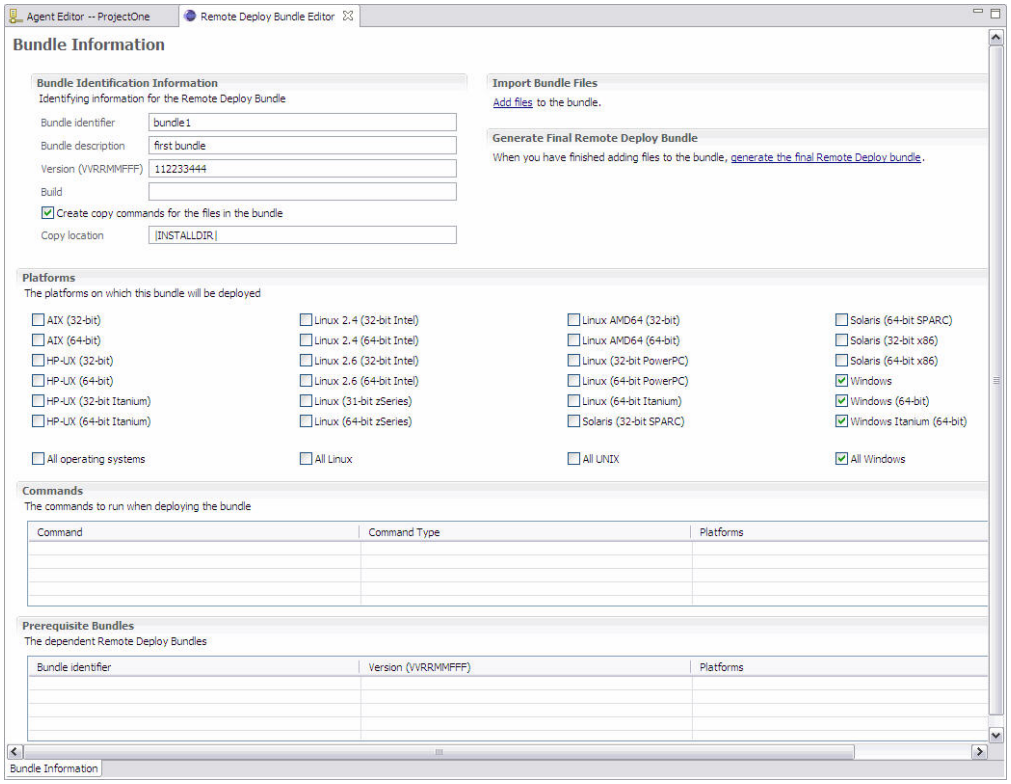


Figure 313. Remote Deploy Bundle Editor

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*|. This is a remote deploy variable that is set from the command-line deployment by specifying 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.

Adding commands to the bundle

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|. This is a special remote deploy variable that, if not set on the deploy command line, defaults to CANDLEHOME. To change the location specified by INSTALLDIR, specify the KDY.INSTALLDIR property when running the addSystem command.

The same directory structure 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 copy command generated copies the file to |INSTALLDIR|/config/myprod.config when the bundle is deployed.

In addition, you can specify additional commands to run during the deploy. To do this, click **Add** in the **Commands** section of the Remote Deploy Bundle Editor. In the Command window Figure 314, select the type of command (Preinstall, Install, Post-Install, or Uninstall) and then specify the command to run.

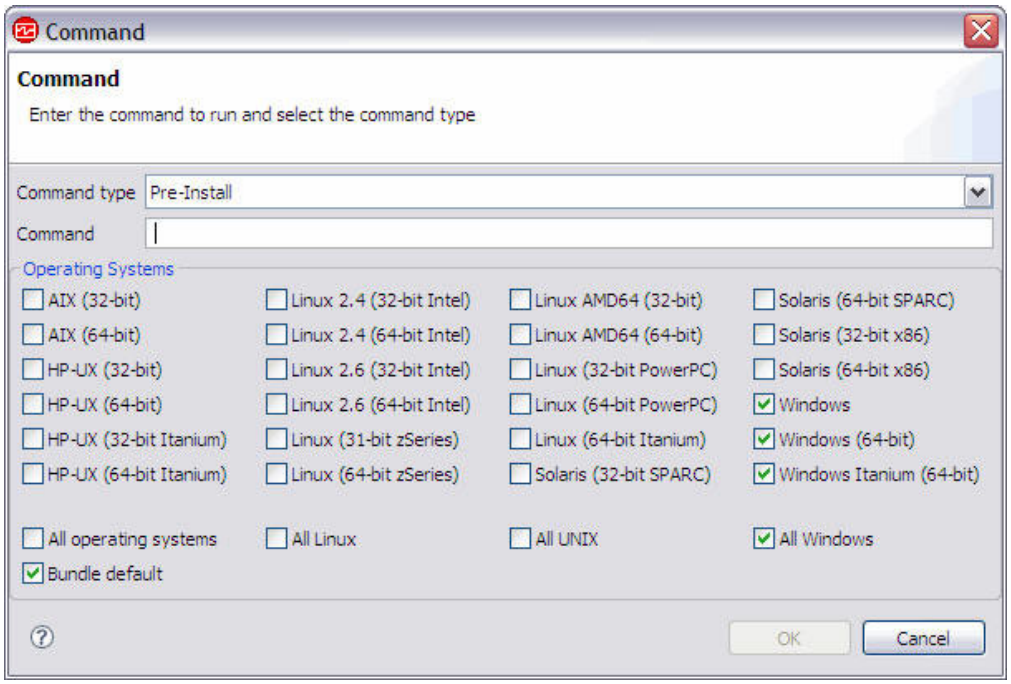


Figure 314. Command window

You must specify the fully qualified path to the command you want to run. For convenience, remote deploy provides a set of predefined variables. To reference the variable for a command, surround the variable with vertical bars, for example, |DEPLOYDIR|. Table 57 on page 626 describes the predefined variables for commands.

Table 57. Predefined Variables for Commands

Variable	Description
DEPLOYDIR	The temporary directory on the endpoint where the bundle resides during the deploy. For example, if you wanted to run myscript.sh, a script that was included in your bundle, you would 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.

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. To add a prerequisite, click **Add** in the **Prerequisites** section. In the New Prerequisite window Figure 315, enter the bundle identifier on which this bundle depends and the minimum version required. Finally, select the operating systems for which this prerequisite is required.

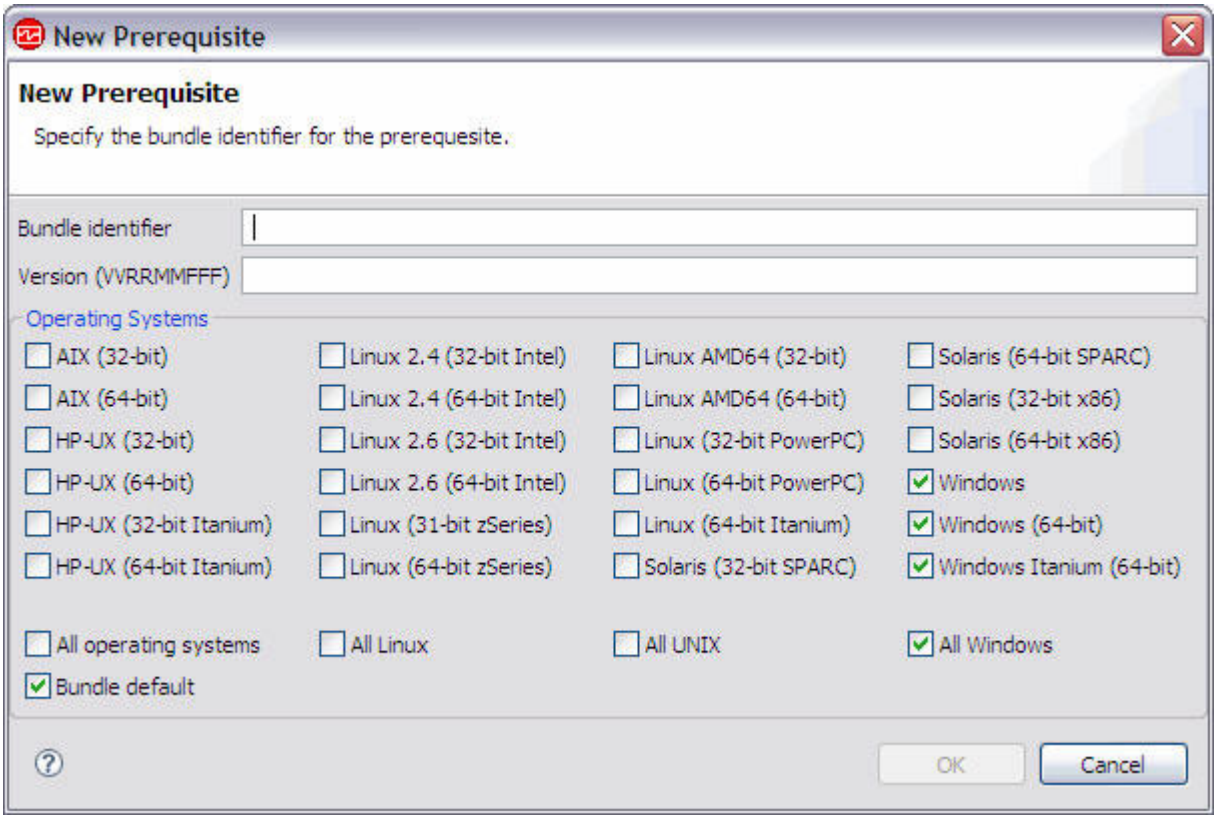


Figure 315. New Prerequisite window

Adding files to the bundle

To add files to the remote deploy bundle, do one of the following procedures:

- In the Bundle Editor, click **Add files to the bundle**.

  —OR—

- Right-click the project in the Navigator tree, then click **IBM Tivoli Monitoring Remote Deploy** > **Add Files to Bundle** as shown in Figure 316.

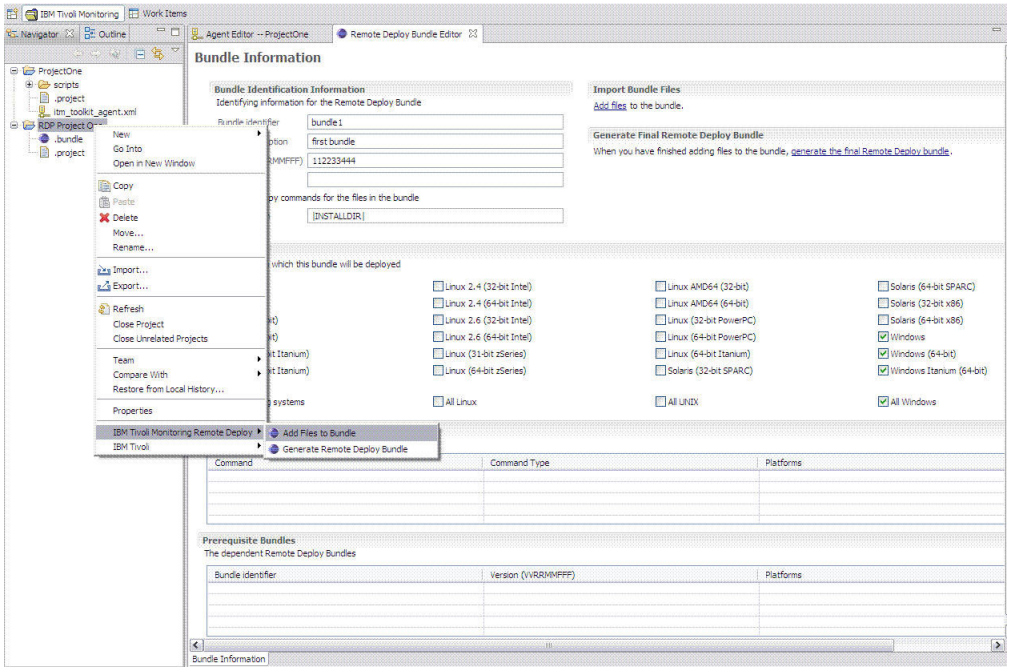


Figure 316. Adding files to the bundle or generating a bundle from the Navigator tree

Both of these actions display the Import Bundle Files window (Figure 317 on page 628).

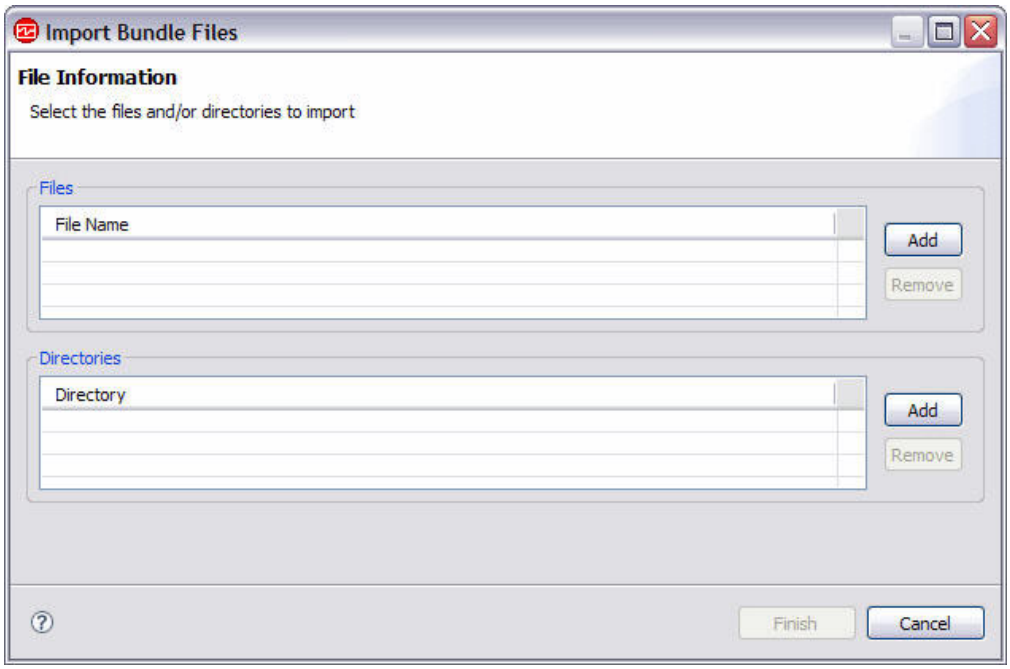


Figure 317. Import Bundle Files window

You can specify individual files or directories containing files. When you click **Finish**, the files or directories that are specified are copied into the project directory. The directory structure in the project is maintained when building the remote deploy bundle. If you want to have the Agent Builder generate the default copy commands, ensure that the files are in the appropriate directory structure in which they need to be when deployed.

Generating the bundle

To generate the remote deploy bundle, perform one of the following procedures to display the Generate Final Remote Deploy Bundle window (Figure 318 on page 629):

- In the Bundle Editor, click **Generate the final remote deploy bundle**.

  —OR—

- Right-click the project in the Navigator tree, then click **IBM Tivoli Monitoring Remote Deploy** > **Generate Remote Deploy Bundle** as shown in Figure 316 on page 627.

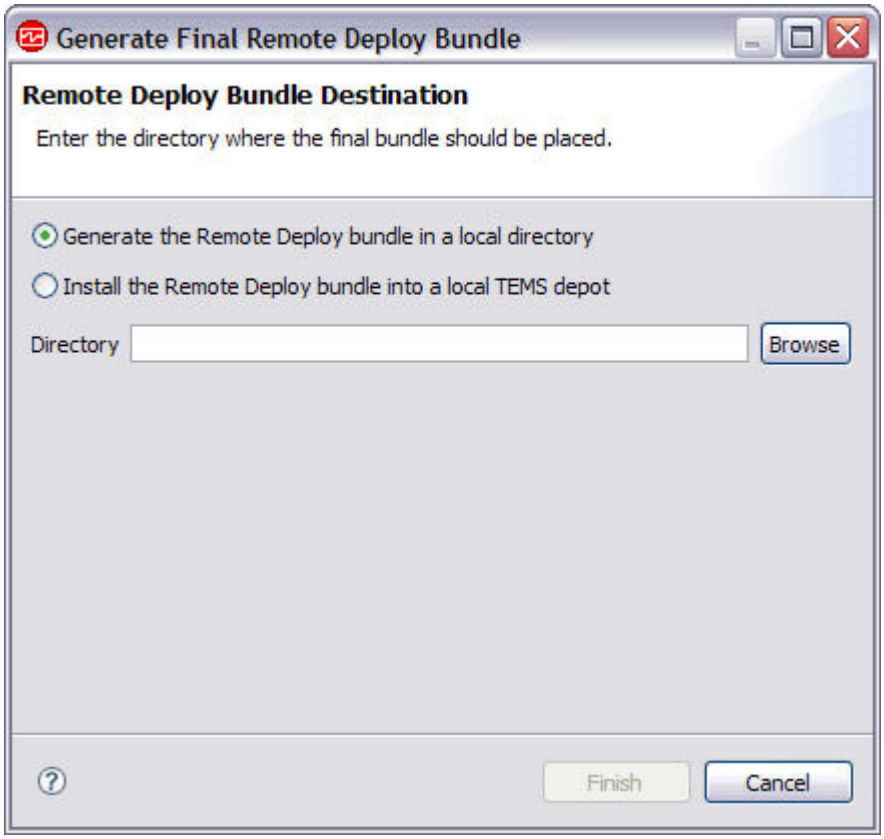


Figure 318. Generate Final Remote Deploy Bundle window

You can 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 location is correct before continuing.

  You must provide Tivoli Enterprise Monitoring Server login information in order 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.

  Note: When deploying 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 below:

  ```
  >tacmd viewDepot
  Product Code : MONITORINGCOLLECTION
  Version : 010000003
  ```

```
Description : MonitoringCollectionScripts
Host Type : WINNT
Host Version : WINNT
Prerequisites:
```

Note: You cannot remote deploy from the Tivoli Enterprise Portal Desktop or Browser. Remote deploy from the TEP 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. 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.

Use the following procedure to create deployable bundles:

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Under **IBM Tivoli OMNIbus Wizards**, select **Package Bundle**.
3. Click **Next**.

   You can then use the OMNIbus Install Bundle Wizard to create the bundles. For information about using this wizard, see the *IBM Tivoli Netcool/OMNIbus V7.3 Installation and Deployment Guide* in the Tivoli Netcool/OMNIbus information center at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.tivoli.nam.doc/welcome_ob.htm.

Appendix I. Dynamic file name support

Some application programs create an output file name that is subject to change 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 specifying the file-name pattern:

- IBM Tivoli Universal Agent dynamic file name syntax
- Regular Expressions

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 615. 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 specifying regular expression patterns, you must also select an option from the **When Multiple Files Match** drop-down list on the Advanced Log File Attribute Group Information page to specify the guidelines for selecting the most current matching file.

Note: Regular expressions is the preferred method to specify file-name patterns.

See Chapter 16, "Monitoring a log file," on page 179, Step 6 on page 181 for more information about how to configure advanced log file attribute group properties. For example, if you specified a file-name pattern:

```
d:\program files\logs\tivoli.*
```

This pattern searches for file names starting 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, 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.
{########}.*	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 6-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 starting 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 starting 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 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 3 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 using 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 pound 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 using one or more $'s to prefix a file name as in {$$$$$$}_abc.log, the number of $'s must exactly match the number of characters in that position in the file name. Also, you cannot specify $'s in multiple locations in a file-name pattern, for example, {$$$}b{$$$}.log will not match abc.log. Given these $ restrictions, use regular expression file-name patterns if there are an indeterminate number of characters in the file names you are searching for.
- The total number of pound signs and question marks 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 performing self-monitoring of Tivoli Monitoring components. For example, if you want to monitor the latest monitoring server log in the /opt/IBM/ITM/logs directory, 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 /logs directory, the Host1_ms_45205946-01.log file is selected for monitoring.

To precisely specify a file name consisting 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. Note that 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.

In more complex cases where a date field is embedded within a longer file name, and the date patterns listed in the previous examples are not sufficient, you can create patterns that mix pound 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', where you are interested in only the #-marked digits and the question marks serve as placeholders that allow you to 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 up.
- The currently monitored file no longer exists due to possible renaming or deleting.
- The existing file contents have changed due to 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

This appendix documents the configuration file used by the SNMP Data Provider to render trap information in a more easily readable form and to assign categories, severities, status, and source IDs to traps. It 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 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.

Using the HP OpenView `trapd.conf` file

The `trapcnfg` file is very similar in format, but not identical, to the HP OpenView Network Node Manager trap configuration file (`trapd.conf`), so 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 pound 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 319 on page 636 shows examples of the second and third types.

The first example in Figure 319 shows an enterprise definition record which defines enterprise OID 1.3.6.1.4.1.311.1.1.3.1.1 as being MS-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 being 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 enclosed within the delimiters SDESC and EDESC.

Examples of configuration record types 2 and 3

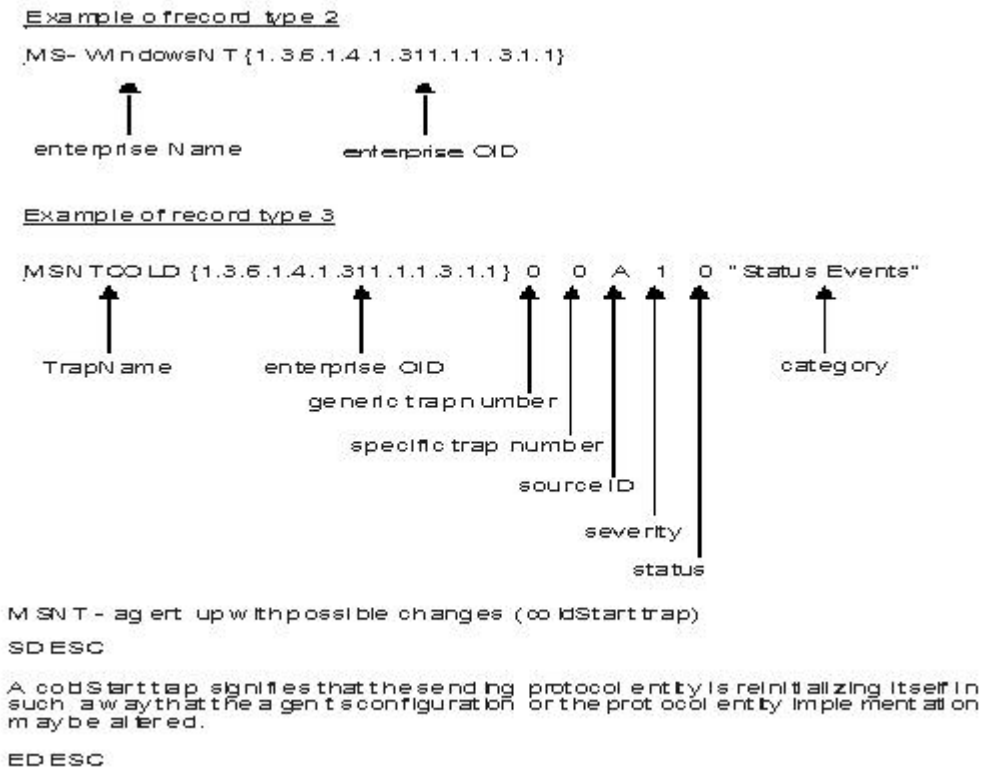


Figure 319. Examples of configuration record types 2 and 3

Defaults for the trapcnfg file

The tables in this section list the defaults supported by the SNMP Data Provider.

Supported categories

Table 58 shows the categories supported by the Agent Builder.

Table 58. 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 59 lists the severities supported by the Agent Builder.

Table 59. 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 60 shows the statuses defined in the Agent Builder configuration file.

Table 60. 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 61 lists the source IDs supported by `trapcnfg`.

Table 61. 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
P	Non-IP traps

Table 61. Source IDs supported by the SNMP Data Provider (continued)

Source ID	Description
r	Tralertd
s	Spappld
S	Security Agent
t	Xnmtrap
T	Trapd
V	Vendor related
?	Unknown

Appendix K. Take Action commands reference

This appendix contains an overview of Take Action commands, references for detailed information about Take Action commands, and descriptions of special Take Action commands that can be included in an Agent Builder monitoring agent.

About Take Action commands

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 referred to 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 or to send a text message to a cell phone.

Advanced automation uses policies to perform actions, schedule work, and automate manual tasks. A policy comprises a series of automated steps 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 that is displayed after the action completes or in a log file. 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 of the Take Action commands for this monitoring agent and a description of each command, see the Special Take Action commands section in this appendix and the information in that section for each individual command.

Special Take Action commands

An Agent Builder monitoring agent can recognize and perform 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 391).

The remaining sections of this appendix contain descriptions of these Take Action commands.

SSHEXEC action

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 configured to monitor the SSH target system. The command is run on the remote system that is represented by the Managed System Name.

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 in order 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 related to Tivoli Monitoring and to the commonly shared components of Tivoli Management Services. These publications are listed in the following categories:

- Tivoli Monitoring library
- Related publications

See *Tivoli Monitoring and OMEGAMON XE Products: Documentation Guide*, SC23-8816, for information about accessing and using the publications. You can find the *Documentation Guide* in the Tivoli Monitoring and OMEGAMON XE Information Center at http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/.

To find a list of new and changed publications, click **What's new** on the Welcome page of the Tivoli Monitoring and OMEGAMON XE Information Center. To find publications from the previous version of a product, click **Previous information centers** on the Welcome page for the product.

Tivoli Monitoring library

The following publications provide information about Tivoli Monitoring and about the commonly shared components of Tivoli Management Services:

- *Quick Start Guide*, GI11-8058

  Introduces the components of Tivoli Monitoring.
- *Installation and Setup Guide*, GC32-9407

  Provides instructions for installing and configuring 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.
- *Configuring the Tivoli Enterprise Monitoring Server on z/OS*, SC27-2313

  Gives detailed instructions for using the Configuration Tool to configure Tivoli Enterprise Monitoring Server on z/OS systems. Includes scenarios for using batch mode to replicate monitoring environments across the z/OS enterprise. Also provides instructions for setting up security and for adding application support to a Tivoli Enterprise Monitoring Server on z/OS.
- *Administrator's Guide*, SC32-9408

  Describes the support tasks and functions required for the Tivoli Enterprise Portal Server and clients, including Tivoli Enterprise Portal user administration.

- *High-Availability Guide for Distributed Systems*, SC23-9768

  Gives instructions for several methods of ensuring the availability of the Tivoli Monitoring components.
- 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*, SC32-9409

  Complements the Tivoli Enterprise Portal online help. The guide provides hands-on lessons and detailed instructions for all Tivoli Enterprise Portal features.
- *Command Reference*, SC32-6045

  Provides detailed syntax and parameter information, as well as examples, for the commands you can use in Tivoli Monitoring.
- *Troubleshooting Guide*, GC32-9458

  Provides information to help you troubleshoot problems with the software.
- *Messages*, SC23-7969

  Lists and explains messages generated by all Tivoli Monitoring components and by z/OS-based Tivoli Management Services components (such as Tivoli Enterprise Monitoring Server on z/OS and TMS:Engine).
- *IBM Tivoli Universal Agent User's Guide*, SC32-9459

  Introduces you to the IBM Tivoli Universal Agent, an agent of Tivoli Monitoring. The IBM Tivoli Universal Agent enables you to use the monitoring and automation capabilities of Tivoli Monitoring to monitor any type of data you collect.
- *IBM Tivoli Universal Agent API and Command Programming Reference Guide*, SC32-9461

  Explains the procedures for implementing the IBM Tivoli Universal Agent APIs and provides descriptions, syntax, and return status codes for the API calls and command-line interface commands.
- *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.

Documentation for the base agents

If you purchased 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 (for example, an OMEGAMON XE product) that includes the commonly shared components of Tivoli Management Services, you did not receive the base agents.

The following publications provide information about using the base agents.

- Operating system agents:
  - *Windows OS Agent User's Guide*, SC32-9445
  - *UNIX OS Agent User's Guide*, SC32-9446
  - *Linux OS Agent User's Guide*, SC32-9447
  - *i5/OS Agent User's Guide*, SC32-9448
  - *UNIX Log Agent User's Guide*, SC32-9471

- 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*, SC23-9767
  - *Warehouse Proxy Agent User's Guide*, SC23-9766
- 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:
  - *Monitoring Agent for IBM Tivoli Monitoring 5.x Endpoint User's Guide*, SC32-9490

Related publications

You can find useful information about the OMEGAMON XE monitoring agent products in the Tivoli Monitoring and OMEGAMON XE Information Center at http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/.

Other sources of documentation

You can also obtain technical documentation about Tivoli Monitoring and OMEGAMON XE products from the following sources:

- IBM Tivoli Integrated Service Management Library

  http://www.ibm.com/software/tivoli/opal

  The Integrated Service Management Library is an online catalog that contains integration documentation as well as other downloadable product extensions. This library is updated daily.
- Redbooks

  http://www.redbooks.ibm.com/

  IBM Redbooks®, Redpapers, and Redbooks Technotes provide information about products from operating system and solution perspectives.
- Technotes

  You can find Technotes through the IBM Software Support Web site at http://www.ibm.com/software/support/probsub.html, or more directly through your product Web site, which contains a link to Technotes (under **Solve a problem**).

  Technotes provide the latest information about known product limitations and workarounds.

Appendix M. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in Tivoli Monitoring Agent Builder enable users to:

- Use assistive technologies such as screen-reader software and a digital speech synthesizer to hear what is displayed on the screen
- Operate specific or equivalent features using only the keyboard
- Magnify what is displayed on the screen

In addition, the product documentation has been modified to include features to aid accessibility:

- All documentation available in both HTML and convertible PDF formats to give the maximum opportunity for users to apply screen-reader software.
- All images provided with alternative text so that users of the documentation with vision impairments can understand the contents of the images.

Using assistive technologies

Assistive technology products such as screen-readers, function with both the text-based and graphical user interfaces found in Tivoli Monitoring Agent Builder. Consult the assistive technology product documentation for specific information about using it to access command line or graphical interfaces.

Magnifying what is displayed on the screen

In all components of Tivoli Monitoring Agent Builder, you can magnify the windows in the product interfaces. Operating systems on which the product is run provide those facilities. For example, in a Windows environment you can change the settings to a lower resolution to enlarge the font sizes of the text. Information about these facilities is provided in the relevant operating system documentation.

Documentation in accessible formats

All user documentation is provided in HTML format, which can be read directly by assistive tools such as screen readers, or in *convertible* PDF format. Convertible PDF files are those that can be converted from PDF to HTML by the Adobe PDF to HTML converter. For information about converting PDF documents to HTML, refer to the Adobe book *Optimizing Adobe PDF Files for Accessibility*

Using alternative text

All documentation images are provided with an alternative text that can be read by assistive tools such as screen readers.

Appendix N. 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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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.

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 "Copyright and trademark information" at www.ibm.com/legal/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.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

abs 81
accessibility 645
Address attribute 564
Administrator authorization 8
agent
  name 20
  naming 20
Agent Builder
  DVD 8
  guidelines for installation 7
  starting 13
  uninstalling 10
Agent Editor pages 37
Agent Information page 38
Agent Watchdog 41
agent wizard, Starting the new 15
Agent XML Editor page 48
agent, committing version 49
agent, creating a basic 15
agent, organizing 30
agents
  after installation 382
  configuring and starting 382
  creating 15
  creation steps 13
  enterprise monitoring 397
  local installation 377
  modifying 35
  new files on your system 382
  regenerating 35
  removing 389
  solutions to problems 493
  system monitor 402
  testing 377
  uninstalling 389
AIX binary error logs
  monitoring 197
AIX Binary Log 548
Application Component attribute 529
Application Support Extension project
  creating 585
Application Support Extensions project
  adding support files 587
atof 81
atoi 81
attribute groups
  deleting 321
attributes 548, 563
  adding 322
  Address 564
  Application Component 529
  Availability 529, 571
  Average Collection Duration 538
  Cache Hit Percent 540
  Cache Hits 539
  Cache Misses 539
  Class Entry 549
  Command Line 532
  Compare String 559
  Counter Notifications 552
attributes *(continued)*
  Counter Value 554
  creating and editing 54
  DateTime 547
  deleting 322
  derived 62
  Description 550
  Device Entry 564
  editing 62
  EntryTime 551
  enumeration type 61
  Error Code 535
  ErrptTimestampp 548
  Event Category 545
  Event ID 545
  Event Log 544
  Event Source 544
  Event Type 544
  Full Name 530
  Functionality Test Message 533
  Functionality Test Status 533
  Gauge Notifications 554
  Gauge Value 556
  High Threshold 556
  HTTP 566, 569
  Identifier 548
  Interval 547
  Interval Unit 546
  Interval Unit Name 547
  Intervals Skipped 540
  JMX Event Attribute Groups 562
  joined 322
  Last Collection Duration 538
  Last Collection Finished 538
  Last Collection Start 537
  LocalTimeStamp 547
  Log File Summary 546
  Log Name 544
  LogFile 550
  LogName 550
  LogPath 551
  Low Threshold 556
  Message 545
  Message Attribute 563
  Modulus 553
  Monitor ID 552, 555, 557, 559
  Monitor Name 558
  Monitor Parameters 558
  Name 529, 565
  Node 529, 534, 540, 544, 546, 548, 552, 554, 557, 558, 562, 563, 566, 569, 571
  Node Description 565
  Node Status 565
  Node Type 565
  Notification Message 554, 557, 560
  Notification Time Stamp 554, 556, 560
  Notification Type 552, 555, 559
  Number of Collections 539
  numeric 58
attributes *(continued)*
  numeric options 58
  Object Name 535
  Object Name Time 570
  Object Response Tim 570
  Object Size 570
  Object Status 535
  Object Type 535
  Observed Attribute 553, 555, 559
  Observed MBean 552, 555, 559
  Occurrences 547
  Offset 553
  Page Faults Per Sec 531
  Page Objects 567
  Page Size 567
  Page Title 568
  Percent Privileged Time 532
  Percent Processor Time 532
  Percent User Mode Time 532
  Performance Object Status 534
  performing operations 62
  PID 532
  Query Name 534
  Refresh Interval 539
  Registered Monitors 557
  removing 62
  reordering 322
  ResourceName 549
  Response Code 568
  Response Time 564, 567
  Sequence Number 563
  Server Type 568
  Source 562
  Status 530, 568
  Status Timestamp 566
  string 57
  String Notifications 558
  String Value 560
  Subnode Affinity 571
  Subnode MSN 571
  Subnode Resource Name 572
  Subnode Type 572
  Subnode Version 572
  System 550
  Thread Count 531
  Thread Pool Active Threads 541
  Thread Pool Avg Active Threads 542
  Thread Pool Avg Job Wait 543
  Thread Pool Avg Queue Length 543
  Thread Pool Max Active Threads 542
  Thread Pool Max Queue Length 543
  Thread Pool Max Size 541
  Thread Pool Min Active Threads 542
  Thread Pool Min Queue Length 543
  Thread Pool Queue Length 542
  Thread Pool Size 541
  Thread Pool Status 540
  Thread Pool Total Jobs 543
  Threshold 553
  Time Generated 545
  timestamp 58

attributes *(continued)*
Timestamp 529, 534, 541, 546, 552, 555, 557, 558, 562, 564, 566, 569, 571
Total Object Size 567
Type 530, 549, 562
types 57
URL 566, 570
URL Alias 569
User Data 563, 569
Virtual Size 531
WMI 106
Working Set Size 531
authorization 8
availability
data types 1
availability filters
removing 322
average 81
Average Collection Duration attribute 538

B

basic agent, creating 15

C

Cache Hit Percent attribute 540
Cache Hits attribute 539
Cache Misses attribute 539
call sp_helpdb 251
call:2 sp_helpdb master 251
call[:*index*] *procedureName* [argument] ... 251
changes
saving 49
CIM
configuration 176
monitoring 173
Class attribute 549
Cognos data model
generating 597
Command Line attribute 532
command return code
monitoring 211
commands 523
configuring a subnode 350
generatelocal 525
generatemappingfile 526
generatezip 527
INSTALL_DIR/_uninst/uninstaller.bin 10
starting Agent Builder 13
tacmd configureSystem 352
commands, Take Action 639
commit agent version 45
committing
agent version 49
steps 50
Common Information Model 1
external data sources 1
monitoring 173
Compare String attribute 559
configuration
Agent Editor 367
CIM 176
configuration *(continued)*
custom 359
HTTP 270
JDBC 249
JMX 157
overriding custom 346
Secure Shell remote connection 371
subnodes 344, 347
Windows data sources 356
Windows remote connection 370
configuration problems
solutions 483
configuring
tuning 418
count 81
Counter Value attribute 554
creating a basic agent 15
creating subnodes
Agent Editor 343
Data Source Location window 335
Data Sources tree 339
New Agent Wizard 335, 339
cumulativeSum 81
custom configuration 359
overrriding 346

D

data
trace logs 476
types monitored by agent 1
data collection
Windows event log 201
data from different sources
monitoring 331
data providers
subnodes 333
data source 25
Data Source Definition page 45
data sources
AIX binary error logs 197
CIM 173
command return code 211
editing properties 53
HTTP 261
ICMP 255
Java API 297
Java Management Extensions 141, 167
JDBC 237
JMX 141, 167
joining 317
log file 179
moving existing 33
process 89
script output 225
Simple Network Management Protocol 121
SOAP 273
Socket 283
Windows event log 201
Windows Performance Monitor 113
Windows service 97
WMI 105
data types 25
DateTime attribute 547
defining
connections 94
derived attributes 62
editing 66
Description attribute 550
Device Entry attribute 564
documentation
See publications

E

eclipse
detailed information 2
working sets 21
Editor pages 37
edits
saving 49
enterprise monitoring agents 397
EntryTime attribute 551
enumeration attribute type 61
environment variables 424
Error Code attribute 535
ErrptTimestamp attribute 548
Event Category attribute 545
event data 422
event duplicate detection 405
event flow control 405
Event ID attribute 545
event identifiers, filtering 208
Event Source attribute 544
event source, filtering 206
event type 206
Event Type attribute 544
event type, filtering 206
eventThreshold 81
excessive data size 484
existing data sources
moving 33
expression matching flags, regular 619
expressions
ICU regular 615
regular metacharacters 615
external data sources 1
data types 1
navigator groups 31
external scripts
external data sources 1

F

fields
JMX 167
Process Monitor 90
file
silent.txt 9
uninstaller 11
file consolidation
file separation 223
files
agent trace 477
convertible PDF documentation 645
installation trace 477
log 413
other trace log 478
revolving message 418
trace logs 476

filter
attribute groups 317
filtering
event type 206
Filtering
attribute groups 66
filters
availability 322
flags
regular expression matching 619
Formula Editor 66
Full Name attribute 530
Functionality Test Message attribute 533
Functionality Test Status attribute 533
functions
abs 81
atof 81
atoi 81
average 81
count 81
cumulativeSum 81
eventThreshold 81
getenv 81
ipAddressToName 81
isSummaryEvent 81
itoa 81
last 81
max 81
min 81
NetWareTimeToTivoliTimestamp 81
occurrenceCount 81
replaceAll 81
replaceFirst 81
round 81
sqrt 81
stddev 81
StringToTivoliTimestamp 81
sum 81
summaryEvent 81
TivoliLogTimeToTivoliTimestamp 81
tokenize 81
UTCtoGMT 81
UTCtoLocalTime 81

G

Gauge Value attribute 556
generatelocal command 525
generatemappingfile command 526
generatezip command 527
getenv 81

H

High Threshold attribute 556
HTML help 645
HTTP 1
configuration 270
monitoring 261
tables 263
HTTP attributes
specific fields 264
Hypertext Transfer Protocol 1

I

IBM Support Assistant 517
IBM Tivoli Distributed Monitoring and Application Management Wiki 517
ICMP
monitoring 255
Identifier attribute 548
incorporating queries 397
incorporating situations 397
incorporating workspaces 397
information, additional
Take Action commands 639
install packages
procedure 380
INSTALL_DIR/_uninst/uninstaller.bin 10
installation
guidelines 7
log file 477
silent 9
installation problems
solutions 483
installing
Agent Builder 8
DVD 8
installing Agent Builder
after 9
installing agents
after you install 381
local 377
over an existing agent 35
instance names 113
Interval attribute 547
Interval Unit attribute 546
Interval Unit Name attribute 547
Intervals Skipped attribute 540
ipAddressToName 81
ISA 517
isSummaryEvent 81
itoa 81

J

Java API 1
monitoring 297
Java application programming interface 1
Java Database Connectivity
monitoring 237
Java Management Extensions 141
external data sources 1
JDBC 1
monitoring 237
JDBC configuration 249
JMX
configuration 157
external data sources 1
fields 167
monitoring 141
monitors 160
notifications 160
operations 160
joined
attributes 322
joining
data sources 317

K

key
missing attribute 486

L

Languages 13
last 81
Last Collection Duration attribute 538
Last Collection Finished attribute 538
Last Collection Start attribute 537
libraries
Tivoli Monitoring 641
local installation 377
LocalTimeStamp attribute 547
log file
monitoring 179
log files
external data sources 1
Log Name attribute 544
LogFile attribute 550
logging
agent trace logs 477, 478
installation log files 477
trace log files 476
LogName attribute 550
LogPath attribute 551
logs
message 430
trace 413
Low Threshold attribute 556

M

magnification, on-screen 645
max 81
Message attribute 545, 563
messages
log 430
metacharacters
regular expression 615
metrics
availability 502
service monitoring 502
zeros returned 502
MIB errors 126
MIB Parsing
SNMP 127
min 81
Modulus attribute 553
Monitor ID attribute 552, 555, 557, 559
Monitor Name attribute 558
Monitor Parameters attribute 558
monitoring
AIX binary error logs 197
CIM 173
command return code 211
Common Information Model 173
Java API 297
Java Database Connectivity 237
JMX 141
log file 179
multiple types of information 332
Perfmon 113
process 89
same data from different sources 331

monitoring *(continued)*
script output 225
SNMP 121
Socket 283
Windows Event Log 201
Windows Management Instrumentation 105
Windows Performance Monitor 113
Windows service 97
monitoring multiple types of information 332
monitoring the same data from different sources 331
monitoring types 25
monitors
JMX 160
multiple instances 24
multiple types of information
monitoring 332

N

Name attribute 529, 565
namespaces 106
naming the agent 20
navigator group
creating 327
navigator groups 31
NetWareTimeToTivoliTimestamp 81
new agent wizard, Starting 15
new in this release 2
Node attribute 529, 534, 540, 544, 546, 548, 552, 554, 557, 558, 562, 563, 566, 569, 571
Node Description attribute 565
Node Status attribute 565
Node Type attribute 565
Notification Message attribute 554, 557, 560
Notification Time Stamp 556
Notification Time Stamp attribute 554, 560
Notification Type attribute 552, 555, 559
notifications
JMX 160
Number of Collections attribute 539

O

Object Name attribute 535, 570
Object name field 114
Object Response Time attribute 570
Object Size attribute 570
Object Status attribute 535
Object Type attribute 535
objects, browsing 114
Observed Attribute attribute 553, 555, 559
Observed MBean attribute 552, 555, 559
occurrenceCount 81
Occurrences attribute 547
Offset attribute 553
OPAL documentation 643
operating systems 7
specifying 86
operations
JMX 160
organizing the agent 30

P

packages
creating 379
installing 380
Page Faults Per Sec attribute 531
Page Objects attribute 567
Page Size attribute 567
Page Title attribute 568
Percent Privileged Time attribute 532
Percent Processor Time attribute 532
Percent User Mode Time attribute 532
Perfmon 113
external data sources 1
Performance Monitor 1
Performance Monitor objects 114
PID attribute 532
Ping 563
platforms 7
prerequisites, software 7
private situations 402
creating 391
problem determination 413
procedures
stored 251
process
monitoring 89
process data unavailable 502
Process Monitor fields 90
processes
connections 94
product code
changing 51
product information
gathering for IBM Software Support 413
project
contents 21
name 13
project files
sharing 519
publications
OPAL 643
Redbooks 643
related 643
Technotes 643
types 641

Q

queries
creating 391
incorporating 397
preparing to create 391
Query Name attribute 534
quick reference, procedures 5

R

RAS trace parameters
setting 480
Redbooks 643
Refresh Interval attribute 539
regular expression matching flags 619
regular expression metacharacters 615
remote
process browsing 94
removing an agent using the Tivoli Enterprise Portal 389
removing an agent without using the Tivoli Enterprise Portal 390
replaceAll 81
replaceFirst 81
Report creation
solutions to problems 513
resource models
upgrading 613
ResourceName attribute 549
Response Code attribute 568
Response Time attribute 567
Response Time Entry attribute 564
round 81
rules for naming agents 20
Runtime Configuration Information page 47

S

same data from different sources
monitoring 331
Sampled Data 418
Saving your edits and changes 49
script data sources
subnode configuration 357
script output
monitoring 225
Secure Shell remote connection 371
Sequence Number attribute 563
server definitions 397
Server Type attribute 568
setup guidelines 5
silent installation 9
silent uninstallation 11
silent.txt file 9
Simple Network Management Protocol 1, 121
Simple Object Access Protocol 1
situations
creating 391
exporting 397
importing 397
incorporating 397
preparing to create 391
private 402
SNMP
MIB Parsing 127
monitoring 121
SOAP 1
monitoring 273
Xpaths 273
Socket 1
monitoring 283
Software Support 517
Source attribute 562
sp_helpdb procedure 251
space requirements 7
special
Take Action commands 639
sqrt 81

SSHEXEC
action 640
starting the Agent Builder 13
Starting the new agent wizard 15
Status attribute 530, 568
status of subnodes 333
Status Timestamp attribute 566
status unknown 502
stddev 81
stored procedures 251
Oracle 251, 252
samples 251
sp_helpdb 251
String Value attribute 560
StringToTivoliTimestamp 81
Subnode Affinity attribute 571
subnode configuration
overcoming limitations 347
script data sources 357
Subnode MSN attribute 571
Subnode Resource Name attribute 572
Subnode Type attribute 572
Subnode Version attribute 572
subnodes
Agent Editor 343
configuration 344
configuration properties 347
configuring from the command line 350
creating 331
creating as first component 335
data providers 333
from Data Sources tree 339
how they work 31
in Navigator tree 32
script data sources 357
status 333
tasks accomplished with 31
Windows data sources 356
sum 81
summaryEvent 81
support assistant 517
System attribute 550
system monitor agents 402
creating situations 391

T

tacmd configureSystem command 352
Take Action Command
SSHEXEC 640
Take Action commands
creating 391
more information 639
overview 639
special 639
Technotes 643
Thread Count attribute 531
Thread Pool Active Threads attribute 541
Thread Pool Avg Active Threads attribute 542
Thread Pool Avg Job Wait attribute 543
Thread Pool Avg Queue Length attribute 543
Thread Pool Max Active Threads attribute 542
Thread Pool Max Queue Length attribute 543
Thread Pool Max Size attribute 541
Thread Pool Min Active Threads attribute 542
Thread Pool Min Queue Length attribute 543
Thread Pool Queue Length attribute 542
Thread Pool Size attribute 541
Thread Pool Total Jobs attribute 543
Threshold attribute 553
Time Generated attribute 545
Timestamp attribute 529, 534, 541, 546, 552, 555, 557, 558, 562, 564, 566, 569, 571
Tivoli Monitoring Agent Builder, overview 1
TivoliLogTimeToTivoliTimestamp 81
tokenize 81
Total Object Size attribute 567
trace logging 413
configuration 416
example 415
format 414
location 414
trace logs 476
trap configuration 402, 635
trapcnfg configuration file 635
defaults in 636, 639
overview 635
types of records in 635, 636
trapcnfg defaults
supported categories 636
supported severities 636
supported source IDs 637
supported statuses 637
trapd.conf, using 635
troubleshooting 413
installation logs 477
uninstallation logs 477
Type attribute 530, 549, 562

U

uninstall command
UNIX 390
Windows 390
uninstall script 390
uninstallation
silent 11
uninstallation problems
solutions 483
uninstallation, log file 477
uninstaller file 11
uninstalling
Agent Builder 10
uninstalling agent 389
URL
monitoring 268
URL Alias attribute 569
URL attribute 566, 570
URLs file 268
User Data attribute 563, 569
UTCtoGMT 81
UTCtoLocalTime 81

V

version
agent 24
IBM Tivoli Monitoring 24
version of the agent
committing 49
limits 49
Virtual Size attribute 531

W

watchdog information 41
Windows data sources
configuration 356
Windows event log
filtering
event type 206
monitoring 201
Windows Event logs
data types 1
Windows Management Instrumentation 1
monitoring 105
Windows Performance Monitor 113
Windows remote connection 370
Windows service
monitoring 97
wizard, Starting the new agent 15
WMI
external data sources 1
monitoring 105
Working Set Size attribute 531
working sets 21
workspace directory
location 13
workspaces
creating 391
exporting 397
importing 397
incorporating 397
preparing to create 391

Z

zeros returned for metrics 502

Printed in USA

SC32-1921-13