Nexus IQ Server Documentation

Contents

1 How to Use This Book **1**

2 Downloads **3**

3 Requirements **4**

3.1 Nexus Solutions 4

3.2 Installation Requirements 6

3.2.1 IQ Server 6

3.2.2 IQ Server Web Application 8

3.2.3 REST API Versioning 8

3.2.4 Nexus IQ CLI 9

3.2.5 Sonatype CLM for Eclipse Requirements 9

3.2.6 IQ for IDEA Requirements 10

3.2.7 IQ for Visual Studio Requirements . . . 10

3.2.8 Nexus IQ for Bamboo Requirements . . . 10

3.2.9 Sonatype CLM for Hudson / Jenkins Requirements . . . 11

3.2.10 Sonatype CLM for Maven Requirements . . . 11

3.2.11 Sonatype CLM for Nexus Pro Requirements . . . 11

3.2.12 Sonatype CLM for SonarQube Requirements . . . 12

3.2.13 JIRA Notifications Requirements . . . 12

4 Quick Start Guide - Nexus Firewall 13

5 Quick Start Guide - Nexus Lifecycle 27

6 IQ Server Setup 35

6.1 Installation . . . 35

6.1.1 Starting the IQ Server . . . 36

6.1.2 License Installation . . . 37

6.1.3 IQ Server Directories . . . 39

6.1.4 Running the IQ Server as a Service . . . 39

6.2 Advanced Configuration . . . 42

6.2.1 Initial Configuration of the IQ Server . . . 43

6.2.2 Running the IQ Server Behind a HTTP Proxy Server 43

6.2.3 Setting the Base URL 44

6.2.4 Reverse Proxy Authentication 44

6.2.5 Appending a User Agent String 46

6.2.6 File Configuration 46

6.2.7 Email Configuration 47

6.2.8 Logging Configuration 47

6.2.9 HTTP Configuration 47

6.2.10 HTTPS/SSL 48

6.2.11 Anonymous Access 48

6.2.12 CSRF Protection 49

6.3 Backing Up the IQ Server 49

6.4 Upgrading the IQ Server 50

6.4.1 Upgrading from Version 1.17 or Earlier to Version 1.18 or Later 50

6.4.2 Upgrading from Version 1.15 or Earlier to Version 1.23 or Later 51

6.4.3 Upgrading from Version 1.16 or Earlier 52

6.4.4 Upgrading from Versions Earlier than 1.9.x 52

7 Security Administration **53**

7.1 Logging In . . . 53

7.2 Product Notifications . . . 54

7.3 User Management . . . 54

7.3.1 Changing the Admin Account Password . . . 55

7.3.2 Creating a User . . . 56

7.3.3 Editing and Deleting User Information . . . 57

7.4 LDAP Integration . . . 58

7.4.1 Configuring LDAP Server Connection . . . 58

7.4.2 LDAP Configuration Parameters . . . 59

7.4.3 Mapping LDAP Users . . . 61

7.4.4 LDAP User Parameters . . . 62

7.4.5 Mapping LDAP Groups . . . 63

7.4.6 LDAP Group Parameters . . . 64

7.4.6.1 Static Groups . . . 65

7.4.6.2 Dynamic Groups . . . 65

7.4.7 Verifying LDAP Configuration . . . 66

7.4.7.1 Test Connection . . . 66

7.4.7.2 Check User and Group Mapping . . . 67

7.4.7.3 Check Login 67

7.4.8 Reordering LDAP Servers 68

7.5 Role Management 69

7.5.1 Viewing Built-in Roles 69

7.5.2 Viewing Permissions of Built-in Roles 70

7.5.3 Understanding the Importance of Hierarchy 73

7.5.4 Managing Administrator Roles 73

7.5.4.1 Viewing Administrator Roles 73

7.5.4.2 Assigning Users to Administrator Roles 74

7.5.5 Managing Organizational Roles 75

7.5.5.1 Viewing Organizational Role Assignments 75

7.5.5.2 Assigning Users to Organizational Roles 76

7.5.5.3 Editing Organizational Role Assignments 77

7.5.5.4 Removing Organizational Role Assignments 79

7.5.6 Creating Custom Roles 80

7.5.7 Assigning Groups to Roles without Searching 83

7.5.8 Viewing Role Assignments 83

8 Organization and Application Management **85**

8.1 Hierarchy . . . 86
8.2 Inheritance . . . 86
8.3 Applications, Evaluations, and Reports . . . 87
8.4 The Root Organization . . . 88
8.4.1 Configuring the Root Organization . . . 88
8.4.2 Creating the Root Organization . . . 89
8.5 Viewing the Root Organization . . . 90
8.6 Creating an Organization . . . 91
8.7 Editing an Organization . . . 92
8.8 Deleting an Organization . . . 93
8.9 Creating an Application . . . 94
8.10 Editing an Application . . . 95
8.10.1 Selecting an Application Contact . . . 96
8.10.2 Removing an Application Contact . . . 97
8.10.3 Copying the Application ID to Clipboard . . . 97
8.10.4 Changing an Application ID . . . 98
8.11 Moving an Application . . . 99
8.12 Deleting an Application . . . 100

8.13 Viewing Organizations and Applications . . . 101

8.14 Managing Organizations and Applications . . . 102

9 Basic Policy Management 104

9.1 What is a Policy? . . . 104

9.2 Getting Started with Policies . . . 105

9.2.1 Downloading the Sample Policy Set . . . 105

9.2.2 Importing Policies . . . 106

9.3 Viewing Policies . . . 108

9.4 Creating Policies . . . 109

9.5 Editing Policies . . . 112

9.6 Deleting Policies . . . 114

9.7 Understanding the Parts of a Policy . . . 114

9.7.1 Policy Name . . . 115

9.7.2 Threat Level . . . 115

9.7.3 Inheritance . . . 116

9.7.4 Constraints and Conditions . . . 117

9.7.5 Actions . . . 121

9.7.6 Notifications . . . 123

9.7.7 JIRA Notifications . . . 125

9.8 Continuous Monitoring of Applications . . . 127

9.9 Proprietary Component Configuration . . . 130

10 Advanced Policy Management **134**

10.1 Component Labels . . . 135

10.1.1 Viewing a Component Label . . . 135

10.1.2 Creating a Component Label . . . 136

10.1.3 Editing a Component Label . . . 137

10.1.4 Deleting a Component Label . . . 138

10.2 License Threat Groups . . . 139

10.2.1 Viewing a License Threat Group . . . 139

10.2.2 Creating a License Threat Group . . . 141

10.2.3 Editing a License Threat Group . . . 143

10.2.4 Deleting a License Threat Group . . . 144

10.3 Application Categories . . . 145

10.3.1 Creating Application Categories . . . 146

10.3.2 Editing an Application Category . . . 147

10.3.3 Deleting an Application Category . . . 148

10.3.4 Assigning an Application Category 149

10.4 Manual Application Evaluation 150

11 The Dashboard **152**

11.1 Using the Dashboard 152

11.2 Filters 153

11.3 Results 156

11.3.1 Policy Violation Trends 156

11.3.2 Violations 158

11.3.3 Components 159

11.3.4 Applications 160

11.4 Viewing Component Details 161

11.5 Exporting Results 162

12 The Application Composition Report **164**

12.1 Accessing an Application Composition Report 166

12.2 Reviewing a Report 168

12.2.1 Summary Tab 170

12.2.2 Policy Violations Tab 172

12.2.3 Security Issues Tab 173

12.2.4 License Analysis Tab . . . 175

12.3 Printing and Reevaluating the Report . . . 176

12.4 The Component Information Panel (CIP) . . . 176

12.5 Resolving Security Issues . . . 183

12.5.1 Security Issues . . . 183

12.5.2 The Component Information Panel (CIP) . . . 185

12.5.3 Editing Vulnerability Status . . . 185

12.5.4 Matching to Violations . . . 187

12.6 License Analysis Tab . . . 188

12.6.1 License Threat Group . . . 189

12.6.2 License Analysis . . . 189

12.6.3 The Component Information Panel (CIP) . . . 190

12.6.4 Editing License Status and Information . . . 191

12.7 Component Identification . . . 193

12.7.1 Matching Components . . . 194

12.7.2 Managing Proprietary Components . . . 195

12.7.3 Claiming a Component . . . 197

12.8 Component Label Overview . . . 200

12.8.1 Where do component labels begin? . . . 200
12.8.2 Assigning a Label . . . 202
12.9 Waivers . . . 203
12.9.1 A Use Case for Waivers . . . 204
12.9.2 Adding a Waiver . . . 205
12.9.3 Viewing and Removing a Waiver . . . 206
12.10 Policy Reevaluation . . . 208
12.11 PDF Report . . . 209
12.11.1 Creating the PDF . . . 209
12.11.2 Reviewing the PDF . . . 210

13 Sonatype CLM and Repository Management 215

14 IQ for Nexus Repository Manager 217

14.1 Integrating Nexus Repository Manager 2.x and IQ Server . . . 218
14.1.1 Connecting to IQ Server . . . 218
14.1.2 Viewing Component Information . . . 220
14.1.3 Component Details . . . 224
14.1.4 Using Staging to Control Releases . . . 225
14.1.4.1 Staging Profile Configuration . . . 225

14.1.4.2 Policy Actions for Staging . . . 226
14.1.4.3 Policy Actions for Release Repositories . . . 227
14.1.5 Using Audit and Quarantine . . . 228
14.1.5.1 Configuring Audit and Quarantine . . . 229
14.1.5.2 Disabling Audit and/or Quarantine . . . 231
14.1.5.3 Releasing a Component from Quarantine . . . 232
14.1.5.4 Re-enabling Audit and/or Quarantine . . . 233
14.1.5.5 Viewing Repository Results . . . 234
14.2 Integrating Nexus Repository Manager 3.x and IQ Server . . . 235
14.2.1 Connecting to IQ Server . . . 235
14.2.2 Viewing Component and Assets Information . . . 237
14.2.3 Using Audit and Quarantine . . . 242
14.2.3.1 Configuring Audit and Quarantine . . . 243
14.2.3.2 Disabling Audit and/or Quarantine . . . 245
14.2.3.3 Releasing a Component from Quarantine . . . 246
14.2.3.4 Viewing Repository Results . . . 247
14.2.3.5 Granting Privileges to View Audit and Quarantine Summary Results . . . 249
14.3 Understanding Repository Results . . . 251

14.3.1 Using the Component Information Panel (CIP) . . . 253

14.3.2 Waiving Repository Policy Violations . . . 257

14.4 Managing Repositories . . . 259

14.5 Managing User Roles . . . 261

14.6 Removing a Repository in IQ Server . . . 261

15 Sonatype CLM and Continuous Integration **263**

16 Nexus IQ for Bamboo **265**

16.1 Install Nexus IQ for Bamboo . . . 266

16.2 Configure Nexus IQ for Bamboo . . . 268

16.3 Adding the IQ Analysis Task . . . 269

16.4 Reviewing IQ Policy Results . . . 271

17 Nexus IQ for Hudson/Jenkins **273**

17.1 Plugin Selection . . . 273

17.2 Integrating Nexus IQ for Hudson/Jenkins 1.x . . . 274

17.2.1 Installation . . . 274

17.2.2 Global Configuration . . . 275

17.2.3 Job Configuration . . . 277

17.3 Integrating Nexus IQ for Jenkins 2.x . . . 280
17.3.1 Installation . . . 280
17.3.2 Global Configuration . . . 281
17.3.3 Job Configuration . . . 281
17.3.3.1 Freestyle or Multi-Configuration Projects . . . 282
17.3.3.2 Pipeline Projects . . . 282
17.3.3.3 Return Value from Pipeline Build . . . 284
17.4 Inspecting Results . . . 287

18 IQ Server and IDEs **289**

19 Sonatype CLM for Eclipse **291**

19.1 Installing Sonatype CLM for Eclipse . . . 291
19.2 Configuring Sonatype CLM for Eclipse . . . 293
19.3 Using the Component Info View . . . 297
19.4 Filtering the Component List . . . 302
19.5 Searching for Component Usages . . . 303
19.6 Inspecting Component Details . . . 303
19.7 Migrating to Different Component Versions . . . 304

20 IQ for IDEA **308**

20.1 Installing IQ for IDEA 308

20.2 Configuring IQ for IDEA 309

20.3 Using the Component Info View 310

21 IQ for Visual Studio **312**

21.1 Installing IQ for Visual Studio 312

21.2 Configuring IQ for Visual Studio 312

21.3 Using IQ for Visual Studio 313

22 Sonatype CLM for SonarQube **314**

22.1 Installation 315

22.2 Configuration 317

22.3 Select the CLM Application 318

22.4 Add and Configure the Sonatype CLM Widget 319

22.5 Accessing the Application Composition Report 320

23 Nexus IQ CLI **321**

23.1 Downloading the Nexus IQ CLI 322

23.2 Locating Your Application ID 322

23.3 Evaluating an Application . . . 323
23.3.1 Additional Options . . . 324
23.4 Example Evaluation . . . 325
23.5 Using the Nexus IQ CLI with a CI Server . . . 326

24 Sonatype CLM for Maven 328

24.1 Evaluating Project Components with Sonatype CLM Server . . . 329
24.1.1 Authentication . . . 331
24.1.2 Simplifying Command Line Invocations . . . 332
24.1.3 Skipping Executions . . . 333
24.2 Creating a Component Index . . . 333
24.2.1 Excluding Module Information Files in Continuous Integration Tools . . . 334
24.3 Creating a Component Info Archive for Nexus Pro CLM Edition . . . 335
24.4 Using Sonatype CLM for Maven with Other IDEs . . . 336
24.4.1 Maven Plugin Setup . . . 336
24.4.2 IntelliJ IDEA . . . 337
24.4.3 NetBeans IDE . . . 339

25 REST APIs 342

25.1 Component Search REST APIs (v2) . . . 343

25.2 Component Details API (v2) . . . 346

25.3 Component Evaluation REST APIs (v2) . . . 349

25.4 Application REST APIs (v2) . . . 356

25.4.1 Deleting an Application . . . 365

25.5 Violation REST API (v2) . . . 365

25.6 Report-related REST APIs (v2) . . . 371

25.7 Accessing REST APIs via Reverse Proxy Authentication . . . 378

26 Webhooks **379**

26.1 Using Webhooks . . . 379

26.2 Configuring Webhooks . . . 380

26.2.1 Creating Webhooks . . . 380

26.2.2 Editing Webhooks . . . 381

26.2.3 Deleting Webhooks . . . 381

26.3 Working with HMAC Payloads . . . 382

26.4 Example Headers and Payloads . . . 383

26.4.1 Policy Management Event . . . 384

26.4.2 Application Evaluation Event . . . 385

26.4.3 Security Vulnerability Override Management Event . . . 386

26.4.4 License Override Management Event 387

A Copyright **388**

List of Figures

4.1 Import Policy Dialog 15

4.2 IQ Server Policy Actions 16

4.3 IQ Server Connection in Nexus Repository Manager 2.x 18

4.4 Nexus Repository Manager Capabilities Tab 19

4.5 IQ Server Connection in Nexus Repository Manager 3.x 20

4.6 Nexus Repository Manager Repository Results 22

4.7 IQ Server Repository Results 23

4.8 Nexus Repository Manager 3.x Repository Results 24

4.9 IQ Server Repository Results 24

4.10 Quickstart Repository Results 26

5.1 Import Policy Dialog 29

5.2 New Organization Dialog 30

5.3 New Application Dialog 31

5.4 Application Composition Report 33

5.5 Component Information Panel 34

6.1 Installing a Product License on IQ Server 37

6.2 IQ Server End User License Agreement Window 38

6.3 Installed Product License on IQ Server 38

7.1 Notifications Panel 54

7.2 Login 55

7.3 Create User 56

7.4 Edit User 57

7.5 Sample LDAP Server Configuration 59

7.6 User Mapping Example 62

7.7 Group Mapping Example 64

7.8 Dynamic Group Options 66

7.9 Testing LDAP Server 67

7.10 Checking User Mapping 67

7.11 Checking User Login 68

7.12 Reordering LDAP Servers Modal 68

7.13 Role and Permission Descriptions 70

7.14 Permissions of Built-in Roles 72

7.15 Viewing Administrator Roles 74

7.16 Assigning Users to Administrator Roles 74

7.17 Viewing Role Assignments 75

7.18 Assigning Users to Roles 77

7.19 Editing Role Assignments 79

7.20 Creating Custom Roles 82

7.21 Viewing Role Assignments 84

8.1 Root Organization 90

8.2 New Organization Dialog 92

8.3 Editing an Organization 93

8.4 Deleting an Organization 93

8.5 New Application Dialog 95

8.6 Editing an Application 96

8.7 Selecting a Contact 97

8.8 Changing an Application ID 99

8.9 Moving an Application 100

8.10 Deleting an Application . . . 101

8.11 Organizations and Applications Filter . . . 102

8.12 Organization & Policies View . . . 103

9.1 Import Policy Dialog . . . 107

9.2 New Policy View . . . 111

9.3 Edit Policy View . . . 113

9.4 Inheritance Settings . . . 116

9.5 Inheritance Settings for Repositories . . . 116

9.6 Constraints and Conditions . . . 118

9.7 Policy Actions . . . 121

9.8 Policy Notifications . . . 124

9.9 JIRA Notification Create . . . 126

9.10 JIRA Notification Configuration . . . 126

9.11 Continuous Monitoring View . . . 128

9.12 Continuous Monitoring Check Box . . . 129

9.13 Proprietary Component Configuration . . . 132

10.1 Component Label Section . . . 136

10.2 Using the Add a Label Button . . . 137

10.3 Editing a Component Label . . . 138

10.4 Viewing License Threat Groups . . . 140

10.5 Creating a License Threat Group . . . 142

10.6 Editing a License Threat Group . . . 144

10.7 Example of Applied Application Categories . . . 146

10.8 Using the Add a Category Button . . . 147

10.9 Editing an Application Category . . . 148

10.10 Assigning an Application Category . . . 150

10.11 Evaluate a Binary . . . 151

12.1 Summary Tab of the Application Composition Report . . . 165

12.2 Reporting Area . . . 167

12.3 Actions Menu . . . 168

12.4 The Four Tabs . . . 169

12.5 Security Issues Summary . . . 170

12.6 License Analysis Summary . . . 171

12.7 Policy Violations Tab . . . 172

12.8 Security Issues Tab . . . 174

12.9 License Analysis Tab . . . 175

12.10 Application Composition Report Buttons For Printing and Reevaluation 176
12.11 Component Information Panel CIP Example 177
12.12 CIP, Policy Section 178
12.13 CIP, Similar Section 179
12.14 CIP, Occurrences Section 180
12.15 CIP, Licenses Section 180
12.16 CIP, Vulnerabilities Section 181
12.17 CIP, Labels Section 181
12.18 CIP, Claim Component 182
12.19 CIP, Audit 182
12.20 Security Issues Tab 184
12.21 Component Information Panel (CIP) 185
12.22 Security Information Modal 186
12.23 Security Information Modal Additional Details 186
12.24 Editing Vulnerabilities 187
12.25 Example of Component with Security Issue, but No Policy Violation 188
12.26 License Analysis Tab 188
12.27 The Default License Threat Groups 189

12.28 Component Information Panel (CIP) . . . 190
12.29 Editing License Using the Select Option . . . 192
12.30 Unknown Component . . . 193
12.31 Filter and Matching Options . . . 194
12.32 Proprietary Component . . . 196
12.33 Add Proprietary Component Matchers . . . 197
12.34 Claim a Component . . . 198
12.35 Claimed Component Indicator . . . 199
12.36 Update or Revoke Claimed Component Indicator . . . 199
12.37 Component labels at the IQ Server Level . . . 200
12.38 Assigning a Label . . . 202
12.39 Waiver Visualization on Policy Tab . . . 204
12.40 Waiver Button . . . 205
12.41 Options to Apply Waiver to the Application or the Entire Organization . . . 206
12.42 View and Remove Waivers . . . 208
12.43 Application Composition Report Buttons For Printing and Reevaluation . . . 208
12.44 Summary Section of the Application Composition Report in PDF Format . . . 211
12.45 Policy Violations Section of the Application Composition Report in PDF Format . . . 212

12.46 Security Issues Section of the Application Composition Report in PDF Format 213

12.47 License Analysis Section of the Application Composition Report in PDF Format 213

12.48 Components Section of the Application Composition Report in PDF Format 214

13.1 The Central Role of A Repository Manager in Your Infrastructure 216

14.1 IQ Server Connection Tab in Nexus Repository Manager 2.x 219

14.2 Typical Search Results in Nexus Repository Manager 2.x 220

14.3 Nexus Repository Manager Search Showing All Versions 221

14.4 Accessing the Component Info Tab 221

14.5 Component Information Panel 222

14.6 CIP Text 223

14.7 CIP Graph 224

14.8 View Details 224

14.9 Staging Profile with an IQ Server Application Configured 226

14.10 Staging and Release Configuration for a Policy in IQ Server 226

14.11 Staging Repository Activity with IQ Server Evaluation Failure and Details 228

14.12 Capabilities Tab in Nexus Repository Manager 231

14.13 Release Quarantine 233

14.14 IQ Policy Violations Column 234

14.15 IQ Server Connection Tab in Nexus Repository Manager 3.x . . . 237
14.16 Associated Assets in Search Results in Nexus Repository Manager 3.x . . . 238
14.17 Asset Information in Nexus Repository Manager 3.x . . . 238
14.18 Viewing Component IQ . . . 239
14.19 Component Information Panel . . . 240
14.20 CIP Text . . . 241
14.21 CIP Graph . . . 241
14.22 View Details . . . 242
14.23 *IQ: Audit and Quarantine* Capability in Nexus Repository Manager . . . 245
14.24 Release Quarantine . . . 247
14.25 IQ Policy Violations Column in Nexus Repository Manager 3.x . . . 248
14.26 Nexus Repository Manager 3.x Capabilties Submenu . . . 249
14.27 Granting Privileges to View Audit and Quarantine Summary Results . . . 250
14.28 Repository Results . . . 251
14.29 The Component Info Tab . . . 253
14.30 The Licenses Tab . . . 254
14.31 The Vulnerabilities Tab . . . 255
14.32 The Labels Tab . . . 256

14.33 Waiving Policy Violations . . . 258

14.34 Waiving Policy Violations . . . 259

14.35 IQ Repositories . . . 260

14.36 The Configuration Section . . . 262

17.1 Jenkins Global Configuration Menu . . . 275

17.2 Global Configuration of Nexus IQ for Hudson/Jenkins 1.x . . . 276

17.3 Build Scan Configuration for a Build Step . . . 278

17.4 Nexus Jenkins Plugin Installation . . . 280

17.5 Nexus Jenkins Plugin Global Configuration . . . 281

17.6 Nexus Policy Evaluation . . . 282

17.7 Nexus Jenkins Plugin Pipeline Syntax . . . 283

17.8 Generate Pipeline Script . . . 284

17.9 Job Overview Page with Application Composition Report Link . . . 287

17.10 Left Menu with Application Composition Report Link . . . 288

19.1 Eclipse Dialog to Install New Software with Sonatype CLM for Eclipse . . . 292

19.2 Activating the Component Info View of Sonatype CLM for Eclipse . . . 293

19.3 Warning after initial installation . . . 294

19.4 Sonatype CLM for Eclipse Configuration Dialog . . . 295

19.5 Example Component Info View 297

19.6 Details for a Component in the Component Info View 299

19.7 Properties of a Component for a Version Range 300

19.8 Filter Dialog for the Component Info View 302

19.9 Example Component Details Display 304

19.10 Migrating to a Newer Component Version 305

19.11 Applying a Dependency Version Upgrade 306

19.12 Selecting Dependency Version or Property Upgrade 306

19.13 Applying a Property Upgrade 307

20.1 IDEA Plugin dialog to install IQ for IDEA 309

20.2 Nexus IQ plugin settings 309

20.3 Example Component Info View 310

21.1 IQ for Visual Studio Extension Settings 313

21.2 IQ for Visual Studio 313

22.1 SonarQube Overview 315

22.2 SonarQube Plugin Directory 316

22.3 CLM Settings Menu 317

22.4 SonarQube CLM Server Settings . . . 318

22.5 SonarQube Sonatype CLM Menu Item . . . 318

22.6 SonarQube Sonatype CLM Application Selection . . . 319

22.7 SonarQube Configure Widgets Button . . . 319

22.8 SonarQube Search for CLM Widget . . . 319

22.9 SonarQube Configure Sonatype CLM Widget options . . . 320

22.10 SonarQube Sonatype CLM Widget Example . . . 320

23.1 Application Overview and Application Identifier . . . 322

23.2 Violations Report After an Evaluation . . . 326

24.1 Creating a Maven Run Configuration for a CLM Evaluation in IntelliJ . . . 338

24.2 Maven Projects View with the CLM Evalulation Run Configuration in IntelliJ . . . 338

24.3 CLM for Maven Output in the Run Console in IntelliJ . . . 339

24.4 Project View with the `pom.xml` in NetBeans . . . 340

24.5 Maven Goal Setup for a CLM Evaluation in NetBeans . . . 340

24.6 CLM for Maven Output in the Output Window in NetBeans . . . 341

26.1 Creating Webhooks . . . 381

List of Tables

3.1 Nexus IQ Server Chapters Per Solution 4

9.1 Threat Levels 115

Preface

A book covering installation and usage of Nexus IQ Server—the technology that powers the solutions: Nexus Lifecycle, Nexus Auditor, and Nexus Firewall.

Last updated and published on 2017-08-07.

Chapter 1

How to Use This Book

This book is the first source of knowledge for using the different components of Nexus Lifecycle and Nexus Auditor. To get the most out of the various chapters and sections, as well as help familiarize yourself with how the book has been organized, take a look at the topics below.

Section Navigation
For all intents and purposes, our documentation is presented as a book with various chapters and sections. The navigation bar on the left-hand side of the documentation allows you to access specific chapters directly, while clicking on TOC (located at the top of any section) will display all chapters. Click Next or Previous (located at the top or bottom of each page) to cycle forwards or backwards through the book.

Search
Documentation for the latest release version of IQ has a search input box at the top right-hand side of the page that accesses relevant sections and pages by search term. This documentation feature is not available for previous release versions.

Interface Highlighting
When referencing a specific part of the user interface (e.g. a label or button). This will be highlighted by italics. E.g. "Click the *New Organization* button to create a new organization."

Code Segments
A code segment or command line input in a paragraph would be looking like this `mvn clean install`.

Larger code segments are encapsulated in a block:

```
$tar xfvz nexus-iq-server-1.31.0-01-bundle.tar.gz
x nexus-iq-server-1.31.0-01.jar
x demo.sh
x config.yml
x eula.html
x demo.bat
x README.txt
```

Callouts

The documentation uses specific callout blocks for notes, tips and important images:

Note
You can download Java from the Oracle website.

Tip
The IQ Server should be set up to run as a service.

Important
IQ Server requires Java 7.

In addition there are blocks for warnings and cautioning alerts:

Warning
Importing a policy will erase existing policies.

Caution
Upgrading components might require you to refactor your codebase.

Screenshots

Application screenshots and other images are typically included as numbered figures and referenced from the flowing text.

Chapter 2

Downloads

The latest release of the IQ Server and all applicable tools can be downloaded from the Sonatype support website and is available as *.tar.gz* or *.zip* archive. The contents of the two files are identical and you can choose to download either one. Successful download should result in files named `sonatype-clm-server-xyz-bundle.tar.gz` or `sonatype-clm-server-xyz-bundle.zip`, where xyz is the version of the latest release e.g. `1.11.0-01` or `1.10.2`.

Chapter 3

Requirements

3.1 Nexus Solutions

Nexus solutions—Lifecycle, Auditor, and Firewall—are powered by IQ Server. Each solution has a corresponding license that controls the availability of IQ Server tools and features. You may have access to a portion or all of IQ Server tools and features depending on the solution you purchased.

For information about your particular solution, see the table below. Chapters that apply to a particular solution are marked with an "X".

Table 3.1: Nexus IQ Server Chapters Per Solution

	Lifecycle	Firewall	Auditor
IQ Server Feature Chapters			
Downloads	X	X	X
Requirements	X	X	X
Quick Start Guide - Nexus Firewall		X	X
Quick Start Guide - Nexus Lifecycle	X		X
IQ Server Setup	X	X	X
Security Administration	X	X	X

Table 3.1: (continued)

	Lifecycle	Firewall	Auditor
Organization and Application Management	X	X	X
Basic Policy Management	X	X	X
Advanced Policy Management	X	X	X
The Dashboard	X	X	X
The Application Composition Report	X	X	X
IQ Server Integration Chapters			
Sonatype CLM and Repository Management	X	X	X
IQ for Nexus Repository Manager	X	X	
Sonatype CLM and Continuous Integration	X		
Nexus IQ for Bamboo	X		
IQ for Hudson/Jenkins	X		
Sonatype CLM and IDEs	X		
Sonatype CLM for Eclipse	X		
IQ for IDEA	X		
IQ for Visual Studio	X		
Sonatype CLM for SonarQube	X		
Nexus IQ CLI	X		
Sonatype CLM for Maven	X		
REST APIs	X	X	X
Webhooks	X	X	X

Tip
As you use this book, at the top of each chapter, you will see the solution(s) and license(s) required to use the features discussed in that chapter.

To learn more about Nexus solutions and licenses, see the knowledge base article Sonatype Product and Solutions Overview, or contact support@sonatype.com.

3.2 Installation Requirements

Each Nexus tool has a specific set of installation requirements, which are gathered here for reference.

- IQ Server
- Nexus IQ CLI
- Nexus IQ for Bamboo
- Sonatype CLM for Eclipse
- IQ for IDEA
- IQ for Visual Studio
- Nexus IQ for Hudson/Jenkins 1.x
- Sonatype CLM for Maven
- IQ for Nexus Repository Manager
- Sonatype CLM for SonarQube

3.2.1 IQ Server

The IQ Server is typically deployed on dedicated hardware. More specific hardware requirements are ultimately a function of the deployment architecture, the primary usage patterns and the scale of deployment.

With these influencing factors in mind, a general recommendation is provided as a starting point.

Development, test or evaluation deployments can be scaled smaller than the recommendations and will continue to function, while performance degradation may be observed.

CPU and RAM

We recommend a processor with at least 8 CPU cores and 8GB of RAM for initial setup. A minimum of 6GB of process space should be available to the IQ Server. Additional RAM can improve the performance due to decreased disk caching.

As an example a IQ Server deployment at Sonatype is using a Dual Intel Xeon E5620 with 2.4Ghz, 12M Cache, 5.86 GT/s QPI, Turbo, HT.

Disk

Storage requirements range with the number of applications projected to use the IQ Server. 500 GB to 1 TB of free disk space should provide more than adequate resources.

Tip

Monitoring disk-space usage will help you gauge the storage needs in your actual deployment and react to growing demands in time.

The IQ Server is an I/O intensive application and disk speed will affect the performance of the IQ Server considerably. We therefore recommend to use local drives or SAN usage. Usage of network mapped storage via NFS or similar is not recommended. It is important to consider the I/O load when running IQ Server in a virtual machine, especially when other virtual machines on the same host are running other I/O intensive applications e.g. the Nexus Repository Manager.

Operating System

Generally, any machine that can run a supported Sun/Oracle Java version should work. Refer to the Oracle documentation for specifics: Oracle JDK 1.7 to 1.8 and JRE 1.7 to 1.8 Certified System Configurations. The most widely used operating system for the IQ Server is Linux and therefore customers should consider it the best tested platform.

Ports

The following ports are used for communication and should be addressed according to your network firewall configuration.

8070

Used by all IQ Server clients to access the IQ Server. This port is configurable.

8071

Used by the local host or other IT monitoring tools for monitoring and operating functions. This port is optional and configurable. If omitted port 8081 will be used.

443

Used by the IQ Server to securely access Sonatype Data Services. This port is not configurable.

Java

Oracle Java 1.7 to 1.8 64 bit

3.2.2 IQ Server Web Application

Browser

In all cases JavaScript must be enabled for the chosen browser.

Internet Explorer

- IE 9
- IE 10
- IE 11

Firefox

- ESR (extended support release)
- "Stable"

Chrome

"Stable"

Safari (On OSX)

- 5.1.9 corresponding to OS X 10.6
- 6.0.4 corresponding to OS X 10.7 and 10.8
- 7.1 corresponding to OS 10.9

3.2.3 REST API Versioning

The IQ Server REST APIs are versioned. As a best practice we recommend using the latest version of the IQ Server and in addition to the latest version of the REST APIs. This ensures your system will take advantage of the latest features and improvements.

However, we also realize that users of previous versions need to maintain this compatibility even when there is an update. For this reason, we do provide support for previous versions based on the criteria below.

Supported API Versions

- Sonatype CLM 1.12 and earlier - **Only** Supports REST API v1
- Sonatype CLM 1.13 and later - Supports **Both** REST API v1 and v2
- IQ Server 1.17 to 1.22 - **Recommend** usage of v2 REST API only.
- IQ Server 1.23 and later - **Only** Supports REST API v2.

Identifying the version of the API is simple. Below we have provided an example using the REST API for retrieval of an organization ID.

```
http://localhost:8070/api/v2/organizations
```

As you can see, the `v2` located just after `api` indicates the version of the API. If you find that the API version you are using is not documented, and would like information on upgrading to the latest version you can contact our support team for assistance.

3.2.4 Nexus IQ CLI

The Nexus IQ CLI requires Java to be installed on the machine running the application, supports version 1.6 to 1.8.

3.2.5 Sonatype CLM for Eclipse Requirements

General Requirements , Eclipse
3.6 (Helios) to 4.6 (Neon)

m2e
(currently supports 1.0 - 1.7.0)

Java
Oracle Java 7 update 6 or newer (7u6+)

Browser Requirements:

In most cases the Eclipse plugin will render browser content using its own browser widget based on JavaFX.

Where JavaFX is not available the plugin will use the system browser. To properly render content the minimum requirement is Internet Explorer (IE) 9, or another modern browser such as Chrome / Firefox.

Note
Sonatype CLM functionality is not supported for those running Sonatype CLM for Eclipse with Internet Explorer 8.

Warning
These requirements assume an installation of the currently released version of the Eclipse plugin and the compatible IQ Server. Before picking a version please verify compatibility.

3.2.6 IQ for IDEA Requirements

General Requirements , IntelliJ IDEA

- 14.1.x
- 15.0.4 - 15.x
- 2016.1.1 - 2016.x

Java
Oracle Java 7 or newer with JavaFX

3.2.7 IQ for Visual Studio Requirements

Visual Studio

- Community 2015-2017
- Professional 2015-2017
- Enterprise 2015-2017

Microsoft .NET
4.5 or newer

3.2.8 Nexus IQ for Bamboo Requirements

Bamboo
5.5.0 to 5.14.4.1

Java
1.7 to 1.8

3.2.9 Sonatype CLM for Hudson / Jenkins Requirements

Hudson

- 2.2.1
- 3.0.0
- 3.1.2

Jenkins

- 1.447.2
- 1.532.1 (LTS)
- 1.550
- 2.32.1 (LTS)

Java
1.6 to 1.8

3.2.10 Sonatype CLM for Maven Requirements

Maven
2.1 to 3.3.9

Java
1.6 to 1.8

3.2.11 Sonatype CLM for Nexus Pro Requirements

Sonatype CLM for Nexus is a part of the Nexus CLM Edition, and does not require installation of an specialized components. However, the CLM capabilities do require a user have at least **Internet Explorer 9** or any other **modern browser**.

For information on Nexus requirements please see the support article detailing Sonatype Nexus System Requirement and the prerequisites section of the Nexus Book.

3.2.12 Sonatype CLM for SonarQube Requirements

SonarQube

- 3.7.4
- 4.3.2
- 4.4
- 4.5.6 (LTS)
- 5.0
- 5.1
- 5.2
- 5.3
- 5.4
- 5.5
- 5.6

Note
SonarQube has removed dashboards as of version 6.2

Java
1.6 to 1.8

3.2.13 JIRA Notifications Requirements

JIRA
6.3 to 7.x

Chapter 4

Quick Start Guide - Nexus Firewall

This guide can help you get IQ Server up and running for the purpose of trying out the associated Nexus Firewall functionality before installing it in your development environment. If you have an available Nexus Repository Manager Pro server available, you can expect to spend 15 to 30 minutes for installation and configuration, a bit longer if you don't.

To dive into Nexus Firewall a bit further, check out the Audit and Quarantine section of the Nexus Repository Chapter.

Note
To integrate Nexus Repository Manager with IQ Server you need Nexus Repository Manager Pro and IQ Server installed with the Repository Pro license that also supports Nexus Firewall. If you do not have a license contact us, and we'll be happy to assist.

Step 1: Installing and Starting IQ Server

Installing the IQ server is really a case of downloading the archived server, picking a location, and unpacking the contents. Since we won't be focused on mimicking a production experience, most laptop and desktop configurations should run IQ Server with no problem. If you are looking to plan for the future though, be sure to review the server requirements section of the Requirements chapter.

1. Create an installation directory in your desired location.
2. Download the latest version of IQ Server to the installation directory.
3. Extract the `tar.gz` or `.zip` file.

Once you've extracted the contents, follow the steps below to run IQ Server

1. Using a command line interface, switch to the nexus-iq-server bundle directory in your installation directory e.g. `nexus-iq-server-x.xx.x-xx-bundle`.
2. Run one of the following commands to start IQ Server:

   Linux or Mac: `./demo.sh`

   Windows: `demo.bat`
3. Open IQ Server in a browser using the default URL: http://localhost:8070
4. Log in using the default Administrator account:

```
Username: admin
Password: admin123
```

5. Install the required product license supplied to you by the Sonatype Support team.
   a. Click *Install License*.
   b. Navigate to the license file (`.lic`) and click *Open*.
   c. Click *I Accept* to accept the End User License Agreement.

Note
IQ Server needs access to an external data service to perform evaluations, which may be blocked in your internal environment. For a workaround, see Running IQ Server Behind a HTTP Proxy Server in the IQ Server documentation.

Step 2: Importing Sample Policies

Policy is at the core of IQ Server's automation capabilities. This is true for both Nexus Firewall and Nexus Lifecycle. While you can create a completely custom set of policies, importing the Sonatype

Sample Policy set set is the quickest way to get started. This set includes multiple policies for triggering violations on security vulnerabilities, licensing issues, architecture issues, and more.

1. In a separate browser tab or window, download the Sonatype Sample Policy set (`.json` file) from the IQ Server documentation.
2. In IQ Server, click the *Organization & Policies* icon on the IQ Server toolbar.
3. The *Root Organization* should be selected in the sidebar. Click the *Actions* menu and select *Import Policies*.
4. In the *Import Policies* dialog, click *Choose File*, select the `.json` file you downloaded, and click *Open*.
5. Click the *Import* button.

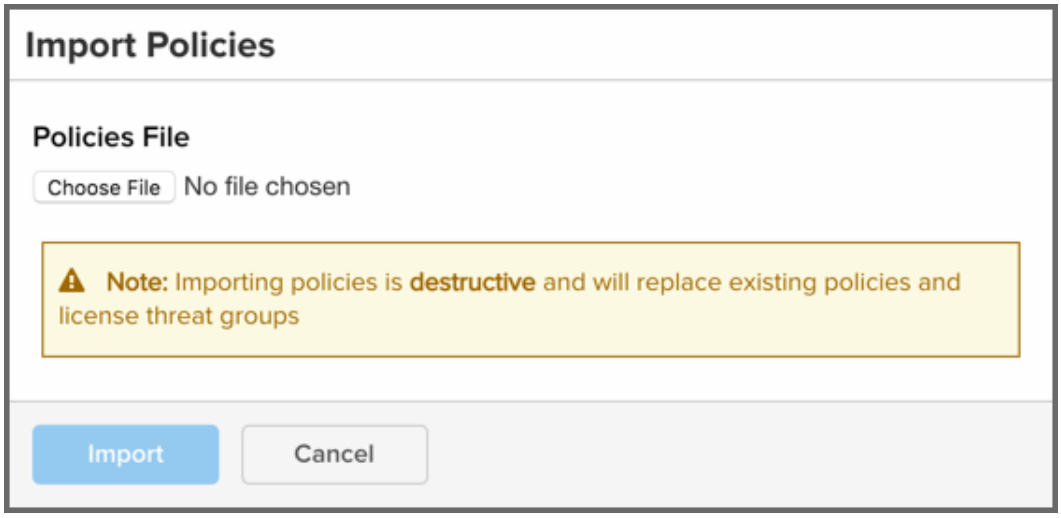


Figure 4.1: Import Policy Dialog

Step 3: Configuring Policy Actions

Policy Actions directly affect how IQ Server can automate processes in the available integrations when policy violations are encountered. In the case of Nexus Firewall, you can set an action to warn, which will audit, or simply display any violations. Alternatively you can set the action to Fail, which will quarantine, or block developers from accessing new components entering a repository that also violate the specified policy. To set Policy actions for the Proxy stage:

1. In IQ Server, click the *Organization & Policies* icon on the IQ Server toolbar.
2. Click on the *Root Organization* in the sidebar, and then click the policies section.
3. Click on the policy you want to add an action to, and in the Proxy column choose *Warn* (Audit) or *Fail* (Quarantine).
4. Click the *Update* button.

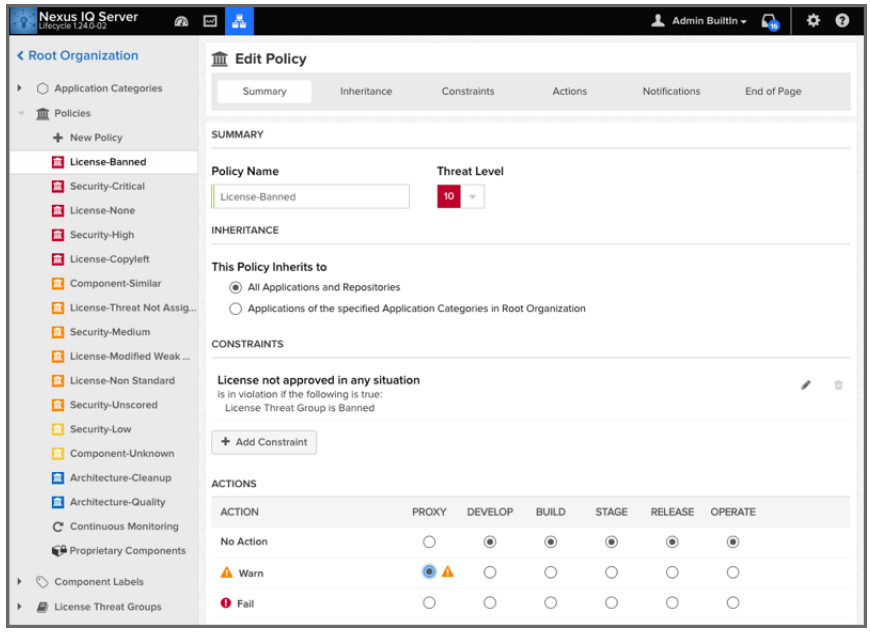


Figure 4.2: IQ Server Policy Actions

Note

When using the Fail action (Quarantine), the repository will need to be configured accordingly. In addition only new components entering the repository can be quarantined. Components with violations that already exist in repository will not be quarantined.

For additional information on what actions can be set and how they can affect automation, be sure to check out the actions section of our chapter on Policy Management.

Step 4: Nexus Repository Manager Configuration

IQ Server for Nexus Repository Manager allows you to integrate IQ Server's policy management and component intelligence features with proxy repositories in Nexus Repository Manager Pro. In order to do this, first you will need to configure the capabilities that allow for communication between IQ Server and Nexus Repository Manager. In addition, because Nexus Firewall is compatible with both Nexus Repository Manager 2.12.x or higher and 3.2.x or higher, there are specific instructions for each major version.

Configuring Nexus Repository Manager 2.x

There are two steps in order to allow IQ Server to interact with an instance of Nexus Repository Manager, and evaluate repositories. First, you need to configure the IQ Server connection:

1. In Nexus Repository Manager 2.x, click the *IQ Server Connection* menu item under *Administration*.
2. Enter the URL for your IQ Server installation.
3. Select an *Authentication Method*:

   User Authentication: Enter the username and password.

   PKI Authentication: Delegate to the JVM for authentication.
4. Click *Save*.

If successfully connected, a list of available applications in IQ Server displays in the *Server Connection* tab.

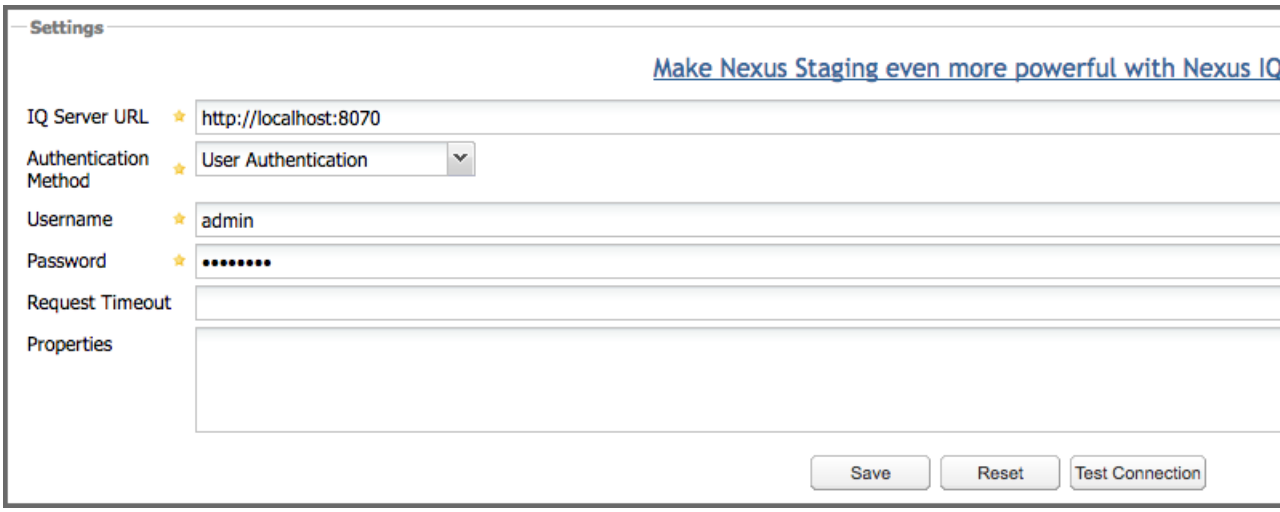


Figure 4.3: IQ Server Connection in Nexus Repository Manager 2.x

Note
For this quick start guide, using the default admin credentials is acceptable. However, for a real implementation, you would want to create a unique user for this integration, making sure to review the section on Section 7.5 in the Security Adminstration chapter.

Next, add the Audit and/or Quarantine capability for each repository you want to evaluate. To configure Audit and/or Quarantine:

1. In Nexus Repository Manager, click *Capabilities* on the *Administration* menu.
2. Click the *New* button on the *Capabilities* tab. The *Create new capability* dialog is displayed.
3. In the *Type* list, choose *IQ: Audit and Quarantine*.
4. Select a specific proxy repository to scan, for example *Central*.
5. Click *Add*.

An audit of the selected repository automatically starts. Nexus Repository Manager contacts IQ Server and evaluates the components within the selected repository against any associated policy.

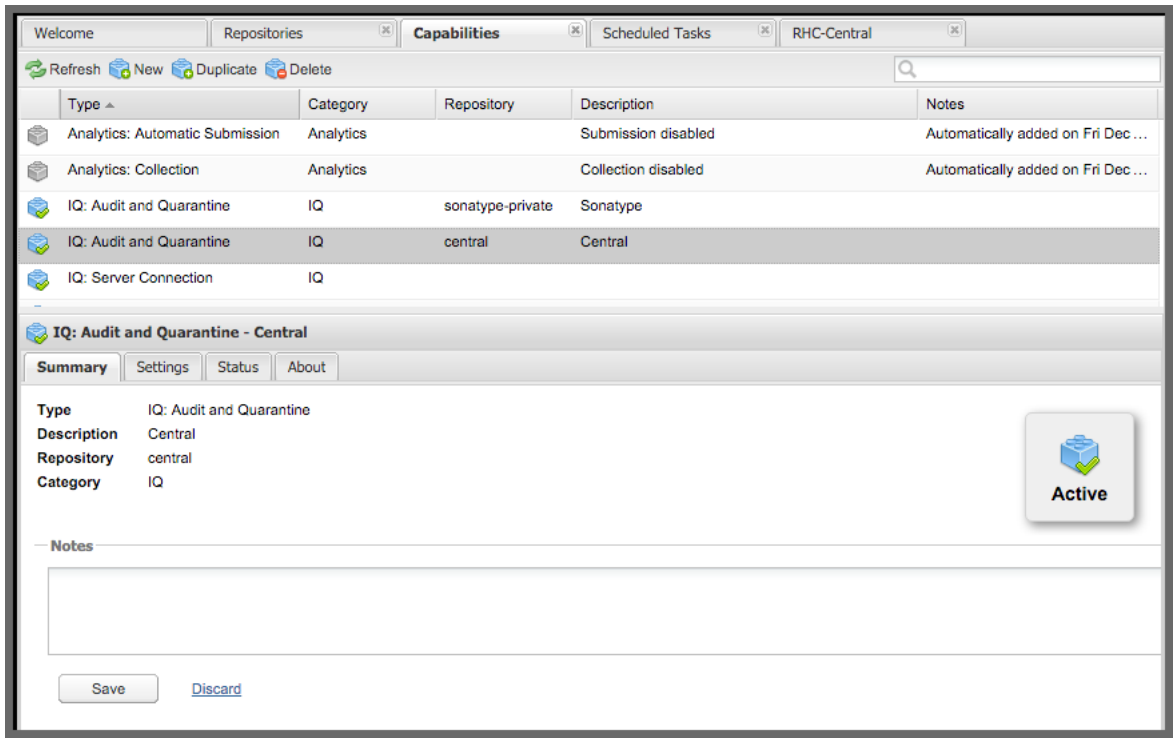


Figure 4.4: Nexus Repository Manager Capabilities Tab

Note
These features use IQ Server policy management to identify, and if desired, prevent a proxy repository from serving unwanted components. If you have chosen to Audit, policies must also be configured with a fail action. Additional information is available in the Audit and Quarantine section of the Nexus Repository Chapter.

Configuring Nexus Repository Manager 3.x

There are two steps in order to allow IQ Server to interact with an instance of Nexus Repository Manager, and evaluate repositories. First, you need to configure the IQ Server connection:

1. In Nexus Repository Manager, click the *Administration* button on the main toolbar.

2. In the Administration main menu, click *Server* under IQ Server.
3. Select *Whether to use IQ Server* to enable IQ Server.
4. Enter the IQ Server URL.
5. Select an *Authentication Method*:

   User Authentication: Enter the username and password.

   PKI Authentication: Delegate to the JVM for authentication.
6. Click *Verify connection* to test if a connection can be established.

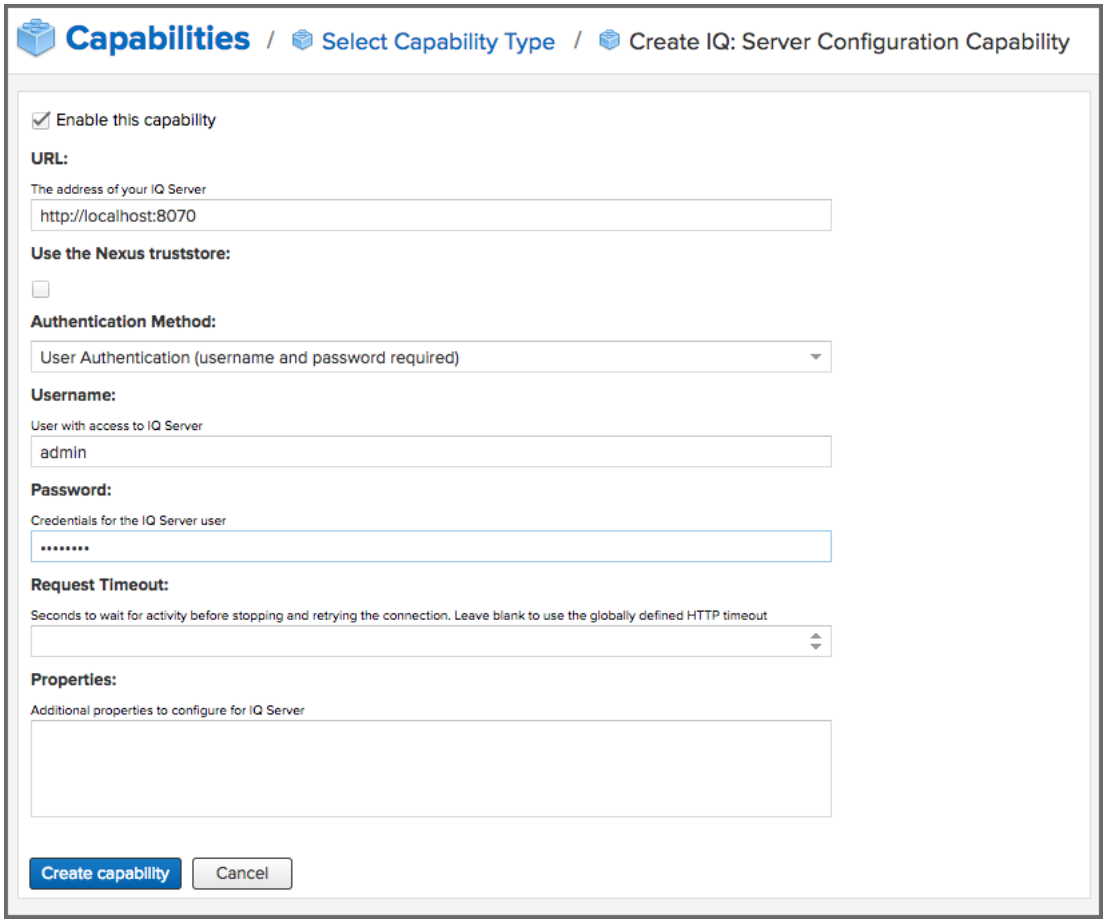


Figure 4.5: IQ Server Connection in Nexus Repository Manager 3.x

Note
For this quick start guide, using the default admin credentials is acceptable. However, for a real implementation, you'd want to review the chapter on Security Adminstration, making sure to review Section 7.5.

Next, add the Audit and/or Quarantine capability for each repository you want to evaluate. To configure Audit and/or Quarantine:

1. In Nexus Repository Manager 3.x, go to the *Administration* main menu and click *Capabilities* under *System*.
2. Click the *Create capability* button.
3. In the *Select Capability Type* view, click *IQ: Audit and Quarantine*.
4. Select a specific proxy repository to scan, for example *Central*.
5. Click *Create capability* to save the new capability for Audit and Quarantine.

An audit of the selected repository is automatically started. Nexus Repository Manager contacts IQ Server and evaluates the components within the selected repository against any associated policy.

Note
These features use IQ Server policy management to identify, and if desired, prevent a proxy repository from serving unwanted components. If you have chosen to Audit, policies must also be configured with a fail action. Additional information is available in the Audit and Quarantine section of the Nexus Repository Chapter.

Step 5: Reviewing Repository Results

Once configured, the evaluation of the repository is automatic and will occur given any repository changes (e.g. adding a new component). Depending on the size (number of components) of the repository you configured, the evaluation could take a minute or so, but in general is very quick.

As you review the results, if you are not continuing on to review Nexus Lifecycle functionality, you can skip ahead to the investigation and remediation section, which provides additional details for drilling

deeper into the results and available intelligence. Of course, a much more in-depth review of Nexus Firewall IQ Server can be found in the Nexus Firewall section of the IQ for Nexus chapter.

Note
Accessing repository results will differ depending on the version of Nexus Repository Manager you have installed (differences highlighted below).

Reviewing Results in Nexus Repository Manager 2.x

To review results in Nexus Repository Manager 2.x, click *Repositories* under the *Views/Repositories* menu. Repository Results are summarized in the *IQ Policy Violations* column of the *Repositories* tab.

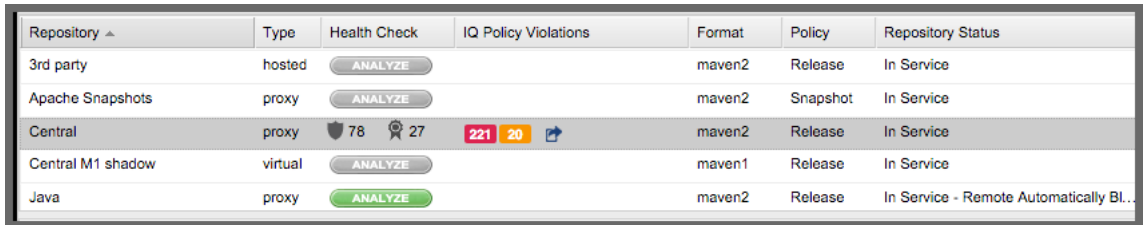


Figure 4.6: Nexus Repository Manager Repository Results

To view detailed results, click the *open* icon in the *IQ Policy Violations* column of the *Repositories* tab. IQ Server will open in a new tab showing detailed Repository Results.

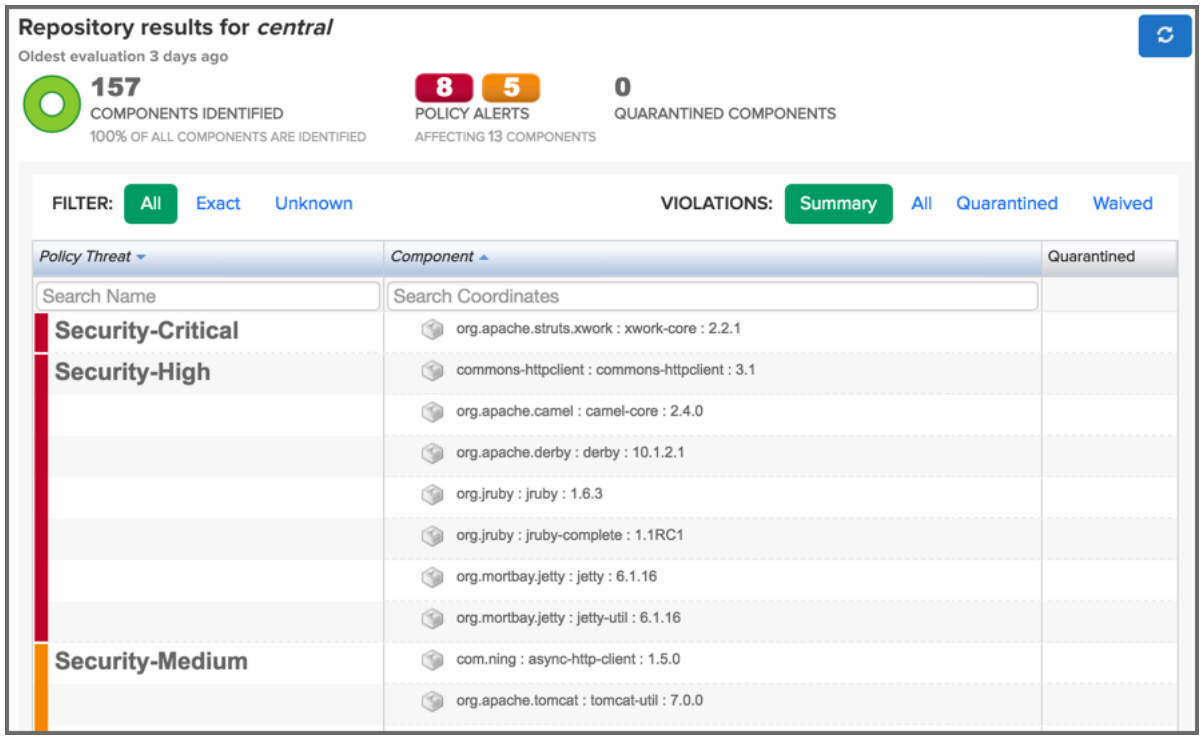


Figure 4.7: IQ Server Repository Results

Reviewing Results in Nexus Repository Manager 3.x

In Nexus Repository Manager 3.x, the results of an audit are summarized in the *IQ Policy Violations* column of the *Repositories* view as shown in the figure below. Access the *Repositories* view from the *Repository* sub menu of the *Administration* menu.

	Name	Type	Format	Status	URL	Health check	IQ Policy Violations
	maven-central	proxy	maven2	Online - Remote Available	copy	83 25	1 2
	maven-public	group	maven2	Online	copy		
	maven-releases	hosted	maven2	Online	copy		
	maven-snapshots	hosted	maven2	Online	copy		
	nuget-group	group	nuget	Online	copy		
	nuget-hosted	hosted	nuget	Online	copy		
	nuget.org-proxy	proxy	nuget	Online - Remote Connection ...	copy	0 0	

Figure 4.8: Nexus Repository Manager 3.x Repository Results

To view detailed results, click the *open* icon in the *IQ Policy Violations* column of the *Repositories* view. IQ Server will open in a new tab showing detailed Repository Results.

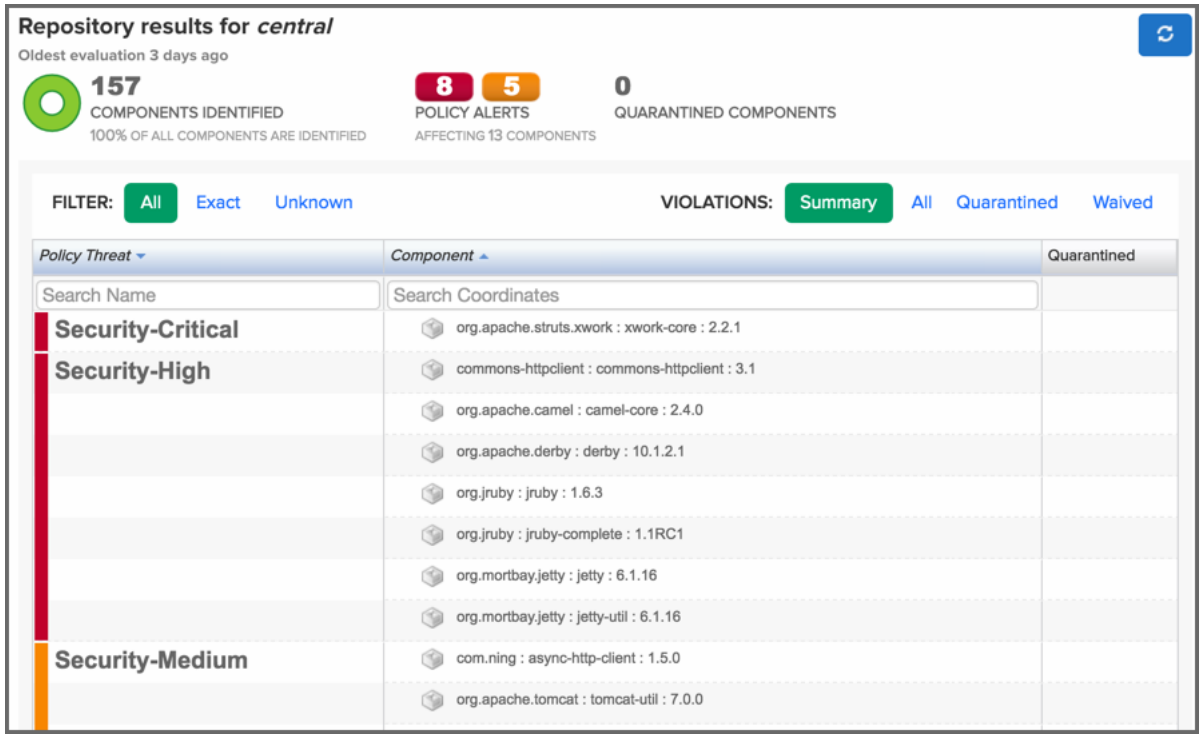


Figure 4.9: IQ Server Repository Results

Step 6: Investigating & Remediating Violations

Repository Results allow you to drill down to learn specific details about a violation, including the ability to isolate quarantined components. Click an individual component to open the *Component Information Panel* (CIP). The CIP displays many details, which are divided into different sections or tabs. To get you started using the CIP, take a look at these sections:

- *Component Info* - In the graph, you can move the vertical bar to learn the differences between versions of a component.
- *Policy* - You can click the Waive button to force IQ Server to ignore a policy violation.
- *Licenses* - You can track your research about a particular license and even override one.
- *Vulnerabilities* - You can click *Info* for a thorough explanation of a component's vulnerability and a recommended action.
- *Claim Component* - You can tell IQ Server to recognize a component even though it was previously identified as unknown.

This is just a small sample of the component information available in the CIP. For a complete discussion of the CIP, see Component Information Panel in the Nexus IQ Server Documentation.

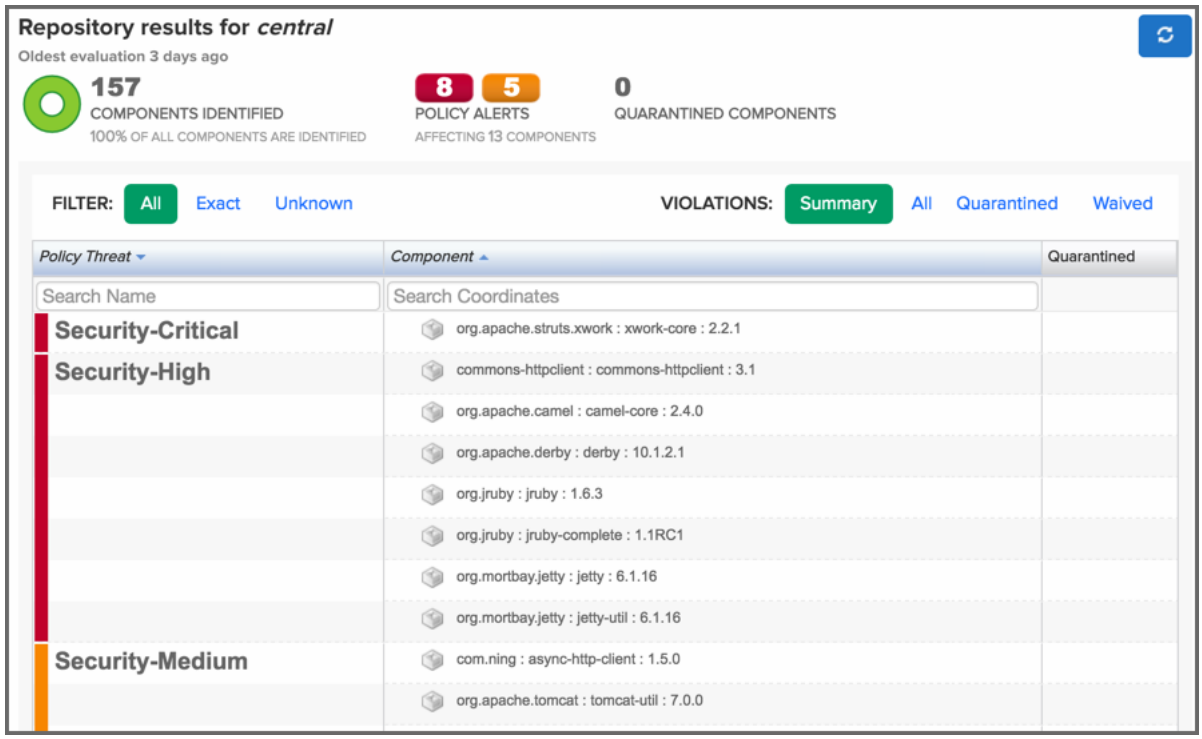


Figure 4.10: Quickstart Repository Results

Chapter 5

Quick Start Guide - Nexus Lifecycle

This guide can help you get IQ Server up and running for the purpose of trying out its features before installing it in your development environment. It should take approximately 15 minutes to complete using sample policies and applications.

Note
Nexus Lifecycle requires a license in order to experience the functionality described in this guide. If you are looking to try or purchase, Nexus Lifecycle, schedule a demo or contact us, and we'll be happy to assist.

Step 1: Installing and Starting IQ Server

Installing the IQ server is really a case of downloading the archived server, picking a location, and unpacking the contents. Since we won't be focused on mimicking a production experience, most laptop and desktop configurations should run IQ Server with no problem. If you are looking to plan for the future though, be sure to review the server requirements section of the Requirements chapter.

1. Create an installation directory in your desired location.
2. Download the latest version of IQ Server to the installation directory.

3. Extract the `tar.gz` or `.zip` file.

Once you've extracted the contents, follow the steps below to run IQ Server

1. Using a command line interface, switch to the nexus-iq-server bundle directory in your installation directory e.g. `nexus-iq-server-x.xx.x-xx-bundle`.
2. Run one of the following commands to start IQ Server:

   Linux or Mac: `./demo.sh`

   Windows: `demo.bat`
3. Open IQ Server in a browser using the default URL: http://localhost:8070
4. Log in using the default Administrator account:

```
Username: admin
Password: admin123
```

5. Install the required product license supplied to you by the Sonatype Support team.
   a. Click *Install License*.
   b. Navigate to the license file (`.lic`) and click *Open*.
   c. Click *I Accept* to accept the End User License Agreement.

Note
IQ Server needs access to an external data service to perform evaluations, which may be blocked in your internal environment. For a workaround, see Running IQ Server Behind a HTTP Proxy Server in the IQ Server documentation.

Step 2: Importing Sample Policies

Policy is at the core of IQ Server's automation capabilities. While you can create a completely custom set of policies, importing the Sonatype Sample Policy set set is the quickest way to get started. This set includes multiple policies for triggering violations on security vulnerabilities, licensing issues, architecture issues, and more.

1. In a separate browser tab or window, download the Sonatype Sample Policy set (`.json` file) from the IQ Server documentation.
2. In IQ Server, click the *Organization & Policies* icon on the IQ Server toolbar.
3. The *Root Organization* should be selected in the sidebar. Click the *Actions* menu and select *Import Policies*.
4. In the *Import Policies* dialog, click *Choose File*, select the `.json` file you downloaded, and click *Open*.
5. Click the *Import* button.

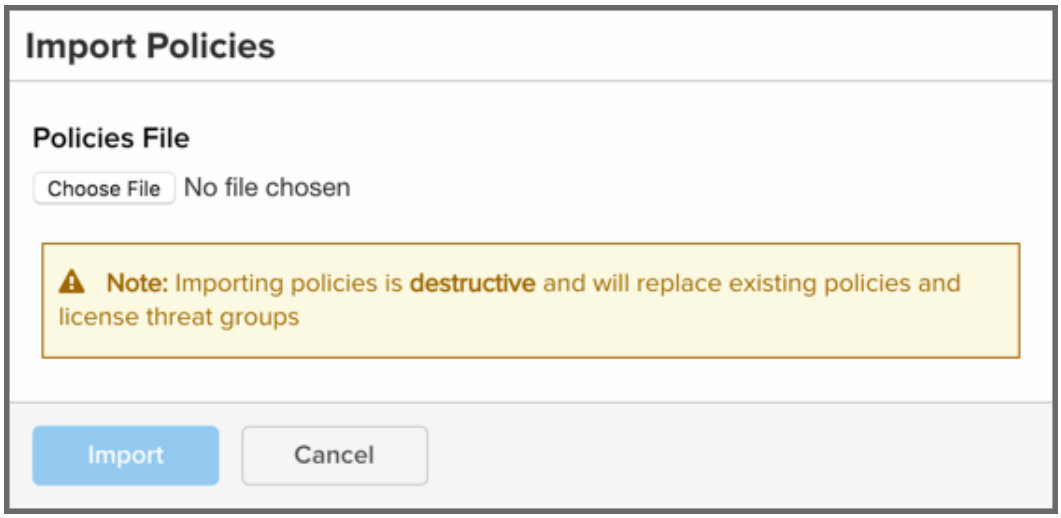


Figure 5.1: Import Policy Dialog

Step 3: Configuring Organizations and Applications

When evaluating applications, understanding IQ Server's system hierarchy is critical: Root Organization, organization, and application. This means policies and other configuration items are inherited from the Root Organization on down. This allows for easier policy management especially when you have multiple organizations and applications. Thus, in order to evaluate an application, you must have at least one organization and a corresponding application.

Creating an organization

1. In the *Organization & Policies* area, with the *Root Organization* selected in the sidebar, click the *New Organization* button.
2. In the *New Organization* dialog, enter a name into the *Organization Name* text box.
3. Click the *Create* button.

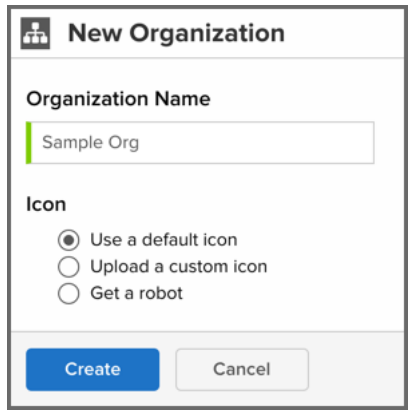


Figure 5.2: New Organization Dialog

Creating an application

1. With your newly created Organization selected in the sidebar, click the *New Application* button.
2. In the *New Application* dialog, enter an *Application Name* and *Application ID*.
3. Click the *Create* button.

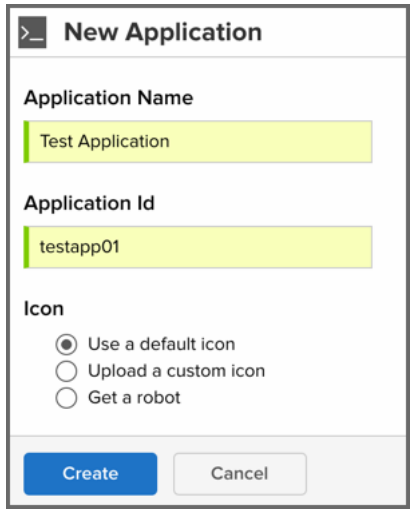


Figure 5.3: New Application Dialog

Step 4: Evaluating Applications

After you install, start, and configure IQ Server, you are ready to evaluate applications. If you need a sample application, you can download WebGoat (`webgoat-container-x.x.x-war-exec.jar`) at https://github.com/WebGoat/WebGoat/releases.

To evaluate an application:

1. In the *Organization & Policies* area, select your application in the sidebar. The file that you evaluate will be associated with this application.
2. Go to the *Actions* menu, and click *Evaluate Binary*.
3. In the *Evaluate a Binary* dialog:
    a. Click the *Choose File* or *Browse* button, select the file to evaluate, and click *Open*.
    b. Click to select any stage to associate with the evaluation (e.g. Build).
    c. Click *No* to prevent sending notifications of policy violations as defined in the policy's configuration settings.
    d. Click the *Upload* button to begin evaluating the selected application. An *Evaluation Status* message is displayed.

e. When the evaluation is complete, click the *View Report* button to open the Application Composition Report for the application.

Step 5: Reviewing Results

Once evaluated, the results of a binary evaluation are displayed in the Application Composition Report, which you can always access by clicking the Reporting icon on the IQ Server toolbar.

The report's information is divided into four tabs:

- *Summary* - An overview of identified components and their policy alerts, security issues, and license analysis.
- *Policy Violations* - A list of violated policies and the components that triggered them sorted by threat level from from highest to lowest.
- *Security Issues* - A list of security vulnerabilities and the components that triggered them sorted by threat level from highest to lowest.
- *License Analysis* - A list of license issues and the components that triggered them sorted by license threat from highest to lowest.

For a more thorough explanation of the report, see the Application Composition Report chapter in the Nexus IQ Server Documentation.

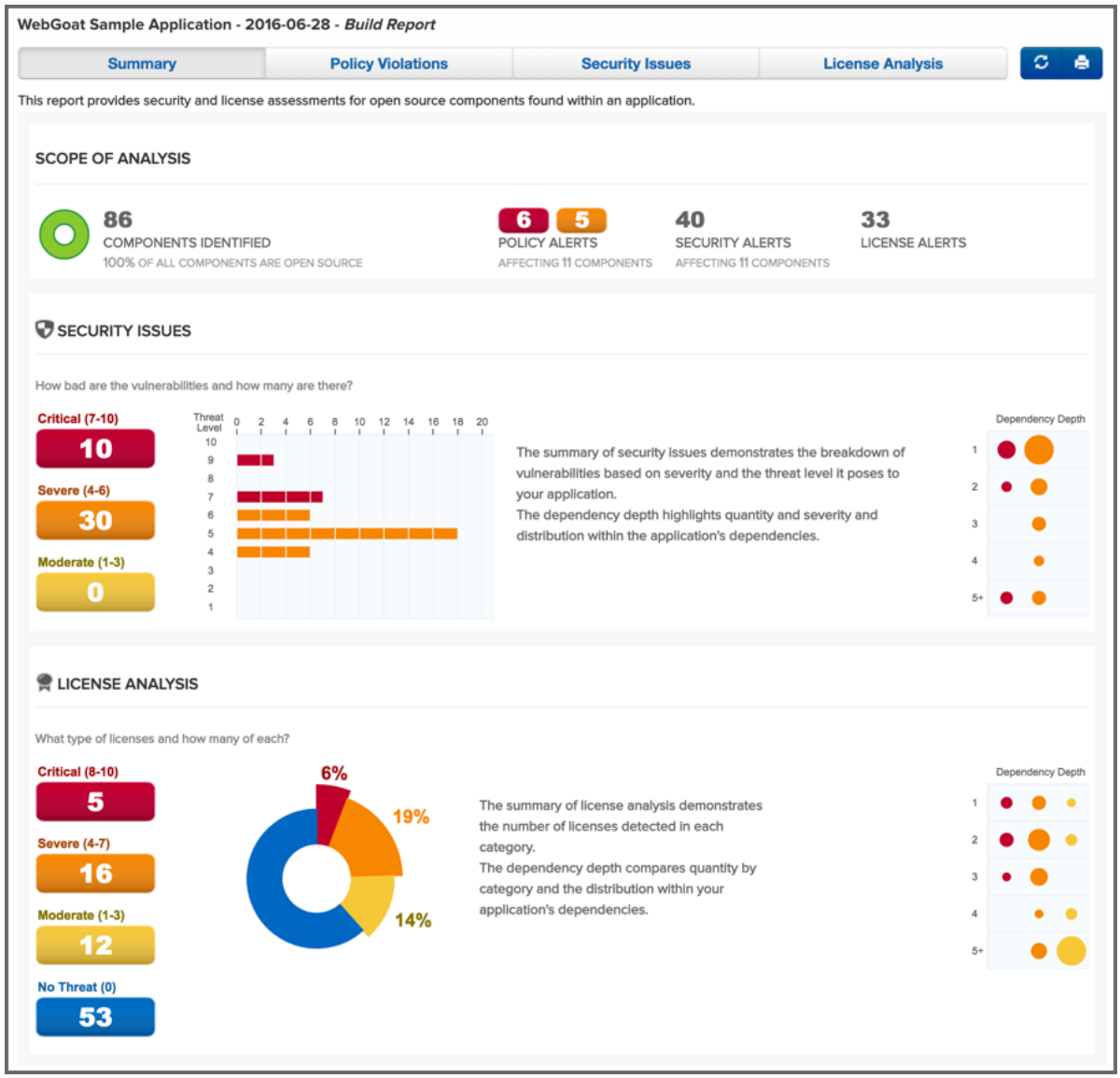


Figure 5.4: Application Composition Report

Step 6: Investigating & Remediating Violations

In the Application Composition Report, you can drill down to learn specific details about a violation. In every tab (except the Summary tab), you can click an individual component to open the *Component Information Panel* (CIP). The CIP displays many details, which are divided into different sections or tabs. To get you started using the CIP, take a look at these sections:

- *Component Info* - In the graph, you can move the vertical bar to learn the differences between versions of a component.
- *Policy* - You can click the Waive button to force IQ Server to ignore a policy violation.
- *Licenses* - You can track your research about a particular license and even override one.
- *Vulnerabilities* - You can click *Info* for a thorough explanation of a component's vulnerability and a recommended action.
- *Claim Component* - You can tell IQ Server to recognize a component even though it was previously identified as unknown.

This is just a small sample of the component information available in the CIP. For a complete discussion of the CIP, see Component Information Panel in the Nexus IQ Server Documentation.

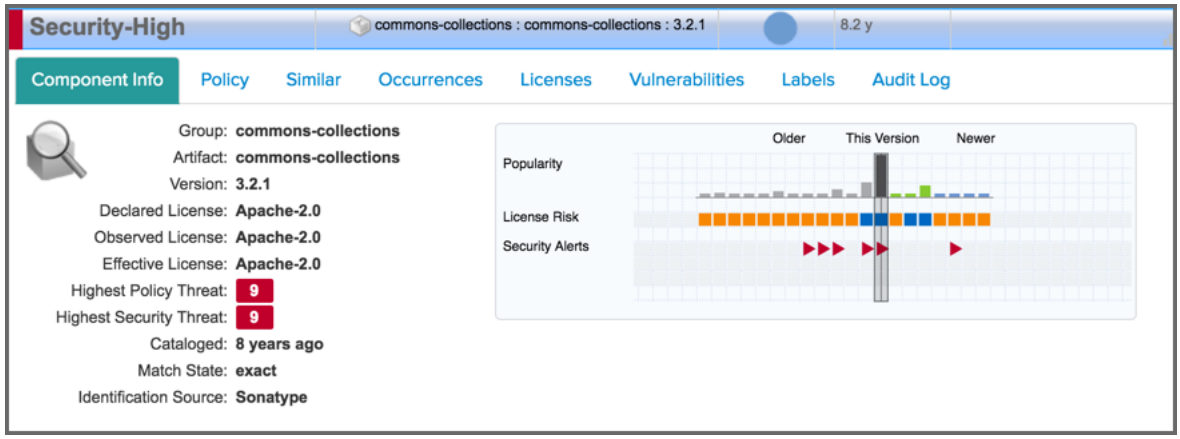


Figure 5.5: Component Information Panel

Chapter 6

IQ Server Setup

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

The IQ Server allows you to create and store policies, manage access, set configuration items, and review the effect of your policies through a variety of reports.

In addition, it will be the centralized location for ensuring seamless integration to systems such as Eclipse, Hudson / Jenkins, and/or Nexus.

The following chapter will provide everything you need to install and deploy the IQ Server, as well as make sure you can setup and define access for your team.

6.1 Installation

Before installation, please check the IQ Server requirements.

After a successful download of the IQ Server bundle archive, you should create an installation directory

in the desired location and move the archive into the directory.

```
cd /opt
mkdir nexus-iq-server
mv ~/Downloads/nexus-iq-server.* nexus-iq-server/
cd nexus-iq-server
```

Moving into the directory and extracting the archive with either one of the commands:

```
unzip nexus-iq-server*.zip
tar xfvz nexus-iq-server*.tar.gz
```

should result in a directory with the following files:

```
README.txt
config.yml
demo.bat
demo.sh
eula.html
nexus-iq-server-1.31.0-01-bundle.tar.gz
nexus-iq-server-1.31.0-01.jar
```

6.1.1 Starting the IQ Server

Once the IQ Server is installed, it can be started with:

```
cd /opt/nexus-iq-server
java -jar nexus-iq-server-*.jar server config.yml
```

This command will start the server with the IQ Server application using the configuration from the config.yml file and logging any output straight to the console. After a complete start your console should display a message similar to:

```
... [main] org.eclipse.jetty.server.AbstractConnector - Started  ↩
    InstrumentedBlockingChannelConnector@0.0.0.0:8070
... [main] org.eclipse.jetty.server.AbstractConnector - Started  ↩
    SocketConnector@0.0.0.0:8071
```

The command to start the server can be modified by adding java configurations parameters such as `-Xmx1024m -XX:MaxPermSize=128m` to improve performance and adapt to the server hardware.

At this stage you can access the web application at port 8070 of your server via any web browser. Initial startup will display a screen for the Section 6.1.2.

Warning
We recommend always using SSL to ensure confidentiality of report and credential data during transit. This can be done by setting up a proxy server. You can find more details in the HTTPS Configuration section.

6.1.2 License Installation

IQ Server requires a license to be installed. The required license file will be supplied to you by the Sonatype support team in the form of a `.lic` file.

Open a web browser and navigate to the IQ Server web application at port 8070 to install the license. Opening the URL, e.g. for a localhost deployment at http://localhost:8070, displays the Product License Configuration of the IQ Server shown in Figure 6.1.

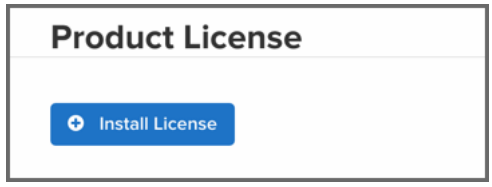


Figure 6.1: Installing a Product License on IQ Server

Press the *Install License* button and select the `.lic` file in the file selector. As a next step you are required to accept the end user license agreement shown in Figure 6.2 by pressing the *I Accept* button.

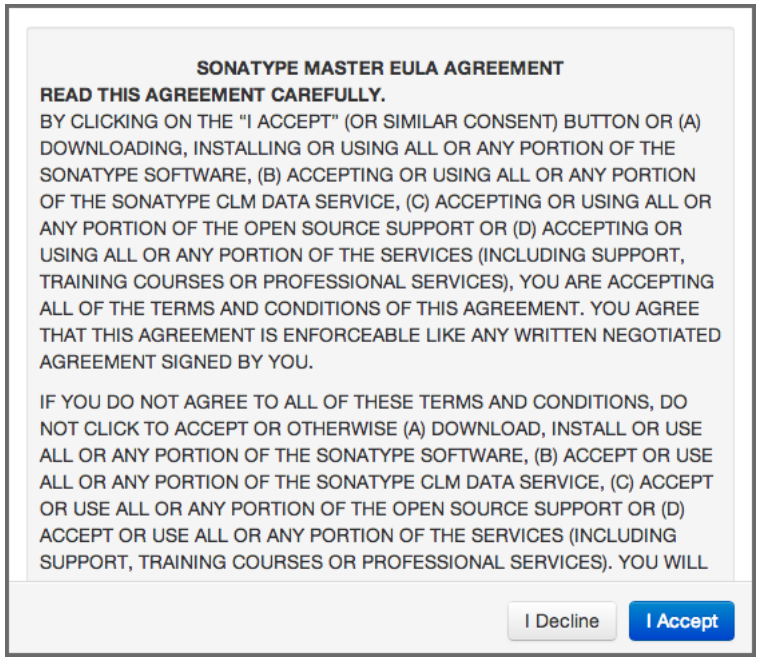


Figure 6.2: IQ Server End User License Agreement Window

After a success message you will be redirected to the Product License page, which will now display the expiry date of the license as visible in Figure 6.3.

Figure 6.3: Installed Product License on IQ Server

6.1.3 IQ Server Directories

When the IQ Server first starts, it creates a directory for the storage of all its data and configuration. This directory is configured in `config.yml` and defaults to `./sonatype-work/clm-server`. This path is relative to the location from which the invoking java command is used.

Using the default startup command from the installation directory, causes `/sonatype-work/clm-server` to be created within it.

If you would like to separate the installation and data directories you can set the `sonatypeWork` to a different location.

Additionally a `log` directory is created within the installation directory and the `currentLogFilename` parameter in `config.yml` can be used to change the location. Further information on logging configuration can be found in Section 6.2.8.

6.1.4 Running the IQ Server as a Service

For production usage, we strongly recommend to set up the IQ Server as a service or daemon. This will ensure that any operating system reboots will include starting up the IQ Server.

A dedicated user for running a service is a well known best practice. This user should have reduced access rights as compared to the root user. Configuration of this user will depend on the operating system and security system used.

Once the user is configured, you need to ensure that full access rights to the IQ Server installation directory are granted. An example command to achieve this for a service user with the username *iqserver* is:

```
chown -Rv iqserver /opt/nexus-iq-server
```

If you have configured the `sonatypeWork` parameter in `config.yml` to point to a different directory, you have to adjust the access rights for it as well.

The principal command for starting the IQ Server can be used in a simple startup script as displayed in Startup Script. The `javaopts` variable should be adjusted to suit the hardware used.

Startup Script

```
#! /bin/sh
cd /opt/nexus-iq-server
javaopts="-Xmx1024m -XX:MaxPermSize=128m"
java  $javaopts -jar nexus-iq-server-*.jar server config.yml
```

A running server can be stopped with a simple shutdown script in Shutdown script.

Shutdown script

```
#!/bin/sh
     pid=`ps aux | grep nexus-iq-server | grep -v grep | awk '{print $2}'`
     kill $pid
```

Typically these approaches are combined to a service script similar to the script listed in Simplistic Service Script for Unix Systems. Saving this script as e.g. `nexus-iq-server` allows you to start the server with its log running to the current shell with

```
./nexus-iq-server console
```

Starting as a background process can be initiated with:

```
./nexus-iq-server start
```

and a running background server can be stopped with:

```
./nexus-iq-server stop
```

This example script can be improved to be more robust against repeat invocations, long running stops and potentially work better across different Unix flavors, but shows the principal functionality. A similar script can be used for Windows.

Simplistic Service Script for Unix Systems

```
#!/bin/sh

# The following comment lines are used by the init setup script like the
# chkconfig command for RedHat based distributions. Change as
# appropriate for your installation.

### BEGIN INIT INFO
# Provides:          nexus-iq-server
```

```
# Required-Start:    $local_fs $remote_fs $network $time $named
# Required-Stop:     $local_fs $remote_fs $network $time $named
# Default-Start:     3 5
# Default-Stop:      0 1 2 6
# Short-Description: nexus-iq-server service
# Description:       Start the nexus-iq-server service
### END INIT INFO

NEXUS_IQ_SERVER_HOME=/opt/tools/nexus-iq-server
VERSION=1.12.0
JAVA_OPTIONS="-Xmx1024m -XX:MaxPermSize=128m"
# The user ID which should be used to run the IQ Server
# # IMPORTANT - Make sure that the user has the required privileges to  ↩
    write into the IQ Server work directory.
RUN_AS_USER=iqserver

do_start()
{
    cd $NEXUS_IQ_SERVER_HOME
    su -m $RUN_AS_USER -c "java -jar $JAVA_OPTIONS nexus-iq-server- ↩
        $VERSION.jar server  config.yml > /dev/null 2>&1 &"
    echo "Started nexus-iq-server"
}

do_console()
{
    cd $NEXUS_IQ_SERVER_HOME
    java -jar $JAVA_OPTIONS nexus-iq-server-$VERSION.jar server config.yml
}

do_stop()
{
    pid=`ps aux | grep nexus-iq-server | grep -v grep | awk '{print $2}'`
    kill $pid
    echo "Killed nexus-iq-server - PID $pid"
}

do_usage()
{
    echo "Usage: nexus-iq-server [console|start|stop]"
}

case $1 in
console) do_console
;;
start) do_start
;;
stop) do_stop
;;
```

```
*) do_usage
;;
esac
```

Setting up this script as a startup script will vary between operating systems and distributions depending on the init system used. Generally the script would be copied to a dedicated startup directory and assigned with run-levels and other characteristics for the start up. As an example on a Debian based systems the following commands could be used:

```
sudo su
cp nexus-iq-server /etc/init.d/
cd /etc/init.d
update-rc.d nexus-iq-server  defaults
service nexus-iq-server  start
```

Depending on the requirements from your system administrator the scripts will have to be modified to fit into your environment and exact deployment scenario.

Tip

Our support team can assist you with operating system and Linux distribution-specific tips and tricks regarding the startup script and installation.

6.2 Advanced Configuration

The main configuration file for the IQ Server installation is a YAML formatted file called *config.yml* found in the installation directory. The IQ Server is an application running on a Dropwizard server.

In addition a number of configuration steps can be taken within the running server user interface.

This section will discuss various configuration options in the config file as well as some other configuration scenarios. When editing the file it is important to preserve the indentations, since they are significant for the resulting values created when parsing the configuration file.

Tip
The `config.yml` format does not support tab characters. Use an editor that displays special characters like tabs when editing the file.

6.2.1 Initial Configuration of the IQ Server

Besides the license installation mentioned earlier, there are a few further configuration steps you should consider before diving right into using the IQ Server. You can configure various aspects in the *System Preferences* section of the IQ Server user interface, which you can access by clicking on the *System Preferences* icon located in the top right of the IQ Server header (resembles a cog/gear) and choose the desired option to configure:

- Configure *Users* and *Roles* in the *System Preferences* menu, potentially combined with *LDAP* as well. Read more about the security setup outlined in the Security Administration documentation.
- Configure *Proprietary Packages* so that the IQ Server can distinguish your own code from other unknown components. For more information, refer to the component match and identification documentation in the report chapter.
- Inspect or update or configure your *Product License*

6.2.2 Running the IQ Server Behind a HTTP Proxy Server

Many organizations filter, control and optimize access to the internet via a proxy server. Any server or even any computer within the organization is forced to connect to the internet via the proxy server. The IQ Server needs to communicate with the Sonatype Data Services via the internet.

To allow the IQ Server to connect via a proxy, you have to specify the connection details in the `proxy` section of the `config.yml file` displayed in Proxy Configuration in `config.yml`.

Proxy Configuration in config.yml

```
proxy:
    hostname: "127.0.0.1”
    port: 80
    username: "anonymous”
    password: "guest"
```

If your proxy server is based on whitelisted URLs, you can use the following list of URLs to ensure that the IQ Server can reach all the required services.

- https://clm.sonatype.com
- http://cdn.sonatype.com/

6.2.3 Setting the Base URL

If your IQ Server is accessed via a https proxy or a proxy server that changes the http port or for other reasons can potentially not determine what the authoritative URL to access the server itself is, you need to configure the `baseUrl` parameter.

```
baseUrl: http://nexus-iq-server.example.com/
```

It is used by the server for any user facing links e.g. located in email notifications sent by the server to direct users to the server.

6.2.4 Reverse Proxy Authentication

Browser-based single sign-on (SSO) configurations allow a user to log into the system in a web browser without the need to log into any individual web applications. Any user navigation to further applications carries the authenticated username through to the application and the user is automatically logged in.

Typically this is implemented with a reverse proxy server and the username is supplied via a HTTP header field.

The IQ Server can be configured to accept this kind of SSO configuration in the `config.yml` file, allowing you to specify the exact header field to be used:

```
# Configures reverse proxy authentication for the web UI.
reverseProxyAuthentication:
    # Set to true to activate authentication
    enabled: true
    # Name of the HTTP request header field that carries the username
    usernameHeader: "REMOTE_USER"
    # Set to true for backward compatibility with old client plugins
    csrfProtectionDisabled: false
```

Note
When using reverse proxy authentication from integration points to IQ Server, Cross-Site Request Forgery (CSRF) protection is enabled by default. If an integration does not support CSRF protection, it should be updated to the latest version. Alternatively, CSRF protection can be disabled by setting `csrfProtectionDisabled:true` in the IQ Server configuration.

The default `config.yml` contains a commented out section for this configuration with some further details.

This authentication method applies to all users, both IQ Server and LDAP users. Incoming usernames are matched first to IQ Server users, then to LDAP users, and then the configuration in the IQ Server determines the access level granted to the user.

Public Key Infrastructure (PKI) Authentication

Note
In order to implement PKI authentication, a reverse proxy server is needed to translate PKI supplied credentials to users known by IQ Server. See the Reverse Proxy Authentication section for details.

Tools and plugins can be configured to use PKI authentication, which delegates authentication to the Java Virtual Machine (JVM). When delegated, the tool or plugin does not handle authentication and instead the JVM supplies PKI information to the reverse proxy for authentication.

For information on setting PKI authentication for a specific tool or plugin, please see the following:

- **CLM for Maven**: Evaluating Project Components with Sonatype CLM Server.
- **Nexus IQ CLI**: Evaluating an Application.
- **Nexus Repository Manager 2**: Connecting Nexus Repository Manager 2.x to IQ Server.
- **Nexus Repository Manager 3**: Connecting Nexus Repository Manager 3.x to IQ Server.
- **Nexus IQ for Bamboo**: Configure Nexus IQ for Bamboo.
- **Nexus IQ for Hudson/Jenkins 1.x**: Nexus IQ for Hudson/Jenkins 1.x Global Configuration.

- **IQ for IDEA**: Configuring IQ for IDEA.
- **IQ for Eclipse**: Configuring Sonatype CLM for Eclipse.
- **CLM for SonarQube**: CLM for SonarQube Configuration.

6.2.5 Appending a User Agent String

To address the firewall configurations set by some organizations, you can customize the user agent header used for HTTP requests. To add a user agent string, add the following line to the IQ Server config.yml:

```
userAgentSuffix: "test string"
```

Note
Control characters are not permitted, and the max length of the string is 128 characters.

6.2.6 File Configuration

IQ Server stores various files and data related to its operations in a work directory. By default this data is stored in a `/sonatype-work/clm-server` directory in the path the server runs. The directory is configurable using the `sonatypeWork` field in File Configuration in `config.yml`.

File Configuration in config.yml

```
sonatypeWork: ./sonatype-work/clm-server
```

In addition, IQ Server uses the system temporary directory during its operation. This folder varies by operating system but is usually controlled by an environmental variable. If a specific directory needs to be used, the IQ Server can be started with a command line flag as such:

```
cd /opt/nexus-iq-server
java -jar -Djava.io.tmpdir=/path/to/tmpdir nexus-iq-server-*.jar server  ↩
    config.yml
```

Note that the user account which the server runs under must have sufficient access rights to both the work and temporary directory in order for IQ Server to function properly.

6.2.7 Email Configuration

The IQ Server can be configured to send email notifications for events such as policy violation notifications. This functionality requires an SMTP server, which is configured along with a number of other options in the `mail` section of the `config.yml file` displayed in Mail Configuration in `config.yml`.

Mail Configuration in config.yml

Here's an example configuration:

```
mail:
    hostname: your.mailserver.com
    port: 465
    username: user@company.com
    password: password
    tls: true
    ssl: true
    systemEmail: "Sonatype@localhost"
```

The connection details are established with `hostname` and `port` and optionally with the addition of `username`, `password`, `tls` and `ssl`. The `systemEmail` parameter will be used as the sender email for any emails the IQ Server sends. **All fields are required.**

Finally, when setting email configuration, make sure you have also set the Base URL, otherwise sending of notification emails may fail.

6.2.8 Logging Configuration

The IQ Server application logging can be configured in the `logging` section of the `config.yml` file. By default a log directory is created in the installation directory and the `clm-server.log` is rotated. Further logging configuration is documented in the Dropwizard manual.

6.2.9 HTTP Configuration

The HTTP configuration in `config.yml` is displayed in HTTP Configuration in `config.yml`. The `port` parameter for the IQ Server allows you to set the port at which the application is available. The

`adminPort` exposes the operational menu. Both ports can be freely changed to other values, as long as these port numbers are not used and in the allowed range of values greater than 1024.

HTTP Configuration in config.yml

```
http:
    port: 8070
    adminPort: 8071
```

6.2.10 HTTPS/SSL

One option to expose the IQ Server via https, is to use an external server like Apache httpd or nginx and configure it for reverse proxying the external connections via https to internal http connection. This reverse proxy can be installed on the same server as the IQ Server or a different server and numerous tutorials for this setup are available on the internet.

A second option is to directly configure SSL support for Dropwizard by modifying the `http:` segment in the `config.yml` file following the example in HTTPS Configuration in `config.yml`.

HTTPS Configuration in config.yml

```
http:
  port: 8443
  adminPort: 8471

  connectorType: nonblocking+ssl

  ssl:
    keyStore: /path/to/your/keystore/file
    keyStorePassword: yourpassword
```

The keystore file can be generated and managed with the `keytool`. Further documentation is available in the Dropwizard documentation and the documentation for keytool.

6.2.11 Anonymous Access

By default the IQ Server requires users to authenticate when submitting applications for evaluation. While not recommended, if you need to allow anonymous application evaluation submissions, add the following

line to the `config.yml`:

```
anonymousClientAccessAllowed: true
```

6.2.12 CSRF Protection

Attacks on the IQ Server could occur via a cross-site request forgery (CSRF). To protect against this, a configuration item *csrfProtection* has been provided. This option is set to *true* by default.

```
# Enables/disables cross-site request forgery protection. Defaults to true ↩
    for increased security.
#csrfProtection: true
```

Note
In cases where the HTTP headers are stripped (e.g. a proxy configuration), this protection would block usage of the UI. To address this, you can disable this protection by setting the configuration item to *false*.

6.3 Backing Up the IQ Server

We highly recommend that you establish a data recovery plan in accordance with your company's policies.

The IQ Server keeps all its configuration and data, besides the startup configuration, in the `sonatypeWork` folder as configured in `config.yml`. By default, this folder will be the `/sonatype-work/clm-server` folder in your installation directory.

There is a tight coupling between report data stored in the file system and the data stored in the H2 database. Any backup strategy for `sonatypeWork` while the IQ Server is still available to users runs the risk that the database will still be open, leading to a corrupt backup. Therefore, the IQ Server should be shut down before performing the backup.

6.4 Upgrading the IQ Server

The latest version of IQ Server can be downloaded from the Sonatype support site.

Before starting any upgrade, make sure you have reviewed all upgrade instructions provided for your current version (see additional sections below), as well as any versions that followed.

Additionally, you should check the Compatibility Matrix located in our Download and Install Knowledge Base article.

1. Stop the IQ Server
2. Perform a backup
3. Make a copy of nexus-iq-server/config.yml
4. Copy the new installation bundle into installation folder
5. Extract the bundle
6. Copy your backed-up config.yml over the default
7. Update any startup scripts as needed
8. Start the IQ Server

Note
Optionally you can manually merge the changes from the existing config.yml into the new, but this is not required.

If there is any concern, please feel free to contact our support team: support@sonatype.com.

6.4.1 Upgrading from Version 1.17 or Earlier to Version 1.18 or Later

IQ Server version 1.18 introduces the Root Organization—a new entity at the top of the system hierarchy that allows you to set policy globally across all organizations and applications. After you update the IQ Server to version 1.18, you should configure and create the Root Organization. It's a one time process,

and occurs when the server is restarted. The process makes a permanent change to the system hierarchy that cannot be undone. It is strongly recommended that you backup the IQ Server and read "Introducing the Root Organization" before proceeding.

In IQ Server version 1.21, the *Sonatype CLM for Hudson and Jenkins* plugin has been updated and rebranded to *Nexus IQ for Hudson/Jenkins 1.x*. If you have a prior version of the plugin installed, then you must uninstall the older version before installing the newer rebranded one. For installation instructions, see the Nexus IQ for Hudson/Jenkins 1.x chapter.

IQ Server version 1.26 introduces CSRF protection for all available plugins that use reverse proxy authentication. This new protection is enabled by default. If you want to upgrade to IQ Server version 1.26 and use reverse proxy authentication in your plugins, you should upgrade your plugins to their latest versions first.

Note
If you would like to upgrade to IQ Server version 1.26 and use reverse proxy authentication with older plugin versions, you will need to disable CSRF protection for reverse proxy authentication. See the section on Reverse Proxy Authentication for more information.

Note
Both Nexus Repository Manager version 2.14.3 and older and Nexus Repository Manager version 3.2.1 and earlier 3.x versions do not support CSRF protection when using reverse proxy authentication. If you want to use reverse proxy authentication with these Nexus Repository Manager versions and IQ Server 1.26 or later, you will need to disable CSRF protection for reverse proxy authentication. See the section on Reverse Proxy Authentication for more information.

6.4.2 Upgrading from Version 1.15 or Earlier to Version 1.23 or Later

Due to data migrations, you will need to upgrade to version 1.16 first, before proceeding to upgrade to version 1.23 or later versions of IQ Server.

6.4.3 Upgrading from Version 1.16 or Earlier

In version 1.17 a rebranding of the Sonatype CLM product took place, and is now known as Nexus IQ. As part of this rebranding two of the binaries also changed during this release:

Server
From: *sonatype-clm-server*
To: *nexus-iq-server*

CLI
From: *sonatype-clm-scanner*
To: *nexus-iq-cli*

If you have any scripts utilizing the previous names, you will want to update these given the change above.

Note
In the example above, only the server name is given. The full binary name would look like nexus-iq-server-1.31.0-01.jar

---

6.4.4 Upgrading from Versions Earlier than 1.9.x

While Sonatype only supports the previous two releases, we are happy to help direct any upgrade needs you may have. If you are upgrading from a version prior to 1.9.x, please contact our support team directly: support@sonatype.com.

Chapter 7

Security Administration

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

This chapter will cover the creation and management of user accounts, passwords, and associated permissions. For the most part this will cover interaction with the IQ Server.

Important
It is important to note, that this chapter is aimed at system administrators, and isn't a guide for other users. Most areas discussed here will require a user account that has been assigned to the System and/or Policy Administrator role.

7.1 Logging In

To log into the IQ Server simply go to the address of your IQ Server (e.g. localhost:8070) and enter your username and password.

If this is your very first time logging in, you will need to use the default Admin account. This user is a preconfigured Admin account that has been assigned to all Administrator roles. Once you log in with this account for the first time, be sure to change the admin password.

To logout, click on the *Log Out* link located in the upper right corner.

Note
The server will timeout after 30 minutes of inactivity.

7.2 Product Notifications

Once logged in, you can check for product notifications, which provide the most up-to-date information about IQ Server. Click the notifications icon on the IQ Server toolbar to view the *Notifications* panel. If you have unread notifications, they are indicated by a count, in blue, displayed over the notification icon.

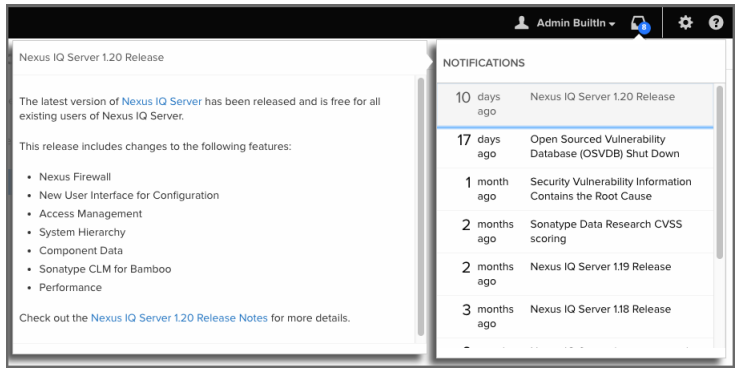


Figure 7.1: Notifications Panel

7.3 User Management

While we recommend using a security protocol such as LDAP for managing users and permissions, the IQ Server realm is still available for those who would like a lighter setup, where all users, groups and rights are stored directly in the IQ Server.

At a minimum, IQ Server requires you to log in before anything else can happen. This can be done by creating a user within the IQ Server realm, or by configuring LDAP, and logging in via one of those connected users.

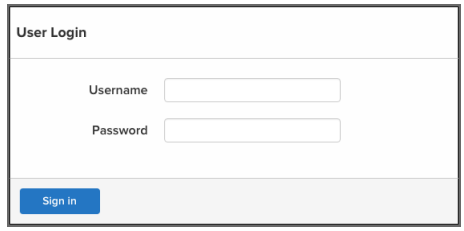


Figure 7.2: Login

7.3.1 Changing the Admin Account Password

The IQ Server ships with a default Admin account with a username `admin` and a password `admin123`. If you do nothing else related to security, be sure to change this password. We'll cover this in the Section 7.3.3. For now, we'll simply detail the process to change the password for the Admin account.

1. After logging in with the Admin account, click on the button with your username to the left of the *System Preferences* gear-shaped, icon. For the default administrator the username will show `Admin Builtin`.
2. A list of options will be displayed, click *Change Password*.
3. Enter the current password (`admin123` for the default admin user), the new password, and then confirm the new password.
4. Click the *Change* button to save the new password.

Note
Any user, including an admin, can change their password following the instructions above. However, only an admin can reset a user's password (discussed later in this chapter) without knowledge of the current password.

7.3.2 Creating a User

To create a new user in the IQ Server realm, follow the instructions below.

1. Log into the IQ Server with a user that has been assigned to the System Administrator role.
2. Click the *System Preferences* icon located in the top right of the header.
3. Choose *Users* from the drop down menu. The **Users** administration area will now be displayed.
4. Click the *New User* button located at the top of the list of users.
5. The **Add New User** form will now be displayed. Enter the following information:
    a. First Name
    b. Last Name
    c. Email
    d. Username
    e. Password
    f. Validate Password
6. Click the *Save* button, to save the new user.

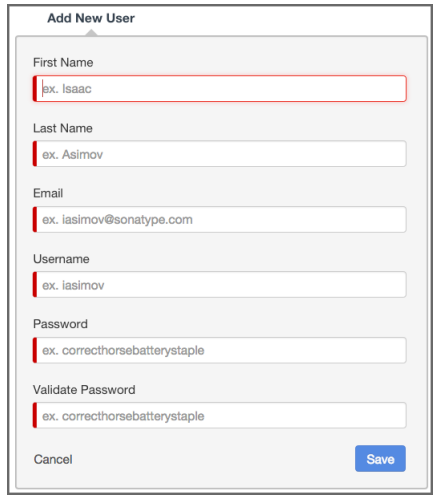


Figure 7.3: Create User

7.3.3 Editing and Deleting User Information

Editing user information is only available to an admin. The information that can be edited includes the first name, last name, email address, and password. To edit an existing user, follow these steps:

1. Log into the IQ Server with a user that has been assigned to the System Administrator role.
2. Click the *System Preferences* icon located in the top right of the IQ Server header.
3. Choose *Users* from the drop down menu. The **Users** administration area will now be displayed.
4. At least one user - the initial `admin` account - will be displayed. If you hover your pointer over the user record you will notice that there are three icons on the right.
    a. The icon shaped like a pencil will allow you to edit user information (i.e. first name, last name, and e-mail address).
    b. The icon shaped like a bag with an arrow back is for resetting a user's password. If you use this option a random, secure password will be generated and displayed in a dialog. Click the icon to the right of the field to copy it to clipboard.
    c. The icon shaped like a trashcan will allow you to delete the user after you confirm the deletion in a dialog.
5. Make any desired changes, and unless you chose to delete the record, click the *Save* button.

Tip
With regard to changing a user's password, a user can always change their own password. However, this requires knowledge of the existing password. If you encounter a user that has forgotten their password, you can reset it for them.

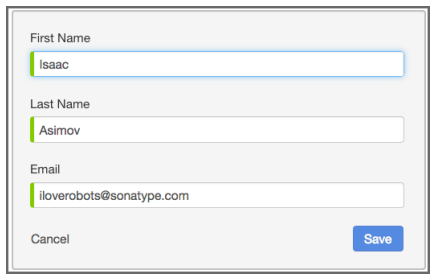


Figure 7.4: Edit User

7.4 LDAP Integration

Note
This section assumes you are familiar with LDAP (Lightweight Directory Access Protocol), and have an LDAP server currently in use.

You can configure IQ Server to work with an LDAP server for the purposes of authenticating users and managing users and groups. The integration process is discussed below in three parts:

- Configuring LDAP Server Connection
- Mapping LDAP Users
- Mapping LDAP Groups

7.4.1 Configuring LDAP Server Connection

The first step in integrating IQ Server and LDAP is to configure the LDAP server connection as follows:

1. Log into the IQ Server using a user account assigned to the System Administrator role.
2. In the System Preferences menu on the IQ Server toolbar, click *LDAP*.
3. In the *LDAP Servers* view, click *Add a Server*.
4. In the *Enter Server Name* box, type a name to identify the LDAP server. Click *Save*. The LDAP editor is displayed.
5. Enter the following parameters using values specific to your LDAP server. More information about these parameters is available in LDAP Configuration Parameters below.
    a. Protocol
    b. Hostname
    c. Port
    d. Search Base
    e. Authentication Type

f. Username
g. Password
h. Connection
i. Retry Delay

6. Click *Test Connection* to verify a connection can be made to the LDAP server.
7. Click *Save* when finished.

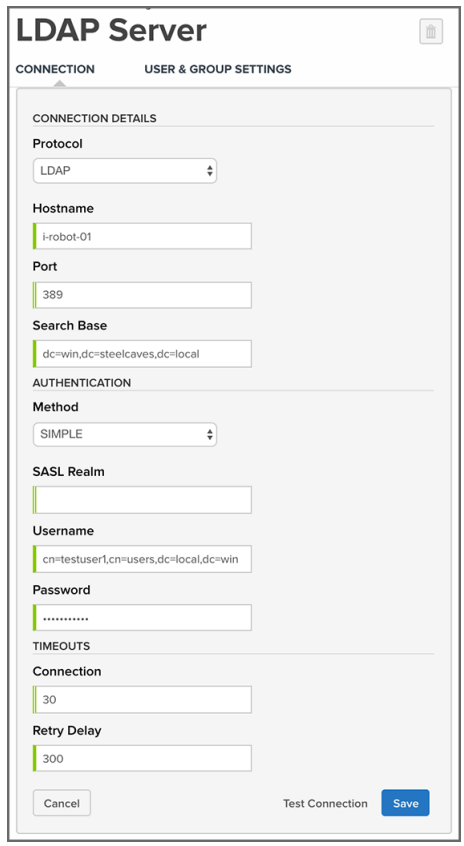


Figure 7.5: Sample LDAP Server Configuration

7.4.2 LDAP Configuration Parameters

Protocol
Valid values in this drop-down are LDAP and LDAPS, which correspond to the Lightweight Directory Access Protocol and the Lightweight Directory Access Protocol over SSL.

Hostname
The hostname or IP address of the LDAP server.

Port
The port on which the LDAP server is listening. Port 389 is the default port for the LDAP protocol and port 636 is the default port for the LDAPS protocol.

Search Base
The search base is the Distinguished Name (DN) to be appended to the LDAP query. The search base usually corresponds to the domain name of an organization. For example, the search base on the Sonatype LDAP server could be "dc=sonatype,dc=com".

Method
Four distinct authentication methods can be used when connecting to the LDAP Server:

- NONE (Anonymous Authentication) - Used when you only need read-only access to non-protected entries and attributes when binding to the LDAP server.
- SIMPLE - Simple authentication is not recommended for production deployments not using the secure LDAPS protocol as it sends a clear-text password over the network.
- DIGESTMD5 - This is an improvement on the CRAM-MD5 authentication method. For more information, see http://www.ietf.org/rfc/rfc2831.txt.
- CRAMMD5 - The Challenge-Response Authentication Method (CRAM) based on the HMAC-MD5 MAC algorithm. In this authentication method, the server sends a challenge string to the client, the client responds with a username followed by a Hex digest which the server compares to an expected value. For more information, see RFC 2195. For a full discussion of LDAP authentication approaches, see http://www.ietf.org/rfc/rfc2829.txt and http://www.ietf.org/rfc/rfc2251.txt.

SASL Realm
The Simple Authentication and Security Layer (SASL) Realm to connect with. Not available if authentication method is NONE.

Username
Username of the Administrative LDAP User to connect (or bind) with. This is a Distinguished Name of a user who has read access to all users and groups.

Password
Password for an Administrative LDAP User.

Connection
The number of seconds to try and connect to the configured server before returning an error.

Retry Delay
The number of seconds to wait before attempting to connect to the configured server again (after an error).

7.4.3 Mapping LDAP Users

The second step in integrating IQ Server and LDAP is to map LDAP users as follows:

1. In the LDAP editor, click the *User & Group Settings* tab.
2. Enter the following parameters under *User Element Mapping* using values specific to your LDAP server. More information about these parameters is available in LDAP User Parameters below.
    a. Base DN
    b. User Subtree
    c. Object Class
    d. User Filter
    e. Username Attribute
    f. Real Name Attribute
    g. E-mail Attribute
    h. Password Attribute
3. Click *Check User Mapping* to verify the correct information is mapped.
4. Click *Save* when finished.

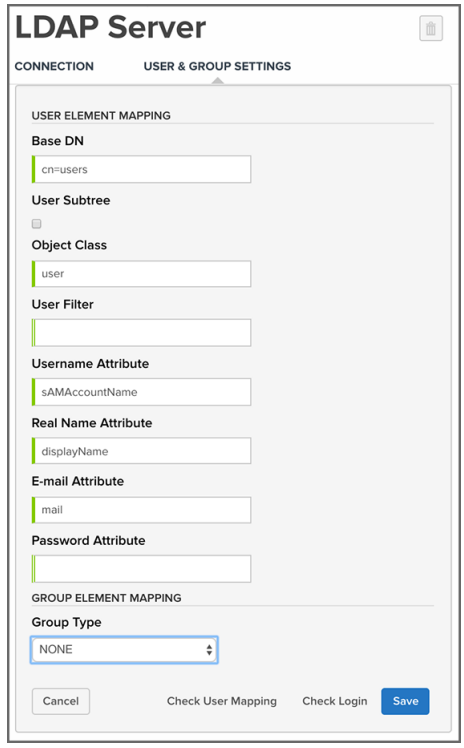


Figure 7.6: User Mapping Example

7.4.4 LDAP User Parameters

When you map LDAP users, some parameters are required and some optional. The required parameters are noted below.

Base DN ***(required)***
: Corresponds to the Base DN (Distinguished Name) containing user entries. This DN is going to be relative to the Search Base. For example, if your users are all contained in "cn=users,dc=sonatype,dc=com" and you specified a Search Base of "dc=sonatype,dc=com" you would use a value of "cn=users"

User Subtree
: Enable this parameter if there is a tree below the Base DN which can contain user entries. For example, if all users are in "cn=users" this field should not be toggled. However, if users can appear in organizational units below "cn=users", such as "ou=development,cn=users,dc=sonatype,dc=com" this field should be toggled

Object Class ***(required)***
The object class defines what attributes are expected for a given object. What is entered here must be the object class for the Username Attribute, Real Name Attribute, Email Attribute, and the Password Attribute.

User Filter
The user filter allows you to isolate a specific set of users under the Base DN.

Username Attribute ***(required)***
This is the attribute of the Object class which supplies the username.

Real Name Attribute ***(required)***
This is the attribute of the Object class which supplies the real name of the user.

E-Mail Attribute ***(required)***
This is the attribute of the Object class which supplies the email address of the user.

Password Attribute
This is the attribute of the Object class which supplies the User Password. By default it is not required, which means authentication will occur as a bind to the LDAP server. Otherwise this is the attribute of the Object class which supplies the password of the user.

7.4.5 Mapping LDAP Groups

In most LDAP implementations, users are mapped into groups with different permissions. If LDAP groups are not mapped, then all LDAP users are pulled in from the Base DN, which may give unintended access to some users.

The third step in integrating IQ Server and LDAP is to map the LDAP groups as follows:

1. In the LDAP editor, click the *User & Group Settings* tab.
2. In the *Group Element Mapping* section, select a *Group Type*.
    a. For a *Dynamic* group, enter a *Member of Attribute* and use the slider to enable or disable *Group Search*.
    b. For a *Static* group, enter the following parameters using values specific to your LDAP server: *Base DN*, *Object Class*, *Group ID Attribute*, *Group Member Attribute*, and *Group Member Format*.

Tip
Disabling group search for Dynamic Groups may improve performance but will not return group results when searching IQ Server users.

1. Click *Save* when finished.

For more information about these group parameters, see LDAP Group Parameters below.

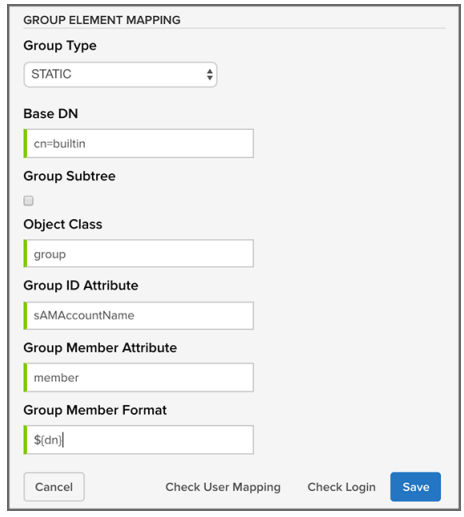


Figure 7.7: Group Mapping Example

7.4.6 LDAP Group Parameters

Groups are generally one of two types in LDAP systems: static or dynamic. For static groups, users are explicitly assigned to the group. For dynamic groups, users are assigned to the group based on a set of common attributes.

Tip
Static groups are preferred over dynamic ones, and will generally perform better if you have a large number of LDAP users.

When you map LDAP groups, some parameters are required and some are optional. The required parameters are noted below.

7.4.6.1 Static Groups

Static groups are configured with the following parameters:

Base DN ***(required)***
: This field is similar to the Base DN field described for User Element Mapping. If your groups were defined under "ou=groups,dc=sonatype,dc=com", this field would have a value of "ou=groups"

Group Subtree
: This field is similar to the User Subtree field described for User Element Mapping. If all groups are defined under the entry defined in Base DN, this field should not be selected. If a group can be defined in a tree of organizational units under the Base DN, this field should be selected.

Object Class ***(required)***
: This is a standard object class defined as a collection of references to unique entries in an LDAP directory, and can be used to associate user entries with a group.

Group ID Attribute ***(required)***
: This field specifies the attribute of the Object class that defines the Group ID.

Group Member Attribute ***(required)***
: This field specifies the attribute of the Object class that defines a member of a group.

Group Member Format ***(required)***
: This field captures the format of the Group Member Attribute, and is used to extract a username from this attribute. For example, if the Group Member Attribute has the format `uid=brian,ou=users,dc=sonatype,dc=com`, then the Group Member Format would be `uid=${username},ou=users,dc=sonatype,dc=com`. If the Group Member Attribute had the format "brian", then the Group Member Format would be `${username}`.

7.4.6.2 Dynamic Groups

If your installation does not use static groups, you can configure LDAP integration to refer to an attribute on the User entry to derive group membership. To do this, select *Dynamic Groups* in the *Group Type* field.

Member of Attribute ***(required)***
: Dynamic groups are configured via the Member of Attribute parameter. This attribute of the user

entry will provide a list of LDAP groups that the user is a member of. In this configuration, a user entry would have an attribute such as *memberOf* which would contain the name of a group.

Group Search

Depending on the size of your enterprise, LDAP search could be slow. If you find this is the case, click the *Enabled* slider below *Group Search* to disable group search.

Tip

Disabling group search will exclude groups from search results when assigning users to roles. Searching for users will remain unaffected.

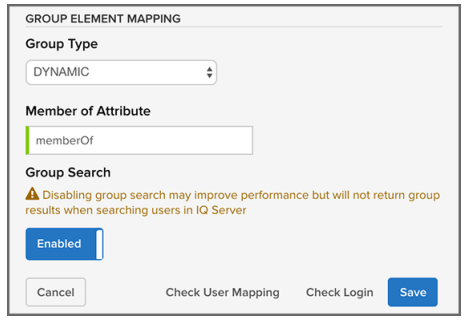


Figure 7.8: Dynamic Group Options

7.4.7 Verifying LDAP Configuration

The LDAP editor provides several ways to verify your LDAP configuration in IQ Server.

7.4.7.1 Test Connection

You can click *Test Connection* to verify your LDAP configuration. If you can't connect to your LDAP server, user and group mapping will fail as well.

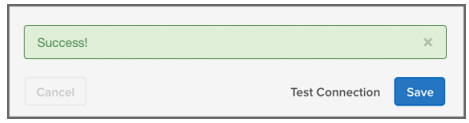


Figure 7.9: Testing LDAP Server

7.4.7.2 Check User and Group Mapping

You can click *Check User Mapping* to verify that usernames, real names, email addresses, and groups have been mapped correctly.

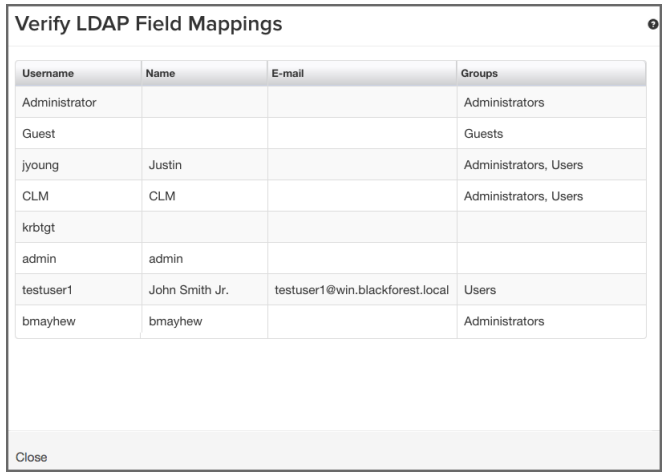


Figure 7.10: Checking User Mapping

7.4.7.3 Check Login

To ensure users can log in, click *Check Login* to open the "Test LDAP Login Credentials" dialog box. Enter a username and password, and click *Test Login.*

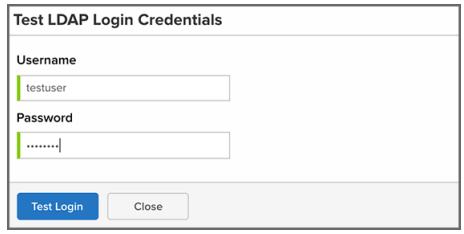


Figure 7.11: Checking User Login

7.4.8 Reordering LDAP Servers

Note
Reordering LDAP servers may impact user authentication and/or authorization.

When authenticating a user, IQ Server connects to LDAP servers in the order displayed in the *LDAP Servers* view. To manage the order, perform the following steps:

1. In the *LDAP Servers* view, click *Reorder List*.
   The *Reorder LDAP Servers* modal opens (see Figure 7.12).
2. Click on a server and use the up and down arrow buttons to reorder the list. Double arrow buttons move the selection to the top or bottom of the list.
3. Click *Save* when you are finished.

The server list in the *LDAP Servers* view updates to reflect the new order.

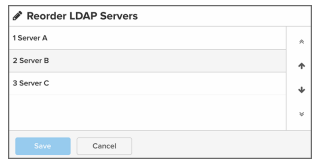


Figure 7.12: Reordering LDAP Servers Modal

7.5 Role Management

Roles provide a set of permissions that grant various levels of access and control over the IQ Server as well as the connected suite of tools. To grant permissions, you assign a user to either a system-wide administrator role or an organizational role at one of the levels in the system hierarchy: root organization, organization, or application. Which role and level you choose for a user determines what permissions that user receives.

You can assign roles to individual users or groups of users. IQ Server has a built-in group called *Authenticated Users* that contains any authenticated user. In addition, LDAP groups may be available, if you configured IQ Server to use an LDAP server with users mapped to specific groups.

7.5.1 Viewing Built-in Roles

IQ Server has several built-in roles, which are shown below. If one does not suit your needs, you can create a custom role.

Administrator Roles

- System Administrator - Manages system configuration and users, which includes LDAP and product license management as well as the ability to assign other users to the System Administrator role.
- Policy Administrator - Provides full control over organizations, applications, policies, policy violations and custom roles. Only the Policy Administrator has the ability to create organizations.

Organizational Roles

- Owner - Manages assigned organizations, applications, policies, and policy violations.
- Developer - Views all information for their assigned organization or application.
- Application Evaluator - Evaluates applications and views policy violation summary results.
- Component Evaluator - Evaluates individual components and views policy violation results for a specified application.

To view roles in IQ Server:

1. Click the *System Preferences* icon on the IQ Server toolbar.

2. Click *Roles* on the *System Preferences* submenu. A list of built-in roles is displayed.

Warning
Only a user assigned to an administrator role can see the information below. If you are using the built-in Admin user account, it is assigned to all administrator roles. It is highly recommended that you change the Admin password.

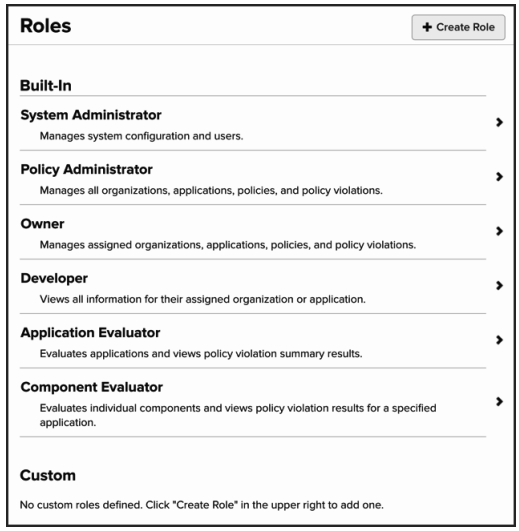


Figure 7.13: Role and Permission Descriptions

7.5.2 Viewing Permissions of Built-in Roles

To view permissions assigned to built-in roles:

1. Click the *System Preferences* icon on the IQ Server toolbar.
2. Click *Roles* on the *System Preferences* submenu. A list of roles is displayed.
3. Click the arrow next to a specific role to view its details and permissions.

The built-in roles have the permissions shown below.

Administrator Roles

Permissions ✓=can, ⊗=cannot	System Administrator	Policy Administrator
Administrator Permissions		
Edit System Configuration and Users	✓	⊗
Edit Custom Roles	⊗	✓
View All Roles	✓	✓
CLM Permissions		
Edit Proprietary Components	⊗	✓
Claim Components	⊗	✓
Edit IQ Elements	⊗	✓
View IQ Elements	⊗	✓
Evaluate Applications	⊗	✓
Evaluate Individual Components	⊗	✓
Add Applications	⊗	✓

Organizational Roles

Permissions ✓=can, ⊗=cannot	Owner	Developer	Application Evaluator	Component Evaluator
Administrator Permissions				
Edit System Configuration and Users	⊗	⊗	⊗	⊗
Edit Custom Roles	⊗	⊗	⊗	⊗
View All Roles	✓	⊗	⊗	⊗
CLM Permissions				
Edit Proprietary Components	✓	⊗	⊗	⊗
Claim Components	⊗	⊗	⊗	⊗
Edit IQ Elements	✓	⊗	⊗	⊗
View IQ Elements	✓	✓	⊗	⊗
Evaluate Applications	✓	⊗	✓	⊗
Evaluate Individual Components	✓	✓	⊗	✓
Add Applications	✓	⊗	⊗	⊗

Figure 7.14: Permissions of Built-in Roles

Note
IQ Elements includes organizations, applications, policies, component labels, license threat groups, application categories, policy violations and waivers.

7.5.3 Understanding the Importance of Hierarchy

The scope of permissions granted to a role is governed by where that role is assigned in the system hierarchy. A role assigned to:

- Root organization - Grants permissions to all organizations, applications, and repositories.
- Organization - Grants permissions to that individual organization and any applications attached to it.
- Application - Grants permissions only to the individual application.

Firewall solution users have an additional entity for which roles can be assigned:

- Repositories - Grants permissions to repositories.

7.5.4 Managing Administrator Roles

To manage administrator roles, you must log into IQ Server as a user assigned to the System Administrator role. By default, the built-in Admin user account is assigned to the System Administrator role.

7.5.4.1 Viewing Administrator Roles

1. Click the *System Preferences* icon on the IQ Server toolbar.
2. Click *Administrators*. A list of administrator roles and assigned members is displayed.

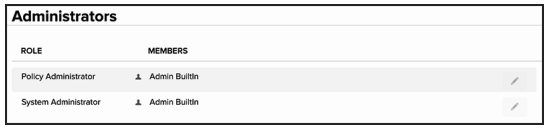


Figure 7.15: Viewing Administrator Roles

7.5.4.2 Assigning Users to Administrator Roles

1. Click the *System Preferences* icon on the IQ Server toolbar.
2. Click *Administrators*. The *Administrators* view is displayed.
3. Click the *Edit Role* button (looks like a pencil) for the role to which you want to add users. The *Administrators* view is updated to display the role and its members
4. Search for a user you want to add to the role by entering a full name or part of a name with an asterisk into the search box, then click *Search*. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, *isa**, **mov*, and **asi** will all match the name, "Isaac Asimov." Any matching names are displayed in the *Available* list.
5. To add a user to the role, click a user's name in the *Available* list and use the right arrow to move the name to the *Associated* list. To remove a user from a role, click a user's name in the *Associated* list and use the left arrow to move the name to the *Available* list.
6. Click *Save* to save the role assignment(s).

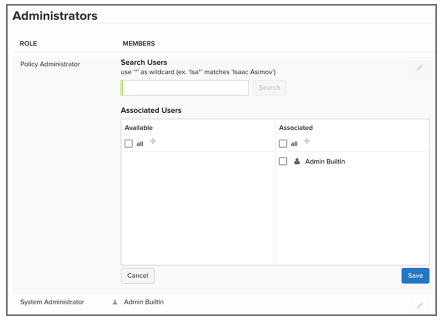


Figure 7.16: Assigning Users to Administrator Roles

7.5.5 Managing Organizational Roles

To manage organizational roles, you must log into IQ Server as a user assigned to the Policy Administrator role or Owner role. By default, the built-in Admin user account is assigned to the Policy Administrator role.

7.5.5.1 Viewing Organizational Role Assignments

To view organizational role assignments:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an entity (organization, application, or *Repositories*) in the sidebar.
3. Click *Access* in the menu bar at the top of the page to scroll to the *Access* section. Assigned roles are grouped as follows:
   - *Local* - Role assignments with a scope that's specific to the selected organization or application.
   - *Inherited* - Role assignments derived from an organization that's higher in the system hierarchy than the currently selected organization or application.

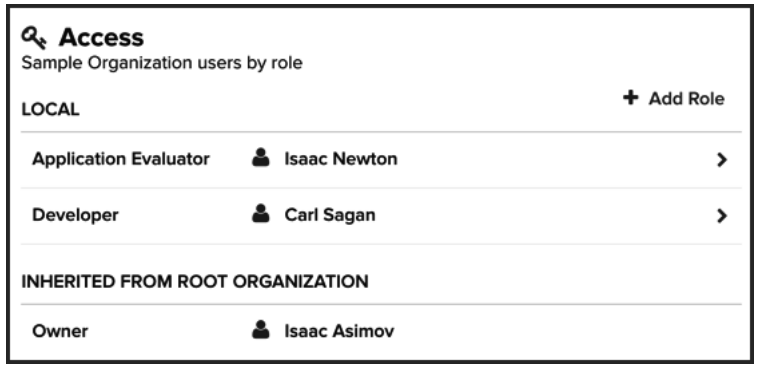


Figure 7.17: Viewing Role Assignments

7.5.5.2 Assigning Users to Organizational Roles

To assign a user to an organizational role:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an entity (organization, application, or *Repositories*) in the sidebar.
3. Click *Access* in the menu bar at the top of the page to scroll to the *Access* section.
4. Click the *Add Role* button. The Access editor is displayed.
5. In the *Role* box, select a user role.
6. In the *Search Users* box, search for a user by entering a full name or part of name with an asterisk, then click *Search*. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, *isa**, **mov*, and **asi** will all match the name, "Isaac Asimov." Any matching names are displayed below in the *Associated Users* list.

   Note
   If you integrated an LDAP server with IQ Server, the LDAP users and groups are also displayed in the search results. If you hover over a list item, the LDAP realm and email address are displayed when available.

7. In the *Associated Users* list, select a user in the *Available* column on the left, then click the right arrow button to move the user to the *Associated* column on the right. If you accidentally add a wrong user, select the user in the *Associated* column, then click the left arrow to return the user to the *Available* column.
8. Click Add.

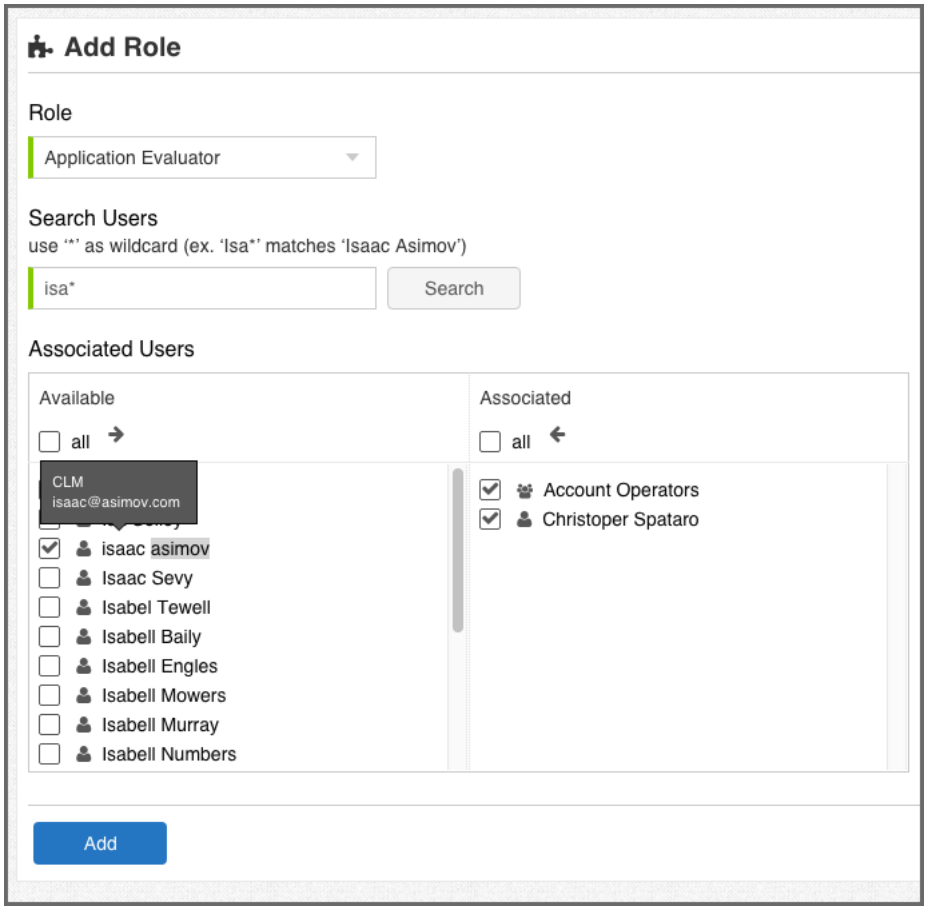


Figure 7.18: Assigning Users to Roles

Tip
If you want to continue adding role assignments for the selected organization, application or Repositories, click *Add Role* in the sidebar.

7.5.5.3 Editing Organizational Role Assignments

To edit an organizational role assignment:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the entity (organization, application or *Repositories*) in which the role is assigned.
3. Click *Access* in the menu bar at the top of the page to scroll to the *Access* section.
4. Click a listed role (or the chevron next to its name) to display the Access editor.
5. In the *Search Users* box, search for a user by entering a full name or part of name with an asterisk, then click *Search*. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, *isa**, **mov*, and **asi** will all match the name, "Isaac Asimov." Any matching names are displayed below in the *Associated Users* list.

Note
If you integrated an LDAP server with IQ Server, the LDAP users and groups are also displayed in the search results. If you hover over a list item, the LDAP realm and email address are displayed when available.

---

6. In the *Associated Users* list, you can add a user to a role by selecting the user in the *Available* column on the left and clicking the right arrow button to move the user to the *Associated* column. To remove a user from a role, select the user in the *Associated* column, then click the left arrow to return the user to the *Available* column.
7. Click *Update*.

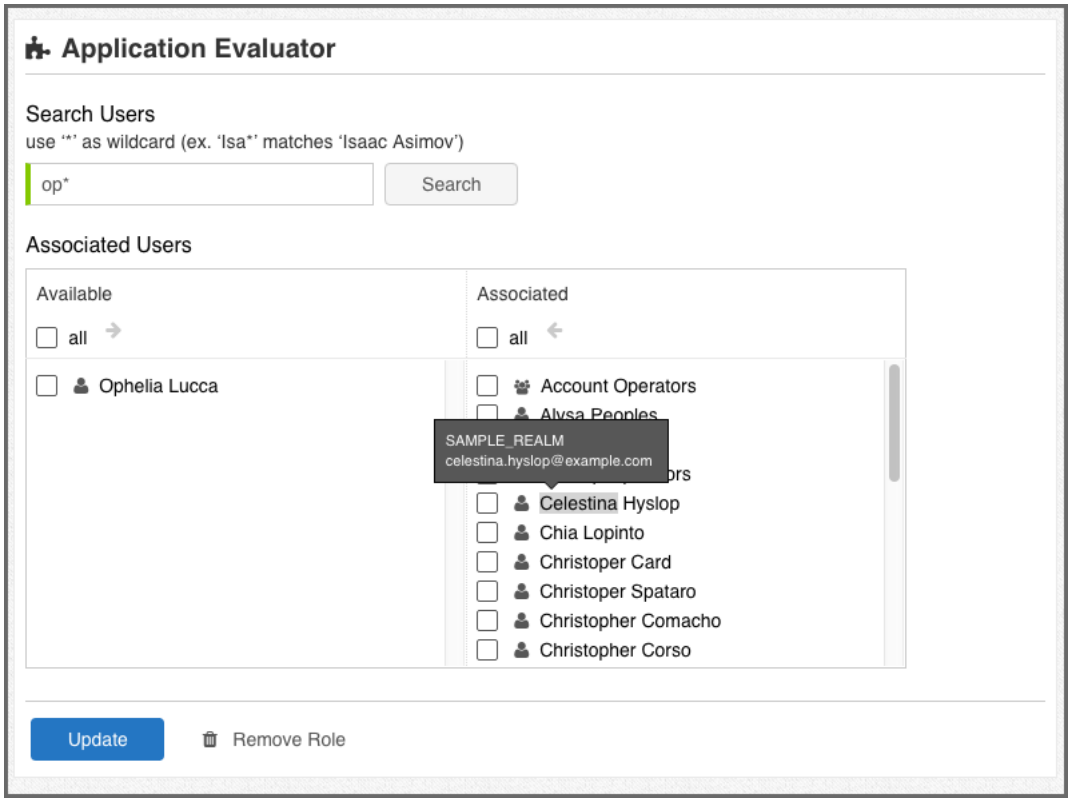


Figure 7.19: Editing Role Assignments

Tip
If you want to continue editing role assignments for the selected organization, application or Repositories, click a desired role in the sidebar.

7.5.5.4 Removing Organizational Role Assignments

To remove organizational role assignments:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.

2. In the sidebar, select the entity (organization, application or *Repositories*) in which the role is assigned.
3. Click *Access* in the menu bar at the top of the page to scroll to the *Access* section.
4. Click a listed role (or the chevron next to its name) to display the Access editor.
5. Click the *Remove Role* button, then click *Continue* to remove the role or click *Cancel* to keep the role.

Note
This is the equivalent of removing or disassociating all users from a role.

7.5.6 Creating Custom Roles

Important
You must have permission to *Edit Custom Roles* if you want create a custom role. The default Admin account and the built-in Policy Administrator role have this permission.

Custom roles allow you to fine tune IQ security permissions for different users. The following permissions are available for custom roles:

Administrator Roles

- View All Roles
- Edit Proprietary Components

Organizational Roles

- Claim Components
- Edit IQ Elements
- View IQ Elements
- Evaluate Applications
- Evaluate Individual Components
- Add Applications

Tip
To achieve desired behavior in the IQ user interface, you may need to assign View IQ Elements along with other permissions. For example, to allow a user to create applications in an organization but not edit the organization, you should add View IQ Elements and Add Applications to the role.

To create a custom role:

1. Click the *System Preferences* icon on the IQ Server toolbar and then click Roles.
2. Click the *Create Role* button.
3. Enter a name and description for the role.
4. Click the *Can/Cannot* slider to enable or disable a permission as desired.
5. Click the *Save* button.

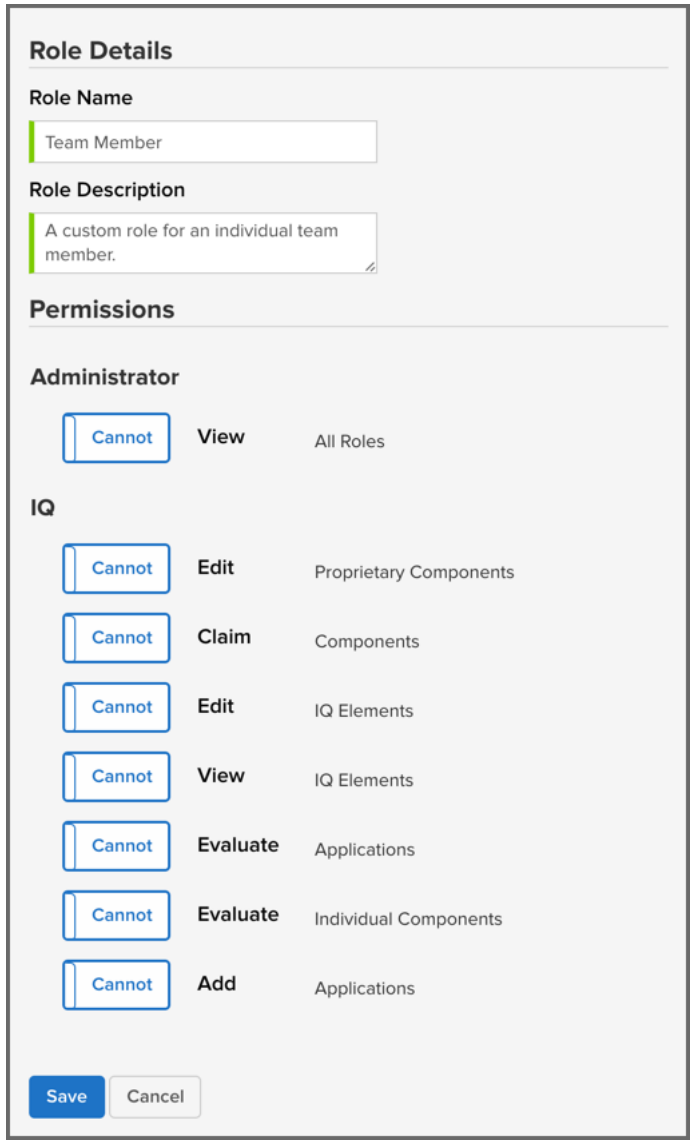


Figure 7.20: Creating Custom Roles

Note
Whenever a user assigned to a custom role with Add Applications permission creates an application, that user is automatically assigned to the Owner role for that application.

7.5.7 Assigning Groups to Roles without Searching

If you have an LDAP configuration that prohibits searching for groups, then the Access editor will have an additional section called *Associate Group.* You can use this section to enter manually a group name and add it to a role.

To assign groups to roles without searching:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select an entity (organization, application or *Repositories*).
3. Click *Access* in the menu bar at the top of the page to scroll to the *Access* section.
4. Open the Access editor by clicking *Add Role* or the chevron next to an existing role.
5. In the *Associate Group* box, enter the group name. The text must be an exact match.
6. Click *Add*. The group name is added to the *Associated* column without performing a search of users and groups.

7.5.8 Viewing Role Assignments

To view role assignments:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an organization or application in the sidebar. A page of customizable settings is displayed.
3. Click *Access* in the menu bar at the top of the page to scroll to the *Access* section. Users are displayed by their assigned roles for the selected entity (i.e. organization or application). The information is grouped by where the role assignments were made: locally in the current entity or inherited from an entity higher in the system hierarchy.

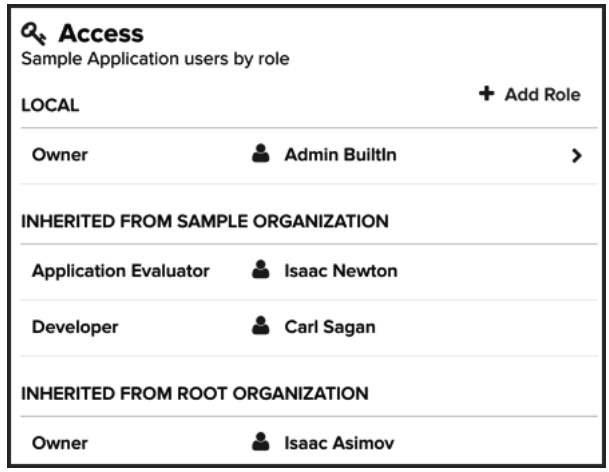


Figure 7.21: Viewing Role Assignments

Chapter 8

Organization and Application Management

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

When you launch the IQ Server for the first time, even after setting up and configuring your security parameters, there will be little to no information displayed. In many ways, it will look like a blank slate.

Before you can go further with anything else (like evaluating your applications and policy management), you must create at least one organization and one application.

In this chapter, we cover details around managing organizations and applications. We'll start off with some basics around the relationship, and then get right into how to create them.

8.1 Hierarchy

While we'll cover policy management a little bit later, it's important to understand policy management is centered around a principal of hierarchy. Specifically, a root organization is at the top of the hierarchy. Below it fall organizations, and then applications.

For many teams, the structure of organizations and applications will follow their own "command and control". In this instance, various business units are responsible for top level policies, and then they may add policies that are specific to their organization, or even to specific applications. As such, each business unit will have its own organization below the root organization, and those organizations will have unique applications below them.

For others, the applications their teams work on create more logical categories, such as "Internal". This allows a business to mimic commercial units where each business unit is a separate organization with applications below it.

For now though, the important pieces to remember are:

- The root organization acts as a container for organizations.
- Those organizations act as containers for applications.
- The root organization has no applications attached to it.
- Applications can only be attached to a single organization.

In the next section we'll talk more about the flow through these various pieces, which we refer to as inheritance.

8.2 Inheritance

Every organization has rules around component usage. While we will discuss policy in much greater detail in the Policy Management chapter, for now, let's just consider policy as a representation of all the rules and actions your business has to help identify which components your development teams should be using.

With rules as guidance, your teams will then simply follow that advice and go no further, or they might

develop additional rules given very specific needs. Whatever the case, all teams are still accountable for a core set of rules.

Now, let's think about how those move through our organization. We like to refer to this flow of rules, or policy, as inheritance. In general, whatever is specified for the root organization will extend out to all organizations. And, whatever is specified for an organization will extend out to any attached applications.

Of course there's opportunity for additional customization here as well. However, the real advantage to inheritance is when you want to make changes that affect more than one organization or one application. For example, if you managed everything at the application level, one change needed for all applications would need to be made to each one. That's not hard if you have ten applications, but what if you have a hundred, or thousands even? The same applies for organizations.

It is recommended that you use the root organization to set policy that applies globally to all organizations. You can fine tune policy at the organization level and allow your applications to inherit those changes.

8.3 Applications, Evaluations, and Reports

An application represents the link between what your team is actually developing and all the information that can be provided. You'll generally want this to be one-to-one. That is for every application your team is developing, you should create an application via the IQ Server. While this application is merely a representation of the real thing, it's no less important.

At the application level, you can do nearly everything you can for an organization. This includes creating policies and all the associated details. Additionally, whenever you perform an evaluation, and a report is generated, all that information will be associated to a specific application.

Don't get too stressed out if all of this seems like too much. In later chapters (see Policy Management and Application Composition Report), we'll discuss this in greater detail.

8.4 The Root Organization

Note
The Root Organization was introduced in IQ Server version 1.18. Your installation of the IQ Server may be affected depending on its version:

- If your original installation is version 1.18 or later, then the Root Organization is already part of your IQ Server system hierarchy. You can skip this section of the documentation.
- If you upgraded from version 1.17 (or earlier) to version 1.18 (or later), then you need to create the Root Organization by following the instructions here.

The Root Organization is a new entity at the top of the system hierarchy that allows you to set policy globally across all organizations and applications. Creating the Root Organization is a one time process, and occurs when the IQ Server is restarted. The process makes a permanent change to the system hierarchy that cannot be undone. For this reason, it is strongly recommended that you backup the IQ Server before you create the Root Organization. For information on backing up the IQ Server, see Backing Up the IQ Server in the IQ Server Setup chapter.

8.4.1 Configuring the Root Organization

The next step in the creation process is deciding how to configure the Root Organization. You have two choices:

- Choose an existing organization to act as a template for the Root Organization, which is configured with all of the policies and policy elements that you want to move to the Root Organization.
- Choose to create an empty Root Organization to which you can later add policy. To help you decide between the two, each is described in more detail below.

Using a Template Organization

Using a template organization enables you to take the configuration of an existing organization and move it to the Root Organization. The policy elements to be moved include all policies, component labels, license threat groups, application categories, and policy monitoring, but not security settings.

When the Root Organization is created, the policy elements in the template organization are compared by name to policy elements throughout the IQ Server hierarchy. Only names are compared, not attributes like constraints, notifications, or applied licenses. When a match is found, all references to the that policy element are changed to refer to the template organization policy element and the match is deleted. The policy element is now inherited from the Root Organization rather than its original organization.

Using a template organization saves you time and effort in configuring the Root Organization, especially if you have many policy elements that you want to use globally throughout your system hierarchy. For this reason, it is the recommended method for configuring the Root Organization.

Creating an Empty Root Organization

If you choose to create an empty Root Organization, it will be blank like any other newly created organization; it will have no policies, component labels, license threat groups, or application categories. You can certainly add policy later to the Root Organization. However, if you want to move a policy element from an existing organization, then the migration process is a manual task. It involves renaming elements and switching between organizations multiple times, which can take time and effort.

8.4.2 Creating the Root Organization

Only users assigned to the Policy Administrator role can initiate the Root Organization creation process. Once the Root Organization is created, its security permissions behave the same as with any other organization. For more information about security, see the Security Administration chapter.

To create the Root Organization:

1. Go to the Root Organization banner that appears below the IQ Server toolbar and click the *Get Started* button.
2. Select the how you want to configure the Root Organization and click *Continue*. Choosing an existing organization to act as a template for the Root Organization configuration is strongly recommended.
3. Restart the IQ Server to complete the Root Organization creation process.

When the IQ Server is restarted, the Root Organization is created. The IQ Server hierarchy is permanently changed; the Root Organization is at the top, followed by organizations, then applications.

8.5 Viewing the Root Organization

When you launch the IQ Server for the first time, it has no organizations or applications except for the root organization. The root organization is always present and cannot be deleted. To view the root organization, click the *Organization & Policies* icon on the IQ Server toolbar. The root organization appears in the sidebar.

Note
You may also see *Repositories* in the sidebar of the **Organization & Policies** area. If you are a Nexus Repository Manager customer who has enabled the Audit and Quarantine features, you can use *Repositories* to set security for repository evaluation results. For more information, see the IQ for Nexus Repository Manager chapter.

Figure 8.1: Root Organization

The root organization allows you to set policy (rules) that are inherited globally by all organizations. This includes policy management features like policies, labels, license threat groups, application categories, and security, all of which are discussed in later chapters (see Basic Policy Management and Advanced Policy Management).

If you have permissions, you can change the name and icon attributes of the root organization:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Click the *Root Organization* icon in the sidebar.
3. Go to the *Actions* menu and click *Edit Org Name / Icon*.
4. In the *Edit Organization* dialog, set the following attributes:
    a. *Organization Name* - Enter a name into the text box.
    b. *Icon* - Select from three options:

- *Default* - Uses the default image.
- *Upload a custom icon* - Click the *Upload Icon Image* button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
- *Get a robot* - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.

5. Click the *Update* button.

8.6 Creating an Organization

Organizations are created via the IQ Server. As a general rule with any activity on the IQ Server, make sure you have proper access. In the case of organization creation, only members of the Policy Administrator role have the permission to perform this actions.

To create an organization:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Click the *Root Organization* icon in the sidebar to display the *New Organization* button.
3. Click the *New Organization* button.
4. In the *New Organization* dialog, set the following attributes:
   a. *Organization Name* - Enter a name into the text box.
   b. *Icon* - Select from three options:
      - *Default* - Uses the default image.
      - *Upload a custom icon* - Click the *Upload Icon Image* button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
      - *Get a robot* - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.
5. Click the *Create* button.

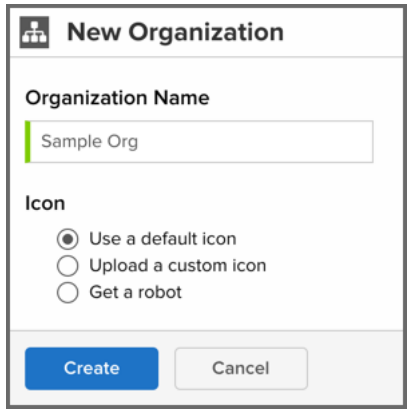


Figure 8.2: New Organization Dialog

8.7 Editing an Organization

To edit an organization:

1. Click the *Organization & Policies* icon on the IQ Server toolbar..
2. In the sidebar, select the organization you want to edit.
3. In the **Organization & Policies** area, go to the *Actions* menu and click *Edit Org Name / Icon*.
4. In the *Edit Organization* dialog, edit any the following attributes:
    a. *Organization Name* - Enter a different name into the text box.
    b. *Icon* - Select from three options:
        - *Use existing icon* - To use the current image.
        - *Upload a custom icon* - Click the *Upload Icon Image* button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
        - *Get a robot* - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.
5. Click the *Update* button to save changes.

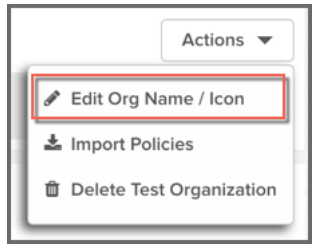


Figure 8.3: Editing an Organization

8.8 Deleting an Organization

To delete an organization:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization you want to delete.
3. In the *Delete <organization name>* dialog, click *Continue* to permanently delete the organization or click *Cancel* to keep the organization.

Warning
When you click *Continue* to delete an organization, this action cannot be undone.

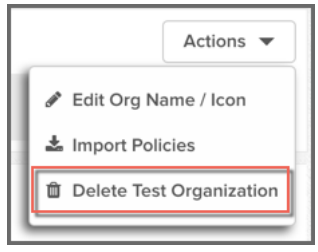


Figure 8.4: Deleting an Organization

8.9 Creating an Application

As a general rule with any activity, make sure you have proper access to the IQ Server. In the case of application creation, you will need to be a member of a role that has been granted the *Edit IQ Elements* permission. In the roles provided by default, this would be the Owner role for an organization.

When you create an application, you can assign the following items for identifying it:

- *Application Name* - The name that appears in the sidebar and at the top of the **Organization & Policies** area when the application is selected.
- *Application ID* - A unique identifier that's used by integration tools (e.g. Nexus IQ for Bamboo, IQ for IDEA, REST APIs and more) when performing component evaluations.
- *Icon* - A graphical representation of the application that appears at the top of the **Organization & Policies** area (when the application is selected in the sidebar) and in Reporting. You can select from the default icon, a robot icon, or a custom icon.

To create an application:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, click the organization to which you want to attach an application. A *New Application* button is displayed.
3. Click the *New Application* button.
4. In the *New Application* dialog, set the following attributes:
    a. *Application Name* - Enter a name into the text box.
    b. *Application ID* - Enter an identifier into the text box.
    c. *Icon* - Select from three options:
        - *Default* - Uses the default image.
        - *Upload a custom icon* - Click the *Upload icon image* button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
        - *Get a robot* - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.
5. Click the *Create* button.

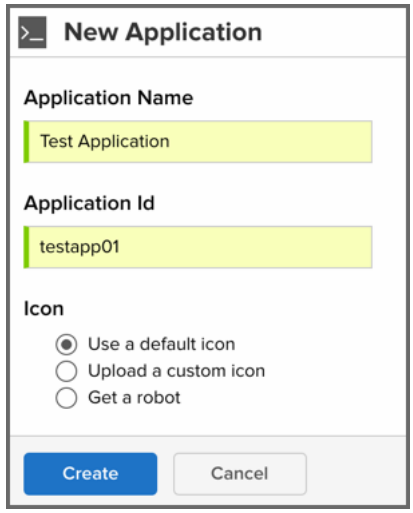


Figure 8.5: New Application Dialog

8.10 Editing an Application

To edit an application:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the application you want to edit.
3. In the **Organization & Policies Management** area, go the *Actions* menu and click *Edit App Name / Icon*.
4. In the *Edit Application* dialog, edit any of the following attributes:
    a. *Application Name* - Enter a different name into the text box.
    b. *Icon* - Select from three options:
        - *Use existing icon* - To use the current image.
        - *Upload a custom icon* - Click the *Upload Icon Image* button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
        - *Get a robot* - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.

5. Click the *Update* button to save changes.

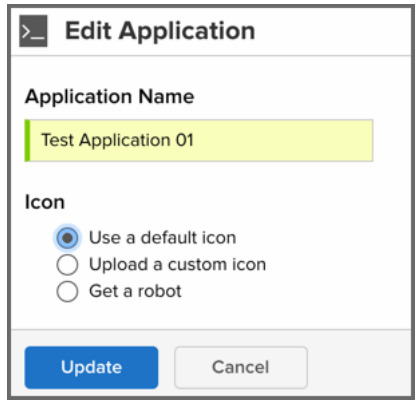


Figure 8.6: Editing an Application

8.10.1 Selecting an Application Contact

Optionally, you can select a contact person for an application. The contact is displayed at the top of the **Organization & Policies** page, as well as in the reporting area of the Nexus and the PDF version of the report.

To select a contact:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *Select Contact*.
4. In the *Select Contact* box, search for a user by entering a full name or part of name with an asterisk, then click *Search*. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, *isa**, **mov*, and **asi** will all match the name, "Isaac Asimov." Any matching names are displayed below the search box.
5. Select a user in the results list and click *Select*. The user's name is displayed at the top of the page below the application name.

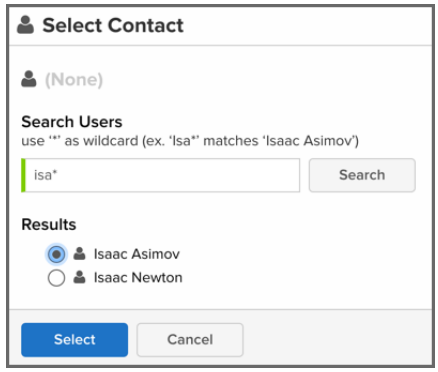


Figure 8.7: Selecting a Contact

8.10.2 Removing an Application Contact

To remove a contact:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *Select Contact*.
4. In the *Select Contact* box, click the *Clear Contact* button. In the alert box, click *Continue*. The user's name is removed from the top of the page.

8.10.3 Copying the Application ID to Clipboard

External tools integrated with IQ Server need to be configured with a public Application ID that identifies which application to use when evaluating components. You can speed the configuration process by copying the Application ID to the clipboard using the steps below.

To copy the Application ID to the clipboard:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.

2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *App ID to Clipboard*.

8.10.4 Changing an Application ID

The Application ID is a unique identifier used by external tools to integrate with IQ Server (e.g. Nexus IQ for Bamboo, IQ for IDEA, REST APIs, etc.) for performing component evaluations. If you change the Application ID, you must also reconfigure the external tool(s).

To change the Application ID for an application:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *Change App ID*.
4. In the *Change Application ID* dialog, enter a unique identifier in the *New Application ID* text box.
5. Click *Change* to save the new Application ID.

Important
Remember to reconfigure your external tools to use the updated Application ID.

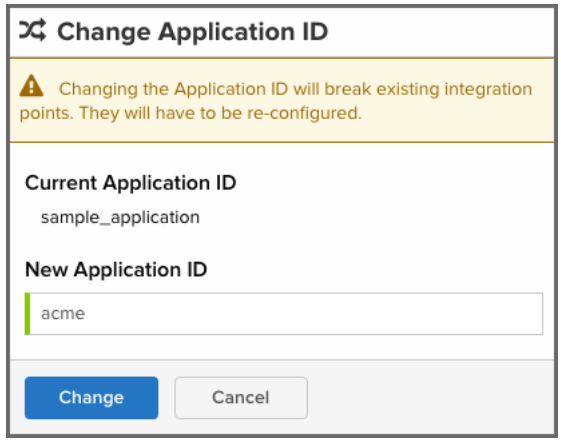


Figure 8.8: Changing an Application ID

8.11 Moving an Application

At some point, you may want to move an application to a different organization. When moving an application, it's not just the entity that moves, but also its configuration (i.e. policies, component labels, and waivers). If the application has any local configuration settings, those settings are unaffected and remain unchanged by a move. However, if the application has inherited any organization-level configuration, there is the potential for incompatibility between the application's source organization and the destination organization.

During a move, IQ Server compares the configuration of the source organization to the destination organization, and looks for matches by name. If the destination has configuration settings that match all of the source's configuration, then the application's parent reference is changed to the destination organization. If the source configuration is not matched in entirety, then IQ Server displays an error message with information about the missing configuration in the destination organization. You need to fix any discrepancy to ensure the source configuration is matched in the destination.

To move an application:

1. Make sure you have permission to modify the application in its original parent organization and permission to create applications in the destination organization.
2. Click the *Organization & Policies* icon on the IQ Server toolbar.

3. Select an application in the sidebar.
4. Go to the *Actions* menu and click *Move <application name>*.
5. In the *Move Application* dialog, select an organization from the *New Parent Organization* list and click *Update*. A *Success* message is displayed. If you get an error instead, the message lists the configuration discrepancies between the source and destination organizations that need to be resolved before the application can be moved.

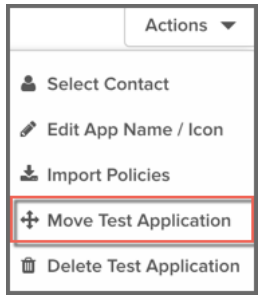


Figure 8.9: Moving an Application

Note
The user who moves an application is assigned to the Owner role for the moved application. After a move, you may want to review user role assignments and notification settings, and adjust them as desired.

8.12 Deleting an Application

To delete an application:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the application you want to delete.
3. In the **Organization & Policies** area, click the *Actions* menu and choose *Delete <application name>*.
4. In the *Delete <application name>* dialog, click *Continue* to permanently delete the application or click *Cancel* to keep the application.

Warning
When you click *Continue* to delete an application, this action cannot be undone.

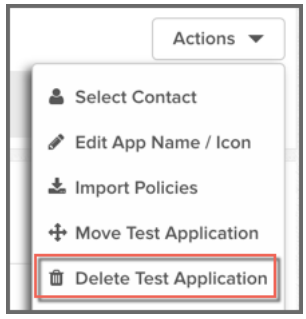


Figure 8.10: Deleting an Application

8.13 Viewing Organizations and Applications

From anywhere on the IQ Server, you can view organizations and applications by clicking the *Organization & Policy* icon on the IQ Server toolbar. Once you are in the **Organization & Policies** area, you'll see all of your organizations displayed in the sidebar. Click the expander arrow next to an organization to view applications associated with that organization.

If you want to search for a specific organization or application, type a name into the filter box. After entering three characters, the sidebar is filtered automatically to display any matching applications and organizations. To remove the filter, click the "*x*" button next to the filter box.

Figure 8.11: Organizations and Applications Filter

8.14 Managing Organizations and Applications

In the **Organization & Policies** area, after you select an organization or application in the sidebar, a page of customizable settings is displayed. You can use the menu bar at the top of the page to navigate to different settings for the selected organization or application. Each setting is discussed fully in later chapters:

- *Application Categories* - See Advanced Policy Management.
- *Policy* - See Basic Policy Management.
- *Component Labels* - See Advanced Policy Management and Application Composition Report.
- *License Threat Groups* - See Advanced Policy Management.
- *Access* - See Security Administration.

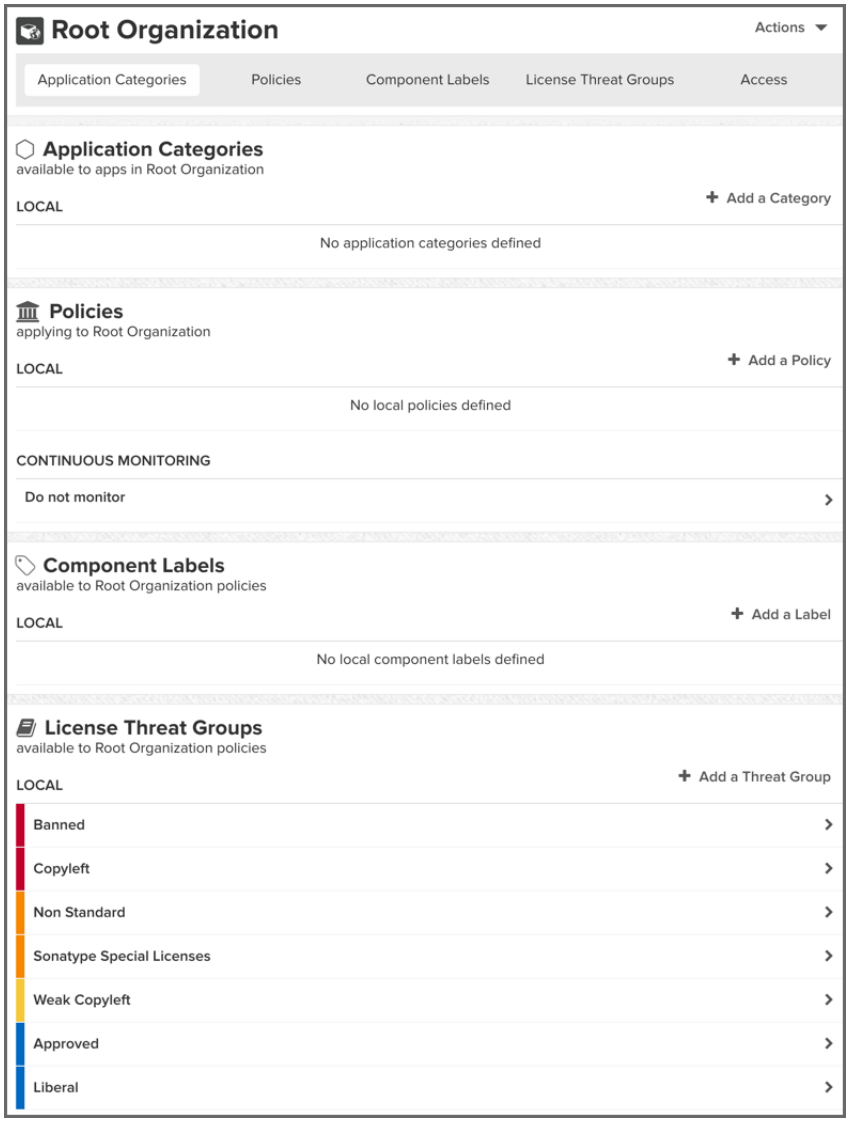


Figure 8.12: Organization & Policies View

Chapter 9

Basic Policy Management

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

This chapter covers how to create the policies that IQ Server uses for evaluating components in your applications and repositories.

9.1 What is a Policy?

A policy is a set of rules that guide certain actions when conditions are met. It's what IQ Server uses to identify and prevent risk associated with open source, third-party, or proprietary components that may enter a repository or exist in an application.

This chapter describes how to configure basic policies. If you want to augment your policies with component labels, application categories, or license threat groups, see the Advanced Policy Management chapter.

9.2 Getting Started with Policies

To get started with policies in IQ Server, it is strongly recommended that you download and import the Sample Policy Set into an organization, described in the section below. Creating policies from scratch can be a complex and labor intensive process, and the Sample Policy Set will give you a head start.

To begin, there are several fundamental questions to ask yourself about risk and the components you use:

- What types of risks do you want to know about: security vulnerabilities, licensing problems, quality issues (like age or popularity), or something else?
- At what stage in the development lifecycle do you want to know about those risks?
- How severe do you think those risks are?
- What actions do you want to take? Receive a warning? Stop a build?
- Who should be notified of those risks? Particular individuals or whole groups?
- Do you want to constantly monitor inventoried components for new risk?
- How should the policies be applied in the system hierarchy? Globally, at the root organization level? More narrowly, at the organization level? Or even more narrowly, at the application level?

9.2.1 Downloading the Sample Policy Set

You can download the sample policy set into an organization from here:

For IQ Server version 1.22 and newer:

Sonatype-Sample-Policy-Set-1.22.json

For IQ Server version 1.21 or older:

Sonatype-Sample-Policy-Set-1.20.json

Note
If you are using IQ Server 1.21 or older, import the 1.20 sample policy set. If you are using IQ Server 1.22, then import the 1.22 sample policy set.

The sample set contains policies for detecting and managing security, licensing, architectural, and popularity issues and includes some advanced policy features like application categories, component labels, and license threat groups. This policy set can help you gather information about the components used to build applications (including unknown and patched components), and understand how policy management will work for your environment.

Once the Sample Policy Set is downloaded, you can import it by following the instructions in the next section.

Tip
The Sonatype Sample Policy set is designed for use at the organization level. If you try to import the sample set into an application, you will receive an error message.

9.2.2 Importing Policies

After you acquire a policy file (in a .json format) such as the Sample Policy Set, follow these steps to import it into IQ Server.

1. Log into IQ Server using an account that has permission to import policies into a specific organization or application (including the Root Organization). At a minimum, the account should be assigned to the Owner role of the organization or application.
2. Click the *Manage Applications and Organizations* icon on the IQ Server toolbar.
3. In the sidebar, click the organization (or application) into which you want to import the policy.
4. Click the *Actions* menu and select *Import Policies.* The *Import Policy* dialog is displayed as shown in the figure below.
5. Click the *Choose File* button and select the policy .json file in the file browser.
6. Click the *Import* button.

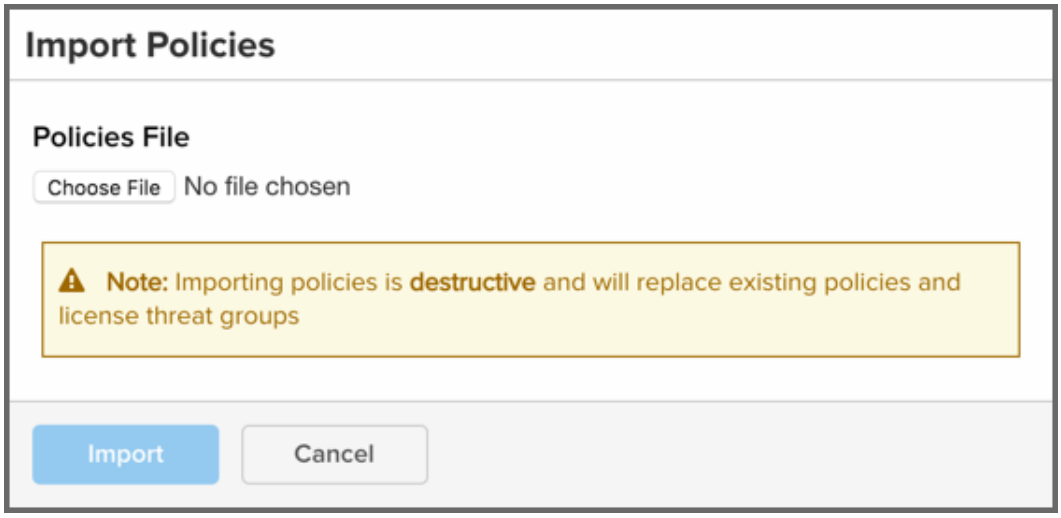


Figure 9.1: Import Policy Dialog

Rules for Importing Policies

If you want to import policies into an organization or application with existing policies (or application categories, component labels, and/or license threat groups), you should consider the following rules:

Importing Policies into an Organization

- Existing policies will be deleted during the import procedure.
- Importing policies also includes application categories, component labels, and license threat groups for which the following logic is used:
  - Application Categories - IQ Server attempts to match application categories against existing ones in a case-insensitive manner. This allows for updating the description or color of existing application categories, while preserving any current matching of categories between policies and applications.
  - Component labels - IQ Server attempts to match component labels against existing ones in a case-insensitive manner. This allows for updating the description or color of existing component labels, while preserving any triage effort already done to apply these labels to components. If your import contains component labels that aren't already present in the system, they will be created.
  - License Threat Groups - IQ Server will delete all existing license threat groups, and then import the new ones.

Importing Policies into an Application

- Duplication of organization policies is not allowed.
- When a policy is imported, any existing application policies are deleted and replaced with the imported one.
- For importing component labels, the same logic applies as at the organization level. That is, IQ Server attempts to match component labels against existing ones in a case-insensitive manner. This allows for updating the description or color of existing component labels, while preserving any triage effort already done to apply these labels to components. If your import contains component labels that aren't already present in the system, they will be created.
- Attempting to import policies that contain application categories will cause the entire import to fail.

9.3 Viewing Policies

You can view policies, including those imported with the Sample Policy Set, by following these steps:

1. Log into IQ Server using an account that has permission to "View IQ Elements" for the specific organization or application. At a minimum, the account should be assigned the role of Owner or Developer for that organization or application.
2. Click the *Organization & Policies* button on the IQ Server toolbar.
3. Select the desired organization or application in the sidebar.
4. Click the *Policies* button in the menubar near the top of the page to scroll to the *Policies* section.
5. Click the desired policy to open a view that displays policy details.

Note that policies are grouped according to where they are located in the system hierarchy:

- Local - The policy was added at the level of the selected organization or application.
- Inherited From [organization name] - The policy was added at some level higher in the system hierarchy.

When you open an inherited policy, the view is read-only. You can expand collapsed sections in the view to see details, but you cannot make changes to the policy settings. For information on how to modify a policy, see the Editing Policies section later in this chapter.

9.4 Creating Policies

Before you begin, you need to decide which level in the system hierarchy to use for new policies:

- Root Organization - Policies at this level are inherited by all organizations and applications. Use this level when you want to apply policies to every application and organization.
- Organization - Policies at this level are inherited by all applications attached to the organization. Use this level when you want to narrow the implementation of policies to a particular set of applications.
- Application - Policies at this level apply to an individual application only. Use this level when you want to apply policies to a single, unique application.

Note
If you have access to the *Audit* and *Quarantine* features of IQ for Nexus Repository Manager, policies for your repositories are managed at the Root Organization level only. They do not require a specific application or organization.

Tip
At the Root Organization and organization levels, you can use application categories to customize the implementation of policies across applications. Application categories provide a way to apply policies to a subset of select applications in an organization. For more details about application categories, see Application Categories in the Advanced Policy Management chapter.

Once you decide at which level to apply policies, you can proceed with creating custom policies. The overall process is only a few steps. However, the extent of customizable settings available to you can complicate the process. This section lists the basic steps for creating a policy, and includes links to more detailed information about each step in the Understanding the Parts of a Policy section of this chapter.

To create policies:

1. Log into IQ Server using an account that has permission to create policies in a particular organization or application (including the Root Organization). At a minimum, the account should be assigned to the Owner role of the organization or application.
2. Click the *Manage Applications and Organizations* icon on the IQ Server toolbar.

3. In the *Policies* section, click *Add a Policy*. A *New Policy* view will be displayed.
4. Enter a name for the policy. For more details, see Policy Name.
5. Select a threat level (from 10-0: 10 is the most severe threat, 0 is no threat). For more information, see Threat Level.
6. If the policy is being created at the organization level, select which applications in the organization the policy should apply to: all applications or only applications with selected application categories. If the latter, then click the specific application categories to select them. For more details, see Inheritance. Note that this setting is not available when creating a policy for an application.
7. Create a constraint with conditions. For detailed information, see Constraints and Conditions.
8. Add actions and/or notifications at a desired stage in the development lifecycle. For more information, see Actions and Notifications.
9. Click *Create* to save the policy.

After at least one policy is created (or imported), you can run an evaluation of an application to gather intelligence about its components and identify any vulnerabilities. The evaluation results, which include policy violations, are displayed in the Application Composition Report. For more information, see the Manual Application Evaluation and The Application Composition Report chapters.

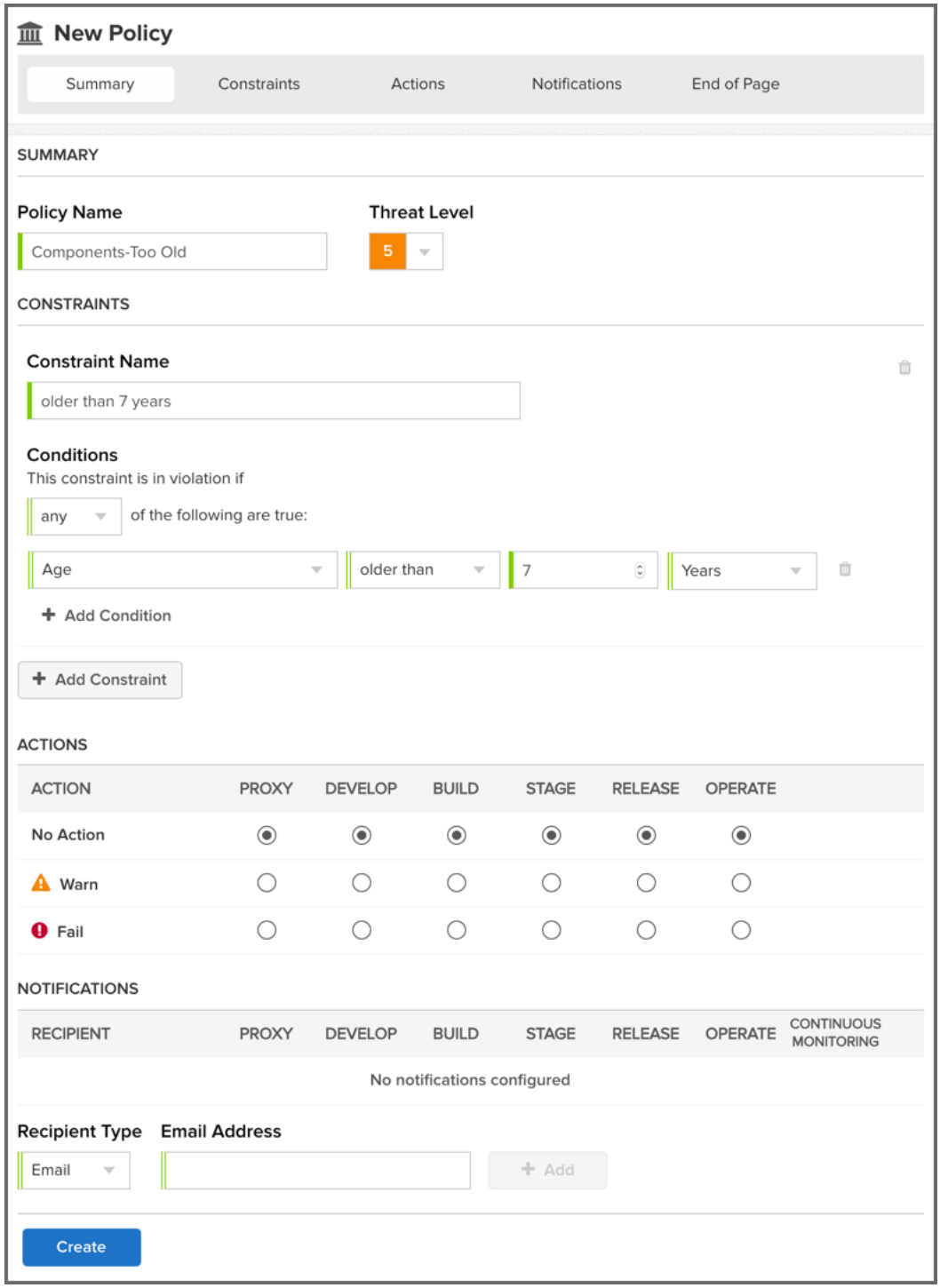


Figure 9.2: New Policy View

9.5 Editing Policies

At some point, you may want to edit an existing policy. For example, you'd like to modify a policy in the Sample Policy Set to suit the needs of your development team. The process for editing a policy is almost the same as creating one; it's only a few steps. However, the extent of customization you can do may make the process more complicated. This section lists the overall steps for editing a policy, and includes links to more detailed information in the Understanding the Parts of a Policy section of this chapter.

To edit policies:

1. Log into IQ Server using an account that has permission to edit policies in a particular application or organization. At a minimum, the account should be assigned to the Owner role of the organization or application.
2. Click the *Manage Applications and Organizations* icon on the IQ Server toolbar.
3. In the sidebar, select the organization or application in which the policy was created.
4. In the *Policies* section under *Local*, click the policy you want to edit. If the policy is listed in an *Inherited From* section, then it was created at a higher level in the system hierarchy; you must go to the level in which it was created to edit it.
5. In the *Edit Policy* view, you can change the following settings:
   - Enter a new name. See Policy Name for more information.
   - Select a different threat level. See Threat Level for more details.
   - If at the organization level, change which applications the policy applies to: all applications or only applications with selected application categories. If the latter, then click the specific application categories to select them. See Inheritance for more information.
   - Add or modify a constraint with conditions. See Constraints and Conditions for further instructions.
   - Add or modify actions and/or notifications. See Actions and Notifications for more details.
6. Click *Update* to save the policy changes.

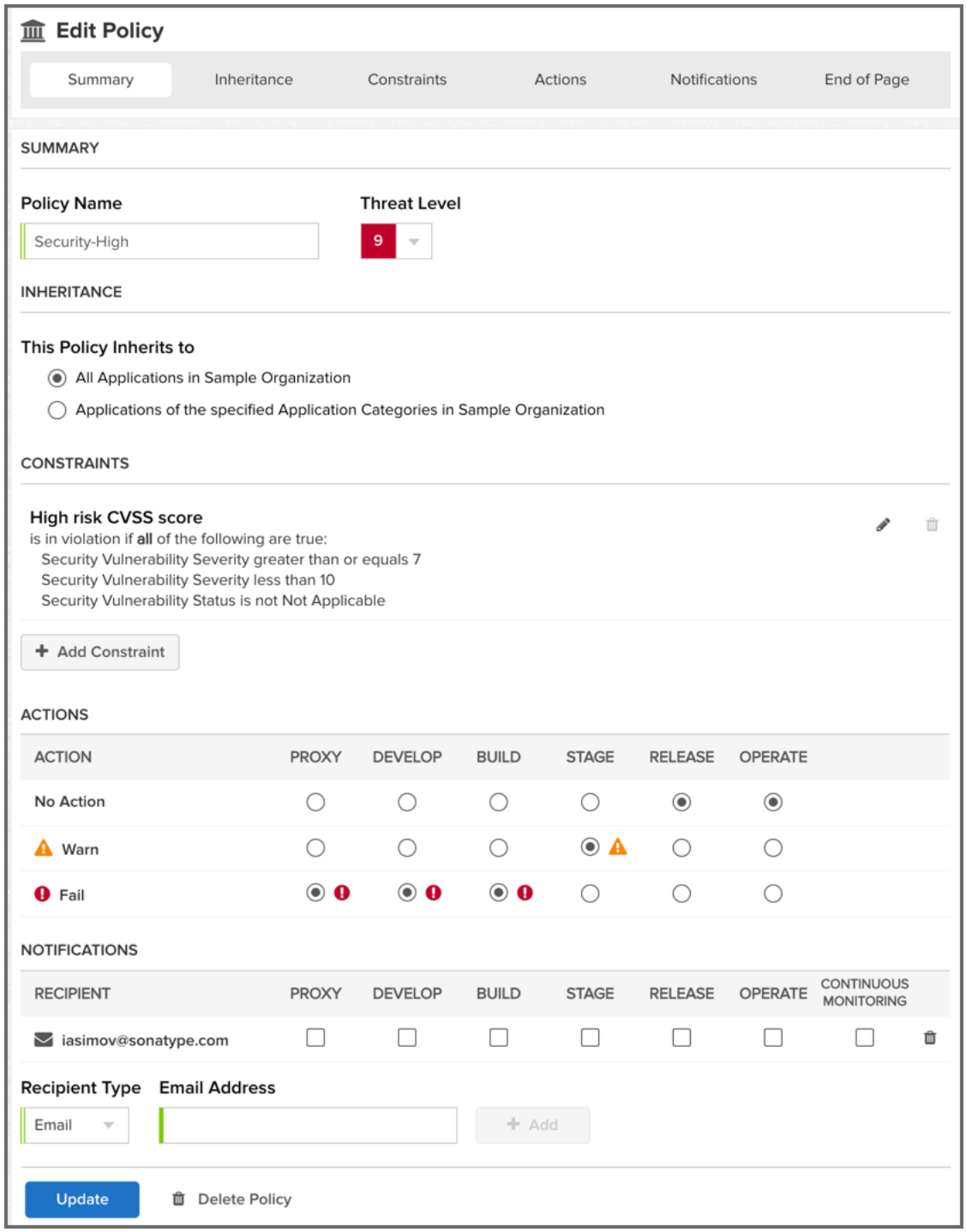


Figure 9.3: Edit Policy View

9.6 Deleting Policies

To delete policies:

1. Log into IQ Server using an account that has permission to delete policies in a particular application or organization. At a minimum, the account should be assigned to the Owner role of the organization or application.
2. Click the *Manage Applications and Organizations* icon on the IQ Server toolbar.
3. In the sidebar, select the organization or application in which the policy was created.
4. In the *Policies* section under *Local*, click the policy you want to delete. If the policy is listed in an *Inherited From* section, then it was created at a higher level in the system hierarchy; you must go to the level in which it was created to delete it.
5. In the *Edit Policy* view, click the *Delete Policy* button.
6. In the confirmation dialog box, click *Continue* to permanently delete the policy or *Cancel* to keep the policy.

Caution
Once you delete a policy, the action cannot be undone.

9.7 Understanding the Parts of a Policy

A policy is the set of rules or criteria used by IQ Server to evaluate components in your applications and repositories. It is made up of the following parts:

- Policy Name
- Threat Level
- Inheritance
- Constraints and Conditions

- Actions
- Notifications

Each of these parts is described below in more detail.

9.7.1 Policy Name

The *Policy Name* should be descriptive of the risk or violation you're trying to detect. In the text box, you can enter up to 60 characters: alphanumerics, underscores (_), periods (.), dashes (-), or spaces. The name you enter is used to identify the policy in IQ Server reports and views. To avoid confusion in your system hierarchy, it is recommended that you assign a unique name to every policy; try not to repeat names of policies created in other organizations.

9.7.2 Threat Level

The *Threat Level* is a subjective value placed on the perceived risk of a vulnerability. Its main purpose is for sorting policy violations in IQ Server reports and views; the violations with the highest threat level appear first followed by those with lower threat levels. The *Threat Level* values are grouped by severity and identified by specific colors as shown in the table below:

Table 9.1: Threat Levels

High	Red	8-10
Medium	Orange	4-7
Low	Yellow	2-3
Informational	Dark Blue	1
None	Light Blue	0

When setting the *Threat Level*, you should avoid causing unnecessary alarm for those who review policy violations. Select the lowest possible value that's helpful to you, such as an informational level (1) or low level (2-3). Save the high level values (8-10) for only the most critical vulnerabilities, if used at all.

9.7.3 Inheritance

The *Inheritance* setting is available only for organizations (including the Root organization). It allows you to specify how a policy is implemented when there are multiple applications attached to a specific organization. There are two choices:

- *All* - The policy is applied to every application attached to the organization.
- *Applications of the specified Application Categories in [organization]* - The policy is applied only to applications that have specific application categories assigned to them. With this setting, you select which application categories to use.

The latter choice lets you tailor the implementation of a policy to applications with similar characteristics by using application categories. For more information on how to create and assign application categories, see Application Categories in the Advanced Policy Management chapter.

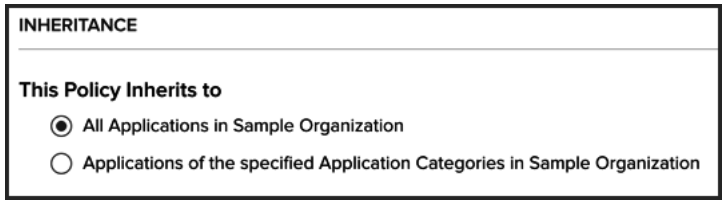


Figure 9.4: Inheritance Settings

Note
For policies at the Root Organization level, if you are a user of the Firewall solution, the *All* setting will include repositories as well applications. This will impact the policies used for the *Audit* and *Quarantine* features of IQ for Nexus Repository Manager.

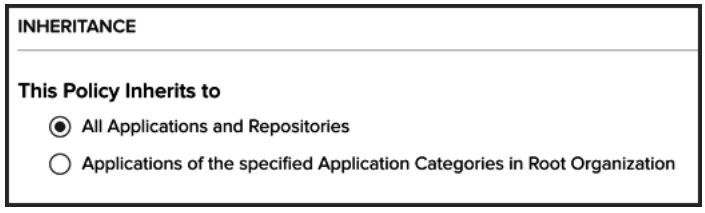


Figure 9.5: Inheritance Settings for Repositories

9.7.4 Constraints and Conditions

Constraints define the violations you want to detect. A constraint is essentially a container for conditions, and conditions are like the `if` part of an `if/then` statement. A policy must have at least one constraint, and each constraint must have a least one condition. When IQ Server evaluates your applications and the conditions of a constraint are met, then the policy is considered violated.

Constraints are made up of the following parts:

- *Constraint Name* - This is a label for the constraint. It should describe the violation you want to detect, for example, `High risk CVSS score` or `License needs legal review`.
- *Any* or *All* - It determines how constraints are evaluated. You can choose one of the following options:
  - *Any* - If any one of the conditions is met, then a policy violation is triggered. It is the equivalent of placing an `or` between each condition. This setting tends to produce a lot of policy violations.
  - *All* - If every condition is met, then a policy violation is triggered. This setting is the equivalent of placing an `and` between each condition. It tends to produce fewer policy violations.
- *Conditions* - IQ Server can detect many types of violations such as security vulnerabilities, licensing problems, quality concerns (like popularity or age), and more. To define a condition, you choose the type of condition you want from a built-in list and set any applicable parameters.

Adding or editing a constraint is basically the same process whether you're creating a new policy or editing an existing one. Once you navigate to the *New Policy* view or *Edit Policy* view, the following options are available for managing constraints:

- Click the *Add Constraint* button to create a new constraint.
- Click the *Add Condition* button to create a new condition.
- Click the *Delete* button (looks like a trash can) next to a constraint name or condition to delete the setting.
- In the *Conditions* section, select one of the condition types from the drop-down list. To set its parameters, you may need to make selections from a list and/or enter a value into a text box.

Figure 9.6: Constraints and Conditions

Available Condition Types

Below is a list of all available condition types in IQ Server with explanations of what each condition type means and the parameters you need to set for each one.

Label
: Verify if a specific component label *is* or *is not* assigned to a component. For information about creating and assigning component labels, see Component Labels in the Advanced Policy Management chapter.

License
: Verify if the component license *is* or *is not* a specified license. If you've used the Component Information Panel to set a component's license status to *Overridden,* then any licenses designated as *Declared* or *Observed* are ignored. If a component's license status has not been overridden, then any occurrence (declared or observed) of the specified license is considered a match. To learn more about licenses, see License Analysis Tab in the Application Composition Report chapter.

License Status
: Verify if the status of a user-defined license *is* or *is not* one of the following values:

- *Open*
- *Acknowledged*

- *Overridden*
- *Selected*
- *Confirmed*

To learn more about licenses, see License Analysis Tab in the Application Composition Report chapter.

License Threat Group
Verify if a component's license *is* or *is not* in a license threat group.

License Threat Group Level
Verify if the threat level of a component's license threat group is *less than or equal* or *greater than or equal* to a specified threat level value. To learn more about license threat groups, see License Threat Groups in the Advanced Policy Management chapter.

Security Vulnerability
Verify if a security vulnerability is *present* or *absent* in data sources searched by IQ Server. To learn more about security vulnerabilities, see Security Issues in the Application Composition Report chapter.

Security Vulnerability Severity
Verify if a security vulnerability with a numeric severity is =, <, ⇐, >, or >= to a specified value. For more information about security vulnerability severity, see Security Issues in the Application Composition Report chapter.

Security Vulnerability Status
Verify if a component's security vulnerability status *is* or *is not* one of the following values:

- *Open*
- *Acknowledged*
- *Not Applicable*
- *Confirmed*

For more information about security vulnerability status, see Security Issues in the Application Composition Report chapter.

Relative Popularity (Percentage)
Verify if the relative popularity of a component's version (as compared to other versions of the same component) is =, <, ⇐, >, or >= to a specified percentage value. For more information about a component's popularity, see Component Information Panel in the Application Composition Report chapter.

Age
Verify if a component is *older than* or *younger than* a specified value.

Match State
Verify if the comparison of a component to known components *is* or *is not* a match in one of the following ways: *Exact*, *Similar*, or *Unknown*. For more information about matching components, see Matching Components in the Application Composition Report chapter.

Coordinates
Verify if a component matches or does not match Maven or A-Name coordinates. For each type of coordinates, you enter specific attributes. You can use a wildcard (*) at the end of an attribute or leave an attribute blank to broaden the search.

Maven - You fill in a component's GAV, i.e. *Group ID*, *Artifact ID*, and *Version*, for example:

```
Group ID: org.sonatype.nexus
Artifact ID: nexus-indexer
Version: 1.0

Group ID: org.sonatype*
Artifact ID: nexus-indexer
Version: 1.*
```

A-Name - A-name is short for Authoritative Name, an identifier created by Sonatype to identify components agnostic of the repository format. You fill in a *Name*, *Qualifier*, and *Version*, for example:

```
Name: log4net
Qualifier: Framework 3.5
Version: 2.0.5

Name: log4net
Qualifier:
Version: 1.*
```

Proprietary
Verify if a component *is* or *is not* considered proprietary. For more information about proprietary components, see Managing Proprietary Components in the Application Composition Report chapter.

Identification Source
Verify if the identification of a component *is* or *is not* one of the following:

- *Sonatype* - When the identification is done based on IQ Server data sources.
- *Manual* - When the identification is done based on a component claimed by you.

For more information about the identification source see Component Identification in the Application Composition Report chapter.

9.7.5 Actions

Policy actions allow you to designate an action to take when violations occur at a particular stage in the development lifecycle. For each stage, you can assign one of the following actions:

- *No Action* - This is the default setting.
- *Warn* - Policy violations are worthy of a warning.
- *Fail* - Policy violations are severe enough to potentially halt the development lifecycle.

If you connected IQ Server with an external tool, the action can have a direct effect on the tool. When an external tool requests a policy evaluation (of an application, repository or component), IQ Server provides policy violation information along with the action, which the tool may (or may not) implement. For example, if you set the *Build* stage to *Fail* in a policy, a CI tool (such as Bamboo, Jenkins, or Hudson) may stop the build of an application when that policy is violated. Similarly, in a different tool, if you set a stage to *Warn*, a warning message may be displayed or logged in a file when policy violations occur. For more details on using actions, see Usage Suggestions for Each Stage.

To add actions to a policy:

1. In the *Organization & Policy* area, create a new policy or open an existing one for an organization or application.
2. In the Policy editor, click the *Actions* button to scroll to the *Actions* section.
3. Click the desired action--*No Action, 'Warn*, or *Fail*--at specific stage(s).
4. Click *Update* (or *Create*) to save the policy.

Figure 9.7: Policy Actions

Usage Suggestions for Each Stage

Proxy

Available only with the Nexus Firewall solution. *Proxy* refers to "Proxy Repository", or the point where components enter your repository manager. This is also referred to as the repository integration point. For more information on how to use the repository integration point, see the IQ for Nexus Repository Manager chapter.

When setting actions, *Warn* will have no effect on what happens to components in the repository. However, if you have enabled Quarantine on a repository, and set the action to *Fail*, any new components added to the repository will be quarantined (unavailable via the Repository Manager).

Warning

Quarantined components will not be available to your development team, including any attempt to build existing projects using those components.

Develop

While actions and notifications can be configured for this stage, they may not affect the functionality of an IDE.

Build

As you manage policies, making necessary adjustments over time, it's best to take an approach that allows for your development teams to be eased into dealing with violations. For this reason, it's better to start by simply warning when the CI build for an application contains components that violate your policies.

Stage Release

Because this stage gives the opportunity to prevent an application from being released with components that have violated policies, setting the action for a *Stage Release* to *Fail* is recommended. This is especially true for any policies that may include risk associated with security and/or licensing.

Release

While there should be the closest scrutiny of policy violations at this point, it is recommended that you fail a release based on severe violations. Ideally, in most cases, you should be finding only new violations.

Tip

If you have setup policy monitoring, it is a good idea to monitor your release stage, as this is likely the best representation of your production application.

Operate

Because the evaluation in the Operate stage is manual, a *Warn* or *Fail* action may not have any effect.

9.7.6 Notifications

Notifications enable you to send a summary of new policy violations when they occur at a specific stage of development. Notifications are sent whenever an application is evaluated either manually (e.g. using the *Evaluate Binary* command in the Organization & Policy area) or automatically via any tool integrated into the IQ Server (e.g. Nexus IQ for Hudson/Jenkins 1.x) or if *Continuous Monitoring* is activated. The notifications can be delivered to email addresses or create a JIRA ticket. The emails are sent to individual addresses or users assigned to a particular role such as Owner or Application Evaluator. The JIRA tickets are created as a specified issue type for the selected JIRA project.

Tip
When a notification is sent, it will only display new violations found in the latest evaluation. If you find yourself not receiving notifications, verify there are new violations, as well as confirm you have configured your IQ Server SMTP settings. For information on SMTP settings in IQ Server, see the Email Configuration section of the IQ Server Setup chapter.

Note
JIRA notifications are only available if a JIRA server is configured. See the JIRA Notifications section below.

When you have repository auditing setup, then notifications will be sent when a new component that violates policy enters your repository manager.

Note
The initial repository audit and re-evaluations of policies on repositories do not send notifications.

To set notifications in a policy:

1. In the *Organization & Policy* area, create a new policy or open an existing one for an organization or application.
2. In the Policy editor, click the *Notifications* button to scroll to the *Notifications* section.
3. Provide recipient information:

a. Select a *Recipient Type*. If *Email*, then enter an email address. If *Role*, then select a user role from the list. For *JIRA* notifications, enter a project and select an applicable issue type.

b. Click *Add* to insert the recipient.

4. Click to select the stage(s) for which to send notifications to a recipient.

Note

For the Continuous Monitoring stage, you must have monitoring activated for the application or a parent organization. To learn more about continuous monitoring, see Continuous Monitoring in Applications later in this chapter.

---

5. Click *Create* (or *Update*) to save the new policy.

Tip

To remove a recipient, click the *Delete* button (looks like a trash can) for a particular recipient.

Note

Be sure to select a stage for a recipient; if omitted, no notifications will be sent to the recipient.

Figure 9.8: Policy Notifications

Usage Suggestions for Notifications at Each Stage

Proxy

Consider setting up notifications to inform repository owners or Nexus Repository Manager administrators that are responsible for safe-guarding components entering the organization. You can also view any policy violations that occur during this stage in the *Repository Results*.

Develop

Policy violations triggered by IDE-related activity generally do not send any notifications.

Build
Consider setting up notifications to inform owners, as well as developers.

Stage Release
If something fails, the development process can not move forward. Make sure to notify anyone who is responsible for the application's release and/or capable of researching and addressing any violations.

Release
Similar to *Stage Release*, make sure to notify anyone responsible for ensuring an application does not go into production with policy violations.

Operate
Typically the application owner, or anyone responsible for ongoing maintenance of an application in production should be notified.

9.7.7 JIRA Notifications

The JIRA notifications, as stated in the previous section, will create a JIRA issue when new policy violations are discovered during the development process. To create JIRA notifications, you must configure a valid JIRA server and credentials in your IQ Server's config.yml. Without this configuration you will not see the *JIRA* option in the *Recipient Type* drop-down. An example config.yml file can be seen below:

```
# Notification JIRA settings.
jira:
  # The JIRA server address
  url: "https://127.0.0.1/"

  # The username used to connect to the JIRA server
  username: "anonymous"

  # The password used to connect to the JIRA server
  password: "guest"

  # Any custom fields that you wish to populate in the JIRA notification.  ↩
     Any
  # required fields must have default values configured here. Examples are ↩
      shown below.
  #customFields:
  #  reporter:
  #    name: "username"
  #  labels:
  #    - test
  #    - bug
  #  environment: "dev"
  #  duedate: "2016-11-01"
```

You may also configure any custom fields to be populated in the JIRA notification by utilizing the *customFields* section in the IQ Server config.yml. Any fields that are required on the issue type used for notifications must be configured this way otherwise the notification will fail.

Caution

A JIRA system user should be configured for integration with IQ Server. Any authenticated user of the IQ Server will be able to view the projects and applicable issue types available to the user configured in the config.yml. IQ Server users who can create and edit policy will be able to set up automated ticket creation for any project over which the configured JIRA user has authority to create tickets.

To configure JIRA notifications:

1. Select the Policy for which you will be notified when that policy is violated.
2. Select *JIRA* from the *Recipient Type* drop-down menu.
3. Select the *Project* and an *Issue Type*.
4. Click *Add* to add the notification.

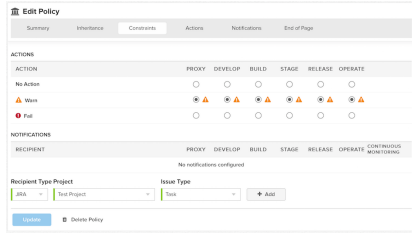


Figure 9.9: JIRA Notification Create

Once you have created the notification, you can then choose at which stage(s) you would like to be notified.

Figure 9.10: JIRA Notification Configuration

Note
In addition to having a valid JIRA server and credentials in the IQ Server's config.yml, if the JIRA issue type used for notifications has any required fields other than *Summary* and *Description*, those fields must have default values associated with them. Otherwise, IQ Server cannot create the issue in JIRA.

Note
In JIRA 7, the *Reporter* field of an issue is required by default. In order to create JIRA notifications, you must either turn off the required setting, assign a default value for the issue type, or configure it using the custom fields section in the config.yml.

9.8 Continuous Monitoring of Applications

At times, you may want to be notified when applications no longer in development (or being built on a regular basis) have components that violate a policy. For example, you'd like to learn of any security vulnerabilities or licensing issues that may arise after applications are deployed. Continuous Monitoring lets you use existing policies with notifications to constantly watch (once a day) for new violations at a specific development stage (such as Release).

Tip
Use Continuous Monitoring judiciously. If too many messages are sent for minor violations, it could result in notification fatigue for your development team. You may want to limit the monitoring to policies that detect high risk violations, like security vulnerabilities or license concerns.

Setting up Continuous Monitoring is a two-step press. First, you turn on Continuous Monitoring at the organization or application level, and specify which stage of the development lifecycle to monitor. Second, you turn on Continuous Monitoring at the policy level by creating a notification and selecting Continuous Monitoring in a policy. Each of these steps is described in more detail below.

Step 1: The Application or Organization Level

Continuous Monitoring, by default, is turned off for the Root Organization. Because all organizations and applications inherit policy settings from the Root Organization, it is turned off for those entities as well. You can turn on Continuous Monitoring for individual applications, or an organization (the parent) and

all of its associated applications (the children). You also specify which stage of the development lifecycle to monitor.

To turn on Continuous Monitoring for an application or organization:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization or application whose policies you want to monitor.
3. In the *Policies* section, under *Continuous Monitoring*, click the chevron next to *Do Not Monitor* (or *Inherit from [parent] (Do not monitor)*).
4. In the *Continuous Monitoring* view, click the desired stage.
5. Click *Update* to turn on Continuous Monitoring.

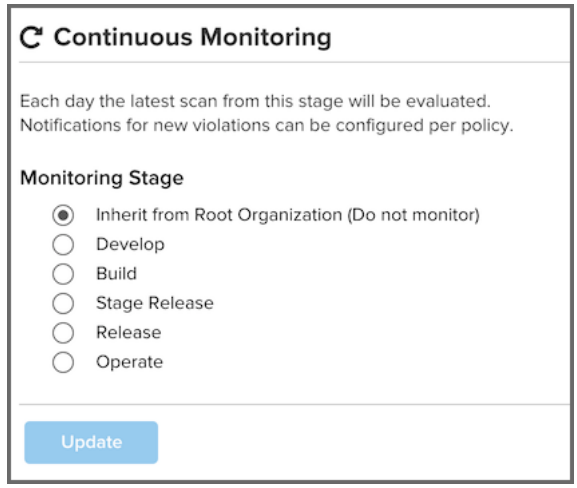


Figure 9.11: Continuous Monitoring View

Step 2: The Policy Level

When you turn on Continuous Monitoring at the policy level, you are identifying who should receive an email message when a violation of the current policy occurs at a particular development stage (specified in Step 1) whenever an evaluation is performed.

To turn on Continuous Monitoring in a policy:

1. In the *Organization & Policy* area, create a new policy or open an existing one for an organization or application.
2. In the Policy editor, click the *Notifications* button to scroll to the *Notifications* section.
3. Make sure the *Notficiations Recipient* list contains the desired email address to use for policy violation notifications. If necessary, add a new recipient.
4. For the desired email address, click *Continuous Monitoring* to select it.
5. Click *Update* to save the policy.

Figure 9.12: Continuous Monitoring Check Box

Note
If you perform Step 1, but omit Step 2, no notifications of policy violations will be sent when a Continuous Monitoring evaluation is run. You must perform Step 1 and Step 2 for Continuous Monitoring to work properly.

Turning off Continuous Monitoring

To turn off Continuous Monitoring:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the desired organization or application.
3. In the *Policies* section, under *Continuous Monitoring*, click the chevron next to the stage that's being monitored.
4. In the *Continuous Monitoring* view, click whichever of the following options is displayed:
   - For the Root Organization, click *Do not monitor*.
   - For other organizations and applications, click *Inherit from [parent] (Do not monitor)*.

5. Click *Update* to save your change.

Note
If an organization or application's parent has monitoring enabled, there is no way to disable its monitoring and the option will read *Inherit from [parent] (*Monitored Stage*)*. Monitoring must be disabled throughout an organization or application's hierarchy in order to disable it.

Setting the Notification Time

Once Continuous Monitoring is turned on, you may want to consider the time of day that notifications are sent. By default, they are sent at 0000 hours or 12:00 a.m. (per IQ Server time). You can change the notification time setting in IQ Server's config.yml file as follows:

```
# Hour of the day(0-23) to schedule Policy Monitoring execution. The
default is midnight.

policyMonitoringHour: 0
```

9.9 Proprietary Component Configuration

Proprietary components are unique or internal to your organization. When you evaluate an application that uses proprietary components, IQ Server is unlikely to find data about those components; they are probably listed under "Unknown" on the "Policy Violations" tab in the Application Composition Report. However, you can configure IQ Server to recognize those components as proprietary.

When you configure proprietary components, you use system hierarchy levels to set the scope for identifying the components:

- Root Organization - Identifies proprietary components in every organization and application.
- Organization - Identifies proprietary components in a particular set of applications.
- Application - Identifies proprietary components in a single application.

You also specify a string search pattern called a *proprietary component matcher* that IQ Server uses to find proprietary components. If matching components are found, they are displayed under *Proprietary*

on the *Policy Violations* tab in the Application Composition Report. There are two types of proprietary component matchers: *Package* and *Regular Expression*, which are described below.

Package Matchers

For *Package*, you specify a package name, for example, `com.sonatype`. In this case, all components that contain a package `com/sonatype` will be marked as proprietary. You should be as specific as possible, for the provided package is compared greedily against your scanned binaries. For instance, `com.sonatype` will match all of the following content locations:

- com/sonatype
- com/sonatype/anything
- com/sonatype/anything/more
- shaded/and/relocated/com/sonatype
- shaded/and/relocated/com/sonatype/anything

On the other hand, the following locations will not be matched:

- org/sonatype
- com/sonatypestuff
- com/sonatypestuff/anything

Regular Expression Matchers

For *Regular Expression*, you specify a regular expression that will be compared against the paths of all files scanned. If a file is found in the path, it is flagged as proprietary. For example, `test\.zip` will recognize anything in the top level directory named test.zip as proprietary. If you wanted to find test.zip nested anywhere in the scanned binaries, use `.*/test\.zip`.

Note
Occurrences inside an identified archive will make the binary proprietary as well. For example, if a proprietary `.zip` is found inside a `.jar`, the `.jar` is also considered proprietary.

For more information on regular expressions, see Oracle's Java documentation.

To configure proprietary components:

1. Click the *Organizations & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the desired organization or application.
3. In the *Policies* section, under *Proprietary Component Configuration*, click the chevron next to the number of matchers (local and/or inherited).
4. In the Proprietary Component Configuration view, add or remove matchers as desired. To add, select a *Package* or *Regular Expression* matcher type and enter a string search value. To remove, click the Delete icon (looks like a trash can) for items in the *Local* section.
5. Click *Update* to modify IQ Server's list of proprietary component matchers for the selected organization or application.

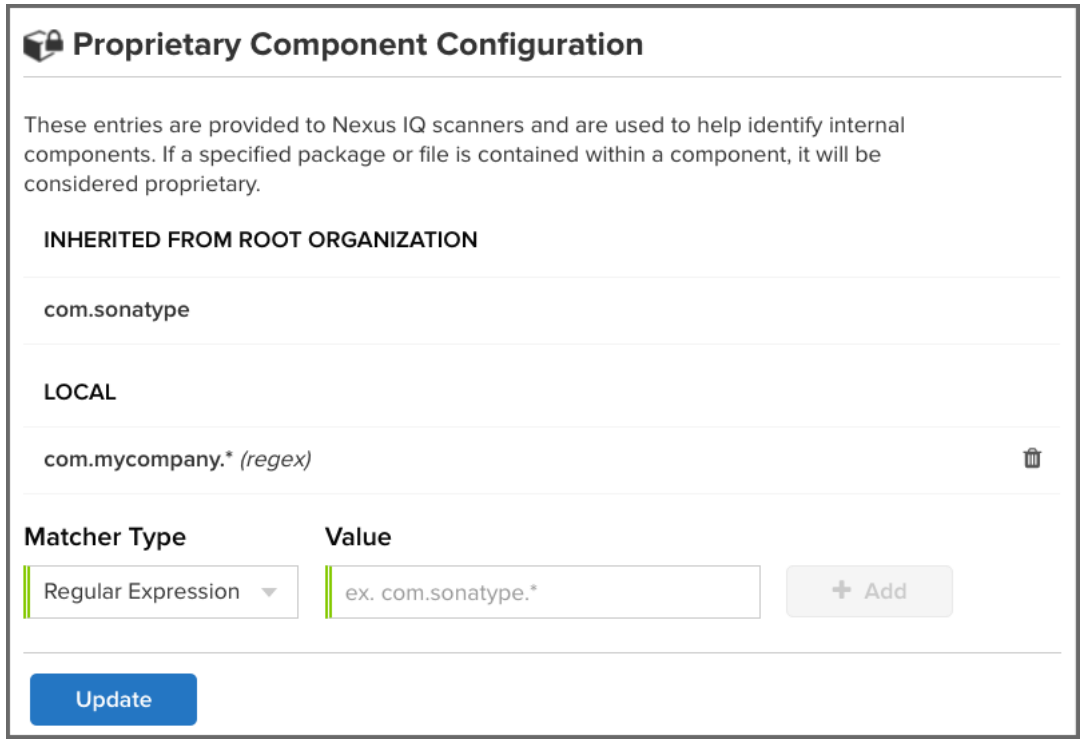


Figure 9.13: Proprietary Component Configuration

Usage Suggestions for Proprietary Components

Once proprietary components are configured, you can use a policy to prevent them from triggering policy violations. There is an example of this in the Sonatype Sample Policy Set; the *Component-Unknown* policy has the following constraint:

The policy is in violation if *all* of the following are true:

- *Match State* is *Unknown*
- *Proprietary* is *false*

This constraint excludes proprietary components from triggering policy violations.

Chapter 10

Advanced Policy Management

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

With a basic understanding of policy you can move on to more advanced topics in policy management. This primarily focuses on policy elements, which will provide greater adjustment when choosing what information triggers a policy violation.

At this time there are three specific policy elements we can work with:

- Component Labels
- License Threat Groups
- Application Categories

This chapter will cover these elements in detail.

10.1 Component Labels

Component labels are actually one of the more powerful features of policy management, and they should have a familiar look, since you've likely used other systems that employ a sort of tagging or labeling.

Essentially, component labels are metadata. More specifically a component label is metadata that is assigned to a component within the context of a particular application or organization. Labels can assist with identifying components you want to review, approve, or even avoid altogether. We call this component label assignment.

When component labels are assigned, this is an action that takes place in the application composition report. Before it can be assigned though, a label needs to exist for a particular organization or application.

As we learned in our Organization and Application Management chapter, inheritance plays a big role in policy. The same thing is true for labels, in that if a component label is created in an organization, any application attached to that organization will also have the label available for use when assigned. In fact, the system will prompt you to choose the scope (organization or application) a label should exist in when it is assigned.

Tip
You can customize a policy to use a component label as a condition when IQ Server evaluates applications. For more information about policies and creating conditions, see the Basic Policy Management chapter.

10.1.1 Viewing a Component Label

To view a component label:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an organization or application in the sidebar. A page of customizable settings is displayed.
3. Click *Component Labels* in the menu bar at the top of the page to scroll to the *Component Labels* section.

The *Component Labels* section displays two types of labels:

- *Local* - Component labels with a scope that's specific to the selected organization or application.
- *Inherited* - Component labels derived from an organization that's higher in the system hierarchy than the currently selected organization or application.

Figure 10.1: Component Label Section

10.1.2 Creating a Component Label

To create a component label:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization or application in which the component label will be used.
3. In the **Organization & Policies** area, click *Component Labels*.
4. In the *Component Labels* section, click the *Add a Label* button. The *New Component Label* editor is displayed.
5. In the *New Component Label* editor, set the following attributes:
    a. *Component Label Name* - Enter a name for the component label that is easily identified.
    b. *Short Description* - Enter a description that provides additional information about the component label.
    c. *Color* - Select a desired color for the component label.
6. Click the *Create* button to add the component label to the selected organization.

A few things to remember:

- An organization's component labels can be seen by any of its applications, the reverse is not true.
- Component labels can only be edited (or deleted) at the level they were created.

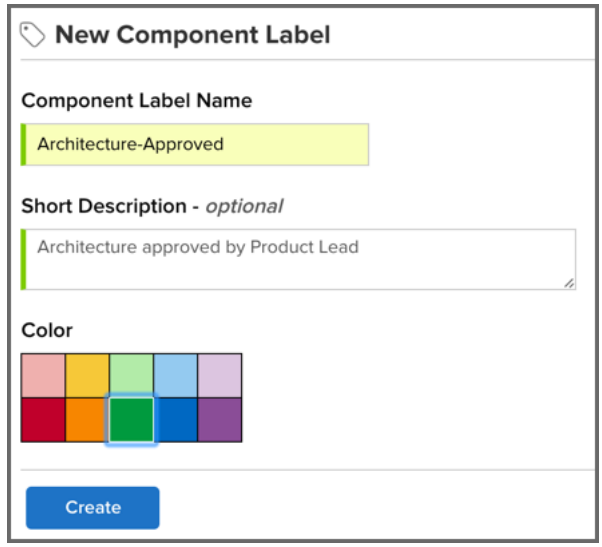


Figure 10.2: Using the Add a Label Button

10.1.3 Editing a Component Label

To edit a component label:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization or application in which the component label was created. The component label is displayed in the *Component Labels* section under the *Local* heading, and has a chevron in its row to indicate it's editable.
3. Click the component label you want to edit. The *Edit Component Label* dialog is displayed.
4. In the *Edit Component Label* dialog, you can change the following attributes:

a. *Component Label Name* - Enter a different name for the component label.

b. *Short Description* - Enter a description that provides additional information about the component label.

c. *Color* - Select a desired color for the component label.

5. Click the *Update* button to save the component label to the selected organization or application.

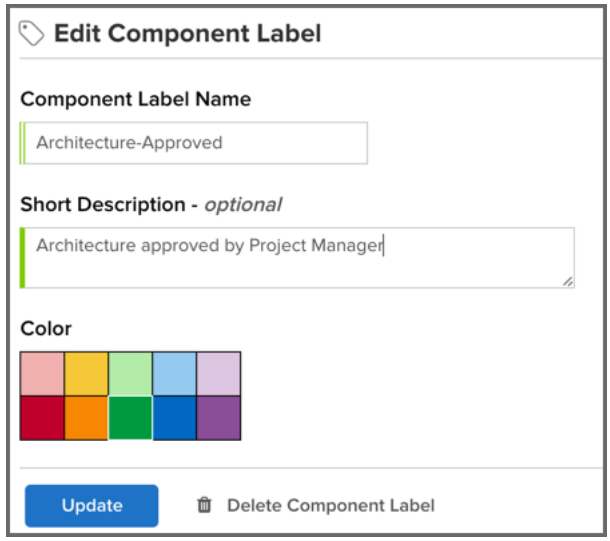


Figure 10.3: Editing a Component Label

10.1.4 Deleting a Component Label

To delete a component label:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.

2. In the sidebar, select the organization or application in which the component label was created. The component label is displayed in the *Component Labels* section under the *Local* heading, and has a chevron in its row to indicate it's editable.

3. Click the component label you want to delete. The *Edit Component Label* dialog is displayed as shown in the figure, Figure 10.3.

4. In the *Edit Component Label* dialog, click the *Delete Component Label* button. A *Delete Label* alert dialog is displayed.

Warning
When you delete a component label, the action cannot be undone.

5. In the *Delete Label* dialog, click *Continue* to delete the component label or *Cancel* to keep the component label.

10.2 License Threat Groups

License threat groups, are simply groups of licenses, broken into categories of severity for the various types of licenses. They can help you to achieve your goals related to enforcing the usage of components with licensing that matches the scope of your application.

Their primary purpose is to serve as the data points for the License section of the Application Composition Report. Moreover, they are a way to group risk, associated with licensing.

Tip
You can customize a policy to use a license threat group (or an unassigned license threat group) as a condition when IQ Server evaluates applications. For more information about policies and creating conditions, see the Basic Policy Management chapter.

10.2.1 Viewing a License Threat Group

To view a License Threat Group:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an organization in the sidebar. A page of customizable settings is displayed.

3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section. The list of License Threat Groups is organized by where the groups are defined: *Local* for the currently selected organization or *Inherited From* for an organization higher in the system hierarchy.

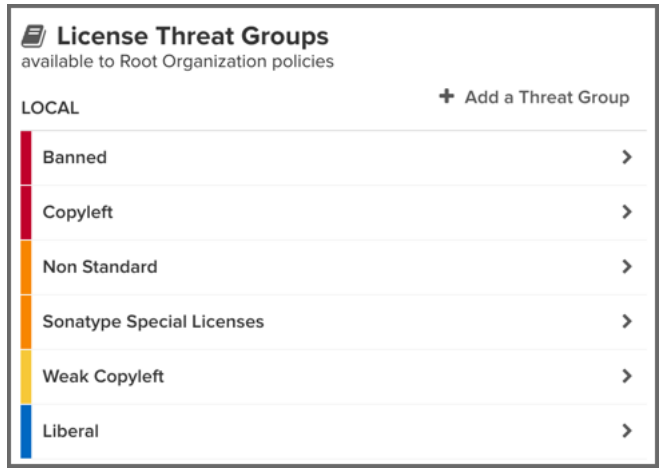


Figure 10.4: Viewing License Threat Groups

The following license threat groups are included by default for the root organization.

Banned
: Any licenses that should not be permitted in any circumstances. This license threat group contains the AGPL licenses by default.

Copyleft
: Strong copyleft licenses go a step further from weak copyleft licenses and mandate that any distributed software that links or otherwise incorporates such code be licensed under compatible licenses, which are a subset of the available open-source licenses. As a result, these licenses have been called viral.

Liberal
: These licenses allow you to do almost anything conceivable with the program and its source code, including distributing then, selling them, using the resultant software for any purpose, incorporating into other software, or even converting copies to different licenses, including that of non-free (so-called “proprietary”) software.

Non Standard
: Something out of the ordinary (e.g. If we ever meet, give me a beer license).

Sonatype Special Licenses
A license threat group for identifying situations where Sonatype has been unable to determine the license of a component.

Weak Copyleft
Free software licenses that mandate that source code that descended from software licensed under them, will remain under the same, weak copyleft, license. However, one can link to weak copyleft code from code under a different license (including non-open-source code), or otherwise incorporate it in a larger software. Otherwise, weak copyleft licenses allow free distribution, use , selling copies of the code or the binaries (as long as the binaries are accompanied by the (unobfuscated) source code), etc.

Note
Consult with your legal department for EXACT definitions. Information provided above is from the following reference.

10.2.2 Creating a License Threat Group

An important aspect of license threat groups is that each one also has a threat level, just like policy (from zero signifying no threat all the way up to 10). Unless you have specific legal recommendation / council, the default license threat groups will suffice, especially in the beginning.

If you desire, you can edit these default groups, or create entirely new ones. When creating license threat groups, keep in mind that they will be inherited from the organization to all associated applications.

To create a license threat group:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select an organization.
3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section.
4. Click the *Add a Threat Group* button.
5. In the *New License Threat Group* editor, set the following attributes:
    a. *License Threat Group Name* - Enter a name for the license threat group that is easily identifiable.

b. *Threat Level* - Select a number for the threat level that this group of licenses represents.

c. *Included Licenses* - Type a string of characters in the filter box or scroll the *Available* list to locate desired licenses by name.

   i. In the *Available* column on the left, select a license in the list, then click the right arrow button to move the license to the *Included* column on the right.

   ii. If you accidentally add a wrong license, select the license in the *Included* column, then click the left arrow to return it to the *Available* column.

6. Click *Create*.

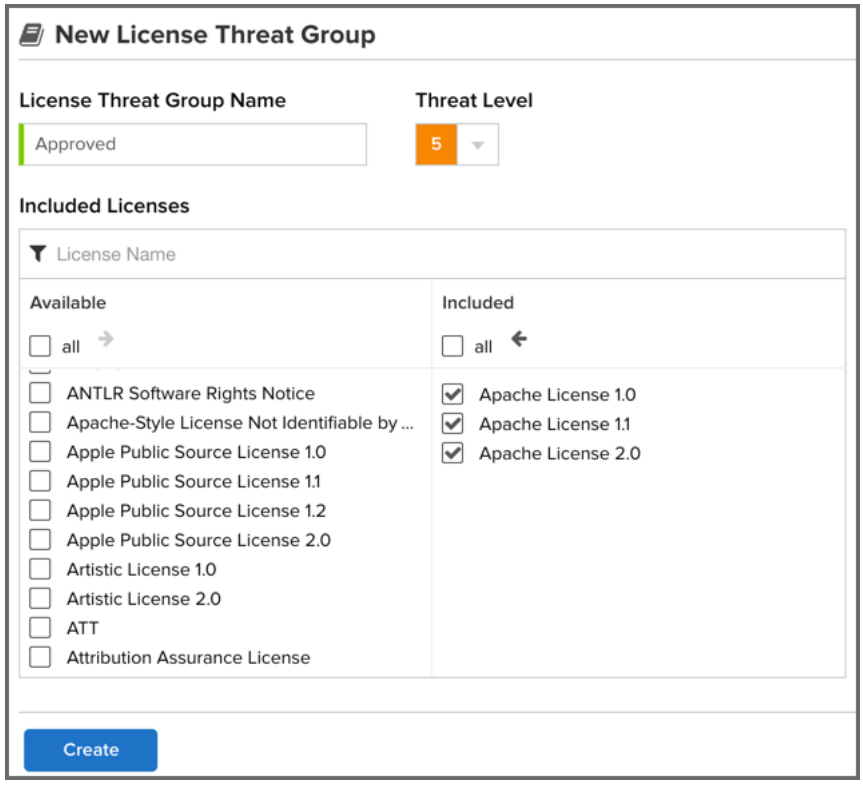


Figure 10.5: Creating a License Threat Group

Note

As of IQ Server 1.20, license threat groups are no longer created at the application level. If you previously had license threat groups in your applications, you can still edit them, but we encourage you to migrate those license threat groups up to the organization.

10.2.3 Editing a License Threat Group

To edit a license threat group:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization (or application, if created prior to IQ Server 1.20) in which a license threat group was created.
3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section.
4. In the list of License Threat Groups, click the one you want to edit (it has a chevron in its row to indicate it's editable).
5. In the *License Threat Group* editor, you can set the following attributes:
    a. *License Threat Group Name* - Enter a different name for the license threat group that is easily identifiable.
    b. *Threat Level* - Select a number for the threat level that this group of licenses represents.
    c. *Included Licenses* - Type a string of characters in the filter box or scroll the columns of licenses to locate desired licenses by name.
        i. To add a license, select the license in the *Available* column on the left, then click the right arrow button to move the license to the *Included* column on the right.
        ii. To remove a license, select the license in the *Included* column, then click the left arrow to return it to the *Available* column.
6. Click *Update*.

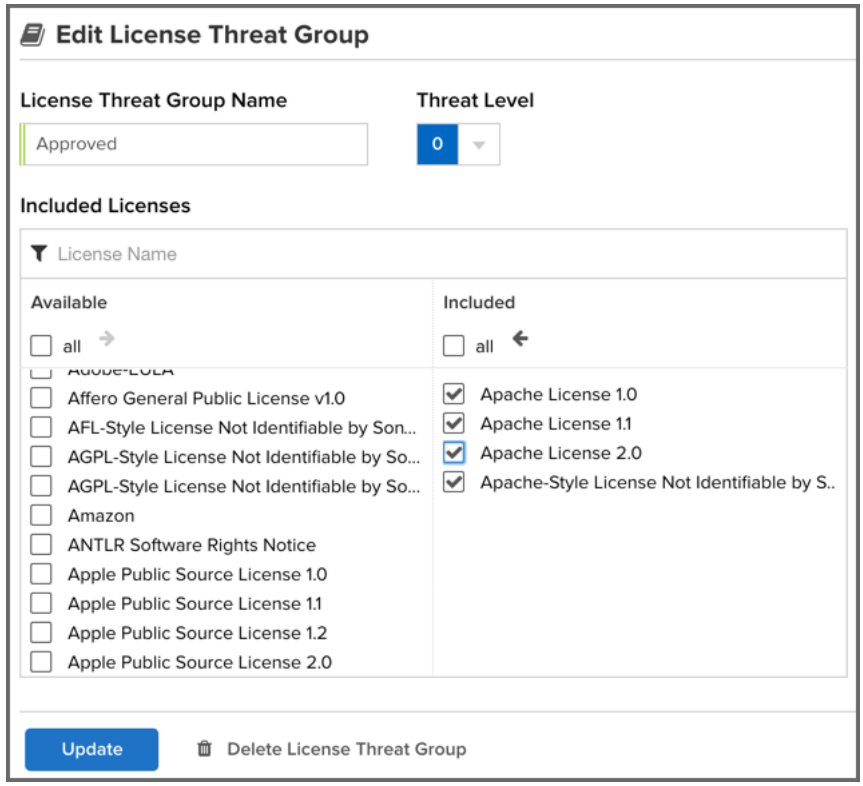


Figure 10.6: Editing a License Threat Group

10.2.4 Deleting a License Threat Group

To delete a license threat group:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization in which a license threat group was created.
3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section.
4. In the list of License Threat Groups, click the one you want to delete (it has a chevron in its row to indicate it's editable).

5. In the *License Threat Group* editor, click the *Delete License Threat Group* button. A warning message is displayed.
6. Click *Continue* to permanently remove the License Threat Group or *Cancel* to keep it.

10.3 Application Categories

In any given business, you could have hundreds, maybe even thousands of applications. Even if you are just getting started, it's likely you have a handful of applications. However, as unique as applications can be, they tend to share some similarities.

For example, you might have applications that process or store sensitive information, maybe even personally identifiable information for your users. Since attacks are often aimed at these types of applications, you will definitely want to make sure your policies that identify high and critical threat security vulnerabilities are included during the evaluation of these types of applications.

Unfortunately, especially as the number of applications in your business increases, identifying an application by name may not be helpful. To address this, **application categories** provide a way to quickly identify characteristics of an application.

Using specific text and color, an application category can help group particular applications with similar attributes. While an application category can ultimately be anything you want, and attached to any application, you will want to take a much more thought-out approach, similar to what is recommended for labels.

As we will see later, in order to maximize the benefits application categories can offer, you will want to take advantage of category matching between policies and applications. For now though, let's see how to create, edit, delete, and apply application categories.

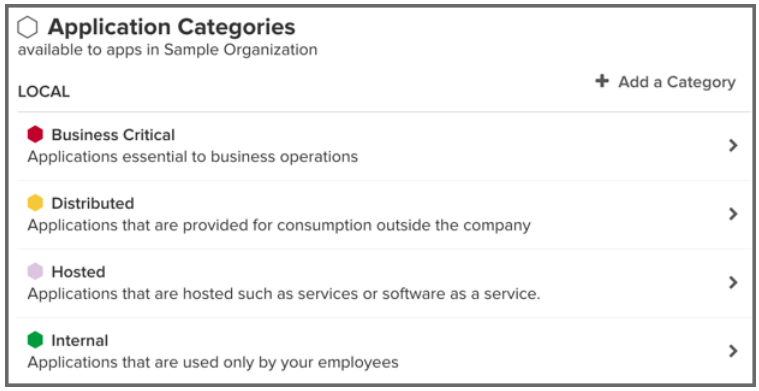


Figure 10.7: Example of Applied Application Categories

10.3.1 Creating Application Categories

Application categories are created, edited, and deleted at the organization level and then assigned individually to each application.

To create an application category:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select an organization.
3. In the *Application Categories* section, click *Add a Category.* The *New Application Category* dialog is displayed.
4. In the *New Application Category* dialog, set the following attributes:
    a. *Application Category Name* - Enter a name that is easily identified for it will be used to match an application to corresponding policies.
    b. *Short Description* - Enter a description that provides additional information about the category.
    c. *Color* - Select a desired color for the category.
5. Click the *Create* button to add the application category to the selected organization.

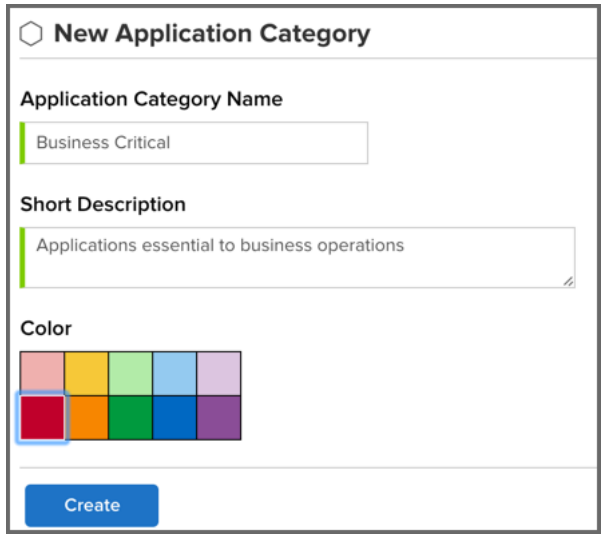


Figure 10.8: Using the Add a Category Button

10.3.2 Editing an Application Category

To edit an application category:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization in which the application category was created. The application category is displayed in the *Application Categories* section under the *Local* heading, and has a chevron in its row to indicate it's editable.
3. Click the application category you want to edit. The *Edit Application Category* dialog is displayed.
4. In the *Edit Application Category* dialog, you can change the following attributes:
    a. *Application Category Name* - Enter a different name.
    b. *Short Description* - Enter a description that provides additional information about the category.
    c. *Color* - Select a desired color for the category.
5. Click the *Update* button to save your changes to the application category.

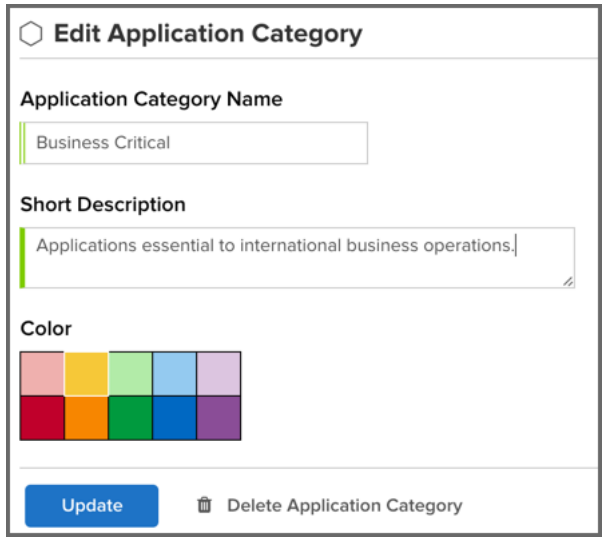


Figure 10.9: Editing an Application Category

10.3.3 Deleting an Application Category

To delete an application category:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization in which the application category was created. The application category is displayed in the *Application Categories* section under the *Local* heading, and has a chevron in its row to indicate it's editable.
3. Click the application category you want to delete. The *Edit Application Category* dialog is displayed as shown in the figure, Figure 10.9.
4. In the *Edit Application Category* dialog, click the *Delete Application Category* button. A *Delete Category* alert dialog is displayed. If there are applications assigned to the application category, they will be listed.

Warning
When you delete an application category, the action cannot be undone.

Note
You cannot delete an application category that's used in a policy to affect policy inheritance. You must first remove the application category from the policy, and then delete the application category.

5. In the *Delete Category* dialog, click *Continue* to delete the application category or *Cancel* to keep the application category.

10.3.4 Assigning an Application Category

In most cases, the people assigning application categories may be different from those creating them. It is important though to understand that while application categories are provided to identify characteristics of an application, a more important usage is to provide a way for policy managers to create specific policies that consider those application characteristics. For this reason, when assigning an application category, your application may be evaluated by a specific set of policies. This is a good thing, but it also makes the use of application categories an act that requires careful consideration.

To assign an application category to an application:

1. Log in to the IQ Server using a user account that's assigned to a role with Owner-level permissions for the application. The built-in Owner role has owner-level permissions by default.
2. Click the *Organization & Policies* icon on the IQ Server toolbar.
3. In the sidebar, select an application.
4. In the *Application Categories* section, click *Assign App Categories.* The *Assign Application Categories* page is displayed.
5. In the list of assigned application categories, select the application categories you want to assign to the selected application.
6. Click *Update* to save your selection(s).

Note
There must be at least one application category defined before you can assign any application categories. For more information, see Creating Application Categories earlier in this chapter.

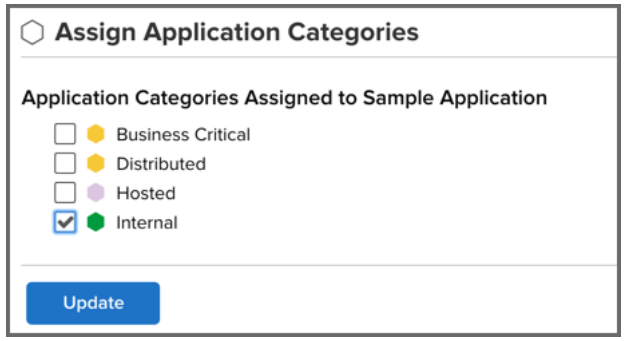


Figure 10.10: Assigning an Application Category

Tip
Once application categories are created and assigned, you can use them to apply policies to a subset of applications in an organization through inheritance. For more information about policy inheritance and application categories, see the Basic Policy Management chapter.

10.4 Manual Application Evaluation

When you evaluate an application, IQ Server generates a report that gives you a baseline of your application's health. Before you can initiate an evaluation, you need the following items:

- At least one organization and one application. For information on creating these entities, see the Organziation and Application Management chapter.
- *View IQ Elements* and *Evaluate Applications* permissions for the application you wish to evaluate. For details about setting permissions, see the Security Administration chapter.

To evaluate an application:

1. Log in to IQ Server as a user assigned to the Policy Administrator or Owner role for the application.
2. Click the *Organization & Policies* icon on the IQ Server toolbar.

3. Select an application in the sidebar.
4. Go to the *Actions* menu, and click *Evaluate Binary*.
5. In the *Evaluate a Binary* dialog, do the following:
   - Click the *Choose File* button and then navigate to the file you want evaluated.
   - Click to select a stage to associate with the evaluation.
   - Click *Yes* or *No* to specify whether notifications of policy violations will be sent as defined in the policy's configuration settings.
6. Click the *Upload* button to begin evaluating the selected application. An *Evaluation Status* message is displayed.
7. When the evaluation is complete, click the *View Report* button to open the Application Composition Report for your application. See the Application Composition Report chapter for more information on understanding the evaluation results.

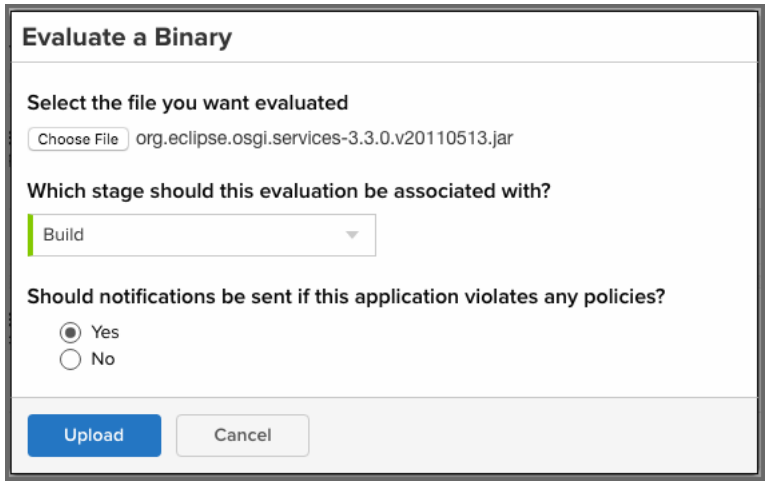


Figure 10.11: Evaluate a Binary

Chapter 11

The Dashboard

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle or Auditor license.

Depending on the number of managed applications, IQ Server can produce a substantial amount of data related to policy violations. Ideally, you want to focus on areas that represent the highest risk, and resolve those quickly. This is easy to do for a single application, but dealing with multiple applications is easier via an aggregate view.

The Dashboard provides the quickest way to review the overall health of the applications you manage. Whether you are looking to find the worst violations or the newest, the Dashboard should be the first place you review.

11.1 Using the Dashboard

When you log into the IQ Server the Dashboard is displayed by default. If you are in any other location of the IQ Server, simply click the Dashboard icon on the IQ Server toolbar.

Note
The Dashboard is only available via IQ Server, and only displays information for applications you are permitted to see. This requires that you, at a minimum, are assigned to the Developer role for at least one application.

The Dashboard is organized into two distinct areas:

- Filters
- Results

Each area is described in detail below.

11.2 Filters

Filters allow you to adjust the data that is displayed in the Dashboard.

The *Filter* menu is located on the left side of the Dashboard. To adjust any of the various filters, click the filter label to see the available options. A short summary of the current selection will be displayed next to each filter. Once you've adjusted the filters, click the *Apply* button to update the violations list.

Tip
After exiting the Dashboard and/or logging out, your most recently applied filters will persist for your account when you return. Clicking *Revert* will also revert all filters to their last applied state.

You can also save and load your favorite filters.

To save the currently applied filter selection:

1. Click the *Manage* button in the top right corner of the filter area to open the *Manage Filters* menu.
2. Click the *Save* button to open the *Save Filter* modal.

3. Enter a name in the *Filter Name* text box.
4. Click *Save* to save the new filter. It will appear in the list of saved filters in the *Manage Filters* menu.

To load a previously saved filter:

1. Click the *Manage* button to open the *Manage Filters* menu.
2. Click the filter name in the list of saved filters. When a saved filter is loaded, the name appears at the top of the *Filter* menu.

To delete a saved filter:

1. Click the *Manage* button to open the *Manage Filters* menu.
2. Click *Delete*.
3. Select one or more saved filters and then click *Delete*.
4. Click *Continue* to delete the selected filter(s).

Note
Saved filters are user-specific; they are not shared with others.

The available filters are described below.

Organizations
The organization filter allows you to view violations from one or more organizations, which includes violations from all applications attached to the organization(s).

Applications
The application filter allows you to select which applications you want displayed in the violation list.

Application Categories
The category filter allows you to isolate violations for applications assigned to particular categories.

Stages

Violations can occur in different stages, and some will be higher priority than others. Using this filter, you can show violations for a specific stage. The available stages include:

- all stages
- Build
- Stage Release
- Release
- Operate

Note

Access to stages is limited by your product license, and the filters will reflect this.

Policy Types

The policy type filter allows you to select which types of policies you want displayed in the violation list. Type is assigned automatically based on conditions included within the policy. The following rules are used to determine a policy's type:

Security
: if it has any security conditions, it is considered a Security policy.

License
: if it has any license conditions, it is considered a License policy.

Quality
: if it has any age or popularity conditions, it is considered a Quality policy.

Other
: if none of its conditions are of types mentioned above, it is considered to be of type *Other*.

Note

A policy can only ever be of one type. If a policy has conditions that meet more than one of the rules above, the order above dictates the type of policy. For example, if a policy has security and license conditions, it is considered to be of type Security.

Policy Threat Levels

The Policy Threat Level filter is a slider that allows you to select the threat level or a range of threat levels for policy violations.

Note

By default, the Policy Threat Level filter has already been set to only display policy violations with a threat greater than or equal to 2. As a result, the *Low* threat violations are not displayed in the violation list.

11.3 Results

The *Results* area displays risk information based on the state a policy was in at that time of the most recent evaluation, while information regarding the age is taken from the fist occurrence of the violation. If policy changes have been made, and a new evaluation has not been conducted, the changes will not be reflected in the currently displayed information.

The *Results* area provides several views of risk information, each of which is described below:

- Policy Violation Trends
- Violations
- Components
- Applications

11.3.1 Policy Violation Trends

At the top of the *Results* area, the *View* menu contains a *Calculate Trends* command. *Calculate Trends* opens a *Policy Violation Trends* dialog displaying policy violations trends that match your current filter.

Note
Calculating trends can take some time depending on the number and size of evaluations that match.

The purpose of *Policy Violation Trends* is to provide a quick, twelve-week look at how risk is entering your applications, and how you are handling that risk. The information is divided into four categories, with each category having four metrics over a twelve-week period.

Trend Categories

Pending
A policy violation that has been *Discovered*, but not yet *Fixed* or *Waived*, is *Pending*.

Tip
Reducing the number of pending violations is a critical task. Weekly deltas above the x-axis indicate there were more discovered violations than those fixed; green bars below the x-axis represent more violations were fixed than discovered.

Waived
This represents a count of policy violations that have been waived. This count is not included in Pending or Fixed, but is included in Discovered.

Note
For more information on waivers, see the Waivers section of the Application Composition Report chapter.

Fixed
A policy violation is *Fixed* when it no longer exists in any stage.

Note
When determining the *Fixed* state of a component, any filtered stages are not considered. That is, if you exclude a stage where a violation has occurred, the count for fixed may increase even though the violation is still present in the other stage.

Discovered
A policy violation is considered *Discovered* when it has been observed for the first time.

Policy Summary Metrics

Count
the total (all-time) count for the category.

AVG
the average age of violations in the category

90%
indicates 90 percent of violations have been in the category less than this time.

Delta
the count for the current week (week twelve), over the first week.

Weekly Deltas
the visual representation of each week's unique delta.

12 Week Trend
the trend over twelve weeks.

Tip
It is not uncommon to see discovered violations trend upwards steeply, especially in the early phases of your implementation, and then plateau as you start developing a better component consumption process. Using your mouse to hover over values in the graphs will display the individual values for each week.

11.3.2 Violations

Violations is the default view for the Dashboard. It displays the first one hundred, newest component violations found in your applications. The data in this view can also be adjusted using the filters, and is organized into a number of columns and rows. These have been described below.

Note
A violation is only considered new the first time it is discovered, even if it is found in different stages. For example, if a violation is found at the first of the month during an evaluation at the Build stage, and then again at the end of the month at the Release stage, only the occurrence at the build stage is considered new.

Threat
The assigned threat level of the violated policy.

Policy
The name of the policy violated.

Application
The name of the application the component violating the policy was found in.

Component
The identifying information for a component. For known components, all available coordinate information will be displayed, while unknown components will have the filename. Clicking on the component will display the Component Detail Page.

Age
Displays the age of the violation based on the most recent date it occurred.

Latest Report
Links to the latest available report.

11.3.3 Components

The *Components* View displays the first 100 highest risk components based on any filters that have been set and your level of access. Risk is represented in several ranges (Total, Critical, Severe, Moderate and Low).

To calculate the total risk for each component, the threat level of all policies the component has violated are added together. In other words, component risk is the sum of policy violation threat levels for the component. A similar calculation is done for each risk range.

Now, this may leave you wondering, "What about the duplication of violations across stages, or even in the same stage?"

Good question.

For all calculations, a violation is only counted once. When there are multiple instances of the same violation, only the most recent occurrence is counted, regardless of stage. Because of this, in cases where a policy has been changed in between evaluations, the violation from the latest evaluation will be included. This will be true, even if the change to the policy included threat level.

Now, let's take a look at each individual column, which has been described below.

Name
: The identifying information for a component. For known components, all available coordinate information will be displayed, while unknown components will have the filename. Clicking the component row will display the Component Detail Page.

Affected Apps
: The sum of applications that are affected by a policy violation due to this component.

Total Risk
: The sum of the threat level for each policy the component has violated. In cases where the same violation is found in multiple stages, only the newest violation is included in this total.

Critical
: The sum of the component's policy violations with a threat level of eight or higher.

Severe
: The sum of the component's policy violations with a threat level higher than three, but less than eight.

Moderate
: The sum of the component's policy violations with a threat level higher than one, but less than four.

Low
The sum of the component's policy violations with a threat level of one.

11.3.4 Applications

The *Applications* view displays the first 100 highest risk applications based on any filters that have been set, and your level of access.

Like a component, risk for an application is associated with the threat level of a policy. In the case of application risk, it is the sum of policy threat levels that correspond to unique policy violations for the components in an application.

This produces a total count by stage. The unique occurrences are then added together to create the total risk of an application. Put another way, application risk is the sum of all unique policy violation threat levels across all stages and policies the application is evaluated against.

Similar to the *By Component* view, for all calculations, a violation is only counted once. When there are multiple instances of the same violation, only the most recent violation is counted, regardless of stage. Because of this, in cases where a policy has been changed in between evaluations, only the violation from the most latest evaluation will be included. This will be true, even if the change to the policy included threat level.

In the *Applications* view, risk is broken down into six columns described below.

Tip
Click the stage name to see the most recent Application Composition Report for the corresponding application and stage.

For additional detail, take a look at the descriptions of each column below.

Application
The name of the application is displayed here. Click the expand icon (the small triangle icon), to display the results for each stage.

Total Risk
The sum of the threat levels for all policy violations in the application. In cases where the same violation is found in multiple stages, only one violation is included in this risk score.

Critical
The sum of policy violations in the application with a threat level of eight or higher.

Severe
The sum of policy violations in the application with a threat level higher than three, but less than eight.

Moderate
The sum of policy violations in the application with a threat level higher than one, but less than four.

Low
The sum of the component's policy violations with a threat level of one.

Tip
Remember, if your filters exclude data in any of these categories, this information will not be displayed.

11.4 Viewing Component Details

As components are used across various applications, and then evaluated, it is very likely some of those components will violate your policies. The Component Detail page presents the known coordinates for the component and then below this, all violations that have been found. These are organized by application. In addition, risk information for each component is provided.

You can click a component in either the Violations view or Components view to open the Component Details page.

Similar to previous views, separate columns display pertinent information related to the component and violations associated with each application it is used in. These are described in additional detail below.

Application
The name of application, preceded by its parent organization.

Share of Risk
The share of risk is displayed as a total for the application, as well as a breakdown for each violated policy.

For the Application
This is the percentage of risk for the displayed component in relation to a specific application. It is calculated by taking the sum of the threat levels for policies an application is evaluated against (and the component has violated), and then dividing by the sum of threat levels for all policies violated across all applications displayed.

For the Policy
This is the percentage of risk for a particular policy violation as it relates to the total risk for the component. It is calculated by taking the threat level of the violated policy, and dividing it by the sum of the threat levels for all violated policies for the displayed component and applications.

Risk
Risk represents the sum of the threat levels for the policies the component has violated.

Stages
Each stage is represented by a column. The amount of time that has passed since discovery of the component in violation of a policy will be displayed in the corresponding column. Abbreviations for time is as follows:

- min = minute
- h = hour
- d = day
- m = month
- y = year

If any actions were taken in the stage (i.e. warn or fail), an icon will be displayed. Only the stages which your IQ Server is licensed for will appear.

Tip
You can click a violation's time stamp to open the most recent Application Composition Report.

11.5 Exporting Results

At times, you may want to export the Dashboard results to a file for use in a spreadsheet or other business intelligence software. The Dashboard's *View* menu has an *Export Violations Data* command that lets you export the data displayed in the current view to a .CSV file on your local computer. The CSV file contains all components, violations, or applications that meet the Dashboard's filter criteria, unlike the *Results* view, which is limited to the first 100 matching items.

To export the Dashboard results:

1. Select the Dashboard view you want to export: *Violations*, *Components*, or *Applications*.
2. Verify all desired filters are applied.
3. From the *View* menu, click *Export Violations Data*. The results are downloaded automatically to your computer as a file with a .CSV extension.

Chapter 12

The Application Composition Report

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

The Application Composition Report represents the health of your application. Ultimately, it serves as a snapshot, a point-in-time report representing risk associated with component usage for a specific application. The report includes information on how the application complies with the policies your team, or business, has established. In many ways, it's the final connector between policies and the components of your application.

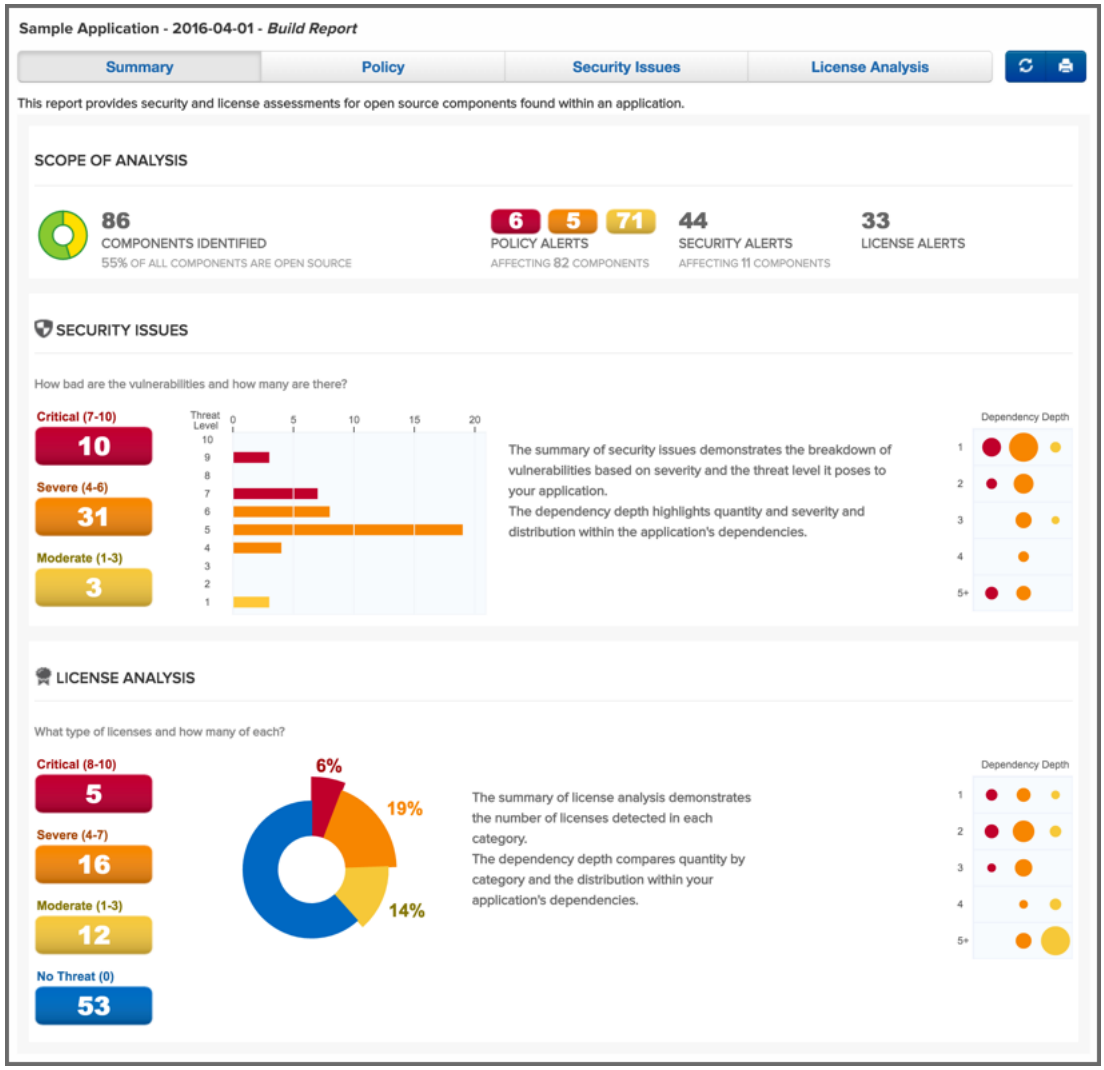


Figure 12.1: Summary Tab of the Application Composition Report

When looking at the report the first time, it can be daunting. If you see tons of red, you may quickly be dismayed. Or perhaps, you don't see enough red and are worried in a different way. These feelings aren't uncommon, and they reveal another important aspect of the Application Composition Report - it contains a lot of information.

More than just reporting the violations components in your application have triggered, it also provides a

way to improve policy management. These reports don't show false positives... ever. If there is a red ,severe policy violation that should really be much lower, communicate back with the team in charge of managing the policies. In fact, of all its uses, the ability to communicate findings to a wide audience is perhaps the most important task of this report.

In this section, we will provide an overview of the various areas of the report and therefore serve as an robust introduction.

For those of you that prefer bulleted lists, here's what we'll cover in this chapter:

- Accessing the Application Composition Report
- Overview of the four tabs and the component list
- Importance of component and violation counts
- Various policy, security, and license related data points
- Printing a bill of materials
- Overview of component information panel (CIP)

This chapter is meant to provide a detailed look at how to access the Application Composition Report, as well as what information is provided.

12.1 Accessing an Application Composition Report

You can access the Application Composition Report in your IQ Server in two ways:

Via the Reports Area

When you log into the IQ Server the Dashboard is displayed by default. Click the *Reports* icon . If multiple applications have been scanned, you will see all of them here.

Note

You will need to be a member of at least the developer group for the application you wish to see a report for.

You will notice, that there are several columns:

Application Name
Links to the **Application Management Area** for the specific application.

Build, Stage Release, and Release Violations
These three columns display the violation counts for the most recent evaluations. The counts are broken down by *Critical*, *Severe*, and *Moderate* with text indicating the time (e.g. *2 minutes ago*) of the most recent evaluation.

Contact
This is the contact for the corresponding application.

Organization
Links to the parent organization for the corresponding application.

To access the Application Composition Report, click the Violation Summary for the corresponding application and stage.

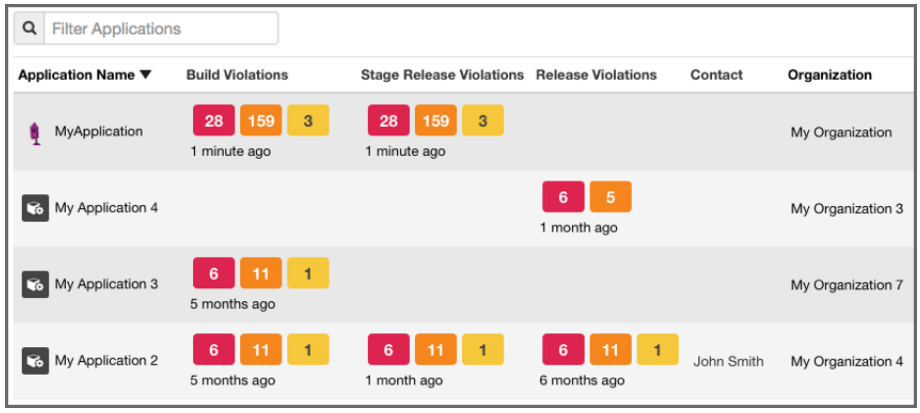


Figure 12.2: Reporting Area

Tip
By default this view will be sorted alphabetically by the application name. In addition to the filter, you can also click on the application or organization columns to sort alphabetically ascending/descending.

Via the Organization & Policies Area
The Organization & Policies area is where you manage policy for IQ Server. You can access an Application Composition Report for a selected application from the *Actions* menu as shown in the figure below. The most recent report is available for the different stage(s) at which the application has been evaluated:

- Build
- Stage Release
- Release

- Operate

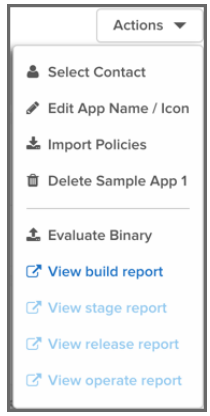


Figure 12.3: Actions Menu

Note
If you use the Nexus IQ CLI and don't specify a stage, it will default to *Build*. When your evaluation completes and the report is uploaded, it is accessible using the *View build report* action on the *Actions* menu.

Tip
Reports can also be accessed via enforcement point tools for CI and the repository manager. However, in each of the tools, they will connect to the IQ Server.

12.2 Reviewing a Report

When you look at the Application Composition Report for the first time, you will likely notice the four tabs:

- *Summary*
- *Policy*
- *Security Issues*
- *License Analysis*

Figure 12.4: The Four Tabs

These tabs represent the basic navigation for the report, and serve to divide information into specific sections. In a sense, the name of each tab represents the theme of the data that will be displayed.

The *Summary* tab displays a summary of violation, security, and license risk information for components in your application and provides a good first overview. The *Policy Violations* tab displays violation data for components in your application. The *Security* tab displays security related risk for components in your application. And the *License Analysis* tab displays license-related data for components in your application.

We'll cover each of these in a bit more detail below. However, it's important to first understand a little bit about what a report represents and the basic sets of data it contains.

In general, each report...

- Corresponds to a single, specific application, indicating the application name, date of the report, and the stage the scan took place in.
- Includes components found during a scan of the application, in most cases, including any dependencies.
- Records violations linked to an application's policies, or the policies inherited from the application's organization.
- Displays available security information for any components found matching components in the Central Repository.
- Displays available license information for any components found to exactly, or partially, match components in the Central Repository, as well as any data recorded manually (e.g. through the claiming process).
- Distinguishes between, external, proprietary and internally identified/claimed components.

Now that you know what forms the basis of the report, let's take a look at each tab individually.

12.2.1 Summary Tab

The *Summary* tab is always the first section of the report displayed. It is broken into three sections:

Scope of Analysis
This section shows counts, giving you an idea of the volume of components that were found during the scan. It also gives a breakdown of those that were identified, including a specific percentage that is represented by open source components. In addition to these numbers, you will also see:

- A count of components with policy violations, displayed by threat level. Only the most severe violation for each component is counted.
- The total number of security alerts found, and the number of affected components.
- The total number of license alerts. Each license alert corresponds to a single component.

Security Issues
The Security Issues section provides three visualizations. The first visualization displays the number of security issues by their particular Common Vulnerability Scoring System CVSS score, breaking the issues into three threat levels - *Critical*, *Severe* and *Moderate*.

Next to this raw count, the same numbers are represented in a bar graph to help distinguish the relative impact for each threat level.

Finally, a dependency depth chart shows where the security issues occur, relative to how many there are, indicated by the size of the circles, as well as what level of dependency they are found in.

Figure 12.5: Security Issues Summary

License Analysis

As with *Security*, the *License Analysis* section breaks the data into four threat level categories. However, these threat levels do not come from an external source, but rather the user-configurable license threat groups that are managed via the IQ Server.

There are four threat level categories:

- Critical (Copyleft)
- Severe (Non Standard)
- Moderate (Weak Copyleft)
- No Threat (Liberal)

These categories used in the report are static and not not configurable.

The first counts that are displayed represent the total number of licenses found in each threat level. Next to this list, a graph indicates percentage of licenses in each threat level category, compared to the total number of licenses found. Finally, a dependency depth chart indicates the volume of licenses found at each dependency level, as well as the color corresponding to the threat level.

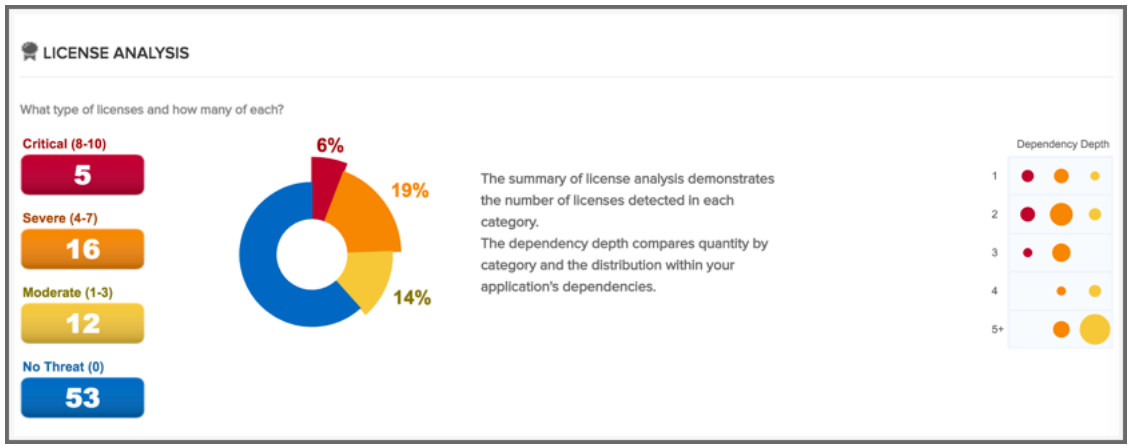


Figure 12.6: License Analysis Summary

12.2.2 Policy Violations Tab

The *Policy Violations* tab displays a list of all components found during the scan of the application. By default components are ordered by their worst policy violation. This is an important distinction, because a component may have more than one violation, and the threat level severity for those violations could vary. If you wish to see all violations there are two options, using the Violation Filter, or the Component Information Panel (CIP). In this chapter we'll discuss both options. However, below we have highlighted the available filters.

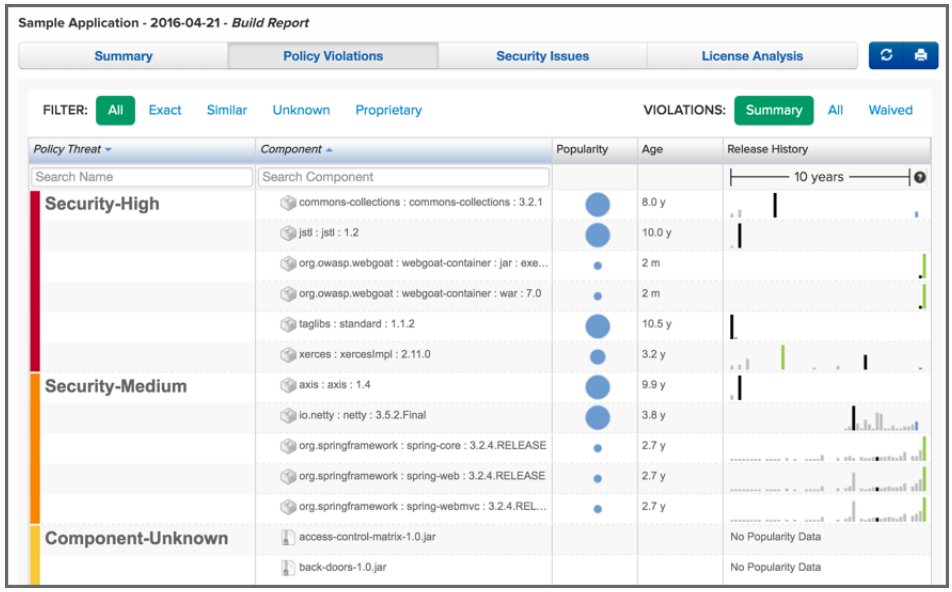


Figure 12.7: Policy Violations Tab

Filter

The filter lists five categories:

- All (*default*)
- Exact
- Similar
- Unknown
- Proprietary

In addition to the main set of filters, you can also filter by violations, including those that have been waived. The available options include:

- Summary (*default*)
- All
- Waived

Clicking on any of these will change the components in the list. We'll discuss each of these in further detail in the sections corresponding to component matching, claiming components, and waiving components sections.

Component List

The list of components, below the filter, displays the *Threat level* posed by the components. The *Policy Threat* column displays the name of the worst violated policy for the component and the severity using a colored bar. The *Component* column displays all available coordinate information for the component.

In addition the list displays the *Popularity* and the *Age* of the component in the Central Repository in separate columns. The *Release History* is displayed in a visualization that includes the most popular version, the most recent version, your version and any other available versions in a timeline.

By clicking on the column header, the list of components can be sorted. If you are looking for a specific policy, or component, you can use the search fields located at the top of each of those columns, directly below the header.

Clicking on a row for a component in list displays the Component Information Panel (CIP), which we will discuss in Section 12.4.

12.2.3 Security Issues Tab

The important thing to remember about the *Security Issues* tab is that information displayed there is related specifically to security vulnerabilities data that has been collected by Sonatype. This data however, is separate from policy violations, which are based on policies that you have created (or imported), and are displayed on the *Policy Violations* tab. That is, you could certainly have a situation where there is a security vulnerability, and no policy violation. Because of this, it is important to treat them independently.

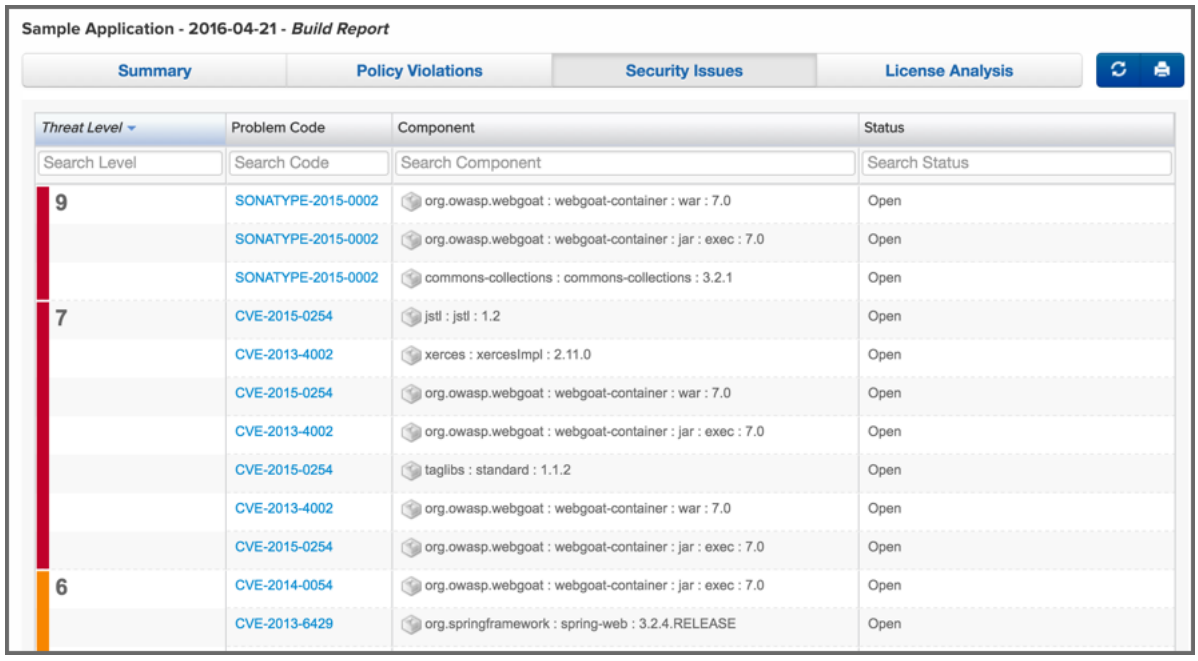


Figure 12.8: Security Issues Tab

The way components are displayed is actually quite different as well. In the *Security Issues* tab, only those components with a security vulnerability are displayed. The data provided for each component is broken into several columns:

- Threat Level
- Problem Code
- Component
- Status

By default the list of components with security vulnerabilities is organized by threat level. This helps you isolate the most critical issues you need to address. However, you may notice that components in this list are repeated. This is because a component may have more than one security vulnerability, and those vulnerabilities in fact may have different scores, thus different threat levels.

To sort the list, simply click the corresponding header. For example, if we wanted to sort by components,

finding a component with multiple vulnerabilities, we would simply click on the *Components* column. Additionally, you can search for a specific component by typing in the search field located directly below each header.

12.2.4 License Analysis Tab

The *License Analysis* tab displays all identified components found in the application scan and their license threat details. Unknown components are not displayed. Similar to the security issues, a license threat does not necessarily correlate to a policy, and as such should be treated independently.

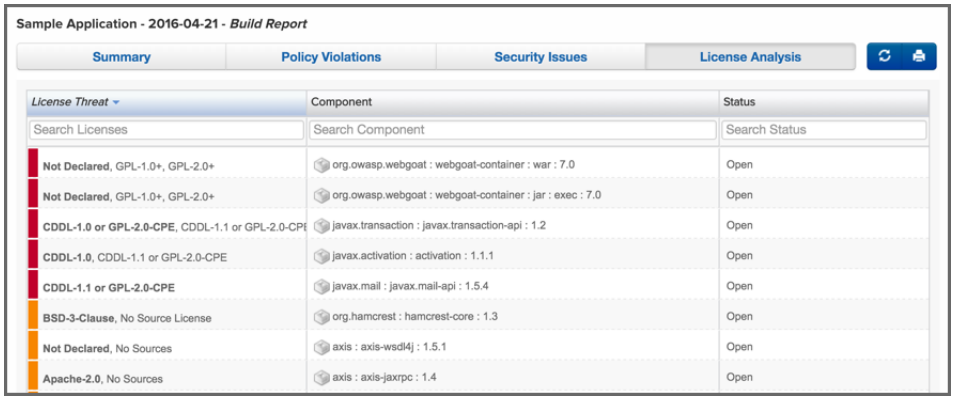


Figure 12.9: License Analysis Tab

For each component listed, the license related data is displayed. This data is based on information collected during a scan. By default, components are listed based on the threat of the corresponding *License Threat Group* that identified license is in. However, like the other tabs, clicking on a column in list will sort the components by that column. Additionally, specific components can be isolated using the search located below each header. The columns displayed include:

- License Threat
- Component
- Status

12.3 Printing and Reevaluating the Report

The top right corner of the report displays two buttons that give you access to refreshing the report as well as printing the report.

The refresh button on the left triggers a re-evaluation of the report. It will take the existing list of components in the report and reevaluate them against the your application policy.This comes in handy when you are making policy changes and want to see how that would affect the current data without having a to rerun a build.

The second icon on the right, the printer icon, allows you to create a PDF version of the report that is nearly identical to the HTML version. You can use this report for actual printing on paper, distribution to recipients without access to the IQ Server or simply for archival purposes. And of course it also works as a great bill of materials for your application.

While these are both small elements, they can prove to be very useful.

Figure 12.10: Application Composition Report Buttons For Printing and Reevaluation

12.4 The Component Information Panel (CIP)

In our previous sections, we briefly indicated that clicking on a specific component causes the Component Information Panel (CIP) to be displayed. We promised to discuss it further, and this section makes good on that promise.

The first thing you should notice, is that the CIP can be accessed for a component on the *Policy*, *Security Issues*, and *License Analysis* tabs. No matter which of these tabs you are on, simply click on the component, and the panel is displayed. Even better, the information displayed is the same, regardless of the tab in which you clicked on the component.

The CIP itself is divided into two areas. The top has a list of various sections, each providing more specific details and functionality related to the component. Below these sections, the panel will display information for the corresponding section. A brief description of each section is included below.

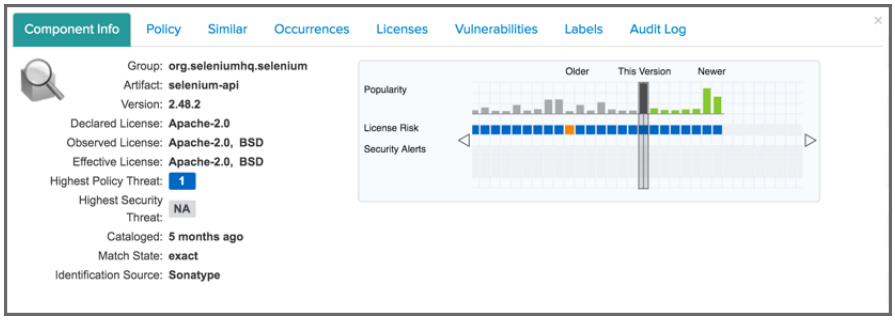


Figure 12.11: Component Information Panel CIP Example

COMPONENT INFO

Declared License
Any license that has been declared by the author.

Observed License
Any license(s) found during the scan of the component's source code.

Effective License
Either any licenses included in the Declared or Observed Group, or the overridden license.

Coordinates
The identifying information for a component. For known components, all available coordinate information will be displayed.

Highest Policy Threat
The highest threat level policy that has been violated, as well as the total number of violations.

Highest Security Threat
The highest threat level security issue and the total number of security issues.

Cataloged
The age of the component based on when it first was uploaded to the Central Repository.

Match State
How the component was matched (exact, similar, or unknown).

Identification Source
Whether a component is identified by Sonatype, or claimed during your own process.

Website
If available, an information icon providing a link to the project is displayed.

The graph itself is laid out like a grid, with each vertical piece representing a particular version. The selected version being identified by a vertical line. The information displayed in the graph includes:

Popularity
: The popularity for each version is shown as a bar graph. the larger the graph the more popular the version.

License Risk
: This will display the license risk based on the application that is selected, and the associated policy and/or license threat groups for that application. Use the application selector to change the application, and corresponding policies the component should be evaluated against.

Security Alerts
: For each version, the highest security threat will be displayed by color, with the highest shown as red, and no marker indicating no threat.

Policy

The *Policy* section displayed in Figure 12.12 has the details of any policy violations for the component. Here you can see the name of the policy that has been violated (and any action that was taken), the name of the constraint that has been violated, and the value that was found. While the *Policy/Action* and *Constraint* names are straight forward, the *Condition Value* may be a little confusing at first.

As we know from our other chapters, a condition is simply the *if* part of an *if/then* statement. *If* a certain condition value is found which is equivalent to a condition being met, *then* the policy will be violated. E.g. if we have a policy that has a condition such that if a security vulnerability is found, our Condition Value column would indicate, *Found x Security Vulnerabilities*. In the same regard, *Constraints* are simply multiple conditions joined together.

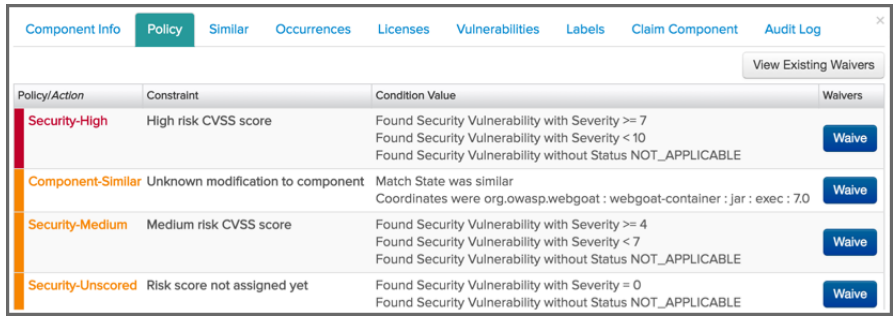


Figure 12.12: CIP, Policy Section

In addition to simply viewing the policy information details, a policy violation can also be waived in this section of the CIP using the *Waive* button.

Similar

You likely have already noticed the *Similar* filter that is available on the *Policy Violations* tab. These two are related, and both are a function of the matching algorithms used when evaluating and identifying components. We won't go into the details of matching at this time. So, for now, know that any components found to be similar to the selected component will be listed in the *Similar* section displayed in Figure 12.13. A similar component could for example be a component that a developer has built locally using the source code of an open source component with minor modifications or additions.

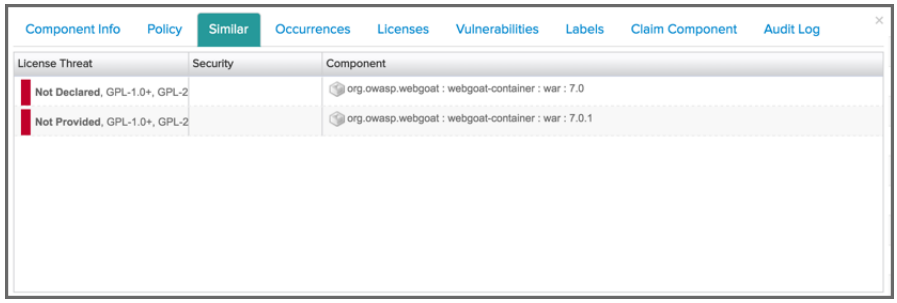


Figure 12.13: CIP, Similar Section

Occurrences

When a file is scanned, it has a filename and location where it was found. In some cases, it may have more than one filename and location. Either way, the path to the location(s), as well as the filename(s), of the component that was scanned is included in this section. In short, the *Occurrences* section displayed in Figure 12.14 lists the file names and locations where the component was encountered. This section can be especially useful to detect accidental shipping of duplicate components archives or a misconfiguration of your actual report creation target e.g. you might be scanning the deployment archive (e.g. a war file) as well as the build output folder used to create the archive.

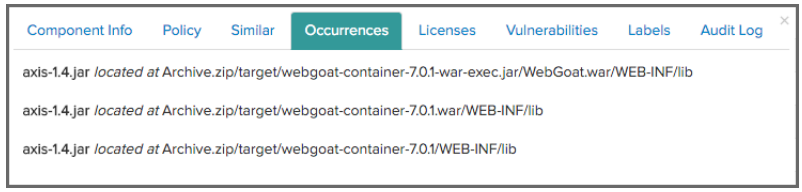


Figure 12.14: CIP, Occurrences Section

Licenses

The *Licenses* section displayed in Figure 12.15 is split into two areas. On the left, any licenses that were identified as declared by the author of the component, as well as any license found during the scan of the component source code are listed. On the right, is the license status area. This functionality directly correlates to the blue *Edit* button we mentioned in the *License Analysis* tab overview. It allows you to set the *Status* of the component license information.

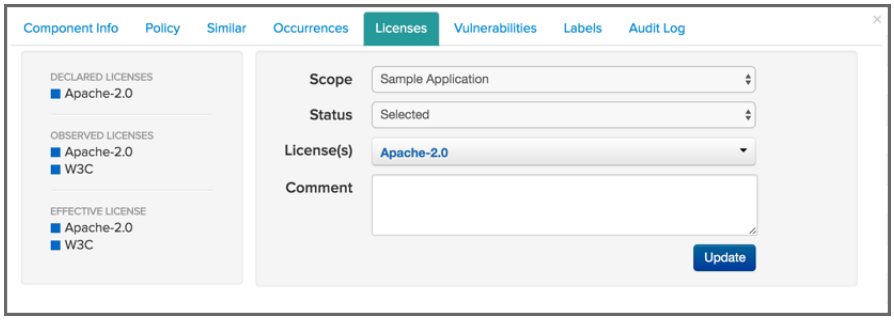


Figure 12.15: CIP, Licenses Section

Vulnerabilities

In much the same way as the *Licenses* section, *Vulnerabilities* displayed in Figure 12.16 is separated into two areas. On the left, all security vulnerabilities related to the component are displayed. Clicking the "i" info button on any of the vulnerability rows will show more details. On the right, the security vulnerability status area. This functionality, which we will discuss later, directly correlates to the blue *Edit* button we mentioned in the *Security Issues* tab.

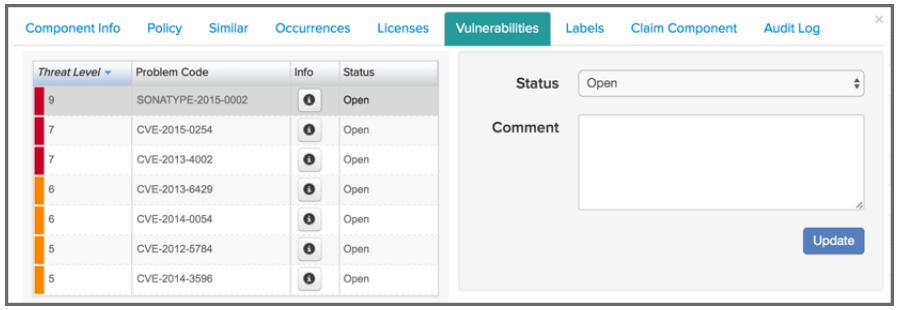


Figure 12.16: CIP, Vulnerabilities Section

Labels

Labels are discussed in more depth later in this chapter. However, the important item to note here, is that the assignment of labels to a component is done in this section of the CIP displayed in Figure 12.17.

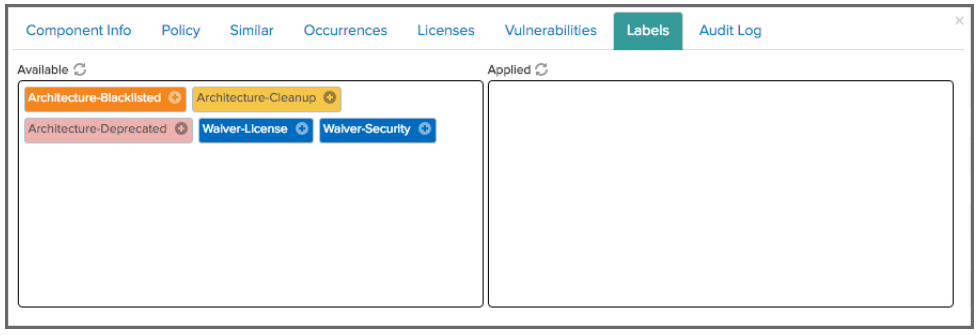


Figure 12.17: CIP, Labels Section

Claiming Components

The Claim Component section displayed in Figure 12.18 is only available for unknown or similar component matches. During a scan, some components are identified as unknown or similar. Since we realize that in many cases, you actually recognize these components, we provide this section to claim these components.

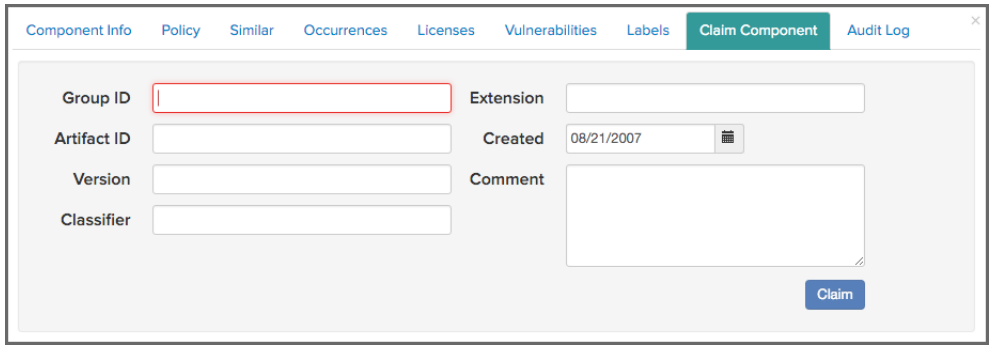


Figure 12.18: CIP, Claim Component

Audit Log

When changes are made to the status of a security vulnerability, or the status of a component's license within the scope of a particular application, that information is recorded in the *Audit Log* section of the CIP for that component displayed in Figure 12.19. As is the case for these last few sections, we'll discuss the *Audit Log* in greater detail along with our upcoming discussion of *Security Vulnerability* and *License Analysis* status.

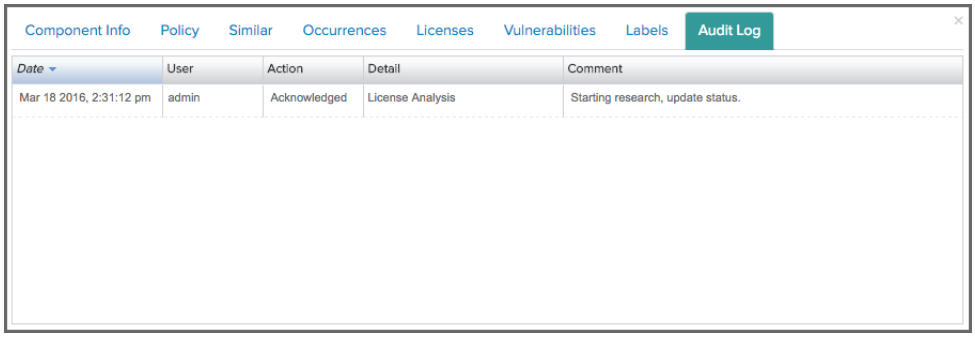


Figure 12.19: CIP, Audit

12.5 Resolving Security Issues

Evaluating your application for the first time, and seeing a huge number of critical security vulnerabilities indicated on the *Summary* tab can be a sobering experience, and in some ways it should be a little worrisome. More importantly though, it should create motivation for further investigation.

The key word there being investigation. That's because even though we've provided accurate data, you still need to have a process to review all available data, and then track your progress. It is not completely uncommon, and quite possible that a vulnerability doesn't apply to your application or, at the very least, isn't a concern given the particular application you are developing, and it's relative exposure points. Where do you start your investigation though?

12.5.1 Security Issues

The component list on the *Security Issues* tab (see example displayed in Figure 12.20) only shows components that have a security vulnerability. In addition, when a component has multiple security vulnerabilities, it is displayed multiple times.

There are a total of four columns: *Threat Level*, *Problem Code*, *Component*, and *Status*. Initially the list of vulnerabilities is ordered by the *Threat Level* column. However, you can sort the list by any other column by simply clicking on a header.

While the *Threat Level* and *Component* columns should be self-explanatory, the two other columns, *Problem Code* and *Status*, deserve a bit more explanation.

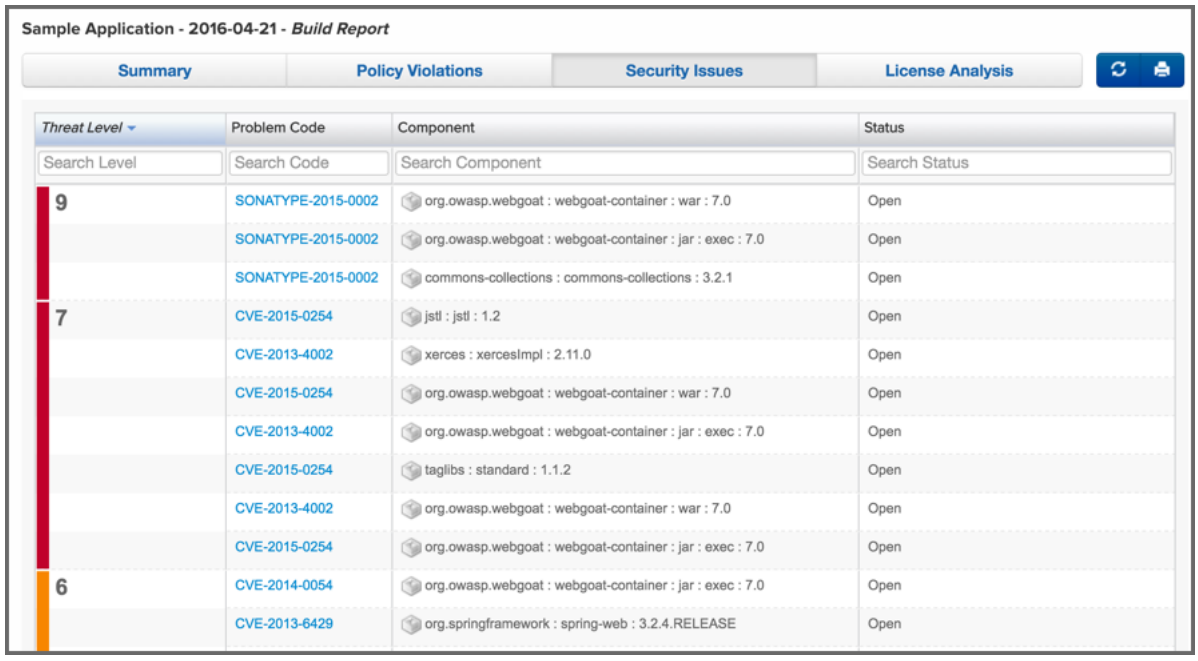


Figure 12.20: Security Issues Tab

Problem Code

The *Problem Code* column provides a link to available details for the security vulnerability on the CVE and OSVDB web sites. This information is provided via the CVE and OSVDB security information sites. These public security databases allow you to get quick information about the security issue and nature of the vulnerability.

Status

The *Status* column allows you to track the state and progress of research of the effect of a security vulnerability with respect to your application. We'll focus on the *Status* column in a bit more detail when we cover the CIP. A key point to remember, is that as long as the status is set to *Open*, *Acknowledged*, or *Confirmed*, the vulnerability will be included in the counts on the summary page. In addition, a policy with a condition related to the presence of a security vulnerability will be met, as long as the status is set to *Open*. That means it's very important to research these issues, so that only those affecting your application remain.

12.5.2 The Component Information Panel (CIP)

To access the CIP as displayed in Figure 12.21, simply click on a component row in the list. There are three sections you should use during your security vulnerability investigation - *Component Info*, *Vulnerabilities*, and *Audit Log*.

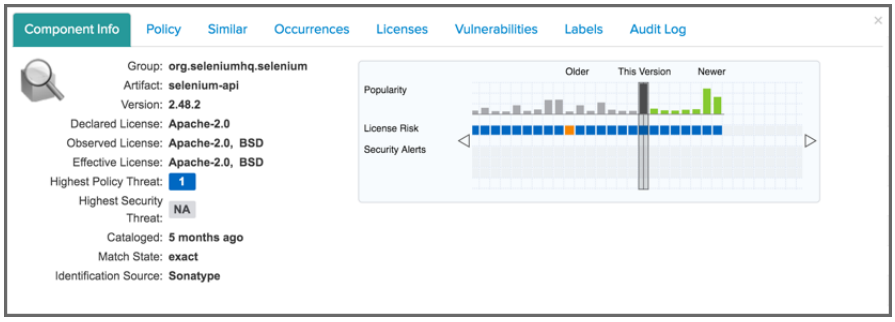


Figure 12.21: Component Information Panel (CIP)

Component Info

One of the first things you should notice in the *Component Info* section, is the *Highest Security Threat*. This field, located on the left side of the panel, displays the highest threat and the threat value (on a scale of 1-9). In addition, it will display the total number of security issues for that particular component.

Component Graph

Next, you should take a close look at the graph to the right of the panel. On the graph, locate the *Security Alerts* field, taking into consideration the other fields as well. This graph will display security vulnerabilities by version, with the current version identified as *This Version*. In some cases there are clear points where security issues have been resolved, as can be seen above. Often this tends to coincide with more popular version, although, that is not necessarily always the case.

12.5.3 Editing Vulnerability Status

After clicking on a component row to display the CIP, click the *Edit Vulnerabilities* section.

Here, the left side will display all security vulnerabilities. Depending on how many, this list may scroll. The list is then organized into three columns:

Threat Level
: Indicates the threat assigned to the security vulnerability and is determined based on the source. This is **not** associated to any policy threat level.

Problem Code
: This is the unique identifier of the security issue as assigned by the source (e.g. CVE-2000-5518). It will change depending on the source of the data.

Information
: Sonatype provides information from public sources, as well as information from our own research team. Clicking on the icon in the corresponding row will display additional details provided about the issue.

Figure 12.22: Security Information Modal

Figure 12.23: Security Information Modal Additional Details

Status
: The status of the security issue as assigned by the drop down to the right. See below for information on changing this status.

To the right of the list of security vulnerabilities is the status drop down and a comments section. To change the status simply select one from the drop down, select the vulnerabilities the status will apply to, enter any associated comments, and finally, click the *Update* button. It is important to mention the status can be changed to any status at any time.

There are four statuses available:

Open
The default status, represents no research being done.

Acknowledged
Represents that the security vulnerability is under review.

Not Applicable
Indicates that research was conducted, and the particular vulnerability does not affect the application.

Confirmed
Demonstrates research was conducted, and it has been determined the security vulnerability is valid and applicable.

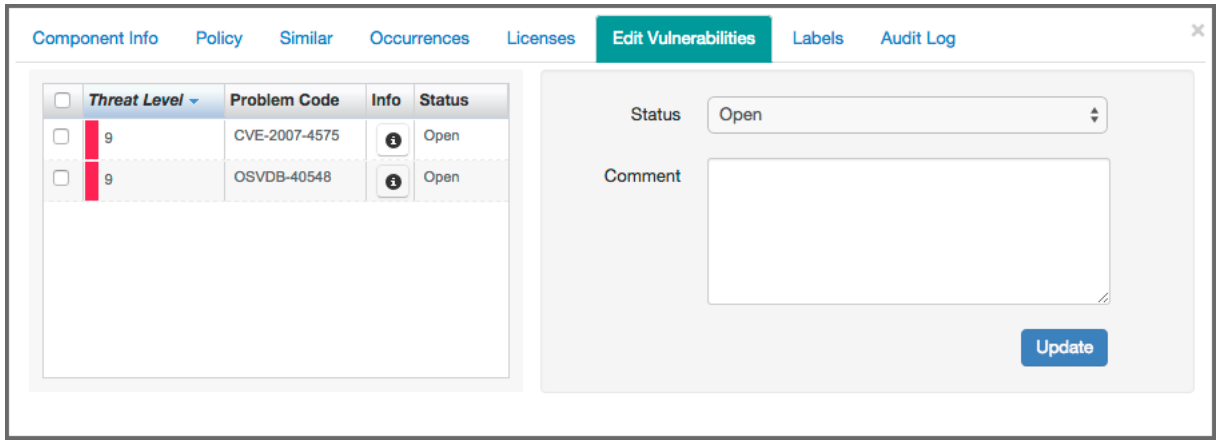


Figure 12.24: Editing Vulnerabilities

12.5.4 Matching to Violations

In some cases, just because there is a security vulnerability, that does not necessarily mean there is a corresponding policy violation. For this reason, it's important to refer back to your *Policy Violations* tab as well. If you are finding that critical security issues you are troubleshooting do not show up as a policy violation as well, you may need to refine your policy so that future security issue trigger a policy violation and thus ensure that they get your attention.

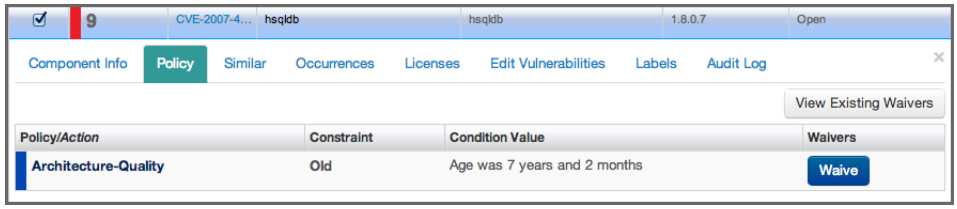


Figure 12.25: Example of Component with Security Issue, but No Policy Violation

12.6 License Analysis Tab

In some cases, the licenses of a component is the last thing a development team will think about. This could simply be due to a misunderstanding of *open source*, or a situation where it's nearly impossible to do the exhaustive research needed to determine the license for a given component, especially dependencies.

Even if you haven't built policies around licenses the *License Analysis* tab provides license information about every component found in during a scan of your application.

This license information is provided via data collected from the Central Repository, as well as research conducted by Sonatype. In addition to the license information for each component, we'll also assess a threat of each license, based on a set of default *License Threat Groups*. As with *Security Issues*, the best place to start is with the component list in the *License Analysis* tab, and then move into looking at additional details for individual components, making any license status changes as you see fit.

Figure 12.26: License Analysis Tab

12.6.1 License Threat Group

License threat groups are based on what is configured for each organization or application. Additional information can be found in the License Threat Groups section of the Policy Management chapter.

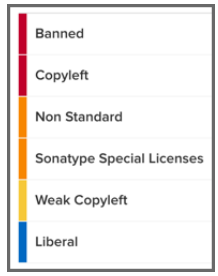


Figure 12.27: The Default License Threat Groups

Tip
How you manage your license threat groups directly impacts how threat is translated in the reports.

12.6.2 License Analysis

The component list on the *License Analysis* tab is more similar to the list on the *Policy Violations* tab, because it is a list of all components, not just those that have a license issue.

The list itself includes columns for *License Threat*, *Component*, and *Status* of the license issue. Clicking on the column provides sorting, while specific items can be searched using the field just below the column heading.

License Threat
The list of components is ordered by license threat which is based on the threats assigned to the license threat groups. Though a single component may actually have several licenses, license threat will only show the highest threat. This threat, as we mentioned earlier, is based on four default categories, which correspond to four default license threat groups of the same name.

- Critical
- Severe

- Moderate
- No Threat

Status

License status, like status for security vulnerabilities, allows you to track the process for license related research. In addition it provides a way to override a license in situation where you believe the license to be incorrect, or there is an option to choose a specific license. We'll discuss that process a little bit further down.

12.6.3 The Component Information Panel (CIP)

To access the CIP for a component on the *License Analysis* tab, simply click on the component row. It will expand providing details in a number of sections. You will likely notice this looks the same as other CIP panels when clicking on other tabs of the Application Composition Report, and you would be correct. There is nothing additional provided by accessing the CIP via the *License Analysis* tab of the report. However, for this section, we want to focus on the license related information in the Component Info section, as well as the entire *Edit Licenses* and *Audit* sections.

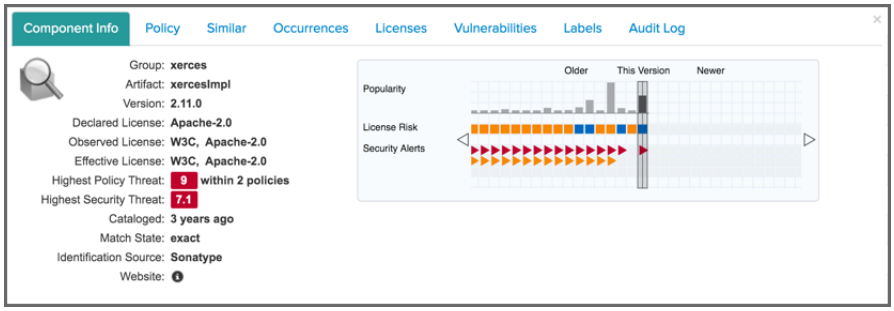


Figure 12.28: Component Information Panel (CIP)

Component Info

Again, the information contained here would be the same, whether or not you clicked on the component in the *License Analysis* tab. However, this gives us the context to talk about the License related fields in this section.

License Identification Types

On the left side of the Component Info section, you should pay attention to three fields, which are described below.

Declared License
these are the licenses that the developer of the component has identified.

Observed License
these are the licenses that have been observed during Sonatype's research.

Effective License
The effective license displays license information based on one of two scenarios. In cases where multiple licenses are found, including any that are observed, these will all be included as effective. If a license is selected, or overridden, then that license will be considered effective, and listed here.

License Identification Values
In cases where there is no declared and/or observed licenses, a message will be displayed. There are several options, each with specific meaning:

No Source License
sources were provided, but there was no license data found.

No Sources
indicates we have no sources for the component.

Not Declared
indicates nothing was declared by the author/developer.

Not Provided
will appear when the license is actually null, and is unique to claimed components, but might also happen while new components are being processed by Sonatype.

Not Supported
indicates Sonatype or the target ecosystem does not currently support automated license collection for this component format.

Component Graph
The graph itself is laid out like a grid, with each vertical piece representing a particular version. The selected version being identified by a vertical line.

While the information displayed in the graph includes popularity, and security information, right now, just take a look at License Risk. This will display the license risk based on the application that is selected, and the associated policy and/or license threat groups for that application. Use the application selector to change the application, and corresponding policies the component should be evaluated against.

12.6.4 Editing License Status and Information

Editing a license can be used for different purposes. One addresses the workflow of your research into a license related issue, while the other allows you to completely override a license all together. We'll cover all this below, but first let's take a look at the information displayed.

After clicking on a component in the list, and then the *Licenses* section of the CIP, the left side of the CIP displays the license(s) declared by the developer of the component, those that have been observed, and what is considered effective (a combination of the previous two). That is, unless they have been manually overridden or a specific license has been selected.

Next to each of these licenses is a box, displaying the severity of the license. This list can get long, so you may have to scroll to see all the licenses. Then, to the right of the license list, there are four drop down lists.

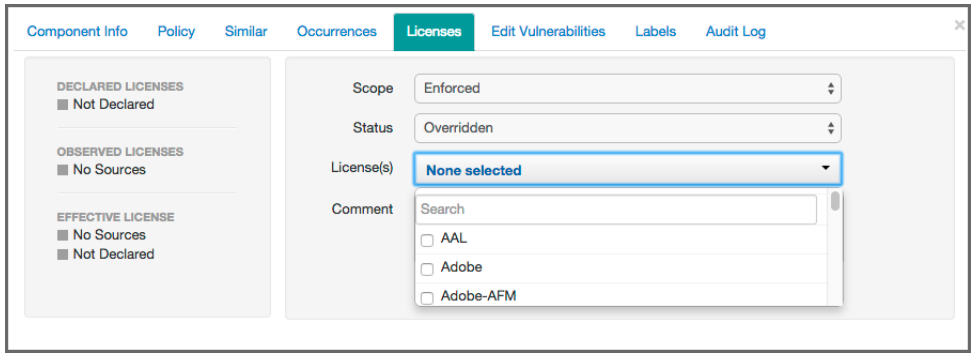


Figure 12.29: Editing License Using the Select Option

Scope

Scope allows you to apply the license status to this component by choosing application or to all components attached to the current application's organization by choosing organization.

Status

As we mentioned previously, *Status* provides a way to track your research, override a license, or select from an option. The available options are included below.

Open

This is default status, and will be included in the count of license issues.

Acknowledged

Acknowledged indicates the issue is being researched, and will still be included in the count of license issues.

Overridden:

This status will allow you to select one or more licenses from the *License(s)* dropdown (located just below the *Status* dropdown). This will override any licenses that have been declared or observed.

Selected

In cases where there are multiple licenses, this option will populate the *License(s)* dropdown

with any licenses found in the component, declared or observed. Multiple licenses can be selected.

Confirmed

Confirmed simply indicates that the license(s) found are indeed correct, and will be included in any count of license issues.

License(s)

The *License(s)* drop down only displays given that a status of selected or overridden has been chosen. Given that it will present either a list of all licenses (if override is chosen) or only the declared and observed licenses (if selected is chosen). The license that is chosen will be displayed in the *Effective License* field in the *Component Info* section of the CIP. In addition, any overridden/selected license will be indicated with a label of same name, next to the license in this field.

Comment

A comment is not required, but is a good element to include whenever you are making changes to the *License Status*. This is because it provides a way to understand, as well as audit, the decisions made to change a license status. This comment will be included with the record in the *Audit Log* section of the CIP.

Once you have made all your selections, and entered any necessary comments, click the *Update* button to save the *License Status* change.

12.7 Component Identification

One of the most important things you can do with regards to understanding the components in your application, is to identify them. What remains unidentified is of obvious concern.

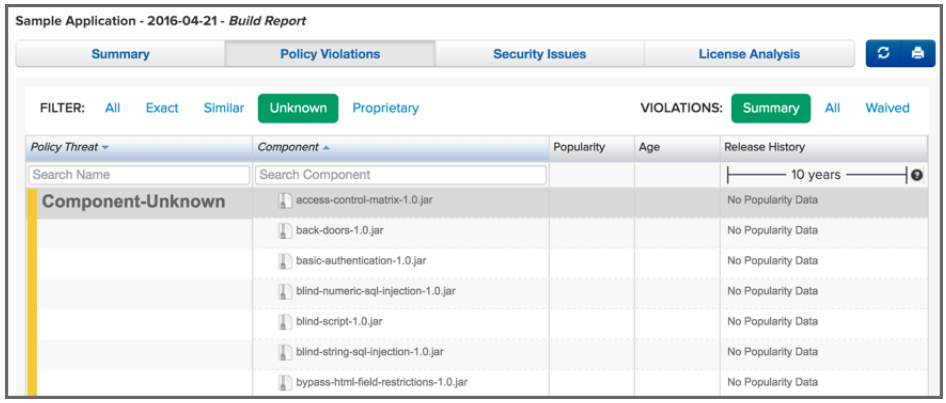


Figure 12.30: Unknown Component

Components can be identified in a number of ways, including:

- Extensive matching via various, proprietary algorithms
- Claiming components
- Establishing proprietary components

In this section, we'll describe all of these in detail, within the context of identifying components using the Application Composition Report, as well as offer our suggestion for best practices.

12.7.1 Matching Components

When an evaluation is performed, hashes of the components in your application are created. This in many ways is like a fingerprint, which is unique to a component. That fingerprint (hash), is compared back to components known to the IQ Server, which will provide all the available component info. This includes: usage statistics, security vulnerability, and license information.

All of this information can be used as parameters in your policy, which translates to more understanding of the component usage in your organization. That data however, can only be linked based on a matching of hashes, which can be exact or similar, and in some cases, unknown. We discuss these three match types below.

Figure 12.31: Filter and Matching Options

Exact
: An exact match means that a one-to-one link was found between a component hash in your application, and a component known to the IQ Server. This is the best case scenario with regard to component identification, and most components should fit in this category.

Similar
: A similar match is found using various, proprietary matching algorithms. In a way it's a "best guess" to match a component that you have in your application with a similar one known to the IQ Server. In some cases, multiple matches may be found, and this is where the *Similar* section of the CIP is important. While the most likely match is used to display any information about a similar

matched component, you can see all other matches in this section of the Application Composition Report. An example is displayed in Figure 12.13.

Unknown
There are instances where not even a similar component match can be determined. This should be considered a serious situation, at least one that needs to be investigated. This could be a case of a component being recompiled and modified so that a match is no longer possible.

However, there is a chance that component is something malicious introduced into the application. Either way, an unknown component should prompt an investigation. Of course, if during your investigation, you are able to identify the component, you can claim that component, via the *Claim Components* section, which we will walk you through in more detail a little bit later. An example is displayed in Figure 12.30.

Note
Unknown components will not be displayed in the *License* tab until they have been claimed.

In addition to the main filters above, you can also control whether all violations for each component will be displayed. By default the summary of violations is shown. This means that only the worst violation for a component will be shown, and the component will only appear once in the list. Choosing All or Waived, will show all violations (including those waived), or only the waived violations, respectively.

Note
Changing the Violations filter can result in the components being displayed in the component list more than once.

12.7.2 Managing Proprietary Components

Proprietary components are unique to your organization. In many cases, these are developed by your organization and distributed among the applications you create.

In the Application Composition Report, you can view proprietary components on the *Policy Violations* tab as shown below. While *Proprietary* is listed under *Filter*, the process for matching exact, similar, and unknown components is separate from identifying a component as proprietary. You can have proprietary components that are an exact match, similar match, or unknown match. If you're trying to determine which of your components are proprietary, a good place to start is to review unknown components.

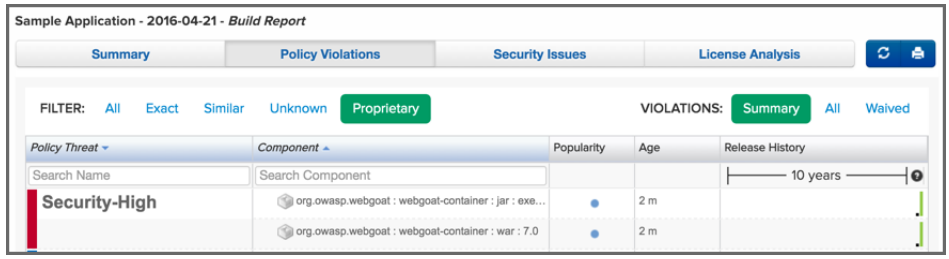


Figure 12.32: Proprietary Component

IQ Server uses *proprietary component matchers* to identify proprietary components when applications are evaluated. Proprietary component matchers are configured in the *Organization & Policies* area, but you can also add them from within the Application Composition Report. This is especially useful when the report contains unknown components that you know are proprietary.

Note

You need to be assigned to a role with "Edit Proprietary Components" permission in order to add proprietary component matchers. The built-in Policy Administrator and Owner roles have this permission.

To add proprietary component matchers:

1. Click to select a component.
2. Click the *Add Proprietary Component Matchers* button.
3. In the dialog, click the occurrences of the component that are proprietary.
4. Optionally, add a regular expression to identify proprietary components.
5. Click *Add*.
6. Re-evaluate the binary application or repository to see the component identified as proprietary.

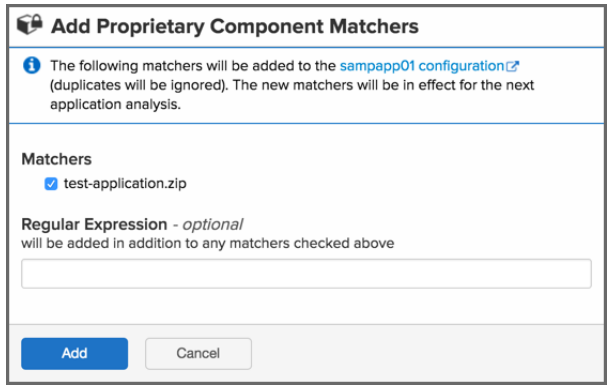


Figure 12.33: Add Proprietary Component Matchers

Tip
Existing reports are unaffected by additions to the *Proprietary Component Matchers* list; only after the next evaluation will you see the newly added components identified as proprietary.

For more information about the *Proprietary Component Matchers*, see Proprietary Component Configuration in the Basic Policy Management chapter.

12.7.3 Claiming a Component

When a component is similar or unknown, yet you are certain the component is recognized by your organization, you can prevent that component from being identified as similar or unknown in future reports. In other words, you can claim the component as your own.

Once claimed, that component will be known to the IQ Server. It will no longer be treated it as *similar* or *unknown*, and instead result in an *exact* match.

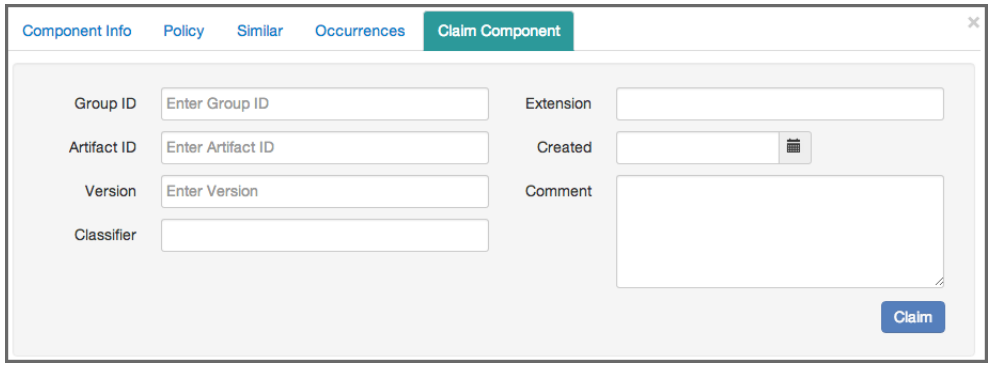


Figure 12.34: Claim a Component

1. Access an Application Composition Report.
2. Click the *Policy Violations* tab, and then click the *Unknown* or *Similar* component filter.
3. Click the row of component you wish to claim in the list - the Component Information Panel is displayed.
4. Click on the *Claim Component* section of the CIP .
5. Enter values for the coordinates of the component.
6. As an option, enter the coordinates classifier, the *Created Date*, and/or a *Comment*. The created date is initialized with the date of the youngest entry in the component to be claimed.
7. Click the *Claim* button, to officially stake your claim for the component.

On review of the existing report, as well as those in the future, there is now an indicator that information about the component has been edited. When hovered over, a tooltip is displayed identifying that the component has been claimed.

We refer to this as the edited component tick mark (a small red triangle) on all future scans for this application, as well as any application with a valid Application ID on the IQ Server.

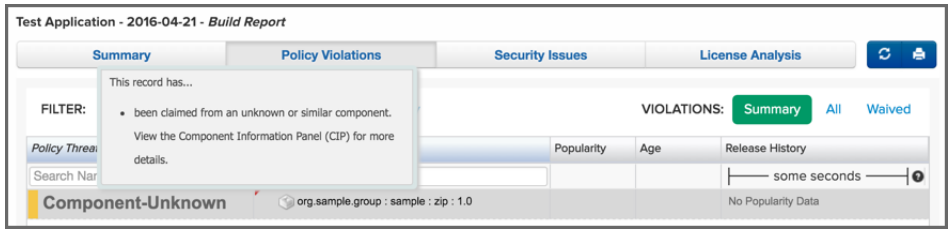


Figure 12.35: Claimed Component Indicator

In addition, the *Component Info* section for the claimed component will now have two new fields, one indicating the *Identification Source* is *Manual*, and the other, *Identification Comment* will include any comments that were entered. While any policy violations will be displayed, the component graph will not.

Finally, if you have made a mistake and wish to revoke the claim on the component or make an edit, click on the Claim Component tab. Then, use the *Revoke* or *Update* buttons respectively.

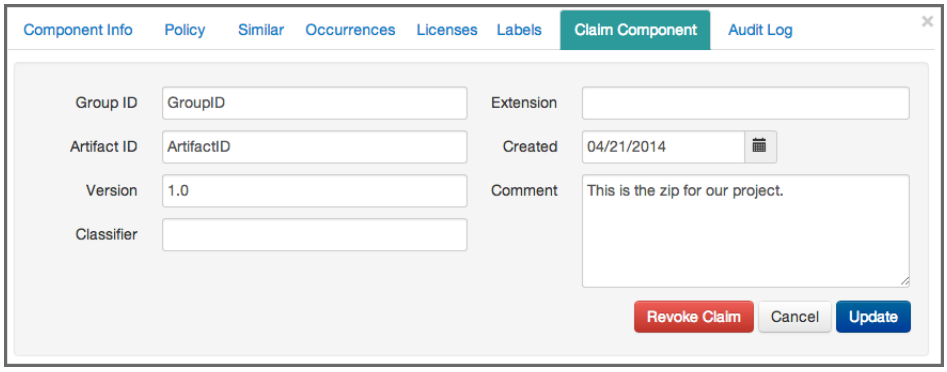


Figure 12.36: Update or Revoke Claimed Component Indicator

Tip
Use the cancel button to undo any changes you made but haven't saved.

12.8 Component Label Overview

Component labels are one of the more powerful features, though they operate similar to other label or tagging systems you have likely used. Basically you create a number of labels or tags as a set of available values and then assign them.

For example in a photo collection you could have labels for the content like *sunset*, *mountain*, *ocean*, *waterfall* and so on. An individual photo could then have multiple of these labels assigned. In a similar way you can use component labels to identify a particular type of component.

A common approach is to use component labels to identify an *approved* component or a component that *needs research*. Component labels can be anything you desire and can have a number of contexts as well. Some component labels might be architecture related, while others are related to legal and security properties and yet others are simply signaling ownership by a specific team. The flexibility of the component label system allows you to design your own use cases and implement them.

Component label creation and management is performed in the IQ Server **Organization & Policies Management** area. Assigning those labels to a particular component is a function of the Application Composition Report.

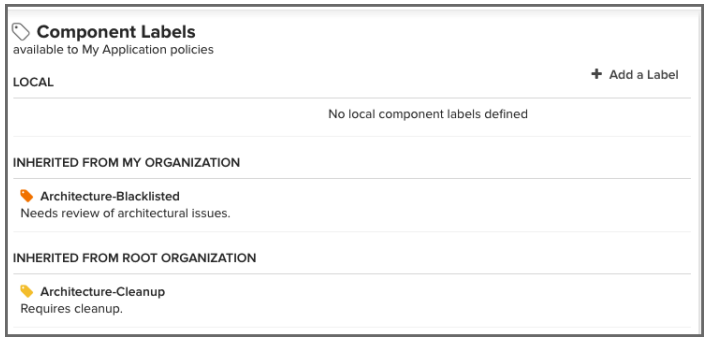


Figure 12.37: Component labels at the IQ Server Level

12.8.1 Where do component labels begin?

Initially, component label assignment might seem like a task that deserves less process. It's actually the opposite.

The component label process actually starts at the IQ Server, where each organization, in many cases application, has a specific set of component labels.

Creating component labels at this level means that they are available to users of the Application Composition Report. So, before you label something, a number of things need to be considered, which might mean you need to go back to component label and policy management. Let's take a look at some questions that we can use to form a baseline for what component labels we need and/or should have.

Do we have a process defining component labels, as well as how and when they should be used? Every component label should have a reason for using it. If it doesn't you don't need it. That's because people will naturally infer meaning from the label. For example, if you have a *needs review* component label, you should have a process for reviewing components with that label. It shouldn't merely be a tag. You should also consider adding component labels such as *reviewed*, so someone can tell if something was reviewed. The possibilities are endless, but this may mean you need to rethink the component labels you have, or should create.

Should this component label apply to only my applications, or all applications for this organization? We like to refer to this as the scope of a component label. A good way to look at scope is the concept of macro and micro. At the macro level we have an organization that may have thousands of applications. If I assign this label to the component at the macro level (organization) it means the label will be seen by all applications under that organization. A component label at the root organization will have an even larger scope. That's a pretty sweeping change, and it could be the right thing to do. However, you might actually be better served by considering the impact at the micro level. In this case, maybe this is a component label that should only be applied to a particular application.

Note

Only component labels that have been created at the root organization, can be assigned to the scope of root organization or organization level. Only components created at the organization can be assigned to the scope of organization.

If I assign this component label, is it part of a policy, and does it escape certain violations? Conditions in a policy can include values for component labels. In many cases, this is best used as a way to prevent a certain component from violating the condition. For example, I could have a policy that requires a component to be no older than three years. However, a safe, and commonly used component is four years old. If I have a process built around reviewing a case like this, where an exception would be valid, I could first place component labels to identify the component to be reviewed for an exception, and then another component label once that exception is approved (or disapproved). In this scenario, if I have built policies correctly, including allowing them to flow through certain stages, even with a violation, development can continue. This of course should occur simultaneously to a review/exception process. It's important to consider scenarios like this when creating component labels, as well as policy.

There are more questions regarding component labels that should be asked as well, but many of these you will discover as you develop your own processes. The key is that component labels can be deceivingly simple, given their implementation in other systems. Now that you have the word of warning, let's take a look at how to assign a label.

12.8.2 Assigning a Label

When assigning a label, you will only have access to those component labels created specifically for the application, or for that application's organization, or the root organization. Given this, if you don't see a label you need, speak with whomever is responsible for managing the component labels for your application and parent organizations.

1. Access the Application Composition Report for your application.
2. Click on the *Policy Violations* tab.
3. Click a component you wish to assign a label to. The Component Information Panel (CIP) displays.
4. Click the *Labels* option from the CIP menu. Two boxes will be displayed:
    a. The *Applied* box on the left represents labels that have been assigned to the component already.
    b. The *Available* box on the right displays all labels.
5. Clicking on the button on the right side of a label will move to the opposite side. Hovering over a label displays the description.
6. Click on the + button on the right side of a label in the *Available* list to assign the label to the component.
7. Click on the - button on the right side of a label in the *Applied* list to remove the label from the component.

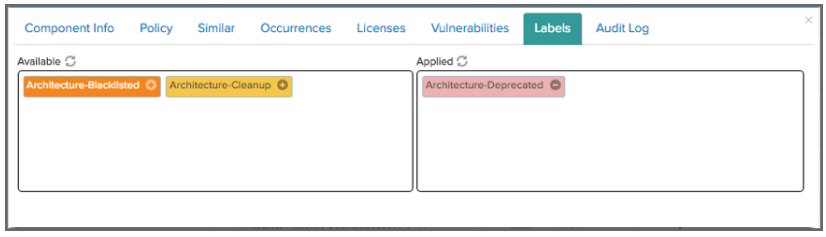


Figure 12.38: Assigning a Label

If the label was created at the organization level, you will be presented with two options:

- Assign the label to the current component in the current application
- Assign the label to the current component within the organization, so that all applications within the organization gain access to the label assigned.

If the label was created at the root organization level, you will be presented with three options:

- Assign the label to the current component in the current application
- Assign the label to the current component within the organization, so that all applications within the organization gain access to the label assigned.
- Assign the label to the current component within the root organization, so that all applications within all organizations gain access to the label assigned.

12.9 Waivers

If you look at policy violations as a pain point preempting the flow of work, you are likely going about your evaluations and policy in the wrong way. In fact, if you saw the title of this section, hoping to find a way past policy violations, there may be a couple issues.

First, your policies should be designed to encourage workflow and communication. If development is being stopped regularly, you might want to revisit your policies, refining them so they present the possibilities for making better choices, not simply halting work altogether.

Second, and perhaps most importantly, there are not false positives when it comes to policy violations. If you are looking for ways just to get past a violation, you've circumvented the goal of policy creation. Why not simply not make it a policy? Better yet, this might be a problem with policy, or the perception of what should and should not be in your application.

So, excluding those possibilities, and working with the idea that you are here to find a way to accommodate the exceptions you may run across, waivers can help.

12.9.1 A Use Case for Waivers

Let's say, you have a component that's violating a policy, which has been created in an organization that houses your application. It's a great policy, in fact, one of many great policies. Unfortunately, one of your applications has a component that is violating a policy.

The problem is, that this policy just doesn't accurately line up to the implementation of this particular application. Your application has a security vulnerability that can be exploited when it connects to the internet. It's a pretty severe vulnerability, but benign given that your application is internal, and doesn't even have the ability to connect to the Internet.

What should you do?

You could...

- Change the *Security Status* to *Not Applicable*.
- Adjust the policy to be less stringent.
- Use labels in a conditions, providing an escape for exceptions.
- ... or, you could add a waiver.

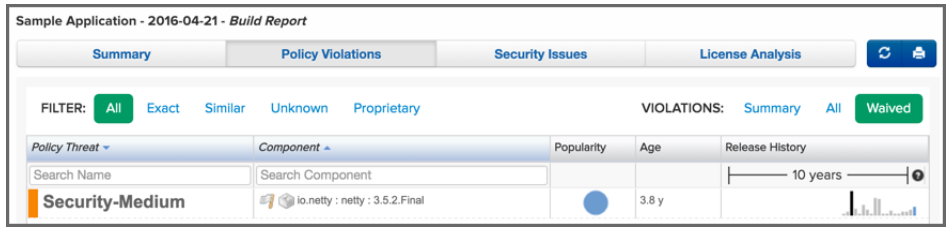


Figure 12.39: Waiver Visualization on Policy Tab

That is, by adding a waiver, you indicate that this particular component, either in the scope of this application (what we would do in our example), or all applications for the organization, is waived from this particular policy. In fact, if you desired you could even specify that you want to waive all components (within scope of an application or organization) from a policy.

Now the important thing to take note of here, is that while this waiver seems to be the answer to policy violations strife, you are actually waiving an entire policy. This means all constraints, and in turn, all

conditions. It's no surprise why that should be something that's limited. For this reason, before waiving something, it's good practice to review some alternatives. A waiver very much does allow you to simply bypass all controls.

OK, we've harped enough on the warnings. The benefits of waivers can be just as numerous. This is because even with endless customizations, you will encounter situations where a policy just doesn't apply. It's important to take a look at the full range of waiver functionality, so let's look at how to add, view, and when necessary, remove a waiver.

12.9.2 Adding a Waiver

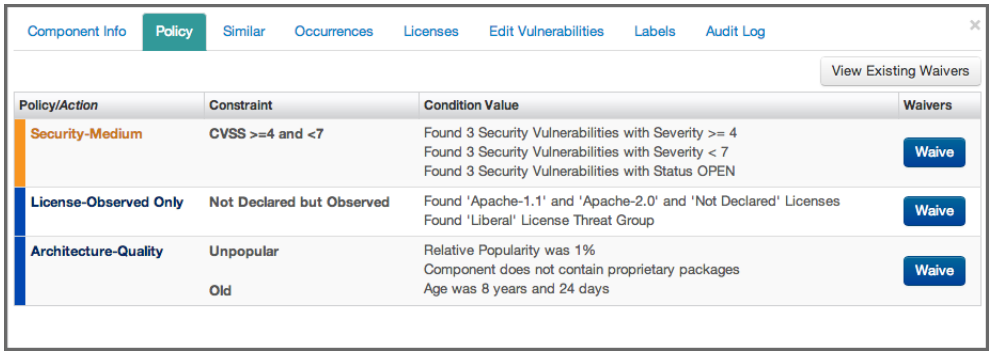


Figure 12.40: Waiver Button

1. Access an Application Composition Report.
2. Navigate to the *Policy Violations* tab on the report, and click on a component that has policy violations. This will display the Component Information Panel (CIP).
3. Click the *Policy Violations* tab. This will display the list of Policy Violations for the Component visible in Figure 12.40.
4. Click the *Waive* button next to the violation you wish to waive. A modal dialog similar to Figure 12.41 will display.
5. There are several options at this point, and each should be carefully considered:
    a. The first option defines the scope for the waiver. This can be either the current application, or all applications for the organization.
    b. The second option defines the target of the waiver. That is the currently selected component, or all components.

6. Enter an optional *Comment*, and then click the *Yes* button to process the waiver.

Warning
When processing a waiver, depending on the options that are chosen, you can effectively waive a policy for all components, for all applications in an organization. Since this will waive the entire policy, not just this violation, it may be a good idea to ensure adjusting the policy would not provide a solution that is more visible to all users.

Figure 12.41: Options to Apply Waiver to the Application or the Entire Organization

12.9.3 Viewing and Removing a Waiver

As we mentioned previously, component violations can be waived for a single component in a single application, all the way up to all components in all applications. This means, that a violation for a component in your application could have been waived elsewhere. A good practice when reviewing the Application Composition Report is to check and see what violations have been waived for components in your application. Here are a couple examples of why this is important:

- Scenario 1: A violation for a component has been waived, and the component has additional violations. Depending on the view selected, at least one of these additional violations will be displayed.

- Scenario 2: The only violation for a component has been waived. Given that the component has no additional violations, it will be moved into the **None** policy threat group (light blue) in the Summary view, while the other views will only show the waived violation.

To view waived violations for your components, follow the instruction below.

1. First, access an Application Composition Report.
2. Navigate to the *Policy Violations* tab on the report. Just above the list of components, and to the right of the report, you will see three options in the Violations filter:
    a. Summary - this is the default view of the Policy tab. It is important to note, that even though this view will display all components, only the highest threat violation per component is displayed. In this view, components with waived violations may have been moved to the **None** policy threat group (light blue).
    b. All - clicking this filter option will display every violation for all components in your application. This may result in the appearance of duplicates in the component list. Violations that have been waived will be indicated by a white flag icon.
    c. Waived - clicking this filter option will display only the waived violations. In this view, you will only see those components where violations have been waived. Each component will have a white flag icon, and it is likely you will not see all components. This view may also produce the appearance of duplicated components.
3. Click on a component to display the Component Information Panel (CIP). For an example, see Figure 12.40.
4. At the top of the of the component list, click on the *View Existing Waivers* button. A modal will be displayed showing all the waivers for the component, as well as the associated descriptions.
5. Click the remove icon, which resembles a minus sign.
6. A message will ask you to confirm this removal. Click the *Remove* button to continue.

Note
Because some waivers can be set for all applications, and even all components, it's important to understand the impact of removing a waiver. Be sure to verify with the application or organization owner, the intended scope of the waiver.

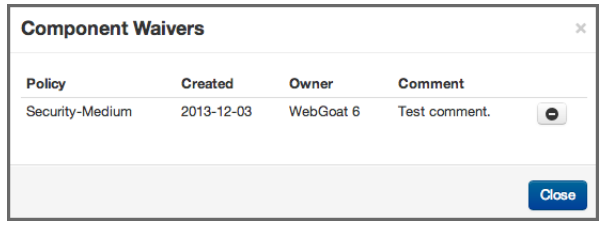


Figure 12.42: View and Remove Waivers

12.10 Policy Reevaluation

You will likely find a number of consistent themes through this documentation. One of these is that regular policy review and refinement should be part of your company's approach to policy management.

Accomplishing this successfully could potentially mean regularly rebuilding applications or publishing them to repositories several times over. Not to mention that in the case of waiting for builds, you might wait hours before an evaluation is able to run.

While there are a variety of reasons this can happen (e.g. build times are slow), the important thing is that access to the new results could be delayed. If you've made a change to policy you won't be able to tell if that made a difference. Then it's highly likely, you'll need to make another change, and then wait again. Luckily there is an alternative which allows you to reevaluate the results of an evaluation.

Using the existing component information from the most recent evaluation against the current policies - which you might have changed since the last build and analysis - you can update an Application Composition Report.

To do this, you can use policy reevaluation to see how your changes affect the current policy. The policy reevaluation button, located in the top right of the Application Composition Report (to the left of the PDF Export/Printer icon). Simply click this button displayed in Figure 12.43, and any policy changes you've made will be considered against the data of the current report.

Figure 12.43: Application Composition Report Buttons For Printing and Reevaluation

Alternatively you can reevaluate policies right from the application configuration screen in the IQ Server. Simply find your application, and locate the stage for the Application Composition Report you want to reevaluate under the application name beside the icon. Any stage that had a report processed will have a reevaluation icon right beside the stage name.

Of course, it's possible other data in the application could have changed, and that might not be realized until the next build. However, this will give you a good idea of how immediate policy changes impact any violations you currently have.

Note
Policy Reevaluation will not enact any actions you may have attached to your policies.

12.11 PDF Report

Not everyone will have access to the IQ Server or any of the integrated enforcement points, and in turn, any of the associated reports. However, certain individuals or teams would likely still benefit from the information the Application Composition Report provides. Even if that's not your particular situation, you may reach a point where you would like to produce an archive of a Application Composition Report for historical and audit purposes. Given this need, every report you produce can be converted into a PDF.

Though the information presented in both the web application and the PDF are nearly identical, there are a few difference, mainly formed out of the contrasting visual and layout capabilities of a web application versus PDF. Below, we'll discuss how to create this PDF version as well as highlight some of the differences between the two.

12.11.1 Creating the PDF

With the Application Composition Report open, find the set of blue icons in the top right. While the first icon is related to reevaluating the report, the button on the right allows you to create a PDF version of the report. Simply, click on this button and your browser will prompt you to download a PDF version of the report.

Note
The report filename will be unique each time you use the button. However, in general the report filename will use the following pattern: `applicationName-stageName-timestamp.pdf`.

12.11.2 Reviewing the PDF

The information provided by the PDF is identical to the information that is provided within the Application Composition Report in the application user interface. This includes the *Summary*, *Policy*, *Security Issues*, and *License Analysis* tabs. Within the PDF, the order of information is presented top to bottom, following the logic of the report tabs from left to right. With the exception of the first page, which provides the *Summary*, each section has a label to indicate the corresponding tab of the Application Composition Report:

Policy Violations
: displays a list of all policy violations, ordered by threat level, for each component.

Security Issues
: displays a list of all security issues, ordered by SV score, for each component.

License Analysis
: displays a list of components, ordered by threat level of the associated license threat group for the license(s) for the component.

Components
: lists all components identified during the scan, as well as a summary of the data provided in the other tabs.

Summary

The summary section is identical to the HTML version of the report and visible in Figure 12.44

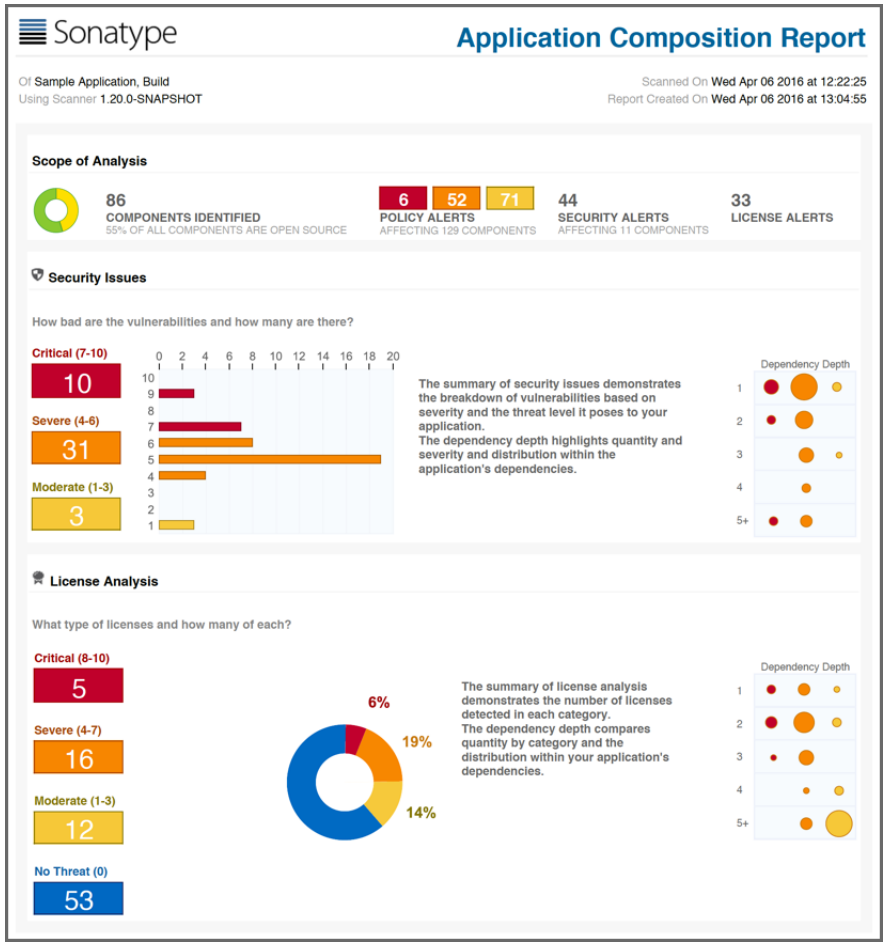
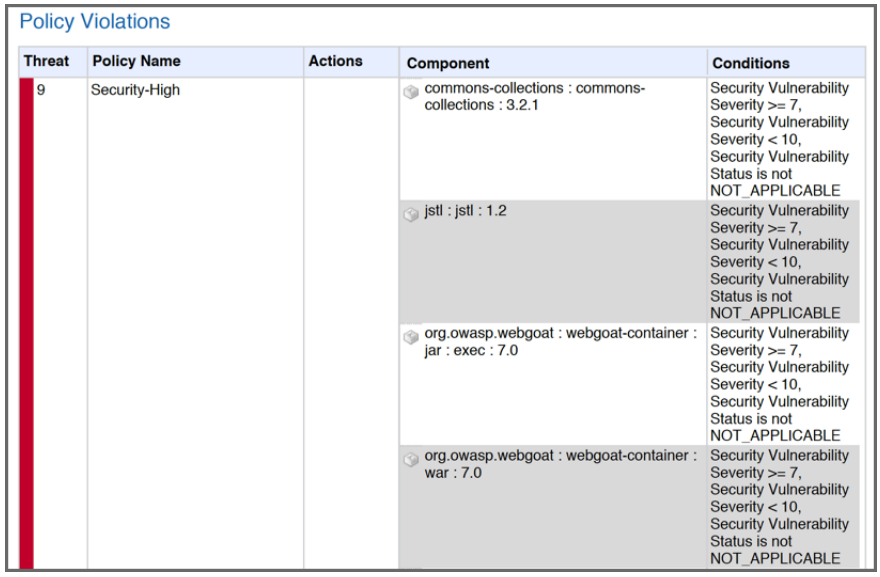


Figure 12.44: Summary Section of the Application Composition Report in PDF Format

Policy Violations

The *Policy Violations* section as visible in Figure 12.45 displays the details for all scanned components. This matches the data displayed in the *Policy* tab of the *Component Information Panel* (CIP). It should be noted, that depending on the number of violations in your application, this section could be very long.



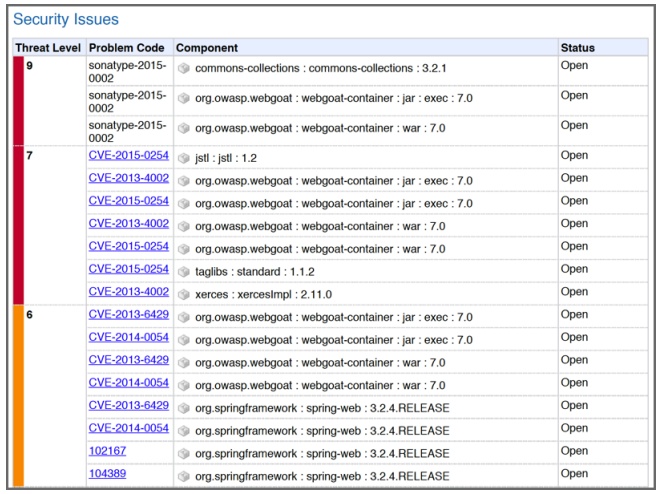
Policy Violations

Threat	Policy Name	Actions	Component	Conditions
9	Security-High		commons-collections : commons-collections : 3.2.1	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE
			jstl : jstl : 1.2	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE
			org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE
			org.owasp.webgoat : webgoat-container : war : 7.0	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE

Figure 12.45: Policy Violations Section of the Application Composition Report in PDF Format

Security Issues

The *Security Issues* section displays a breakdown of all security issues found in the scan of the application, matching what is displayed in the HTML version of the report. An example is available in Figure 12.46



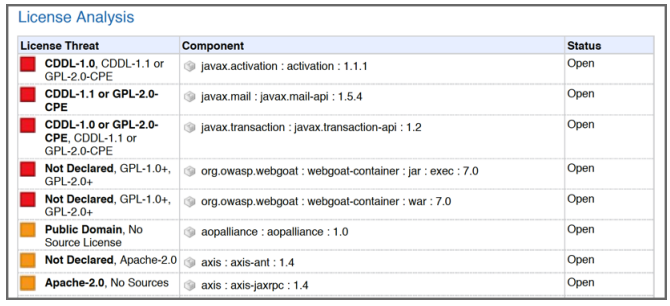
Security Issues

Threat Level	Problem Code	Component	Status
9	sonatype-2015-0002	commons-collections : commons-collections : 3.2.1	Open
	sonatype-2015-0002	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	sonatype-2015-0002	org.owasp.webgoat : webgoat-container : war : 7.0	Open
7	CVE-2015-0254	jstl : jstl : 1.2	Open
	CVE-2013-4002	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2015-0254	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2013-4002	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2015-0254	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2015-0254	taglibs : standard : 1.1.2	Open
	CVE-2013-4002	xerces : xercesImpl : 2.11.0	Open
6	CVE-2013-6429	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2014-0054	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2013-6429	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2014-0054	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2013-6429	org.springframework : spring-web : 3.2.4.RELEASE	Open
	CVE-2014-0054	org.springframework : spring-web : 3.2.4.RELEASE	Open
	102167	org.springframework : spring-web : 3.2.4.RELEASE	Open
	104389	org.springframework : spring-web : 3.2.4.RELEASE	Open

Figure 12.46: Security Issues Section of the Application Composition Report in PDF Format

License Analysis

The *License Analysis* section displays a breakdown of all license issues found in the scan of the application, matching what is displayed in the HTML version of the report. It should be noted that depending on your license threat groups, and license assignments, this section of the report could be very long. A short example is displayed in Figure 12.47.



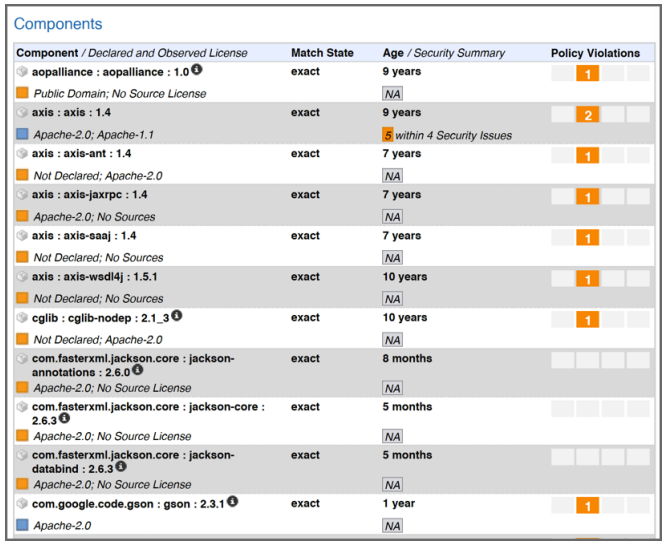
License Analysis

License Threat	Component	Status
CDDL-1.0, CDDL-1.1 or GPL-2.0-CPE	javax.activation : activation : 1.1.1	Open
CDDL-1.1 or GPL-2.0-CPE	javax.mail : javax.mail-api : 1.5.4	Open
CDDL-1.0 or GPL-2.0-CPE, CDDL-1.1 or GPL-2.0-CPE	javax.transaction : javax.transaction-api : 1.2	Open
Not Declared, GPL-1.0+, GPL-2.0+	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
Not Declared, GPL-1.0+, GPL-2.0+	org.owasp.webgoat : webgoat-container : war : 7.0	Open
Public Domain, No Source License	aopalliance : aopalliance : 1.0	Open
Not Declared, Apache-2.0	axis : axis-ant : 1.4	Open
Apache-2.0, No Sources	axis : axis-jaxrpc : 1.4	Open

Figure 12.47: License Analysis Section of the Application Composition Report in PDF Format

Components

As mentioned above, this section brings together information from all the others. It displays the highest security issue identified (and the associated CVS Score), any declared and/or observed licenses (and the highest threat level of the associated), the match state, age, and the policy violation counts for each threat level band (red, orange, yellow, and blue) for each component. An example is displayed in Figure 12.48. In most cases this section can be used as a detailed bill of materials.



Components

Component / Declared and Observed License	Match State	Age / Security Summary	Policy Violations
aopalliance : aopalliance : 1.0	exact	9 years	1
Public Domain; No Source License		NA	
axis : axis : 1.4	exact	9 years	2
Apache-2.0; Apache-1.1		5 within 4 Security Issues	
axis : axis-ant : 1.4	exact	7 years	1
Not Declared; Apache-2.0		NA	
axis : axis-jaxrpc : 1.4	exact	7 years	1
Apache-2.0; No Sources		NA	
axis : axis-saaj : 1.4	exact	7 years	1
Not Declared; No Sources		NA	
axis : axis-wsdl4j : 1.5.1	exact	10 years	1
Not Declared; No Sources		NA	
cglib : cglib-nodep : 2.1_3	exact	10 years	1
Not Declared; Apache-2.0		NA	
com.fasterxml.jackson.core : jackson-annotations : 2.6.0	exact	8 months	
Apache-2.0; No Source License		NA	
com.fasterxml.jackson.core : jackson-core : 2.6.3	exact	5 months	
Apache-2.0; No Source License		NA	
com.fasterxml.jackson.core : jackson-databind : 2.6.3	exact	5 months	
Apache-2.0; No Source License		NA	
com.google.code.gson : gson : 2.3.1	exact	1 year	1
Apache-2.0		NA	

Figure 12.48: Components Section of the Application Composition Report in PDF Format

Note
In some cases a URL for the project is provided. This is indicated by an information icon.

Chapter 13

Sonatype CLM and Repository Management

Repository managers allow you to manage software components required for development, deployment, and provisioning and fulfill a central role for component lifecycle management. A repository manager greatly simplifies the maintenance of your own internal repositories and access to external repositories. Using a repository manager is a recommended best practice for development efforts using Apache Maven or other build systems with declarative and automated transitive dependency management.

By proxying external repositories as well as providing a deployment target for internal components, a repository manager becomes the central and authoritative storage platform for all components. You can completely control access to, and deployment of, every component in your organization from a single location. It allows you to control, which components get into your products from external sources as well as examine, and keep track of artifacts produced by your build systems.

In terms of the incoming components a repository manager allows you to secure the connection to an external repository and ensure that your component usage is not publicly exposed.

Just as Source Code Management (SCM) tools are designed to manage source artifacts, repository managers have been designed to manage and track external dependencies and components generated by your build. They are an essential part of any enterprise or open-source software development effort. They enable greater collaboration between developers and wider distribution of software bring increased build performance due to local component availability and reduced bandwidth needs by avoiding repeated downloads to your setup.

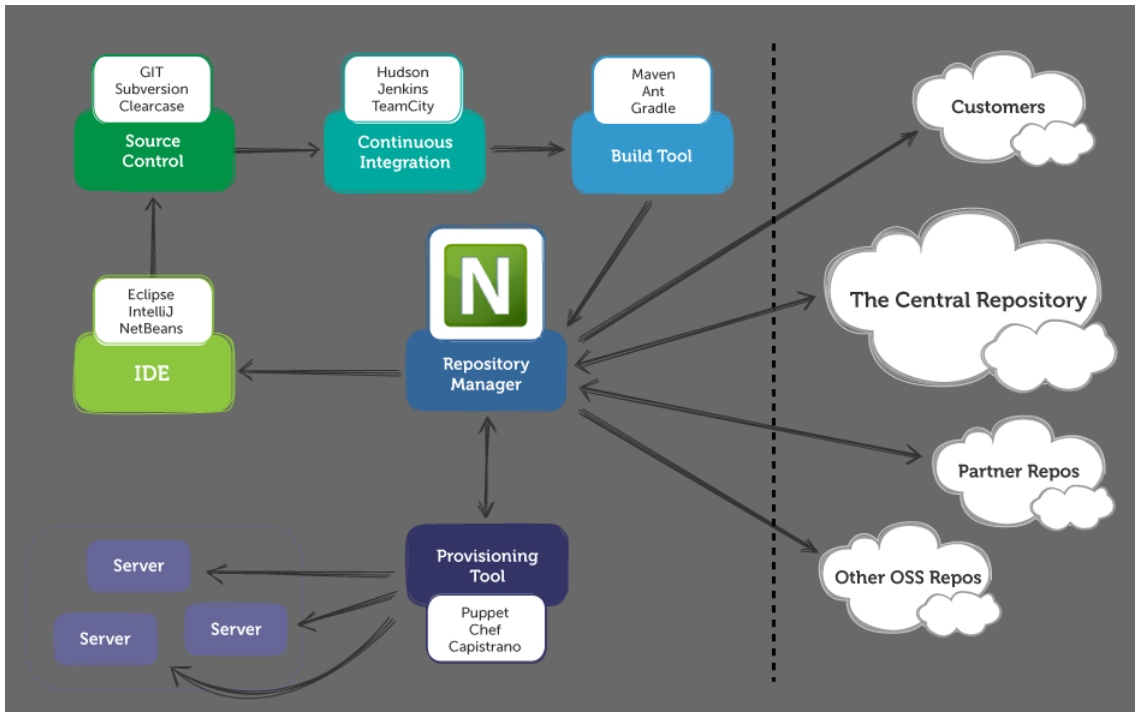


Figure 13.1: The Central Role of A Repository Manager in Your Infrastructure

For additional information, the Repository Management with Nexus book provides an extensive introduction to repository management, its advantages, and stages of adoptions for further reference.

Chapter 14

IQ for Nexus Repository Manager

Tip
The features discussed in this chapter require IQ Server and Nexus Repository Manager with the Repository license plus either the Firewall or Lifecycle license. The only exception is Audit and Quarantine, which require the Repository and Firewall licenses.

IQ for Nexus Repository Manager allows you to integrate IQ Server's policy management and component intelligence features with proxy repositories in Nexus Repository Manager Pro.

This chapter assumes you have installed Nexus Repository Manager Pro (not Nexus Repository Manager OSS) and IQ Server with the appropriate licenses for the features you want. To integrate Nexus Repository Manager and IQ Server, you need the Repository license plus either the Firewall or Lifecycle license. To use the Audit and Quarantine features, you need the Repository and Firewall licenses.

14.1 Integrating Nexus Repository Manager 2.x and IQ Server

Tip
The features discussed in this section require IQ Server and Nexus Repository Manager Pro with the Repository license plus either the Firewall or Lifecycle license.

14.1.1 Connecting to IQ Server

The first step to integrating IQ Server features with Nexus Repository Manager 2.x is connecting to IQ Server from Nexus Repository Manager.

To configure the connection to IQ Server, follow these instructions:

1. Click on the *IQ Server Connection* menu item in *Administration*, located on the left of the Nexus Repository Manager application window.
2. Enter the URL for your IQ Server installation.
3. Select an *Authentication Method*:
    a. *User Authentication*: Enter the username and password.
    b. *PKI Authentication*: Delegate authentication to the JVM.

    Tip
    It is recommended that you create a unique machine account with desired permissions for linking IQ Server with Nexus Repository Manager. At a minimum, the account needs Evaluate Individual Components permission at the repositories level for Audit and Quarantine features and/or Evaluate Applications permission at the application level for Staging functionality. For more information about permissions, see Role Management in the Security Administration chapter.

4. Optionally, you can configure these properties:
    a. Enter a *Request Timeout*.
    b. Enter information in the *Properties* input field using a key=value definition per line. For example:

```
procArch=false
ipAddresses=true
operatingSystem=false
```

These properties are passed to IQ Server and can, for example, determine what properties are logged as part of a validation. Consult the IQ Server documentation for suitable parameters. In most use cases you will not need to configure any properties.

5. Click *Test Connection* to verify the information you have entered is correct and a connection to IQ Server can be established.
6. Click the *Save* button.

If successfully connected, a list of available applications in IQ Server is displayed as shown in the figure below.

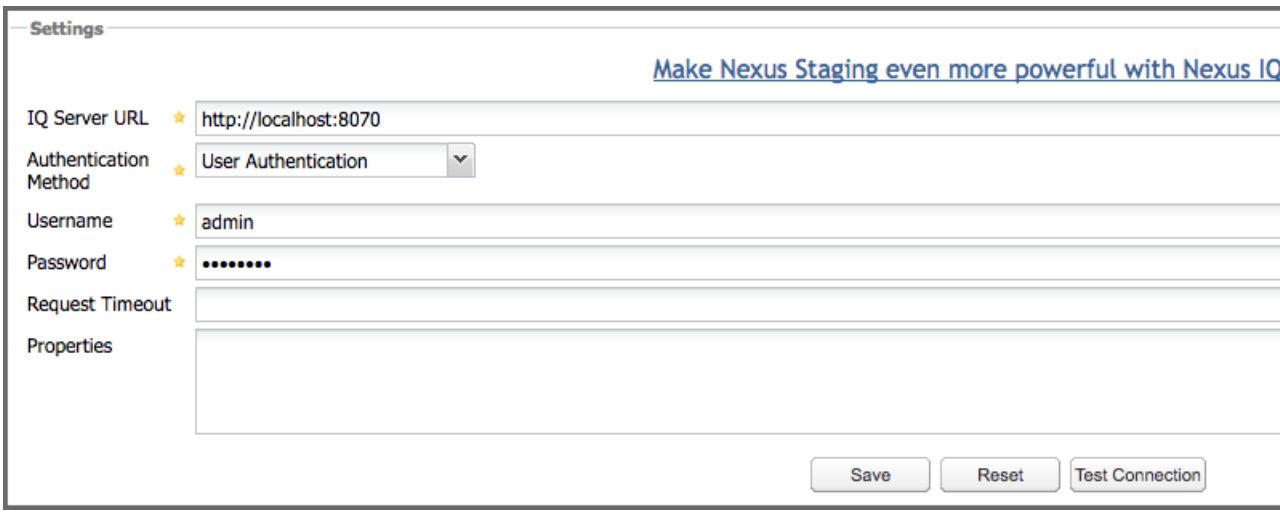


Figure 14.1: IQ Server Connection Tab in Nexus Repository Manager 2.x

Alternatively you can enable, disable, and/or configure IQ Server integration by adding the *IQ: Server Connection* capability like any other capability as documented in the Accessing and Configuring Capabilities section of the Nexus Repository Manager book.

Note
The features described here require licenses for Nexus Repository Manager as well as IQ Server. These are made available through purchase of our solutions. You can obtain them from the Sonatype Support team.

14.1.2 Viewing Component Information

In Nexus Repository Manager, the *Artifact Search* feature helps you find components in your repositories. In the search results, you can drill down for more detailed information. For example, after you perform a search, click *Show All Versions*, if available, in the search results to see information such as version, age, popularity, and more. This is displayed in the figure below.

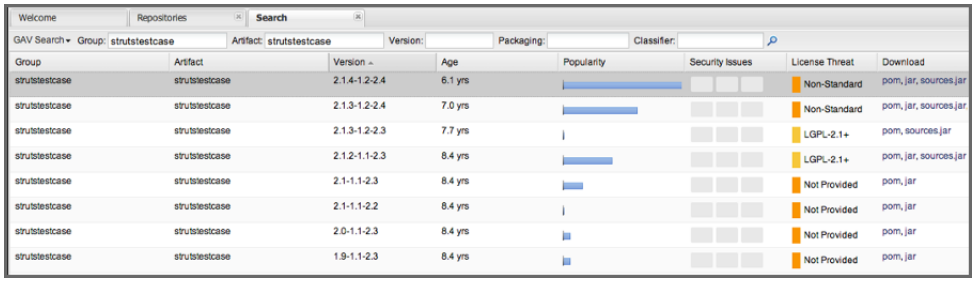


Figure 14.2: Typical Search Results in Nexus Repository Manager 2.x

Tip
To get results that are not in the local Nexus Repository Manager cache, you will want to make sure the *Download Remote Index* option is enabled for the proxy repository. For guidance on this, check out section 6.2.4 (specifically Fig 6.9): Configuring Repositories in the Nexus Repository Manager book.

Once you've configured the IQ Server connection, additional component information such as security issues is displayed in the Nexus Repository Manager search results, for example:

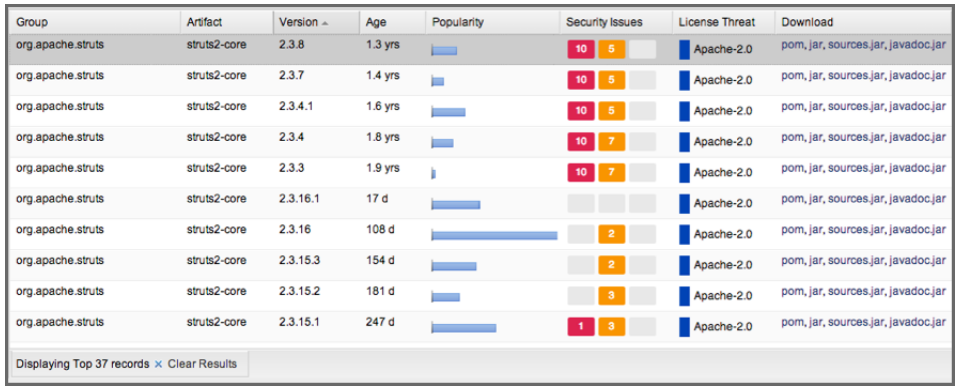


Figure 14.3: Nexus Repository Manager Search Showing All Versions

Note
Nexus Repository Manager search is only available for open source Java components.

You can access more detailed component information by selecting a component and clicking the *Component Info* tab located below the search results.

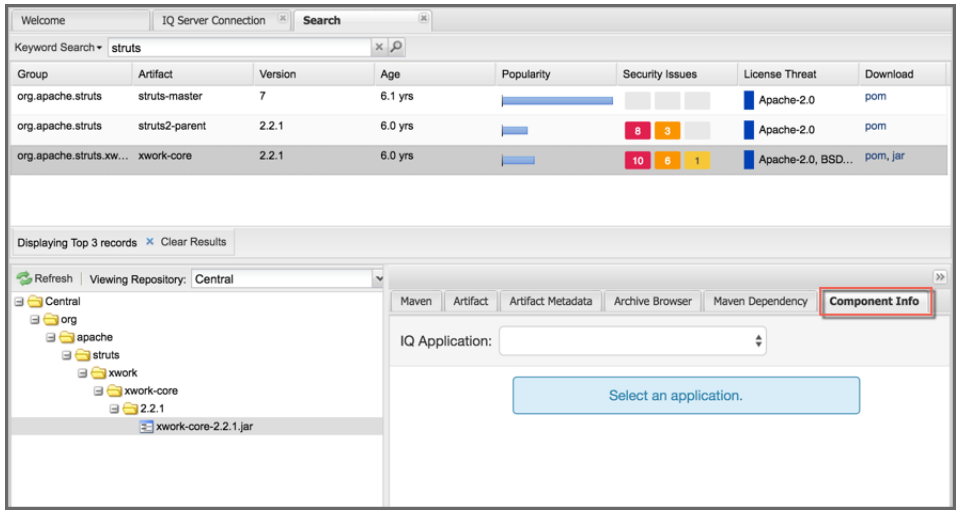


Figure 14.4: Accessing the Component Info Tab

Note
Only users that are logged in will be able to see the Component Info tab.

On the *Component Info* tab, when you select one of the applications configured in your IQ Server, the Component Information Panel (CIP) is displayed. It contains the most granular details about a component.

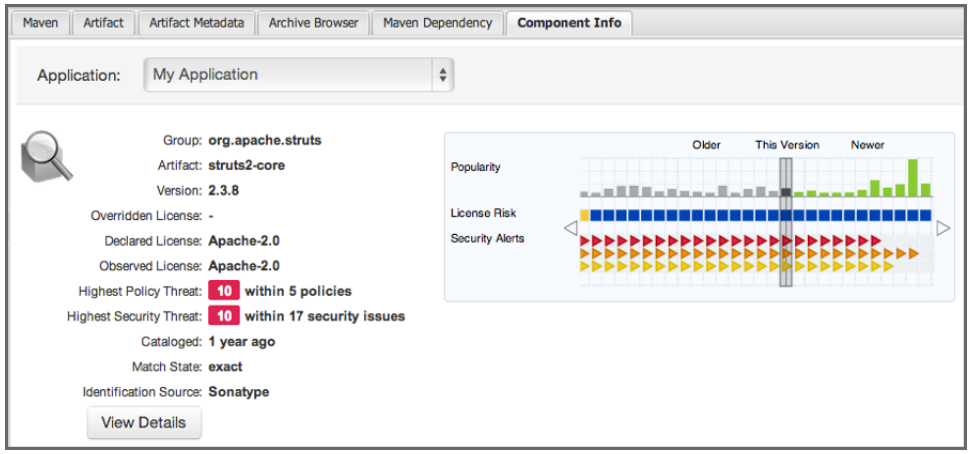


Figure 14.5: Component Information Panel

Component Info

The *Component Info* tab displays the following information about a specific component:

- *Declared License* - Any license(s) that has been declared by the author.
- *Observed License* - Any license(s) found during the scan of the component's source code.
- *Effective License* - Either all licenses included in the Declared or Observed Group, or the overridden license.
- *Coordinates* - The identifying information for a component. For known components, all available coordinate information will be displayed.
- *Highest Policy Threat* - The highest threat level policy that has been violated, as well as the total number of violations.
- *Highest Security Threat* - The highest threat level security issue and the total number of security issues.

- *Cataloged* - The age of the component based on when it was first uploaded to an accessible storage site such as the Central Repository, for example.
- *Match State* - How the component was matched (exact, similar, or unknown).
- *Identification Source* - Whether a component is identified by Sonatype, or claimed by your own process.
- *Website* - If available, an information icon providing a link to the project is displayed.

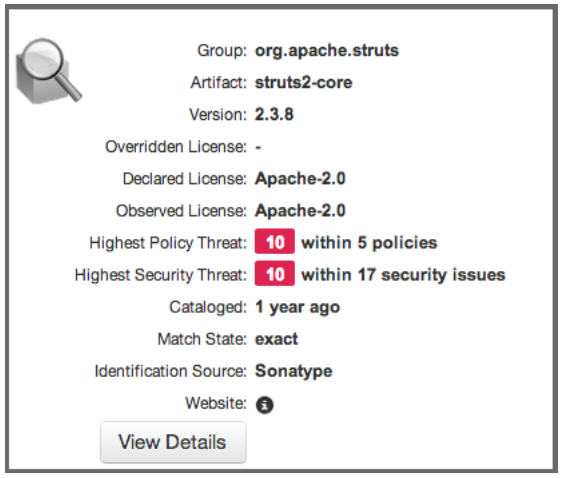


Figure 14.6: CIP Text

The *Component Info* tab also includes a graph, which is laid out like a grid with each vertical column representing a particular version. The selected version is identified by a vertical line. You can move the line horizontally to learn about different versions of a component. The information includes:

- *Popularity* - The relative popularity for each version is shown as a bar graph. The taller the bar the more popular the version.
- *License Risk* - A display of risk based on license threat group settings from IQ Server.
- *Security Alerts* - For each version, the highest security threat will be displayed by color, with the highest shown as red, and no marker indicating no threat.

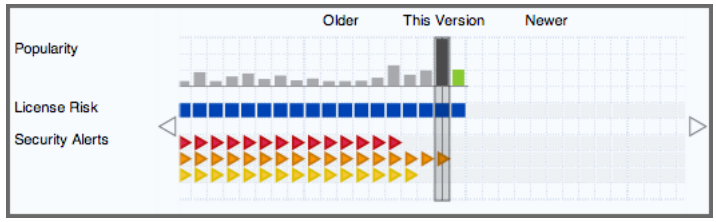


Figure 14.7: CIP Graph

14.1.3 Component Details

The *Component Info* tab in Nexus Repository Manager has a *Component Details* button that opens a new tab with information about any policy violations, license issues, or security vulnerabilities that are known about a specific component.

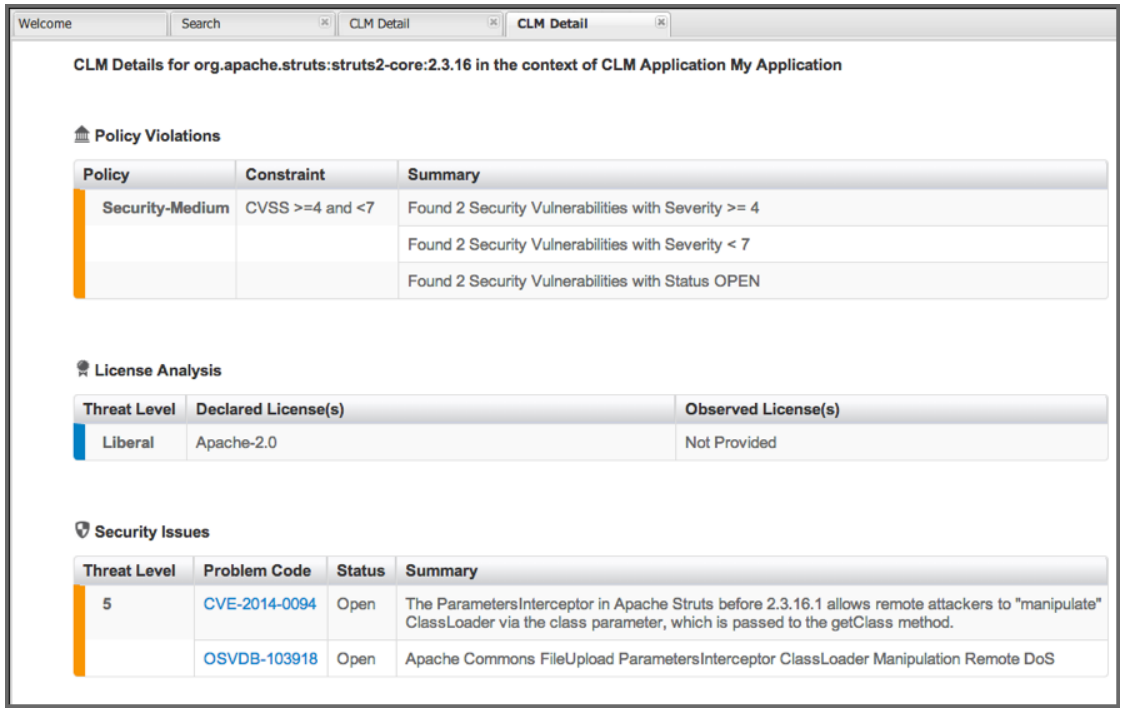


Figure 14.8: View Details

Note
In order to see the details for additional components, select another component from the search results, or select a different version in the CIP, and then click the *View Details* button.

14.1.4 Using Staging to Control Releases

With Staging, you can combine the release process controls in Nexus Repository Manager with the component intelligence from IQ Server to test a release automatically before its deployed.

To use IQ Server with Nexus Repository Manager 2.x, you must first create the following items:

In IQ Server

- An Organization
- An Application
- A Policy

In Nexus Repository Manager

- A Staging Profile

Note
Before using IQ Server for staging you should be familiar with the general setup and usage patterns of the Nexus Repository Manager Staging Suite documented in the chapter on staging, located in the Nexus Repository Manager book. There, you will be guided through the process to get Nexus Repository Manager prepared to handle your staging needs.

14.1.4.1 Staging Profile Configuration

To utilize IQ Server evaluation and policy features as part of your build promotion you will need to select an *IQ Server Application* as part of the staging profile configuration. This is done via Nexus Repository Manager. An example is provided below.

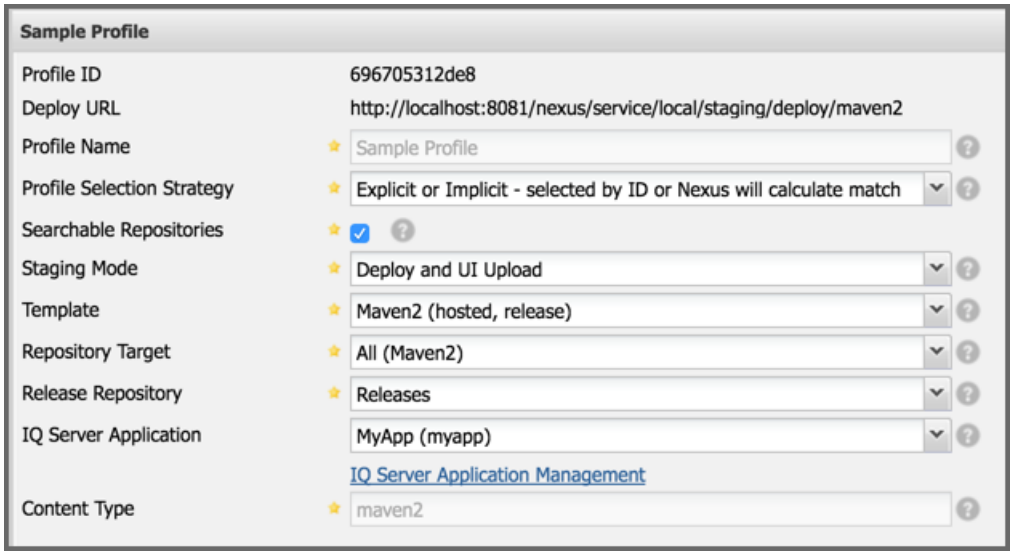


Figure 14.9: Staging Profile with an IQ Server Application Configured

14.1.4.2 Policy Actions for Staging

While not a requirement for using IQ Server with Nexus Repository Manager staging, IQ Server does have the ability to Fail or Warn on staging closure. This is managed by setting the *Stage Release* and *Release* actions for each policy. These policy actions can be configured to warn, fail, or no action (default). The figure below provides an example policy that would warn for a staging deployment and fail a release.

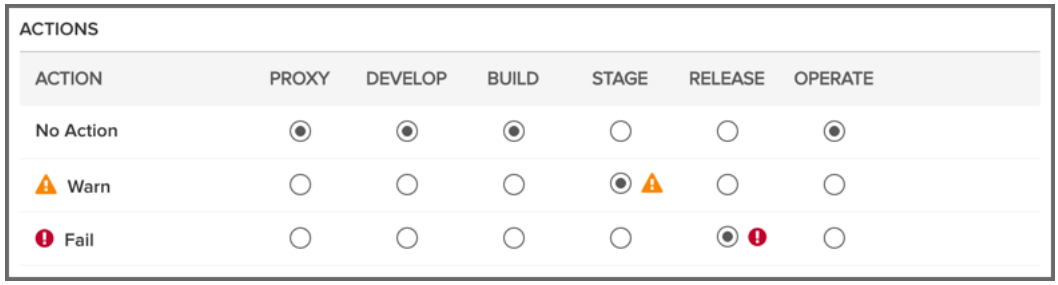


Figure 14.10: Staging and Release Configuration for a Policy in IQ Server

The configuration of the *Stage Release* action is used for closing the staging repository. Based on the action chosen, the staging repository responds to policy violations as follows:

- If the policy action is set to *Fail*, closing the staging repository fails.
- If the policy action is set to *Warn*, the staging repository closes successfully, but a warning is produced.
- If the policy action is set to *No Action*, the staging repository closes successfully regardless of any policy violations.

For more information on setting these actions see the Actions section in the Basic Policy Management chapter.

14.1.4.3 Policy Actions for Release Repositories

Nexus Repository Manager also has actions specific to the *Release* feature, and these can be configured to fail, warn or do nothing and are used for releasing or promoting the staging repository.

Once the staging profile is configured with the IQ Server application ID, any deployment triggers an evaluation with IQ Server. The results are visible as *Activity* for the staging repository as shown in the figure below. Any rule failures are provided with further information in the detail panel. The *View Full Report* button links back to the detailed Application Composition Report.

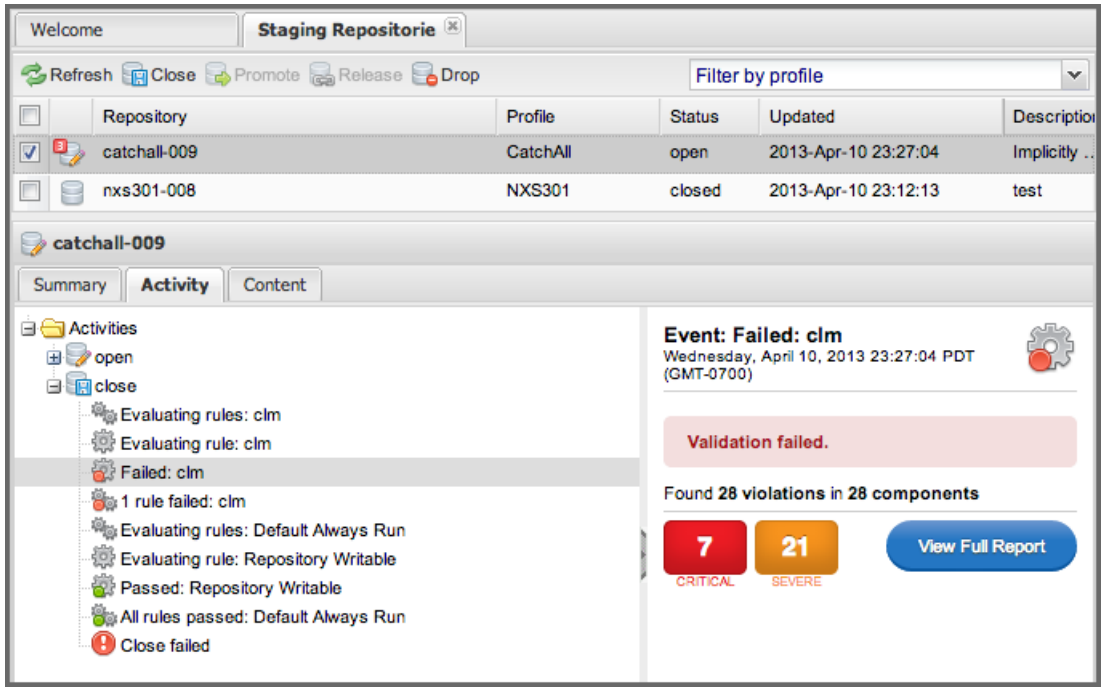


Figure 14.11: Staging Repository Activity with IQ Server Evaluation Failure and Details

14.1.5 Using Audit and Quarantine

Tip

The features discussed in the Using Audit & Quarantine section require Nexus Repository Manager Pro and IQ Server with the following licenses: Repository and Firewall.

The Audit and Quarantine features provide a way to protect your development environment from risky or undesirable components. These features use IQ Server policy management to identify, and if desired, prevent a proxy repository from serving unwanted components.

Before activating Audit and Quarantine, there are several items you need to complete:

- Both Nexus Repository Manager and IQ Server must be running and must have a working connection

between the two systems. For more information, see the Section 14.1.1 section in this chapter.

- In Nexus Repository Manager, you need the following privileges to use Audit and Quarantine:
  - Add, edit, and delete privileges for capabilities, which allows you to configure, enable, and disable the Audit and Quarantine features.
  - Read privilege for repositories, which lets you view a results column in the Repositories tab.
    For information on assigning privileges, see the Managing Privileges section in the Nexus Repository Manager book.
- For IQ Server, you must be assigned to a role in the root organization with permissions to view and edit IQ Elements. The built-in roles of Policy Administrator and Owner have these permissions. For more information on assigning roles and permissions, see the Security Administration chapter. To learn more about the root organization, see the Organization and Application Management chapter.
- Also with regard to IQ Server, you should create a policy in the root organization that defines the rules or criteria to use when evaluating components of a proxy repository. The policy must be at the root organization level in the system hierarchy; policies at other levels are ignored by Audit and Quarantine. To learn more about creating a policy, see the Basic Policy Management chapter.

Once these items are completed, you are ready to configure Audit and Quarantine and view audit results. Each of these actions is described below in more detail.

14.1.5.1 Configuring Audit and Quarantine

You configure the Audit and Quarantine features by adding them to Nexus Repository Manager as a plug-in capability.

To configure Audit and Quarantine:

1. In Nexus Repository Manager, click *Capabilities* on the *Administration* menu.
2. Click the *New* button on the *Capabilities* tab. The *Create new capability* dialog is displayed.
3. In the *Type* list, choose *IQ: Audit and Quarantine*.
4. Configure Settings as follows:
   a. *Enabled* - Make sure the check box is selected to activate the Audit feature. The check box is selected by default.
   b. *Repository* - Select a specific proxy repository to scan, for example, Central.

c. *Quarantine* - Select the check box to quarantine any components that violate policy whenever you add new components to the selected proxy repository. This setting affects only components that are added to the repository after *Quarantine* is enabled. When a component is quarantined, the Nexus Repository Manager prevents it from being served from the proxy repository. The check box is deselected by default.

5. Click *Add* to create the new capability for Audit and Quarantine.

At this point, an audit of the selected repository is automatically started. Nexus Repository Manager contacts IQ Server and evaluates the components within the selected repository against any associated policy. The results are displayed in *Repository Results*, which is described in the next section.

Note

To successfully quarantine components when the Quarantine feature is enabled, the policy used to evaluate components must be configured to fail when policy violations occur at the proxy stage in the development lifecycle. If the policy is set to warn (rather than fail), the quarantining of components will not occur. For more information about setting policy and the proxy stage, see the Basic Policy Management chapter.

After the *IQ: Audit and Quarantine* capability is added, it appears on the *Capabilities* tab in Nexus Repository Manager as shown in the figure below.

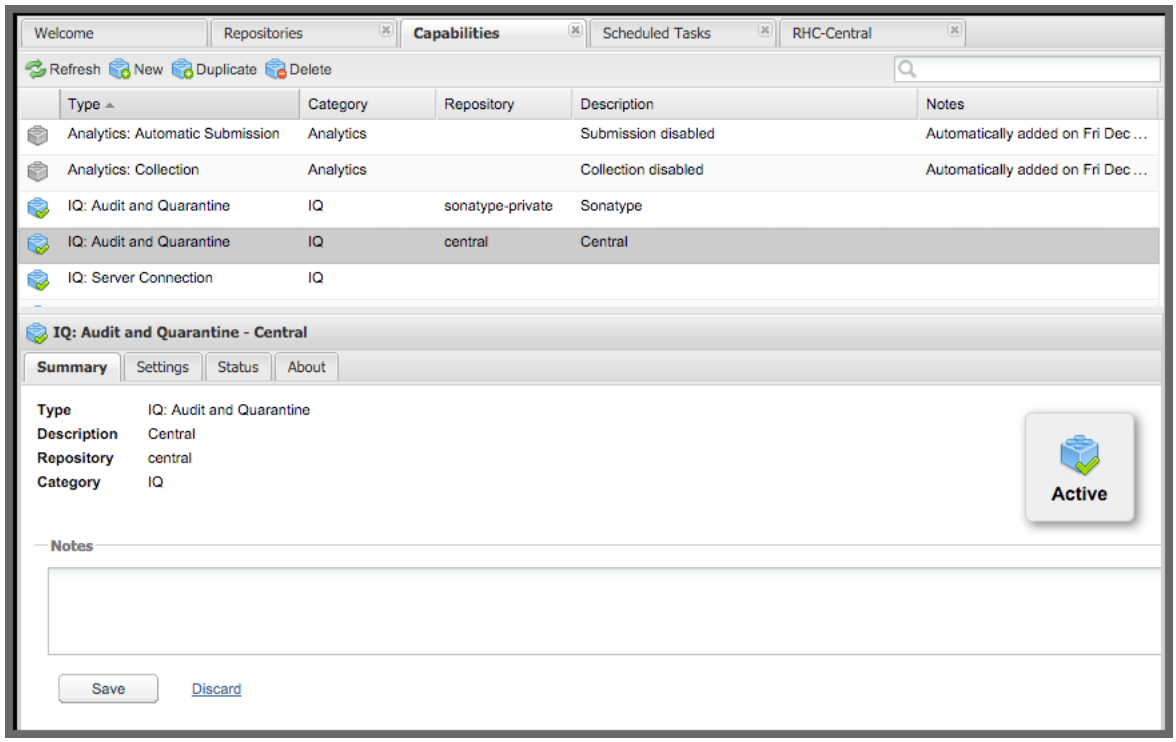


Figure 14.12: Capabilities Tab in Nexus Repository Manager

14.1.5.2 Disabling Audit and/or Quarantine

To disable Audit and/or Quarantine:

1. In the Nexus Repository Manager interface, click *Capabilities* on the *Administration* menu.
2. Click the *IQ: Audit and Quarantine* capability for a specific repository.
3. Click the *Settings* tab of the *IQ: Audit and Quarantine* capability and set the following attributes:
    a. Click the *Enabled* check box to deselect it and disable the Audit feature.

> **Note**
> When you disable the *IQ: Audit and Quarantine* capability, Quarantine is also disabled.

b. Click the *Quarantine* check box to deselect it and disable only the Quarantine feature.

Caution
When Quarantine is disabled, all quarantined components are made available for download from your proxy repository. This remains true, if you re-enable Quarantine. That is, any previously quarantined components are not quarantined again; only new components are evaluated for quarantine when you re-enable the Quarantine feature.

4. Click *Save* to save your changes or click *Discard* to undo your changes.

14.1.5.3 Releasing a Component from Quarantine

When a component is quarantined due to a violation, it is not available for download from the proxy repository. You must first resolve the violation(s) that caused the quarantine before releasing the component and making it downloadable. For information on resolving violations from labels, security vulnerabilities, or license issues, see the Application Composition Report chapter. For information on waiving policy violations, see the Waiving Repository Policy Violations section of this chapter. Once the violations are resolved, you can proceed with releasing a component from quarantine.

To release a component from quarantine:

1. In Nexus Repository Manager, select a repository that has been evaluated.
2. Click the *IQ Policy Violations* count for a repository. This opens the Repository Results hosted on IQ Server.
3. Navigate to the *Policy* tab, and click the *Quarantined* filter.
4. Click a quarantined component. This expands the row to display the *Component Information Panel* (CIP).
5. Click the *Policy* tab, and then click the *Release Quarantine* button.
6. In the confirmation box, click the *Release* button.

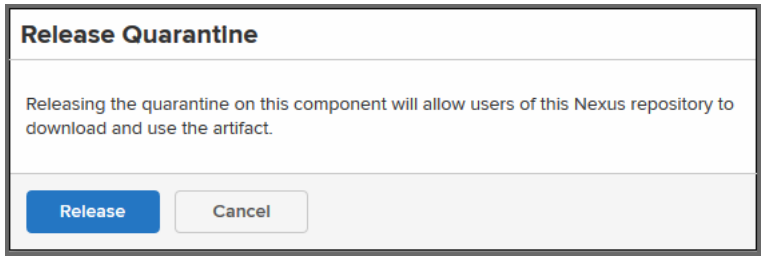


Figure 14.13: Release Quarantine

Note
Once a component is released from quarantine, it cannot be put back into quarantine even if it has subsequent policy violations. If you want to re-quarantine a component, you must delete the component from its repository. The component will be quarantined again if, during an audit, it violates a policy that is set to *Fail* at the Proxy stage.

14.1.5.4 Re-enabling Audit and/or Quarantine

To re-enable Audit and/or Quarantine:

1. In Nexus Repository Manager, click *Capabilities* on the *Administration* menu.
2. Click the *IQ: Audit and Quarantine* capability for a specific repository.
3. Click the *Settings* tab of the *IQ: Audit and Quarantine* capability and set the following attributes:
    a. Click the *Enabled* check box to enable the Audit feature.
    b. Click the *Quarantine* check box to enable to the Quarantine feature.

    Note
    Any previously quarantined components are not quarantined again even though they were quarantined in the past. Only new components are evaluated for quarantine when the Quarantine feature is re-enabled.

4. Click *Save* to save your changes or click *Discard* to undo your changes.

14.1.5.5 Viewing Repository Results

Once the Audit and Quarantine features are enabled, whenever you add a component to a proxy repository (or delete one), Nexus Repository Manager contacts IQ Server to evaluate the components within the proxy repository against any associated policy. The **IQ Policy Violations**, are summarized in Nexus Repository Manager, and detailed in IQ Server.

In Nexus Repository Manager:

The results of an audit are summarized in the *IQ Policy Violations* column of the *Repositories* tab as shown in the figure below.

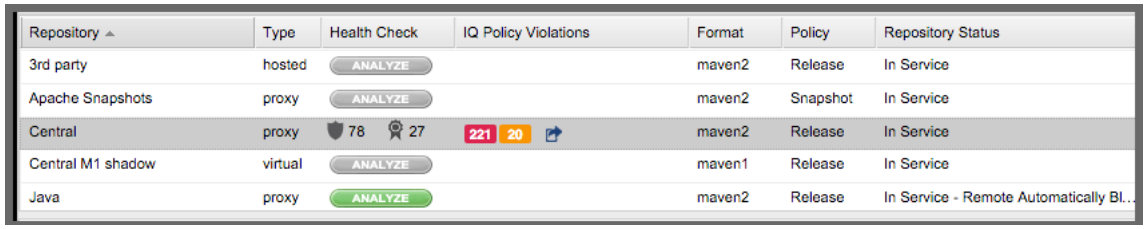


Figure 14.14: IQ Policy Violations Column

The *IQ Policy Violations* column includes the following items:

- A count of components by their highest policy violation level.
- A count of quarantined components.
- A link to *Repository Results* on IQ Server

The *IQ Policy Violations* column will also alert you if there are any errors in the audit and quarantine process. If there is an error, for example if Nexus Repository Manager cannot communicate with IQ Server, a red exclamation mark will appear to the right of the *Repository Results* link along with text pertinent to the error that occurred. Additional information will be available in the Nexus Repository Manager logs.

If you have permissions to add capabilities in Nexus Repository Manager, then you can also access *Repository Results* from the *Capabilities* tab:

1. In the *Type* list of capabilities, select *IQ: Audit and Quarantine*.
2. Click the *Status* tab of the *IQ: Audit and Quarantine* capability.
3. Click *View Results*.

Both methods open *Repository Results* on IQ Server. To learn more about the details displayed in the *Repository Results*, see Understanding Repository Results.

14.2 Integrating Nexus Repository Manager 3.x and IQ Server

Tip
The features discussed in this section require IQ Server and Nexus Repository Manager Pro with the Repository license plus either the Firewall or Lifecycle license.

14.2.1 Connecting to IQ Server

The first step to integrating IQ Server features with Nexus Repository Manager 3.x is connecting to IQ Server from Nexus Repository Manager.

To configure the connection to IQ Server:

1. In Nexus Repository Manager, click the *Administration* button on the main toolbar.
2. In the *Administration* main menu, click *Server* under *IQ Server*.
3. Configure the following settings:
   - Click to select *Where to use IQ Server* to enable IQ Server.
   - Enter the *IQ Server URL*.
   - Select an *Authentication Method*:
     - *User Authentication*: Enter a username and password.
     - *PKI Authentication*: Delegate authentication to the JVM.

Tip
It is recommended that you create a unique machine account with desired permissions for connecting IQ Server with Nexus Repository Manager. At a minimum, the account needs Evaluate Individual Components permission at the repositories level to use any available IQ Server features.

- Optionally, you can configure these properties:
  - Enter a *Request Timeout*.
  - Enter information in the *Properties* input field using a key=value definition per line. For example:

```
procArch=false
ipAddresses=true
operatingSystem=false
```

    These properties are passed to IQ Server and can, for example, determine what properties are logged as part of a validation. In most use cases you will not need to configure any properties.

4. Click *Verify connection* to test if a connection can be established.

If successful, a list the applications from IQ Server is displayed, and Dashboard appears under IQ Server on the Administration main menu.

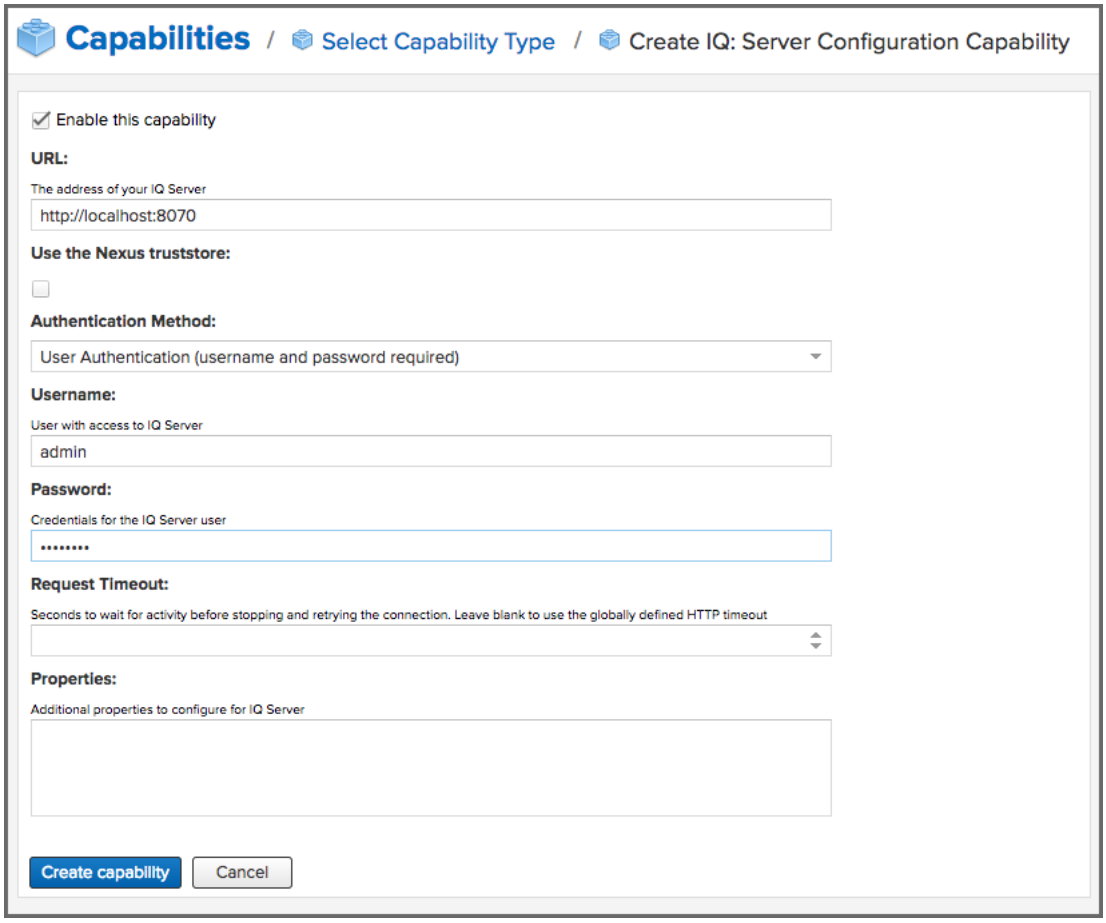


Figure 14.15: IQ Server Connection Tab in Nexus Repository Manager 3.x

14.2.2 Viewing Component and Assets Information

In Nexus Repository Manager, the *Search* feature helps you find assets and components in your repositories. In the search results, you can drill down for more detailed information. For example, after you perform a search, click a component to see its associated assets.

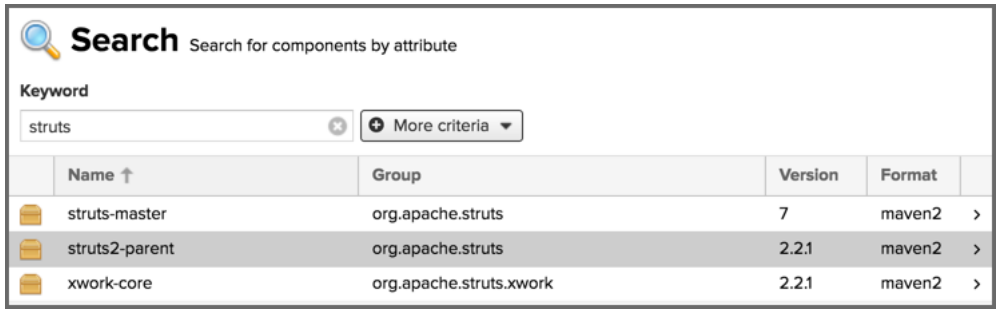


Figure 14.16: Associated Assets in Search Results in Nexus Repository Manager 3.x

Click an asset to access its summary information, attributes, and component intelligence.

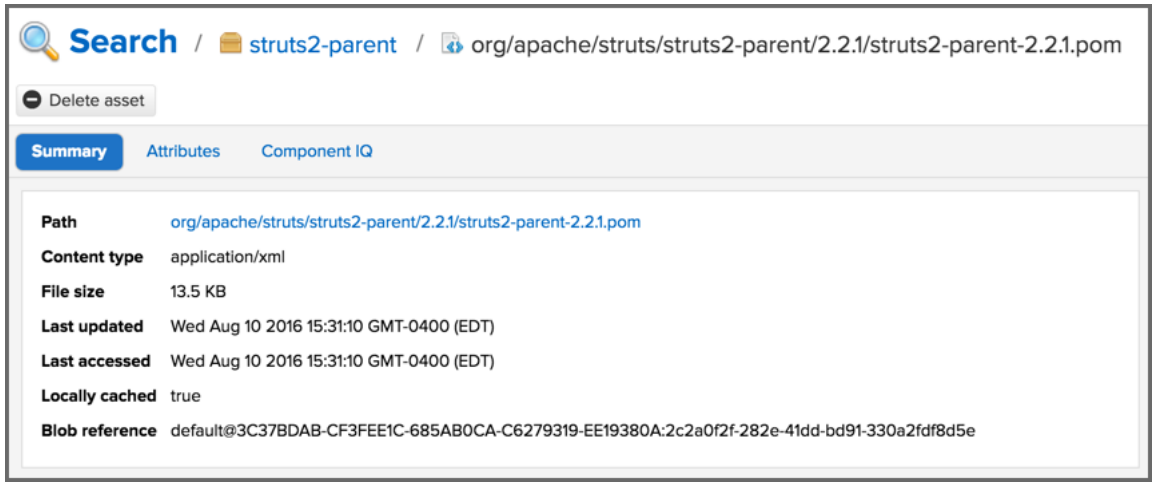


Figure 14.17: Asset Information in Nexus Repository Manager 3.x

Click *Component IQ* to get more detailed information.

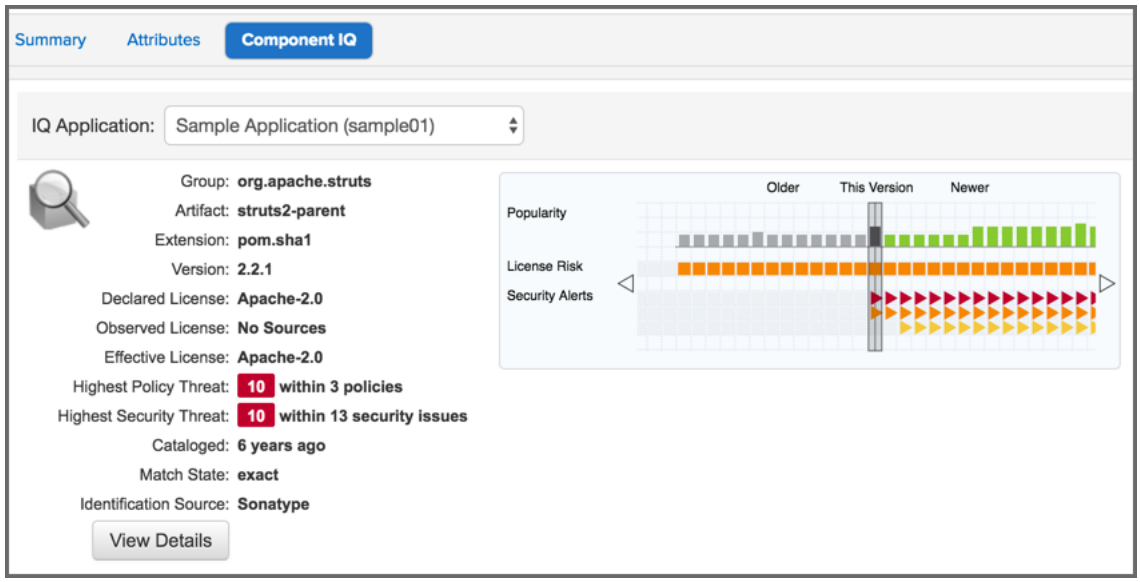


Figure 14.18: Viewing Component IQ

Component intelligence is presented in the context of an IQ Server application. Go to the *IQ Application* list and select one of the applications configured in your IQ Server. The Component Information Panel (CIP) is displayed, which contains the most granular details about a component.

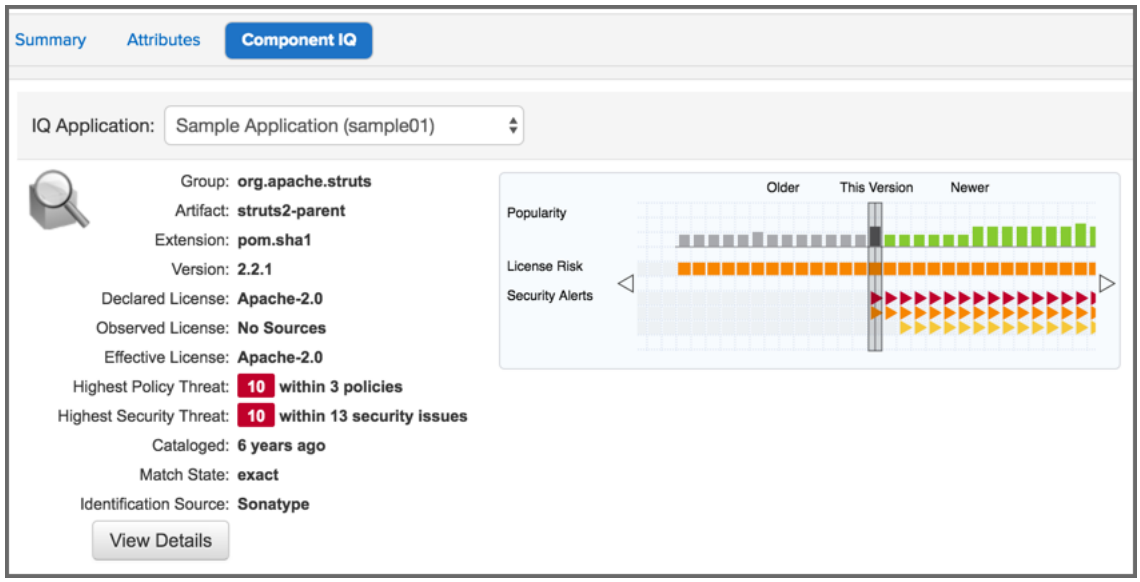


Figure 14.19: Component Information Panel

Component IQ

Component IQ displays the following information about a specific component:

- *Declared License* - Any license(s) that has been declared by the author.
- *Observed License* - Any license(s) found during the scan of the component's source code.
- *Effective License* - Either all licenses included in the Declared or Observed Group, or the overridden license.
- *Coordinates* - The identifying information for a component. For known components, all available coordinate information will be displayed.
- *Highest Policy Threat* - The highest threat level policy that has been violated, as well as the total number of violations.
- *Highest Security Threat* - The highest threat level security issue and the total number of security issues.
- *Cataloged* - The age of the component based on when it was first uploaded to an accessible storage site such as the Central Repository, for example.
- *Match State* - How the component was matched (exact, similar, or unknown).

- *Identification Source* - Whether a component is identified by Sonatype, or claimed by your own process.
- *Website* - If available, an information icon providing a link to the project is displayed.

Figure 14.20: CIP Text

Component IQ also includes a graph, which is laid out like a grid with each vertical column representing a particular version. The selected version is identified by a vertical line. You can move the line horizontally to learn about different versions of a component. The information includes:

- *Popularity* - The relative popularity for each version is shown as a bar graph. The taller the bar the more popular the version.
- *License Risk* - A display of risk based on license threat group settings from IQ Server.
- *Security Alerts* - For each version, the highest security threat will be displayed by color, with the highest shown as red, and no marker indicating no threat.

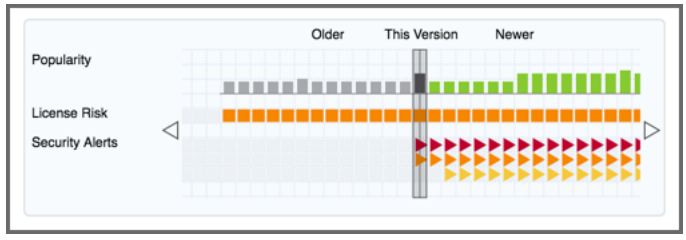


Figure 14.21: CIP Graph

For even more granular information about a specific component, click *View Details*. Any known policy violations, license issues, or security vulnerabilities are displayed on a new tab in your browser.

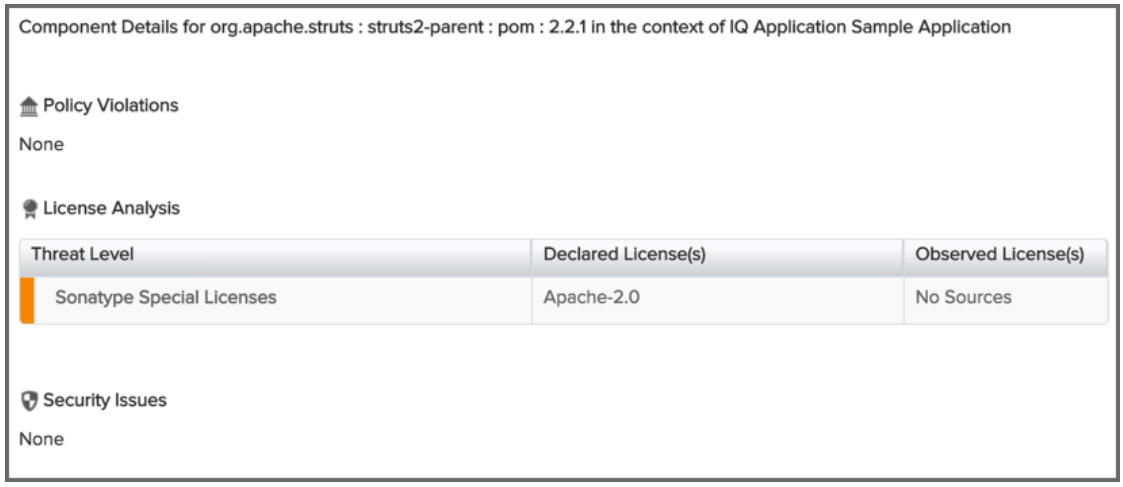


Figure 14.22: View Details

14.2.3 Using Audit and Quarantine

Tip
The features discussed in this section require Nexus Repository Manager Pro and IQ Server with the following licenses: Repository and Firewall.

The Audit and Quarantine features provide a way to protect your development environment from risky or undesirable components. These features use IQ Server policy management to identify, and if desired, prevent a proxy repository from serving unwanted components.

Before activating Audit and Quarantine, there are several items you need to complete:

- Both Nexus Repository Manager 3.x and IQ Server must be running and must have a working connection between the two systems.
- In Nexus Repository Manager 3.x, you need the following privileges:

  - Add, edit, and delete privileges for capabilities, which allows you to configure, enable, and disable the Audit feature.
  - Read privilege for repositories, which lets you view a results column in Repositories (under *Repository* in the *Administration* main menu).

    For information on assigning privileges, see the Privileges section in the Nexus Repository Manager 3.x book.

- For IQ Server, you must be assigned to a role in the root organization with permissions to view and edit IQ Elements. The built-in roles of Policy Administrator and Owner have these permissions. For more information on assigning roles and permissions, see the Security Administration chapter. To learn more about the root organization, see the Organization and Application Management chapter.
- Also with regard to IQ Server, you should create a policy in the root organization that defines the rules or criteria to use when evaluating components of a proxy repository. The policy must be at the root organization level in the system hierarchy; policies at other levels are ignored by Audit. To learn more about creating a policy, see the Basic Policy Management chapter.

Once these items are completed, you are ready to configure Audit and Quarantine and view audit results. Each of these actions is described below in more detail.

14.2.3.1 Configuring Audit and Quarantine

You configure the Audit and Quarantine features by adding them to Nexus Repository Manager 3.x as a capability.

To configure Audit and Quarantine:

1. In Nexus Repository Manager 3.x, go to the *Administration* main menu and click *Capabilities* under *System*.
2. Click the *Create capability* button.
3. In the *Select Capability Type* view, click *IQ: Audit and Quarantine*.
4. In the *Create IQ: Audit and Quarantine* view, configure the following settings;
    a. *Enable this capability* - Make sure the check box is selected to activate the Audit feature. The check box is selected by default.
    b. *Repository* - Select a specific proxy repository to evaluate, for example, maven-central.

c. *Quarantine* - Select the check box to quarantine any components that violate policy whenever you add new components to the selected proxy repository. This setting affects only components that are added to the repository after *Quarantine* is enabled. When a component is quarantined, the Nexus Repository Manager prevents it from being served from the proxy repository. The check box is deselected by default.

5. Click *Create capability* to save the new capability for Audit and Quarantine.

At this point, an audit of the selected repository is automatically started. Nexus Repository Manager contacts IQ Server and evaluates the components within the selected repository against any associated policy.

The results are displayed in *Repository Results*, which is described in the next section Understanding Repository Results.

Note

To successfully quarantine components when the Quarantine feature is enabled, the policy used to evaluate components must be configured to fail when policy violations occur at the proxy stage in the development lifecycle. If the policy is set to warn (rather than fail), the quarantining of components will not occur. For more information about setting policy and the proxy stage, see the Basic Policy Management chapter.

This screenshot needs to be added when next version of nxrm ships:

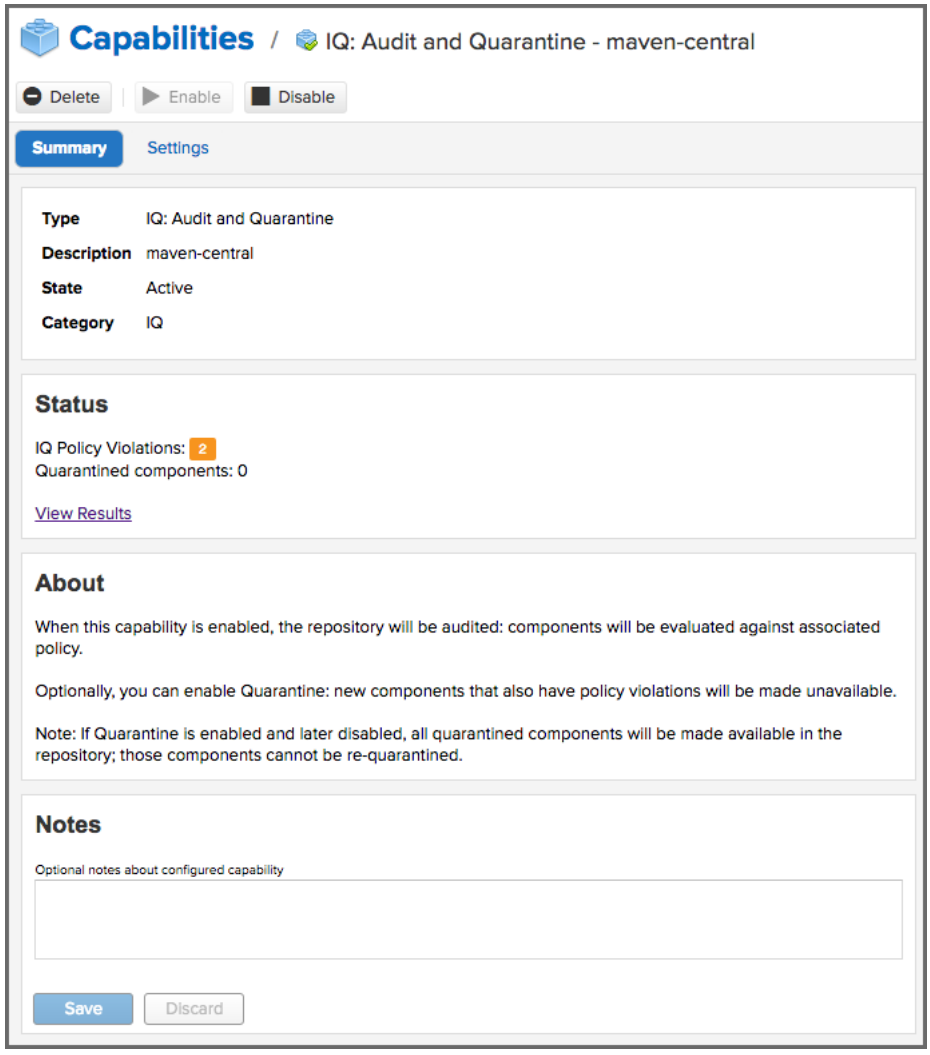


Figure 14.23: *IQ: Audit and Quarantine* Capability in Nexus Repository Manager

14.2.3.2 Disabling Audit and/or Quarantine

To disable Audit and/or Quarantine:

1. In Nexus Repository Manager, go to the *Administration* main menu and click *Capabilities* under

System.

2. Click the *IQ: Audit and Quarantine* capability for a specific repository.
3. To disable Audit, click the *Disable* button. Note that Quarantine is disabled as well.
4. To disable Quarantine only, deselect the *Enable Quarantine for Repository* check box.

Caution
When Quarantine is disabled, all quarantined components are made available for download from your proxy repository. This remains true, if you re-enable Quarantine. That is, any previously quarantined components are not quarantined again; only new components are evaluated for quarantine when you re-enable the Quarantine feature.

5. Click *Save* to save your changes or click *Discard* to discard them.

14.2.3.3 Releasing a Component from Quarantine

When a component is quarantined due to a violation, it is not available for download from the proxy repository. You must first resolve the violation(s) that caused the quarantine before releasing the component and making it downloadable. For information on resolving violations from labels, security vulnerabilities, or license issues, see the Application Composition Report chapter. For information on waiving policy violations, see the Waiving Repository Policy Violations section of this chapter. Once the violations are resolved, you can proceed with releasing a component from quarantine.

To release a component from quarantine:

1. In Nexus Repository Manager 3.x, go to *Repositories* on the *Administration* menu, and click the *IQ Policy Violations* count of an evaluated repository. This opens the Repository Results hosted on IQ Server.
2. Click the component you want to release from quarantine. This opens the *Component Information Panel* (CIP).
3. Click the *Policy* tab, and then click the *Release Quarantine* button.
4. In the confirmation box, click the *Release* button.

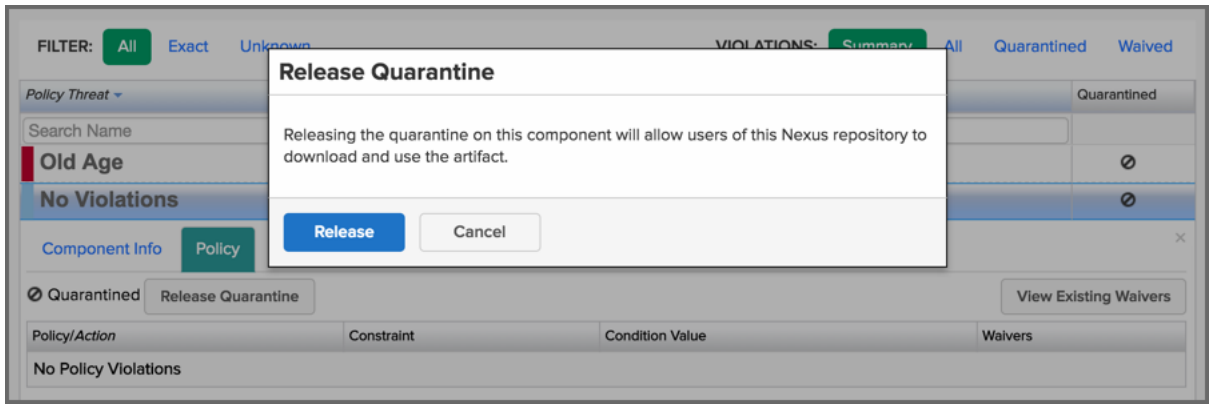


Figure 14.24: Release Quarantine

Note
Once a component is released from quarantine, it cannot be put back into quarantine even if it has subsequent policy violations. If you want to re-quarantine a component, you must delete the component from its repository. The component will be quarantined again if, during an audit, it violates a policy that is set to *Fail* at the Proxy stage.

14.2.3.4 Viewing Repository Results

Once the Audit is enabled, whenever you add a component to a proxy repository (or delete one), Nexus Repository Manager contacts IQ Server to evaluate the components within the proxy repository against any associated policy. The *IQ Policy Violations* are summarized in Nexus Repository Manager, and detailed in IQ Server.

In Nexus Repository Manager 3.x, the results of an audit are summarized in the *IQ Policy Violations* column of the *Repositories* view as shown in the figure below. You can access the *Repositories* view from the *Repository* sub menu of the *Administration* menu.

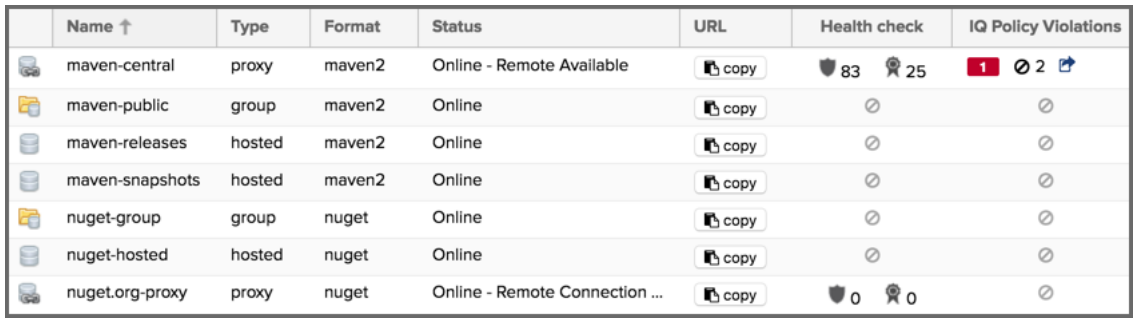


Figure 14.25: IQ Policy Violations Column in Nexus Repository Manager 3.x

The *IQ Policy Violations* column includes the following items:

- A count of components by their highest policy violation level.
- A count of quarantined components.
- A link to *Repository Results* on IQ Server.

The *IQ Policy Violations* column will also alert you if there are any errors in the audit and quarantine process. If there is an error, for example Nexus Repository Manager cannot communicate with IQ Server, a red exclamation mark will appear to the right of the *Repository Results* link along with text pertinent to the error that occurred. Additional information will be available in the Nexus Repository Manager logs.

Note

If the *IQ Policy Violations* column displays only *Audit Enabled* or *Quarantine Enabled*, then you do not have permission to view audit and quarantine summary results. For more information about this permission, see Granting Privileges to View Audit and Quarantine Summary Results later in this chapter.

If you have permissions to add capabilities in Nexus Repository Manager, you can also access *Repository Results* from the *Capabilities* submenu on the Administration menu:

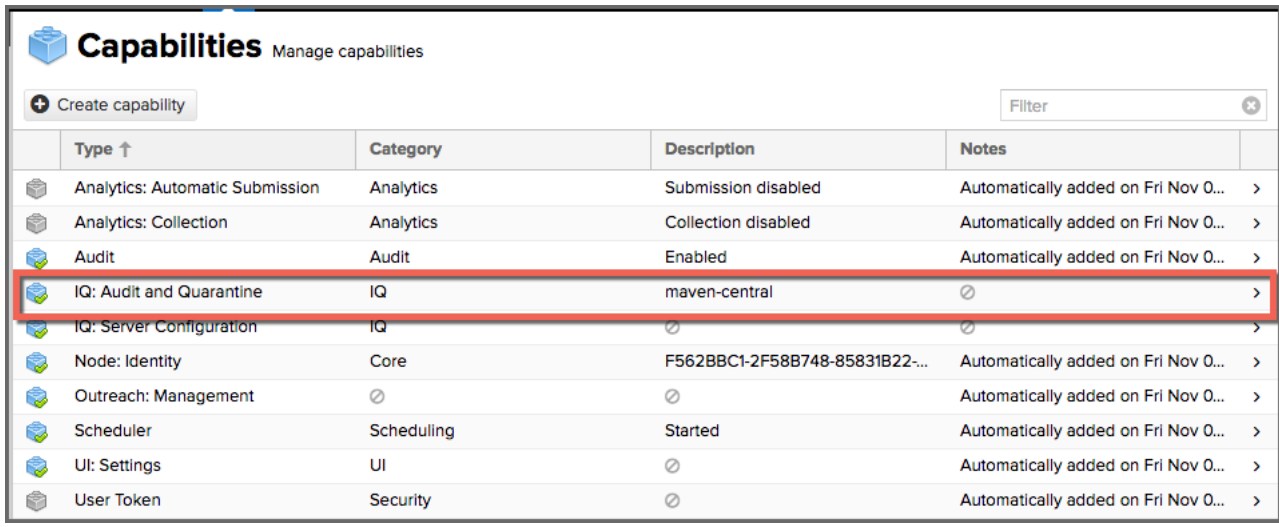


Figure 14.26: Nexus Repository Manager 3.x Capabilties Submenu

1. In the *Type* list of capabilities, click *IQ: Audit and Quarantine* for a specific repository.
2. In the *Capabilities / IQ: Audit and Quarantine* view, go to the Status section and click *View Results*.

To learn more about the details displayed in the *Repository Results*, see Understanding Repository Results in the section below.

14.2.3.5 Granting Privileges to View Audit and Quarantine Summary Results

In Nexus Repository Manager 3.x, the "nexus:iq-violation-summary:read" privilege allows you to view audit and quarantine summary results in the IQ Violations column of the Repository view. This privilege is assigned to the Nexus admin role by default. If users are assigned to custom roles, this privilege needs to be added to those roles in order for them to view audit and quarantine summary results.

To grant view privileges for audit and quarantine:

1. In Nexus Repository Manager 3.x, go to *Security* on the *Administration* menu and click *Roles*.
2. In the *Manage Roles* view, either create a new role or click to select an existing custom role.

3. If creating a new role, enter a *Role ID*, *Role name*, and *Role description.*
4. In the Privileges list, move the following privileges to the *Given* column:
   - nx-repository-view-*-*-read
   - nexus:iq-violation-summary:read
5. Save the role changes by clicking *Create Role* or *Save*.

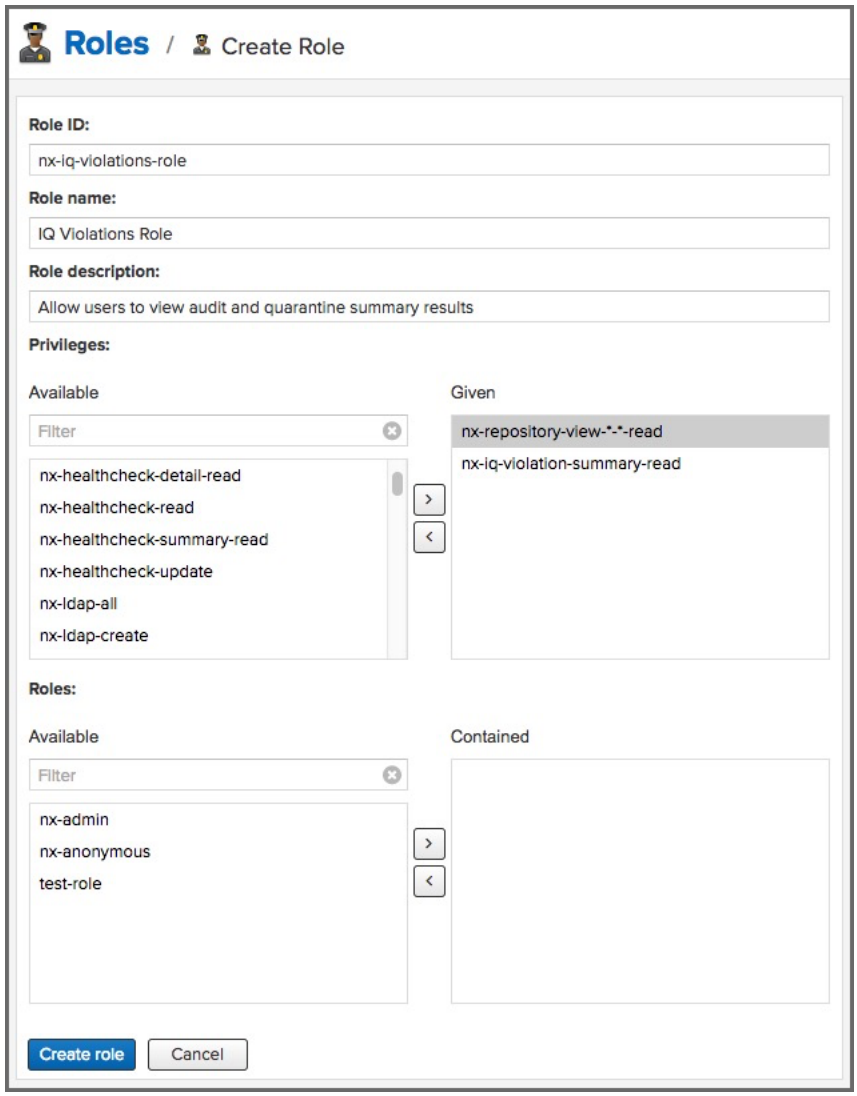


Figure 14.27: Granting Privileges to View Audit and Quarantine Summary Results

For information on assigning privileges, see the Privileges section in the Nexus Repository Manager 3.x book.

14.3 Understanding Repository Results

The *Repository Results* are displayed on IQ Server. They contain detailed information about policy violations and the components that violated those policies, as well as components that don't have violations.

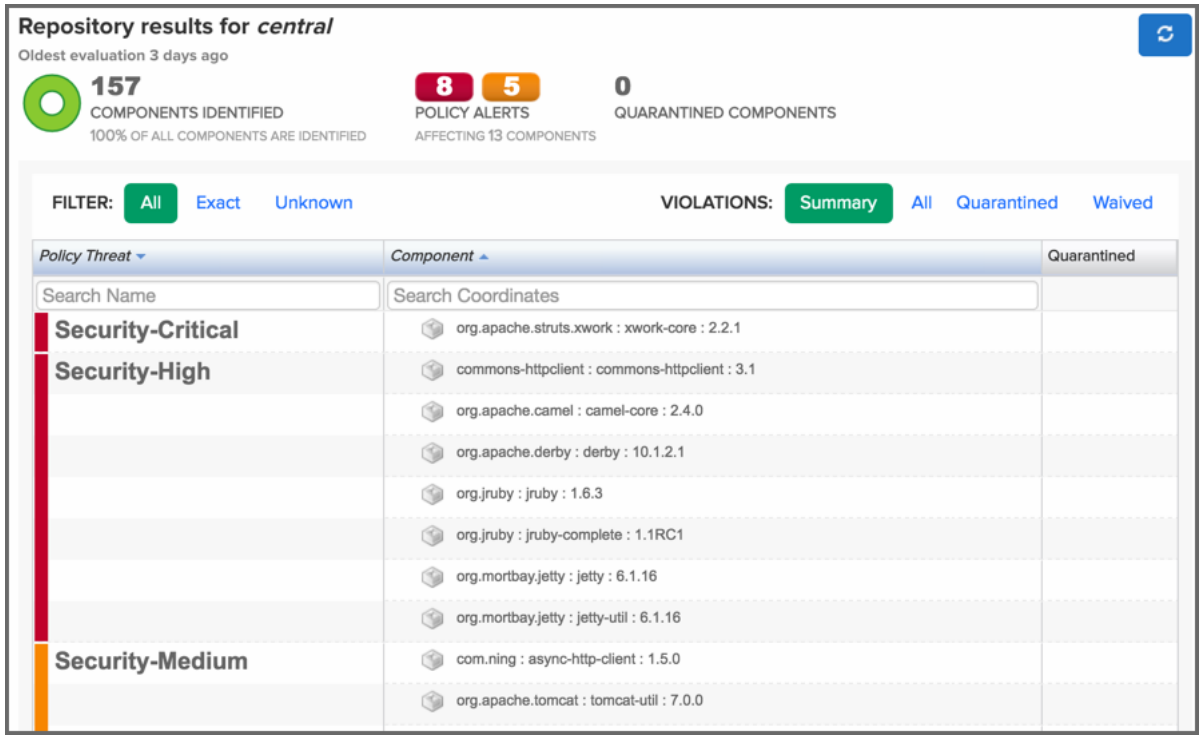


Figure 14.28: Repository Results

At the top of the *Repository Results* view (shown in the figure above) is a summary section with the following information:

- A count of components that were identified and scanned in the selected proxy repository.

- A percentage of scanned components that are identified.
- A count of policy violation alerts displayed by threat level.
- A count of components affected by policy violations.
- A count of quarantined components.

Below the summary section is a list of policy violations and the components that violated those policies. By default, this information is ordered by the highest policy threat level. You can refine the list using one of the following filter categories:

Filter

- *All* - Every component in the proxy repository.
- *Exact* - Components in the proxy repository that have an exact match to a component known to IQ Server.
- *Unknown* - Components in the proxy repository that have no exact match in IQ Server and cannot be identified.

Violations

- *Summary* - The most severe policy violation of each component.
- *All* - Every policy violation and the components that violated those policies. A component may appear more than once, if it violated multiple policies.
- *Quarantined* - Components that are prevented from being served by the proxy repository because they violate policy.
- *Waived* - Only policy violations that have been waived.

Note

You can update the audit results for the entire proxy repository by clicking the Re-evaluate Policy button in the upper right corner of the Audit View. This is useful especially after an associated policy is added or modified on IQ Server. However, it may take some time, if the repository is large.

During re-evaluation any previously quarantined components remain quarantined, no matter whether they still violate policy.

With quarantine enabled, if you delete a quarantined component, its quarantine status is also deleted. If you add the component back in, it is evaluated again just like any new addition to the repository. Currently the only way to remove a component from quarantine is to change the policy accordingly, then delete and add back the component.

Also, whenever you add or delete a component in the proxy repository, the audit results are automatically updated for the individual component only (not the entire repository).

14.3.1 Using the Component Information Panel (CIP)

When you click an individual component in the Repository Results, the Component Information Panel (CIP) opens with the *Component Info* tab displayed.

Component Info

This tab contains the same granular details about an individual component as the *Component Info* tab in Nexus Repository Manager. For an explanation of those details, see Component Info earlier in this chapter.

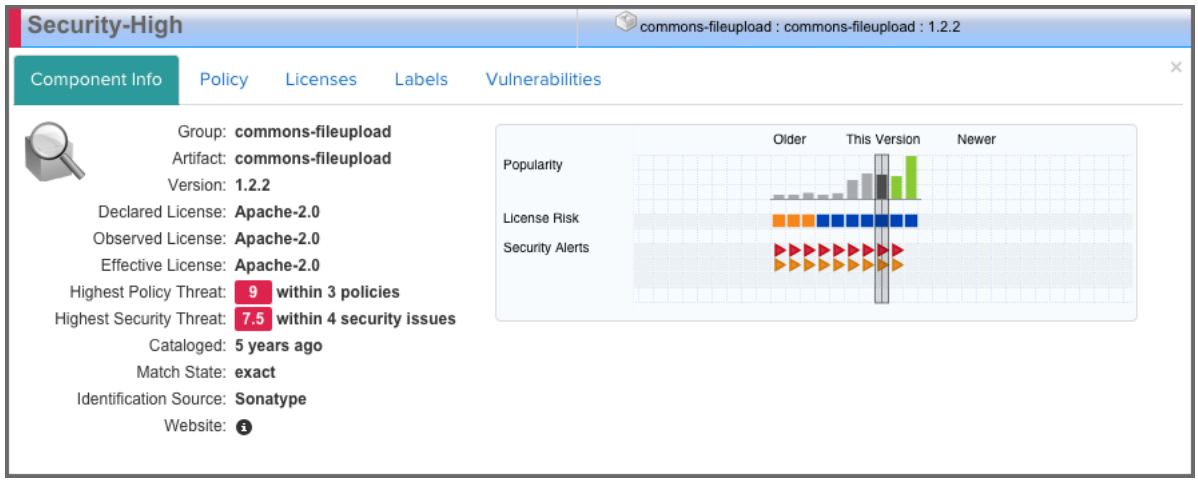


Figure 14.29: The Component Info Tab

Policy

The *Policy* tab displays all policies that were violated by a component. Here you can see the name of the policy that has been violated (and any action that was taken), the name of the constraint that has been violated, and the value that was found.

While the *Policy/Action* and *Constraint* names are straight forward, the *Condition Value* may be a little confusing at first. A condition is simply the *if* part of an *if/then* statement. *If* a certain condition value is found which is equivalent to a condition being met, *then* the policy will be violated. E.g. if we have a policy that has a condition such that if a security vulnerability is found, our Condition Value column would indicate, *Found x Security Vulnerabilities*. In the same regard, *Constraints* are simply multiple

conditions joined together.

The Policy Tab

image::figs/web/audit-view-policy.png

Licenses

The *Licenses* tab displays all *Effective* licenses, any licenses identified as declared by the author of the component, as well as any license found during the scan of the component source code. It also allows you to override the *Effective* license. To do this:

1. Select the *Scope* of the override
2. Select the *Status*
3. Select one, or more, of the *License(s)*
4. Optionally, but advised, provide a *Comment*
5. Click *Update*

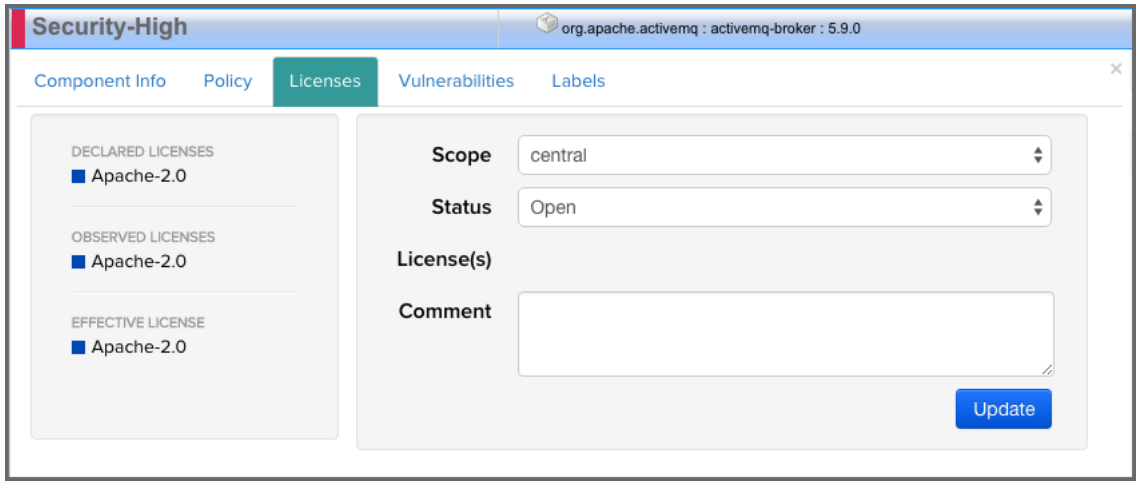


Figure 14.30: The Licenses Tab

Vulnerabilities

The *Vulnerabilities* tab displays all security vulnerabilities related to a component. The list of vulnerabilities is sorted by *Threat Level* from higher to lower risk. The *Problem Code* column displays unique identifiers obtained from security information web sites such as CVE and OSVDB. The *Info* button provides additional information about each security vulnerability. Lastly, the *Status* column tracks the state of your research regarding the vulnerability.

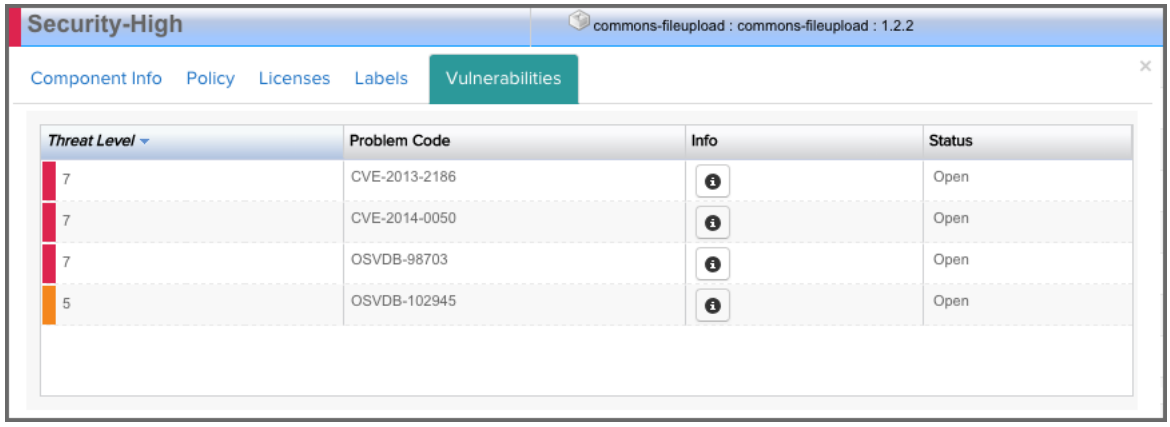


Figure 14.31: The Vulnerabilities Tab

If desired, you can change the security vulnerability status of a component in a proxy repository. This can help you keep track of your research when you investigate any security vulnerabilities identified by IQ Server.

To change the security vulnerability status of a component:

1. In the *Repository Results*, click a desired component to open the *Component Information Panel* (CIP).
2. Click the *Vulnerabilities* tab.
3. In the list of vulnerabilities on the left, click one to select it.
4. In the *Status* list on the right, select one of the following settings:
   - *Open* - The security vulnerability has not been reviewed; no research is under way.
   - *Acknowledged* - The security vulnerability is under review.
   - *Not Applicable* - The security vulnerability has been researched and deemed as having no effect on the repository.

- *Confirmed* - The security vulnerability has been researched and deemed as valid and applicable.

5. Click *Update* to save the changed setting.

Labels

The *Labels* tab displays any component labels that have been defined previously at the root organization level on IQ Server. Component labels are metadata that is assigned to a component within the context of a particular application or organization.

Figure 14.32: The Labels Tab

Assigning a Label

When assigning a label, you will only see labels defined on the root organization.

To assign a label:

1. Click a component you wish to assign a label to. The Component Information Panel (CIP) is displayed.
2. Click the *Label* option from the CIP menu. Two boxes are displayed:
   - The *Available* box on the left displays all labels.
   - The *Applied* box on the right displays labels that have been assigned to the component.
3. Click the button on the right side of a label to move it to the opposite side. You can hover over a label to view its description.

4. Click on the + button on the right side of a label in the *Available* list to assign the label to the component.
5. Click on the - button on the right side of a label in the *Applied* list to remove the label from the component.

When applying a label, you have the following options:

- Assign label for a repository
- Assign label for All Repositories
- Assign label for all within the Root Organization

14.3.2 Waiving Repository Policy Violations

Policy violations for components found in your repositories can be waived with a number of options for the scope and target of the waiver. As with all features, make sure to verify you have the appropriate level of access provided by the role you have been assigned.

Note
Waiving policy violations for components in your repository is different than waiving for an application. See Section 12.9.2 for additional information on waiving components at that level.

Waive Policy Violation

1. From within Nexus Repository Manager select a repository that has been evaluated.
2. Click the *IQ Policy Violations* count for a repository. This will open the Repository Results hosted on IQ Server.
3. Click a component that has a policy violation. This will expand the row to display the Component Information Panel (CIP).
4. Click the *Policy* tab within the CIP to display the current policy violations for the selected component.
5. Click the *Waive* button next to the policy violation you wish to waive.
6. A dialog is displayed with the following settings:

a. Determine the scope of the waiver:
   i. Repository *selected repository* [default]
   ii. All repositories
   iii. Organization *Root Organization* (This is displayed only if you have the appropriate level of access.)
b. Determine the targeted component of the waiver:
   i. Selected component *component name* [default]
   ii. All components
c. Comments - Add a brief note if desired.

7. Click the *Waive* button to complete the waiving process.

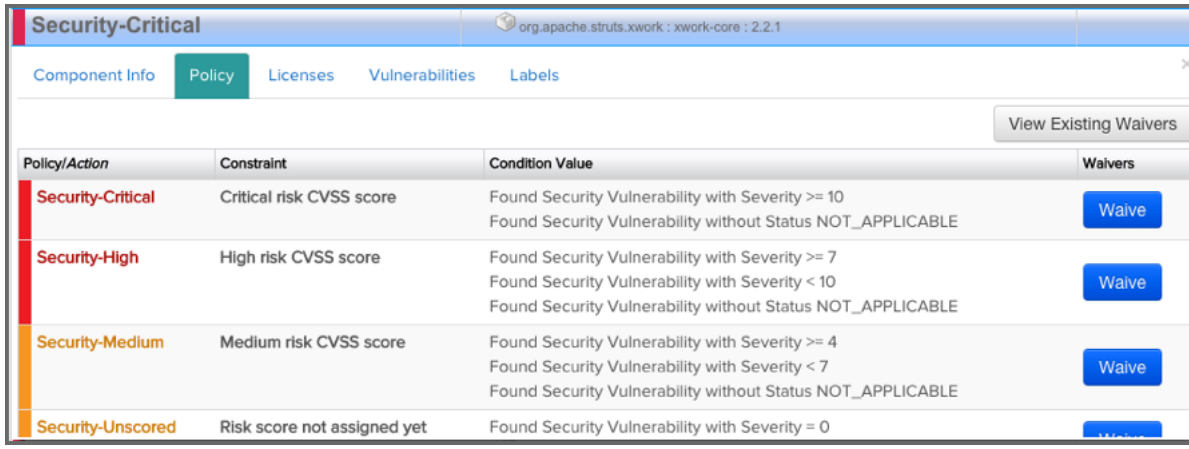


Figure 14.33: Waiving Policy Violations

View/Remove Existing Waivers

1. From within Nexus Repository Manager select a repository that has been evaluated by IQ Server.
2. Click the *IQ Policy Violations* count for a repository. This will open the Repository Results hosted on IQ Server.
3. Just above the list of components, you will see three options in the Violations filter. Click *Waived*, and then click one of the displayed components.
4. Click the *Policy* tab within the CIP to display the current policy violations for the selected component.

5. Click the *View Existing Waivers* button located above the list of policy violations. The Component Waivers dialog is displayed.
6. If you wish to remove a waiver, click the *Remove* icon (shaped like a minus sign). A confirmation dialog is displayed. Click the *Remove* button to remove the waiver.

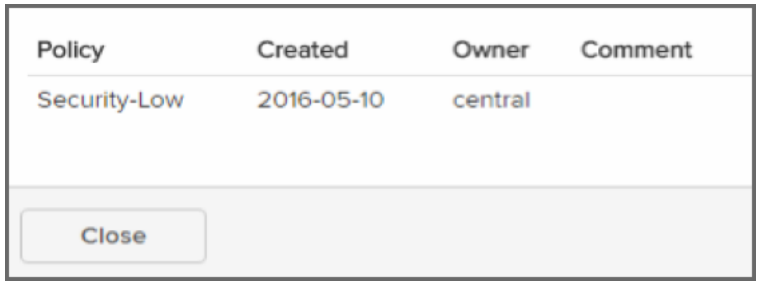


Figure 14.34: Waiving Policy Violations

Note
Waivers will not be applied until a re-evaluation of the Repository Results has occurred. This will occur automatically if the targeted component is left to the default settings (i.e. not set to All). In cases where the selected component is set to All, a manual re-evaluation will need to occur for any results previously applying the violation.

14.4 Managing Repositories

The creation, modification, and deletion of repositories is managed via Nexus Repository Manager. However, IQ Server also displays information about any connected repositories.

To view this information:

1. Click the *Organization & Policies* button located in the IQ Server toolbar.
2. Click on *Repositories*, located in the sidebar on the left side of the screen. The Configuration tab is displayed, as shown in the figure below.

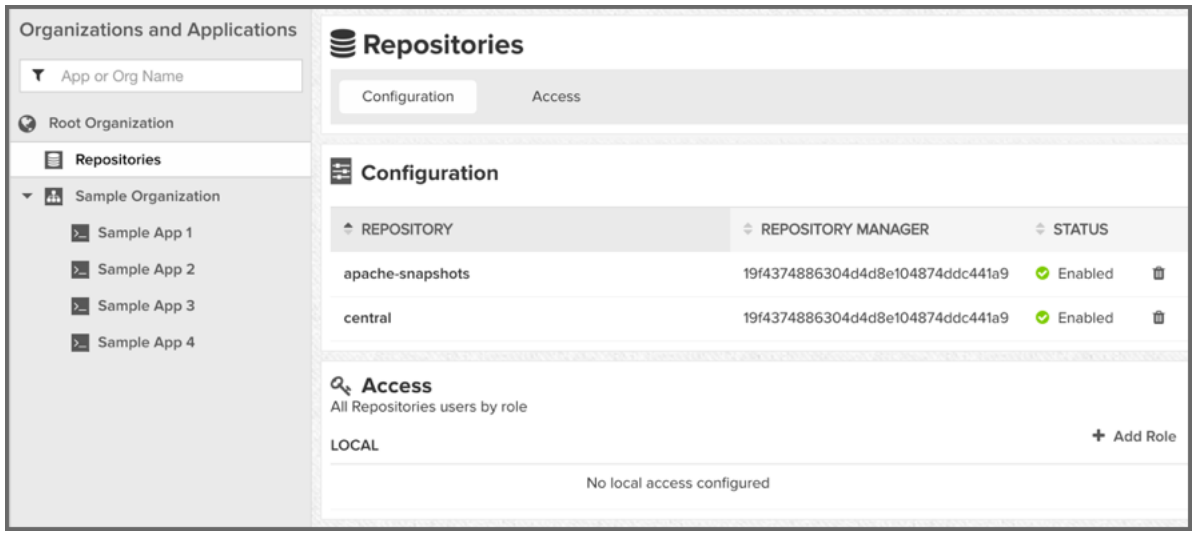


Figure 14.35: IQ Repositories

Details on repositories include:

- The public id of the repository
- The instance id of the Nexus Repository Manager hosting the repository
- The current audit-enabled state of the repository

Clicking the *Delete* button (shaped like a trash can) allows you to delete the repository after you confirm the deletion in a dialog.

Note
The deletion of a repository in IQ Server will NOT be replicated to Nexus Repository Manager.

14.5 Managing User Roles

The *Repositories* page, accessible from the sidebar of the **Organization & Policies** area, lets you adjust access settings for repository evaluation results. The process is the same as managing roles and permissions for organizations and applications on IQ Server. Through role assignments, you have the ability to grant users different permissions for repository evaluation results without granting them access to organizations and applications. For example, to grant a user the ability to view repository results, you assign the user to a role with View IQ Elements permission. To edit repository results, you assign the user to a role with Edit IQ Elements permission. The role assignments affect all repositories, not individual ones. For more information about assigning user roles, see Role Management in the Security Administration chapter.

Note
Any role assignments made at the Root Organization level are inherited automatically by *Repositories*. However, if you set a role in *Repositories*, the Root Organization is unaffected.

14.6 Removing a Repository in IQ Server

To remove a repository:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Click *Repositories* in the sidebar to display the Repositories page.
3. Locate the repository you want to remove in the *Configuration* section and click its *Remove Repository* icon (looks like a trash can).
4. In the *Remove Repository* confirmation box, click *Continue* to delete the entity, or click *Cancel* to keep it.

This action affects only IQ Server, not Nexus Repository Manager. While the repository entity and its data are permanently removed from IQ Server, the repository in Nexus Repository Manager remains unchanged.

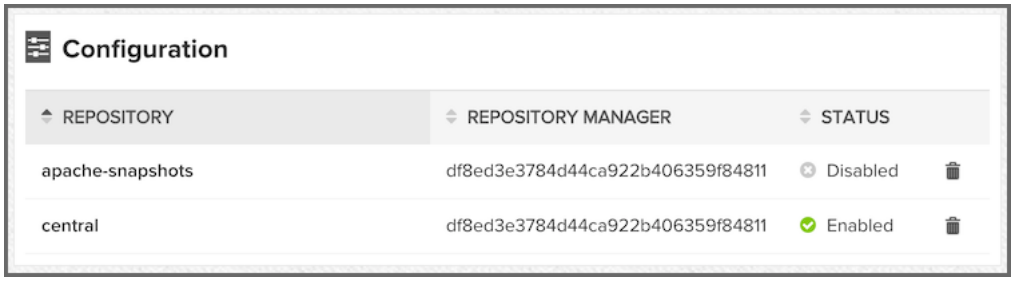


Figure 14.36: The Configuration Section

Chapter 15

Sonatype CLM and Continuous Integration

The idea of continuous integration is that software development efforts are much easier to manage when test failures and other bugs can be identified closer to the time they were introduced into a complex system. As a consequence the differences between the working and the failing system are smaller and therefore easier to detect.

The terms continuous integration was coined by Martin Fowler and Kent Beck in their book *Extreme Programming Explained* published 1999. They introduced the idea of creating a system that continuously builds your software and executes any tests against it on a regular base as well, all in response to any changes of the source code.

Since its introduction, usage of continuous integration servers became an established and well understood best practice across the entire software development industry.

A number of commercial as well as open source servers are now available for installation in your own infrastructure as well as a managed service running remotely. Typical CI installations are often comprised of a a number of servers running the actually build and being orchestrated by one master and build running on the CI servers range from simple compile builds to running large integration test suites or regression tests in an automated fashion. In addition CI servers are increasingly used for continuous deployment, where a series of successful build and test runs results in actual production deployment of the software.

Sonatype CLM can analyze the components used in your software development for security and license

characteristics. When integrated with a continuous integration server it becomes a dynamic analysis performed on a regular basis occurring potentially with each build running on the server.

Depending on your application server software you can use the Sonatype CLM Hudson and Jenkins, Bamboo, the CLI or Maven.

Every one of these tools allows you to perform a full security and license analysis of the artifacts produced by the configured build backed by your Sonatype CLM server. It will provide you access to the analysis report.

Chapter 16

Nexus IQ for Bamboo

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle license.

Getting Nexus IQ for Bamboo up and running is not difficult. However, there are a few things we expect you to have completed prior to getting started.

- Install and configure the IQ Server
- Create an organization, and at least one application
- Evaluated the application at least once

If anything in the list above looks completely new, or has not been completed, bring everything to a full stop. Even if you aren't responsible for those areas of IQ Server, you will need to have them complete before configuring Nexus IQ for Bamboo.

Don't feel bad, we know you just wanted to get started as quickly as possible. To get the most out of it however, you will want to start with the following chapters:

- IQ Server Installation and Configuration chapter

- Policy Management chapter.

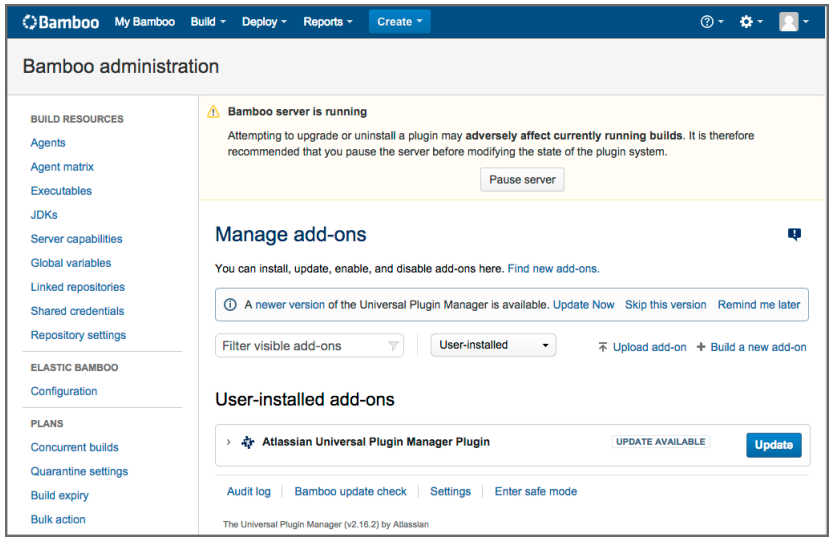
You may want to thumb through our Application Composition Report chapter as well.

Yeah, that's a lot of reading, but then again we wrote it just for users like you. Go ahead, we promise it will be worth it. If you do, we know you will be better prepared to extract the most value out of your Nexus IQ for Bamboo installation.

Once you are ready, head on over and download Nexus IQ for Bamboo from our Nexus IQ Server downloads page.

16.1 Install Nexus IQ for Bamboo

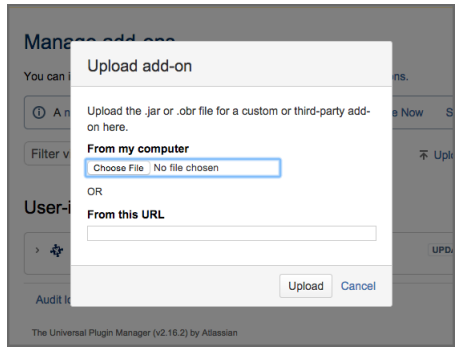
You should have the Nexus IQ for Bamboo file downloaded. Now would be a good time to double-check the location you saved it to. That's something easy to forget. Got it? Good. Now, follow these steps:



Caution
Nexus IQ for Bamboo has been rebranded in version 1.0.7 and as such has a new distribution name. When upgrading to 1.0.7 it is important you first uninstall Sonatype CLM for Bamboo before continuing with the installation.

1. First, access Bamboo's *Manage Add-ons* by clicking the *Gear* icon (within Bamboo) and then clicking *Add-ons* from the drop down list.
2. Next, click the *Upload Add-on* link. A modal will display, allowing you to specify a file, or a URL location. In this example, we'll be using the file you downloaded previously.
3. Choose the location of the Nexus IQ for Bamboo file, click *Open*, and then the *Upload* button. The upload only takes a few seconds, but during this time, a progress modal will display.
4. Upon successful upload, a confirmation modal will display giving you a bit of information about the plugin. Click *Close*.

Awesome job! If everything went as planned, Nexus IQ for Bamboo is now listed under *User-installed add-ons*. Let's head to the next step, configuration.



Tip
In most cases, pausing your Bamboo server is a good idea. Also, if you ever decide to uninstall Nexus IQ for Bamboo, you can do it from the *Manage add-ons* area as well.

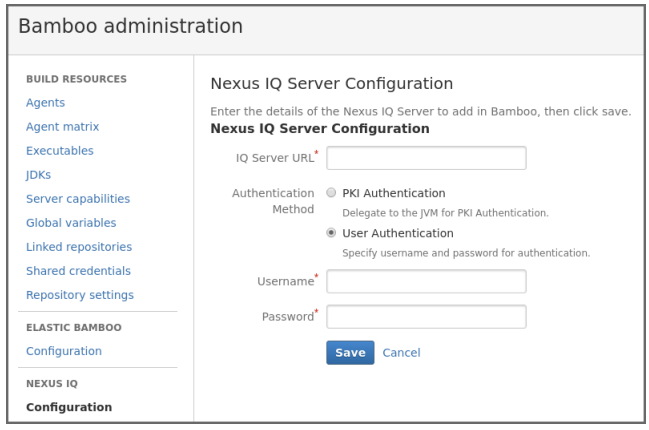
16.2 Configure Nexus IQ for Bamboo

Ready to configure Nexus IQ for Bamboo? Of course you are, but if you just joined us, we need to check and make sure you've installed it.

You have? Perfect!

Now you are ready to configure Nexus IQ for Bamboo. You should be in the Bamboo administration area.

1. In the left hand navigation area / menu, locate a section titled *Nexus IQ*. In that section, click on the *Configuration* link to open the Nexus IQ Server Configuration window.



2. Enter the IQ Server URL - the URL for your IQ Server.
3. Select an Authentication Method:
    a. **PKI Authentication**: Delegate to the JVM for authentication.
    b. **User Authentication**: Enter a username and password for authentication.

> **Tip**
> We recommend that you create a unique machine account that has access to the application(s) you wish to link to your Bamboo Build(s)/Plans.

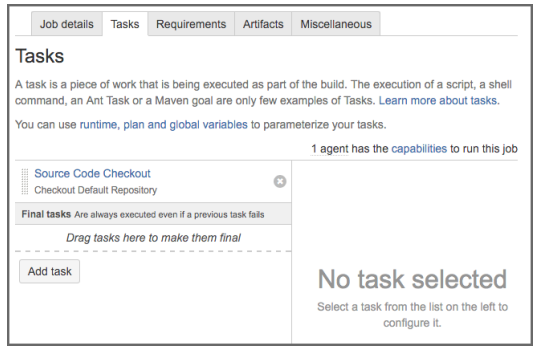
4. Click the *Save* button. Your configuration is saved, displaying the application(s) the user has access to.

16.3 Adding the IQ Analysis Task

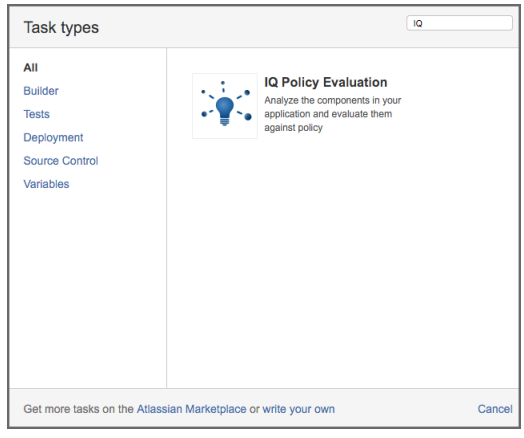
So now it's time to put everything you did to install and configure Nexus IQ for Bamboo to good use, and add an IQ Analysis Task.

The IQ Analysis Task is available once you've installed and configured Nexus IQ for Bamboo. The following steps will walk you through adding this new task to a job.

1. After navigating into a Bamboo Project > Plan > Stage > and then Job, click on the *Add task* button.



2. A modal will display offering a list of *Task Types*. The IQ Analysis Task is listed in the *Test* type, or you can simply use search.



3. Enter the following information:

Task Description
A simple description to remember what the task does.

Application

The list of Applications corresponds to the account used during Nexus IQ for Bamboo configuration. Remember, this is the Application containing the policies that components in the build will be evaluated against.

Fail build when IQ Server is unable to evaluate

Check this option if you want to fail the build when an IQ evaluation can't be performed. Once checked, if for any reason the evaluation is not generated, the build will be failed. An example of this might be if the IQ Server is inaccessible. In the same example, but where the *Fail the build* option is left unchecked, the build would continue as it would have normally.

Tip

In any case where the IQ Server is unable to evaluate an application, details are provided in the job/build-specific log.

Stage

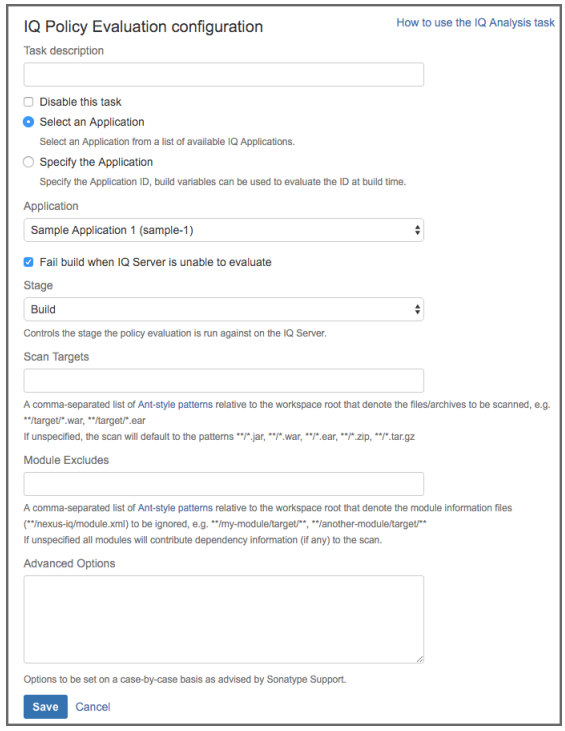
This corresponds to the stage you wish the policy evaluation of the application/project to be run against. Additionally, this will correspond to the stage location when viewing report information via the IQ Server. For example, if you chose the Build stage, summary and dashboard violation results will be displayed accordingly.

Scan Targets

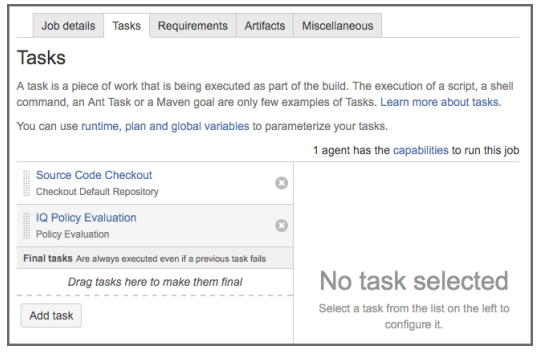
The scan targets setting allows you to control which files should be examined with an Apache Ant styled pattern. The pattern is relative to the project workspace root directory and inherits the global configuration.

Module Excludes

If you are using the Sonatype CLM for Maven plugin, module files are created, and can contribute to results found during an evaluation. For information on how to exclude these files, please see the Excluding Module Information Files in Continuous Integration Tools of the Sonatype CLM for Maven chapter.



4. Click the *Save* button.

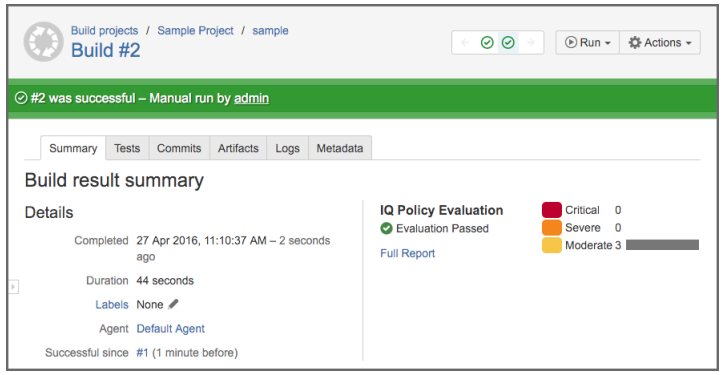


16.4 Reviewing IQ Policy Results

After your Bamboo job has completed, and your application has been successfully evaluated by the IQ Server, a summary of the results will be provided on the Job Summary page. The summary results give a breakdown of the three threat level categories for policy:

- Critical (8-10)
- Severe (4-7)
- Moderate (2-3)

In addition to counts for each of these categories, a status for the success of the evaluation is provided, as well as a link to the *Full Report* located on your IQ Server is also provided. These are located just to the left of the summary results.



Tip
In the event IQ should encounter an issue during the evaluation related to the IQ Server itself, this will be indicated by one of three statuses: *Passed*, *Passed with Warnings*, or *Failed*.

Chapter 17

Nexus IQ for Hudson/Jenkins

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle license.

Eclipse Hudson and Jenkins are powerful and widely used open source continuous integration servers providing development teams with a reliable way to monitor changes in source control and trigger a variety of builds.

17.1 Plugin Selection

IQ Server has two integrations for Jenkins: Nexus IQ for Jenkins 2.x and Nexus IQ for Hudson/Jenkins 1.x. A general suggestion is to use Nexus IQ for Jenkins 2.x unless you have a specific need detailed below:

Select Nexus IQ for Jenkins 2.x if you are:

- Using Jenkins 2.x.
- A new IQ for Jenkins user.

- Using Jenkins pipelines.
- Using the Jenkins Credentials plugin.

Select Nexus IQ for Hudson/Jenkins 1.x if you are:

- Using Jenkins 1.x.
- Using Hudson.
- An existing IQ for Jenkins user that does not use Jenkins pipelines and does not want to migrate configuration.
- Reliant on the CLM Maven plugin's index goal and the IQ Server for Hudson/Jenkins module analysis feature.

Both plugins are available for download from Sonatype Support.

17.2 Integrating Nexus IQ for Hudson/Jenkins 1.x

Nexus IQ for Hudson/Jenkins 1.x evaluates the project workspace after a build for all supported component types, creates a summary file about all the components found and submits that to the IQ Server. The IQ Server uses that data to produce an analysis with the security and license information and send it back to the CI server. It will then use these results to render the analysis reports.

The file types supported for analysis are in tar/zip like format with the extensions tar, tar.bz2, tb2, tbz, tar.gz, tgz and zip or in Java archive formats of the type jar, ear, war, hpi, wsr, har, sar, rar, mar and nbm.

Historically the Hudson project and community split into two groups, with Jenkins as well as Hudson emerging as sibling products with a different focus going forward while sharing a common API for plugins. In general, with regard to the IQ for Hudson/Jenkins functionality, the interaction will be near identical, with only a few differences, which are inherent to Hudson and Jenkins, and not IQ Server.

17.2.1 Installation

Nexus IQ for Hudson/Jenkins 1.x is distributed as a Hudson plugin package (`.hpi` file) and is compatible with Jenkins and Hudson. Download the plugin from Sonatype Support.

In order to install the plugin, log into Jenkins or Hudson as an administrator and then select *Manage Jenkins/Manage Hudson* to get to the global configuration menu displayed in Figure 17.1 in the Jenkins look. The Hudson look will be similar in content, yet different in colors and styling.

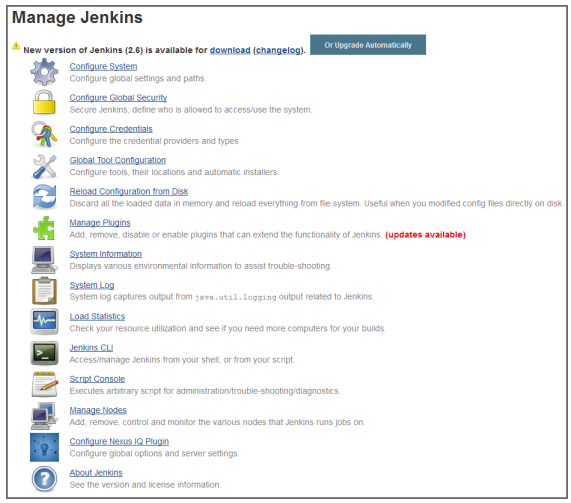


Figure 17.1: Jenkins Global Configuration Menu

From the displayed configuration menu, select *Manage Plugins* and in the plugin management section, choose the *Advanced* tab.

The advanced plugin management allows you to upload a plugin distribution file (`.hpi`) in the section entitled *Manual Plugin Installation* on Hudson and *Upload Plugin* on Jenkins. Click *Choose File* and select Nexus IQ for Hudson/Jenkins 1.x hpi file named `nexus-iq-jenkins-plugin-x.y.z.hpi` with `x.y.z` representing a version number like `2.11.2` in the file selection dialog. Then click the *Upload* button. Once the plugin has been uploaded to the server, you need to restart your continuous integration server.

17.2.2 Global Configuration

After a successful installation of Nexus IQ for Hudson/Jenkins 1.x, a new option will be available in the Jenkins/Hudson management area, *Configure Nexus IQ Plugin*. Follow these instructions to configure Jenkins or Hudson to connect to your IQ Server.

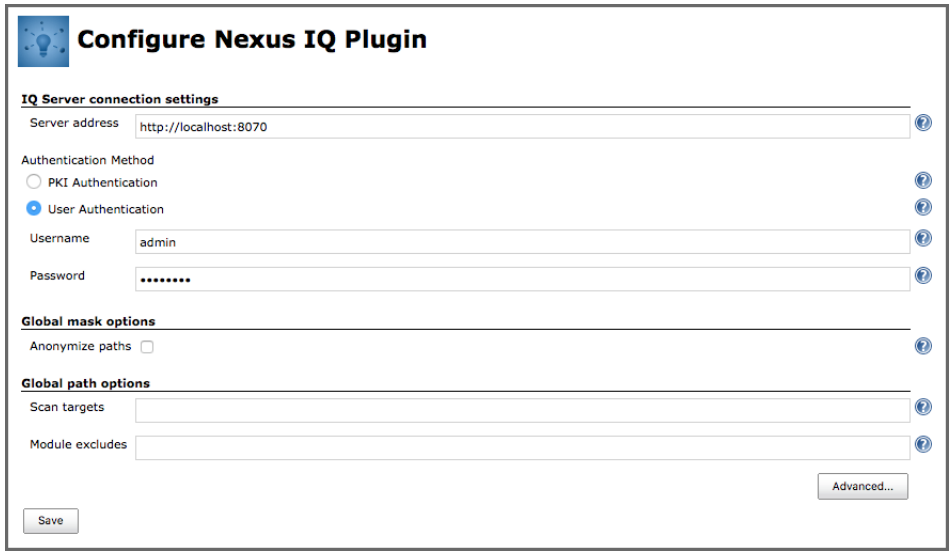


Figure 17.2: Global Configuration of Nexus IQ for Hudson/Jenkins 1.x

IQ Server settings (required)

Server address

This is the address for the IQ Server as it can be reached from the Jenkins/Hudson server. By default, the IQ Server address is `http://localhost:8070`.

If your IQ Server is behind a proxy server for serving HTTPS or other reasons, you have to use the public URL as it is reachable from the continuous integration server. Only the master Jenkins/Hudson server connects to the IQ Server and you therefore only need to ensure connectivity in terms of open firewall ports and proxy server settings between the master CI server and the IQ Server.

Authentication Method

Select an authentication method:

1. Select *PKI Authentication* to delegate authentication to the JVM.
2. Select *User Authentication* to specify a username and password:
   a. Username: Enter the username you wish to connect to the IQ Server.

> **Tip**
> Since these settings will be used across all projects for your Jenkins/Hudson installation, we suggest creating a single account on IQ Server, and then associating that account with the Application Evaluator role for the organizations or applications you will be linking to Nexus IQ for Hudson/Jenkins 1.x.

b. Password: Enter the password for the username entered above.

> **Tip**
> Username and password can also be configured per job.

Global mask options

Anonymize paths
Enabling this feature will anonymize all paths before data is sent to the IQ Server. Ultimately, this prevents the Application Composition Report from reporting the locations/occurrences of components.

Global path options

Scan targets
The scan targets setting allows you to control which files should be examined. The configuration uses an Apache Ant styled pattern, is relative to each project's workspace root directory, and has a useful default setting that includes all `jar`, `war`, `ear`, `zip` and `tar.gz` files. The default value is therefore

```
**/*.jar, **/*.war, **/*.ear, **/*.zip, **/*.tar.gz
```

> **Note**
> This default only applies if, and only if, neither global nor job config specify scan targets. Adding to this, if you are using a private Maven repository, our default pattern will include your entire Maven repo. This could greatly increase the time necessary for your evaluation, as well as skew evaluation results. To avoid this, consider using a more specialized pattern like **/target/*.jar.

Module excludes
If you are using Sonatype CLM for Maven, you may have noticed the creation of module information files. The process for excluding modules is documented in the Excluding Module Information Files in Continuous Integration Tools section of the Sonatype CLM for Maven chapter.

Advanced options
A number of additional parameters can be supplied to the plugin using this input field. Typically these parameters will be determined by Sonatype support.

17.2.3 Job Configuration

After a completed installation (see Section 17.2.1) and global configuration (see Section 17.2.2) of Nexus IQ for Hudson/Jenkins 1.x, you are ready to configure an invocation as part of a specific job.

Depending on your job type it will be available as a pre and/or post-build step as well as an invocation as a main build step. A pre-build step or a main build step executed before your main build invocation step could be used to examine components existing in the workspace or being placed into the workspace by an earlier build step.

The typical invocation would be as main build step, after the package that should be examined has been created. An example configuration from Jenkins is displayed in Figure 17.3.

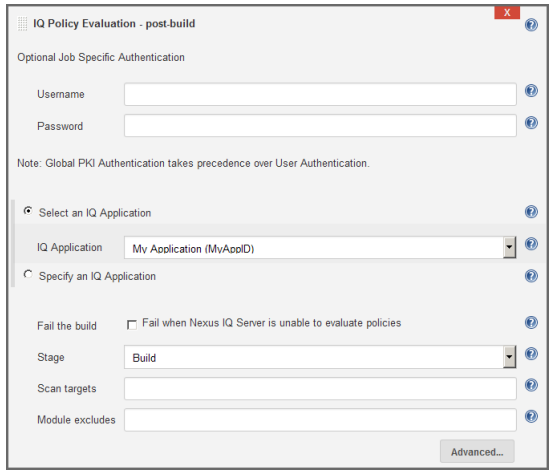


Figure 17.3: Build Scan Configuration for a Build Step

The configuration options for Nexus IQ for Hudson/Jenkins 1.x invocations mimic the parameters from the global configuration described in Section 17.2.2 and are appended to the global parameters. The configuration parameters are:

Optional Job Specific Authentication

While username and password can be configured globally, in some cases you may want a certain job to be associated with a user who has permissions to specific organization and/or applications. Job Specific Authentication allows you to configure a user for this job and use the associated permissions to select the application for the evaluation.

Username

The IQ Server username you wish to use for this job.

Password

The password for the username above.

Note
When configuring job specific authentication, please note that global PKI Authentication takes precedence over User Authentication.

Depending on what application is used, the policies associated to the application will be used for the analysis of this build job output. There are two options for choosing what IQ application to associate with the build:

Select an IQ Application
The IQ application dropdown will be populated with the names of applications based on the permissions for the configured user name and password.

Specify an IQ Application
If you want to use a build variable to provide the IQ Application ID, you can enter it in the field displayed after selecting this option. Click on the help icon to the right of the field for information on using build variables (e.g. ${THE_APPLICATION_ID}) to evaluate the application at build time.

Fail the build
Check this option if you want to fail the build when a policy evaluation can't be performed. Once checked, if for any reason the evaluation is not generated, the build will be failed.

An example of this might be if the IQ Server is inaccessible. In this scenario, the build would fail. In the same example, but where the *Fail the build* option is left unchecked, the build would be marked unstable.

Stage
This corresponds to the stage you wish the policy evaluation of the application/project to be run against. Additionally, this will correspond to the stage location when viewing report information via the IQ Server (e.g. if you chose the Build stage, summary and dashboard violation results will be displayed accordingly).

Note
Depending on how your policies are configured, this may impact warning and fail actions.

Scan targets
The scan targets setting allows you to control which files should be examined with an Apache Ant styled pattern. The pattern is relative to the project workspace root directory and inherits the global configuration.

Module excludes
You can exclude modules from being scanned with module information files configured in this setting. The default value is inherited from the global configuration.

Advanced options
: A number of additional parameters can be supplied to the plugin using this input field. Typically these parameters will be recommended to you by the Sonatype support team.

17.3 Integrating Nexus IQ for Jenkins 2.x

Nexus IQ for Jenkins 2.x evaluates a project workspace for all supported component types, creates a summary file about all the components found, and submits that to the IQ Server. The IQ Server uses that data to produce an analysis with security and license information and sends it back to the Jenkins server. These results are then used to render analysis reports.

Note
Nexus IQ for Jenkins 2.x is only compatible with Jenkins versions 2.x and above.

17.3.1 Installation

Install Nexus IQ for Jenkins 2.x using the following steps: . Login to Jenkins as an administrator. . Select *Manage Jenkins* from the left-navigation menu. . Select *Manage Plugins* from the list of configuration options. . In the *Plugin Manager* window, select the *Available* tab and enter *nexus platform plugin* in the *Filter* search box. . Select the *Install* checkbox next to *Nexus Platform Plugin* and then click the either the *Install without restart* or 'Download now and install after restart* button:

Figure 17.4: Nexus Jenkins Plugin Installation

A message displays on the screen when Nexus IQ for Jenkins 2.x is successfully installed.

17.3.2 Global Configuration

Use the following instructions to configure Jenkins to connect to your IQ Server:

1. Select *Manage Jenkins* from the left-navigation menu.
2. Select *Configure System* from the list of configuration options.
3. In the *Sonatype Nexus* section, select *Nexus IQ Server* from the *Add Nexus IQ Server* dropdown menu and then enter the following:
    a. **Server URL**: The location of your IQ Server.
    b. **Credentials**: Select the *Add* button to enter your IQ Server username and password using the *Jenkins Provider Credentials: Jenkins* modal window. Once added, select your IQ Server username and password from the *Credentials* dropdown list and click the *Test Connection* button.

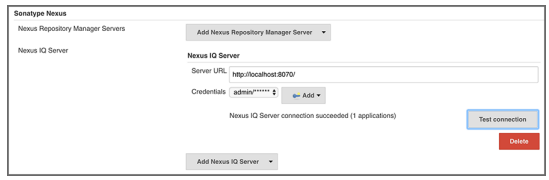


Figure 17.5: Nexus Jenkins Plugin Global Configuration

4. After a successful connection to IQ Server, click the *Save* button.

Note
Only one IQ Server instance can be configured.

17.3.3 Job Configuration

After a completed installation and global configuration of Jenkins, you are ready to configure a build-step invocation as part of a specific job.

17.3.3.1 Freestyle or Multi-Configuration Projects

The freestyle build job is a flexible and configurable option, and can be used for any type of project. A multi-configuration build job should be used as a parameterized build job that automatically runs with all the possible acceptable combinations of parameters.

Use the following steps to add a Nexus Policy Evaluation build step to a freestyle or multi-configuration build:

1. In the *Build* section of the project configuration screen, click the *Add Build Step* dropdown button and then select *Nexus Policy Evaluation*. Enter the following parameters:
   a. **Stage**: Select *Build*, *Stage Release*, *Release*, or *Operate*. This controls the stage the policy evaluation is run against on the IQ Server. Only the stages you are licensed to appear in the list.

      Note
      Depending on how your policies are configured, this may impact warning and fail actions.

   b. **Application**: Select an application from the list of available IQ Server applications. This determines the policy elements (policies, labels, and license threat groups) to associate with this build and is managed via the IQ Server.
   c. **Advanced options**: A number of additional parameters can be supplied to the plugin using this input field. Typically these parameters are determined by Sonatype support.

Figure 17.6: Nexus Policy Evaluation

2. Complete your freestyle or multi-configuration build as desired and click *Save*.

17.3.3.2 Pipeline Projects

Jenkins Pipeline is a suite of plugins that support implementing and integrating continuous delivery pipelines into Jenkins.

For IQ Server, build pipelines allow for policy evaluation at any point during the build, providing a way to gain a bill of materials of components that may not exist during final delivery. In addition, this allows for a policy gate to be set anywhere along the build and delivery process.

Use the following steps to add a Nexus Policy Evaluation build step to a pipeline build:

1. In the *Pipeline* section of the project configuration screen, click the *Pipeline Syntax* link.

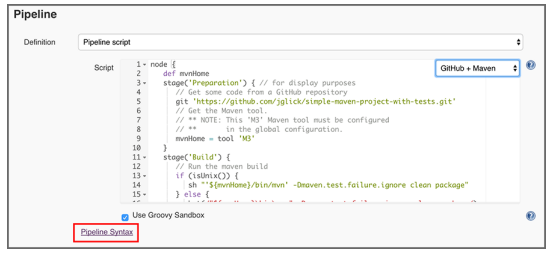


Figure 17.7: Nexus Jenkins Plugin Pipeline Syntax

2. In the *Steps* section of the *Snippet Generator* window, select the following:

   a. **Sample Step**: Select *NexusPolicyEvaluator: Nexus Policy Evaluation*.

   b. **Stage**: Select *Build*, *Stage Release*, *Release*, or *Operate*. This controls the stage the policy evaluation is run against on the IQ Server. Only the stages you are licensed to appear in the list.

   Note
   Depending on how your policies are configured, this may impact warning and fail actions.

   ---

   c. **Application**: Select an application from the list of available IQ Server applications. This determines the policy elements (policies, labels, and license threat groups) to associate with this build and is managed via the IQ Server.

   d. **Advanced options**: A number of additional parameters can be supplied to the plugin using this input field. Typically these parameters are determined by Sonatype support.

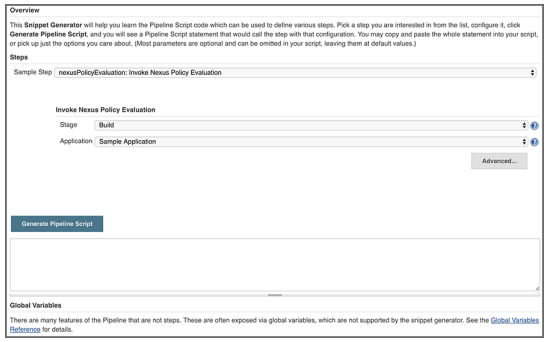


Figure 17.8: Generate Pipeline Script

3. Click the *Generate Pipeline Script* button.
4. Copy the generated script and paste it into the desired stage of your pipeline script.

   An example pipeline script is shown below:

```
nexusPolicyEvaluation failBuildOnNetworkError: false, iqApplication: ' ↩
    SampApp', iqStage: 'build',
jobCredentialsId: ''
```

5. Complete your pipeline build as desired and click *Save*.

17.3.3.3 Return Value from Pipeline Build

Utilizing the Nexus IQ for Jenkins 2.x plugin provides full component intelligence and the ability to run policy against your application. Jenkins pipelines let you invoke a build step as part of the build process and use the results for complex workflows. For example, objects returned from a build pipeline step can be used to provide feedback in-application about the process of an evaluation. In the following example, we'll setup a pipeline build that uses results of a policy evaluation to push data to a GitHub pull request:

First, start a new Jenkins pipeline job for a Maven build. Next, add a build stage that calls Maven clean package, wrapped in a few GitHub status updates. For this example, we're going to send an update pending when we start the build and have it update the status to indicate if the build succeeded:

```
stage('Build') {
  postGitHub commitId, 'pending', 'build', 'Build is running'
  sh  "${mvnHome}/bin/mvn clean package"

  if (currentBuild.result == 'FAILURE') {
    postGitHub commitId, 'failure', 'build', 'Build failed'
```

```
    return
  } else {
    postGitHub commitId, 'success', 'build', 'Build succeeded'
  }
}
```

Next, we'll add a small helper that sends a cURL to the GitHub API to update the status. To do this, set and declare the `gitHubApiToken` and `project` variables and then create the following function:

```
def postGitHub(commitId, state, context, description, targetUrl) {
  def payload = JsonOutput.toJson(
    state: state,
    context: context,
    description: description,
    target_url: targetUrl
  )
  sh "curl -H \"Authorization: token ${gitHubApiToken}\" --request POST -- ↩
      data '${payload}'
  https://api.github.com/repos/${project}/statuses/${commitId} > /dev/null ↩
      "
}
```

After the build, we need to run a Nexus Lifecycle Analysis to determine what the policy evaluation results are. This will let GitHub know Nexus Lifecycle Analysis is running and then do the policy evaluation. The easiest way to do this is using the Jenkins Pipeline Syntax helper to generate a Groovy script.

Click the *Pipeline Syntax* link on your build configuration screen to open the Pipeline Syntax helper. Select "NexusPolicyEvaluator" as the sample step, select the stage and application you'd like to evaluate against, and then click "Generate Pipeline Script." Copy the script and paste it into your pipeline script.

In addition to calling the evaluation, you will want to do something with the evaluation results. To do this, set a `policyEvaluationResult` variable to the result. Like with the build, you'll want to look whether the build result is a failure or not to determine how to update your GitHub pull request. You can also add a link to the Application Composition Report. Having this available in your pull request lets you go directly to the evaluation, see why it failed, and perform remediation. An example Nexus Lifecycle Analysis script is shown below:

```
stage('Nexus Lifecycle Analysis') {
  postGitHub commitId, 'pending', 'analysis', 'Nexus Lifecycle Analysis is ↩
       running'

  def policyEvaluationResult = nexusPolicyEvaluation  ↩
      failBuildOnNetworkError: false, iqApplication: 'webgoat',
  iqStage: 'build', jobCredentialsId: ''
```

```
  if (currentBuild.result == 'FAILURE'){
    postGitHub commitId, 'failure', 'analysis', 'Nexus Lifecycle Analysis ↩
        failed',
    "${policyEvaluationResult.applicationCompositionReportUrl}"
    return
  } else {
    postGitHub commitId, 'success', 'analysis', 'Nexus Lifecycle Analysis ↩
        succeeded',
    "${policyEvaluationResult.applicationCompositionReportUrl}"
  }
}
```

Save your pipeline and then run the build. When the build has completed, you should see successful steps for Build and Nexus Lifecycle Analysis. Going to your GitHub pull request, you will see the successful checks and you can click the "details" link to view your report and address alerts or reevaluate policy.

Note
When you access various properties on objects in a pipeline build, Jenkins has security permissions that restrict unauthorized use. In *Manage Jenkins* navigate to *In-process Script Approval* and approve scripts (ie application composition report, affected component count) for the analysis to run.

In the above example, we utilized cURL to interact with the GitHub API and provide analysis feedback to GitHub. Many companies use GitHub for code review, and noting the status of CI builds in your pull request is a powerful analysis tool. In addition to GitHub, you can use the object returned from the policy evaluation with almost anything that provides an interactive API or other Jenkins plugins.

Want to see this in action? Here is a video example.

In addition to the Composition Report URL, you can set the following DTOs:

```
public class PolicyEvaluationResult {
        private final List<PolicyAlert> policyAlerts;
        private final int affectedComponentCount;
        private final int criticalComponentCount;
        private final int severeComponentCount;
        private final int moderateComponentCount;
        private final boolean reevaluation;
        private final String applicationCompositionReportUrl;
}

public class PolicyAlert {
        private final PolicyFact trigger;
```

```
        private final List<? extends Action> actions;
}

public class Action {
        private final String actionTypeId;
        private final String target;
        private final String targetType;
}

public class PolicyFacts {
        private final String policyId;
        private final String policyName;
        private final int threatLevel;
}
```

17.4 Inspecting Results

Once a specific build has successfully completed, a link to the *Application Composition Report* is shown on the project screen. Clicking on the link directs you to a display of the report within the IQ Server. The three boxes (red, orange, and yellow) located below the link give you counts for policy violations and are based on the associated severities (critical, severe, and moderate).

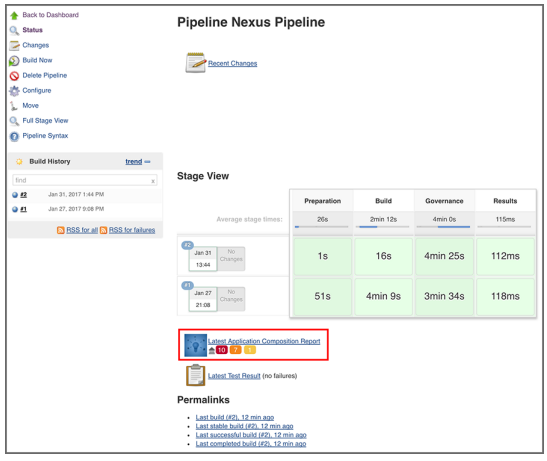


Figure 17.9: Job Overview Page with Application Composition Report Link

Note
If using Nexus IQ for Hudson/Jenkins 1.x, accessing this information may require a login. Also, if you are using a version of Jenkins prior to version 2.11, and IQ Server 1.7, a message will display indicating your report has been moved. Following this link takes you to the report on the IQ Server.

If you are looking for previous report results, navigate to a specific build in the *Build History*. If you previously scanned the application during that specific build, a new item is shown in the left menu, titled *Application Composition Report*. Following this link takes you to the report on the IQ Server.

Figure 17.10: Left Menu with Application Composition Report Link

Chapter 18

IQ Server and IDEs

An Integrated Development Environment, or IDE for short, is a software developer's workbench. It typically provides features to assist a developer in all their activities writing and maintaining software.

These include:

Powerful code editing features
with syntax highlighting, code completion and others helpful tools for efficient editing.

Integration of development tools
like compilers, debuggers, and profilers.

Support for version control systems
including tasks like checkout, commit, update and others.

Integration with language specific libraries
for accessing help, browsing libraries, completing code.

Build automation
within the tool e.g. for incremental compile as well as support for external build tools.

Testing support
for unit and integration test authoring and execution.

Integration with other software development tools
like code review systems, continuous integration servers, or repository managers.

The aim of an IDE is to maximize the productivity of a software developer during the entire software development process. Thus, an increasingly important part of these tasks is understanding, controlling, and managing the components that are part of your software product. This also includes carrying out the component lifecycle management related tasks like assessing security and license issues, as well as ensuring your component is using the best version possible.

Note
At this time, IQ Server supports the Eclipse IDE, IntelliJ IDEA and Microsoft Visual Studio as a direct add on. However, other IDEs can be integrated using Sonatype CLM for Maven.

Chapter 19

Sonatype CLM for Eclipse

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle license.

Often only called Eclipse, the Eclipse IDE is a very powerful, open source IDE written mostly in Java and managed by the Eclipse Foundation. It can be used for development in a number of languages, and is the most widely used IDE for Java development. It features a powerful plug-in system that allows you to customize the IDE, with features that support a large number software development-related tasks including localization options, version control systems, and myriad of other tasks.

19.1 Installing Sonatype CLM for Eclipse

Sonatype CLM for Eclipse can be installed by adding a new software repository. Navigate to the *Help* menu and select *Install New Software*. Press the *Add* button in the dialog displayed in Figure 19.1 and create a new repository with the *Location* set to the URL for Sonatype CLM for Eclipse releases from URL for the Sonatype CLM for Eclipse repository and a *Name* of your choice. Once you press *OK* a list of available releases is downloaded and an entry for the latest version of Sonatype CLM for Eclipse is displayed. Uncheck the item *Show only the latest versions of available software*, if you need to install an older release. Figure 19.1 shows a list of releases available.

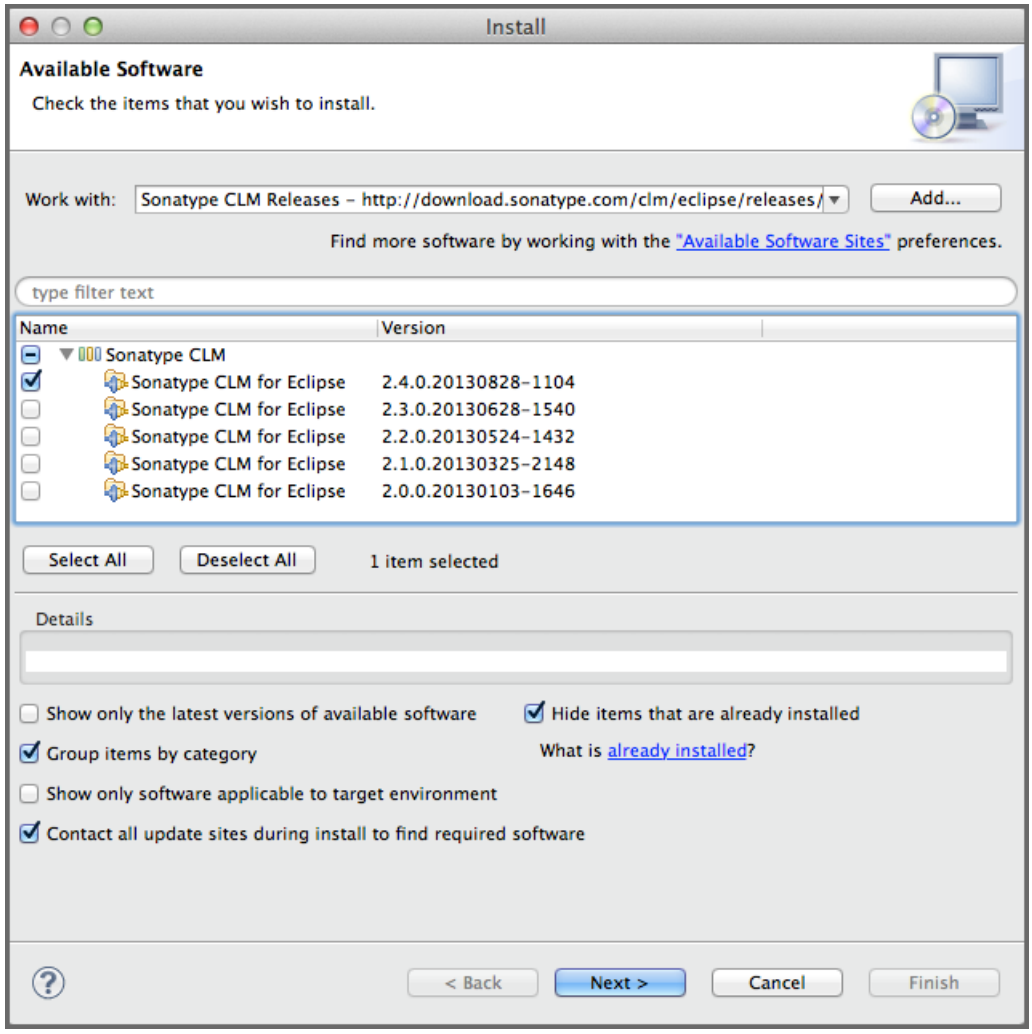


Figure 19.1: Eclipse Dialog to Install New Software with Sonatype CLM for Eclipse

URL for the Sonatype CLM for Eclipse repository

```
https://download.sonatype.com/clm/eclipse/releases/
```

Select the version of Sonatype CLM for Eclipse you would like to install and press *Next>*, proceed through accepting the end user license agreement and restart Eclipse to complete the installation.

Tip
Be sure to check the Section 3.2.5 before attempting to install.

19.2 Configuring Sonatype CLM for Eclipse

After successful installation of Sonatype CLM for Eclipse, you will be able to choose to show the Sonatype CLM view displayed in Figure 19.2.

To access this view:

1. Choose the *Window* menu and select *Other* in the *Show View* submenu.
2. Locate the *Sonatype CLM* section with *Component Info* as shown in Figure 19.2.
3. Select it and press *OK* and the view will appear in your IDE.

Tip
By typing "Compo" in the filter input, Component Info is automatically highlighted.

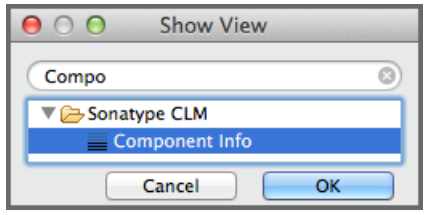


Figure 19.2: Activating the Component Info View of Sonatype CLM for Eclipse

Once the view is displayed, a warning will appear. This is because the you need to point Eclipse at your Sonatype CLM Server.

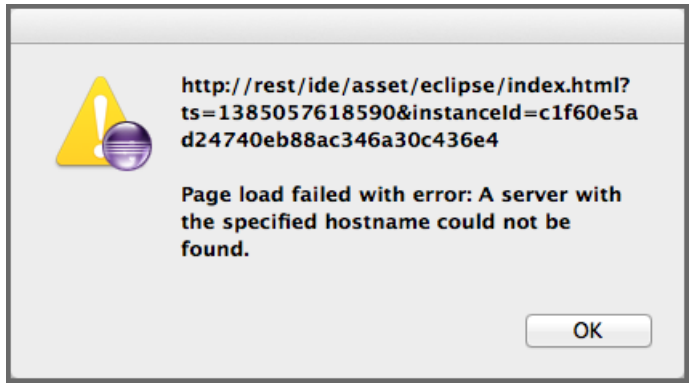


Figure 19.3: Warning after initial installation

To configure the Sonatype CLM for Eclipse plugin, simply press the *Configure* button in the top right-hand side of the component view.

Once in Sonatype CLM for Eclipse Configuration area, there are a number of parameters you will need to complete before you can review data from Sonatype CLM. These are covered below.

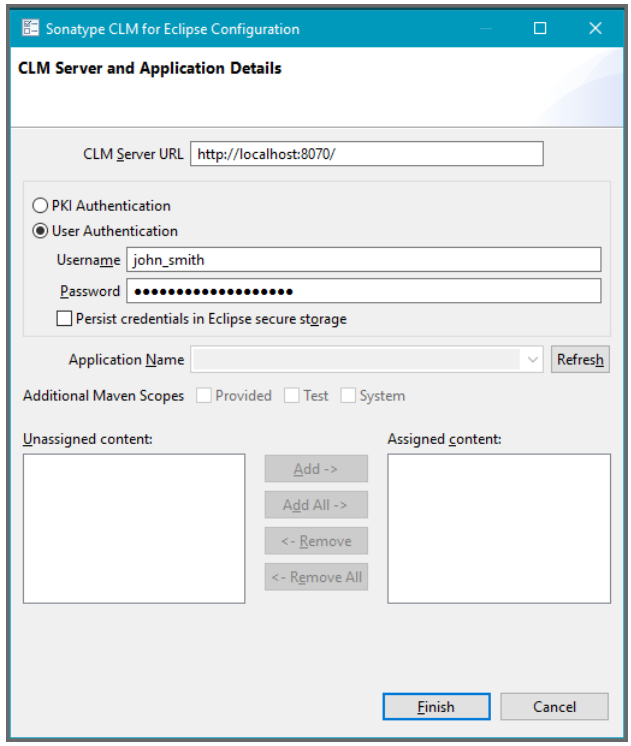


Figure 19.4: Sonatype CLM for Eclipse Configuration Dialog

CLM Server URL

The *CLM Server URL* input field has to be configured with the URL of your Sonatype CLM server.

User Authentication

Select this option to enter the username and password your system administrator has assigned you. In many cases this will simply be your single sign on credentials (e.g. LDAP), though it may also be a unique username.

Note

Selecting the option to persist credentials in Eclipse secure storage will reuse your credentials after a restart. If this is not selected you will need to reenter your credentials after a restart.

PKI Authentication

Select this option to delegate authentication to the JVM.

Application Name
The *Application Name* is the application which has been configured in the CLM server for you. This should match the common name you associate with the application. If you don't see a name you recognize, contact your Sonatype CLM Administrator.

Note
The drop down will display a list of all available applications after pressing the *Refresh* button.

Additional Maven Scopes
The compile and runtime scopes will always be considered. Additional scopes (provided, test, and system) you would like CLM to include can also be selected.

Assigned vs. Unassigned Content
After selecting an application name that represents a collection of policies configured in your CLM server, you can determine the Eclipse projects that should be analyzed. The list on the left titled *Unassigned content* contains all projects in your current Eclipse workspace that have not been assigned to a Sonatype CLM Application. Select a project from that list and add it to the *Assigned content* list on the right by clicking the *Add* button. This will add the project to the component analysis via the CLM server. In order to perform an analysis, the project needs to be open. To select multiple projects use the Shift and Control keys, and then click the *Add* button. The *Add All*, *Remove* and *Remove All* buttons help you to control the projects to analyze for different analysis sessions.

Note
Projects can, at most, be assigned to a single application.

With a finished selection of the projects you want to analyze, press the *Finish* button and wait for the component list to be displayed in the view. Section 19.3 documents how to inspect the results of the analysis and further features available from this information.

Tip
Only open projects will be taken into account as part of the component analysis.

19.3 Using the Component Info View

Once configured and the component analysis is completed a component view will look similar to the example displayed in Figure 19.5. The list of components will reflect an analysis of the build path.

Note
For Maven projects we include the compile and runtime scopes in the CLM evaluation. If you wish to include additional dependencies found in provided, test, and system scope, these can be configured.

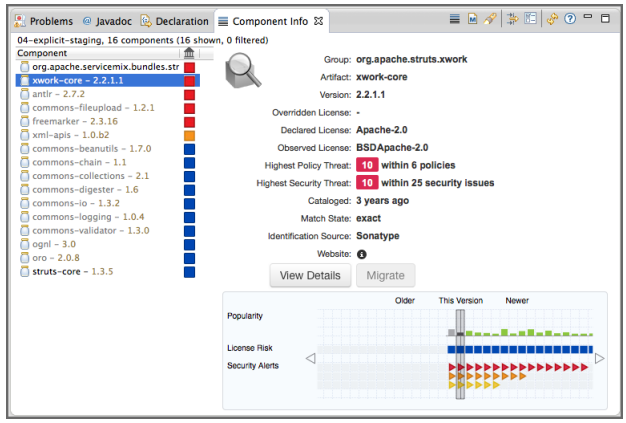


Figure 19.5: Example Component Info View

The top left-hand corner of the Sonatype CLM for Eclipse *Component Info* view displays either the number of projects currently being examined in the view, or the name of the specific project. Next to that, the number of components found, and the number of components shown in the list is displayed.

The top right-hand corner provides a number of buttons to access the following features of Sonatype CLM for Eclipse:

Open Component Details::

Opens another window with more details about the selected component including policy violations, license analysis and security issues.

Open POM
Opens the Maven pom.xml file of the selected component from the list in the Maven POM Editor.

Locate Declarations
Starts a search, that displays all usages of a selected component in the projects currently examined as documented in Section 19.5.

Filter
Brings up the filter selection, that lets you narrow down the number of components visible in the view as documented in. Section 19.4.

Configure
Activates the configuration dialog for the component analysis.

Refresh
Refreshes the component list and analysis results.

Show information about the plugin
Displays the Sonatype CLM for Eclipse support pages in an external browser.

Minimize
Minimize the view.

Maximize
Maximize the view.

The left-hand side of the view contains the list of components found in the project and identified by their artifact identifier and version number. A color indicator beside the components signals potential policy violations. The right-hand side of the view displays the details of the selected component from the list on the left.

Tip
You may notice some components are black or gray. This indicates components you have included (black) in your application, versus components that are included via a transitive dependency (gray).

By clicking on the list header on the left, the list can be ordered by the threat level of the policy a component has violated. In cases where there is no violation, the threat is simply light blue.

When you select a specific component in the list, the details, various properties, and a visualization of the different versions is displayed to the right of the list.

Tip
Depending on your screen size, the visual display may be shown below the component list. Try adjusting your screen size, or adjusting the panel.

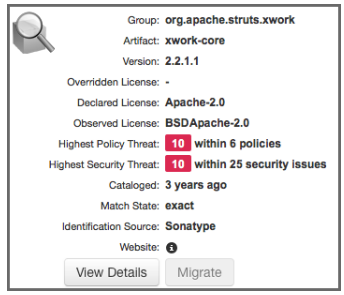


Figure 19.6: Details for a Component in the Component Info View

The details of a specific component as displayed in Figure 19.6 include properties about the component and provide access to further features:

Group
The Maven `groupId` the component was published with. In many cases this is equivalent with the reverse domain name of the organization responsible for the deployment or running the project.

Artifact
The Maven `artifactId` of the component acts as a short and ideally descriptive name.

Version
The Maven `version` of the component. A version string ending in `-SNAPSHOT` signifies a transient, in development version, any other version is a release version.

Overridden License
The value of a license override configured in your Sonatype CLM server.

Declared License
The software license declared by the developer of the project, which in some cases, is identified during research by Sonatype, or directly from the Maven POM file.

Observed License
The licenses found by the Sonatype CLM server in a source code analysis.

Highest Policy Threat
The highest threat level policy that has been violated, as well as the total number of violations.

Highest Security Threat
The highest security threat level as well as the number of issues found with the respective level.

Patch Available
This is a future feature that will provide details in instances where a patch is available. Patches will be provided and verified by Sonatype.

Cataloged
The age of the component in the Central Repository.

Identification Source
The catalog in which a component identification match was found. This includes either a match made by Sonatype (e.g. the catalog of the Central Repository), or a match made manually (i.e. through the Sonatype CLM claiming process).

Website
If available, an information icon providing a link to the project is displayed.

View Details
Press this button to display the details view for the selected component as detailed in Section 19.6.

Migrate
Press this button to start a project refactoring that allows you to change all usages of the current component to a different version as documented in Section 19.7.

Custom Metadata
This is a future feature that will allow you to display all custom metadata tags assigned to the component.

Figure 19.7: Properties of a Component for a Version Range

The visualization chart displayed in Figure 19.6 shows a number of properties for different, available versions of the selected component. Older versions are displayed on the left and newer versions on the right. Click on any section in the visualization, and all information for that particular version will be highlighted, with the specific version number at the bottom. In addition, the details for that version of the component will display in the left-hand list of properties. Arrows to the left and right of the visualization allow you to view the full range of available versions.

The properties displayed include:

Popularity
: the relative popularity of a version as compared to all other component versions.

License Conflict
: displays an indicator, if the observed licenses in the component are creating a legal conflict, e.g. GPL V2 and Apache V2 are not compatible for distribution of one component.

License Risk
: the risk posed based on what has been set within the license threat groups. While defaults are available, these are configurable via the Sonatype CLM Server.

Security Alerts
: indicators for the severity of security alerts affecting the component version.

You will likely notice a number of colors within the visualization chart. The value for each of these colors is as follows:

For Popularity

- Grey for any versions older than the current version.
- Green for newer, but within the same major version of the component.
- Blue for newer component versions, but with a greater major version than the current component.

For License and Security

- Blue - no security or license risk
- Yellow - minor security or license risk
- Orange - medium security or license risk
- Red - severe security or license risk

19.4 Filtering the Component List

The list of components found in the analysis and displayed in the component info view can be configured by pressing the *Filter* button. The filter dialog, displayed in Figure 19.8, allows you to narrow down the components shown.

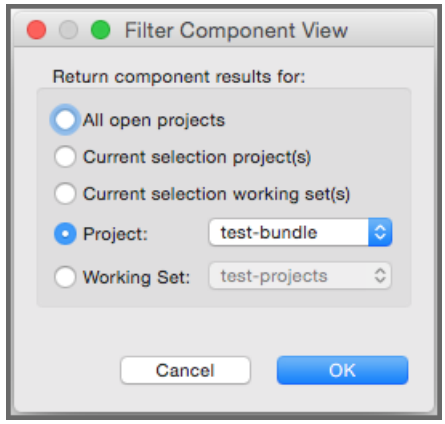


Figure 19.8: Filter Dialog for the Component Info View

The *Scope* setting determines, which projects' components are displayed in the list:

All open projects
include all the components, from all open projects.

Current selection project(s)
include the components from the project currently selected in the package explorer.

Current selection working set(s)
include the components from all the projects in the working set currently selected in the package explorer.

Project
include the components from the project selected in the drop down.

Working Set
include the components from all the projects in the working set selected in the drop down.

19.5 Searching for Component Usages

Once you have selected a specific component in the list on the left of the component info view, Sonatype CLM can determine in which projects the component is used. After pressing the *Locate Declarations* button, and once the search has completed, you will see the results in a tree view of projects and project pom.xml files, all displayed in the *Search* window.

Inspecting this list can help you assess the impact of a potential upgrade to a new component version. Further detail is documented in Section 19.7. Looking at the found projects, you can potentially remove wrong declarations, determine side effects from transitive dependencies, or find out why this component shows up as dependency at all.

19.6 Inspecting Component Details

Press the *Open Component Details* button in order to access the details about policy violations, license analysis and security issues for a specific component selected in the list. Figure 19.9 displays an example details view.

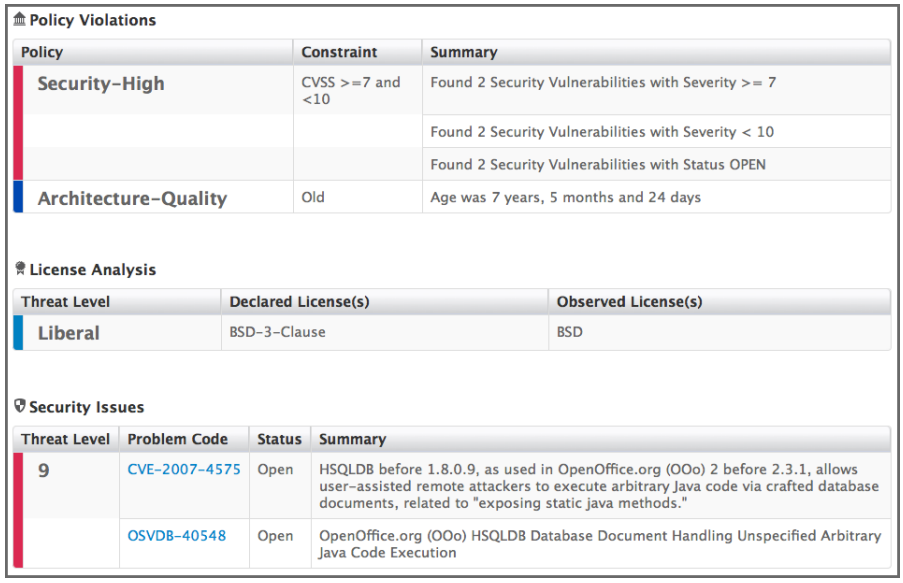


Figure 19.9: Example Component Details Display

The *Policy Violations* section in the top contains a list of all the policies that have been violated by the component.

The *License Analysis* section contains the *Threat Levels* posed by the licenses declared for each component, as well as those that have been observed in the source code.

The *Security Issues* section below contains the list of issues found. They are sorted from higher to lower risk, with each issue detailed by a *Threat Level* ranging from 0 to 10, or potentially with the value *Unscored* and a descriptive text in the *Summary* column. In addition, links to the security vulnerability database entry are added as links in the *Problem Code* column.

19.7 Migrating to Different Component Versions

If you determine that a component upgrade is required to avoid a security or license issue or a policy violation, after reviewing your component usage, Sonatype CLM for Eclipse can be used to assist you in the necessary refactoring.

NOTE
This feature relies on the project being a Maven project.

The first step to start the migration is to select a newer version for the component in the visualization chart. An example is displayed in Figure 19.10.

Figure 19.10: Migrating to a Newer Component Version

Once you have selected a different version than the one currently used, the *Migrate* button will become active. Pressing the button opens a dialog that assists you in the migration to the newer component. The complexity of this task varies considerably from project setup to project setup. The migration wizard is able to detect circumstances such as the component being a transitive dependency or versions managed in a property.

The simplest flow is when a dependency version can be applied, and the result is a single dialog like the one displayed in Figure 19.11.

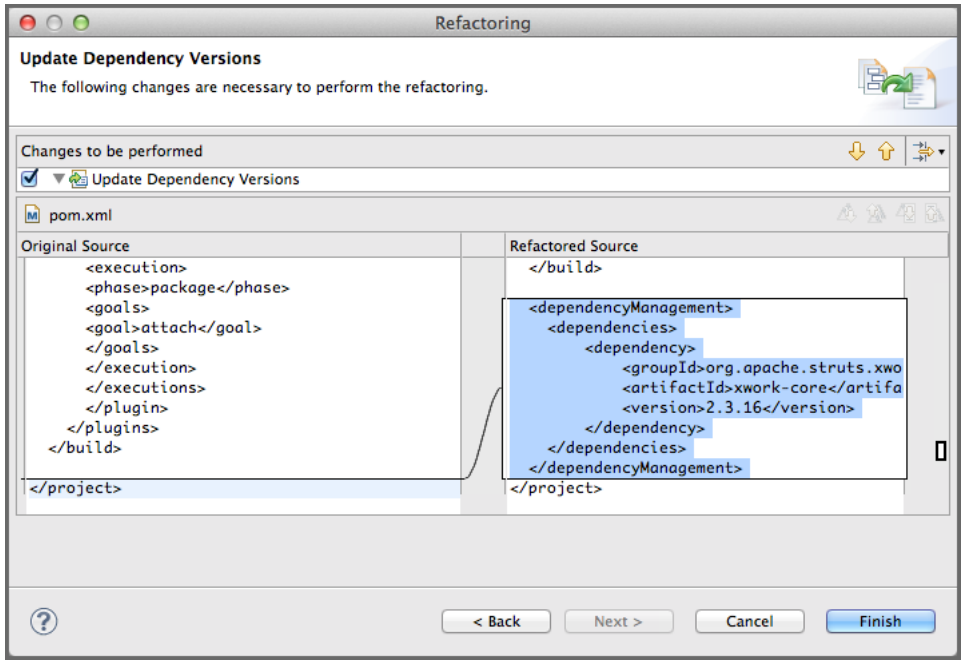


Figure 19.11: Applying a Dependency Version Upgrade

If the version is managed in a property, the initial screen from Figure 19.12 allows you to select if you want to continue with a property upgrade, or perform a replacing version upgrade.

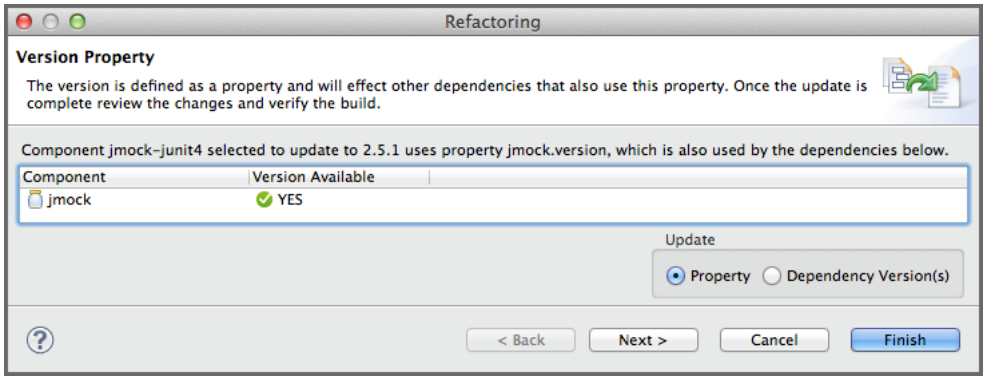


Figure 19.12: Selecting Dependency Version or Property Upgrade

Once you have selected to perform a property upgrade, you will be able to apply it in the next screen, *Refactoring*, visible in Figure 19.13.

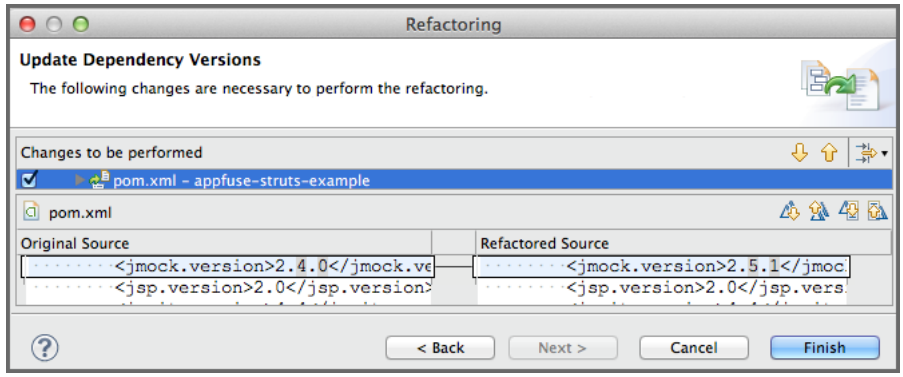


Figure 19.13: Applying a Property Upgrade

The *Refactoring* screen features navigation tools allowing you to view all potential changes in the dialog, and step through them one by one before deciding to continue.

After you have completed the refactoring of your project files, you should perform a full build, as well as a thorough test, to determine that you can proceed with the new version in your development.

Typically smaller version changes will have a higher chance of working without any major refactorings, or adaptations, of your code base and projects, while larger version changes potentially give you more new features or bug fixes.

Your release cycle, customer demands, productions issues and other influencing factors, will determine your version upgrade choices. You might decide a multi-step approach, where you do a small version upgrade immediately to resolve current issues and then work on the larger upgrade subsequently to get the benefits of using a newer version. Or, you might be okay with doing an upgrade to the latest available version straight away. Potentially, a combination of approaches in different branches of your source code management system is used to figure out the best way of going forward with the upgrade.

Sonatype CLM for Eclipse and other tools of the Sonatype CLM suite can assist you through the process of upgrading, as well as monitoring, the applications after upgrade completion.

Chapter 20

IQ for IDEA

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle license.

IntelliJ IDEA is a full featured IDE used for Java development. IQ for IDEA provides component analysis for both the Community and Ultimate edition of IntelliJ IDEA.

20.1 Installing IQ for IDEA

IQ for IDEA supports installation via a zip file from disk. Installation is performed similarly to other plugins, using the Settings/Preferences dialog. Click on Plugins from the left hand pane which will expose the option to Install from disk. From there, browse to the plugin zip file and select it. Remember to Restart IntelliJ IDEA before continuing to access the plugin.

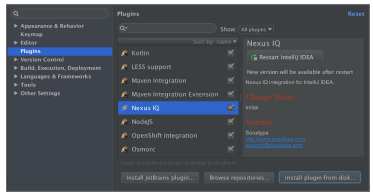


Figure 20.1: IDEA Plugin dialog to install IQ for IDEA

20.2 Configuring IQ for IDEA

After the successful installation of IQ for IDEA, the plugin must be configured to connect to IQ Server. The configuration can be accessed via the Settings/Preferences dialog. Expanding Other Settings in the left hand pane will reveal Nexus IQ. Click on Nexus IQ to set up the plugin for IQ Server.

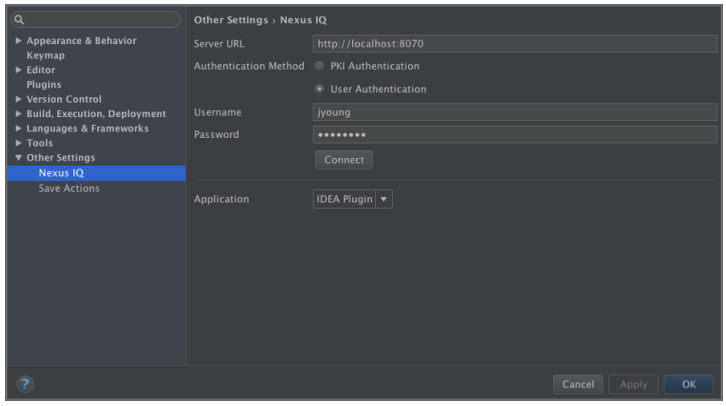


Figure 20.2: Nexus IQ plugin settings

Server URL
Enter the url of IQ Server

Authentication Method

a. **PKI Authentication**: Delegate authentication to the JVM.
b. **User Authentication**: Enter the username and password your IQ Server Administrator has assigned you.

Note
You will be prompted for your Master Password (or to set up a Master Password) when saving the Preferences/Settings. This allows IDEA to store your IQ Server password securely.

Once the IQ Server information is provided, click Connect to verify the connection to IQ Server. Next select an Application from the dropdown to run policy against.

20.3 Using the Component Info View

The IQ for IDEA tool window can be accessed by clicking the Nexus IQ tab on the bottom tool strip of IDEA. If not accessible from there, it should also be available in View under Tool Windows. Once configured and the component analysis is completed a component view will look similar to the example displayed in Figure 20.3. The list of components will reflect an analysis of the project's libraries.

Note
By default, all project libraries are included in the list. Filters can be applied to adjust which libraries are included by scope: Compile, Test, Runtime, and Provided.

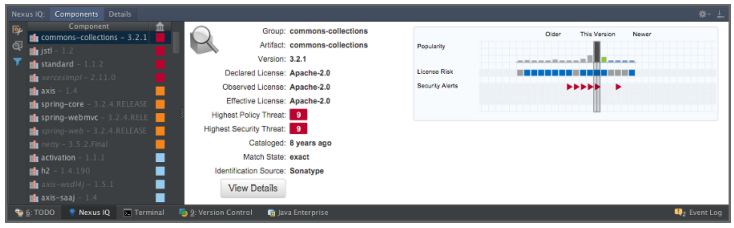


Figure 20.3: Example Component Info View

Right clicking on any component will bring up a menu of actions. Maven projects should allow for the following: View Details, Find Usages, and Open Maven POM.

- View Details will open the details screen providing more context to the component.

- Find Usages will bring up a list of every module the component is used in. Clicking on a module will bring up the location in the Maven POM where the component is declared.
- Open Maven POM will open the Maven POM of the component selected.

Chapter 21

IQ for Visual Studio

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle license.

Visual Studio is a full featured IDE. IQ for Visual Studio provides component analysis for both the Community, Professional and Enterprise version of Visual Studio.

21.1 Installing IQ for Visual Studio

IQ for Visual Studio can be installed from within Visual Studio using the Extensions manager or via the Microsoft Visual Studio Marketplace.

21.2 Configuring IQ for Visual Studio

IQ Server options are available from within the Visual Studio Options dialog. A URL, Username and Password can be entered at any time and an Application may be chosen for each solution when opened.

The Verify button can be used to verify the connection if a solution is not opened whereas the Reload button will load available applications when a solution is opened.

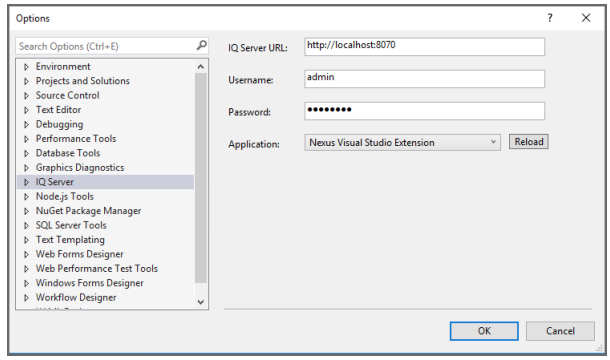


Figure 21.1: IQ for Visual Studio Extension Settings

21.3 Using IQ for Visual Studio

The IQ for Visual Studio tool window can be accessed by clicking the Nexus IQ tab on the bottom tool strip of Visual Studio. If not accessible from there, it should also be available in View under Other Windows. Once configured and the component analysis is completed, a component view will look similar to the example displayed in Figure 21.2. Component details are available by double clicking on the component name via the View Details button in the component view.

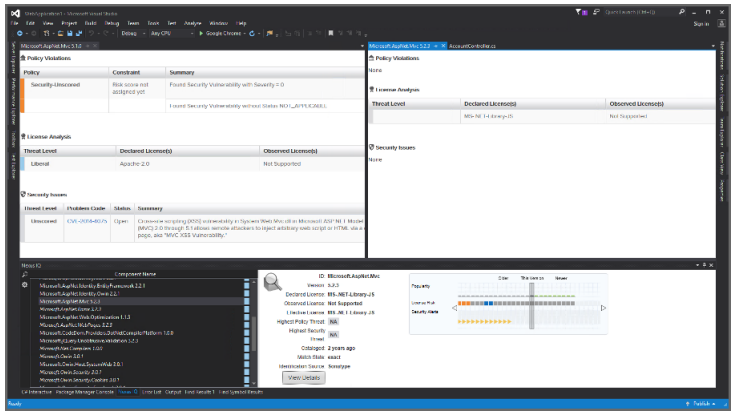


Figure 21.2: IQ for Visual Studio

Chapter 22

Sonatype CLM for SonarQube

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle license.

Note
The rebranding and renaming of *Sonatype CLM* to *Nexus IQ Server* started with the 1.17 release. You may still see references to *Sonatype CLM* in the product or documentation. We realize this may cause some confusion, and appreciate your patience as we move forward.

IQ Server integrates with a wide range of external enforcement points that include continuous integration servers (Hudson/Jenkins, Bamboo, the CLI and Maven), the IDEs (Eclipse and IntelliJ IDEA), and repository management (Nexus).

The enforcement points are a common aspect of the development lifecycle, and in IQ Server, each represents a unique stage. This creates an invaluable integration of IQ Server with industry standard tools that already make the lives of your business and development process even better. This also means, your team has greater overall control in identifying and reducing open source component risk.

Better component usage doesn't just lead to risk reduction though, it also leads to better applications. This is something that ties closely with code analysis, and tools such as SonarQube.

As a user of SonarQube, you know first hand the impact that principles such as the 7 Axes of Code Quality can have on the applications and projects your teams create. Paralleling this, as a user of IQ Server you also know how policy management is a critical and essential part of open source component usage.

Sonatype CLM for SonarQube brings both of these together, and in this chapter we'll cover everything you need to get going as quickly as possible. This includes:

- Download, installation, and configuration
- Application Composition Report access

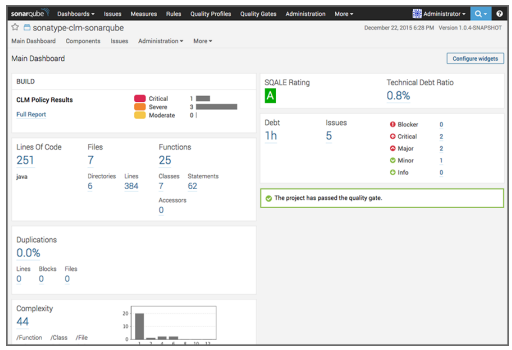


Figure 22.1: SonarQube Overview

Note
See the Requirements Chapter for information on Sonatype CLM for SonarQube supported versions.

22.1 Installation

There are a few things that must be done prior to getting Sonatype CLM for SonarQube running:

- Install and configure the IQ Server.
- Create at least one organization and one application.

- Evaluate the application at least once.
- Have an existing SonarQube project.

When the above items are complete, use the following instructions to install Sonatype CLM for SonarQube:

1. Download CLM for SonarQube from our Nexus IQ Server downloads page.

Warning
If your installation of SonarQube is running, stop it before adding the plugin.

2. Once downloaded, find the *extensions* > *plugins* directory in your installation of SonarQube.
3. Copy the Sonatype CLM for SonarQube JAR file (the one downloaded from the link above) into this directory.
4. Start your SonarQube instance, and log in with your administrator account.

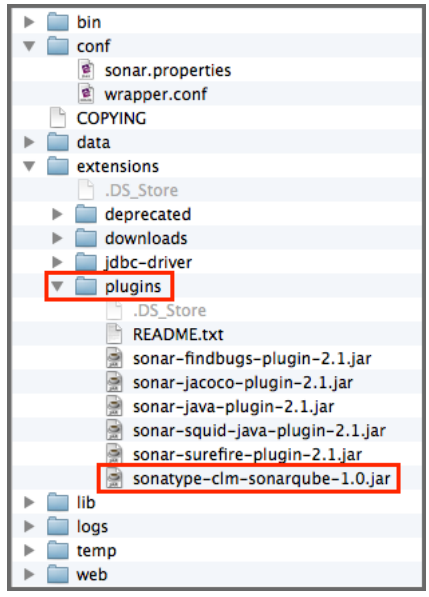


Figure 22.2: SonarQube Plugin Directory

22.2 Configuration

The Sonatype CLM Server settings allow you to specify the location of your CLM server. In the example below, basic defaults for configuration have been used. Yours will likely be different.

1. From the main SonarQube interface, click the *Administration* menu item.
2. From the Administration menu, select *Configuration*, and then click *CLM Settings*.

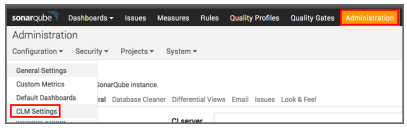


Figure 22.3: CLM Settings Menu

3. In the CLM Settings window, enter your Sonatype CLM Server URL.
4. Select an *Authentication Method*:
   a. select *User Authentication* and enter your username and password (must be at least a developer role for the application associating with this SonarQube project), or
   b. select *PKI Authentication* to delegate authentication to the JVM.

   Note
   These settings will be used across all projects for your SonarQube installation. Because of this we suggest creating a single account in Sonatype CLM for SonarQube, and then associating that account with the Developer role for the applications you will be linking to SonarQube.

5. Click the *Save CLM Settings* button to save your Sonatype CLM settings.

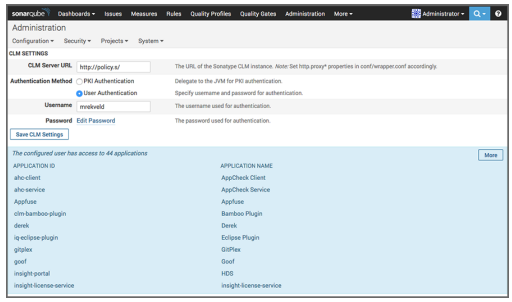


Figure 22.4: SonarQube CLM Server Settings

Proxy Configuration

In some instances, your Sonatype CLM Server may be setup behind a proxy. If this is the case, have your SonarQube Administrator configure your proxy via the wrapper.conf file located within the conf directory of your SonarQube installation. For more information, please refer to SonarQube's documentation.

22.3 Select the CLM Application

The next step is selecting the application that is associated with your SonarQube project.

1. Open the project you want to associate the application with, and then click *Administration* > *Sonatype CLM*.

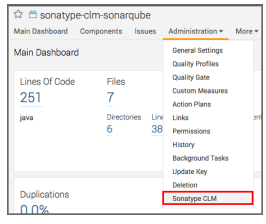


Figure 22.5: SonarQube Sonatype CLM Menu Item

2. In the Sonatype CLM Configuration Area, select an application from the drop down list that should

be associated to your SonarQube Project.

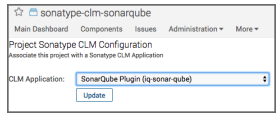


Figure 22.6: SonarQube Sonatype CLM Application Selection

3. Click the *Update* button.

22.4 Add and Configure the Sonatype CLM Widget

The final step is to add the Sonatype CLM Widget to your SonarQube project. This is done from the *SonarQube Widget Configuration area*.

1. Click on the *Configure Widgets* button located in the top-right section of the screen.

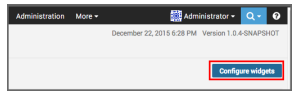


Figure 22.7: SonarQube Configure Widgets Button

2. The easiest way to find the Sonatype CLM Report Summary widget is by using the SonarQube widget search (just enter Sonatype). Click on the *Add Widget* button to add the widget to your dashboard.

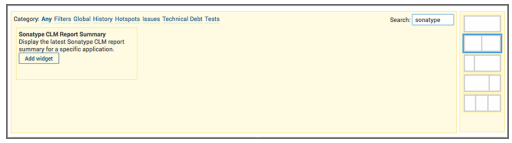


Figure 22.8: SonarQube Search for CLM Widget

3. Next, click on the *Edit* link in the top right of the Sonatype CLM Report Summary widget box. Several options will display.

a. Select the *Project* you want to see Sonatype CLM data in.

> **Tip**
> The option to select a project is only available when adding the widget from a non-project-specific dashboard.

b. Enter a *Title*. This appears above the summary information for the Sonatype CLM data.
c. Choose the *CLM Stage*. The CLM stage selected affects which Application Composition Report will be used to display summary-level data. Be sure to pick the stage that best represents the state of your application when it is scanned by SonarQube. *Default* will use the *Build stage*.

> **Note**
> Due to technical constraints, the dropdown option also includes stages that might not be available for your license. Selecting any of those will however yield an error when accessing the IQ Server server.

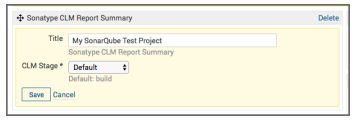


Figure 22.9: SonarQube Configure Sonatype CLM Widget options

4. Click the *Save* button to save your selections.

22.5 Accessing the Application Composition Report

Within SonarQube, you are provided with basic summary information for the Application, as well as a link to the associated Application Composition Report. To access the detailed information provided by this report, click on the *Full Report* link displayed in the Sonatype CLM Report Summary widget in your project.

Figure 22.10: SonarQube Sonatype CLM Widget Example

Chapter 23

Nexus IQ CLI

Tip
The topics discussed in this chapter require IQ Server with the Nexus Lifecycle or Nexus Auditor license.

While tools offering full integration for evaluations are provided (e.g. Nexus, Eclipse, and Hudson/Jenkins), any application can be evaluated against your policies simply by using the Nexus IQ CLI.

Before you get started, there are a few things you should make sure you have, including:

- A general familiarity with the CLI (you don't need to be an expert, but basic knowledge helps).
- Installed and set up the IQ Server
- Set up an organization and application
- Created or imported a policy for your application, or the organization that contains your application.

Note
With regard to integrating the Nexus IQ CLI with CI servers, you should also have a familiarity with the way your particular CI utilizes build steps to launch a simple script.

23.1 Downloading the Nexus IQ CLI

To use the Nexus IQ CLI, first download the jar file named similar to `nexus-iq-cli-1.31.0.jar` from the Sonatype Support website and place the file in its own directory.

Note
Be sure to remember where you placed the file you downloaded. As a recommendation, it's best to have the Nexus IQ CLI in its own directory, and not shared with the IQ Server.

23.2 Locating Your Application ID

Before you start an evaluation, you need to have your Application ID. To get this...

1. Log into your IQ Server with a user account with at least a developer role for the application you plan on evaluating.
2. Click the Organization & Policies icon .
3. Click on an organization in the menu on the left to display a list of applications, and then click on the application you want to find the Application ID for. You will see a screen similar to Figure 23.1
4. Locate the text to the right of the application name. The application ID is the text in parenthesis.

Note
The application ID will be truncated if it is too long. The full application ID can be copied to the clipboard from the Actions menu.

Figure 23.1: Application Overview and Application Identifier

23.3 Evaluating an Application

Now that you have the Nexus IQ CLI set up, you are ready to evaluate an application. The application can be an archive file, a directory containing such archives or a Docker image. For Docker images, a Twistlock environment is required.

As a Java application, it can be started using the `java` command, and adding the necessary parameters. The syntax below represents the minimum set of options required to evaluate an application.

If the application is an archive or directory:

```
java -jar [scanner jar] -i [application id] -s [server URL] [target]
```

If the application is a Docker image:

```
java -cp [scanner jar] com.sonatype.insight.scan.cli. ↩
    TwistlockPolicyEvaluatorCli -i [application id] -s [server URL] -- ↩
    twistlock-scanner-executable [Twistlock scanner executable] --twistlock ↩
    -console-url [Twistlock console URL] --twistlock-console-username [ ↩
    Twistlock console username] --twistlock-console-password [Twistlock  ↩
    console password] [target]
```

`scanner jar`
: This is the path to the Nexus IQ CLI jar file e.g. `./nexus-iq-cli-1.31.0.jar`.

`--authentication`
: Using the switch `-a`, enter the user name:password (e.g. `MyUserName:MyUserPassword`).

Note
Authentication will permit (or prevent) the ability to submit an application for evaluation, as well as retrieve the summary results and URL.

`--pki-authentication`
: Delegate to the JVM for authentication.

`--application-id`
: Using the switch `-i`, enter the application id for your application (see instructions above).

`--server-url`
Using the switch `-s` enter the location of your IQ Server (e.g. `http://localhost:8070`).

`Target`
This is the path to a specific application archive file, a directory containing such archives or the ID of a Docker image. For archives, a number of formats are supported, including `jar`, `war`, `ear`, `tar`, `tar.gz`, `zip` and many others.

`--twistlock-scanner-executable`
The path to the Twistlock scanner binary, e.g. /opt/twistlock/twistlock-scanner

`--twistlock-console-url`
The URL for the Twistlock console, e.g. https://localhost:8083

`--twistlock-console-username`
The user name used to connect to the Twistlock console.

`--twistlock-console-password`
The password for the user name used to connect to the Twistlock console.

Tip
Listed in the options below, you can specify the specific stage. However, if you do not include this option the system will default to the Build stage.

23.3.1 Additional Options

There are several additional options that can be used in the construction of the syntax for evaluating an application with the Nexus IQ CLI.

`--fail-on-policy-warnings`
using the switch `-w` will cause a failure of the evaluation if any warnings are encountered. By default, this is set to false.

`--ignore-system-errors`
Using the switch `-e`, allows you to ignore any system errors (e.g. IO, Network, server, etc.). This is most helpful when using the Nexus IQ CLI with continuous integration servers, as these errors can cause the unintentional failure of a build.

`--proxy`
Using the switch `-p`, you can specify a proxy to use in connecting to the IQ Server. The format is <host[:port]>.

--proxy-user
Using the switch `-U`, you can specify credentials for the proxy. The format is <username:password>.

--result-file
Using the switch `-r`, you can specify the name and location of a JSON file that will store the results of the policy evaluation in a machine-readable format.

--stage
Using the switch `-t`, you can specify the stage you wish the report to be associated with. This is an optional parameter, and if it is not specified, the report will be associated with the Build stage by default.

Note
At this time only the Build, Stage Release, and Release stages will display a report in the IQ Server Reports area. For a full list of stages, use the CLI help provided with the tool.

--twistlock-tlsverify
Pass the --tlsverify param to the Twistlock scanner, e.g. "--twistlock-tlsverify false" will pass "--tlsverify=false" to the Twistlock scanner.

23.4 Example Evaluation

In an example scenario, let's say you have copied the `nexus-iq-cli-1.31.0.jar` as well as the application you want to examine to a specific directory e.g. `~/nexus-iq-server-test`. The application's filename is `sample-application.zip`.

To evaluate this application you have to identify the application ID and supply it with the `-i` switch as well as supply the URL of your IQ Server with `-s`. As an option, and what is demonstrated below, you can also specify a particular stage.

The full command line for an Application ID `Test123` and a URL of `http://localhost:8070` is

```
java -jar ./nexus-iq-cli-1.31.0.jar -i Test123 -s http://localhost:8070
-t release sample-application.zip
```

To access help content for the Nexus IQ CLI, run it without supplying parameters:

```
java -jar ./nexus-iq-cli-1.31.0.jar
```

Go ahead and try an evaluation yourself. The Nexus IQ CLI will accept a number of file types, including jar, war, and zip files. If your evaluation is successful, the log output of the command execution will provide a summary as well as a link to the produced results similar to:

```
[INFO] Policy Action: Warning
[INFO] Summary of policy violations: 4 critical, 85 severe, 46 moderate
[INFO] The detailed report can be viewed online
at http://localhost:8070/ui/links/application/my-app/report/95c4c14e
```

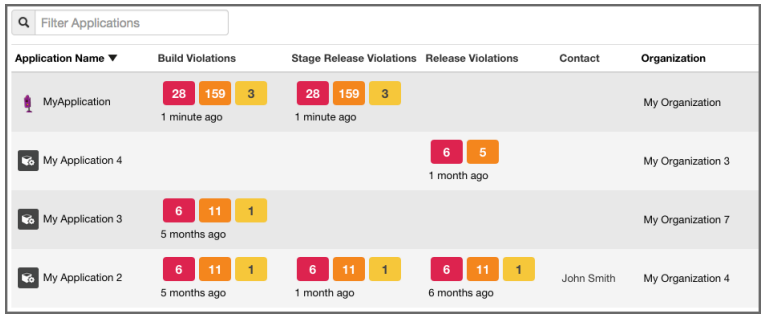


Figure 23.2: Violations Report After an Evaluation

Note
If using the Nexus IQ CLI, and you kept our defaults, the report will be listed under Build Violations. You should see something similar to the results displayed in Figure 23.2.

23.5 Using the Nexus IQ CLI with a CI Server

We won't be covering a specific CI here, but in general, all you need to identify (in your CI), is the location for adding a build step that includes processing a simple shell script during the building of your application.

Once you are there, make sure your script calls the Nexus IQ CLI using the following syntax:

```
java -jar [ScannerJar] -i [AppID] -e [IgnoreSystemErrors] -w [ ↩
    FailOnPolicyWarning] -s [ServerURL] [Target]
```

Each of the areas in the syntax above have been described in the previous section Evaluating an Application.

Given a typical setup, your syntax, including all available options will likely look similar to this:

```
java -jar /scanner/nexus-iq-cli-1.31.0.jar -i tester123 -s http:// ↩
    localhost:8070 ./target/sample-app.war
```

Now, when your application is built, the build step you have added will call the Nexus IQ CLI, evaluate your application, and upload results of the evaluation to the IQ Server. By default this will be placed below the build column in the Reports and Application area on the IQ Server, for your application.

Note
We advise you to use a separate application identifier for each of your unique applications. Using the same application identifier will result in report results being overwritten each time an application is built. While this is always the case, matching the latest evaluation to the right application can prove difficult.

Chapter 24

Sonatype CLM for Maven

Tip
The topics discussed in this chapter require IQ Server with the Lifecycle license.

Sonatype CLM for Maven allows users to evaluate any Maven-based software projects, in the same way our integrated tools (e.g. Nexus Pro, Eclipse, Hudson/Jenkins) do.

This means you can access the same robust, reporting features no matter what toolset you use. It can be run on a command line interface and can therefore be executed on any continuous integration server, as well as a number of popular IDEs.

How you use the Sonatype CLM for Maven plugin widely depends on how you enforce policy. However, in general, when using the plugin on a multi-module project in most cases you will only configure an execution for the modules that produce components that will be deployed as an application.

Typically these are `ear` files or `war` files for deployment on application servers or `tar.gz` or other archives that are used for production deployments or distribution to users. That said, you can also analyze all modules of a project. Again, this will largely depend on what your CLM policy is enforcing and what you want to validate.

In the following sections, we'll provide details on these goals and their usage:

Index
The `index` goal of the plugin allows you to prepare data for analysis with Sonatype CLM for CI.

Attach
The `attach` goal aids your integration with Sonatype Nexus CLM Edition and the release process using the staging tools of Nexus.

Evaluate
The `evaluate` goal can trigger an evaluation directly against a Sonatype CLM server.

Note
The `help` goal provides documentation for all the goals and parameters and you can invoke it with an execution like *`mvn com.sonatype.clm:clm-maven-plugin:help`*.

24.1 Evaluating Project Components with Sonatype CLM Server

The `evaluate` goal scans the dependencies and build artifacts of a project and directly submits the information to a Sonatype CLM Server for policy evaluation.

If a policy violation is found and the CLM stage is configured to `Fail`, the Maven build will fail. If invoked for an aggregator project, dependencies of all child modules will be considered.

The `evaluate` goal requires the Sonatype CLM Server URL as well as the application ID to be configured. Optionally a CLM stage can be configured.

The command line arguments are:

`clm.serverUrl`
the URL for the CLM server, this parameter is required.

`clm.serverId`
used for authentication and must match the id given to the CLM Server specified in your Maven settings.

`clm.username`
the username used to authenticate access to the CLM server.

Note
This is not required when using `clm.serverId`, but can be used to overwrite those settings.

`clm.password`
the password for the username indicated above.

Note
This is not required when using `clm.serverId`, but can be used to overwrite those settings.

`clm.pkiAuthentication`
delegate to the JVM for authentication.

`clm.applicationId`
the application identifier for the application to run policy against, this parameter is required

`clm.resultFile`
the path for specifying the location of a JSON file where the following information will be stored:

- applicationId : Application ID
- scanId : Organization ID
- reportHtmlUrl : URL to the HTML version of the report
- reportPdfUrl : URL to the PDF version of the report
- reportDataUrl : URL to the Data version of the report (for use via CURL, or similar tool)

`clm.stage`
the stage to run policy against with the possible values of `develop`, `build`, `stage-release`, `release` and `operate` with a default value of `build`.

`clm.additionalScopes`
the additional scopes you would like CLM to include components from during the evaluation. Values include `test`, `provided`, and `system`. In cases where you want to include more than one of these, separate the list using a comma (see examples below).

An example invocation is:

```
mvn com.sonatype.clm:clm-maven-plugin:evaluate -Dclm.additionalScopes=test ↩
    ,provided,system -Dclm.applicationId=test -Dclm.serverUrl=http:// ↩
    localhost:8070
```

You can avoid specifying the parameters on the command line by adding them to your `settings.xml` or `pom.xml` as properties.

```
<properties>
   <clm.serverUrl>http://localhost:8070</clm.serverUrl>
   <clm.applicationId>test</clm.applicationId>
</properties>
```

Sonatype CLM for Maven can be executed against an aggregator project. When executed in an aggregator project, it calculates the dependencies and transitive dependencies of all child modules and takes all of them into account for the policy evaluation.

The `evaluate` goal logs its activity and provides the location of the generated report.

```
[INFO] --- clm-maven-plugin:2.6.0-01:evaluate (default) @ test-app ---
[INFO] Starting scan...
[INFO] Scanning ../repository/org/codehaus/plexus/plexus-utils/3.0/plexus- ↩
   utils-3.0.jar...
[INFO] Scanning ../repository/org/apache/maven/maven-settings/3.0/maven- ↩
   settings-3.0.jar...
[INFO] Scanning target/test-app-1.0-SNAPSHOT.jar...
[INFO] Saved module scan to /opt/test-app/target/sonatype-clm/scan.xml.gz
[INFO] Uploading scan to http://localhost:8070 ...
[INFO] Evaluating policies... (ETA 5s)
[INFO] Policy Action: None
Summary of policy violations: 0 critical, 0 severe, 0 moderate
The detailed report can be viewed online at
http://localhost:8070/ui/links/application/test/report/f4582a1570634dc2ac8
```

Note
The evaluate goal cannot be bound to a lifecycle phase.

After a successful build the report can be accessed in the Sonatype CLM server under the application that was configured. A direct link is provided on the log.

24.1.1 Authentication

To configure authentication to the CLM Server, you will need to add your Sonatype CLM Server information to your Maven `settings.xml` file:

```
<settings>
   ...
   <servers>
      <server>
         <id>clm_server</id>
         <username>my__clm_login</username>
         <password>my_clm_password</password>
         <!--username and password are not required if using JVM (PKI)  ↩
             authentication-->
     </server>
     ...
   </servers>
   ...
</setting>
```

Note
In our example we have not encrypted our password. This is generally recommended. The Apache Maven project provides instructions for password encryption. Additionally, username and password can still be specified at the command line, and will be used in place of these settings.

24.1.2 Simplifying Command Line Invocations

If you happen to use the plugin frequently by running it manually on the command line and want to shorten the command line even more, you can add a plugin group entry to your Maven `settings.xml` file:

```
 <settings>
   ...
   <pluginGroups>
     <pluginGroup>com.sonatype.clm</pluginGroup>
     ...
   </pluginGroups>
   ...
 </settings>
```

This enables you to invoke the plugin using its shorthand prefix form:

```
mvn ... clm:index
```

24.1.3 Skipping Executions

The `clm.skip` parameter can be used, when a CLM plugin execution is configured in your project's `pom.xml` file, but you want to avoid the execution for a particular build. An example execution is:

```
mvn clean install -Dclm.skip=true
```

The parameter can also be set in your IDE configuration for Maven build executions or as a property in your settings.xml or pom.xml:

```
<properties>
   <clm.skip>true</clm.skip>
</properties>
```

24.2 Creating a Component Index

When evaluating a Maven-based software project, Sonatype CLM for Maven can take advantage of the dependency information contained in the project's `pom.xml` files and the information about transitive dependencies available to Maven.

The `index` goal of Sonatype CLM for Maven allows you to identify component dependencies and makes this information available to Sonatype CLM CI tools (e.g. Sonatype CLM for Hudson/Jenkins or Bamboo). You can invoke an execution of the `index` goal manually as part of your command line invocation by executing the index goal after the package phase:

```
mvn clean install com.sonatype.clm:clm-maven-plugin:index
```

Alternatively you can configure the execution in the `pom.xml` files `build` section or in a `profile`'s build section.

```
<build>
   <plugins>
     <plugin>
       <groupId>com.sonatype.clm</groupId>
       <artifactId>clm-maven-plugin</artifactId>
       <version>2.6.0-01</version>
       <executions>
         <execution>
           <goals>
           <goal>index</goal>
```

```
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
```

With the above configuration a normal Maven build execution with e.g. `mvn clean install` will trigger the CLM plugin to be executed in the `package` phase and result in a log output similar to

```
[INFO] --- clm-maven-plugin:2.6.0-01:index (default) @ test-app ---
[INFO] Saved module information to /opt/test-app/target/sonatype-clm/ ↩
    module.xml
```

If you want to manually configure the lifecycle phase to execute the plugin, you have to choose a phase after `package`.

The generated `module.xml` file contains the information that will be picked up by Sonatype CLM for CI and incorporated into the CLM evaluation. This improves the analysis since Sonatype CLM for Maven is able to create a complete dependency list rather than relying on binary build artifacts.

Note

By default only dependencies in the `compile` and `runtime` scopes will be considered, since this reflects what other Maven packaging plugins typically include. Dependencies with the scopes `test`, `provided` and `system` must be manually added, and are described in the Evaluating Project Components with Sonatype CLM Server section.

24.2.1 Excluding Module Information Files in Continuous Integration Tools

When using the Sonatype CLM Maven plugin and the `index` goal, module information files are created. If desired, you can exclude some of the modules from being evaluated. For example, you may want to exclude modules that support your tests, and don't contribute to the distributed application binary.

The default location where the module information files are stored is `${project.build.directory}/sonatype-clm/module.xml`.

In the supported CI tool, you will see a section labeled *Module Excludes*. On this area, use a comma-

separated list of Apache Ant styled patterns relative to the workspace root that denote the module information files (`**/sonatype-clm/module.xml`) to be ignored.

Here's an example of the pattern described above:

```
**/my-module/target/**, **/another-module/target/**
```

If unspecified, all modules will contribute dependency information (if any) to the evaluation.

24.3 Creating a Component Info Archive for Nexus Pro CLM Edition

The `attach` goal scans the dependencies and build artifacts of a project and attaches the results to the project as another artifact in the form of a `scan.xml.gz` file. It contains all the checksums for the dependencies and their classes and further meta information and can be found in the `target/sonatype-clm` directory. A separate `scan.xml.gz` file is generated for each maven module in an aggregator project in which the plugin is executed.

This attachment causes the file to be part of any Maven `install` and `deploy` invocation. When the deployment is executed against a Sonatype Nexus CLM Edition server the artifact is used to evaluate policies against the components included in the evaluation.

To use this goal, add an execution for it in the POM, e.g. as part of a profile used during releases:

```
<build>
   <plugins>
     <plugin>
       <groupId>com.sonatype.clm</groupId>
       <artifactId>clm-maven-plugin</artifactId>
       <version>2.6.0-01</version>
       <executions>
         <execution>
           <goals>
           <goal>attach</goal>
           </goals>
         </execution>
       </executions>
     </plugin>
   </plugins>
 </build>
```

Once configured in your project, the build log will contain messages similar to

```
[INFO] --- clm-maven-plugin:2.6.0-01:attach (default) @ test-app ---
[INFO] Starting scan...
[INFO] Scanning ...plexus-utils-3.0.jar
[INFO] Scanning ...maven-settings-3.0.jar...
[INFO] Scanning target/test-app-1.0-SNAPSHOT.jar...
[INFO] Saved module scan to /opt/test-app/target/sonatype-clm/scan.xml.gz
```

The attachment of the `scan.xml.gz` file as a build artifact causes it to be stored in the local repository as well as the deployment repository manager or the Nexus staging repository ending with `-sonatype-clm-scan.xml.gz`. This file will be picked up by Sonatype Nexus CLM Edition and used in the policy analysis during the staging process. It improves the analysis since Sonatype CLM for Maven is able to create a complete dependency list rather than relying on binary build artifacts.

24.4 Using Sonatype CLM for Maven with Other IDEs

While the integration with Eclipse offered by Sonatype CLM for IDE is the most powerful tooling for developers available, user of other popular integrated development environments are not left without support. All common Java IDEs have powerful integration with Apache Maven and therefore can be used together with Sonatype CLM for Maven to evaluate projects against your Sonatype CLM server.

This chapter showcases the integration with IntelliJ IDEA from JetBrains and NetBeans IDE from Oracle.

24.4.1 Maven Plugin Setup

In our example setup for the usage with other IDE's we are going to add a plugin configuration for Sonatype CLM for Maven into the `pom.xml` file of the project we want to analyze as documented in Example Configuration of Sonatype CLM for Maven. This configuration defines the `serverUrl` of the CLM server to be contacted for the evaluation, the `applicationId` used to identify the application in the CLM server to evaluate against and the `stage` configuration to use for the evaluation.

Example Configuration of Sonatype CLM for Maven

```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
```

```
        <groupId>com.sonatype.clm</groupId>
        <artifactId>clm-maven-plugin</artifactId>
        <version>2.6.0-01</version>
        <configuration>
          <serverUrl>http://localhost:8070</serverUrl>
          <applicationId>test</applicationId>
          <stage>develop</stage>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

With this configuration in place a user can kick off an evaluation with the command line `mvn package clm:evaluate`.

This will result in an output detailing the components to be analyzed, any policy violations and a link to the resulting report in the Sonatype CLM server.

Note
To speed the build up you can skip the test compilation and execution by passing `-Dmaven.test.skip=true` on the command line invocation, since it is not needed for the CLM evaluation.

24.4.2 IntelliJ IDEA

IntelliJ IDEA supports Maven projects natively and you can simply open a project in the IDE by opening the `pom.xml` file.

Once your project is opened and you have added the plugin configuration for Sonatype CLM for Maven from Example Configuration of Sonatype CLM for Maven, you can create a configuration to run the desired Maven command.

Select *Edit Configurations* from the *Run* menu, press the + button and select *Maven* to add a new configuration. Enter the command line and other desired details as displayed in Figure 24.1.

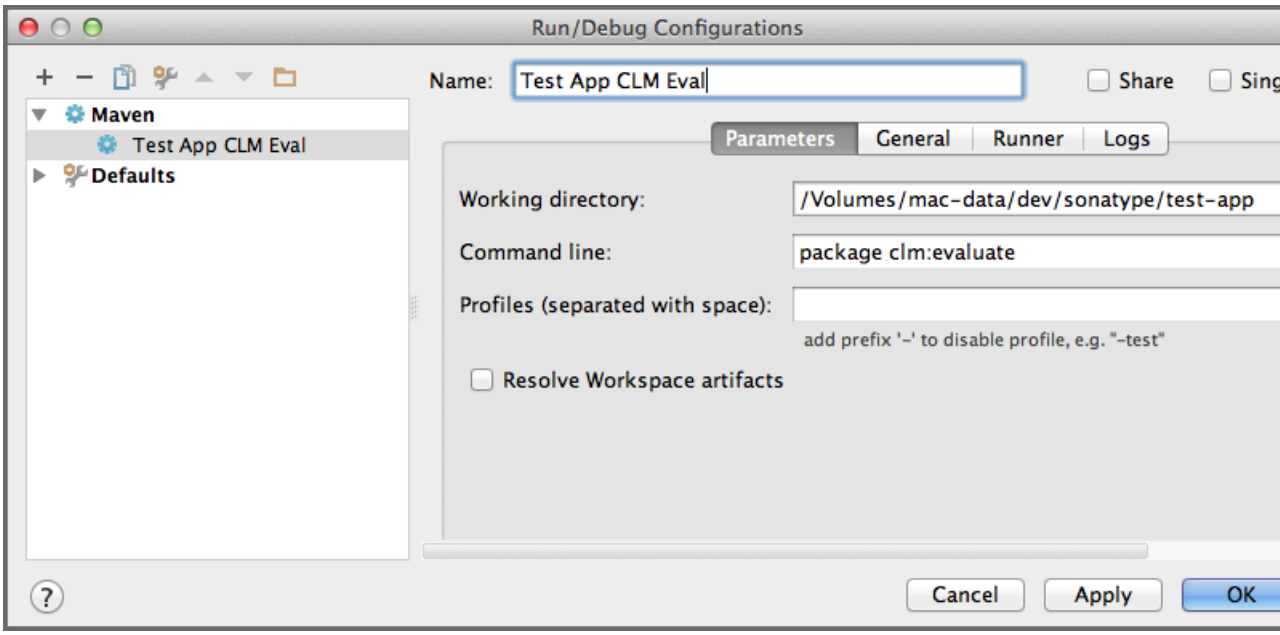


Figure 24.1: Creating a Maven Run Configuration for a CLM Evaluation in IntelliJ

After pressing *OK* in the dialog the new configuration will be available in the run configuration drop down as well the *Maven Projects* view. You can open the view using the *View* menu, selecting *Tools Window* and pressing *Maven Projects*. You will see the window appear in the IDE looking similar to Figure 24.2. It displays the run configuration selector with the green play button on the top as well as the Maven project with the CLM evaluation run configuration.

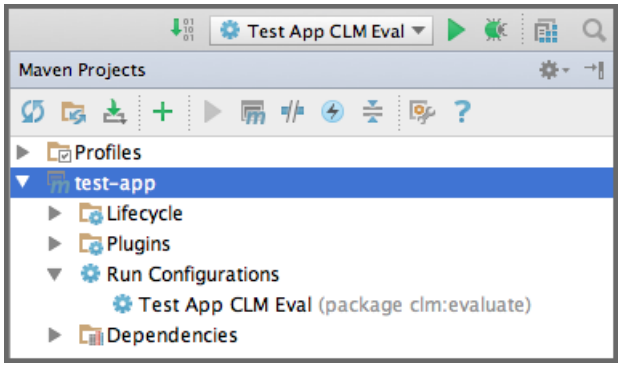


Figure 24.2: Maven Projects View with the CLM Evalulation Run Configuration in IntelliJ

You can press the green play button beside the run configuration, or the configuration entry itself in the *Maven Projects* window, to start a build. The build will run in an embedded console window in the IDE as displayed in Figure 24.3, and show all the output from the Maven build including any error messages and the link to the produced report in the Sonatype CLM server. Policy violations can be configured to result in a build failure.

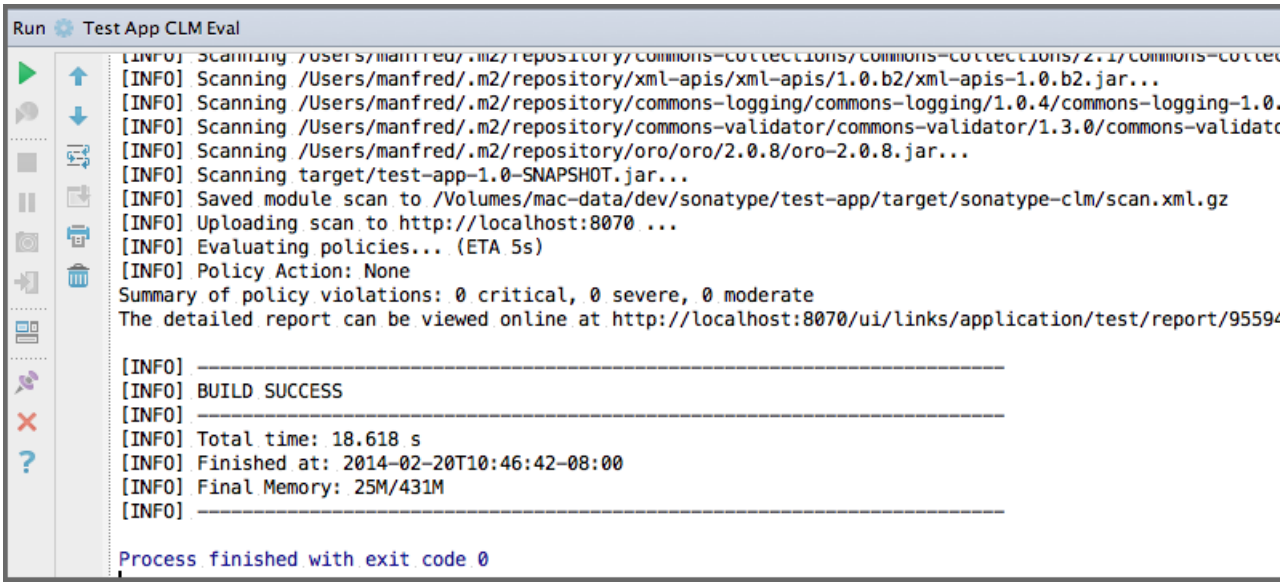


Figure 24.3: CLM for Maven Output in the Run Console in IntelliJ

24.4.3 NetBeans IDE

NetBeans IDE supports Maven projects natively and you can simply open a project in the IDE by choosing *Open Project* from the *File* menu and navigating to the directory that contains your project.

Once your project is opened, you can expand the *Project Files* section in the *Projects* window as displayed in Figure 24.4. Double-click on the `pom.xml` file and add the plugin configuration for Sonatype CLM for Maven from Example Configuration of Sonatype CLM for Maven.

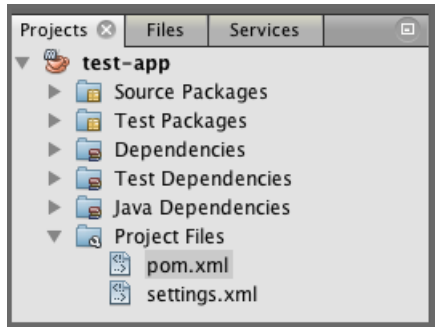


Figure 24.4: Project View with the `pom.xml` in NetBeans

If you right-click on the `pom.xml` file, you can choose *Run Maven* and *Goals*, to display the dialog displayed in Figure 24.5. Enter the configuration as displayed and don't forget to select *Remember as:* providing a name. This will allow you to simply start this defined configuration from the *Run Maven* context menu of the `pom.xml` file.

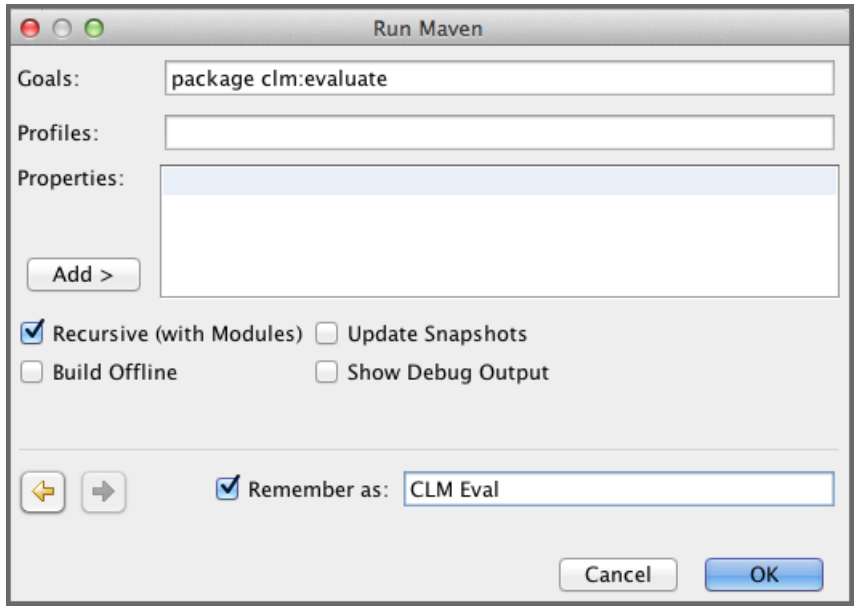


Figure 24.5: Maven Goal Setup for a CLM Evaluation in NetBeans

After pressing *OK* the defined Maven execution will start and display the output including any error messages and the link to the produced report in the Sonatype CLM server in the Output window displayed in Figure 24.6. Policy violations can be configured to result in a build failure.

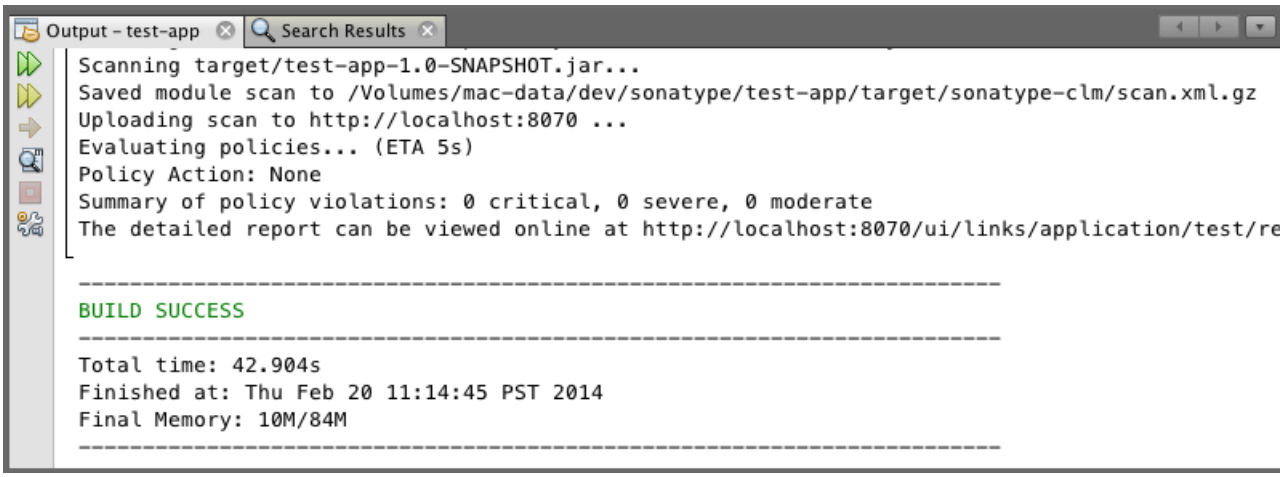


Figure 24.6: CLM for Maven Output in the Output Window in NetBeans

Chapter 25

REST APIs

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

Using REST API calls, the IQ Server provides functionality to create and update applications, as well as retrieve values for policy violations.

These APIs have been designed for system-to-system functionality; however, examples are provided using the HTTP client cURL. Following along, you can initiate the described API REST request via the command line tool.

Warning
REST APIs are versioned. This document represents the most recent version. If you plan to use any of these, we highly recommend updating to the latest version of the IQ Server to ensure compatibility.

25.1 Component Search REST APIs (v2)

The Component Search API allows you to find a particular component, as well as get information back about that component. Using GET requests it allows you to retrieve component information such as application ID, application name, report URL, component hash, component coordinates, and the highest threat level of the policy violations (for the found component).

Below, we've provided an example of the GET request. We've done this using the HTTP client cURL. Of course, you could use any HTTP client tool. Additionally, to help demonstrate use of the API, we've broken out the various pieces for this request and provided an example of data that is retrieved.

Note
Compared to the other APIs, Component Search is fairly simple. However, you should have some basic information about the component coordinates, as well as the stage (e.g. Build) where the component was analyzed.

Searching for Components First, make sure your IQ Server is started. Also, as we mentioned, you will need to have evaluated at least one application. With those two things completed, let's take a look at the GET API.

```
GET /api/v2/search/component
```

Now, in addition to this, you will need to set the stage, and then add your search parameters. Let's take a look at stage first.

Stage

Typically the stage represents the development lifecycle of your product. There are four stages that are currently supported. These include:

- `build`
- `stage`
- `stage-release`
- `operate`

Entering any of these for the stage ID will pull from that specific stage's evaluation data.

Next up, you need to set the component search parameters using any combination of these options:

Parameters

- hash - the component hash (e.g. hash=1234567890).
- componentIdentifier - this is an object that contains the the coordinates of the component and its format.
- format - this is the format of the component (e.g. "maven").
- coordinates - this object contains the name/value pairs for identifying coordinates (e.g. artifactId, groupId, and version).

Now, let's look at an example. Consider a case where we wanted to find all components within the group ID "tomcat", for any applications evaluated during the Build stage. Using the information above, as well as cURL and an encoded URL, here is what we would have...

```
curl -u admin:admin123 -X GET
"http://localhost:8070/api/v2/search/component?stageId=
build&componentIdentifier=%7B%22format%22%3A%22maven%22%2C%
22coordinates%22%3A%7B%22groupId%22%3A%22tomcat%22%7D%7D"
```

Of course the above is an encoded URL, so just for a bit of help, here is what a non-encoded URL would look like. This should help you identify the JSON in the example above.

```
"http://localhost:8070/api/v2/search/component?stageId=
build&componentIdentifier={"format":"maven","coordinates"
:{"groupId":"tomcat"}}"
```

Note

Included in our cURL example is the default IQ Server location (localhost:8070) as well as the default username and password for the admin account. Your account credentials may vary, but are necessary in order for the request to be processed. The results provided will be filtered based on the permissions of the credentials you use. For example, if you are not included in at least the developer role for an application, no results will be returned for that application. Given this, some results may be obscured.

Alright, so in our case, the API above produced the following results. If you have any tomcat components, you could expect something similar...

```
{
   "criteria":{
```

```
      "stageId":"build",
      "hash":null,
      "componentIdentifier":{
         "format":"maven",
         "coordinates":{
            "groupId":"tomcat"
         }
      }
   },
   "results":[
      {
         "applicationId":"MyApp-1234",
         "applicationName":"My Application 2",
         "reportUrl":"http://localhost:8070/ui/links/appl
         ication/MyApp-1234/report/c81991938f304f30bc139e
         a13cf93cd5",
         "hash":"1249e25aebb15358bedd",
         "componentIdentifier":{
            "format":"maven",
            "coordinates":{
               "artifactId":"tomcat-util",
               "classifier":"",
               "extension":"jar",
               "groupId":"tomcat",
               "version":"5.5.23"
            }
         }
```


Note
The data above was formatted to make it a bit more readable.

Using Wildcards Of course, you may come across instances where you want to produce more results with less specific component details. If this is the case, the Component Search API does support the use of wildcards when searching using the GAV (coordinates).

If you are familiar with the coordinates policy condition, it follows the exact same logic. You can read more on the Coordinates condition in Step Five in the Policy Management chapter.

25.2 Component Details API (v2)

The Component Details API provides all available (to Sonatype) security vulnerability, license data, age, and popularity information for a specified component. What is not included, is any information related to policy violations for an evaluated application.

Note
If you are looking for component information for a component that has been evaluated as part of an application, please see the Component Details by Report API.

This API uses **POST** REST resource

Below, we have provided a step-by-step example using the HTTP client cURL, though any HTTP client could be used.

Step 1 - Get the component HASH or component identifier Depending on the type of component, and the information you have, the API allows you to specify the component hash, the component identifier, or both. In our example we'll be searching using Maven coordinates.

Note
If desired you can specify more than one component.

Step 2 - Submit the specified component to retrieve details First let's take a look at the POST resource:

```
POST api/v2/components/details
```

You will also need to include JSON data specifying the component information you are providing.

```
{
   "components":{
      "hash":null,
      "componentIdentifier":{
         "format":"maven",
         "coordinates":{
            "artifactId":"tomcat-util",
```

```
            "extension":"jar",
            "groupId":"tomcat",
            "version":"5.5.23"
         }
      }
   }
```

Putting this together with the cURL command, as well as including the IQ Server URL for the POST resource path, you should have something that looks like this:

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d'{" ↩
    components":[{"hash": null,"componentIdentifier": {"format":"maven"," ↩
    coordinates": {"artifactId":"tomcat-util","extension":"jar","groupId":" ↩
    tomcat","version":"5.5.23"}}}]}' 'http://localhost:8070/api/v2/ ↩
    components/details'
```

The IQ Server will then respond with the component details. An example is provided below.

```
{
   "componentDetails":[
      {
         "component":{
            "hash":"1249e25aebb15358bedd",
            "componentIdentifier":{
               "format":"maven",
               "coordinates":{
                  "artifactId":"tomcat-util",
                  "classifier":"",
                  "extension":"jar",
                  "groupId":"tomcat",
                  "version":"5.5.23"
               }
            }
         },
         "matchState":"exact",
         "catalogDate":"2008-01-29T01:45:22.000-05:00",
         "relativePopularity":100,
         "licenseData":{
            "declaredLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "observedLicenses":[
               {
                  "licenseId":"No-Sources",
```

```
                "licenseName":"No Sources"
             }
          ]
       },
       "securityData":{
          "securityIssues":[
             {
                "source":"cve",
                "reference":"CVE-2007-3385",
                "severity":4.3,
                "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE ↩
                   -2007-3385",
                "threatCategory":"severe"
             },
             {
                "source":"cve",
                "reference":"CVE-2007-5333",
                "severity":5.0,
                "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE ↩
                   -2007-5333",
                "threatCategory":"severe"
             },
             {
                "source":"cve",
                "reference":"CVE-2011-2526",
                "severity":4.4,
                "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE ↩
                   -2011-2526",
                "threatCategory":"severe"
             },
             {
                "source":"cve",
                "reference":"CVE-2012-0022",
                "severity":5.0,
                "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE ↩
                   -2012-0022",
                "threatCategory":"severe"
             },
             {
                "source":"osvdb",
                "reference":"37071",
                "severity":4.3,
                "url":"http://osvdb.org/37071",
                "threatCategory":"severe"
             },
             {
                "source":"osvdb",
                "reference":"41435",
                "severity":5.0,
```

```
                    "url":"http://osvdb.org/41435",
                    "threatCategory":"severe"
                },
                {
                    "source":"osvdb",
                    "reference":"73797",
                    "severity":4.4,
                    "url":"http://osvdb.org/73797",
                    "threatCategory":"severe"
                },
                {
                    "source":"osvdb",
                    "reference":"73798",
                    "severity":4.4,
                    "url":"http://osvdb.org/73798",
                    "threatCategory":"severe"
                },
                {
                    "source":"osvdb",
                    "reference":"78573",
                    "severity":5.0,
                    "url":"http://osvdb.org/78573",
                    "threatCategory":"severe"
                }
            ]
        }
    }
  ]
}
```


25.3 Component Evaluation REST APIs (v2)

The Component Evaluation REST API allows for a single component, or multiple components, to be evaluated against a specific application and the associated policies.

This API uses the following REST resources:

- POST - to submit a component or list of components, as well as the application containing the polices the component(s) will be evaluated against.
- GET - to check the status of the evaluation and retrieve the results when completed.

For the API we provide a step-by-step example using the HTTP client cURL, though any HTTP client tool could be used. In addition we'll reference other APIs such as the one required to obtain an application's internal ID.

Step 1 - Get Component Information First, you need to decide which components you want to evaluate. You'll need a bit of component information in the form of the GAVE (Group, Artifact, Version, and Extension) or the hash for each component you wish to evaluate.

There are a variety of ways to get the GAVE, the most common is to get the information from the repository that houses the component. For example, The Central Repository has the GAVE for all components stored there.

If you wish to use the hash, Sonatype uses sha1 for all components it processes.

Tip

Only components known to Sonatype will provide security, license, and popularity data related to policies. However, depending on the policies for the application you choose to evaluate the component against, other policy conditions, such as match state, will help in determining if the component is known to your instance of the IQ Server.

Step 2 - Get the Application ID Next, you will need pick the application you want to evaluate your component against. This begins with obtaining the application's public ID, which is used to retrieve the internal application ID.

The public ID can be found via the IQ Server GUI by navigating to the respective application and copying the application ID, which is located just below the application name.

This is done using the following GET REST resource from our application API:

```
GET /api/v2/applications?publicId={YourPublicId}
```

Using the cURL command, it would look like this:

```
curl -u admin:admin123 -X GET
'http://localhost:8070/api/v2/applications?publicId=MyApplicationID'
```

If you like, you can also return multiple applications by appending additional `&publicId={SomeOtherPublicId}` to the command above. Alternatively, if desired, all applications can always be returned

by omitting the reference to the public id. This will give the public ID and internal application ID for all applications.

information will be returned (unique to your application). This has been formatted for readability:


```
{
    "applications": [
        {
            "id": "4537e6fe68c24dd5ac83efd97d4fc2f4",
            "publicId": "MyApplicationID",
            "name": "MyApplication",
            "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
            "contactUserName": "NewAppContact",
            "applicationTags": [
                {
                    "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
                    "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
                    "applicationId": "4bb67dcfc86344e3a483832f8c496419"
                }
            ]
        }
    ]
}
```


From the information returned above, make note of the `id`. This is the internal application ID.

Step 3 Submit Component for Evaluation Alright, by now you should have either the GAVE or hash for your component, as well as the internal application ID. We'll put this information together using the POST REST resource:

```
POST /api/v2/evaluation/applications/{applicationInternalId}
```

Added to this will be a JSON formatted body:


```
{
   "hash":null,
   "componentIdentifier":{
      "format":"maven",
      "coordinates":{
         "artifactId":"commons-fileupload",
         "groupId":"commons-fileupload",
         "version":"1.2.2",
         "extension":"jar"
      }
   }
}
```

Tip
If desired, multiple components can be included, but remember, the extension is required when entering components by GAVE.

Together, this should look like this:

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json"
-d'{"components": [{"hash": null,"componentIdentifier": {"format": "maven ↩
    ","coordinates": {"artifactId": "commons-fileupload","groupId": " ↩
    commons-fileupload","version": "1.2.2","extension":"jar"}}}]}' 'http:// ↩
    localhost:8070/api/v2/evaluation/applications/
529b7f71bb714eca8955e5d66687ae2c'
```

A successful POST will result in JSON formatted data providing a confirmation that the evaluation was submitted.

```
{
   "resultId":"917e0c5e92a646a5b8879a40890078bc",
   "submittedDate":"2015-02-02T10:43:22.975-05:00",
   "applicationId":"529b7f71bb714eca8955e5d66687ae2c",
   "resultsUrl":"api/v2/evaluation/applications/
   529b7f71bb714eca8955e5d66687ae2c/results/
   917e0c5e92a646a5b8879a40890078bc"
}
```

Be sure to make note of *resultID* and "applicationID". These will be used to to check and retrieve results. If you are following along with cURL, you can simply copy the *resultsURL*.

Step 4 Get Evaluation Results Now, depending on how many components you evaluated, the IQ Server may need some time to process them. You can check the status and obtain results using the following GET REST Resource:

```
GET /api/v2/evaluation/applications/{applicationInternalId}/results/{ ↩
    resultId}
```

In our example, we're going to use the *resultsURL* and as before, cURL:

```
curl -u admin:admin123 -X GET
'http://localhost:8070/api/v2/evaluation/applications/
529b7f71bb714eca8955e5d66687ae2c/results/
917e0c5e92a646a5b8879a40890078bc'
```

If the report is ready, you will receive results similar to these below:

Note
If the report is not ready, you will receive a (404) error.


```
{
   "submittedDate":"2015-02-13T18:01:42.082-05:00",
   "evaluationDate":"2015-02-13T18:01:42.084-05:00",
   "applicationId":"529b7f71bb714eca8955e5d66687ae2c",
   "results":[
      {
         "component":{
            "hash":null,
            "componentIdentifier":{
               "format":"maven",
               "coordinates":{
                  "artifactId":"commons-fileupload",
                  "extension":"jar",
                  "groupId":"commons-fileupload",
                  "version":"1.2.2"
               }
            },
            "proprietary":false
         },
         "matchState":"exact",
         "catalogDate":"2010-07-29T16:41:12.000-04:00",
         "licenseData":{
            "declaredLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "observedLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "overriddenLicenses":[

            ],
            "status":"Open"
         },
         "securityData":{
            "securityIssues":[
```

```
            {
               "source":"cve",
               "reference":"CVE-2013-2186",
               "severity":7.5,
               "status":"Open",
               "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE ↩
                  -2013-2186",
               "threatCategory":"critical"
            },
            {
               "source":"osvdb",
               "reference":"98703",
               "severity":7.5,
               "status":"Open",
               "url":"http://osvdb.org/98703",
               "threatCategory":"critical"
            },
            {
               "source":"cve",
               "reference":"CVE-2014-0050",
               "severity":5.0,
               "status":"Open",
               "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE ↩
                  -2014-0050",
               "threatCategory":"severe"
            },
            {
               "source":"osvdb",
               "reference":"102945",
               "severity":5.0,
               "status":"Open",
               "url":"http://osvdb.org/102945",
               "threatCategory":"severe"
            }
         ]
      },
      "policyData":{
         "policyViolations":[
            {
               "policyId":"0ccc3321662d491c86b32554ea84e4e9",
               "policyName":"Security-High",
               "threatLevel":9,
               "constraintViolations":[
                  {
                     "constraintId":"d2dfc91f3bf4460ba0da3f7201e4adb7",
                     "constraintName":"CVSS >=7 and <10",
                     "reasons":[
                        {
                           "reason":"Found 2 Security Vulnerabilities  ↩
```

```
                                    with Severity >= 7"
                                },
                                {
                                    "reason":"Found 4 Security Vulnerabilities  ↩
                                        with Severity < 10"
                                },
                                {
                                    "reason":"Found 4 Security Vulnerabilities  ↩
                                        with Status OPEN"
                                }
                            ]
                        }
                    ]
                },
                {
                    "policyId":"0f1b57e6788c4e9bb0f65cde3b6e9f1c",
                    "policyName":"Security-Medium",
                    "threatLevel":7,
                    "constraintViolations":[
                        {
                            "constraintId":"12a71811de694876b51e4c06c10bad76",
                            "constraintName":"CVSS >=4 and <7",
                            "reasons":[
                                {
                                    "reason":"Found 4 Security Vulnerabilities  ↩
                                        with Severity >= 4"
                                },
                                {
                                    "reason":"Found 2 Security Vulnerabilities  ↩
                                        with Severity < 7"
                                },
                                {
                                    "reason":"Found 4 Security Vulnerabilities  ↩
                                        with Status OPEN"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "isError":false,
    "errorMessage":null
}
```

Note
The last bit of information indicates if there were any errors encountered during the evaluation. In this example, there were no errors.

25.4 Application REST APIs (v2)

The primary functions of the Application REST APIs are creating, updating, and deleting applications. In addition to this, tags can be added and users / groups mapped to roles (permissions) for the application.

The currently available APIs include:

- GET - used to retrieve information, such as a list of organizations, their internal IDs, and their tags. This is also used for retrieving roles and mapping those roles to an application.
- POST - used to create applications.
- PUT - used for mapping roles to the application, as well as making any necessary changes to modifiable application properties (i.e. name, tags, or contact).
- DELETE - used to delete applications.

As mentioned in the introduction we will provide both the API, as well as an example using the HTTP client cURL. Additionally, to help demonstrate use of the APIs, we've approached this in a step-by-step manner that will start with gathering the necessary information to create and set permissions for an application.

Tip
If you already have an application, and wish to make an update, you can jump to the final step now.

Step 1 - Get the Organization ID In order to create an application, it must have a parent organization. Thus, we must start with our GET API call...

```
GET  /api/v2/organizations
```

and find the organization ID. Additionally, any tags available to be applied to the application will be provided. To follow along using cURL, enter the following command:

```
curl -u admin:admin123 -X GET
'http://localhost:8070/api/v2/organizations'
```

This will produce a list of your organizations and associated information in a JSON format. Here is an example of what might be returned.

```
{
    "organizations": [
        {
            "id": "36d7e629462a4038b581488c347959bc",
            "name": "My Organization",
            "tags": [ ]
        },
        {
            "id": "f48b5344fa204a4c88df96b2d455d521",
            "name": "My Organization 2",
            "tags": [
                {
                    "id": "cd8fd2f4f289445b8975092e7d3045ba",
                    "name": "Distributed",
                    "description": "Applications that are provided for ↩
                        consumption outside the company",
                    "color": "yellow"
                },
                {
                    "id": "004d789684834f7c889c8b186a5ff24b",
                    "name": "Hosted",
                    "description": "Applications that are hosted such as ↩
                        services or software as a service.",
                    "color": "grey"
                },
                {
                    "id": "da9e09887c754157a2113831ae7e99ac",
                    "name": "Internal",
                    "description": "Applications that are used only by ↩
                        your employees",
                    "color": "green"
                }
            ]
        }
    ]
}
```

id

This is the internal id for the organization.

name
This is the name of the organization, and is visible in the IQ Server GUI.

tags
Tags represent identifying characteristic for an application. Tags are created at the organization level, and then applied to applications. Policies can then select which tags, and in turn applications, the policy will be evaluated against.

id (tags)
the internal id for the tag. This will be used when creating the application.

name (tags)
the name of the tag, which is visible in the IQ Server GUI.

description (tags)
each tag is required to have a description. This description provides information related to the type of application it should be applied to.

In many cases, you will have more than one organization, as does our example.

Step 2 - Create an Application To create an application, you need several pieces of information. As we've already mentioned, you will need the organization ID, but you will also need:

"publicId" (application ID)
This is the application ID for the application. In the IQ Server GUI this is represented by the "Application" field. It must be unique.

"name" (application name)
This is the name of the application. In the IQ Server GUI this corresponds to the "Application Name" field. It must be unique.

"contactUserName" (application contact)
In the IQ Server GUI, this corresponds to the contact field for the application. The value entered here should represent the user name. If you are using LDAP and have configured it to work with the IQ Server, that user name can be used here.

"tagId" (internal tag ID)
As mentioned previously, a tag applied to an application will ensure it is evaluated by any policies that have selected the corresponding tag. The tag ID is returned as part of the organization information.

Tip
The application contact and tags are optional, but it is generally recommended to include them.

We'll be using the POST API call...

```
POST  /api/v2/applications
```

and appending the information mentioned above into a JSON-formatted body. Here is an example of the body that's been formatted for easier review:

```
{
    "publicId": "MyApplicationID",
    "name": "MyFirstApplication",
    "organizationId": "f48b5344fa204a4c88df96b2d455d521",
    "contactUserName": "AppContact",
    "applicationTags": [
        {
            "tagId": "cd8fd2f4f289445b8975092e7d3045ba"
        }
    ]
}
```

Putting this all together, and using our cURL example, you should enter the following command:

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json"
-d '{"publicId": "MyApplicationID","name": "MyFirstApplication"," ↩
    organizationId":"f48b5344fa204a4c88df96b2d455d521
","contactUserName":"AppContact","applicationTags": [{"tagId":" ↩
    cd8fd2f4f289445b8975092e7d3045ba"}]}' 'http://localhost:8070/api/v2/ ↩
    applications'
```

If your application creation was successful, the system will respond with the following:

```
{
    "id": "4bb67dcfc86344e3a483832f8c496419",
    "publicId": "MyApplicationID",
    "name": "MyFirstApplication",
    "organizationId": "f48b5344fa204a4c88df96b2d455d521",
    "contactUserName": "AppContact",
    "applicationTags": [
        {
            "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
            "tagId": "cd8fd2f4f289445b8975092e7d3045ba",
            "applicationId": "4bb67dcfc86344e3a483832f8c496419"
        }
    ]
}
```

Tip
Be sure to make note of the "applicationId". This is the internal application ID that will be used in the next steps when finding roles and mapping a user to them.

Step 3 - Find the Currently Available Roles Before mapping any roles, it's a good idea to take a look at all the available roles for applications. This can be done using a simple GET:

```
GET /api/v2/applications/roles
```

This returns all the available roles for all applications:

```
{
    "roles": [
        {
            "id": "1da70fae1fd54d6cb7999871ebdb9a36",
            "name": "Developer",
            "description": "Allows to evaluate policies."
        },
        {
            "id": "1cddabf7fdaa47d6833454af10e0a3ef",
            "name": "Owner",
            "description": "Allows to manage policies."
        }
    ]
}
```

id
This is the internal ID for the role. There is no corresponding field in the IQ Server GUI.

name
This is the name of the role, which is exactly the same as the *Role Name* displayed in the IQ Server GUI. As we mentioned, there are two role names in the example above `Developer` and `Owner`.

description
This provides a description of the level of access the role grants.

As you may notice, there are two roles available for applications at this time. Before proceeding to the next step, take note of the `roleId` (in this example we will be adding an owner). This will be needed for mapping a user to the specific role.

Step 4 - Map Users to Roles To map users to roles, we'll need the `id` (application ID) from our successful application creation, the `roleId` for the role returned above that you will be mapping a user to, and the following information:

type
: The type of user, which at this time, can only be either `USER` or `GROUP`.

userOrGroupName
: This is the username for the user you want to add to the role. If you are using LDAP and have configured it to work with the IQ Server, the LDAP user or group name can be used here.

For this step, we'll be using the following PUT API call...

```
PUT /api/v2/applications/{applicationInternalId}/roleMembers
```

with an appended JSON-formatted body. Here is an example of the body that's been formatted for easier review:

```
{
    "memberMappings": [
        {
            "roleId": "1cddabf7fdaa47d6833454af10e0a3ef",
            "members": [
                {
                    "type": "USER",
                    "userOrGroupName": "user-owner"
                }
            ]
        }
    ]
}
```

Putting this altogether, and using our cURL example, you should enter the following command:

```
curl -u admin:admin123 -X PUT -H "Content-Type: application/json"
-d'{"memberMappings": [{"roleId": "1da70fae1fd54d6cb7999871ebdb9a36"," ↩
    members": [{"type": "USER","userOrGroupName": "user-b"}]}]}' 'http:// ↩
    localhost:8070/api/v2/applications/
4bb67dcfc86344e3a483832f8c496419/roleMembers'
```

As before, standard HTTP response codes will be returned.

Tip
You can map multiple roles in a single PUT.

Step 5 - Update Application Information

This final step isn't a requirement to complete an application, but it will be necessary for any application you want to update.

Before updating the application, you will need the `id` for the application. If you haven't recorded this, you can obtain it by using the `publicId`. To retrieve the information for the application use the following GET API call...

```
GET /api/v2/applications?publicId={YourPublicId}
```

Tip
All applications can always be returned by omitting the reference to the public id.

Using the cURL command, it would look like this:

```
curl -u admin:admin123 -X GET
'http://localhost:8070/api/v2/applications?publicId=MyApplicationID'
```

If you like, you can also return multiple applications by appending additional `&publicId={SomeOtherPublicId}` to the command above.

Placing your application's publicId where `{YourPublicId}` is, the following information will be returned (unique to your application). This has been formatted for readability:

```
{
    "applications": [
        {
            "id": "4bb67dcfc86344e3a483832f8c496419",
            "publicId": "MyApplicationID",
            "name": "MySecondApplication",
            "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
            "contactUserName": "NewAppContact",
            "applicationTags": [
                {
```

```
                    "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
                    "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
                    "applicationId": "4bb67dcfc86344e3a483832f8c496419"
                }
            ]
        }
    ]
}
```

From the information returned above, make note of the following:

- "id"
- "organizationId"

The information for an application that can be updated, follows the same rules as editing an application via the IQ Server GUI. In addition, you will also need any information you will be editing, which can include any or all of the following:

- "publicId" (see Important note below)
- "name"
- "contactUserName"
- "applicationTags" (tagID)

Note
If you are looking to update the role mapping for the application, simply follow the instructions from step 4.

Now, you should be ready to apply your changes. Before we move on to the command, let's take a look at a sample body for the attached JSON file:

```
{
    "id": "4bb67dcfc86344e3a483832f8c496419",
    "publicId": "MyApplicationID",
    "name": "MySecondApplication",
    "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
    "contactUserName": "NewAppContact",
```

```
    "applicationTags": [
        {
            "tagId": "cfea8fa79df64283bd64e5b6b624ba48"
        }
    ]
}
```

Using cURL enter the following command:

```
curl -u admin:admin123 -X PUT -H "Content-Type: application/json"
-d '{"id":"4bb67dcfc86344e3a483832f8c496419","publicId":
"MyApplicationID","name":"MySecondApplication",
"organizationId":"bb41817bd3e2403a8a52fe8bcd8fe25a",
"contactUserName":"NewAppContact","applicationTags":
[{"tagId":"cfea8fa79df64283bd64e5b6b624ba48"}]}' 'http://localhost:8070/ ↩
    api/v2/applications/
4bb67dcfc86344e3a483832f8c496419'
```

If your update is successful, you will see something similar to the formatted JSON file below.

```
{
    "id": "4bb67dcfc86344e3a483832f8c496419",
    "publicId": "MyApplicationID",
    "name": "MySecondApplication",
    "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
    "contactUserName": "NewAppContact",
    "applicationTags": [
        {
            "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
            "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
            "applicationId": "4bb67dcfc86344e3a483832f8c496419"
        }
    ]
}
```

Important
If you change an application PublicId, remember to update your integration tools to use the new value.

25.4.1 Deleting an Application

In case you wish to delete an application permanently, the Application REST APIs offer the ability to do so.

Important
Deleting an application is a destructive action that will permanently remove an application and all data associated with it. Ensure first that you are deleting the correct application and that deleting the application will not adversely affect any integrations that depend on the presence of the application.

Before deleting the application, you will need the `id` for the application. See updating an application for more information on how to obtain the application `id` using the application `publicId`.

The following DELETE API call is available for deleting an application...

```
DELETE /api/v2/applications/{applicationInternalId}
```

Using cURL:

```
curl -u admin:admin123 -X DELETE 'http://localhost:8070/api/v2/ ↩
    applications/4bb67dcfc86344e3a483832f8c496419'
```

Standard HTTP status codes will be returned in response to a request to delete an application.

25.5 Violation REST API (v2)

The Policy Violation REST APIs allow you to access and extract policy violations gathered during the evaluation of applications. In most cases the desire for getting to this data is to integrate into other tools your company may have. For example you may have a specific dashboard or reporting application that should have this data.

Whatever the case, just as with the other REST APIs, this is all done using REST API calls. For accessing policy violation information the following API is used:

GET
: Used to retrieve policy information, such as a list of policy ids as well as a list of violations based on a specific Policy ID, or list of IDs.

As mentioned previously, we will provide both the API, as well as examples using the HTTP client cURL. This is only for demonstration purposes and displaying the necessary input, and desired output.

Additionally, to help demonstrate this, we've approached this in a step-by-step manner that will start with gathering policy ids, and then requesting the violations.

Before You Get Started
: As with the other REST APIs, you will need a username and password to interface with the IQ Server. In addition, because access to this data is granted based on the roles (permissions) you have set up, you may wish to create one specifically for this process.

  Other than this, the only piece you may need in order to follow along with our instructions is cURL, or a comparable HTTP client.

Step 1 - Get the Policy IDs To access policy violation information you need the Policy ID(s). For this reason, we start with the GET API call...

```
GET  /api/v2/policies/
```

which will return a list of all Policy IDs. To follow along using cURL, enter the following command:

```
curl -u admin:admin123 -X GET
'http://localhost:8070/api/v2/policies'
```

The action above will produce a list of your policies in a JSON format. Here is an example of what might be returned.

```
{
    "policies": [
        {
            "id": "6984017845c645b0ad0c95401ad4f17d",
            "name": "My Application Policy",
            "ownerId": "36d7e629462a4038b581488c347959bc",
            "ownerType": "APPLICATION",
            "threatLevel": 5,
            "policyType": "quality"
        },
    ]
}
```

Note
As you can see above, we've used the admin user which is shipped with the IQ Server, as well as the default IQ Server location. The user you use may differ depending on your configuration.

id
This is the internal id for the policy.

name
This is the name of the policy, and is visible in the IQ Server GUI.

ownerId
This is the internal id for the organization or application that the policy resides in, and is not visible within the IQ Server GUI.

ownerType
This indicates whether the policy is for an organization or application.

threatLevel
This is the threat level that is set for the policy.

policyType
This is the policy type, which is based on the conditions used in the policy.

Tip
In many cases, you will have many policies, especially if you are retrieving information for an account that has access to many applications and/or organization.

Step 2 - Get the Policy Violations Now that you have the Policy IDs, they can be used to gather a list of policy violations. To do this, you will need the Policy IDs you retrieved from step one. For example:

```
"id": "6984017845c645b0ad0c95401ad4f17d"
```

Note
Policy IDs are unique, and what is used above is just an example.

Slightly different from before, we will use the GET API call...

```
GET /api/v2/policyViolations?p=policyid1
```

which passes a simple query for Policy IDs. For each policy we want to retrieve violations for, we will include that ID. If desired we can retrieve violations for multiple Policy IDs. To do this, just make sure you add &p="The Policy ID" for each policy you want included. Here is an example of the API with the Policy ID we retrieved:

```
GET /api/v2/policyViolations?p=6984017845c645b0ad0c95401ad4f17d
```

Putting this all together, and using our cURL example, you should enter the following command:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/ ↩
    policyViolations?
p=6984017845c645b0ad0c95401ad4f17d'
```

If your query was successful, the system will respond with something like this:

```
{
   "applicationViolations":[
      {
         "application":{
            "id":"529b7f71bb714eca8955e5d66687ae2c",
            "publicId":"MyAppID1",
            "name":"MyApplications",
            "organizationId":"36d7e629462a4038b581488c347959bc",
            "contactUserName":null
         },
         "policyViolations":[
            {
               "policyId":"6984017845c645b0ad0c95401ad4f17d",
               "policyName":"Security-High",
               "stageId":"build",
               "reportUrl":"ui/links/application/MyAppID1/report/c0ddef
               c4512f42d0bcbe29029e2be117",
               "threatLevel":9,
               "constraintViolations":[
                  {
                     "constraintId":"19011de290b147a38c820ad7bd5c653d",
                     "constraintName":"CVSS >=7 and <10",
                     "reasons":[
                        {
                           "reason":"Found 2 Security Vulnerabilities with ↩
                                Severity >= 7"
                        },
                        {
```

```
                        "reason":"Found 4 Security Vulnerabilities with ↩
                            Severity < 10"
                     },
                     {
                        "reason":"Found 4 Security Vulnerabilities with ↩
                            Status OPEN"
                     }
                  ]
               }
            ],
            "component":{
               "hash":"384faa82e193d4e4b054",
               "componentIdentifier":{
                  "format":"maven",
                  "coordinates":{
                     "artifactId":"tomcat-util",
                     "classifier":"",
                     "extension":"jar",
                     "groupId":"tomcat",
                     "version":"5.5.23"
                  }
               },
               "proprietary":false
            }
         }
      ]
   }
]
}
```


And there you have it, you've just retrieved policy violations. Below, each of the categories of data that is returned, as well as each field, have been described.

application
: This category contains specific information about the application.

  id
  : This is the internal id.

  publicId
  : This is the application ID. In the IQ Server GUI this is represented by the "Application" field.

  name
  : This is the name of the application. In the IQ Server GUI this corresponds to the "Application Name" field.

  organizationId
  : This is the internal id for the organization that the application resides in, and is not visible

within the IQ Server GUI.

contactUserName
This is typically the person in charge of the application. In the IQ Server GUI, it corresponds to the contact field for the application.

policyViolations
This is a subcategory of the application, and provides specific information about the policy and corresponding violations that were found.

policyId
This is the internal id for the policy.

policyName
This is the name of the policy, and is visible in the IQ Server GUI.

stageId
This is the stage in which the policy violation occurred in. It is displayed in various places within the IQ Server GUI, including the associated Application Composition Report.

reportUrl
This is the URL to the Application Composition Report associated with the evaluation that found the listed policy violations.

threatLevel
This the threat level of the policy that was violated.

constraintViolations
This is a subcategory for Policy Violations, and includes all information related to specific constraint that was violated.

constraintId
This is the internal id for the constraint, and is not visible in the IQ Server GUI, or in the associated Application Composition Report.

constraintName
This is the name of the constraint and is visible in the policy area where the policy was created (i.e either the organization or application). It is also displayed in the Application Composition Report and various tools that connect to the IQ Server.

reasons
This is a subcategory of Constraint Violations, and gives the reason why the violation occurred.

reason
The reason is formed by the value(s) for the condition(s) violated. Conditions are visible where the policy was created (i.e either the organization or application). It is also displayed in the Application Composition Report and various tools that connect to the IQ Server.

component
Component is a subcategory of Policy Violations, and includes information about the component(s) causing the violation to occur.

Note
The component field only refers to a single component. If another component violates the same policy, another entry for that in "policyViolations" would be present.

hash
This is the hash value for the component. For example, in the case of Java components, this would be matched to a Java repository (e.g. Central). If you have proprietary components configured, it would be matched against your list of proprietary components.

componentIdentifier
This is simply a container for the component information. It will always include the format and the coordinates.

format
This is the format the component is in, and will determine what type of coordinate information is displayed.

coordinates
This will depend on the format. An example would be Maven, which uses a G : A : E : C : V (Group, Artifact Id, Extension, Classifier, and Version) for the component. In this example, the fields provided are: groupId, artifactId, and version.

25.6 Report-related REST APIs (v2)

The Application Composition Report is created when an evaluation occurs. There are two available REST APIs associated with obtaining information related to these reports:

- Reports URL
- Component Details by Report

Reports URL REST API (v2)

The Reports API provides a summary of an application's most recent reports across the various stages (e.g. Build, Stage Release, and Release). The API consists of using the GET REST resource that is part of both the applications and reports APIs.

To assist in learning to use this API we will provide both the API as well as an example using the HTTP client cURL. We've approached this in a step-by-step manner that will start with gathering the internal application ID and then retrieving the report information.

Note
In our example below, we have isolated retrieving a report for a single application. However, if desired the reports for all applications can also be retrieved. In this case, skip to Step 2, where a corresponding tip has been included to assist.

Step 1 - Get the Application ID First, you will use the application's public ID to retrieve the internal application ID. This is done using the following GET REST resource from our application API...

```
GET /api/v2/applications?publicId={YourPublicId}
```

Tip
All applications can always be returned by omitting the reference to the public ID.

Using the cURL command, it would look like this:

```
curl -u admin:admin123 -X GET
'http://localhost:8070/api/v2/applications?
publicId=MyApplicationID'
```

If you like, you can also return multiple applications by appending additional `&publicId={SomeOtherPublicId}` to the command above.

information will be returned (unique to your application). This has been formatted for readability:

```
{
    "applications": [
        {
            "id": "4537e6fe68c24dd5ac83efd97d4fc2f4",
            "publicId": "MyApplicationID",
            "name": "MyApplication",
            "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
            "contactUserName": "NewAppContact",
            "applicationTags": [
                {
```

```
                    "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
                    "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
                    "applicationId": "4bb67dcfc86344e3a483832f8c496419"
                }
            ]
        }
    ]
}
```


From the information returned above, make note of the `id`. This is the internal application ID.

Note
The public ID can be found via the IQ Server GUI by navigating to the respective application and copying the application ID located just below the application name.

Step 2 - Get Report URLs Next we will take the internal application ID from Step 1 and use that with the GET REST resource for the Reports API. This is done with our GET REST resource associated with our Reports API...

```
GET /api/v2/reports/applications/{applicationInternalId}
```

Tip
If you would like to obtain the report information for all applications, simply leave off the "id" and all report information for all applications will be provided.

Using cURL, it will look like this...

```
curl -u admin:admin123 -X GET
'http://localhost:8070/api/v2/reports/applications/
4bb67dcfc86344e3a483832f8c496419'
```

If any report information is found for the application, you will receive JSON formatted data. An example is included below.


```
[
    {
        "stage": "build",
```

```
        "applicationId": "4537e6fe68c24dd5ac83efd97d4fc2f4",
        "evaluationDate": "2015-01-16T13:14:32.139-05:00",
        "reportHtmlUrl": "ui/links/application/Test123/report/474 ↩
            ca07881554f8fbec168ec25
        d9616a",
        "embeddableReportHtmlUrl": "ui/links/application/Test123/report ↩
            /474ca07881554f8fbec168ec25
        d9616a/embeddable",
        "reportPdfUrl": "ui/links/application/Test123/report/474 ↩
            ca07881554f8fbec168ec25
        d9616a/pdf",
        "reportDataUrl": "api/v2/applications/Test123/reports/474 ↩
            ca07881554f8fbec168ec25
        d9616a"
    },
    {
        "stage": "stage-release",
        "applicationId": "4537e6fe68c24dd5ac83efd97d4fc2f4",
        "evaluationDate": "2014-07-14T15:08:09.501-04:00",
        "reportHtmlUrl": "ui/links/application/Test123/report/2 ↩
            dbfd514e5ad497598086c3e4c
        89c413",
        "embeddableReportHtmlUrl": "ui/links/application/Test123/report/2 ↩
            dbfd514e5ad497598086c3e4c
        89c413/embeddable",
        "reportPdfUrl": "ui/links/application/Test123/report/2 ↩
            dbfd514e5ad497598086c3e4c
        89c413/pdf",
        "reportDataUrl": "api/v2/applications/Test123/reports/2 ↩
            dbfd514e5ad497598086c3e4c
        89c413"
    },
    {
        "stage": "release",
        "applicationId": "4537e6fe68c24dd5ac83efd97d4fc2f4",
        "evaluationDate": "2014-08-31T12:01:46.179-04:00",
        "reportHtmlUrl": "ui/links/application/Test123/report/97 ↩
            cccbeeb58d4aac83264993ef
        6bf4d8",
        "embeddableReportHtmlUrl": "ui/links/application/Test123/report/97 ↩
            cccbeeb58d4aac83264993ef
        6bf4d8/embeddable",
        "reportPdfUrl": "ui/links/application/Test123/report/97 ↩
            cccbeeb58d4aac83264993ef
        6bf4d8/pdf",
        "reportDataUrl": "api/v2/applications/Test123/reports/97 ↩
            cccbeeb58d4aac83264993ef
        6bf4d8"
    }
```

```
]
```

Component Details by Report REST API (v2)

When an application is evaluated, information found during the evaluation is provided via the Application Composition Report. While this is the easiest way to review this information, it can also be exported to a JSON file using the Component Information API.

The API works by sending a REST request to the IQ Server. This request involves using a specially formatted URL and any HTTP client. In the example we have provided we make use of cURL and the command line. In addition, we format the JSON to make it a bit more readable.

Step 1 - Sending the Request First, let's take a look at the GET API we'll be using:

```
GET
http://localhost:8070/api/v2/applications/
[applicationPublicId]/reports/[reportId]
```

As you may have noticed, this API uses a URL specific to the location of the report. There are two pieces of information you will need in order to retrieve the results.

- applicationPublicId - The public ID of the specific application.
- reportId - The ID of the specific report.

There are a variety of ways to retrieve this information, including gathering it by using the Component Search API. However, in our example, we're just going to pull this information from the log output of a an

```
...

********************************************************************************* ↩

[INFO] Policy Action: None
[INFO] CLM stage: build
[INFO] Summary of policy violations: 6 critical, 11 severe, 1 moderate
[INFO] The detailed report can be viewed online at
http://localhost:8070/ui/links/application/
MyApp-1234/report/68b6bdb1573a40eeb4205d890b602525
[INFO]
********************************************************************************* ↩
```

NOTE:

We'll use the link for the report to gather the information we need:

```
http://localhost:8070/ui/links/application/
MyApp-1234/report/68b6bdb1573a40eeb4205d890b602525
```

In the example above, the section of the URL following "application" and before the section "report", is the application's public ID. Similarly, the section after "report" is the report ID.

Note
The report is also presented in a full GUI. To access this, simply paste the link in your browser. However, you will need to be at least a member of the developer group for the application that has been evaluated, or you will not be able to access the report.

Now, we can download the data using our HTTP request tool.

Step 2 - Downloading Component Information Again, in our example we are going to use cURL, though any HTTP client could be used. Here is what our request looks like:

```
curl -u admin:admin123 -X GET
"http://localhost:8070/api/v2/applications/
MyApp-1234/reports/68b6bdb1573a40eeb4205d890b602525"
```

Note
Included in our cURL example is the default username and password for the admin account. Your account credentials may vary, but are necessary in order for the request to be processed. If the username and password provided are not at least within the developer role for the application, an error will be returned.

Looking at the result through a JSON Viewer, you should see something like this:

```
{
   "components":[
      {
         "hash":"1249e25aebb15358bedd",
         "componentIdentifier":{
```

```
            "format":"maven",
            "coordinates":{
               "artifactId":"tomcat-util",
               "groupId":"tomcat",
               "version":"5.5.23"
            }
         },
         "proprietary":false,
         "matchState":"exact",
         "pathnames":[
            "sample-application.zip/tomcat-util-5.5.23.jar"
         ],
         "licenseData":{
            "declaredLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "observedLicenses":[
               {
                  "licenseId":"No-Sources",
                  "licenseName":"No Sources"
               }
            ],
            "overriddenLicenses":[

            ],
            "status":"Open"
         },
         "securityData":{
            "securityIssues":[
               {
                  "source":"cve",
                  "reference":"CVE-2007-3385",
                  "severity":4.3,
                  "status":"Open",
                  "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE ↩
                      -2007-3385",
                  "threatCategory":"severe"
               }
            ]
         }
      }
   ]
}
```

Now that you have access to this data, it can be packaged in a variety of ways. For example, you may want to use this as a bill of materials to include associated license data.

Note
In cases where you want a version of the report without any of the associated IQ Server navigation elements (e.g. for use in a custom tool), use the embeddableReportHtmlUrl.

25.7 Accessing REST APIs via Reverse Proxy Authentication

When authentication is handled by a reverse proxy server as described in the section Reverse Proxy Authentication, API requests that change data, i.e. POST, PUT and DELETE requests, are subject to cross-site request forgery (CSRF) protection. For these requests to be accepted by IQ Server, they need to include the HTTP header `X-CSRF-TOKEN` along with an HTTP cookie named `CLM-CSRF-TOKEN` where both the header and the cookie carry the same value. The specific value chosen is irrelevant, it only needs to be the same for the header and the cookie.

Please refer to the documentation of your respective HTTP client on how to supply the header and cookie. For the cURL tool used in our earlier examples, this can be accomplished as follows:

```
curl --header "X-CSRF-TOKEN: api" --cookie "CLM-CSRF-TOKEN=api" ...
```

Chapter 26

Webhooks

Tip
The topics discussed in this chapter require IQ Server with one of the following licenses: Lifecycle, Firewall, or Auditor.

Webhooks are HTTP callbacks that POST data to defined URLs. They let you build integrations registered to certain events on IQ Server. When a registered event is triggered, an HTTP POST payload is sent to the webhook's defined URL.

26.1 Using Webhooks

Use webhooks to receive notifications about events that happen in IQ Server. When an event occurs - for example, when an application evaluation is completed - IQ Server creates an event object. This object has relevant information about what just happened, such as the type of event and any data associated with the event. IQ Server then sends the event object to defined URLs using HTTP POST requests.

IQ Server lets you use webhooks for policy management, application evaluation, security vulnerability override management, and license override management events.

Each webhook is defined by the following:

- The URL to POST the payload.
- An optional secret key that ensures authenticity of the source.
- Event types subscribed by the webhook.

26.2 Configuring Webhooks

Note
Webhooks can only be created by a System Administrator. Webhooks do not have a permissions model and will send HTTP requests for every event in the system. Be aware that the system consuming the webhooks will have access to all the data provided by the event types.

26.2.1 Creating Webhooks

To create a webhook, perform the following steps:

1. Click the *System Preferences* icon in the IQ Server toolbar and then click *Webhooks*.
2. Click the *Create Webhook* button.
3. Enter the webhook URL.
4. Optional: Enter the webhook secret key.
5. Select one or many 'Event Type's.
6. Click *Create*.

After creating the webhook, it appears in the list of webhooks.

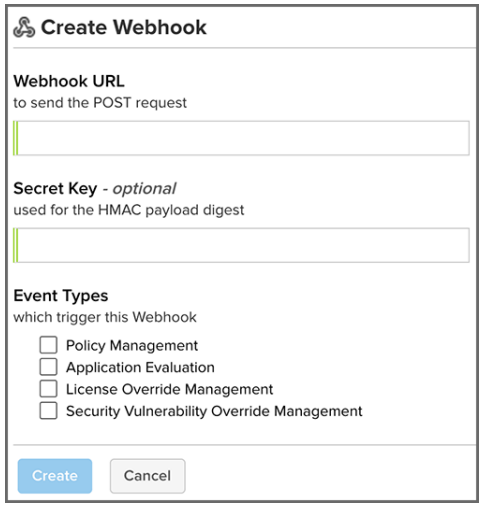


Figure 26.1: Creating Webhooks

26.2.2 Editing Webhooks

To edit a webhook, perform the following steps:

1. Click the *System Preferences* icon in the IQ Server toolbar and then click *Webhooks*.
2. Click the webhook you want to edit.
3. Edit the URL, secret key, or selected event types and then click *Update*.

26.2.3 Deleting Webhooks

To delete a webhook, perform the following steps:

1. Click the *System Preferences* icon in the IQ Server toolbar and then click *Webhooks*.
2. Click the webhook you want to delete.
3. Click *Remove Webhook*.

4. Click *Continue* in the *Remove Webhook* modal.

26.3 Working with HMAC Payloads

If you enable a secret key to generate an HMAC digest, a special header is sent with all of your webhook payloads. This header is *X-Nexus-Webhook-Signature* and ensures that you receive an authentic message.

Webhooks can be consumed easily in node.js. Use the following setup to get started, substituting *foo* for the secret key you configured with your webhook:

Setup in terminal

```
npm init
npm install express
npm install body-parser
echo {\"secretKey\":\"foo\"} > settings.json
```

Note
When verifying the HMAC digest, the HmacDigest value should match the signature value.

Example Webhook Consumer

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const settings = require('./settings.json');
const crypto = require('crypto');

app.use(bodyParser.json());

app.post('/', function(req, res) {
  const body = req.body;
  const signature = req.headers['x-nexus-webhook-signature'];
  var hmacDigest = crypto.createHmac("sha1", settings.secretKey).update( ↩
      JSON.stringify(body)).digest("hex");

  console.log('Webhook received');
  console.log('Headers: ' + JSON.stringify(req.headers));
```

```
  console.log('Body: ' + JSON.stringify(req.body));
  console.log('HmacDigest: ' + hmacDigest);
  console.log('Signature: ' + signature);
  res.send();
});

app.listen(3000, function() {
  console.log('Server running on port 3000.');
});
```

26.4 Example Headers and Payloads

It is important to understand the payload being received. Each event contains special headers that help describe the event.

The following headers are of special importance:

Header	Description
X-Nexus-Webhook-ID	This is the event type. For example, *iq:policyManagement*.
X-Nexus-Webhook-Delivery	A unique UUID identifying the event.
X-Nexus-Webhook-Signature	The HMAC digest of the payload body, if an optional secret key has been configured.
X-Nexus-Webhook-Signature-Algorithm	The algorithm that calculates the HMAC digest of the body, currently only HmacSHA1.

Example Header

```
Content-Type: application/json; charset=UTF-8
User-Agent: Sonatype_CLM_Server/1.24.0-SNAPSHOT (Java 1.7.0_25; Mac OS X  ↩
    10.11.5)
X-Nexus-Webhook-Signature: 687f3719b87232cf1c11b3ef7ea10c49218b6df1
X-Nexus-Webhook-Id: iq:policyManagement
X-Nexus-Webhook-Delivery: 7f4a6dde-5c68-4999-bcc0-a62f3fb8ae48
```

A payload is returned with each event type. An example application evaluation payload is shown below:

Example Payload

```
{
    'applicationEvaluation': {
        'policyEvaluationId': 'debceb1d-9209-485d-8d07-bd5390de7ef5',
        'stage': 'build',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'evaluationDate': '2015-05-05T23:40:12Z',
        'affectedComponentCount': 10,
        'criticalComponentCount': 2,
        'severeComponentCount': 5,
        'moderateComponentCount': 3,
        'outcome': 'fail'
    }
}
```

Event Fields The data structure of the event payload differs by event. Event types share the following common fields:

Field	Description
Timestamp	An ISO 8601 representation of the time.
Initiator	userId or "anonymous", "system" for system events.

26.4.1 Policy Management Event

Policy Management events include updates to owners, policies, tags, labels, license threat groups, and owner membership mappings.

Policy Management events have the following fields:

- *action*: i.e. CREATED, UPDATED, DELETED.
- *type*: the type of entity which was updated i.e. APPLICATION, ORGANIZATION, APPLICATION_CATEGORY, LABEL, LICENSE_THREAT_GROUP, ACCESS, POLICY.
- *id*: system ID used to identify the entity which was updated.

Example payload

```
{
    'owner': {
        'id': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
```

```
        'publicId': 'webhooks_application',
        'name': 'Webhooks Application',
        'parentOwnerId': 'abaed4e0-d31e-4a67-9f71-1a8861641077',
        'type': 'APPLICATION',
        'tags': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Tag Foo',
            'description': 'A tag description.',
            'color': 'dark_red'
        }],
        'labels': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Label Foo',
            'description': 'A label description.',
            'color': 'dark_red'
        }],
        'licenseThreatGroups': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'LTG Foo',
            'threatLevel': 5
        }],
        'policies': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Policy Foo',
            'threatLevel': 5
        }],
        'access': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Developers',
            'members': [{
                'type': 'USER',
                'name': 'jyoung'
            }]
        }]
    }
}
```

26.4.2 Application Evaluation Event

Application Evaluation events are those occurring during the lifecycle of a policy evaluation. *Evaluation completed* is the only evaluation event currently available.

Application Evaluation events have the following fields:

- *id*: ID of the policy evaluation.

Example payload

```
{
    'applicationEvaluation': {
        'policyEvaluationId': 'debceb1d-9209-485d-8d07-bd5390de7ef5',
        'stage': 'build',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'evaluationDate': '2015-05-05T23:40:12Z',
        'affectedComponentCount': 10,
        'criticalComponentCount': 2,
        'severeComponentCount': 5,
        'moderateComponentCount': 3,
        'outcome': 'fail'
    }
}
```

26.4.3 Security Vulnerability Override Management Event

Security Vulnerability Override Management events are issued when a security vulnerability override is created, updated, or deleted.

Security Vulnerability Override Management events have the following fields:

- *id*: ID of the security vulnerability override.
- *action*: CREATED, UPDATED, DELETED.

Example payload

```
{
    'securityVulnerabilityOverride': {
        'id': 'd08a4954c2f942e6bbd95517030ebcf7',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'hash': '46c81da3225f991faa2b',
        'source': 'cve',
        'referenceId': 'CVE-2016-0788',
        'status': 'ACKNOWLEDGED',
        'comment': 'Ack'
    }
```

```
}
```

26.4.4 License Override Management Event

License Override Management events are issued when a license override is created, updated, or deleted.

License Override Management events have the following fields:

- *id*: ID of the license override.
- *action*: CREATED, UPDATED, DELETED.

Example payload

```
{
    'licenseOverride': {
        'id': 'cafdf38d458d461583ec6cd509dc8c31',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'status': 'OVERRIDEN',
        'comment': '',
        'licenseIds': [
            'Apache-2.0'
        ],
        'componentIdentifier': {
            'format': 'maven',
            'coordinates': {
                'artifactId': 'foo',
                'classifier': '',
                'extension': 'jar',
                'groupId': 'net.java.bar',
                'version': '1.9'
            }
        }
    }
}
```

Appendix A

Copyright

Copyright © 2011-Present Sonatype, Inc. All rights reserved.

Online version published by Sonatype, Inc,

Sonatype Nexus Repository Manager OSS™, Nexus Repository Manager Pro™, Nexus Lifecycle™, Nexus Auditor™, Nexus Firewall™, IQ Server™, and all Nexus-related logos as well as Sonatype CLM are trademarks or registered trademarks of Sonatype, Inc., in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle, Inc., in the United States and other countries.

IBM® and WebSphere® are trademarks or registered trademarks of International Business Machines, Inc., in the United States and other countries.

Eclipse™ is a trademark of the Eclipse Foundation, Inc., in the United States and other countries.

Apache and the Apache feather logo are trademarks of The Apache Software Foundation.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as

trademarks. Where those designations appear in this book, and Sonatype, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.