

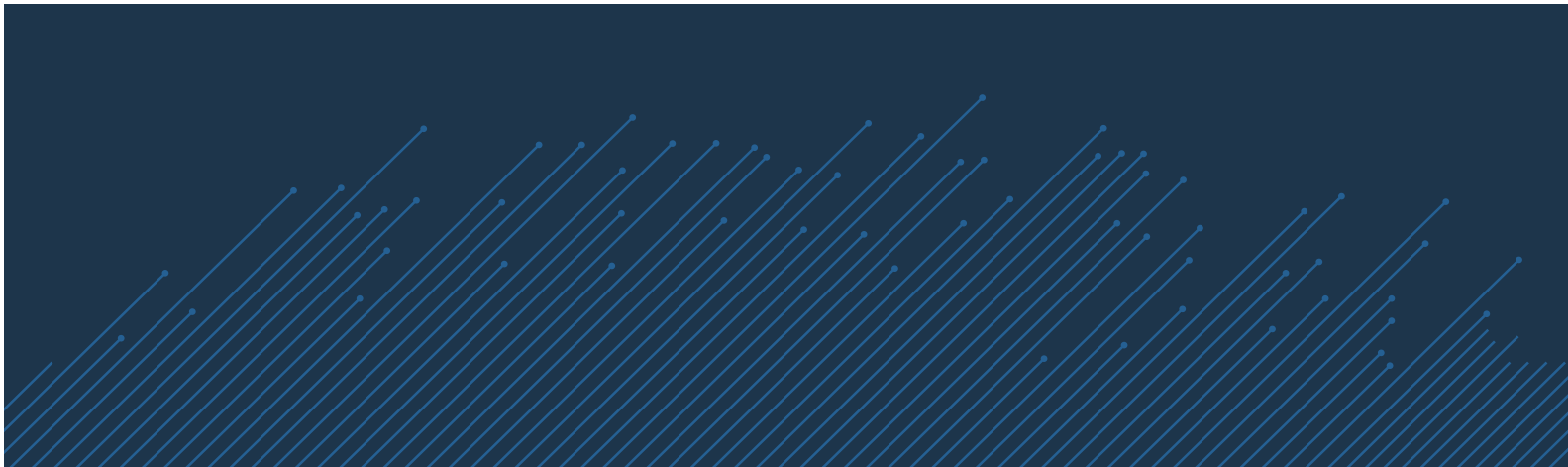


DIGITAL GUARDIAN®



Digital Guardian Rule Implementation Guide

Version 7.5.1



Copyrights and Trademarks

Corporate Headquarters

Digital Guardian
860 Winter Street, Suite 3
Waltham, MA 02451-1414
Tel: 781 788-8180
Fax: 781 788-8188

www.digitalguardian.com

Trademarks

Digital Guardian and the Digital Guardian logo are registered trademarks of Digital Guardian, Inc.

© 2019 Digital Guardian, Inc. Information contained in this document is confidential and proprietary to Digital Guardian, Inc. and its affiliates. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any means electronic or mechanical, including photocopying and recording for any purpose beyond that provided for in the license agreement. The contents of this document are subject to change without notice.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such.

Copyright © 2009 The Go Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- * Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Patents

Digital Guardian products are protected by one or more patents. Contact Digital Guardian, Inc. for more information.

Table of Contents

CHAPTER 1	<i>Getting Started With DG Agent Rules</i>	1
	Who Should Read This Book?	2
	After Reading This Book	2
	Companion Books	3
	Definitions	3
	<i>Policy</i>	3
	<i>Rule</i>	4
CHAPTER 2	<i>Using Rules</i>	9
	DG Rule Evaluation Process	11
	Rule System Logic	13
	Classification Rules	14
	Control Rules	14
	<i>Control Rules With Write Restrictions</i>	15
	<i>Control Rules and Alerts</i>	16
	Filter Rules	17
	Trusted Process Rules	17
	Rule Severity Levels	18

<i>Rule Severity Levels and Alerts</i>	18
Rule Development Methodology	19
Creating Rules	20
Using Windows Environment Variables in Rules	23
<i>Examples of Environment Variables in Rules</i>	23
<i>Displaying a List of Windows Environment Variables</i>	26
Writing Rules To Block Microsoft Windows Explorer Activity	27
Guidelines for Creating Network Transfer Upload Rules	28
<i>Considerations When Creating Prompt Actions</i>	28
<i>Using Prompts in NTU or NTD Blocking Rules</i>	29
Limitations on Detecting NTU and NTD Events	29
Working With Network Operation Events	29
<i>Identifying a Specific Type of Network Operation Event</i>	30
<i>Sample Rule Created With the constOpNetworkEx Event Type</i>	31
About the Static Rule Analyzer	31
<i>How the Rule Analyzer Works</i>	32
<i>Running the Rule Analyzer</i>	33
<i>Understanding the Output From the Analyzer</i>	33
Editing Rules	36
Renaming Rules	37
Copying Rules	38
Activating Rules	39
Deactivating Rules	39
Exporting Rules	40
Importing Rules	41
Creating Parameterized Rules	42
<i>Requirements for Writing Parameterized Rules</i>	43
<i>To Create Parameterized Rules</i>	43
<i>Parameterizing Rule Functions</i>	44
<i>Component List Attributes</i>	48
<i>Using “Choice” With Property Value Types</i>	54
<i>Base Rule Attributes</i>	57

CHAPTER 3 *Rule Definitions* 61

Rule Evaluation Order	62
-----------------------	----

The Return Element 62

Rule Syntax 64

Syntax Example 65

Definition Example 66

CHAPTER 4

Rule Properties 67

Rule Properties and Agent Operating Systems 68

Process Types 69

Current Process 69

Parent Process 69

Source Process 70

Using Time Properties in Rules 70

Global Rule Properties 72

Using evtIsAfterOperation 108

Address/URL Rule Properties 110

Address Properties 110

URL List Properties 111

Application Data Exchange Operation Rule Properties 113

Classification Rule Properties 119

Using the <NOT> Operator With Policy Tag Classification Rules 125

Referring to Content Patterns in Classification Rules 125

Data Vault Rule Properties 126

Device Rule Properties 129

Device Event Properties 136

Control Web Camera Devices With Rules 140

Document Properties Rule Properties 142

Microsoft Windows Explorer Document Properties 145

Default Microsoft Office Document Properties 146

Custom Document Properties 148

Enabling Document Property Rules 148

Using Document Properties in Rules 149

Prerequisites 149

Creating Rules That Target Document Properties 150

Viewing Document Properties on Network Shares 151

Email Operation Rule Properties 152

Message Recipients and Rule Evaluation 152

DLL Load (Image Load) Rule Properties	156
Network File Transfer Operation Rule Properties	160
<i>Using Source and Destination File Paths in Network Transfer Operations</i>	163
Network Operation Rule Properties	163
OLE Insertion Operation Rule Properties	174
Print Operation Rule Properties	174
Process-Related Rule Properties	175
Windows Environment Variables	176
<i>Using the environmentVariable Rule Property</i>	176
<i>Using an Environment Variable in Another Rule Property</i>	176
Windows Registry Rule Properties	178
<i>Registry Event Properties</i>	179
<i>Registry Event Operation Types</i>	180
<i>Sample Registry Rules</i>	181
Property Value Types	183
<i>bluetoothAddress</i>	183
<i>bool</i>	183
<i>dateTime</i>	183
<i>ipAddress</i>	183
<i>constant</i>	184
<i>int</i>	184
<i>int32</i>	184
<i>int64</i>	184
<i>irdaAddress</i>	184
<i>macAddress</i>	185
<i>md5Hash</i>	185
<i>sha1</i>	185
<i>sha256</i>	185
<i>string</i>	185
<i>time</i>	186
<i>Valid Relational Operators for Each Value Type</i>	186
Email Address Format	187
<i>SMTP Format</i>	187
<i>X.400 Format</i>	187

CHAPTER 5 *Symbolic Constants* 189

Symbolic Constants and Agent Operating Systems	189
--	-----

Application Data Exchange Types	190
Bus Types	190
Connection Types	192
Days of the Week	193
Drive Types	194
Email Recipient Types	195
<i>Encryption Types</i>	196
Network Adapter Types	198
Operation Types	199
Product Types	210
Protocol Types	211
Registry Event Types	212
Registry Event Values	213
Rule Actions	214
Wireless Authentication Types	215
Wireless Encryption Types	216
Wireless Infrastructure Mode Types	216

CHAPTER 6 *Logical Operators* 217

AND	218
OR	218
NOT	218
Nested Logical Operators	219

CHAPTER 7 *Relational Operators* 221

Using Process Names With Relational Operators	222
Equal	223
Greater Than	223
In	223
IP Mask	224
Less Than	224

Like	225
<i>Find Value Anywhere in a String</i>	225
<i>Find Value at Beginning of a String</i>	226
<i>Find Value at End of a String</i>	226
<i>Find Unicode Values</i>	226
Op Operator	228
Match Attributes	229
<i>Match All</i>	230
<i>Match Any</i>	230
<i>Supported Properties</i>	230
Regular Expressions	231

CHAPTER 8 *Using MD5 in Rules* 233

Generating GUIDs of MD5 Hash	234
Adding MD5 Hash to a Rule Definition	234
MD5 Rule Example	235

CHAPTER 9 *Rules for SharePoint Operations* 237

Digital Guardian Features With SharePoint	238
<i>Classification</i>	238
Supported SharePoint Clients	238
SharePoint Terminology	238
Operations the Agent Can Report, Monitor and Control	239
Writing Control Rules for SharePoint Operations	243
<i>Write Rules for Managing File Download Events</i>	244
<i>Write Rules for Managing File Upload Events to SharePoint</i>	246
<i>About Writing Rules for Microsoft Excel File Events Involving SharePoint</i>	249
Writing Classification Rules for SharePoint Operations	251
Viewing SharePoint-Related Events in the DGMC	253

CHAPTER 10 *Rule Variables* 255

Introducing Rule Variables	257
Rule Variable Use Case Examples	257

<i>Example 1 — Rule Counting Operations or Events</i>	258
<i>Example 2 — Rules Tracking Time Spans</i>	259
<i>Example 3— Control Rule Application</i>	260
<i>Example 4 — Classification Rule Application</i>	261
Creating Rule Variables	261
<i>Rule Variable Types</i>	263
Pairing Rule Variables With Types	265
<i>Rule Variable Names</i>	266
<i>Rule Variable Scopes</i>	267
<i>Lifetimes of Rule Variables</i>	270
Rule Variable Arrays and Lists	272
<i>Creating Rule Variable Arrays — AsArray Statement</i>	272
<i>Control How DG Adds Values to Arrays</i>	274
Adding Rule Variables to Rules	276
Referring to Variables in Other Rules	277
Rule Evaluation Logic With Variables	277
Operators for Rule Variables	279
<i>Setting Variables With Set</i>	279
<i>Adding Variables With Add</i>	280
<i>Subtracting Variables With Sub</i>	280
<i>Using Sub to Declare a Rule Variable</i>	280
<i>Using Sub to Remove Information From an Array</i>	281
<i>Rules with Variable Operators and No Other XML</i>	282
Naming Rule Variables	284
Variable Syntax	284
Using Data in Rule Variables	285
<i>Property Values in Rule Variables</i>	285
<i>Variable Type Conversion</i>	286
Adding Variables	287
<i>Adding Arrays of Strings</i>	288
<i>Adding Arrays of Integers or Strings</i>	289
<i>Adding Mixed Types</i>	292
Subtracting Variables	293
<i>Subtracting Arrays of Strings</i>	293
Testing Variable Values	294
<i>Array Processing Commands</i>	296
Returning the Index Value of an Array Element	300
<i>Using GetIndexOfArrayItem</i>	300

Deleting an Array Item With Dispose	303
Using the Eval Operator	303
<i>Adding the Eval Operator to Rules</i>	304
Posting Rule Variables to the DGMC	305
<i>Mapping a Rule Variable</i>	306
<i>Writing Rules With Rule Variables</i>	306

CHAPTER 11

Component Rules and Rule Functions 309

Create Component Rules	310
Call a Component Rule	311
<i>Example</i>	311
Export and Import Component Rules	312
<i>Exporting Component Rules</i>	313
<i>Importing Component Rules</i>	313
About Rule Functions	314
<i>How Functions Fit in the Workflow</i>	316
What Do Functions Look Like?	317
<i>String Management Functions</i>	319
<i>Array Management Functions</i>	320
<i>Timer Functions</i>	322
<i>Custom Events Functions</i>	325
<i>Function To Kill Processes</i>	328
Classification Functions	329
<i>Function for Resetting the Classification Stream</i>	329
<i>Function for Erasing the Classification Stream</i>	329
<i>Function for Removing Tags From the Classification Stream</i>	332
<i>Refresh the Classification Information for An Event</i>	336
Process Flag Functions	337
Registry Key Functions	341
<i>Support for Monitoring Registry Keys</i>	343
<i>Registry Key Monitoring Operation</i>	345
<i>Registry Value Monitoring Operation</i>	345
<i>evtRegistryOperationType</i>	346
Incident Remediation Function	349
<i>Function for Quarantining Files</i>	349
<i>Potential Use Cases</i>	349
<i>Example—Quarantine Classified Files by Extension</i>	353

Agent Configuration Information Functions	355
<i>Test Agent Feature Status in Rules</i>	355
<i>Restore a Rule Variable to Unused State</i>	357
<i>Check for an Active Process</i>	357
Event Caching Functions	359
<i>DG_SendCachedData</i>	359
<i>DG_ClearCachedData</i>	360
<i>DG_AggregationControl</i>	360
Functions To Capture Files	362
<i>DG_CaptureFile Function</i>	362
<i>DG_CaptureFileByPath Function</i>	366
<i>Encrypting and Decrypting Files With Functions</i>	368

CHAPTER 12

Component Lists 371

<i>Types of Component Lists</i>	372
<i>Sources of Component Lists</i>	373
<i>Component List Size</i>	374
<i>DG License Requirements for Component Lists</i>	375
<i>Format of Component Lists</i>	375
<i>Component List Storage on the DG Agent</i>	377
Creating and Updating Component Lists	377
<i>Create Component Lists</i>	377
<i>View Component List Details</i>	381
<i>Edit Component Lists</i>	383
<i>Restore a Deleted Entry</i>	384
Managing Component Lists	386
<i>Managing Very Large Component Lists</i>	386
<i>Import Component Lists</i>	386
<i>Export Component Lists</i>	387
<i>Delete Component Lists</i>	388
<i>Component List Rule Event Reporting</i>	388
<i>View Component List Deployment</i>	388
About Rules for Component Lists	389
<i>Calling a Component List</i>	390
<i>Example Control Rule</i>	391
List Compile Job	392
About the Component List API	394

Using the Component List API	395
<i>Creating and Deploying Component Lists With the API</i>	397
<i>Managing Component Lists With the API</i>	397
<i>Stored Procedure Reference Pages</i>	399

CHAPTER 13

Data Vault Rules 419

Trigger Rules	419
Data Vault Rules	420
Using File Save As	422
Multiple Data Vaults	422
Creating Data Vault Rules	422
<i>Creating a Trigger Rule</i>	423
<i>Creating a Data Vault Rule</i>	424
Examples of Data Vault Rules	426
<i>Trigger Rule To Flag Notepad Process as Data Vaulted</i>	426
<i>Trigger Rule To Create Data Vault in Response To On-Screen Content</i>	427
<i>Prevent User From Saving Files Outside of c:\DocRepository Using a Data Vaulted Notepad Process</i>	428
<i>Prevent User From Performing Network Uploads When a Specific Data Vault Is Active</i>	429
<i>Block Application Data Exchange From Application Screen Content</i>	430
<i>Prevent Uploads and CD Burning When Any Data Vault Is Active</i>	431
<i>Prevent Screen Prints When a Specific Rule Is Active</i>	432
<i>Block File Writes Outside Specific Directory</i>	433

APPENDIX A

Common Use Cases 435

Examples of Classification Rules	436
<i>Classify Files Containing More Than One Credit Card Number</i>	436
<i>Classify Word, Excel, and Text Files Containing an Employee Name</i>	437
<i>Classify Files Downloaded From a Particular Domain</i>	438
<i>Classify Excel and Text Files Containing a User Name</i>	439
<i>Classify Excel, Word and Text Files Containing Widgets</i>	440
<i>Classify Files That Contain More Than 9 Social Security Numbers and 19 Credit Card Numbers</i>	441
<i>Classify Word, Excel and Text Files</i>	443
<i>Erasing the Classification Stream on a File</i>	444

Examples of File System Rules 446

- Reading Files on a Specified File Server Generates an Alert 446*
- Block File Delete Events for Non-owner of File 447*
- Test a User Security ID 447*
- Only Approved Applications Can Open Specified Extensions 449*
- Block File Copy or Move on Linux Computer 450*
- Block File or Folder Writes to Removable Drives 451*
- Block File or Folder Activity on Removable Media 452*
- Block Starting Instant Messenger Based on Original File Name 453*
- Block File Overwrites on Removable Drives 454*
- Restrict File Writes, Copies and Moves to List of Approved Removable Drives 455*
- Block File Activity Based on Document Properties 456*
- Detect File Writes to Unknown or Removable Drives 457*
- Block Attempts To Delete or Rename Files Contained in Specified Directory 458*
- Allow Users To Write or Copy File Only to Specified Folders on the File Server 459*
- Prevent All File Operations Within A Specified Folder if the Laptop Is Outside the Corporate Network for Over 12 Hours 460*
- User Cannot Compress or Zip Source Files 461*
- User Cannot Send Specified File Extensions Using IM Applications 462*
- Do Not Allow Kazaa To Share Music Files 463*
- Prevent Users From Changing the Extensions of Source Code Files 464*
- Prevent Visitor Account Users From Copying Files to Removable Drives 465*
- Block Registry Editor From Starting if User Name Is “tester” and Agent Driver Is Running 466*
- Prevent Users Who Are not Administrators From Renaming or Using Save As To Save Executable Files to New Filenames 467*
- Prevent RME From Attaching to Specified USB Drives 468*
- Encrypt Files Based on Rule Evaluation 469*

Examples of Network Rules 470

- Certain Subnets Cannot Connect to Sensitive Computers 470*
- Allow Remote Connections Only to Authorized Servers 471*
- Block Connections to Computers Outside the Subnet 472*
- Block Downloads of Mp3 Files 473*
- Block Access to Specified Ports on Remote Computers 474*
- Block Uploads if too Many Adapters Connect to Known Network 475*
- Detect FTP Connections to Unauthorized Computers 476*
- Allow Only Web Browsers and Web Services To Make Network Connections Using Ports 80 and 443 477*
- Block Network Uploads From the Specified Directory 478*
- Only Allow NETBIOS Calls on Port 139 in the Corporate Subnet 478*
- Limit Browser Access to Company Internal Web Sites 479*

<i>Block Network Operations Over WiFi Connections</i>	479
Examples of Print Rules	480
<i>Block Printing Unless the DG Server Is Available</i>	480
<i>Generate Alert When Printing Specified Files</i>	480
<i>Allow Printing Only on Authorized Printers</i>	482
<i>Block Printing When Laptop Computers Are Not on the Corporate Network</i>	483
<i>Block Printing From Specified Computers</i>	484
<i>Block Printing of Certain File Extensions After 6:00 PM and Before 8:00 AM</i>	485
Examples of CD/DVD Rules	486
<i>Allow CD/DVD Writing Using Only Authorized Applications</i>	486
<i>Allow CD/DVD Writing Only During Office Hours</i>	487
<i>Prevent Laptops From Writing to CD/DVD Drives</i>	487
<i>Block File Reads From CD Drives</i>	488
<i>Block File or Folder Activity on Removable Media</i>	489
Examples of Device Rules	490
<i>Control Web Camera Video Stream Use</i>	490
Examples of Application Data Exchange Rules	491
<i>Generate an Alert When Specified Applications Perform Application Data Exchange Actions</i>	491
<i>Block Paste Actions Into IM Applications</i>	492
<i>Block Application Data Exchange Using Unauthorized Applications</i>	493
<i>Prevent Users From Exchanging Content From Microsoft Word Into AOL Instant Messenger</i>	494
<i>Prevent Users From Cutting or Dragging and Dropping From any Process Containing the String “acro”</i>	495
<i>Block Screen Capture of File Content in Selected Processes</i>	496
Examples of Email Rules	498
<i>Block Emails Sent to BCC Recipients</i>	498
<i>Block Excel Files Sent to External Recipients</i>	499
<i>Block All Emailed .doc File Attachments</i>	500
<i>Block Email Attachments Greater Than 1KB in Size</i>	500
<i>Block Emails Larger Than 1Kb in Size</i>	501
<i>Test Email Address Against List of Addresses</i>	501
Examples of Filter Rules	503
<i>Filter File Read Events From Specified Applications</i>	503
<i>Filter Microsoft Outlook Express Noise</i>	504
<i>Filter MSN Toolbar Update Noise</i>	504
<i>Filter Noisy File Extensions</i>	505
<i>Filter Noisy Microsoft Office Application Files</i>	506
<i>Filter Microsoft Outlook Noise</i>	507

Filter X1 File Activity Noise **507**

Examples of Trusted Process Rules **508**

Identify Notepad as a Trusted Process **508**

Identify a List of Trusted Applications **509**

Examples of File Capture Rules **510**

Capture Source Files in File Recycle or Delete Events **510**

Capture and Hash the Source File in a File Copy Operation **512**

Capture the Destination Files When a User Moves a File **513**

Capture Source Files in Network Upload Events **514**

Examples of Virtualization and RDP Rules **516**

Copying Tagged Files Over Remote Desktop Connection **516**

Capture Remote Desktop Actions With Tagged Files **516**

Capture Microsoft TeamViewer Software Accessing Tagged Files **517**

Capture Tagged File Access From Virtual Machines **517**

APPENDIX B

Rule Optimization **519**

Minimize Use of Regular Expressions **519**

Place More Restrictive Rule Conditions at the Bottom of the Rule Definition **520**

Use Trusted Process Rules **521**

APPENDIX C

Common Rule Variable Uses **523**

Counting With Rule Variables **524**

Prevent User From Copying ZIP Archives to Removable Devices **524**

Tracking Time **525**

Limit File Copying to 1GB per Day **525**

Running Processes **529**

Transfer Events Immediately From Agents **529**

Capture Executable File (.exe) Exit Codes **530**

Index 533

CHAPTER 1 Getting Started With DG Agent Rules

This is an implementation guide for developing Agent rules on the Digital Guardian (DG) platform. It provides the syntax, usage and context for developing control, filter, trusted process, data vault and component rules on the DG platform. DG Agent rules enable your enterprise to monitor and act on actions at the point of use. While some organizations use Digital Guardian only for monitoring and logging events, the DG Agent rules engine lets you customize the activities and events that DG records and regulates.

Digital Guardian Agent rules can provide many benefits, including the following:

- Prevent the leakage of data onto uncontrolled media including USB, CD/DVD, hardcopy, Instant Messenger and Web mail.
- Harden operating system environments against attacks.
- Prevent the inadvertent disclosure of sensitive or confidential documents or drawings through their insertion into a publicly available document.
- Alert security and executive management on the movement of sensitive or classified data across the enterprise.

To enable organizations to understand and implement DG to its greatest potential in a minimum amount of time, this document introduces the following DG concepts:

- Definition of policy, rules, and types of rules.
- Typical rule implementation methodology.
- Comprehensive walkthrough for implementing rules from how to log into the Digital Guardian Management Console to deploying policies containing customized rules.
- Examples of typical DG rule use cases as well as their corresponding implementation in the DG Extensible Markup Language (XML) rules syntax.

Who Should Read This Book?

The audience for this document includes the following people:

- **DG Rule Implementers** — Rule implementers use this document to write rules.
- **Security Architects and Solution/Design Leads** — Architects and solution/design leads use this document to understand the implementation and functionality of DG Agent rules.
- **Security Functional Analysts** — SF analysts use this document to understand how and which security functional requirements map to DG Agent rules functionality.
- **Solution Providers** — Solution providers learn what types of custom functionality DG can implement. Also how to incorporate the DG rule development methodology into their requirements analysis, solution development, functional specification, and technical implementation methodologies.

After Reading This Book

After reading this book, you will understand the process for creating and implementing Digital Guardian Agent rules.

Companion Books

The following table lists all the manuals in the Digital Guardian documentation set:

Table 1-1. The Documentation Set

When to Read	Title
Before You Install	<i>Digital Guardian Installation and Upgrade Guide</i>
	<i>Digital Guardian Release Notes</i>
	<i>Digital Guardian What's New</i>
While You Install	<i>Digital Guardian Installation and Upgrade Guide</i>
	<i>Digital Guardian Release Notes</i>
After You Install	<i>Digital Guardian Management Console User's Guide</i>
	<i>Digital Guardian Online Help</i>
	<i>Digital Guardian Release Notes</i>

Definitions

The following entries define some commonly used concepts in implementing Agent rules for Digital Guardian:

Policy

A policy is a container for a set of rules. DG policies deliver the set of rules to DG Agents so that user and application activity can be monitored, controlled, recorded, or filtered appropriately. DG includes several different types of policies, as described in “Working With Policies” in the *Digital Guardian Management Console User's Guide*.

Policies can have a status of Active, Inactive, or Trial. The DG Server deploys active policies and does not deploy inactive policies. A policy can contain a com-

bination of active and inactive rules. If the policy status is Active, all rules in that policy that have Active status are applied.

Trial policies are used as part of a test and development process to test the effectiveness of the rules that will be contained in a policy before enforcing the rules at the desktop. When Trial status is enabled, all user activity that triggers a rule that has Active status will send an alert to the DGMC regardless of the response that you have specified for the rule. The alert is not visible to the user.

Rule

A rule is the collection of tags and content that specifies point-of-use activity. It is a member of a policy. Rules have two statuses—Active or Inactive. An active rule takes effect if it is a member of an active policy. An inactive rule does not take effect. A policy with both active and inactive rules only deploys the active rules to Agents. The following sections describe the types of rules.

Classification Rules

A classification rule identifies files based on certain criteria. Once you have classified files, you can write control rules that regulate user activity based on the classification of your files. For more information on classification, refer to *Digital Guardian Management Console User's Guide*.

Control Rules

Control rules trap point of use activity and perform an action upon them. Digital Guardian's core functionality resides in these rules. The weights assigned to each rule action determines which rules execute first. By default, rules execute in the following order:

1. Block
2. Encrypt
3. Prompt

4. Alert
5. Data Vault

By default, all block rules execute before prompt rules, which execute before alert rules, and so on. This execution order applies only when two or more rules have different response types for the same event.

For any activity at the point of use, the DG Agent executes all rules either until it identifies a match for a rule or has executed all rules. Control rules execute regardless of any filter rules.

For information about changing the default order of rule execution, refer to the appendix on rule weights in *Digital Guardian Management Console User's Guide*.

Filter Rules

Filter rules exclude point of use activity from being logged. These rules are useful in reducing event generation by background processes. Examples of activity that an enterprise may want to filter using filter rules include the following:

- File open and read events from the following:
 - c:\windows\inf
 - Files with an extension of .joboptions
- NetBIOS over TCP activity between Outlook and an Exchange server.
- Activity for policy, regulatory, or statutory reasons DG should not monitor.

While these filter rules provide fine-grained reduction in the types of logged activity, the most efficient method to reduce activity logs is by configuring the DG Agent install process to filter entire classes of user activity. These classes include the following:

- Application Data Exchange (for example, clipboard and drag-and-drop)
- CD Burning
- Copying Files
- Deleting Files

- Editing Files
- Moving Files
- Network Operation (disabled by default)
- Network Transfer Upload
- Printing Files
- Reading Files (disabled by default)
- Recycling Files
- Renaming Files
- Restoring Files
- Sending Emails
- User Logon
- User Logoff
- User Logon Failed
- Writing Files (disabled by default)

If activity triggers a control rule, the DG Agent does not filter it.

Trusted Process Rules

Trusted process rules exclude specified executables from being logged or trapped. Examples of typical trusted processes include the following:

- Virus scanners
- Host-based firewalls and intrusion detection systems
- Compilers and linkers

Trusted process rules execute before filter and control rules. As such, the DG Agent does not execute any rules on activity generated by these trusted processes.

Component Rules

Component rules are rules that you can reference from the definition of other rules, similar to a function in a programming language. By using component rules, you can reduce the amount of time required to write and maintain your rules.

Component rules do not contain action, severity, or status information. This information is supplied by the rule that uses the component rule. For more information on component rules, refer to [“Component Rules and Rule Functions” on page 309](#).

Data Vault Rules

A data vault is a set of additional rules that you can apply to an application process. A data vault is created when user activity meets a set of conditions outlined in a trigger rule. When the trigger rule is activated, the data vault is created and additional restrictions are applied to user activity. The data vault remains in effect until the user closes the application that triggered the data vault.

CHAPTER 2 Using Rules

Digital Guardian rules provide a way to define which user events are important, require monitoring, or can be filtered. There are several types of events that can trigger rules or be filtered:

- Application Data Exchange Events
- CD/DVD Burn Events
- Device Events
- Email Events, such as sending an email attachment to a recipient outside your organization, or to an unapproved domain
- File Events, such as file copy and file delete events
- Network File Transfer Events, such as file upload events
- Network Operations Events, such as connections to Web site URLs
- Print Events
- Process Events
- User Logon and Logoff Events

Rules are not specific to a given event type. For example, you can create a single rule that governs both application data exchange and network events.

DG Rules support the following types of rules:

- **Classification Rule** — Applies classification tags to a file in response to rule criteria and document content. Classification rules dealing with document content are available with Digital Guardian Adaptive Content Inspection.
- **Component Rule** — A rule definition that you can reference from within another rule of the same type, similar to a function in a programming language. You can use component rules to centralize your commonly used rule definitions.
- **Control Rule** — Governs user activity. For example, a control rule could block users from burning CDs containing Outlook address book files.
- **Data Vault Rule** — Rules that take effect only when the conditions outlined in a trigger rule have been met. You can use data vault rules to create additional security that takes effect only in specific cases. For more information, refer to [“Data Vault Rules” on page 419](#).
- **Filter Rule** — Prevents DG Agents from reporting application events described in the rule. Filtered activities are still tracked by the Agent and regulated by control rules. Filtered activities are not included in reports. For example, a filter rule could prevent Digital Guardian from reporting the event generated by the deletion of temp files created by Microsoft Word.
- **Trusted Process Rule** — Prevents DG Agents from recording all event activity related to a specific process. For example, a trusted process rule naming an anti-virus application would prevent Digital Guardian from recording or reporting events involving that application. Trusted process rules do not prevent Digital Guardian from recording that the named process has started and stopped.

After creating rules, you add them to policies. Rules that are not included in a policy have no effect.

After you associate a rule with a policy, you can apply that policy to users, groups, or computers. Control Rules can apply to users so they are monitored no matter what computer they log in to. Alternatively, you can apply all rule types to a specific computer. In this configuration, the rules apply regardless of which user is logged on.

Note: You cannot delete rules after you have created them. In most cases, however, you can rename them. For more about renaming rules, refer to [“Renaming Rules” on page 37](#).

DG Rule Evaluation Process

To enhance your current rules with rule variables, you need to understand how Digital Guardian evaluates the rules you apply to Agents.

DG provides a rule engine that processes rules to determine whether events or alerts on an Agent machine match operations or configurations specified in the rules in place on the Agent. With basic DG rules, the rule engine either:

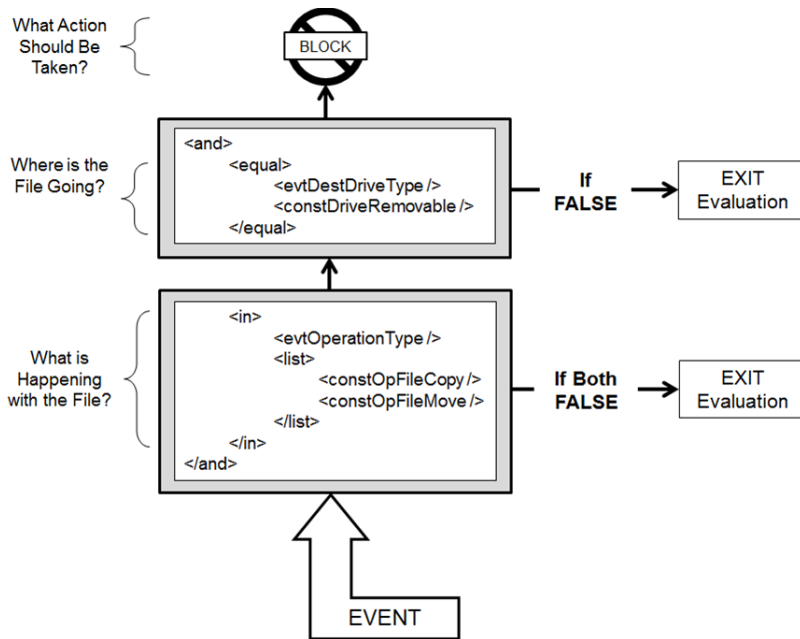
- Exits from the rule when one clause in the rule evaluates to false.
- Applies the rule action when all rule clauses evaluate to true.

Rule variables add another option to the rule behavior—the engine can set a rule variable to contain information about the rule evaluation. With a rule variable in the rule, you can save rule information and use it in other rules, or later when the same rule is triggered again.

When the user performs an operation that generates an event, the Agent captures the event and the rule engine goes to work. It evaluates the event and compares it to all of the rules in place on the Agent, beginning with the highest-priority rule. Starting at the bottom of each rule, the engine compares the details of the event, such as source file location or file extension, to the constants and properties in each rule clause, looking for matches. Evaluation for each rule continues as long as each clause in a rule is true. When the engine encounters a clause that is not true, the engine stops evaluating the rule and moves on to the next highest-priority rule.

In the figure, you see the rule engine starting at the bottom of the rule with the event, evaluating the event operation (what is happening to the file), and then evaluating the destination drive (where is the file going) for the event. If all of the evaluations are true, the rule engine directs the Agent to block the event. If a clause in the rule, either the type of operation or the destination drive type, does

not match the event, the rule engine stops evaluating the rule and directs the Agent to allow the event.



In more detail, in this sample rule, the value of `evtOperationType` (the operation being performed by the user) is evaluated to determine whether it is equal to either `constOpFileCopy` (copying a file) or `constOpFileMove` (moving a file). If the operation is a copy or a move, the rule engine evaluates this clause to true and moves up to the next clause and its operator. If the user performed any other file operation, the clause evaluates to false and the rule engine exits the rule.

Note: If the first operation returns a null value (there is no value to feed to the variable), the rule engine evaluates the operation as true, and moves on to the next criterion.

In the second operation, the value of `evtDestDriveType`, which contains the destination of the file operation the user is performing, is evaluated to determine whether it is equal to `constDriveRemovable`—is the destination of the file operation a removable drive? If the destination is a removable drive, the rule engine evaluates this clause to true and moves on to perform the action defined by the

rule, such as blocking the operation. If the destination for the file is any other type of drive, the clause evaluates to false and the rule engine exits the rule.

Rule System Logic

Before diving into rule variable integration, examine the process flow below to gain visibility into how the rule engine evaluates standard rule logic.

In this use case, the rule directs the Agent to prevent all attempts to copy or move files to removable media.

The rule engine evaluates rule logic from the bottom up. The user activity is first evaluated to determine whether it meets the first set of criteria, which is File Copy or File Move. If the user is either copying or moving a file, the first evaluation is true, so the rule engine moves up to the next criteria. If the user is not performing a file copy or move operation, the rule engine evaluates the criteria as false and exits.

To optimize performance of the rule engine, place the arguments that are most likely to produce a false value toward the bottom of the rule logic. This structure ensures that the rule engine moves on to the next rule as soon as possible and does not waste time evaluating more rule criteria unnecessarily.

In this example, after the user operation criterion is met, the rule engine moves on to evaluate the next criteria which is the destination drive type. If the file copy or move is going to a removable device, (such as a USB drive), the block action assigned in the rule header is implemented. If the user is copying or moving the file to any other drive type, the destination drive type criterion evaluates to false, and the rule engine exits from the rule. All criteria must be met (evaluate to true) for the rule action to be implemented.

Standard rules are stateless, meaning the highest priority rule is evaluated first until its component's value (or state) negates further evaluation, and the rule exits.

Classification Rules

A classification rule identifies files based on specified criteria. After you have classified files, you can write control rules that regulate user activity based on the classification of your files. For more information on classification, refer to *Digital Guardian Management Console User's Guide*.

Digital Guardian classification takes place on two levels:

- **Basic Classification** — Basic classification uses standard Digital Guardian rule properties and events to classify files. Any event or property that you could use to control a file, you can also use to classify a file.
- **Adaptive Content Inspection** — Digital Guardian supports an add-on feature — Digital Guardian Adaptive Content Inspection (ACI). ACI enables the DG Agent to examine the contents of your files for keywords and patterns. If the Agent recognizes file content, it classifies the file and identifies how many times the content pattern appears in the file.

Unlike other Digital Guardian rules, classification rules do not require an event. You can create classification rules that consist solely of properties. For example, you could create a classification rule that classifies all .vb files in a particular directory as "Source Code."

Control Rules

DG Agents can respond to control rules in the following ways:

- **Non-visible Alert** — DG Agent informs the DG Server that an event has occurred but does not interfere with or inform the user.
- **Prompt** - DG Agent informs the user that the event is being monitored, and prompts the user in one of the following ways:
 - **Block Prompt** — Displays a message to users that activity they are attempting is blocked.
 - **Custom Prompt** — Presents a custom prompt that you develop.
 - **Decide Prompt** — Presents a prompt that allows users to decide whether to continue the activity.

- **Justify Prompt** — Displays a prompt that presents a message asking users to enter text that justifies performing the activity. Users can enter the text or cancel the activity. The Agent does not read or parse the text.
- **Password Prompt** — Presents users with a prompt requesting a password. The password is required to decrypt files that are targeted by Removable Media Encryption.
- **User Decision Prompt** — Displays a message to users regarding the activity they are attempting to perform. Users can continue the activity or cancel the activity.
- **Warn Prompt** — Displays a warning to users regarding the activity they are performing. Users are allowed to continue their activity.
- **Block** — DG Agent informs the user that the event is forbidden and prevents the user from proceeding.

For more information about prompts, refer to “Working with Prompts” in *Management Console User’s Guide*.

Control Rules With Write Restrictions

You can write control rules that restrict writes, saves and other activities for both files and folders. Rules that restrict file-related activities will apply to both.

For example, rules that restrict file activities on removable media can block the following activities:

- Copying a folder with files to removable media
- Renaming a file or folder on removable media
- Deleting a file or folder on removable media
- Creating a new folder on removable media
- Saving a file back to removable media
- Changing attributes on a file
- Changing attributes on a folder

Additionally, when a rule specifies to block writes to files on removable media and a user attempts to open such a file, the file opens as read-only.

Caution: For write restriction rules to apply to all file types, you must use the Overwrite Agent Default Configuration Wizard to disable implicit filtering for events occurring on removable drives. This allows the rule engine to evaluate the events and apply policies as necessary. For instructions, refer to “Overriding Agent Configuration” in *Digital Guardian Management Console User’s Guide* or refer to the “DG Agent Wizard” topic in the DGMC Help.

For sample write restriction rule XML, refer to [“Block File or Folder Activity on Removable Media” on page 452](#).

The DGMC shows blocked folder activity as well as blocked file activity in the Forensic reports.

Control Rules and Alerts

When a user violates a control rule, the DG Agent sends an alert to DG Server. If an alert is generated and you have chosen to receive alert notification emails, you receive an email that indicates that a control rule has been violated. You can also view and resolve the alerts in the Alerts report. However, when a control rule is violated, you can prevent the DG Agent from sending an alert to DG Server.

You can prevent alerts from being generated for specific rules by changing the Send Alert option. When you select None from the Send Alert drop-down list, DG Agent does not send an alert to DG Server, and thus, you do not receive an email notification when the rule is violated. Choosing this option helps reduce the number of alerts that you receive, and the number of alerts that you must resolve.

The Send Alert option depends on the rule action you choose in the Action drop-down list. The None option is available only for the Block and Prompt rule actions. You cannot select None for the Non-visible Alert action.

Filter Rules

A filter rule excludes extra information from being captured by DG Agents. If you do not set up filter rules, the DG Agent will capture all events, including events that are not a security risk.

Digital Guardian filtering takes place on two levels:

- **Implicit Filtering** — Digital Guardian includes an Implicit Filter that automatically removes low level activity from your reports based on file extension and file type (signature).
- **Filter Rules** — Like Control rules and Trusted Process rules, you can create Filter rules where the symbolic constants, properties and operators available for rule creation remain the same. After you create a filter rule, you must add it to a Filter policy for it to take effect.

Trusted Process Rules

Trusted process rules identify a process when that process starts. If the process is a trusted process, Digital Guardian recognizes the process and ignores all further activity from that process. Trusted process rules execute before filter and control rules. As a result, the DG Agent does not execute any rules on activity generated by trusted processes. By identifying trusted processes, you can reduce the number of processes that the DG Agent must monitor.

Examples of typical trusted processes include the following:

- Virus scanners
- Host-based firewalls and intrusion detection systems
- Compilers and linkers

Rule Severity Levels

Every Control Rule has an associated severity level. The severity level determines the seriousness of a violation of the rule. The Severity Level you assign a Rule translates into the Severity Level of its Alert, which is sent by the Agent to the DG Server. Alert Managers use Alert Severity Levels when setting up **Alert E-mail Notifications**.

Use discretion when you define Rule Severity Levels, because they determine Alert Severity Levels, as shown in the table below. An Alert Manager's assigned Alert Severity Level determines the threshold of Alerts they will receive via Alert E-mail Notifications.

Best practice is to assign the **Critical** Rule Severity Level only to user activity that poses the highest risk to your organization. This helps to prioritize Alerts as they are sent to the DG Server and enables Alert Managers to explicitly review and resolve Alerts that identify the user activity that poses the highest risk to your organization.

Rule Severity Levels and Alerts

Use this table to see the relationship between Rule Severity Levels and Alerts.

Rule Severity Level	Alert Severity Level	Email Notification Management Threshold
Critical	Critical	Critical
High	High	High and Critical
Medium	Medium	Medium, High and Critical
Low	Low	Low, Medium, High and Critical
Informational	Informational	All Alerts

Rule Development Methodology


Rule developers can deploy DG Agent rules and policies using the following process:

1. Define the problem to solve. Examples of typical problems that DG rules help solve include the following:
 - a. How can we pass our SAS 94 IT audit so that the auditors certify that we comply with the Sarbanes-Oxley Act?
 - b. How can we prove that all information provided under a non-disclosure agreement is deleted upon termination of our right to use that information?
 - c. What assets (data) do we need to monitor or protect?
 - d. How do we hinder or prevent trusted insiders from leaking proprietary intellectual property?
2. Design a solution to the overall problem.
 - a. Define the business requirements of the problem.
 - b. Define the components of the overall solution. This includes off-the-shelf software (Digital Guardian), methodologies, or custom development.
 - c. Describe the overall solution process flow.
3. Define the functional requirements that each component must provide to satisfy its portion of the overall solution. For the Digital Guardian component, perform the following:
 - a. Define the business rules.
 - b. Identify the user operations, data, and resources to trap.
 - c. Specify the Digital Guardian events, such as application data exchange, file write, or application start, and event properties, such as source or destination file, destination network address, or executable name, that match the user operations to capture.
 - d. Specify the logic for the user operations to include or exclude using Boolean operators (AND, OR, NOT).

- e. Specify how to evaluate the relationships between operations, data, and resources—equality, greater than, less than, string regular expression match, collections of values, or IP masks.
- f. Specify data and resources by type—string, Boolean, IP address, MAC address, integer, time or MD5 Hash (for processes).

Creating Rules

Users with the Control Policy Manager, Filter Policy Manager, or Trusted Process Policy Manager role can create and maintain an unlimited number of rules for that rule type. For a description of rule types and the activities they govern, refer to [“Using Rules” on page 9](#).

1. Hover over Policies and select **Policies**. The Manage Policies page opens.
2. In the Rules section, select the rule type to create. A list of existing rules of that type appears.
3. Click  and select Create New Rule. The Control Rules page opens.
4. Enter the following information:

Name — The name of the rule. The name of the rule must be unique across all rule types. For example, you cannot have a Control rule and a Trusted Process rule that share the same name. Do not use the special characters & (ampersand), < > (less than and greater than), \$ (dollar), % (percent), ; (semicolon) or ? (question mark) in the rule name. These characters cause the rule name to display incorrectly in the DGMC or cause other problems managing the rules.

Description — Enter a description for the rule.

Status — Select Active to enable the Rule. Select Inactive to disable the rule.

Category — The category to which the rule will belong. You must assign all rules to a category. You can assign a rule to an existing category or cre-

ate a new category. To create a new category, select **New Category** and enter a new category name.

Action — (Control rules only) Select the action that occurs when a rule is violated. Digital Guardian supports the following types of actions:

- **Block** — The user action is blocked. A Microsoft Windows message may appear informing the user that the action is forbidden.
- **Continue** — The user's activity is allowed, but an alert is generated and an email notification is sent. No visible prompt is displayed.
- **Encrypt** — The DG Agent encrypts the associated file. The user may be prompted to enter an encryption password, depending on the encryption type that is selected. This action becomes available if you use Removable Media Encryption.
- **Prompt** — The rule displays a Digital Guardian prompt. Selecting an Action type of prompt displays another option where you specify the prompt to use.
- **Vault** — The user's activity is allowed, but the application process is data vaulted and an additional set of rules is then enforced. For more information on data vault rules, refer to [“Data Vault Rules” on page 419](#).

Description — A brief description of what this rule does. The text you enter here will appear in the Description section of the Rule Listing on the Rules page.

Send Alert — Allows you to prevent alerts and alert email notifications from being generated when this rule is violated. Select:

- **Immediately** to allow alert generation and notification, and send the alert to DG Server immediately after violation, or
- **Never** to prevent alert generation and notification. This option is only available for the Block and Prompt rule actions.

Severity — (Control rules only) The Severity Level you select determines the Severity Level of the rules alert and Email notification processing. This field is disabled if you set **Send Alert** to **Never**.

Continue Rule Evaluation — Specify whether to continue to evaluate the event against lower priority rules after it triggers this rule.

Execute Rule After Operation — Select this option to allow your rules to run after the operation. This allows you to perform other operations after

a file operation, such as issue a prompt or capture the destination for the file in File Capture. Primarily used for file capture, changing rule variable states and executing other function calls.

Policy Container — Select the policy container that should contain your rule.

View Component Reference Rules — Displays the Component Reference Rules dialog box. Component Reference Rules are reusable rule definition segments that you can reference from other rules. You can use component rules to minimize the maintenance that your rule definitions require. For more information on component rules, refer to [“Component Rules and Rule Functions” on page 309](#).

Rule Definition — The Rule text. A rule is only triggered when all of its conditions are met. Implement the Digital Guardian XML representation of the logic, evaluations, events, event properties, and data or resources as described in [“Rule Definitions” on page 61](#).

Note: The DG Agent evaluates XML rule expressions from bottom to top, or Last-In, First-Out.

5. Click  . The rule is saved.

Using Windows Environment Variables in Rules

DG allows you to use Microsoft Windows environment variables in rules. Using environment variables enables you to define configuration parameters that can vary from user to user and from system to system. For rules that you want to apply to different users and different systems, using environment variables saves you from having to hard code each configuration parameter. During rule execution, environment variable strings used in a rule are replaced by specific string values and the rule is processed like any other rule.

Support for environment variable expansion applies to all rule string values and variables that contain environment variable substrings. DG supports both default Windows environment variables and user-defined environment variables that are created according to Windows OS specifications. Depending on how they are created, user-defined environment variables might be visible only by the user who created them or system wide. For more information on user-created environment variables, refer to Microsoft documentation.

Environment variables in rules require `##` delimiters. For example, you would write the `userprofile` environment variable as `##userprofile##`. During rule processing, `##userprofile##` expands to `c:\users\<username>`, and `<username>` is translated to a specific username.

Examples of Environment Variables in Rules

Suppose you want to write a block or prompt rule and apply it to all users of a computer. Because each user of a system has a unique user path on that system, entering a hard-coded path for each user would be time-consuming and inefficient. By using the `##userprofile##` environment variable in the rule, you can regulate all users with the same rule property. For example:

```
<and>
  <i n>
    <evtSrcFilePath/>
    <list>
      <string value="##userprofile##\AppData\Roaming\testAppData\CreditCardStatement.pdf" />
      <string value="##userprofile##\AppData\Roaming\testAppData\BankAccount.docx" />
    </list>
```

```
</i n>  
</and>
```

In this rule fragment, the `##userprofile##` environment variable will be user dependent. That is, it will expand to different paths depending on the logged-on user. When the rule is executed, `##userprofile##` is translated to a user-specific profile folder. For example, for a user logged on as `jsmith`, `##userprofile##` will be expanded to `c:\users\jsmith`. The `<curProcessFilePath>` parameter will be replaced with the fully qualified file path of the current process. For `jsmith` user, the rule will be processed as if it was written as follows:

```
<and>  
  <i n>  
    <evtSrcFi l ePath/>  
    <l i s t>  
      <stri ng val ue="c: \users\j smi th\AppData\Roami ng\testAp-  
pData\Credi tCardStatement. pdf" />  
      <stri ng val ue=" c: \users\j smi th\AppData\Roami ng\testAp-  
pData\BankAcct. docx" />  
    </l i s t>  
  </i n>  
</and>
```

Example Using `<equal>` Operator

```
<equal >  
  <stri ng val ue="##systemroot##\system32\cal c. exe" />  
  <curProcessFi l ePath/>  
</equal >
```

When this rule is executed, `##systemroot##` is translated to the Windows root folder, usually `c:\windows`. The `<curProcessFilePath>` parameter will be replaced with the fully qualified file path of the current process. If this rule is processed in context of `calc.exe` process, `<curProcessFilePath>` will be expanded to `c:\win-dows\system32\calc.exe`.

The rule will be processed as if it was written as follows:

```
<equal >  
  <stri ng val ue="c: \wi ndows\system32\cal c. exe" />  
  <curProcessFi l ePath/>  
</equal >
```


Example Using <set> Operator

```
<set>
  <varstring name="MySetVar" scope="global" />
  <string value="##systemroot##\system32\mycalc.exe" />
</set>
```

Executing this rule sets MySetVar to c:\windows\system32\mycalc.exe. The rule will be processed as if it was written as follows:

```
<set>
  <varstring name="MySetVar" scope="global" />
  <string value="c:\windows\system32\mycalc.exe" />
</set>
```

Example Using <in> Operator

```
<in>
  <curProcessFilePath/>
  <list>
    <string value="##systemroot##\system32\mycalc.exe" />
    <string value="##systemroot##\system32\notepad.exe" />
  </list>
</in>
```

The <in> operator will match curProcessFilePath against two string values, which will be expanded to c:\windows\system32\mycalc.exe and c:\windows\system32\notepad.exe, respectively. When the environment variable strings are expanded, the rule will be processed as if it was written as follows:

```
<in>
  <curProcessFilePath/>
  <list>
    <string value="c:\windows\system32\mycalc.exe" />
    <string value="c:\windows\system32\notepad.exe" />
  </list>
</in>
```

Example Using <add> Operator

```
<add mru="false">
  <varstring name="MyProcessFilePath2" scope="global" />
  <string value="##systemroot##\system32\mycalc.exe" />
</add>
```

When this rule is executed, the string value will be expanded to `c:\windows\system32\mycalc.exe`, and the rule will be processed as if it was written as follows:

```
<add mru="false">
  <varstring name="MyProcessFilePath2" scope="global" />
  <string value="c:\windows\system32\mycalc.exe"/>
</add>
```

Displaying a List of Windows Environment Variables

To display a complete list of Windows environment variables available on your system, open a command prompt and enter the `set` command. The environment variables available to you depend on the specific system. Here is an example.

```
C:\Users\test1\AppData\Roaming\testappdata>set

ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\test1\AppData\Roaming
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=IL-VM-WIN81X64
ComSpec=C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_CHECK=NO
HOMEDRIVE=C:
HOMEPATH=\Users\test1
LOCALAPPDATA=C:\Users\test1\AppData\Local
LOGONSERVER=\\IL-VM-WIN81X64
NUMBER_OF_PROCESSORS=2
OS=Windows_NT
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 42 Stepping 7, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=2a07
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=$P$G
```

```

PSModulePath=C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules\
PUBLIC=C:\Users\Public
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\WINDOWS
TEMP=C:\Users\test1\AppData\Local\Temp
TMP=C:\Users\test1\AppData\Local\Temp
USERDOMAIN=IL-VM-WIN81X64
USERDOMAIN_ROAMINGPROFILE=IL-VM-WIN81X64
USERNAME=test1
USERPROFILE=C:\Users\test1
windir=C:\WINDOWS

```

Writing Rules To Block Microsoft Windows Explorer Activity

When you write rules intended to control or block user operations in Microsoft Windows Explorer, you must also block activity by `DLLHost.exe`, because Windows Explorer uses `DLLHost` for some operations. This is especially true when you enable User Access Controls (UAC). To block `DLLHost` activity, add the following clause to your rules:

```

<equal>
  <curProcessImageName />
  <string value="dllhost.exe" />
</equal>

```

For example, this rule can prevent users from using Windows Explorer to transfer files to Dropbox.

```

<and>
  <or>
    <equal>
      <curProcessImageName />
      <string value="dllhost.exe" />
    </equal>
    <equal>
      <curProcessImageName />
      <string value="explorer.exe" />
    </equal>
  </or>

```

```
<like expr = "%\dropbox%">
<evtDestFilePath/>
</like>
<equal >
<evtOperationType/>
<constOpFileMove/>
</equal >
</and>
```

It is very important that you know which actions you intend to block with your rules. Adding the `DLLHost.exe` clause to your rules might cause the Agent to block operations that you do not intend to block. Be sure to verify your rules behave the way you intend.

Guidelines for Creating Network Transfer Upload Rules

Considerations When Creating Prompt Actions

When you create control rules for Network Transfer Upload (NTU) operations, selecting a Prompt action might appear to result in unpredictable behavior. For example:

- Prompts can introduce a delay in the timing of upload operations and can cause application (website) time-outs.
- Prompts might be displayed multiple times resulting from a website continuously re-trying the upload.
- The DG local forensic report might display multiple upload events or a single event with the same file shown multiple times. This reporting might overstate the number of files that you attempted to upload.
- When you upload multiple files, Block or Prompt/Block actions can have different results depending on the target website. On sites that upload multiple files individually, a block of a single file does not affect the remaining files. On sites that upload multiple files as a single request, a block of one file causes all files to be blocked, whether they match a policy or not.

Using Prompts in NTU or NTD Blocking Rules

Many websites do not handle blocking of a requested NTU or NTD and may need to time-out the operation, causing the website to appear unresponsive. Some websites may not recover gracefully even after some time. Depending on the website, the page may appear to hang or show a progress bar continuing, even though there is no upload in progress. Therefore, when writing rules to block NTU or NTD operations, best practices suggest using a DG prompt to recommend that the user refresh the page (Ctrl+5) if DG is blocking an upload or download operation and the site becomes unresponsive. The prompt can also indicate that the user should close the website if it remains unresponsive for an extended period.

Limitations on Detecting NTU and NTD Events

Digital Guardian is not able to detect NTU and NTD events on websites that use a second level of encryption in addition to HTTPS. An example of such a website is MTPProto Mobile Protocol, where the data is first encrypted between a client, such as a web browser using HTTPS, and then encrypted again by a web server.

Working With Network Operation Events

A network operation event represents a connection having been established for TCP/IP or a received or sent datagram for UDP.

Network operations are cached and not sent immediately. This allows DG to aggregate similar activity as it occurs within a time window. If many network operation events are reported within a short period of time, the Agent reports new events that match events in the cache as one event and indicates how many times the event has been repeated (the repeat count). This reduces the amount of noise caused by network operation events, thereby reducing the load on the Server.

DG can capture and report DNS requests. You can use the `constProtocolDNS` protocol type to determine whether a network operation event involved the DNS

protocol. Whenever a DNS request completes, a network operation event is constructed. The network operation shows the hostname and the remote IP address.

Identifying a Specific Type of Network Operation Event

The `constOpNetworkEx` event type allows you to get specific information about a network operation event. `constOpNetworkEx` comes with the `evtNetworkOperation` subtype, which DG uses to specify the type of network operation, such as a connect operation. Using the `evtNetworkOperation` property with `constOpNetworkEx`, you can identify the specific type of network operation the event represents (connect, accept, listen, or unknown). When using `constOpNetworkEx` in rules, you should always check against `evtNetworkOperation` for the subtype you are looking for.

`constOpNetworkEx` also provides the `evtRepeatCount` rule property. The repeat count is the number of repeated network operation events that match the original network operation event within approximately 150 seconds (resolution of 5 seconds). This can be set in the Windows registry:

```
HKLM\System\CurrentControlSet\Services\DgMaster\Parameters\network  
CacheFlushTimeoutSeconds REG_DWORD
```

When used in rules, the `evtRepeatCount` value (n) means that the rule engine will trigger an action when the total number of matching repeated events reaches $n + 1$. The original occurrence of the event is not included in calculating the repeat count. When writing rules, you use `evtRepeatCount` with an integer value to indicate the rule action to take if the repeat count value is reached.

For example, suppose you write a rule to trigger a block action after a certain number of attempts to make an HTTP connection. If you set the repeat count value to 4, the rule engine will trigger the block action when the number of repeated matching events within a given time period is greater than 5.

When you create a report that includes the Repeat Count field, the report displays a Repeat Count column. For example, if an event occurs once and is not repeated, the count will show zero. If the event occurs twice (the original event plus one repeat), the repeat count will show 1 (one), and so on.

Sample Rule Created With the constOpNetworkEx Event Type

The following rule triggers a block action if an attempt to make an HTTP connection occurs more than 5 times within a given time window.

Rule Action: Block

Rule Syntax:

```
<and>
  <and>
    <equal >
      <evtProtocol Type />
      <constProtocol HTTP />
    </equal >
    <greaterThan>
      <evtRepeatCount />
      <i nt value="4" />
    </greaterThan>
  </and>
  <and>
    <evtNetworkOperati on />
    <constNetOpSend />
  </and>
  <equal >
    <evtOperati onType />
    <constOpNetworkEx />
  </equal >
</and>
```

About the Static Rule Analyzer

The static rule analyzer inspects rules before you save them—either on demand or when you save the rule. This saves you time and effort by reducing the chance that you deploy invalid rules. With the rule analyzer in operation, you are less likely to deploy rules that damage your Agent computers or prevent them from operating properly.

Best practices suggest that you test all rules thoroughly before you deploy them in your production environment.

Running the rule analyzer does the following:

- Highlights problems with the rule syntax.
- Performs static analysis of the rule via an extensible XML-based system. The analyzer warns you about potential issues it finds and highlights the affected area of the rule. The warning messages from the analyzer provide guidance for repairing the problems it identifies.

The System > Settings page (Server Configuration — General Settings) in the DGMG provides access to the Static Rule Analyzer configuration options. With these, you specify the client platforms from which you want to see warnings.

How the Rule Analyzer Works

When you click **Validate Rule**, the rule analyzer performs the following actions:

1. Validate the rule text by the existing rule validate method.
2. Run the static rule analysis validation.
3. Open the Rule Analyzer dialog box displaying the results of the validation process:
 - Display one or more rule validation errors if step 1 fails.
 - Display a list of static rule analysis warnings if there are any.
 - Display the "No issues detected" message if both step 1 and step 2 are successful.

When you click , DG runs the Static Rule Analyzer to perform the following actions:

1. Validate the rule text by the existing rule validate method.
2. Run the static rule analysis validation.
3. Do one of the following based on the results of the validation process:

- If the rule analyzer does not detect any issues with the rule, it saves the rule.
- If the rule analyzer detects errors and warnings, the Static Rule Analyzer dialog box opens listing the issues detected:

If there are any errors, you cannot save your changes to the rule. The **Close** option is the only choice. You need to fix the problems with the rule before you can save it.

If the list contains only warnings, you are prompted to confirm the changes or to cancel the save process. Clicking **Confirm** saves any rule changes and closes the analyzer dialog box. Clicking **Cancel** discards all rule changes and closes the dialog box.

4. Click **Confirm** or **Cancel** or **Close** (if available) to dismiss the analyzer dialog box.

Running the Rule Analyzer

To run the rule analyzer to validate your rule, click **Validate Rule** on the rule page. The Validate Rule option is available both when you create a new rule in the Rule Wizard, and when you view the details of a rule by clicking the rule name on the Manage Rules page in the DGMC.

After you click **Validate Rule**, the Rule Analyzer dialog box opens showing you the results of the validation.

Understanding the Output From the Analyzer

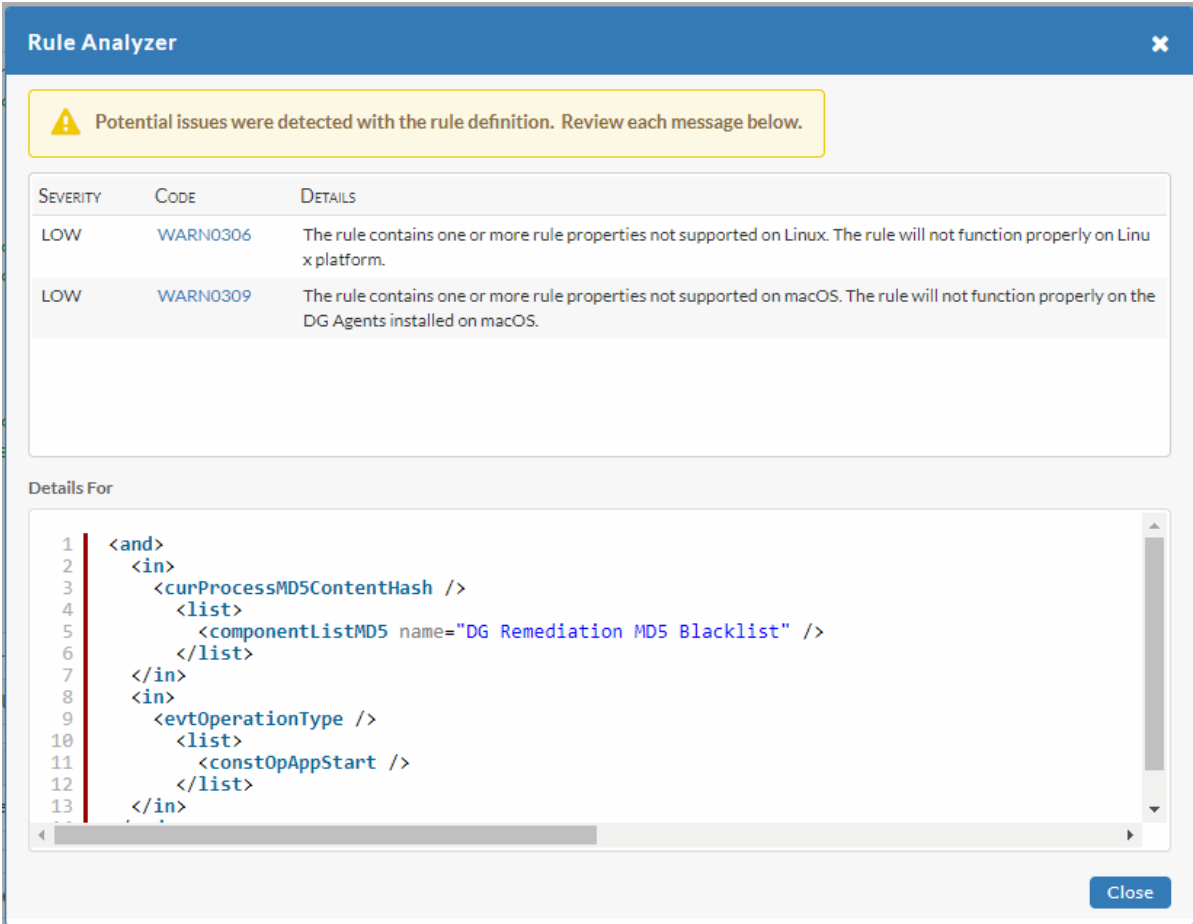
The rule analyzer inspects your rule, testing for a variety of conditions that may make your rule suspect in operation. It looks for rule content that may be inappropriate, such as having AND or NOT operators in a Block or Prompt rule, or rule elements that are not supported on one or more platforms, and reports what it finds in the Rule Analyzer dialog box.

The Rule Analyzer Dialog Box

When you click Validate Rule or when you save a rule, the analyzer runs and reports the findings in the Rule Analyzer dialog box. The dialog box provides details of what the analyzer found. It provides information about the severity of any potential issues, a link in the Code column that takes you to the location in the rule text that has the problem, and the details about the issue.

Severity ranges from low to high.

Severity	Description
Low	Usually indicates compatibility and other issues with the rule.
Medium	The rule as written might affect the performance of your DG environment.
High	The rule as written might have serious effects on your DG environment, including both the stability and performance of your networks.
Error	The rule as written is incorrect and cannot be saved.



The dialog box also shows the XML text for the rule, with highlighting that indicates the potential problems. A scroll bar on the right lets you view the entire rule XML. With this information, you can modify your rule to correct the issues and then revalidate and save the rule.

The dialog box highlights the following kinds of problems:

- Lines that have XML errors
- Lines the have XSD validation errors
- Lines that match rule analyzer criteria



You do not see highlighting for a variety of other problems:

- Missing component rules
- Invalid rule names for a rule element
- Invalid tag name for a rule element
- Invalid content pattern name for an element that is marked internal
- Other validation errors
- Analyzer warnings indicating that some properties or constants are missing or not sufficient

If the rule analyzer does not find any potential issues, you see a message that no issues were detected with the rule.

Editing Rules

You can open and edit existing rules in DGMC.

1. Hover over Policies and select **Policies**. The Manage Policies page opens.
2. In the Rules section, select the rule type that you want to edit. A list of existing rules of that type appears.
3. Click on the rule to edit. The details of the rule appear on the page.
4. Click . The rule becomes editable.
5. Make any desired changes to the rule.
6. Click . The changes are saved.

If you edit a rule that is assigned to one or more policies, the Update Policies using Rules dialog box appears. You can update all policies using the edited rule, or update only specific policies that contain the edited rule. In the Update Policies using Rules dialog box:

Select **Update all Policies using this Rule** to send the edited rule to all policies that contain it.

To update only specific policies, select Update selected Policies to use this Rule. A list appears in the Policies Using this Rule section of the dialog box, listing all the policies that contain the edited rule. Select the check box next to the policies to update. Enter a unique name for the rule in the **New Rule** name field. Click **OK** to save your changes.

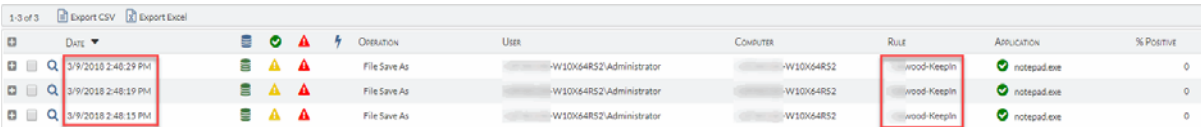
Renaming Rules










You might decide to rename rules to implement a standard format for naming rules after you have created some rules. Standardized rule names can make managing your rules more efficient.

When you rename a rule, the new name replaces earlier instances of the original rule name in earlier alerts that reflect the rule, such as prompts and alerts. As a result, older alerts that had the name of the original rule now show the new rule name.

For example, if you have a rule named “Warn on NTU from Fixed Device” and you rename the rule “Warn on Network Transfer Download from Fixed Device,” earlier alerts and events that the DGMC reports that involve “Warn on NTU from Fixed Device” change to “Warn on Network Transfer Download from Fixed Device.”

The following graphics show the Alerts view in the DGMC before and after you change the name of a rule from “KeepIn” to “KeepIn-test2.” The alerts are the same but they have new rule names.



1-3 of 3 Export CSV Export Excel									
	Date	Icons	Operation	User	Computer	Rule	Attachment	% Positive	
<input checked="" type="checkbox"/>	3/9/2018 2:40:29 PM	  	File Save As	W10X64RS2\Administrator	W10X64RS2	wood-KeepIn-test2	notepad.exe	0	
<input checked="" type="checkbox"/>	3/9/2018 2:40:19 PM	  	File Save As	W10X64RS2\Administrator	W10X64RS2	wood-KeepIn-test2	notepad.exe	0	
<input checked="" type="checkbox"/>	3/9/2018 2:40:15 PM	  	File Save As	W10X64RS2\Administrator	W10X64RS2	wood-KeepIn-test2	notepad.exe	0	




	Date	Operation	User	Computer	Rule	Application	% Positive
1 of 3	5/9/2018 2:48:29 PM	File Save As	W10K64RS2\Administrator	W10K64RS2	wood-keepin-test2	notepad.exe	0
2 of 3	5/9/2018 2:48:19 PM	File Save As	W10K64RS2\Administrator	W10K64RS2	wood-keepin-test2	notepad.exe	0
3 of 3	5/9/2018 2:48:15 PM	File Save As	W10K64RS2\Administrator	W10K64RS2	wood-keepin-test2	notepad.exe	0



If you rename a rule and change the rule action, earlier alerts generated by the rule get the new name, but the icon representing the rule action does not change. For example, if you change a rule action from prompt to block, and change the name as well, earlier alerts generated by the rule still have the prompt action icon, but also have the new rule name. In the preceding graphics, the rule action changed from warn to block between the first and second alerts at 11:17 and 11:22. The block icon does not appear until the third alert, because that is the first alert generated by the blocking version of the rule.

Caution: Entering a new name for a rule in the Rule Builder Wizard or the Edit Rule page does not create a new rule with the new name. It replaces the name of the original rule. The original rule name no longer appears in the DGMC or in reports.

Copying Rules



You can duplicate the settings and definition of an existing rule to create a new rule. This functionality allows you to easily create rules with similar properties.

1. Hover over Policies and select **Policies**. The Manage Policies page opens.
2. In the Rules section, select the rule type to copy. A list of existing rules of that type appears.
3. Click on the rule to copy. The details of the rule appear on the page.
4. Click . A duplicate of the rule is created with the same name as the original incremented by one. For example, if you duplicate a rule named PreventCDBurn, the copy is named PreventCDBurn(1).

5. Click . The rule becomes editable.
6. Make any necessary changes to the duplicate and click . The rule is saved. The new rule appears in the table on the Rules page.

Activating Rules



You must activate a rule before it takes effect. You can leave a rule inactive while you are still developing it.

1. Hover over Policies and select **Policies**. The Manage Policies page opens.
2. In the Rules section, select the rule type to edit. A list of existing rules of that type appears.
3. Click on the rule to edit. The details of the rule appear on the page.
4. Click . The rule becomes editable.
5. Specify a rule status of Active and click . The changes are saved and the rule is now active within any policies to which you have applied it.

Deactivating Rules

You can deactivate a rule to change its rule definition. When a rule is inactive, it is inactive in all policies that contain that rule.


1. Hover over Policies and select **Policies**. The Manage Policies page opens.
2. In the Rules section, select the rule type to deactivate. A list of existing rules of that type appears.

3. Click on the rule to edit. The details of the rule appear on the page.
4. Click . The rule becomes editable.
5. Specify a rule status of Inactive and click . The changes are saved. The rule is now inactive within any policies that contain it.

Exporting Rules

Users with the role of System Administrator, Control Policy Manager, Filter Policy Manager, or Trusted Process Policy Manager can export and save copies of the specific rule type they manage to any location on the network. Exported rules are stored as XML files that you can then import back into Digital Guardian.

You can export rules to move them from evaluation instances to production instances, or to preserve them as part of a disaster recovery plan.

1. Hover over Policies and select **Policies**. The Manage Policies page opens.
2. In the Rules section, select the rule type to export. A list of existing rules of that type appears.
3. Check the rules to export.
4. Click  and select **Export Selected Rules**. The Save As dialog box opens.
5. Specify a location and filename for the exported rules and click **Save**. The rules are exported to an XML file.

Importing Rules

You can import rules that you have exported from Digital Guardian. You might take this step to move rules from an evaluation instance to a production instance, or to restore an earlier version of exported rules.


Imported rules retain the status they had when they were exported. For example, a rule that was active when you exported it will be active when you import it.

When you import multiple items to Digital Guardian, you must import those items in the following order to maintain referential integrity:

1. Import Custom Skins
2. Import Custom Prompts
3. Import Component Rules
4. Import Rules

Caution: When you import vault rules, import all rules the vault rules depend on, such as control rules, before you import the vault rules.

These items contain references to one another. If you import an item before the object of its reference is present, Digital Guardian interprets the references as an error and rejects the import. For example, rules can refer to component rules. If you import a rule that references a nonexistent component rule, Digital Guardian rejects the import. By importing the component rules first, you allow Digital Guardian to validate the references and maintain its integrity.

1. Hover over Policies and select **Policies**. The Manage Policies page opens.
2. In the Rules section, select the rule type to import. A list of existing rules of that type appears.
3. Click  and select **Import Rules**. The Import Utility dialog box opens.

Note: You can import rules only into their original type. For example, you cannot import a control rule as a filter rule.

4. Navigate to the XML file that contains the rules to import and click **Import**. The rules are imported to Digital Guardian. A message box appears to confirm a successful import. The imported rules now appear in the listing of rules.

Creating Parameterized Rules

Parameterized rules are rules that include attributes that allow you to specify values and lists for common items that a DGMC user might want to change. This provides DGMC users the ability to modify certain aspects of read-only rules through the console.

For example, suppose you have a read-only rule that restricts printing documents to authorized corporate printers. To allow DGMC users to choose the printers to allow or deny, you can parameterize the rule by adding attributes in your rule that enable DGMC users to choose a component list of allowed corporate printers and, if needed, a component list of denied corporate printers, and modify the lists as needed. (For detailed information about component lists, refer to the “Component Lists” chapter of this guide.)

The original rule could have been written in the DGMC or provided in a policy pack imported into the DGMC. In either case, after you parameterize the rule, DGMC users who view the rule parameters (by clicking the Rule Parameters tab) see lists that allow them to select and edit the component lists. For an illustration, refer to [“Viewing Parameterized Rule Choices” on page 53](#).

If you use rule functions on your rule set, you can apply parameterization to those rules as well. With a rule function, such as `DG_SetTimerEvent`, you use parameters to enable the rule user to change the input options for the function. For information, refer to [“Parameterizing Rule Functions” on page 44](#).

Requirements for Writing Parameterized Rules


You can write parameterized rules that apply to Control rules and Classification rules.

To write parameterized rules, you use the “choice” attribute and associated attributes. Use the “choice” attribute:

- To modify a component list that the rule refers to, as described in [“Component List Attributes” on page 48](#).
- To provide additional configurable options, such as edit fields and check boxes. For details, refer to [“Using “Choice” With Property Value Types” on page 54](#).



Caution: You can write multiple parameterized rules for a policy, but you should apply those rules only to a single policy. Otherwise, you may unintentionally change other policies as a side effect. This can result in problems managing the system.

To Create Parameterized Rules

1. In the DGMC, hover over Policies, and, under Manage, select **Rules**. The Rules page opens.
2. In the Rules section, select the rule type to create (**Control** or **Classification**). A list of existing rules of that type appears.
3. Click  and select Create New Rule. The Control Rules page or Classification page opens, depending on the rule type you selected.
4. Enter the required information to create the rule. For details about the fields, refer to [“Creating Rules” on page 20](#).
5. To allow DGMC users to change rule behaviors of base rule attributes, click the Parameter Settings tab and select the options you want to enable. For details, refer to [“Base Rule Attributes” on page 57](#).

6. To create a parameterized rule, enter rule XML in the Rule Definition field, using the attributes required for creating parameterized rules. For details, refer to [“Component List Attributes” on page 48](#) and [“Using “Choice” With Property Value Types” on page 54](#).

Note: You can also parameterize existing rules, such as copies of rules imported in a policy pack.

7. After creating the parameterized rule, click . The Rule Analyzer analyzes the rule and lists any issues detected with the rule definition.
8. Click **Close**.
9. Click .
10. Add the rule to a policy.

Parameterizing Rule Functions

If you use rule functions on your rule set, you can apply parameterization to those rules as well.

When you parameterize a function, you create the lists of input arguments for the function and provide the options for each function. When rule users view the rule parameters, they see the input argument label and the list of possible choices or a field into which they enter values.

Example 1 — DG_SetTimerEvent Rule

To set an event timer, you write a rule that uses the DG_SetTimerEvent rule function. DG_SetTimerEvent uses one input argument, Event Timeout. The rule creates a parameter to allow the rule user to define the timeout value. This line of XML defines the label and input field for the event timeout value:

```
int64 id="inputID1" choice="1" label="Event Timeout:" description="Enter a timeout value" units="seconds" value="30"/>
```

A full rule that uses this parameter to set an event timeout looks like the following:

```
<and>
<function name="DG_SetTimerEvent">
<!-- Set a 30 second timer -->
<parameters>
  <int value="0"/>
  <int64 id="inputID1" choice="1" label="Event Timeout: " description="Enter a timeout value" units="seconds" value="30"/>
</parameters>
<return>
  <variant name="TempTimerId" scope="event"/>
</return>
</function>
<in>
  <curProcessImageName/>
  <list>
    <string value="notepad.exe"/>
  </list>
</in>
<in>
  <evtOperationType/>
  <list>
    <constOpAppStart/>
  </list>
</in>
</and>
```

Example 2 — DG_KillProcess Rule

To allow the rule user to specify the process to kill, this rule defines a parameter Event Process ID that accepts the process ID for the process to kill. On the Parameters tab, the rule displays the Event Process ID with a description and a field to enter the process ID. This is the XML that creates the parameter:

```
<int id="inputID1" choice="1" label="Event Process ID: " description="Enter a process ID value to kill" value="30"/>
```

Here is the full rule:

```
<and>
<function name="DG_KillProcess">
  <parameters>
```

```
<int id="inputID1" choice="1" label="Event Process ID: " description="Enter a
process ID value to kill" value="30"/>
</parameters>
<return>
  <varbool name="killResult" scope="event" />
</return>
</function>

<in>
  <curProcessImageName/>
  <list>
    <string value="notepad.exe"/>
  </list>
</in>
<in>
  <evtOperationType/>
  <list>
    <constOpAppStart/>
  </list>
</in>
</and>
```

Example 3 — DG_GenerateCustomEvent Rule

The following rule creates a custom event. To create the event, the function requires as input a process ID. Frequently, this function creates the rule in the context of currentProcessID. For the parameterized rule, however you use a process ID as an input.

```
<and>
<function name="DG_GenerateCustomEvent">
  <parameters>
    <int id="inputID1" choice="1" label="Event Process ID: " description="Enter a
process ID value to kill" value="30"/>
  </parameters>
  <return>
    <varbool name="result" scope="thread" />
  </return>
</function>

<in>
  <curProcessImageName/>
  <list>
    <string value="notepad.exe"/>
```

```
</list>
</in>
<in>
  <evtOperationType/>
  <list>
    <constOpAppStart/>
  </list>
</in>
</and>
```

Component List Attributes

This table describes the component list attributes for writing a parameterized rule that allows the selection and editing of component lists.

Attribute	Values	Default Value	Description
choice	true (1) / false (0)	true (1)	<p>When “choice” is set to true (1), the component list is considered parameterized. The DGM user will be given a choice of the list to use. In the user interface (UI), a drop-down menu appears for this component list. See “Viewing Parameterized Rule Choices” on page 53. The default name of the list is shown in the UI based on the “name” attribute given in the rule.</p> <p>Note: Do not confuse this name with the UI field name taken from the “id” attribute.</p>
id	Letters and numbers. Letters must come first. Can start with an underscore or contain an underscore. Other punctuation marks, mathematical symbols, or special characters are not permitted.	None	Required when “choice” is added to a component list element. The id becomes the field name for merging the user’s choice into the settings.

Attribute	Values	Default Value	Description
label	Valid Values Combinations of alphanumeric characters and _ (underscore). The following special characters are also valid: space : (colon) ? (question mark) % (percent sign) ; (semicolon) [] (square brackets) () (parentheses) / (forward slash) \ (back slash) ' (apostrophe)	None	(Optional, but recommended) When included in the parameterized rule, the label appears to the left of the component list menu box.
description	Any string	None	(Optional, but recommended) When included in the parameterized rule, the description appears above the component list menu box.
listChoices	List of component list names	None	Example: listChoices="archiverList1, archiverList2, archiverList3"
optional	true (1) / false (0)	false (0)	Adding optional ="1" to the parameterized rule indicates that the component list chosen may be an empty list. You might use optional ="1" to allow an empty component list to be populated by a threat feed or by the DGMC user.

Example of Rule Parameterization

The following example code is part of a parameterized rule that would allow users to print only to authorized printers.

```
<and>
<or>
  <not>
    <in op = "like" like = "both" match = "any">
      <varstring name = "Var-evtPrinterName" scope = "event"/>
    <list>
      <componentListString name = "allowed-printer-list" choice = "1" id =
= "evtpri nternameall owedcorporatepri nterlist" listChoi ces =
"allowed-printer-list, allowed-printer-list2, allowed-printer-
list3" label = "Allowed Pri nter Li st:" description = "Please enter a
list of allowed corporate printers by name. (Required)"/>
    </list>
  </in>
</not>
    <in op = "like" like = "both" match = "any">
      <varstring name = "Var-evtPrinterName" scope = "event"/>
    <list>
      <componentListString name = "denied-printer-list" choice = "1" id =
= "evtpri nternamedeni edcorporatepri nterlist" listChoi ces = "deni ed-
pri nter-list, denied-printer-list2, denied-printer-list3" label =
"Denied Pri nter Li st:" description = "Please enter a list of denied
corporate printers by name. (Optional)"/>
    </list>
  </in>
</and>
<equal >
  <varArrayLength name = "evtpri ntername-al lowed-corporate-pri nter-
list"/>
    <int value = "0"/>
  </equal >
  <equal >
    <varArrayLength name = "evtpri ntername-deni ed-corporate-
pri nter-list"/>
      <int value = "0"/>
    </equal >
  </and>
</or>
<set>
  <varstring name = "Var-evtPrinterName" scope = "event"/>
  <evtPri nterName/>
```

```
</set>  
<equal >  
    <evtOperati onType/>  
    <constOpPri nt/>  
</equal >  
</and>
```

Example of Using the Optional Parameter

The following code fragment uses the “optional” parameter default setting (false). Including this code in your parameterized rule would prevent having an empty file extension component list.


```
<i n>  
    <evtDestFileExt/>  
    <list>  
        <componentListString choice="1" optional="0" id="FileExtensionstoInspect" name="FileExtensions_ACI" label="File Extension List:" description="File Extensions to be Inspected"/>  
    </list>  
</i n>
```

Viewing Parameterized Rule Choices

After you write a parameterized rule and add it to a policy, the user can make choices from the policy's Rule Parameters tab. The following example shows the choices available within a policy that contains the parameterized printer control rule. The user can select an allowed printer lists if more than one exist and edit the selected list.

Rule: DLP6011-DI-Classified Control Printer Usage

Control Classified Print operations to printers not listed in Printer Whitelist.

Platforms: 

Select the intended behavior when the rule evaluates to true.

Rule Action:

☒ Alert

☐ Prompt

☐ Block

Please enter a list of allowed corporate printers by name. (Required)

Allowed Printer List::

dlp-printerwhitelist

Edit List

For this rule to target specific classification tags, enter a comma-separated list of tags with no spaces between them. You can optionally place a '%' wildcard to the left, to the right or on both sides of each tag. A single '%' targets any tag.

Classification tags::

%

53

In this example:

- The Rule Action list is provided by the DG Server, as described in [“Base Rule Attributes” on page 57](#).
- The label Allowed Printer List is produced by the “label” attribute in the rule definition.
- The Please enter a list... string that appears above the list is produced by the “description” attribute in the rule definition.
- The **Edit List** button is added automatically to component list items that use the “choice” attribute. DGM users can select a list from the drop-down menu and then click **Edit List** to add or remove members.

Using “Choice” With Property Value Types

You can use the “choice” attribute with property value types in your parameterized rules. This gives you the ability to provide additional configurable options, such as edit fields and check boxes.

Creating Check Boxes

The following code snippet is part of a rule definition that restricts users from reading or writing data to unauthorized USB removable storage devices.

This example uses the bool property value with “choice” to create check boxes (selected, because the default value is “true”) of bus types to which the associated block rule will apply (USB, eSata and FireWire). When the DGM user clears a check box, the resulting rule is filled in with “false.”

```
<or>
  <and>
    <equal >
      <evtSrcBusType/>
      <constBusUSB/>
    </equal >
    <equal >
      <bool choice = "1" id = "choice1394A1" value = "true" label = "FireWire"/>
      <bool value = "true"/>
    </equal >
  </and>
```

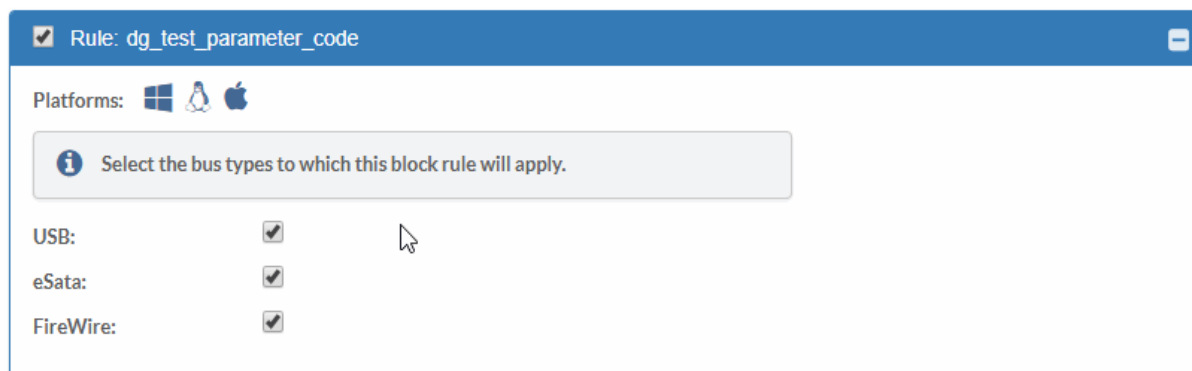
```

<and>
  <equal >
    <evtSrcBusType/>
    <constBusUSB/>
  </equal >
  <equal >
    <bool choice = "1" id = "choiceSataA1" value = "true" label = "eSata"/>
    <bool value = "true"/>
  </equal >
</and>
<and>
  <equal >
    <evtSrcBusType/>
    <constBusUSB/>
  </equal >
  <equal >
    <bool choice = "1" id = "choiceUSB1" value = "true" label = "USB" description
= "Select the bus types to which this block rule will apply."/>
    <bool value = "true"/>
  </equal >
</and>
</or>

```

The following figure shows the `nba_test_parameter_code` section of the Rule Parameters tab for the policy containing the rule. The parameterized rule allows DGMC users to clear bus type check boxes to which they do not want the rule to apply. The rule also allows DGMC users to select and edit component lists.

The DG Server handles the rule action for the alert level, as described in [“Base Rule Attributes” on page 57](#).



Creating Edit Fields

When creating a parameterized rule, you can use the “choice” attribute in your rule definition to create an edit field that allows DGMC users to replace the default value with another allowed value. The default value is shown in the “value” attribute. The “id” attribute is required with the “choice” attribute. You should also include the “label” and “description” attributes.

This rule fragment allows the DGMC user to change the default string value:

```
<set>
  <varstring name="variable" scope="event"/>
  <string value="ABC" choice="1" id="uniqueID"/>
</set>
```

This rule fragment allows the DGMC user to change the default integer value:

```
<set>
  <varint64 name="variable" scope="event"/>
  <int64 value="72347" choice="1" id="uniqueID"/>
</set>
```


This rule fragment allows the DGMC user to change the default time value:

```
<set>
  <vari nt64 name="vari abl e" scope="event"/>
  <i nt64 val ue=" 30" choi ce="1" i d="uni quel D" uni ts="seconds"/>
</set>
```

Base Rule Attributes

The DG Server stores “base rule attributes” for each rule. These attributes define what rule properties are displayed in the DGMC and what actions the DGMC user can take. DGMC users can choose from the displayed properties on the Parameter Settings tab, as shown in the following figure. (Depending on the attributes stored for the rule, additional options may appear.)

The screenshot displays the 'Parameter Settings' tab in the DGMC interface. At the top, there are three tabs: 'Rule Settings', 'Rule Definition', and 'Parameter Settings'. Below the tabs, a message box states: 'Specify the options that the user may select from for the rule properties below. (These properties apply only to rules that have the corresponding parameters defined within the body of their rule definition.)'.

The main content area is titled 'Parameter Settings' and includes a red asterisk icon with the text '= Required'. It is divided into several sections:

- Show in Policy 'Rule Parameters' Tab:** Three radio buttons: 'Normal' (selected), 'Hide', and 'Reference Only'.
- Enable Alert Level Choices:** A checkbox that is unchecked. Below it are five checkboxes: 'Information', 'Low', 'Medium', 'High', and 'Critical'.
- Enable Encryption Type Choices:** A checkbox that is unchecked. Below it are five checkboxes: 'AES256-adaptive', 'Std. ZIP auto-generate PW', 'AES256 ZIP auto-generate PW', 'AES256-password protected', and 'S/MIME'.
- Action Choices:** A section with two columns of checkboxes. The first column has 'Alert' (checked), 'Block', and 'Encrypt'. The second column has 'Prompt' and 'Vault'.
- Enable Prompt Choices:** A checkbox that is unchecked. Below it is a scrollable list of checkboxes: 'Debug Block Prompt', 'DEFAULT AUTHENTICATION', 'DEFAULT BLOCK', 'DEFAULT JUSTIFY', 'DEFAULT PASSWORD', 'DEFAULT PASSWORD CHALLENGE', 'DEFAULT USER DECISION', and 'DEFAULT WARN'.

The options for the rule properties are:

- **Show Rule in Rule Parameters Tab** — Specifies whether the rule appears in the Rule Parameters tab. The options are:

Normal — Displays the selectable options in the Rule Parameters tab.

None — No selectable options are displayed in the Rule Parameters tab. No changes are allowed to the rule behavior.

References Only — Displays only component lists in the Rule Parameters tab. This allows the DGMC user to modify only component lists.

- **Action Choices** — Lists the action options (Alert, Prompt, Block). When only one action appears here, the rule does not provide other choices to select. If you choose the Prompt action choice, both a prompt list and an Alert list appear in the Rule Parameters tab.
- **Enable Alert Level Choices** — If enabled, displays a list of Alert levels that you can select from. If the list is empty, the rule does not allow you to select the level.
- **Enable Encryption Type Choices** — If enabled, displays the list of Encryption types that you can select from. If the list is empty, the rule does not allow you to select the type of encryption.
- **Enable Prompt Choices** — If enabled, displays all DG default prompts, as well as user-created prompts, that you can select from. If the list is empty, the rule does not allow you to select a prompt choice.

The following example shows the Rule Parameters tab for a policy that activates CD media encryption. The rule parameters shown result from the choices made on the rule's Parameter Settings tab. The user can choose the rule action and the prompt, as well as the rule status. The user can also clear the Policy and Rule check boxes to disable the rules, the policy, or both.

CHAPTER 3 Rule Definitions

Rule definitions consist of an event operation and a rule property. The event operation uses a symbolic constant to define the event that the rule governs. The rule property defines the event condition that triggers the rule. For example, CD Burn is an event operation, defined by the symbolic constant `constOpCDBurn`. A file extension of `.doc` is a property, defined by the property `evtSrcFileExt`. A Control Rule that contained both the CD Burn event and the `.doc` property would prevent users from writing word documents to CDs.

The symbolic constants, properties, and operators available in Digital Guardian work for all types of rules, including Classification Rules, Control Rules, Filter Rules, and Trusted Process Rules. The effect of a rule is determined by the rule type and action. The syntax for the rule definitions remains the same, regardless of rule type.

When you have multiple event operations, or multiple rule properties, you can use logical operators to specify the Boolean relationships between those expressions. Logical operators can wrap an entire rule definition, and can be nested. For example, most rules are wrapped with the `<AND>` operator to specify that all the conditions within the rule must be true. Within that `<AND>`, you might also have a nested `<OR>` operator, if you require only some rule properties to be true.

You can use relational operators to perform detailed evaluations of event operations and property values. Relational operators can construct strings to match property values, evaluate lists of property values, or compare multiple expressions to one another.

Rule Evaluation Order

Digital Guardian evaluates rules from the bottom to the top. As soon as the DG Agent meets a rule condition while evaluating the rule that does not match the current activity, it stops evaluating that rule. By placing the most restrictive rule conditions at the bottom of a rule definition, Digital Guardian more quickly determines whether the current user activity is subject to a particular rule. For more information about rule evaluation order, refer to [“Rule Optimization” on page 519](#).

When the rule engine encounters a rule that uses an unusable property, the engine evaluates the condition that uses the property to FALSE and continues to process the rule.

You cannot use the <NOT> operator to change the evaluation of the unusable property condition from FALSE to TRUE in your rule.

Another rule property, `evtSrcFilePolicyTag`, is available only to files that have been inspected by the classification engine.

The Return Element

You can return immediately from a rule at any point by adding a <return> element. For example, this rule snippet always returns True.

```
<return>
  <bool value="true" />
</return>
```

You can also place complex logic or call component rules within a return.

Example:

```

<or>
  <return>
    <and>
      <equal >
        <evtSrcFilePolicyTag />
        <string value="confidential" />
      </equal >
      <equal >
        <evtOperationType />
        <constOpFileCopy />
      </equal >
    </and>
  </return>
  <and>
    <return>
      <varbool name="evaluateCopyOn">
    </return>
    <equal >
      <varbool name="evaluateCopyOn">
      <bool value="false" />
    </equal >
  </and>
</or>

```

In the above example rule, evaluating from the bottom up, if the variable `evaluateCopyOn` is `False`, the rule immediately returns `False` by returning what's in that variable; otherwise if it is a file copy and the file is confidential, it returns `True`. Technically you wouldn't need the second `<return>`, since without it, the above rule would be completed with a `True` result anyway in that case. However, let's say you always want the rule to return `False`.

Example

```

<eval >
  <return>
    <bool value="false" />
  </return>
  <and>
    <equal >
      <evtSrcFilePolicyTag />
      <string value="confidential" />
    </equal >
  </and>
</eval >

```

```
<equal >
  <evtOperationType />
  <constOpFileCopy />
</equal >
</and>
<and>
  <return>
    <varbool name="evaluateCopyOn">
      </return>
    <equal >
      <varbool name="evaluateCopyOn">
        <bool value="false" />
      </equal >
    </and>
  </eval >
```

Note: You can also use return in component rules to return True or False immediately from a component rule. This includes component rules that you use in for-each() loops.

Rule Syntax

All rule definitions share the following general syntax. You can add as many levels as needed to identify the events you want to detect.

- Rule syntax is case-sensitive, but the values provided for comparison are not case-sensitive.
- To insert comments in the rule definition use:

```
<!--insert comment here -->
```


Syntax Example

This example shows the general syntax for a Digital Guardian rule.

```
<and>
<i n>
<expression/>
<list>
<!-- Enter the conditions for the rule. Use the <list> element for
mul ti ple values. -->
<expression/>
<expression/>
</list>
</i n>
<or>
<!-- Use the <or> operator to combine mul ti ple operation types>
<equal >
<evtOperationType/>
<expression/>
<!--Enter the operation type that you want to control.-->
</equal >
<equal >
<evtOperationType/>
<!--Enter the operation type that you want to control.-->
<expression/>
</equal >
</or>
</and>
```

Definition Example

This example shows a rule that prevents users from deleting or recycling files with certain extensions from a local hard drive.

```
<and>
<i n>
<evtSrcFileExt/>
<l i s t>
<!--Each string statement provides a value for the
evtSrcFileExt property. For multiple string values, use a list ele-
ment.-->
<string value="xls"/>
<string value="doc"/>
<string value="c"/>
<string value="cpp"/>
<string value="h"/>
<string value="ppt"/>
<string value="txt"/>
<string value="rtf"/>
</l i s t>
</i n>
<equal >
<!-- Specifies a source drive type of fixed/local.-->
<evtSrcDriveType/>
<constDriveFixed/>
</equal >
<or>
<!--Use the "or" operator to include multiple operation types.-->
<equal >
<!--evtOperationType defines the operations governed by the rule. -
->
<evtOperationType/>
<constOpFileRecycle/>
</equal >
<equal >
<!--evtOperationType defines the operations governed by the rule. -
->
<evtOperationType/>
<constOpFileDelete/>
</equal >
</or>
</and>
```

CHAPTER 4 Rule Properties

Rule properties define the type of event to which the rule applies. You use properties to create rules that achieve exactly the level of security you have in mind. Depending on the property, you can apply it to all events and Agents (in the case of global properties) or the property is specific to a given type of event.

Every rule property has a data type that governs the type of information it can contain. If the rule data type is string, you can give the rule property a regular expression to make the rule more flexible. For example, you can use regular expressions to create a rule property that governs filenames that match a particular pattern rather than creating a list of literal filenames. For more information about creating rules and using regular expressions, refer to [“Rule Definitions” on page 61](#).

The following links take you directly to the sections noted.

- [“Rule Properties and Agent Operating Systems” on page 68](#)
- [“Using Time Properties in Rules” on page 70](#)
- [“Global Rule Properties” on page 72](#)
- [“Address/URL Rule Properties” on page 110](#)
- [“Application Data Exchange Operation Rule Properties” on page 113](#)

- [“Classification Rule Properties” on page 119](#)
- [“Data Vault Rule Properties” on page 126](#)
- [“Device Rule Properties” on page 129](#)
- [“Document Properties Rule Properties” on page 142](#)
- [“Email Operation Rule Properties” on page 152](#)
- [“OLE Insertion Operation Rule Properties” on page 174](#)
- [“Print Operation Rule Properties” on page 174](#)
- [“Process-Related Rule Properties” on page 175](#)
- [“Windows Registry Rule Properties” on page 178](#)
- [“Property Value Types” on page 183](#)
- [“Email Address Format” on page 187](#)
- [“Process Types” on page 69](#)
- [“Network File Transfer Operation Rule Properties” on page 160](#)
- [“Network Operation Rule Properties” on page 163](#)
- [“Print Operation Rule Properties” on page 174](#)
- [“DLL Load \(Image Load\) Rule Properties” on page 156](#)
- [“Registry Event Properties” on page 179](#)
- [“Registry Event Operation Types” on page 180](#)

Rule Properties and Agent Operating Systems

Some rule properties do not apply to all Digital Guardian Agent operating systems. For example, the rule property `evtRegistryDestPath` applies only to rules running on Microsoft Windows operating systems. If you use this property in rules applied to Agents running on a non-Windows operating system, the conditions required to trigger the rules might never occur, creating a potential security hole.

Best practices strongly suggest that you maintain separate rule sets for each operating system.

Process Types

Digital Guardian interprets rule properties based on the relationship of a process to the current process. You can use the following processes as rule properties:

- Current Process
- Parent Process
- Source Process

Current Process

The current process is the active process when you are evaluating an event against a rule. For example, if you are pasting content into a Notepad file, Notepad is the current process.

You can recognize rule properties that deal with the current process by their `curProcess` prefix.

Parent Process

A parent process is a process that has spawned a child process. For example, an instance of Microsoft Excel that opens a second instance would be a parent process. In Windows, most processes are children of Windows Explorer.

You can recognize rule properties that deal with the parent process by their `parentProcess` prefix.

Parent processes are particularly important for Data Vault rules, where you want to maintain the same level of security for all children of the parent application. For more information about Data Vault rules, refer to [“Data Vault Rules” on page 420](#).

Source Process

The source process is the originating process for an application data exchange. For example, if you copied content out of a Microsoft PowerPoint file, PowerPoint is the source process. The process into which you paste the content is the current process. Source processes are used only with ADE events.

You can recognize rule properties that deal with the previous process by their `evtSrc` prefix. Source processes are important particularly for ADE-related rules to regulate users who open one application and attempt to paste content into another application.

Using Time Properties in Rules

Many rule properties use time as a measure, for example `agentCurrentTime`, `agentCurrentDateTime`, `curProcessModifyTime` and `parentProcessAccessTime`. The Agent tracks time in 100 nsec intervals (0.1 μ sec). For that reason, do not use equal operations (equal to in the rule wizard) when you are comparing time values in rules. When you enter time in rules, you enter hours and minutes. The chances of a successful match are very low if you enter hours and minutes while the system is using hours, minutes, seconds and milliseconds to match the time.

Follow these guidelines when you use time in comparisons in rules:

- If possible, use `agentCurrentDateTime` rather than `agentCurrentTime` in rules that use timers.
- Use the relational operators `greaterThan` and `lessThan` (greater than and less than in the rule wizard).
- Specify a range of times by configuring a rule like this example that tests for time between values:

```
<and>
  <lessThan>
    <dateTimeCreatedTime />
    <dateTimeValue="07/25/2014 12:46 AM" />
  </lessThan>
  <greaterThan>
    <dateTimeCreatedTime />
```

```
        <dateTime value="07/25/2014 12:44 AM" />  
    </greaterThan>  
<and>
```

- agentCurrentTime resets to 0 each day. It does not account for date changes. When you use it, consider the following example:

Suppose you set a timer for 10 minutes that is triggered at 23:55 on the Agent. At midnight, the Agent time resets to 00:00:00. If that happens, the 10 minute timer never expires because it expects to run until 24:05, a time that agentCurrentTime never reaches. In other words, when you use agentCurrentTime, consider carefully whether the time will roll over to the next day and reset to 00:00. Construct your rules to avoid this operation. One approach is to use agentCurrentDateTime that tracks both time of day and date.

One case for using agentCurrentTime is to monitor and report events that might occur over the same time span on any day. For example, monitoring your enterprise for events that occur outside of your working hours every day.

Global Rule Properties

Global rule properties apply across your installation. Valid operating systems are:

- Windows
- Linux
- macOS

Note: Operation type “All” does not include user logon and logoff.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
agentAdapterCount	integer	All	Windows, Linux, macOS	The number of active network adapters on the Agent computer.
agentCurrentDayOfWeek	constant	All	Windows	The current day of the week. For a list of valid days, refer to “Days of the Week” on page 193 .
agentCurrentDateTime	time	All	Windows	Contains the current system time and date on the Agent. The date and time are calculated as follows: Current date and time is equal to ([number of seconds since 31 Dec. 1600 to current date] * [10 ⁷]) because the time and date are based on the system clock that runs in 100 nanosecond intervals. As a result, current time and date is a very large number.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
agentCurrentTime	time	All	Windows, Linux, macOS	<p>Contains the current system time on the Agent.</p> <p>The time is calculated as follows:</p> <p>Current date and time is equal to ([number of seconds since 31 Dec. 1600 to current date] * [10⁷]) because the time and date are based on the system clock that runs in 100 nanosecond intervals. As a result, current time is a very large number.</p> <p>Resets to 00:00 at midnight.</p>
agentFileVersion	string	All	Windows	The version of the Digital Guardian Agent.
agentGatewayMac	macAddress	All	Windows, Linux, macOS	Contains the gateway Mac address values of all available network cards on the Agent.
agentIPAddress	ipAddress	All	Windows, Linux, macOS	Contain the IP address value of the available network card. In addition, it includes the value for local host (127.0.0.1). Can be an array of all available network card addresses as well.
agentIsLaptop	Boolean	All	Windows, Linux	True when the Agent computer is a laptop; otherwise False.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
agentIsRegistered	Boolean	All	Windows, Linux, macOS	True when the Agent has successfully contacted the DG Server; otherwise False. Agents can lose registered status after five unsuccessful attempts to contact the server or if the Agent IP address or routing tables change.
agentIsTerminal Server	Boolean	All	Windows Linux, macOS	True when the Agent computer is a terminal server; otherwise False.
agentIsVirtualSession	Boolean	All	Windows, Linux	True when a process is running over Windows Remote Desktop to an Agent running on a Citrix/ Terminal server; otherwise False.
agentLastServer Comm	integer	All	Windows, Linux, macOS	Specifies, in minutes, the time elapsed since the DG Agent successfully communicated with the DG Server.
agentType	string	All	Windows, Linux, macOS	Lets you specify the type of Agent that should trigger the rule. Valid types are: <ul style="list-style-type: none"> • constAgentTypeLinux • constAgentTypeMac • constAgentType Windows

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
componentListIpAddress	string	All	Windows, Linux	Specify the contents of a component list as IP Addresses.
componentListMD5	string	All	Windows, Linux	Specify the contents of a component list as MD5 hashes.
componentListSHA1	string	All	Windows	Specify the contents of a component list as SHA1 hashes.
componentListSHA256	string	All	Windows	Specify the contents of a component list as SHA256 hashes.
componentListSize	string	All	Windows	Specify the current number of items in a component list.
componentListString	string	All	Windows, Linux, macOS	Specify the contents of a component list as strings—domains, email addresses and generic strings.
componentListVersion	string	All	Windows	Specify the current version of a component list.
curProcessAccessTime	int64	All	Windows	The time the process file was last accessed.
curProcessModifyTime	int64	All	Windows	The time the process was last modified. Reported as the local file time, not UTC.
curProcessCmdLine	string	All	Windows, Linux, macOS	The contents of the command line used to start the current process.
curProcessCompanyName	string	All	Windows, macOS	The company name associated with the process file version resource block.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
curProcessCreated Time	int64	All	Windows	The time the process file was created. Reported as a local file time, not UTC.
curProcessFile Attributes	int32	All	Windows	The attributes on the process file.
curProcessFile Description	string	All	Windows	The file description associated with the process file version resource block.
curProcessFilePath	string	All	Windows, Linux, macOS	The fully qualified file path of the current process.
curProcessFile PolicyTag	string	All	Windows	Gets the tag for the ID of the parent process associated with the current process.
curProcess FileVersion	string	All	Windows	<p>The file version associated with the process file version resource block.</p> <p>Note: File Version properties use the application version as a string. To view this information, view the file properties for the application executable. Under the Version tab, in the Other version information area, select File Version. The string shown is the version you should use with your rules.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
curProcessID	int32	All	Windows, Linux, macOS	The ID of the current process. For more information, refer to “Process-Related Rule Properties” on page 175 .
curProcess ImageName	string	All	Windows, Linux, macOS	The image name of the current process. For Linux Agents, this property checks against the executable name.
curProcessIsBrowser-PluginLoaded	Boolean	Network Operations	Windows, macOS	The process that identifies whether the DG extension for Firefox is loaded and enabled in the browser. True when the plugin is enabled.
curProcess LegalCopyright	string	All	Windows	The legal copyright associated with the process file version resource block.
curProcessMD5 ContentHash	md5Hash	All	Windows, Linux, macOS	The GUID representing the MD5 content signature of the process.
curProcessModify Time	int64	All	Windows	The time the process file was last modified. Reported as a local file time, not UTC.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
curProcessNoPrompting	Boolean	All	Windows	<p>True when the current process has been marked with the process flag MPO_NO_PROMPTING; otherwise False.</p> <p>For more information about this process flag, refer to <i>Digital Guardian Management Console User's Guide</i>.</p>
curProcessOriginalName	string	All	Windows	<p>The original file name property associated with the current process. Most software manufacturers provide this property with executable files.</p> <p>For example, the original file name for Windows Solitaire is sol.exe. Even if a user renames the file, the original file name remains sol.exe.</p> <p>To view the Original File Name property of an executable, right-click on a non-installer executable file, select Properties, and select the Version tab in the Properties dialog box.</p>
curProcessProcessFlags	string	All	Windows	<p>The process flags assigned to the process that generated the event. For example NI+ND+DMG.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
curProcess ProductName	string	All	Windows, macOS	The product name associated with the process file version resource block.
curProcess ProductVersion	string	All	Windows, macOS	The product version associated with the current process file version resource block.
curProcessSHA1 ContentHash	sha1hash	All	Windows	The SHA1 hash of the process that generated the event.
curProcessSHA256 ContentHash	sha256hash	All	Windows	The SHA256 hash of the process that generated the event.
curProcessTopUrl IPAddress	string	Network Operations	Windows	<p>Translates supported Web browser top level URL domain names to IP addresses. Applies only to IPv4 addresses. You can use this property with ipMask and IPAddress rule properties. For example:</p> <pre> <not> <i pMask mask="10. 10. 13. 0/11"> <curProcessTopUrl IPAddress /> </i pMask> </not> </pre>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
curProcess WindowTitle	string	All	Windows, Linux	<p>The main window title string of the current process. For web browsers (Internet Explorer, Chrome, Firefox, and Edge), the window title of the tab that has the focus. Websites determine what is displayed for the title.</p> <p>For processes other than browsers, if a dialog box has focus, Digital Guardian traces back to the main window to determine the title string.</p>
dnsHostAvailable	Boolean	All	Windows, macOS	<p>True when the DNS host specified in the name attribute is available on the network; otherwise False.</p> <p>Note: Use an IP address to specify the DNS host. For example, <dnsHostAvailable name="10. 10. 10. 251" /></p> <p>Use this property to determine if the Agent is running on a known network. DNS Host lookup uses nslookup on the IP address provided to determine if it is a DNS server.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
driverRunning	Boolean	All	Windows	<p>True when the specified driver is running; otherwise False.</p> <p>In rule syntax, do not include the .sys driver file extension.</p>
evtConnectionType	Boolean	All	Windows	<p>Use this property to determine the type of connection the Agent computer is using. Evaluates to True when the connection type matches the constant type. Specify the connection types with one of the following constants:</p> <ul style="list-style-type: none">• constConnNormal• constConnWireless

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtCurrentRuleAction	string	All	Windows	<p>Allows you to write rules that take action based on the result of rules that have already matched the event (higher priority rules) but where you specified Continue Rule Evaluation when you wrote the rule. With this new property, you can:</p> <ul style="list-style-type: none">• Evaluate the current action the rule system is returning for an event.• Use the result of a prompt from a previous rule, such as a user decision prompt, in a later (lower priority) rule.• Base additional rules on whether previous rules would have blocked the event.• Change the action of a previous rule by matching on a higher-execution-order rule.• Check lower priority rules to see what a previous higher priority rule set. <p>Use with the following constants:</p> <ul style="list-style-type: none">• constRuleActionBlock• constRuleActionContinue• constRuleActionEncrypt• constRuleActionNone• constRuleActionVault <p>constRuleActionNone assesses whether any rule has matched the event under evaluation.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtCurrentUserId	string	All	Windows, Linux, macOS	Evaluate the security ID (SID) of the currently logged on user. For examples of using this property and determining the user SID, refer to “Block File Delete Events for Non-owner of File” on page 447 and “Test a User Security ID” on page 447.
evtDestBusType	constant	All cd and file operations	Windows, Linux, macOS	The bus type of the destination file. For a list of valid bus types, refer to “Bus Types” on page 190.
evtDestDriveType	constant	All CD, DVD, file and data exchange operations	Windows, Linux, macOS	The drive type of the destination file. If no destination drive type is available, the source drive type will be used. For a list of valid drive types, refer to “Drive Types” on page 194. Note: For rules governing File Write events use the evtSrcDriveType property rather than the destination drive property. File write does not have destination file paths.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtDestEncryptionType	int8	All CD, DVD, file and data exchange operations	Windows	<p>The encryption type applied to the event destination file if the file is encrypted. The possible values are:</p> <ul style="list-style-type: none">• 0 = No Encryption• 1 = AES256 Adaptive• 2 = 3DES Adaptive• 3 = Std ZIP Adaptive• 4 = Std ZIP Password• 5 = AES256 ZIP Adaptive• 6 = AES256 ZIP Password• 7 = AES256 Password Protected• 8 = 3DES Password Protected• 9 = In Policy Raw• 10 = Cryptography client default (for clients that do not care)• 11 = SMIME• 255 = NTFS
evtDestFileExt	string	All cd, file and Data Exchange operations	Windows, Linux, macOS	The extension of the destination file, if available.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtDestFileIs Encrypted	Boolean	All	Windows, Linux, macOS	True when the destination file has been encrypted by Digital Guardian Removable Media Encryption; otherwise False.
evtDestFileModified Time	DateTime	All	Windows, Linux	The local time the destination file used in the process was most recently modified.
evtDestFilePath	string	All cd, file and Data Exchange operations	Windows, Linux, macOS	The fully qualified file path of the destination file, if available.
evtDestFileSize	integer	All	Windows, Linux, macOS	<p>The size, in bytes, of the destination file.</p> <p>Note: You must include the <code>int64</code> property in your rule when you use <code>evtDestFileSize</code> to control destination files by file size. For example, to block access to files larger than a specified value <code>filesize</code>, your rule could include the following rule XML:</p> <pre><greaterThan> <evtDestFileSize/> <int64 value="filesize"/> </greaterThan></pre> <p>The <code>int64 value="filesize"</code> command is required for the rule to fire properly.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtDestFileSupportsADS	Boolean	File	Windows, macOS	Detects whether a destination file system supports writing file/alternate data streams. If the file system does not support ADS, classification information may be lost as a result of the operation. The property returns False or True for local drives and removable drives. However, the property returns True for all network file systems. Use this property only in rules that apply to fixed and removable drives.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtDestIsWindows CdBurnFolder	Boolean	CD operations	Windows	<p>Detects whether the destination of an event is the native Microsoft Windows CD burn directory. This is the directory Microsoft operating system uses to stage files that users write to CD/DVDs.</p> <p>This property can be used to detect when you need to encrypt files before the user writes to the CD/DVD.</p> <p>Note: When the rule engine evaluates this property, it makes a single call to the registry in the lifetime of the current process to retrieve the process's User's Identity-based value. In performance-sensitive applications, put this property at the top of the rule to be evaluated last.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtDestSupportsEncrypt	Boolean	File Operations	Windows, Linux, macOS	Detects whether the destination volume supports encryption and you can write rules to encrypt files there. If this is False, you cannot write rules to encrypt files there. You could use this to block file operations when the file cannot be encrypted as desired. This returns False or true for local drives and removable drives. However, it returns true for all network file systems.
evtDomainName	string	All	Windows, Linux, macOS	The domain that contained the user who generated the event. If the user is not in a domain, this is the computer name.
evtDriveType	constant		Windows	The drive type in the operation. For a list of valid drive types, refer to “Drive Types” on page 194 .
evtEventDisplay-Name	string	All	Windows, Linux, macOS	Gets the name of the event. Typically used for Custom Events to ensure that the event is displayed in the DG forensics reports.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtExtMatches FileType	Boolean	All file operations.	Windows	<p>True when the file extension of the specified file matches the file type as defined by the file header information. This information can help you identify files that are trying to hide their purpose.</p> <p>Caution: Use this only in rules that include the file extension or the file type to test. Using this without specifying file types or extensions may generate a very large number of events. Your Collection database might fill up and ETL might fail.</p> <p>If the Agent cannot determine whether the file type matches the extension, the result is False. Any file whose extension is not in <code>impfl.t.xml</code>, but whose signature matches a signature within the test file, returns False. Also, the default value for this property is False, so entries in the cache that were not checked for file type, such as directories, return False as well.</p>
evtHookAddress	int64	Hooking operations	Windows	The starting address for the hooked function.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtHookFunctionName	string	Hooking operations	Windows	The name of the hooked function.
evtHookHookedBeforeDG	int8	Hooking operations	Windows	Whether another module hooked the function before DG.
evtHookModuleName	string	Hooking operations	Windows	The name of the hooked module.
evtHookOwnerModule	string	Hooking operations	Windows	The name of the owner of the hooked module.
evtIsAfterOperation	Boolean	All application file operations except Save As, Print and CD/DVD write	Windows	Directs classification rules to execute after the operation. This ensures that the operation is complete before the Agent acts, such as being sure the destination file exists before trying to classify it. Refer to “Using evtIsAfterOperation” on page 108 for more information.
evtIsDomainUser	Boolean	All	Windows, Linux, macOS	True when the user is a domain user, False if the user is a local user.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtIsLocalAdmin	Boolean	All	Windows, Linux, macOS	<p>True when the current user has administrator rights on the local machine. False otherwise.</p> <p>True for members of the local administrator group on Windows if the Microsoft group SID associated with the group's access token has the SE_GROUP_MANDATORY attribute. This attribute disables changing the group membership for the access token. The DG rule implements the most restrictive interpretation of the administrator group membership. False if SE_GROUP_MANDATORY is not set.</p> <p>For more information on SID attributes in an access token, refer to https://msdn.microsoft.com/en-us/library/windows/desktop/aa379596(v=vs.85).aspx.</p>
evtIsOverwrite	Boolean	All	Windows	<p>True when the event is overwriting an existing file; otherwise False. This event includes overwrites performed by file copy, file move, and file rename events. It does not cover file save as or write events.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtOperationType	constant	All	All	The operation type of the event being performed. For a list of valid operation types, refer to “Operation Types” on page 199 .
evtParentOperation Type	constant	All	Windows	<p>The operation type of the low-level event being performed by a parent process. Generally, you should use this property with a <not> operator to filter out low-level events performed within a higher level event. For example, you could use this property with a <not> operator to filter out file read, file create, and file write events performed within a file copy.</p> <p>For a list of valid operation types, refer to “Operation Types” on page 199.</p>
evtRegistryDestPath	string	Registry Event	Windows	The destination path for the operation. This is an empty string except for rename operations.
evtRegistryOperation Type		Registry Event	Windows	The operation type that occurred. This is defined by an enumeration. For more information, refer to Registry Event Types.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtRegistrySrcPath	string	Registry Event	Windows	The source path for the operation. This always contains the full path to the registry item the operation was performed on.
evtRegistryValue DWORD	int32	Registry Event	Windows	The actual DWORD of the value involved in the operation. This is the final value in the case of set operations.
evtRegistryValue MultiStringSZ	string	Registry Event	Windows	List of strings involved in the multi-string operation. This can be quite long. The DG Server will only hold roughly the first 2048 bytes of data in the DB.
evtRegistryValue QWORD	int64	Registry Event	Windows	The quad word of the value involved in the operations.
evtRegistryValue StringSZ	string	Registry Event	Windows	The string value involved in the operation.
evtRegistryValue Type	int8	Registry Event	Windows	Specifies the value type the operation was on. This is an enumeration. Refer to "Registry Value Types."
evtRuleName	string	All	Windows	The name of the triggered rule.
evtSrcBusType	constant	All	Windows, Linux, macOS	The bus type of the source file. For a list of valid bus types, refer to "Bus Types" on page 190 .
evtSrcDriveMatches Dest	Boolean	All	Windows, Linux, macOS	True when the event is taking place on the same drive, otherwise False.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtSrcDriveType	constant	All except network operations	Windows, Linux, macOS	The drive type of the source file. For a list of valid drive types, refer to “Drive Types” on page 194 .
evtSrcEncryptionType	integer	All CD, DVD, file and data exchange operations	Windows	The encryption type applied to the event source file if the file is encrypted. The possible values are: <ul style="list-style-type: none"> • 0 = No Encryption • 1 = AES256 Adaptive • 7 = AES256 Password Protected • 11 = SMIME
evtSrcFileExt	string	All except network operations	Windows, Linux, macOS	The extension of the source file.
evtSrcFileIsEncrypted	Boolean	All	Windows, Linux, macOS	True when the source file has been encrypted by Digital Guardian Removable Media Encryption; otherwise False.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtSrcFileModified Time	DateTime	All	Windows, Linux	The local time the source file used in the process was most recently modified. To use this property, the dgfsmon_EnableFileMod-Time registry setting must be enabled (set to 1). This setting is stored in: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControl Set\services\DGMaster\Parameters\dgfsmon
evtSrcFilePath	string	All except network operations	Windows, Linux, macOS	The fully qualified file path of the source file.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtSrcFileSize	integer	All	All	<p>The size, in bytes, of the source file.</p> <p>Note: You must include the <code>int64</code> property in your rule when you use <code>evtSrcFileSize</code> to control access to source files by file size. For example, to block access to files larger than a specified value <code>filesize</code>, your rule could include the following section:</p> <pre><greaterThan> <evtSrcFileSize/> <int64 value="filesize"/> </greaterThan></pre> <p>The <code>int64 value="filesize"</code> command is required for the rule to fire properly.</p>
evtSrcFileSupportsADS	Boolean	File	Windows, macOS	<p>Detects whether a source file system supports writing alternate data streams. If it does not, classification information may be lost as a result of the operation. The property returns false or true for local drives and removable drives. However, the property returns true for all network file systems. Use this property only in rules that apply to fixed and removable drives.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtSrcFileType	string	All operations	Windows	<p>Contains a string that reports the file type as determined by the implicit filter file. For example, if the filter entry in <code>implicit.xml</code> is <code><filetype name="document/msofficelegacy"></code>, <code>evtSrcFileType</code> will contain the value <code>document/msofficelegacy</code> for a Microsoft Office document with the extensions <code>doc</code>, <code>ppt</code>, <code>xls</code>, <code>vsd</code> or <code>msi</code>. Refer also to evtExtMatches FileType.</p> <p>You can edit the implicit filter file (<code>implicit.xml</code>) to add new file types. This enables you to allow and test against new types, which can be useful in protecting against data loss and advanced threats.</p> <p>For the default list of file types included in the implicit filter file, refer to "Default Implicit Filter File Types and Events" in <i>Digital Guardian Management Console User's Guide</i>.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtSrcIsWindows CdBurnFolder	Boolean	CD operations	Windows	<p>Detects whether the source of an event is the native Microsoft Windows CD burn stage folder or a temporary directory. This is the directory Microsoft operating systems use to stage the files users write to CD/DVDs.</p> <p>You can use this property to detect when you need to encrypt files before the user writes the files to CD/DVD.</p> <p>Note: When the rule engine evaluates this property, it makes one call to the registry in the lifetime of the current process to retrieve the process's User's Identity-based value. In performance-sensitive applications, put this property at the top of the rule to be evaluated last.</p>
evtSrcProcess AccessTime	int64	Application Data Exchange	Windows	The last time the source process for the ADE event was accessed.
evtSrcProcess CreatedTime	int64	Application Data Exchange	Windows	The time that the process file for the source process was created. Reported as local time, not UTC.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtSrcProcessFileAttributes	int32	Application Data Exchange	Windows	The file attributes on the source process that generated the event.
evtSrcProcessFilePolicyTag	string	Application Data Exchange	Windows	Gets the tag for the process ID of the source process that generated the event.
evtSrcProcessModifyTime	int64	Application Data Exchange	Windows	The last time that something modified the source process of the ADE event.
evtSrcProcessProcessFlags	string	Application Data Exchange	Windows	The process flags assigned to the source process that generated the ADE event. For example "NI+ND+DMG".
evtSrcProcessSHA1ContentHash	sha1	Application Data Exchange	Windows	The SHA1 hash of the source process that generated the ADE event.
evtSrcProcessSHA256ContentHash	sha256	Application Data Exchange	Windows	The SHA256 hash of the source process that generated the ADE event.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtSrcSupportsEncrypt	Boolean	File Operations	Windows, Linux, macOS	Detects whether the source volume supports encryption, and you can write rules to encrypt files there. When this is false, DG cannot encrypt the source file and you cannot write rules to encrypt files there. You could use this to block file operations when the file cannot be encrypted as desired. This returns false or true for local drives and removable drives. However, it returns true for all network file systems. Use this property only in rules that apply to fixed and removable drives.
evtTimerID	int64	constOpTimer	Windows, Linux, macOS	Identifies the timer involved in an event. Use this to verify the timer in rules that handle timers to ensure you are receiving the expected or correct timer event.
evtTimerTimeoutMS	int64	All	Windows, Linux	Specify, in milliseconds, when the timer fires after your rule sets it.
evtUserName	string	All	Windows, Linux, macOS	The name of the user who generated the event.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtUserResponse	string	All	Windows	The justification string a user entered when prompted for justification. The user response is put into the evtUserResponse property for the event. You can write a rule to prompt for the user response and then use evtUserResponse in another rule to make a decision.
evtUserSurveyResponseNumber	integer	All	Windows	The number (integer) of the response a user selected in a prompt survey. The response number is put into the evtUserSurveyResponseNumber property for the event. You can write a rule to prompt for the response number and then use evtUserSurveyResponseNumber in another rule to make a decision. In this case, the text of the selected survey entry is placed in evtUserResponse.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
hostIsAvailable	Boolean	All	Windows, macOS	Determine whether a specified location is available. You specify an IP address or host name with the name= element. The Agent creates a list of unique sites from the hostIsAvailable properties rule set. Periodically, the Agent checks the availability of each location on the list and stores the result. The Agent tests location availability by checking port 80 over TCP. If a connection to the host can be established, or the host port is open but refuses the test connection, the host is considered available. When the rule fires, the rule engine checks the list of locations to see if the specified location is marked available, returning true if it is. Otherwise the property returns false.
parentProcess AccessTime	int64	All	Windows	The last time the parent process for the event was accessed.
parentProcess CmdLine	string	All	Windows, Linux, macOS	The contents of the command line used to start the parent process.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
parentProcess CompanyName	string	All	Windows, macOS	The company name associated with the process file version resource block of the parent process.
parentProcess CreatedTime	int64	All	Windows	The time the parent process was created for the source process of the event. Reported as local time, not UTC.
parentProcess FileAttributes	int32	All	Windows	The file attributes on the parent process for the process that generated the event.
parentProcess FileDescription	string	All	Windows	The file description associated with the process file version resource block of the parent process.
parentProcessFilePath	string	All	Windows, Linux, macOS	The fully qualified file path of the parent process.
parentProcess FilePolicyTag	string	All	Windows	Gets the tag for the process ID of the parent process of the current process that generated the event.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
parentProcess FileVersion	string	All	Windows	<p>The file version associated with the process file version resource block of the parent process.</p> <p>Note: File Version properties use the string version of the application version. To view this information, view the file properties for the application executable. Under the Version tab, in the Other version information area, select File Version. This string is the version that you should use with your rules.</p>
parentProcessID	int32	All	Windows, Linux, macOS	<p>The ID of the parent process for the current process. For more information, refer to “Process-Related Rule Properties” on page 175.</p>
parentProcess ImageName	string	All	Windows, Linux, macOS	<p>The image name of the parent process.</p> <p>For Linux Agents, this property checks against the executable name.</p>
parentProcessIsBrowserPluginLoaded	Boolean	Network Operations	Windows	<p>The parent process of the current process that identifies whether the DG extension for Firefox is loaded and enabled in the browser. True if the plugin is enabled..</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
parentProcessLegalCopyright	string	All	Windows	The legal copyright associated with the process file version resource block of the parent process.
parentProcessMD5ContentHash	md5Hash	All	Windows, Linux, macOS	The GUID representing the MD5 content signature of the parent process.
parentProcessModifyTime	int64	All	Windows	The last time that something modified the parent process for the event.
parentProcessProcessFlags	string	All	Windows	The process flags assigned to the parent process of the process that generated the event. For example "NI+ND+DMG".
parentProcessProductName	string	All	Windows, macOS	The product name associated with the process file version resource block of the parent process.
parentProcessProductVersion	string	All	Windows, macOS	The product version associated with the process file version resource block of the parent process.
parentProcessSHA1ContentHash	sha1	All	Windows	The SHA1 hash of the parent process of the process that generated the event.
parentProcessSHA256ContentHash	sha256	All	Windows	The SHA256 hash of the parent process of the process that generated the event.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
parentProcessTopUrlIpAddress	string	Network Operations	Windows	<p>When a child process is performing an operation in the browser, this captures the URL of the parent process in the browser address bar and returns the corresponding IP address. You can test this IP address against known IP addresses. You might, for example, use this in an NTU control rule you apply to Mozilla Firefox default file uploads to IP addresses. For default file uploads, Firefox uses a child process pl ug-i n conta i ner. exe to do the file upload. Your DG rule needs to use this property to trigger based on IP address:</p> <pre><and> <or> <i n> <parentProcessTopUrl I P Address /> <l i s t> <i pAddress val ue="i p1 for Yahoo. com" /> <i pAddress val ue="i p2 for yahoo. com" /> </l i s t> </i n> <equal > <curProcessI mageName /> <stri ng val ue="pl ugi n- conta i ner. exe" /> </equal > </or> <equal > <evt0perati onType /> <const0pNetTransferUpl oad /> </equal > </and></pre>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
parentProcess WindowTitle	string	All	Windows, Linux	<p>The main window title string of the parent process, if any. For web browsers (Internet Explorer, Chrome, Firefox, and Edge), the parent process's window title of the tab that has the focus. Websites determine what is displayed for the title.</p> <p>For non-browser processes, if a dialog box has focus, Digital Guardian traces back to the main window to determine the title string.</p>
prevProcessID	int32	Data Exchange operations	Windows	The ID of the process that provided the data for the ADE operation; the source of the copy and paste operation.
prevProcess WindowTitle	string	All	Windows, Linux	<p>The main window title string of the previous process, if any. For web browsers (Internet Explorer, Chrome, Firefox, and Edge), the window title of the tab that has the focus. Websites determine what is displayed for the title.</p> <p>For non-browser processes, if a dialog box has focus, Digital Guardian traces back to the main window to determine the title string.</p>

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
serverAvailable	Boolean	All	Windows, macOS	When a rule that uses this property fires, this directs the Agent to contact the server. The result is True when the Agent can communicate with the DG Comm server. False otherwise. Best practice is to use this in your rules instead of using registerOnIpChange in your Agent configuration. To keep the server availability status current (reducing the need for your Agent to contact the server to set the server available status), add updateNetworkPropertiesTimeoutinSec to the configuration file for your Agents.
similarProcess Running	Boolean	All	Windows	True when the image name or MD5 hash matches a running process; otherwise False.

Using evtIsAfterOperation

A rule property, `evtIsAfterOperation`, is available for you to use with classification rules. During the rule operation, `evtIsAfterOperation` evaluates to False, so the rule stops evaluating. After the operation is done, `evtIsAfterOperation` evaluates to True. This effectively ensures that the rule runs only “post” operations, that is, after operations such as file copy or move.

Here is an example of the property in a classification rule that classifies the destination file. Classification of the destination file happens because the rule only evaluates to True after the operation, and the target file for classification after the operation is the destination file. The source files of any operation will not be classified:

```
<and>
  <like expr="%\declassified%">
    <evtDestFilePath/>
  </like>
  <equal>
    <evtIsAfterOperation />
    <bool value="true" />
  </equal>
</and>
```

With `evtIsAfterOperation` set to True, DG does not classify source files of operations. It classifies destination files after operations (post operation).

`evtIsAfterOperation` applies to file operations that involve applications, such as copy and save in Microsoft Word or Microsoft Excel. It does not apply to Save As, Print or CD/DVD write operations. This property is available in the Rule Wizard as the **Execute Rule After Operation** option for control rules and app management rules, but not for classification, filter, or trusted process. Best practice suggests that to configure control and app management rules to execute post operation, use **Execute Rule After Operation** rather than adding `evtIsAfterOperation` to your rule XML. To configure classification rules or other rules to run as post operations, add `evtIsAfterOperation` to the rule XML.

Address/URL Rule Properties

Address/URL rule properties govern activities that involve the URL field of a Web browser.

The Digital Guardian Agent updates its Address Bar and URL information whenever a page changes. The following events trigger an update:

- Redirecting a Web page
- Entering an address into the address field and pressing Enter
- Clicking on a link

Address Properties

Address properties govern the string in the Address field of the browser. You could use these properties to delineate between corporate and non-corporate sites.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
curProcess AddressBar	string	All	Windows	The URL displayed in the Address field of the current web browser process (Internet Explorer, Chrome, Firefox, and Edge).

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
parentProcess AddressBar	string	All	Windows	The URL displayed in the Address field of the web browser parent process, if any (Internet Explorer, Chrome, Firefox, and Edge).
prevProcess AddressBar	string	Data Exchange	Windows	<p>The URL displayed in the Address field of the previous web browser process, if any (Internet Explorer, Chrome, Firefox, and Edge).</p> <p>This property is for the address bar of the source browser window for a paste operation. For example, if you use the clipboard to copy text from a browser window into another window (browser or non-browser), prevProcess AddressBar will have the URL that was present in the source browser window's address bar.</p> <p>This property is not available if the source process of a clipboard operation is not a browser.</p>

URL List Properties

URL list properties record all of the URLs used to assemble and display a Web page. When the page changes, the list is discarded and rebuilt for the new page. You could use these properties to determine the origin of a Web page, or to determine if elements of the page are coming from untrusted sources.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
curProcess URLList	string	All	Windows	Deprecated. The list of URLs used to render the current Web page. This property has been deprecated and will be removed in a future release. Rules using this property should be rewritten without it.
parentProcess URLList	string	All	Windows	Deprecated. The list of URLs used to render the Web page of the parent process. This property has been deprecated and will be removed in a future release. Rules using this property should be rewritten without it.

Application Data Exchange Operation Rule Properties

Application data exchange (ADE) rule properties govern activities that involve cutting and pasting, dragging and dropping, or the Windows clipboard. You can regulate user activities in this area using these properties.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtADEType	int32	Data Exchange operations	Linux, macOS	The type of application data exchange activity to regulate. For a list of valid exchange types, refer to “Application Data Exchange Types” on page 190 . If you do not specify a subtype in a rule, the rule applies to all exchange events.
evtBufferEntity Frequency	int32	Data Exchange, Classification	Windows, macOS	Specifies the frequency with which a particular content pattern or query file item appears in the buffer.
evtDestDoc PropertyDate	dateTime	File, Print and Data Exchange events	Windows	A document property in the destination document containing date and time information, such as Creation Date or Last Save Time.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtDestDocPropertyInt	int32	File, print and data exchange events	Windows	A document property in the destination document containing integer information, such as Number of Words or Number of Paragraphs.
evtDestDocPropertyString	string	File, print and data exchange events	Windows	A document property in the destination document containing string information, such as Category or Keywords.
evtDestDriveType	int32	File	Windows, Linux, macOS	The type of drive for the destination file. Note—if the Agent cannot determine the destination drive type (type = None or Unknown), it sets the drive type to Removable. On the Alerts page and exported CSV file, the details for an alert will display True for Removable Drive in this case.
evtDestFileExt	string	File	Windows, Linux, macOS	The extension of the destination file.
evtDestFilePath	string	File	Windows, Linux, macOS	The fully-qualified path to the destination file.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtSrcDoc PropertyDate	dateTime	File	Windows	A document property in the source document containing date and time information, such as Creation Date or Last Save Time.
evtSrcDoc PropertyInt	int	File	Windows	A document property in the source document containing integer information, such as Number of Words or Number of Paragraphs.
evtSrcDoc PropertyString	string	File	Windows	A document property in the source document containing string information, such as Category or Keywords.
evtSrcDriveType	constant	All	Windows, Linux, macOS	The type of drive for the source file.
evtSrcFileCan ContentInspect	Boolean	File,	Windows, Linux, macOS	Whether the source file can be content inspected.
evtSrcFileExt	string	All except network operations	Windows, Linux, macOS	The extension of the source file.
evtSrcFilePath	string	All except network operations	Windows, Linux, macOS	The fully qualified file path of the source file.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtSrcProcess CompanyName	string	Data Exchange operations	Windows, macOS	The company name associated with the process file version resource block of the source process.
evtSrcProcess FileDescription	string	Data Exchange operations	Windows	The file description associated with the process file version resource block of the source process.
evtSrcProcess FilePath	string	Data Exchange operations	Windows, Linux, macOS	The fully qualified file path of the source process.
evtSrcProcess FilePolicyTag	string	Data Exchange operations	Windows	Gets the tag for the process ID of the source process that generated the event.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtSrcProcess FileVersion	string	Data Exchange operations	Windows	<p>The file version associated with the process file version resource block of the source process.</p> <p>Note: FileVersion properties use the string version of the application version. To view this information, view the file properties for the application executable. Under the Version tab, in the Other version information area, select File Version. This string is the version that you should use with your rules.</p>
evtSrcProcess ImageName	string	Data Exchange operations	Windows, Linux, macOS	<p>The image name of the source process.</p> <p>For Linux Agents, this property checks against the executable name.</p>
evtSrcProcess LegalCopyright	string	Data Exchange operations	Windows	<p>The legal copyright associated with the process file version resource block of the source process.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtSrcProcessMD5ContentHash	md5Hash	Data Exchange operations	Windows, Linux, macOS	The GUID representing the MD5 content signature of the source process.
evtSrcProcessProductName	string	Data Exchange operations	Windows, macOS	The product name associated with the process file version resource block of the source process.
evtSrcProcessProductVersion	string	Data Exchange operations	Windows, macOS	The product version associated with the process file version resource block of the source process.
evtSrcProcessWindowTitle	string	Data Exchange operations	Windows, Linux, macOS	The main window title string of the source process application.
evtSrcProcessCmdLine	string	Data Exchange operations	Windows, Linux	The contents of the command line used to start the source process.

Classification Rule Properties

Classification rule properties govern files and content that Digital Guardian has classified using either Adaptive Content Inspection or context-based Adaptive File Classification. Depending on the Digital Guardian features that you have enabled, you can use these properties to regulate user activity based on the classifications that Digital Guardian has applied to your files and content.

For more information about Digital Guardian Adaptive File Classification, refer to *Digital Guardian Management Console User's Guide*.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtBufferEntity Frequency	integer	Data Exchange, Email, and Network Transfer Upload	Windows, macOS	<p>Specifies the frequency with which a particular content pattern or query file item appears in the buffer.</p> <p>Buffer content represents any information not stored in a file. Buffers contain information typed into Web forms or instant messenger clients, or contained in Windows Clipboard content. The DG Agent monitors all buffer content as the user interacts with it, classifying the content as it is entered or edited.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtBufferPolicyTag	string	Data Exchange, Email, and Network Transfer Upload	Windows, Linux, macOS	<p>Identifies the classification tags from the source file associated with the buffer content. Classification tags do not propagate to the buffer content from the source file.</p> <p>Buffer content includes all information not stored in a file. Buffers contain information entered into Web forms, instant messenger clients, or information contained in the Microsoft Windows Clipboard. The DG Agent monitors all buffer content as the user interacts with it, classifying the content as it is entered or edited.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtDestFileCanContentInspect	Boolean	All except Network Operations	Windows, Linux, macOS	<p>True when a content inspection feature can examine and classify the contents of the destination file. False if the DG Agent is unable to examine and classify the contents of the destination file.</p> <p>This property only requires that the Agent be <i>able</i> to read and classify the file. It does not require that the Agent actually classify the file.</p>
evtDestFileEntityFrequency	integer	All except Network Operations	Windows, Linux, macOS	<p>Specifies the frequency with which a particular content pattern or query file item appears in the destination file.</p> <p>For example, you might have a rule that restricts emailing a file with more than ten instances of credit card information.</p> <p>This rule property requires the Digital Guardian Adaptive Content Inspection add-on module.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtDestFileIs Classified	Boolean	All except Network Operations	Windows, Linux, macOS	True when the destination file is classified; otherwise False.
evtDestFilePartner-PolicyTag	string	All except Network Operations	Windows	Identifies the classification name or names associated with a destination file and provided by a DG partner application. You must use the “id” attribute with this property; otherwise, the rule will not compile. User Classification and User Classification Integration rules require a tag id value of 12.
evtDestFile PolicyTag	string	All except Network Operations	Windows, Linux, macOS	Identifies the classification name or names associated with the destination file.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtSrcFileCanContentInspect	Boolean	All except Network Operations	Windows, Linux, macOS	<p>True when a content inspection feature can examine and classify the contents of the source file. False if the DG Agent is unable to examine and classify the contents of the source file.</p> <p>This property only requires that the Agent be <i>able</i> to read and classify the file. It does not require that the Agent actually classify the file.</p>
evtSrcFileEntityFrequency	integer	All except Network Operations	All	<p>Specifies the frequency with which a particular content pattern or query file item appears in the source file.</p> <p>For example, you might have a rule that prevents users from moving or copying files with more than ten instances of credit card numbers.</p> <p>This rule property requires the Digital Guardian Adaptive Content Inspection add-on module.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtSrcFileIsClassified	Boolean	All except Network Operations	Windows, Linux, macOS	True when the source file is classified; otherwise False.
evtSrcFilePartnerPolicyTag	string	All except Network Operations	Windows	Identifies the classification name or names associated with a source file and provided by a DG partner application. You must use the “id” attribute with this property; otherwise, the rule will not compile. User Classification and User Classification Integration rules require a tag id value of 12.
evtSrcFilePolicyTag	string	All except Network Operations	Windows, Linux, macOS	Identifies the classification name or names associated with the source file.

Using the <NOT> Operator With Policy Tag Classification Rules

Digital Guardian does not support the use of the <not> logical operator with the following rule properties:

- evtBufferPolicyTag
- evtDestFilePolicyTag
- evtSrcFilePolicyTag

Referring to Content Patterns in Classification Rules

When referring to a content pattern from within a classification rule or control rule, always refer to that content pattern in lowercase text. For example, if you have a content pattern called “Top Secret”, refer to it as “top secret” within your rule xml.

Example

```
<evtSrcFileEntityTypeFrequency name="top secret"/>
<int value="0"/>
```

Data Vault Rule Properties

Data vault rule properties govern activities that occur once a data vault is in effect. These properties allow you to fine tune the effect of the data vault based on the trigger rule and the process that is currently vaulted. For more information about data vaulting, refer to [“Data Vault Rules” on page 419](#).

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
activeVaultRuleNames	string	All	Windows, macOS	The names of data vault trigger rules that are currently active. You could use this property to apply additional security if any of a specific list of data vault trigger rules have been activated in any process on the Agent computer.
anyProcessVaulted	Boolean	All	Windows, Linux, macOS	True when any process is currently data vaulted; otherwise False.
clipboardVaulted	Boolean	All	Windows	True when the application source of the clipboard content is vaulted; False otherwise.
clipboardVaultRuleName	string	All	Windows	The name of the data vault trigger rule that vaulted the application source of the clipboard content.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
curProcessSaveAsActive	Boolean	All	Windows	<p>True when the current process has a Save As dialog box open, otherwise False.</p> <p>When used in a rule, this property allows data vaulted applications to perform internal file writes except when a File Save As dialog is open. Generally, you should use this property in data vault rules governing File Write events.</p>
curProcessVaulted	Boolean	All	Windows, Linux	True when the current process has activated a Data Vault trigger rule, otherwise False.
curProcessVaultRuleName	string	All	Windows, Linux	The name of the Data Vault rule that created the data vault applied to the current process.
parentProcessFilePolicyTag	wstring	All	Windows	Gets the tag for the ID of the parent process of the current process that generated the event.
parentProcessVaulted	Boolean	All	Windows, Linux	True when the parent process is data vaulted; otherwise False.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
parentProcess VaultRuleName	string	All	Windows, Linux	The Data Vault rule name associated with the process file version resource block of the parent process.
prevProcessVaulted	Boolean	Data Exchange	Windows	True when the source of the data exchange event has activated a Data Vault trigger rule, otherwise False.
prevProcess VaultRuleName	string	All	Windows	The Data Vault rule name associated with the process acting as the source of the application data exchange.
similarProcess Running	Boolean	All	Windows	True when the image name or MD5 hash of the executable matches a running process; otherwise False.
similarProcess Vaulted	Boolean	All data vaulted processes	Windows	True when the image name or the MD5 hash of the executable file's contents matches a data vaulted process; otherwise False.

Device Rule Properties

Device rule properties govern activities that involve USB devices connected to the DG Agent computer. You can regulate USB and other devices based on the device serial number, vendor, or other properties. You can view device property information for these rules by attaching a device to a computer and viewing its properties from Windows Explorer.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtDestCustomId	string	All except Network Operations and Email Operations	Windows, Linux	The custom ID of the destination USB device. For information about configuring custom IDs for your USB devices, contact your Digital Guardian account representative.
evtDestDeviceClass	string	constOpDeviceOpen	Windows, Linux, macOS	(For future use) Specifies the type of destination device.
evtDestDeviceGuid	GUID	All except Network Operations and Email Operations	Windows	GUID assigned to the destination device.
evtDestFriendlyName	string	All except Network Operations and Email Operations	Windows, Linux, macOS	(For future use) Name assigned by the device vendor to the destination device. You can see the friendly name in the device properties in Microsoft Windows device manager.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtDestProduct	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The product name of the destination USB device. A typical product name might be Traveldrive or Cruzer.
evtDestProductId	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The four character product ID assigned to the destination USB device by the USB Implementers Forum (www.usb.org).

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtDestRemoval Policy	integer	All except Network Operations and Email Operations	Windows, Linux, macOS	<p>How the destination device is expected to be removed from the machine. Combines with the Storage Bus Type field. Categories are:</p> <ul style="list-style-type: none"> • Expect No Removal—device is not removable (the hardware, not the media) • Expect Orderly Removal—users use the Remove Hardware Safely option in Windows • Expect Surprise Removal—users may exercise hot removal of the device (applies to USB devices often)
evtDestRevision	string	All except Network Operations and Email Operations	Windows	The revision of the destination device.
evtDestSerial Number	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The manufacturer's serial number assigned to the destination USB device.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtDestVendor	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The name of the vendor that manufactured the destination device. A typical vendor name might be SanDisk or Memorex.
evtDestVendorId	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The vendor ID assigned to the destination device. USB vendor IDs are assigned by the USB Implementers Forum (www.usb.org).
evtSrcCustomId	string	All except Network Operations and Email Operations	Windows, Linux	The custom ID of the source device. For information about configuring custom IDs for your devices, contact your Digital Guardian account representative.
evtSrcDeviceClass	string	constOp DeviceOpen	Windows, Linux, macOS	Specifies the type of source device. The supported string is 'image' to specify Web cameras that use the USB bus. Most external and internal cameras use the USB bus.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtSrcFriendly Name	string	All except Network Operations and Email Operations	Windows, Linux, macOS	Name assigned by the device vendor to the source device. You can see the friendly name in the device properties in Microsoft Windows device manager.
evtSrcDeviceGuid	GUID	All except Network Operations and Email Operations	Windows	GUID assigned to the source device.
evtSrcProduct	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The product name of the source USB device. A typical product name might be Travel-drive or Cruzer.
evtSrcProductId	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The product ID assigned to the destination device. USB product IDs are assigned by the USB Implementers Forum (www.usb.org).

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtSrcRemoval Policy	integer	All except Network Operations and Email Operations	Windows, Linux, macOS	<p>How the source device is expected to be removed from the machine. Combines with the Storage Bus Type field. Categories are:</p> <ul style="list-style-type: none">• Expect No Removal—device is not removable (the hardware, not the media)• Expect Orderly Removal—users use the Remove Hardware Safely option in Windows• Expect Surprise Removal—users may exercise hot removal of the device (applies to USB devices often)
evtSrcRevision	string	All except Network Operations and Email Operations	Windows	The revision of the source device.
evtSrcSerialNumber	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The manufacturer's serial number assigned to the source USB device.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtSrcVendor	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The name of the vendor that manufactured the source USB device. A typical vendor name might be SanDisk or Memorex.
evtSrcVendorId	string	All except Network Operations and Email Operations	Windows, Linux, macOS	The vendor ID assigned to the destination device. USB vendor IDs are assigned by the USB Implementers Forum (www.usb.org).

Device Event Properties

Device event properties allow Digital Guardian to log information about USB devices, such as memory sticks and flash drives, plugged into users's systems. In addition to USB devices, device event properties work with other devices, such as SATA drives or FireWire devices. With device event symbolic constants in rules, you can direct the Agent to do the following operations:

- Report information collected about devices. Device events collect a subset of the information that generally available for devices.
- Recognize when a device exists at boot time, as well as when a device is inserted or removed at a later time. DG can report the following device-related events to the server. The Rule Builder Wizard includes the events on the Choose Operation Types page in the wizard:
 - Device Added — Sends an event when users add devices to their computers. The equivalent symbolic constant is `constOpDeviceAdded`.
 - Device Detected — Sends an event when the Agent detects a device on a user's computer at start time. The equivalent symbolic constant is `constOpDeviceDetected`.
 - Device Missing — Sends an event when a device the Agent detected on a previous computer start is no longer available or attached to the computer. The equivalent symbolic constant is `constOpDeviceMissing`.
 - Device Open — Sends an event when the Agent detects a user opening a USB device. Use this event to control access to and use of Web cameras based on the process that opens the device, such as Video for Windows, Microsoft DirectShow or Skype®. The symbolic constant to use in a rule is `constOpDeviceOpen`.
 - Device Removed — Sends an event when users remove devices from their computers. The equivalent symbolic constant is `constOpDeviceRemoved`.

To use device events in a rule, select **Device Related** on the Choose Operation Categories page (Step 1 of 5) in the Rule Builder Wizard. When you write rules, use the `evtOperationType` as shown in the example rule below that triggers on the Device Added or Device Detected operations:


```
<i n>
  <evtOperationType />
  <list>
    <constOpDeviceAdded />
    <constOpDeviceDetected />
  </list>
</i n>
```

Device Event Symbolic Constants

Device Event Symbolic Constant	Reported Event
constOpDeviceAdded	User inserted a USB device into a port on the computer.
constOpDeviceDetected	At computer start, the Agent detected a new USB device.
constOpDeviceMissing	At computer start, a device that the Agent detected at shut down is no longer detected.
constOpDeviceOpen	User or application process is opening a USB device for use. Use this constant primarily for detecting when the user or application, such as Skype, opens a Web camera.
constOpDeviceRemoved	User removed a USB device from the computer.

Best practices suggest that you log all device-related events. To do this, select the Device Added, Device Detected, Device Missing, Device Open and Device Removed options under **User Actions to Log** on the Data tab in the Core Settings Resource for your workstations.

The device ID is added to file-related events to enable you to track device usage. The device data source enables reporting on devices that exist in your enterprise. One report under Custom Reports uses device data—Device - All Known Devices—that provides full details about each known device

This report returns the IDs of all devices that generated events, with detailed information about each device.

You can write rules to alert on the Device Added, Device Opened or Device Detected operations and you can write rules based on standard file control rules that take advantage of the device information. The Device Detected and Device Missing operations do not create alerts and you cannot use them to block operations.

You cannot write rules to alert on or block when users remove USB devices (device removed event). Use `constOpDeviceRemoved` only to send events to the DGMC. The Static Rule Analyzer prevents you from saving such a device removed rule by telling you that `constOpDeviceRemoved` cannot be used in Alert or Block rules.

You cannot use `constOpDeviceAdded`, `constOpDeviceDetected`, `constOpDeviceMissing` or `constOpDeviceRemoved` in Block rules.

The following device-specific fields enable you to report on events that involve the devices:

Field Name	Description
Custom ID	Reports the identification value assigned to the device, if any.
Device Class	Reports the class of the device.
Device ID	Reports the ID assigned to the device by the manufacturer.
Drive Type	Reports the type of drive for the device, such as USB.
Friendly Name	Reports the common name for the device.
Product ID	Identification number of the product assigned by the manufacturer. Compare to serial number.

Field Name	Description
Product Name	Manufacturer's name for the product.
Removal Policy	<p>Specifies how the device is expected to be removed from the machine. Combines with the Storage Bus Type field. Categories are:</p> <ul style="list-style-type: none"> • Expect No Removal—device is not removable (the hardware, not the media) • Expect Orderly Removal—users use the Remove Hardware Safely option in Windows • Expect Surprise Removal—users may exercise hot removal of the device (applies to USB devices often)
Serial Number	Reports the device serial number.
Storage Bus Type	Reports the type of bus the device uses, such as USB or SCSI.
Supports Predict Failure	Identifies whether the device supports failure prediction. Either Yes or No.
Vendor	Reports the name of the device manufacturer
Vendor ID	Reports the ID of the manufacturer.

An important point is that the device referred to might be the hardware, not the medium. For example, when users attach an SD card reader to the computer, the card reader is the device, not the SD card inserted or removed from the reader. Events related to the device might return the hardware information rather than the medium information.

Removal policy is also related to whether the event is connected to the hardware or the storage medium.

Removal policy for a device refers to the hardware, not the storage medium. As one example, DG treats DVD or CD-ROM drives as non-removable. So the removal category is Expect No Removal in events related to CD/DVD drives, even though users can remove the disks (CD or DVD) from the drive. The medium is removable, the device is not.

Control Web Camera Devices With Rules

You can use rules to monitor and control Web cameras that use USB connections. Most external cameras and many internal cameras, such as those provided on laptops, use USB connections so you can monitor and control those camera video streams with rules.

To monitor and control cameras, complete these two tasks:

- A. Enable camera control by adding a required configuration element—`<deviceOpenEventMonitorIds>`—to the Agent configuration file on computers.
- B. Create and deploy rules that use the `constOpDeviceOpen` constant and the `evtSrcDeviceClass` rule property to monitor and control user cameras.

Enabling Camera Control

To enable the ability to monitor cameras, you add a configuration element to the file `config.xml` on any Agent where you plan to monitor camera use. The element you add is:

```
<deviceOpenEventMonitorIds>{65E8773D-8F56-11D0-A3B9-00A0C9223196}  
</deviceOpenEventMonitorIds>
```

where the value `{65E8773D-8F56-11D0-A3B9-00A0C9223196}` is required.

Creating Rules To Control Cameras

After you add this element to your Agent configuration files and deploy the files to your Agents, you can deploy rules that monitor cameras and those rules will run successfully on the Agent. Without the configuration element, the Agent will not honor rule-driven camera controls.

The following block rule example prevents camera video access when users start Skype:

```
<and>
  <equal >
    <curProcessImageName />
    <string value="skype.exe" />
  </equal >
  <equal >
    <evtSrcDeviceClass />
    <string value="image" />
  </equal >
  <equal >
    <evtOperationType />
    <constOpDeviceOpen />
  </equal >
</and>
```

The symbolic constant `constOpDeviceOpen` matches events generated by users when they open cameras or when applications like Skype (in this example) open cameras. `evtSrcDeviceClass` with the string value "image" defines the device of interest as a video camera. So this rule evaluates to true when the application Skype tries to use the camera. Defining this rule as a block rule prevents Skype users from seeing the video stream.

Note: When you block the video stream from a camera, the audio stream remains available to the users.

Document Properties Rule Properties

Document properties rule properties govern events based on the document properties associated with a file. Rules using document properties can govern events based on the values contained in the properties of a file.

File properties can be assigned by software applications and by the Microsoft Windows operating system.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtDestDocPropertyDate	dateTime	File, Print and Data Exchange events	Windows	A document property in the destination document containing date and time information, such as Creation Date or Last Save Time.
evtDestDocPropertyInt	integer	File, Print and Data Exchange events	Windows	A document property in the destination document containing integer information, such as Number of Words or Number of Paragraphs.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtDestDocPropertyString	string	File, Print and Data Exchange events	Windows	A document property in the destination document containing string information, such as Category or Keywords.
evtSrcDocPropertyDate	dateTime	File, Print and Data Exchange events	Windows	A document property in the source document containing date and time information, such as Creation Date or Last Save Time.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtSrcDocPropertyInt	integer	File, Print and Data Exchange events	Windows	A document property in the source document containing integer information, such as Number of Words or Number of Paragraphs.
evtSrcDocPropertyString	string	File, Print and Data Exchange events	Windows	A document property in the source document containing string information such as Category or Keywords.

Typical document property values might look like the following examples:

Example 1

This example shows how the property evtSrcDocPropertyDate can match the “creation date” property to a specific date and time.


```
<equal >  
<evtSrcDocPropertyDate name="creation date" />  
<dateTime value="04/21/2004 10:33 AM" />  
</equal >
```

Example 2

This example shows how the property `evtSrcDocPropertyString` can match the “author” property to any string containing the value “batista”.

```
<like expr="%batista%">  
<evtDestDocPropertyString name="author" />  
</like>
```

Digital Guardian does not validate rule property names. If a rule contains a property name that does not exist, Digital Guardian ignores that property. Be sure to test your rules and properties to be sure that they are working properly.

Microsoft Windows Explorer Document Properties

Microsoft Windows Explorer properties are created and applied by the Microsoft Windows operating system. These properties typically overlap with the Microsoft Office document properties.

To view the Microsoft Windows Explorer properties for a file, open Explorer and navigate to the file to view. Right-click the file and select Properties. The Properties dialog box opens. The available properties vary from file type to file type.

Default Microsoft Office Document Properties

Microsoft Office documents contain a number of default properties. Depending on the Office application, the available properties can vary. For example, the Number of Hidden Slides property is available only for Microsoft PowerPoint files. The following properties are present Across all Microsoft Office applications.

Property	Value Type
Application Name	string
Author	string
Byte Count	integer
Category	string
Character Count	integer
Character Count (with spaces)	integer
Comments	string
Company	string
Creation Date	dateTime
Format	string
Hidden Slides	integer
Keywords	string
Last Author	string
Last Print Date	dateTime
Last Save Time	dateTime
Line Count	integer
LinksDirty	integer
Manager	string
Multimedia Clips	integer
Notes	integer

Property	Value Type
Pages	integer
Paragraph Count	integer
Revision Number	string
Scale	integer
Security	integer
Slides	integer
Subject	string
Template	string
Title	string
Total Editing Time	dateTime
Word Count	integer

To view the Microsoft Office document properties for a file, open the file in the appropriate Office application. From the File menu, select Properties. The Properties dialog box opens. The available properties vary from file type to file type.

In some cases, the property name differs across Office applications. Digital Guardian uses the above property names for all Office applications. If the property name displayed in the Windows Properties dialog box differs from the name listed on the previous pages, use the property name listed in this document in your rules.

Note: Digital Guardian does not validate document property names. Always verify that you are using the correct document property name.

Custom Document Properties

You can create custom document properties for Office documents. For information about creating and editing these properties, view Microsoft help for the product. You can also create custom document properties for PDF documents. You typically do this using Adobe Acrobat.

Digital Guardian can enforce rules and record events based on custom document properties, as described in [“Using Document Properties in Rules” on page 149](#). If you have a document where a custom property shares the same name as a default property, Digital Guardian uses the value of the default document property and ignores the value of the custom property.

Enabling Document Property Rules

To use document properties in rules you need to enable them first. You can do this by editing the Agent configuration or by editing a Core Settings resource, as follows:

1. Hover your cursor over System and, under Settings, select **Resources**. The Process Flags page opens.
2. Select Core Settings in the left pane. The Core Settings page opens.
3. Click the edit icon on the row for the type of Agent to modify. For example, click the edit icon on the Windows Workstation Default row to configure the default Windows Agent settings. The appropriate page opens.
4. Select the **Data** tab. The tab page opens.
5. Click **Edit**. The page becomes editable.
6. Click **Advanced Settings**. The Advanced Data Collection Settings dialog box opens.
7. For **Enable Document Property Rules**, select **Yes**.

8. Click **Save Changes**. Any new Agents of this type that you install will have the document property rules enabled.

Using Document Properties in Rules

The Digital Guardian Agent for Windows can read document properties in Microsoft Office documents and PDF documents. You can therefore write control rules that directly target Office and PDF document properties. For example, to prevent egress of a PDF document that has a custom “Keywords” document property set to “Confidential,” you can write a control rule that monitors or blocks actions based on that property.

For Office documents, the ability to use control rules applies to Word, Excel, and PowerPoint files. Control rules will apply to:

- Saved documents that contain document properties.
- New, unsaved documents that contain document properties. In this case, you can use control rules to control Save As and Print events.
- Existing documents that contain document properties, where a user attempts to overwrite the existing file. In this case, you can use control rules to control Save As events.
- Existing documents that contain document properties, if the user changes the document properties before re-saving the file.

Prerequisites

Before using document properties in rules:

- Enable document property rules, as described in [“Enabling Document Property Rules” on page 148](#).
- Verify that the following settings are enabled (1) in the Agent configuration file (config.xml)

```
<docpropsProcessDestFileForContent>1</docpropsProcessDestFileForContent>
```

```
<docpropsProcessSrcFileForContent>1</docpropsProcessSrcFileForContent>
```

If you need to enable these settings, you must reboot the Agent computer for the new settings to take effect.

- If you want to control PDF documents based on custom document properties, you must install a subset of the ACI feature package. When you perform a default Agent installation, the full ACI feature package is installed automatically. If, however, you choose to exclude the full ACI package from the installation using DG MSI Tool, you must install a subset of the ACI package. For details, refer to “Installing DG Agents and Features” in the *Digital Guardian Installation and Upgrade Guide*.

Creating Rules That Target Document Properties

To create rules targeting document properties, use a rule property such as `evtSrcDocPropertyString`. When a file with document properties is examined by the rule engine, the Agent copies the document properties to an alternate data stream (ADS) when a process accesses the file. The rule cannot be a component rule. The ADS is a hidden file that resides on disk next to the original file. The ADS contains metadata about the original file, such as the document properties read by the Agent. The Agent reads the metadata to apply the rule.

You can use the DG Content Inspection Application utility (DGCIApp) to view document properties metadata and determine whether the Agent is obtaining the correct document properties values. For details on using DGCIApp, refer to “DG Content Inspection Application Utility” in the *Digital Guardian Utilities Guide*. One process that commonly causes stream generation is `explorer.exe`. Processes flagged with ND (No Document Properties) or SK (Skipped) process flags do not cause stream generation.

The following example shows a control rule that creates a DG tag using the `evtSrcDocPropertyString` rule property. As shown, if either the source file or destination file has the “Keywords” document property set to “Confidential,” the rule prevents sending the file by email, uploading it, and printing it.

```

<and>
  <or>
    <equal >
      <evtSrcDocPropertyString name="Keywords" />
      <string value="Confidential" />
    </equal >
    <equal >
      <evtDestDocPropertyString name="Keywords" />
      <string value="Confidential" />
    </equal >
  </or>
  <i n>
    <evtOperationType/>
    <list>
      <constOpSendMail />
      <constOpNetTransferUpload/>
      <constOpPrint/>
    </list>
  </i n>
</and>

```

Control rules read the document properties through rule properties such as `evtSrcDocPropertyString`. If you want the document property value to be reported on the DGMC, write one control rule per document property value (such as Confidential or Restricted). The rule name in the generated alert will indicate what the document property value was. Alternatively, use one control rule and send document property data as Custom Rule data.

Viewing Document Properties on Network Shares

To see Document Properties on network shares, the Agent requires Classify Files on Network Shares to be enabled. To enable this setting:

1. Hover your cursor over System and, under Settings, select **Resources**.
2. In the left pane, select Core Settings.
3. Click the edit (pencil) icon for the resource you want to configure.
4. Click the **Data** tab and then click **Edit**. Set Classify Files On Network Shares to **Yes**. Click **Save Changes**.

Email Operation Rule Properties

Email operation rule properties govern email messages sent by your users. Email operation rule properties can govern activities based on recipient addresses and recipient types.

For emails sent to local mailing lists, Digital Guardian examines the group address and then expands the group and examines the addresses of the recipients in the group. Users cannot circumvent Digital Guardian rules by adding recipients to a local mailing list.

Message Recipients and Rule Evaluation

Rules evaluation is tested in the following order:

1. Recipients only — Digital Guardian evaluates persons, aliases and groups for rules with a rule action type of Block. The block can originate from the rule action type or from a Block prompt. Any blocked recipient prevents the email from being sent and Digital Guardian does not evaluate the email message further.

Digital Guardian does not expand groups and does not evaluate attachments at this stage.

2. Expanded groups — Digital Guardian evaluates the expanded recipients of any groups. Any blocked expanded recipient in a group prevents the email from being sent.
3. Email attachments and body content — Digital Guardian evaluates any email attachments and the body of the email for all rule actions.

Consider the order of recipient and rule evaluation when you plan your rules.

Note: For information about formatting email addresses for use with Digital Guardian rules, refer to [“Email Address Format” on page 187](#).

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtMailAttachmentSize	integer	Email Operations only	Windows, Linux	Specifies the size, in bytes, of the any email attachment. You could use this property with a greater than relational operator to govern the maximum size of attachments.
evtMailRecipients	string	Email Operations only	Windows, Linux, macOS	The actual addresses of recipients of the email. This property applies to all recipients of the message, regardless of recipient type.
evtMailRecipientTypes	constant	Email Operations only	Windows, Linux, macOS	The type of the email recipient. This property indicates whether the message recipient was listed on the To, CC, or BCC line. For a list of valid recipient types, refer to “Email Recipient Types” on page 195 .
evtMailSMIMEEncrypted	Boolean	Email Operations only	Windows	Reports whether Outlook will send the email encrypted. Reported before Outlook sends the message.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtMailSMIME Signed	Boolean	Email Operations only	Windows	Reports whether Outlook will send the email signed. Reported before Outlook sends the message.
evtMailSubject	string	Email Operations only	Windows, Linux	Inspects the content of the subject line in an email. This allows you to compare the contents of the subject line against a string, for example.
evtMailTotalSize	integer	Email Operations only	Windows	Specifies the combined total size, in bytes, of the body of the email, all attachments, and all OLE embedded objects. You could use this property to specify the maximum size of any email, regardless of the size and number of attachments.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtOriginalMailRecipients	string	Email Operations only	Windows, Linux, macOS	The email addresses of the original recipients of the email. This property applies to all recipients of the message, regardless of recipient type. Use this property to write rules against email groups.
evtOriginalMailRecipientTypes	constant	Email Operations only	Windows, Linux, macOS	The type of the original email recipient. This property indicates whether the recipient was listed on the To, CC, or BCC line. For a list of valid recipient types, refer to “Email Recipient Types” on page 195. Use this property to write rules against email groups.

DLL Load (Image Load) Rule Properties

When you use the symbolic constant `constOpImageLoad` as the type in `evtOperationType` to monitor loading of DLL images, DG provides the following properties to use to specify information about the loaded dynamic link library (DLL).

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
<code>dllAccessTime</code>	<code>datetime</code>	Image Load	Windows	The last time something accessed the DLL.
<code>dllCompanyName</code>	<code>string</code>	Image Load	Windows	The name of the company that created the DLL.
<code>dllCreatedTime</code>	<code>int64</code>	Image Load	Windows	The create time of the DLL on the computer.
<code>dllFileAttributes</code>	<code>string</code>	Image Load	Windows	The file attributes on the DLL file.
<code>dllFileDescription</code>	<code>string</code>	Image Load	Windows	The version string of the DLL that holds the file description.
<code>dllFilePath</code>	<code>string</code>	Image Load	Windows	The full path with the name of the DLL file.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
dllFileVersion	string	Image Load	Windows	The version string of the DLL that holds the DLL file version. To obtain the version number of the DLL file, use a resource editor such as Resource Tuner 2 that displays a String File Info block, and use the File Version shown in that block. Do not use Windows Explorer to obtain the version number. Explorer displays binary version information, which, if used in a rule, will cause the rule to fail.
dllImageBase	int64	Image Load	Windows	The location the DLL is loaded into in memory.
dllImageName	string	Image Load	Windows	The name of the DLL file.
dllImageSize	int64	Image Load	Windows	The size, in bytes, of the DLL loaded in memory.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
dllLegalCopyright	string	Image Load	Windows	The version string of the DLL that holds the legal copyright for the DLL.
dllLoadedByAppInit	string	Image Load	Windows	The DLL was loaded by application initialization.
dllLoadedByReflection	string	Image Load	Windows	The DLL was loaded by reflection into memory. Loading by reflection results in a library loading itself. The loaded library is then largely invisible to the system and processes. The Agent can monitor such operations and report them.
dllLoadedUnder Impersonation	Boolean	Image Load	Windows	DLL was loaded under impersonation to bypass process-level restrictions.
dllLoadedUsing RemoteThread	Boolean	Image Load	Windows	DLL was loaded by a remote thread.
dllMD5ContentHash	GUID	Image Load	Windows	The 16-byte MD5 hash of the DLL.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
dllModifyTime	int64	Image Load	Windows	The last time that something modified the DLL.
dllProductName	string	Image Load	Windows	The version string in the DLL that holds the product name associated with the DLL.
dllProductVersion	string	Image Load	Windows	The version string for the DLL that holds the description for the product.
dllSHA1ContentHash	sha1	Image Load	Windows	The 20-byte SH1 hash of the DLL.
dllSHA256ContentHash	sha256	Image Load	Windows	The 32-byte SHA256 hash of the DLL.
dllWriteTime	int64	Image Load	Windows	The last time something wrote to the DLL file.

Network File Transfer Operation Rule Properties

Network file transfer operation rule properties govern activities that involve uploading and downloading files across a network, similar to network operations. Network file transfer operation rule properties can govern activities based on drive type, transport protocols, ports, IP addresses, file extensions and domains.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtDomain	string	Network Operations only	All	The domain (host) name of the request.
evtIsOutboundConn	Boolean	Network Operations only	Windows, Linux, macOS	True when the Agent computer initiated the connection, otherwise False.
evtLocalPort	integer	Network Operations only	Windows, Linux, macOS	The port used by the local computer involved in the operation. This value can be from a list of predefined constants, or a user-defined port value between 1–65535.
evtProtocolType	constant	Network Operations only	Windows, Linux, macOS	The protocol used by the operation (TCP, UDP, Bluetooth, IrDA). For a list of valid protocol types, refer to “Protocol Types” on page 211 .

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtRemoteAddress	ipAddress bluetooth Address	Network Operations only	Windows, Linux, macOS	The IP address of the remote computer involved in the operation.
evtRemotePort	integer	Network Operations only	Windows, Linux, macOS	The port used by the remote computer involved in the operation. This value can be from a list of predefined constants, or a user-defined port value from 1–65535.
evtSrcFileExt	string	All except network operations	Windows, Linux, macOS	The extension of the source file.
evtSrcFilePath	string	All except network operations	Windows, Linux, macOS	The fully qualified file path of the source file.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtUrlPath	string	Network Operations only (not NTD or NTU operations)	Windows, Linux, macOS	<p>The URL used for the network operation.</p> <p>To prevent false positives with this property, use it in conjunction with the property evtProtocolType and the constant constProtocol-HTTP.</p> <p>On Microsoft Windows platforms, this property supports Microsoft Internet Explorer 11 and later. On Mozilla Firefox, this property supports the two most recent ESR and regular browser versions. On Linux, this property is browser-independent.</p>
evtUserDomain	string	Network Operations		The domain of the user involved in the network operation.

Using Source and Destination File Paths in Network Transfer Operations

DG cannot always determine a source path or file name that makes sense for download operations from any given Web database or interface. In those case, files might be represented by numbers or codes or some other form. The Agent can get the source path and file name for downloads from Microsoft SharePoint sites, but in cases like the viewer in Google Gmail, the files do not have a source path and name other than the domain name. Only certain sites and download mechanisms will resolve a real source file name and path. In the case of SharePoint, the Agent parses the SharePoint API to get the file name. As a result, source file paths and file names might be reported incorrectly in reports in the DGMC.

Similarly, the Agent cannot always determine a destination path or file name that makes sense for upload operations to any given Web database or interface. In those case, files might be represented by numbers or codes or some other form. The Agent can get the destination path and file name for uploads to Microsoft SharePoint sites, but in cases like the viewer in Google Gmail, the files do not have a destination path and name other than the domain name. Again, reports in the DGMC might report the destination file name and path incorrectly for upload operations to Web sites or databases.

Network Operation Rule Properties

Network operation rule properties govern activities that involve moving files across a network. Network operation rule properties can govern activities based on transport protocols, ports, and domains.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtAdapterName	string (an array if more than one adapter)	Network Operations only	Windows, macOS	<p>The name of the network adapter in use for the operation. Adapter names match the network interfaces returned by the <code>ipconfig</code> or <code>ifconfig</code> command in a terminal window or Command Prompt. Only adapters with IP4V addresses appear in this property.</p> <p>Not supported on Microsoft Windows Server 2012 R2. Rules that use this with wireless devices will not set the wireless flag. Events from these rules will not be marked as Wireless.</p>
evtDomain	string	Network Operations only	All	The domain (host) name of the request.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtIsOutboundConn	Boolean	Network Operations only	Windows, Linux, macOS	True when the Agent computer initiated the connection, otherwise False.
evtIsPrivate	Boolean	Network Operations Only	Windows, Linux, macOS	True when the connection is over one of the following IP address ranges: 10.0.0.0/8 127.0.0.0/8 172.16.0.0/12 192.168.0.0/16 168.254.0.0/16 Otherwise False.
evtLocalPort	integer	Network Operations only	Windows, Linux, macOS	The port used by the local computer involved in the operation. This value can be from a list of predefined constants, or a user-defined port value between 1–65535.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtPhysicalMedium	constant	Network Operations only	Windows	<p>The type of network adapter used by the operation. For a list of valid device types, refer to “Network Adapter Types” on page 198.</p> <p>Not supported on Microsoft Windows Server 2012 R2.</p> <p>Rules that use this with wireless devices will not set the wireless flag. Events from these rules will not be marked as Wireless.</p>
evtProtocolType	constant	Network Operations only	Windows, Linux, macOS	<p>The protocol used by the operation (TCP, UDP, Bluetooth, IrDA). For a list of valid protocol types, refer to “Protocol Types” on page 211.</p>
evtRemoteAddress	ipAddress bluetooth Address	Network Operations only	Windows, Linux, macOS	<p>The IP address of the remote computer involved in the operation.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtRemotePort	integer	Network Operations only	Windows, Linux, macOS	The port used by the remote computer involved in the operation. This value can be from a list of predefined constants, or a user-defined port value from 1–65535.
evtRepeatCount	integer	Network Operations only	Windows, Linux, macOS	The count of new Agent-reported events that match previously reported events, aggregated as one value.

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtUrlPath	string	Network Operations only	Windows, Linux, macOS	<p>The URL used for the network operation.</p> <p>To prevent false positives with this property, use it in conjunction with the property evtProtocolType and the constant constProtocol-HTTP.</p> <p>On Microsoft Windows platforms, this property supports Microsoft Internet Explorer 11 and later. On Mozilla Firefox, this property supports the two most recent ESR and regular browser versions.. On Linux, this property is browser-independent.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtWirelessAccessPoint	macAddress	Network Operations only	Windows	<p>The MAC address of the wireless router that allows the computer to connect to a wired network. In the case of an ad hoc network, this is the MAC address of the machine hosting the wireless network. Related to the property evtWirelessSSID.</p> <p>Not supported on Microsoft Windows Server 2012 R2.</p> <p>Rules that use this with wireless devices will not set the wireless flag. Events from these rules will not be marked as Wireless.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtWirelessAuthenticationMode	constant	Network Operations only	Windows	<p>Describes the types of wireless authentication to regulate in your rule. For example, you can write rules that prevent your wireless cards from using insecure authentication types.</p> <p>Not supported on Microsoft Windows Server 2012 R2. Rules that use this with wireless devices will not set the wireless flag. Events from these rules will not be marked as wireless.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtWireless Encryption	constant	Network Operations only	Windows	<p>Describes the types of encryption used to secure your wireless communications. For example, you might write rules that require specific encryption.</p> <p>Not supported on Microsoft Windows Server 2012 R2.</p> <p>Rules that use this with wireless devices will not set the wireless flag. Events from these rules will not be marked as wireless.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtWirelessInfrastructureMode	constant	Network Operations only	Windows	<p>Describes the types of wireless infrastructure to regulate in your rule. For example, you might write rules that prevent your users from forming ad hoc wireless groups.</p> <p>Not supported on Microsoft Windows Server 2012 R2. Rules that use this with wireless devices will not set the wireless flag. Events from these rules will not be marked as Wireless.</p>

Property Name	Data Type	Application Operation Type	Valid Operating Systems	Description
evtWirelessSSID	string	Network Operations only	Windows, Linux	<p>The name of the wireless network the computer is connected to, such as "corporateguest." Affects only the property evtWirelessAccessPoint.</p> <p>Not supported on Microsoft Windows Server 2012 R2.</p> <p>Rules that use this with wireless devices will not set the wireless flag. Events from these rules will not be marked as Wireless.</p>

Note: Digital Guardian Agent rules do not prevent Agent computers from establishing controlled wired or wireless connections. Digital Guardian does act on traffic over the controlled connections. For example, if you have an active rule to block WEP encrypted wireless connections applied to laptop computers and the user establishes a wireless connection to a WEP encrypted network, with no other active network connections, this is not blocked. If the user then tries to browse the Web using the Microsoft Internet Explorer browser, connect to an FTP site, or connect to another network, those actions are blocked.

OLE Insertion Operation Rule Properties

OLE operation rule properties govern activities that involve inserting one file into another using Microsoft Object Linking and Embedding (OLE). You can regulate user activities in this area using these properties.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtObjectType	string	OLE file insertions using the constO-pAdeInsert-NewObject event	Windows	The name of the object type being inserted. You can govern any object type listed in the Windows Insert Object dialog box. In most Office applications, you reach this dialog box from the Insert > Object menu.

Print Operation Rule Properties

Print operation rule properties govern activities that involve printing files to a local or network printer. You can regulate user activities in this area using these properties.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
evtPrinterName	string	Print operations only	Windows, Linux, macOS	The fully qualified UNC printer name. The Digital Guardian Agent does not distinguish between local and network printers.

Process-Related Rule Properties

With a new event, you can monitor, control and log all the DLLs that a process loads. You get the hash of each DLL. The Agent reports the DLL load operation to the DG Server the same way it reports File Write operations.

You can use the following process-related rule properties to capture the ID for processes.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
curProcessID	int32	All	Windows, Linux, macOS	The ID of the current process.
parentProcessID	int32	All	Windows, Linux, macOS	The ID of the parent of the current process. Available for all types of operations.
prevProcessID	int32	All	Windows	The ID of the source process where the data came from in an ADE operation.

To capture processes being loaded, use `constOpImageLoad` with `evtOperationType`.

Windows Environment Variables

Windows environment variables allow you to create values based on local settings without knowing exactly what that value is in advance.

For a listing of Windows environment variables, refer to Microsoft’s Windows documentation.

Using the environmentVariable Rule Property

This property lets you evaluate events by comparing them to the values of specific environment variables. For example, you could evaluate the current user to determine if he or she is an administrator.

Property Name	Data Type	Applicable Operation Types	Valid Operating Systems	Description
environment Variable	string	All	Windows	A valid Windows environment variable.

Using an Environment Variable in Another Rule Property

Digital Guardian allows you to use Windows environment variables in rule properties that accept string values. For example, each user of a computer has a unique My Documents folder. Without the #HOMEPATH# variable, you would have to write a rule explicitly regulating each user’s path by name. By referring to a Windows environment variable, you can regulate all users of the computer with the same rule property.

To refer to an environment variable in a rule property, use it in a <regex> or <like> relational operator.


```
<regExp expr=
" ^c: \\documents and setti ngs\\##username##\\l ocal setti ngs\\. *" >
  <evtSrcFi lePath/>
</regExp>
```

This example shows how you would refer to a the local settings folder using the ##username## environment variable.

Note: Regular expressions use a number of parameters to help you construct a value. For more information about using regular expressions, refer to [“Regular Expressions” on page 231](#).

Windows Registry Rule Properties

Windows Registry rule properties reference values from the Windows Registry at rule execution time. Digital Guardian supports values from the following Windows Registry hives:

- HKEY_LOCAL_MACHINE (or HKLM)
- HKEY_USERS (or HKU)
- HKEY_CURRENT_CONFIG (or HKCC)

For information about the values contained within the Windows Registry, refer to your Microsoft Windows documentation.

Digital Guardian rules that use the registry properties may not work consistently with the HKEY_CURRENT_USER hive. Similarly, rules that use the registry properties do not support the HKEY_CLASSES_ROOT hive. The HKEY_CLASSES_ROOT hive is a user-mode hive that draws values from the other hives.

To reference a value from HKEY_CLASSES_ROOT, locate that value in one of the other hives. The HKEY_CURRENT_USER hive does not exist until a user logs on to the computer under a user profile. In some cases, users might not have profiles while they access the computer. To ensure that your registry rules work, test them before you deploy them.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
registryInt	integer	All	Windows	The integer value contained in the Windows registry key.
registryString	string	All	Windows	The string value contained in the Windows registry key. String values are not case sensitive.

This example shows a rule that prevents file downloads if the local security product version reported in the Windows Registry is less than 10. For the rule example, the registry entry is a REG_DWORD, so it uses the registryInt property,

testing with "int value=" If the value in the registry is a REG_SZ, you use the registryString property and compare the value with "string value="

```
<and>
<lessThan>
<registryInt key="hkey_local_machine\software\
Security Product\Security Product Client"
value="SecurityProduct Version"/>
<int value="10"/>
</lessThan>
<equal>
<evtOperationType/>
<constOpNetTransferDownload/>
</equal>
</and>
```

Registry Event Properties

Registry events have some unique properties you can use in rules, along with the common properties.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtRegistryOperationType	int8	All	Windows	The operation type that occurred. This is defined by an enumeration. Refer to "Registry Event Types" for more information.
evtRegistrySrcPath	wstring	All	Windows	The source path for the operation. This always contains the full path to the registry item the operation was performed on.
evtRegistryDestPath	wstring	All	Windows	The destination path for the operation. This is an empty string except for "rename" operations.

Property Name	Data Type	Application Operation Types	Valid Operating Systems	Description
evtRegistryValueType	int8	All	Windows	Specifies the value type the operation was on. Refer to “Registry Event Operation Types” on page 180 for more information.
evtRegistryValueDWORD	int32	All	Windows	The actual DWORD of the value involved in the operation. In set operations, this is the final value in the registry.
evtRegistryValueQWORD	int64	All	Windows	The QWORD of the value involved in the registry operations.
evtRegistryValueStringSZ	wstring	All	Windows	The string value involved in the registry operation.
evtRegistryValueMultiStringSZ	wstring or vector	All	Windows	A list of strings involved in the multiple string operation. This list can be quite long. The DG Server holds roughly the first 2048 bytes of data in the database.

Registry Event Operation Types

Some registry properties return a numeric value associated with the operation—`evtRegistryOperationType` and `evtRegistryValueType`. The returned values reflect the operation, shown in this table.

Operation	Operation/Type Value	Associated Symbolic Constant
Registry Create Key	1	<code>constRegistryCreateKey</code>
Registry Rename Key	2	<code>constRegistryRenameKey</code>

Operation	Operation/ Type Value	Associated Symbolic Constant
Registry Delete Key	3	constRegistryDeleteKey
Registry Delete Value	4	constRegistryDeleteValue
Registry Set Value	5	constRegistrySetValue

Note: The Agent returns registry operation information only if you have rules related to registry events. Registry operations are not available without event rules applied to the Agent.

Sample Registry Rules

To help you understand how registry operations work, here are two rule examples.

Example 1 — Rule To Activate Registry Events for a Particular Value

After any copy or move operation, this rule tells the Agent to monitor the value test under the registry\machine\software\google\ key. After the DG_SetRegistryValueEvents function runs, anytime the data for test is set or the value test is deleted, the Agent generates a registry event. The event will then be processed by any existing control rules. Generally, you will write a separate rule to process the registry event and do something, such as block the operation.

```
<and>
  <function name="DG_SetRegistryValueEvents">
    <parameters>
      <string value="\registry\machine\software\google\test"/>
    </parameters>
  </function>
  <in>
    <eventOperationType />
    <list>
      <constOpFileMove />
      <constOpFileCopy />
    </list>
  </in>
</and>
```

Example 2 — Rule To Process Registry Events

Based on the example rule in [“Example 1 — Rule To Activate Registry Events for a Particular Value” on page 181](#), this rule matches True when the test value in the registry changes or gets deleted.

```
<and>
  <like expr="%\test" >
    <eventRegistrySrcPath>
  </like>
  <equal >
    <eventOperationType />
    <constOpRegistry />
  </equal >
</and>
```

Property Value Types

Each value type is validated for correctness on the DG Server before you can save the rule. Value types determine the values that you can enter in a rule property.

bluetoothAddress

An IP address that represents a 6-byte Bluetooth wireless device address.

```
<bluetoothAddress value="00: 0E: 9B: DA: BA: 70" />
```

bool

The bool type is a logical value, which can be assigned the literal true or false.

```
<bool value="true" />
```

dateTime

The dateTime type represents a date and time. Any datetime value must be of the form mm/dd/yyyy hh:mm. For example, a dateTime value might resemble the following example:

```
<dateTime value="11/25/2006 11:00 pm" />
```

With a dateTime value, you can optionally specify whether the time value is am, pm, or utc time. If you do not specify am, pm, or utc, Digital Guardian interprets the time as am.

ipAddress

An IP address represents an IPv4 address.

```
<i pAddress value="10. 10. 10. 10" />
```

constant

A constant is one of a list of predefined values for that property. Constants are described in detail in [“Symbolic Constants” on page 189](#).

```
<evtSrcDriveType/>  
<list>  
<constDriveUnknown/>  
<constDriveRemovable/>  
</list>
```

int

The int type represents a signed 32-bit integer. Even though the data type is signed, negative numbers are not currently supported.

```
<int value="10" />
```

int32

The int32 type represents a signed 32-bit integer. Even though the data type is signed, negative numbers are not currently supported.

```
<int32 value="10" />
```

int64

The int type represents a signed 64-bit integer. Even though the data type is signed, negative numbers are not currently supported.

```
<int64 value="10" />
```

irdaAddress

An IP address that represents a 4-byte (8-hexadecimal) IrDA infrared wireless device address.


```
<irdaAddress value="a9a88"/>
```

macAddress

A MAC address represents an Ethernet card's layer 2 hardware address.

```
<macAddress value="00-09-6B-E3-78-4A"/>
```

md5Hash

The MD5 Hash type is a GUID representing the MD5 content signature of the process.

```
<md5Hash value="{eaf0b7c2-381c-98eb-fer6-ac7d95266247}"/>
```

For more information about using MD5 in your rules, refer to [“Using MD5 in Rules” on page 233](#).

sha1

The SHA1 Hash type is a 20-byte integer representing the SHA1 content signature of the process.

```
<sha1hash value="b591a42e1e298315d4fa744c555cf54aedb348eb"/>
```

sha256

The SHA256 Hash type is a 32-byte integer representing the SHA256 content signature of the process.

```
<sha256Hash value="3aecae448a9b2a3bed7127b85039bdbe85e725a4037a441b4e5015a3bd791d5a"/>
```

string

A string represents a sequence of characters. String values are case insensitive (or optionally case sensitive with the `preserveCase` property attribute).

```
<string value="i explore.exe"/>
```

time

The time type represents a specific 24 hour time. If no AM/PM is assigned, the time is assumed to be a 24 hour time value. If you do not specify a UTC value, Digital Guardian uses local time.

```
<time value="13:00 UTC"/>
```

Valid Relational Operators for Each Value Type

The following table lists the value types and the relational operators that are valid for each value type.

Value Type	Valid Relational Operator
bluetoothAddress	Equal and in
bool	Equal
dateTime	Equal, less than and greater than
int	Equal, greater than, less than, and in
ipAddress	Equal, ipMask, and in
irdaAddress	Equal and in
macAddress	Equal and in
md5Hash	Equal and in
string	Equal, in, like, and regular expression
time	Less than and greater than (does not use equal with time type)

Email Address Format

For rules governing email events, you can include email address information in the text of the rule itself. Depending on the email client you are regulating, write your rules to match that client's address format.

In most cases, the DG Agent can extract a standard SMTP format email address from an email. In the event that the Agent cannot extract the email address, it uses any X.400 format address information it can find. If you are writing rules and you want to be sure that you are covering all potential email address formats, use the <or> logical operator to join multiple address formats.

SMTP Format

Most SMTP and Web clients use the following format for email addresses:

<user name>@<domain>

A full email address might look like the following example:

lmanion@example.com

X.400 Format

Microsoft Outlook and IBM Notes also use the X.400 format for email addresses. X.400 addresses use the following format:

cn=<user name>/ou=<organizational unit>/o=<domain>

where

- cn = <user name> provides the user name of the recipient.
- ou = <organizational unit> is the name of each group to which the recipient belongs. Multiple groups are listed in alphabetical order. For example, you might see something like this for two groups:
/ou=Engineering/ou=Support
- o = <domain> is the domain of the recipient.

A full X.400 format email address might resemble the following example:

cn=al eksey Efimov/ou=ADG/ou=Marketing/o=digital guardian.com

CHAPTER 5 Symbolic Constants

Some rule properties are limited to a specific set of valid values called symbolic constants. For example, the property `evtSrcDriveType` is limited to the values defined for [Drive Types](#). You can use symbolic constants in your rule definitions to identify the actions and objects that you want to observe and regulate.

Symbolic Constants and Agent Operating Systems

Some symbolic constants are not applicable to all Digital Guardian Agent operating systems. For example, the symbolic constant `constOpFileRecycle` is only applicable to rules running on Windows operating systems. If you use this constant in a rule applied to an Agent running on a non-Windows operating system, the Agent ignores the rule, potentially creating security holes in your enterprise.

Digital Guardian strongly suggests that you maintain separate rule sets for each operating system.

Application Data Exchange Types

Application Data Exchange types describe the application data exchange event that you want to regulate in your rule. For example, you might want to control cut and paste actions, but allow drag and drop actions. **To record these events, you must have an alert associated with these events.**

Note: For Digital Guardian version 5.0 and later Agents for Windows, Application Data Exchange events are now governed by the `evtOperationType` property. Existing rules for Windows Agents based on the `evtADEType` property will continue to function normally.

For more information, refer to [“evtOperationType” on page 92](#) and [“Operation Types” on page 199](#).

Name	Valid Operating Systems	Description
<code>constCutPaste</code>	Linux	Application data exchange involving keyboard or menu based cut and paste.
<code>constDragDrop</code>	Linux	Application data exchange involving drag and drop.
<code>constPrintScreen</code>	Linux	Application data exchange involving the Print Screen key.

These constants are used by the `evtADEType` rule property. For more information, refer to [“evtADEType” on page 113](#).

Bus Types

Bus types describe the type of system bus that you want to regulate in your rule. For example, you might want to regulate firewire (1394) drives, or filter out events performed on RAID drives.

Name	Valid Operating Systems	Description
constBus1394	Windows, Linux, macOS	The bus is a Firewire (1394) bus.
constBusATA	Windows, Linux, macOS	The bus is an Advanced Technology Attachment bus.
constBusATAPI	Windows, Linux, macOS	The bus is an Advanced Technology Attachment Packet Interface bus.
constBusScsi	Windows, macOS	The bus type is an Internet Small Computer System Interface bus.
constBusRAID	Windows, Linux, macOS	The bus type is a Redundant Array of Inexpensive Disks.
constBusSas	Windows, macOS	The bus type is a Serial Attached SCSI bus.
constBusSata	Windows, macOS	The bus type is a Serial ATA bus.
constBusScsi	Windows, Linux, macOS	The bus type is a Small Computer System Interface bus.
constBusSD	Windows, macOS	The bus type is Secure Digital non-volatile memory (SD card).
constBusSSA	Windows, Linux, macOS	The bus type is a Serial Storage Architecture bus.
constBusFibre	Windows, Linux, macOS	The bus type is a fibre channel host bus.
constBusUnknown	Windows, Linux, macOS	The bus type is unknown.

Name	Valid Operating Systems	Description
constBusUSB	Windows, Linux, macOS	The bus type is a Universal Serial Bus.
constBusMmc	Windows, macOS	The bus type is multimedia card.
constBusVirtual	Windows, macOS	The bus type is virtual. Associated with virtual disks.
constBusFileBacked-Virtual	Windows, macOS	The bus type is a file-backed virtual bus, a virtual disk.

These constants are used by the `evtDestBusType` and `evtSrcBusType` rule properties. For more information, refer to [“evtDestBusType” on page 83](#) and [“evtSrcBusType” on page 93](#).

Connection Types

Connection types describe the type of connections to regulate in your rule. To enable you to specify types of connections, the property `evtConnectionType` supports these constants:

Name	Valid OS	Description
constConnNormal	Windows	Evaluates to true when the user is connected over a connection that is not wireless
constConnWireless	Windows	Evaluates to true when the user is connected over a wireless connection

Days of the Week

Days of the week describes the values used to identify days of the week in a rule.

Name	Valid Operating Systems	Description
constSunday	Windows	Sunday
constMonday	Windows	Monday
constTuesday	Windows	Tuesday
constWednesday	Windows	Wednesday
constThursday	Windows	Thursday
constFriday	Windows	Friday
constSaturday	Windows	Saturday

These constants are used by the evtOperationType rule property. For more information, refer to [“agentCurrent DayOfWeek” on page 72](#).

Drive Types

Drive types describe the type of storage drive to regulate in your rule. For example, you might regulate removable drives, but not fixed drives.

Drive types can also determine the source content for an action. For example, the constant `constDriveScreen` specifies content on an application screen.

These constants are used by the `evtDestDriveType` and `evtSrcDriveType` rule properties. For more information, refer to [“`evtDestDriveType`” on page 83](#) and [“`evtSrcDriveType`” on page 94](#).

Name	Valid Operating Systems	Description
<code>constDriveCDRom</code>	Windows, Linux, macOS	The drive is a CD-ROM or DVD drive.
<code>constDriveFixed</code>	Windows, Linux, macOS	The disk cannot be removed from the drive and the drive cannot be removed from the computer.
<code>constDriveRamDisk</code>	Windows, Linux, macOS	The drive is a RAM disk.
<code>constDriveRemote</code>	Windows, Linux, macOS	The drive is a network drive.
<code>constDriveUnknown</code>	Windows, Linux, macOS	The operating system could not determine the drive type.
<code>constDriveRemovable</code>	Windows, Linux, macOS	The drive can be removed from the computer. Usually a USB device.

Name	Valid Operating Systems	Description
constDriveScreen	Windows, Linux, macOS	The source content is on an application screen.
constDriveUrl	Windows, Linux, macOS	The source content is on a web page.

Email Recipient Types

Email recipient types describe the types of recipients included on an email. For example, you might want to regulate which recipients can be included on the BCC line of an email.

Name	Valid Operating Systems	Description
constMailBCC	Windows, Linux, macOS	Recipients on the BCC line of an email.
constMailCC	Windows, Linux, macOS	Recipients on the CC line of an email.
constMailTo	Windows, Linux, macOS	Recipients on the To line of an email.

These constants are used by the `evtMailRecipientTypes` rule property. For more information, refer to [“evtMail RecipientTypes” on page 153](#).

Encryption Types

Encryption types describe types of encryption applied to the email, such as S/MIME or adaptive AES256.

Name	Value	Valid Operating Systems	Description
constEncryptionType 3DESAptive	2	Windows	Specifies that adaptive Triple DES encryption was used to encrypt the file.
constEncryptionType 3DESPasswordProtected	8	Windows	Specifies that password-protected Triple DES encryption was used to encrypt the file.
constEncryptionType AES256Adaptive	1	Windows	Specifies that adaptive AES256 encryption was used to encrypt the file.
constEncryptionType AES256PasswordProtected	7	Windows	Specifies that password-protected Triple DES encryption was used to encrypt the file.
constEncryptionType AES256ZipAdaptive	5	Windows	Specifies that adaptive AES256 ZIP encryption was used to encrypt the file.
constEncryptionType AES256ZipPassword	6	Windows	Specifies that adaptive AES256 ZIP encryption with a decryption password was used to encrypt the file.
constEncryptionType DefaultEncryption	10	Windows	Specifies that default encryption was used to encrypt the file.
constEncryptionType InPolicyRaw	9	Windows	Specifies that In Policy Raw encryption was used to encrypt the file.
constEncryptionTypeNone	0	Windows	Specifies that the file is not encrypted.
constEncryptionTypeNTFS	255	Windows	Specifies that NTFS encryption was used to encrypt the file.

Name	Value	Valid Operating Systems	Description
constEncryptionType SMIME	11	Windows	Specifies that S/MIME encryption was used to encrypt the message and attachments.
constEncryptionType StdZipAdaptive	3	Windows	Specifies that adaptive standard ZIP encryption was used to encrypt the file.
constEncryptionType StdZipPassword	4	Windows	Specifies that standard adaptive ZIP encryption with a decryption password was used to encrypt the file.

Network Adapter Types

Network Adapter types describe the type of network adapter that you want to regulate in your rule. For example, you might want to regulate Ethernet or VPN connections.

Name	Valid Operating Systems	Description
const1394	Windows	The network adapter is a Firewire port.
constBluetooth	Windows	The network adapter is a Bluetooth port.
constEthernet	Windows	The network adapter is an Ethernet port
constIrDA	Windows	The network adapter is an infrared port.
constVPN	Windows	The network adapter is Virtual Private Network connection. This constant supports monitoring and controlling operations over Microsoft VPN only, the VPN client built into Windows 10 operating systems.
constWireless	Windows	The network adapter is a wireless port.

These constants are used by the `evtPhysicalMedium` rule property. For more information, refer to [“evtPhysicalMedium” on page 166](#).

Operation Types

Operation types describe the operation to regulate in your rule.

Name	Valid Operating Systems	Operation Description
constOpAdeCut	Windows, macOS	Application Data Exchange event involving cutting or copying content from a source document.
constOpAdeInsertFile	Windows	OLE insertion of an existing saved file. Use source file name properties to govern the file being inserted.
constOpAdeInsertNewObject	Windows	OLE insertion of a new, unsaved object.
constOpAdePaste	Windows, Linux, macOS	Application Data Exchange event involving pasting content by drag-and-drop or the Windows clipboard.
constOpAdePrintProcess	Windows, Linux, macOS	Application Data Exchange event involving an attempt to print the window of the current application using the Alt+Print Screen keys.
constOpAdePrintScreen	Windows, Linux, macOS	Application Data Exchange involving an attempt to print the entire screen using the Print Screen key.

Name	Valid Operating Systems	Operation Description
constOpAdeScreenCapture	Windows, Linux, macOS	Application Data Exchange involving screen captures by non-native screen capture application such as TechSmith SnagIt or Adobe Captivate.
constOpAppScreenBuffer	Windows, Linux	<p>Scans the contents of the current application screen. Use this property to match the contents of the screen text with content patterns and keywords.</p> <p>This operation type is designed to serve as a data vault trigger rule. When an application process displays sensitive content on screen, Digital Guardian can create a data vault for that process. After a user leaves the sensitive screen, the vault can be lifted.</p> <p>Note: This operation type works only with 3270 applications and Internet Explorer.</p> <p>This operation type requires the Digital Guardian Adaptive Content Inspection add-on to function.</p>

Name	Valid Operating Systems	Operation Description
<p>constOpAppStart</p> <p>Note: Rules that use this constant generate events only when you prevent the application executable from starting.</p>	Windows, Linux, macOS	<p>Use in Control Rules to block an application from starting.</p> <p>Note: You cannot use this to trigger vaulting rules.</p> <p>Note: Prevents other file operations in the same rule from triggering. This rule demonstrates the problem—if the rule triggers on constOpAppStart, file operations that should trigger the rule do not:</p> <pre><and> <equal > <evtSrcFileExt /> <string value="exe" /> </equal > <equal > <evtSrcDriveType /> <constDriveRemovable /> </equal > <in> <evtOperationType /> <list> <constOpFileCopy /> <constOpFileMove /> <constOpFileOpen /> <constOpAppStart /> </list> </in> </and></pre> <p>To avoid problems, create two rules—one for application start processes, the other for file operations.</p> <p>When you write rules to block applications from starting, include only properties that identify the application, such as path or image name.</p>

Name	Valid Operating Systems	Operation Description
constOpCDBurn	Windows, Linux, macOS	CD/DVD Burn/Write
constOpCustomEvent	Windows, Linux, macOS	Trigger on a user-defined custom event. For more information, refer to “Custom Events Functions” on page 325.
constOpDeviceAdded	Windows, Linux, macOS	User inserted a USB device into a port on the computer.
constOpDeviceDetected	Windows, Linux, macOS	At computer start, the Agent detected a new USB device.
constOpDeviceMissing	Windows, Linux, macOS	At computer start, a device that the Agent detected at shut down is no longer detected.
constOpDeviceOpen	Windows, Linux, macOS	User or application process is opening a USB device for use. Use this constant primarily for detecting when the user or application, such as Skype, opens a Web camera.
constOpDeviceRemoved	Windows, Linux, macOS	User removed a USB device from the computer.
constOpDocumentRepository	Windows	Identify document repositories from third parties, such as Microsoft Sharepoint.

Name	Valid Operating Systems	Operation Description
constOpFileArchive	Windows, Linux	File archive using tar or mkisofs command on Linux. Also supports the WinZip and WinRAR utilities on Microsoft Windows. When you use this constant to block an archive operation, the archive process creates an empty archive, but you cannot add files to the archive.
constOpFileClose	Windows, Linux	Capture the file close operation. To have this work in rules, you must set Execute Rule After Operation to Yes in the rules.
constOpFileCopy	Windows, Linux, macOS	File copy.
constOpFileCreate	Windows, Linux	File create. This event type is intended for use only with the Encrypt rule action. Note: To regulate file create events, use rule properties that govern the source process. You can recognize these properties based on their evtSrc prefix. File create is a source file process. No destination file path is available.

Name	Valid Operating Systems	Operation Description
constOpFileDecrypt	Windows, Linux	Manual file decryption by user. Note: This event refers only to files encrypted by Digital Guardian Adaptive File Encryption.
constOpFileDelete	Windows, Linux, macOS	Delete a file without moving it to the recycle bin.
constOpFileEdit	Windows, Linux	Change and save a file in place.
constOpFileMove	Windows, Linux, macOS	Move a file from one location to another.
constOpFileOpen	Windows, Linux, macOS	Create a handle, file object, or file stream to provide read/write access to the file. Does not read or write the file.
constOpFileRead	Windows, Linux, macOS	Read from the file stream, handle, or object to read the contents of a file.
constOpFileRecycle	Windows, Linux, macOS	Send a file to the Microsoft Windows recycle bin.
constOpFileRename	Windows, Linux, macOS	File and folder rename.
constOpFileRestore	Windows, Linux	Restore a file from the Microsoft Windows recycle bin expand a .tar file.
constOpFileSaveAs	Windows, Linux, macOS	Save a file with a new name or possibly a new location.

Name	Valid Operating Systems	Operation Description
constOpFileView		Obsolete. Rules referencing this constant will compile without errors, but no event will be generated.
constOpFileWrite	Windows, Linux, macOS	Write information to a file. The system writes to the object, stream or handle associated with the file. File write does not have destination file paths.
constOpHook	Windows	Capture a hooking event for a process, such as notepad.exe, with a property such processImageName or currentImageProcessName. Use with evtOperationType. Also use to report hooking of APIs, such as MoveFile or CopyFileEx.
constOpLogon	Windows, Linux, macOS	Monitor and report when a user logs on to a computer. When you use this in rules, select No for Execute Rule After Operation when you create the rules. The rule engine treats logging on events as happening before the event, so you cannot use Execute Rule After Operation set to Yes.

Name	Valid Operating Systems	Operation Description
constOpLogoff	Windows, Linux, macOS	Monitor and report when a user logs off a computer. DG does not support blocking the log off process. Rules to block log off will report the log off process but cannot block it.

Name	Valid Operating Systems	Operation Description
constOpLogonFailed	Windows	<p>Monitor and report when users attempt to log on to Windows computers and the logon requests fail. The DG Agent receives failed logon events from Windows computers only if you enable settings in the Windows Local Group Policy Object to audit logon attempts on those computers.</p> <p>To enable auditing for logon events, enable three options in the Advanced Audit Policies > Logon/Logoff section:</p> <ul style="list-style-type: none">a) Configure the Audit Account Lockout policy setting for Failure.b) Configure the Audit Logon events policy setting for Failure.c) Configure the Audit Other Logon/Logoff events policy setting for Failure. <p>To learn more about this Windows Local GPO and how to enable and configure it, refer to Microsoft's documentation for configuring Local Security Policy.</p>

Name	Valid Operating Systems	Operation Description
constOpMailAttach	Windows, Linux	File attached to an Outlook or IBM Notes message. Mail attach events take place immediately in IBM Notes and when a user saves a draft of the message or sends the message in Microsoft Outlook. You can block, encrypt, or classify on this event. If you create a rule to encrypt the attachment, Digital Guardian maintains the encryption when sending the message unless another rule changes the encryption state. Mail attach events support Adaptive Encryption only.
constOpNetTransferDownload	Windows, Linux, macOS	Download file over network using any connection based protocol such as FTP or HTTP.
constOpNetTransferUpload	Windows, Linux, macOS	Upload file over network using FTP, Internet Explorer, or instant messaging applications

Name	Valid Operating Systems	Operation Description
constOpNetwork	Windows, Linux, macOS	Identifies all network operations (connect, listen, accept connections, and more). On Microsoft Windows platforms, you can use this to block UDP traffic, but the Agent will not display block prompts.
constOpNetworkEx	Windows, Linux, macOS	Identifies all network operations (connect, listen, accept connections, and more). Allows you to identify the specific type of network operation the event represents, such as a DNS look-up or an HTTP connection, using the evtNetworkOperation property.
constOpOperation	Windows	Reports when the DG Agent has performed an operation on the host machine. Currently reports some file capture events.
constOpPrint	Windows, Linux, macOS	Print file.
constOpSendMail	Windows, Linux, macOS	Send e-mail

These constants are used by the evtOperationType rule property. For more information, refer to [“evtOperationType” on page 92](#).

Product Types

Note: Digital Guardian no longer supports working with Documentum repositories.

(Obsolete) Product types specify the type of repository you are using. The associated rule property is `evtDocRepositoryProductType`. This table lists the available constants.

Name	Valid Operating System	Description
<code>constDocRepositoryWebtop</code>	Windows	(Obsolete) Specifies that you are accessing an EMC Documentum repository.

Protocol Types

Protocol types describe the transfer type that you want to regulate in your rule.

Name	Valid Operating System	Description
constProtocolBluetooth	Windows, Linux, macOS	Bluetooth Protocol
constProtocolDNS	Windows, macOS	Domain name system (DNS)
constProtocolHTTP	Windows, Linux, macOS	Hypertext Transfer Protocol
constProtocolIrDA	Windows, Linux, macOS	Infrared Data Association (IrDA)
constProtocolTCP	Windows, Linux, macOS	Transport Control Protocol (TCP)
constProtocolUDP	Windows, Linux, macOS	User Datagram Protocol (UDP)

These constants are used by the `evtProtocolType` rule property. For more information, refer to [“evtProtocolType” on page 166](#).

Registry Event Types

The property `evtRegistryOperationType` uses a set of symbolic constants to define the registry operation in the rule.

Name	Value	Valid Operating Systems	Description
<code>constRegistryCreateKey</code>	1	Windows	The operation created a registry key.
<code>constRegistryDeleteKey</code>	3	Windows	The operation deleted a registry key.
<code>constRegistryDeleteValue</code>	4	Windows	The operation deleted the value in a registry key.
<code>constRegistryQueryValue</code>	6	Windows	The registry operation queried a value.
<code>constRegistryRenameKey</code>	2	Windows	The operation renamed a registry key.
<code>constRegistrySetValue</code>	5	Windows	The operation set a registry value in a key.
<code>constRegistryUnknown</code>	0	Windows	The registry operation is not known.

Registry Event Values

The property `evtRegistryOperationType` uses a set of symbolic constants to define the value of registry entries in the rule. The rule returns the value dependent on the type of value in the registry.

Name	Value	Valid Operating Systems	Description
<code>constRegistryValueTypeDWORD</code>	1	Windows	Returned when the value of the key is a DWORD.
<code>constRegistryValueTypeMultiStringSZ</code>	4	Windows	Returned when the value type is list of strings involved in the multiple string operation. This list can be quite long. The DG Server holds roughly the first 2048 bytes of data in the database.
<code>constRegistryValueTypeQWORD</code>	2	Windows	Returned when the value of the key is a QWORD.
<code>constRegistryValueTypeStringSZ</code>	3	Windows	Returned when type of the value is a string.
<code>constRegistryValueTypeUnknown</code>	0	Windows	Returned when the type of the value in the operation is not known.

Rule Actions

When you use Continue Rule Evaluation in your rules, you can use the rule property `evtCurrentRuleAction` to test which earlier (higher priority) rules matched the event, if any.

Continuing rule evaluation allows you to write rules that take action based on the result of rules that have matched the event during rule evaluation and where you specified Continue Rule Evaluation. With these properties, you can:

- Use the result of a prompt in a second rule that has lower priority.
- Base additional rules on whether previous rules would have blocked the event.
- Change the action of a previous rule by matching on a higher-execution-order (lower priority) rule.
- You can check lower-priority rules to see what a previous higher-priority rule has set.

Constant Name	Valid Operating System	Description
<code>constRuleActionBlock</code>	Windows	Previous rule action was Block.
<code>constRuleActionContinue</code>	Windows	Previous rule action was Continue.
<code>constRuleActionEncrypt</code>	Windows	Previous rule action was Encrypt.
<code>constRuleActionNone</code>	Windows	Previous rule action was None. No previous rule has matched the event being evaluated.
<code>constRuleActionVault</code>	Windows	Previous rule action was Vault.

Wireless Authentication Types

Wireless Authentication types describe the types of wireless authentication that you want to regulate in your rule. For example, you might write rules that prevent your wireless cards from using insecure authentication types.

Name	Valid Operating Systems	Description
constOpen	Windows	Open, unsecured access
constShared	Windows	Wi-Fi shared access
constWPA	Windows	Wi-Fi Protected Access
constWPA2	Windows	Wi-Fi Protected Access 2
constWPA2PSK	Windows	Wi-Fi Protected Access 2 with Pre-shared Key
constWPANone	Windows	Wi-Fi Protected Access with no key
constWPAPSK	Windows	Wi-Fi Protected Access with Pre-shared Key

These constants are used by the evtWirelessAuthenticationMode rule property. For more information, refer to [“Network Operation Rule Properties” on page 163](#).

Wireless Encryption Types

Wireless Encryption types describe the types of encryption used to secure your wireless communications. For example, you might write rules that require a certain type of encryption.

Name	Valid Operating Systems	Description
constAES	Windows	Advanced Encryption Standard (AES)
constNone	Windows	No encryption applied
constTKIP	Windows	Temporal Key Integrity Protocol (TKIP)
constWEP	Windows	Wired Equivalent Privacy encryption (WEP)

These constants are used by the `evtWirelessEncryption` rule property. For more information, refer to [“Network Operation Rule Properties” on page 163](#).

Wireless Infrastructure Mode Types

Wireless Authentication types describe the types of wireless infrastructure that you want to regulate in your rule. For example, you might write rules that prevent your users from forming ad hoc wireless groups.

Name	Valid Operating Systems	Description
constAccessPoint	Windows	Wireless access points network
constAdHoc	Windows	Ad hoc wireless network

These constants are used by the `evtWirelessInfrastructureMode` rule property. For more information, refer to [“Network Operation Rule Properties” on page 163](#).

CHAPTER 6 Logical Operators

In cases where you have multiple event operations, or multiple rule properties, you can use logical operators to specify the Boolean relationships between those expressions. Logical operators can wrap an entire rule definition, and can be nested. For example, most rules are wrapped with the `<and>` operator to specify that all the conditions within the rule must be true. Within that `<and>`, you might also have a nested `<or>` operator, if, for example, you only require some rule properties to be true.

The DG Agent supports the nesting of logical operations provided they are syntactically correct. You can use logical operators to examine events in your rules.

AND

Evaluates the logical AND operation on two or more expressions. Expressions connected by AND are evaluated from the bottom expression up. Evaluation stops as soon as the truth or falsehood of the result is known.

```
<and>  
  <expressi on/>  
  <expressi on/>  
  <expressi on/>  
</and>
```

OR

Evaluates the logical OR operation on two or more expressions. Expressions connected by OR are evaluated from the bottom expression up. Evaluation stops as soon as the truth or falsehood of the result is known.

```
<or>  
  <expressi on/>  
  <expressi on/>  
  <expressi on/>  
</or>
```

NOT

Evaluates the logical NOT operation on one expression. Expressions connected by NOT are evaluated from the bottom expression up.

```
<not>  
  <expressi on/>  
</not>
```

Nested Logical Operators

The following example demonstrates the format of a rule that uses a nested operation:

```
<and>  
  <expression/>  
  <or>  
    <expression/>  
    <expression/>  
  </or>  
  <not>  
    <expression/>  
  </not>  
</and>
```

CHAPTER 7 Relational Operators

Relational operators evaluate one or two system properties, symbolic constants, or user defined values. Relational operators can construct strings to match property values, evaluate lists of property values, or compare multiple expressions to one another.

Nesting within a relational operator is not allowed but is allowed within a logical operator. Digital Guardian supports the following relational operators:

- [“Equal” on page 223](#)
- [“Greater Than” on page 223](#)
- [“In” on page 223](#)
- [“IP Mask” on page 224](#)
- [“Less Than” on page 224](#)
- [“Like” on page 225](#)
- [“Op Operator” on page 228](#)

Using Process Names With Relational Operators

There are some circumstances under which the rule engine removes characters in process names longer than 14 characters. Best practice is to limit the length of process names in rules to 14 characters or fewer.

For example, when you use the Equal operator with a process name, the rule engine takes the first 14 characters of the name and ignores the rest. For example, if you deploy the following rule:

```
<and>
  <equal >
    <curProcessImageName />
    <string value="ZipSendService.exe" />
  </equal >
  <equal >
    <eventOperationType />
    <constOpAppStart />
  </equal >
</and>
```

The rule will fire for any process image name for which the first 14 characters are ZipSendService.

Also, when you use the Like operator or regular expressions, your process name-based rules will not fire if you include more than 14 characters in the process name. If you deploy this rule, it will not fire on any process name:

```
<and>
  <like expr="%ZipSendService.exe%">
    <curProcessImageName />
  </like>
  <equal >
    <eventOperationType />
    <constOpAppStart />
  </equal >
</and>
```

Equal

This operator performs an equality test on two operands. All value types, except time, are valid operands.

```
<equal >
<evtDestDriverType/>
<constDriverRemovable/>
</equal >
```

Greater Than

This binary operator performs a left to right greater than comparison on two operands. The expression will evaluate to true if the first value is greater than the second, otherwise it will evaluate to false. Only time and integer values are valid operands.

```
<greaterThan>
<agentLastServerComm/>
<int value="720"/>
</greaterThan>
```

In

This operator takes one property and matches it against a list of possible values. The values are compared from bottom to top. Evaluation stops as soon as the truth or falsehood of the result is known. IP and MAC addresses, MD5 hashes, and integer values are valid operands. The list can contain only values of the same data type.

```
<in>
<evtRemotePort/>
<list>
<int value="80"/>
<int value="443"/>
</list>
</in>
```

IP Mask

This operator takes one operand and enables Digital Guardian to compare an IP address to values that are valid for a given subnet. Specify the IP network using the slash notation. Only IP address values are valid operands.

```
<i pMask mask="10. 10. 10. 0/24">  
<evtRemoteAddress/>  
</i pMask>
```

Network Masks

Bit Length	Network Masks	Usable IP Addresses
/32	255.255.255.255	0 (single-host net mask)
/31	255.255.255.254	0 (point-to-point)
/30	255.255.255.252	2
/29	255.255.255.248	6
/28	255.255.255.240	14
/27	255.255.255.224	30
/26	255.255.255.192	62
/25	255.255.255.128	126
/24	255.255.255.0	254

Less Than

This binary operator performs a left to right less than comparison on two operands. The expression will evaluate to true if the first value is less than the second, otherwise it will evaluate to false. Only time and integer values are valid operands.

```
<LessThan>
```



```
<agentCurrentTime/>
<time value="09:00 utc"/>
</lessThan>
```

Like

This operator locates substrings within a rule property. The `<like>` operator is faster, but less flexible than a regular expression. You can use the `<like>` operator with any rule property that has a data type of string.

For rule properties that contain multiple values, the `<like>` operator can also include a match attribute. For more information on match attributes, refer to [“Match Attributes” on page 229](#).

You can use the `%` character as a wildcard indicator at either end of a string. For example, using the expression `“%adobe”` would only match property values that ended in adobe. The `%` character is the only supported wildcard for the `<like>` operator. For more complex comparisons, use regular expressions. For information about regular expressions, refer to [“Regular Expressions” on page 231](#).

The `<like>` operator is not case sensitive.

Note: The `%` wildcard must appear at either end of a string. Placing the `%` wildcard in the middle of a string is syntactically invalid. Each `<like>` expression must contain at least one wildcard.

Find Value Anywhere in a String

To find a substring anywhere within a string, place the `%` wildcard at the beginning and end of the search string, as shown in the following example:

```
<like expr="%adobe%">
<curProcessCompanyName/>
</like>
```

This example would evaluate to true if the company name property contained the value "adobe" in any location. For example, all of the following values would evaluate to true:

- the adobe company
- adobe inc.
- copyright adobe

Find Value at Beginning of a String

To find a substring at the start of a string, place the % wildcard at the end of the search string, as shown in the following example:

```
<like expr="//SecureServer%">  
<evtSrcFilePath/>  
</like>
```

This example evaluates to True when the source file path property starts with \\SecureServer\\. Strings that contain the \\SecureServer\\ value elsewhere in the path evaluate to False.

Find Value at End of a String

To find a substring at the end of a string, place the % wildcard at the beginning of the search string, as shown in the following example:

```
<like expr=".exe">  
<curProcessImageName/>  
</like>
```

This example would evaluate to True If the process image name property ended with .exe. Strings that contain the .exe value elsewhere in the image name would evaluate to False.

Find Unicode Values

The <like> operator also supports unicode characters. For unicode strings, use the <like> operator just as you would for ASCII strings.

For example, you could have the following example in a rule:

```
<like expr="% 鯨鵠騙 %" >
  <curProcessCompanyName/>
</like>
```

Like Attribute for <in> Like Operator

An attribute for the <in op = "like"> syntax for rules lets you specify wildcard matching without adding the % sign to the string to match. The attribute "like" takes one of three values:

- Left—places the % character at the left end of the string to match any characters at the start of the substring
- Right—places the % character at the right end of the string to match any characters at the end of the substring
- Both—places the % character at the both ends of the string to match any characters at the start or end of the substring

For example, without the attribute, a rule to match some strings might look like this:

```
<in op="like" >
  <some_property />
  <list>
    <string value="%testvalue1%" />
    <string value="%testvalue2%" />
  </list>
</in>
```

The rule matches any strings that include either testvalue1 or testvalue2.

With the attribute, you add the "like" to specify where the wildcard character goes for matching:

```
<in op="like" like="both" >
  <some_property />
  <list>
    <string value="testvalue1" />
    <string value="testvalue2" />
  </list>
</in>
```

This attribute version of the rule matches any strings that include either testvalue1 or testvalue2. It is the same as the rule that uses strings with % signs at both ends.

For brevity, you can omit the `op` attribute. If you do, the example rule changes to this:

```
<i n l i k e="both" >
    <some_property />
    <l i s t>
        <s t r i n g   v a l u e="testval ue1" />
        <s t r i n g   v a l u e="testval ue2" />
    </l i s t>
</i n>
```

Op Operator

To determine if a rule should fire on an operation, you might compare a property value against a list of values specified in the rule. The following example demonstrates one way to do this—the `Op` operator. With the `Op` operator, you use one of a few valid properties to test a property value against specified values.

The `op` operator works with these properties:

- `const` — refer to [“Symbolic Constants” on page 189](#)
- `like` — refer to [“Like” on page 225](#)

You can combine these operators with the optional `Match` attribute and the match settings “any” and “all.”

The `Op` operator does not support using regular expressions (regex) in rules.

This hypothetical syntax example uses `Like` to evaluate a property value against a list of property values in a rule:

```
<i n o p="like" >
    <some_property />
    <l i s t>
        <s t r i n g   v a l u e="%testval ue1%" />
    </l i s t>
</i n>
```

```

        <string value="%testvalue2%" />
    </list>
</in>

```

The following example SendMail rule uses Op with Like and Match to evaluate email recipient addresses against a list of domains for the company Lol.

```

<and>
  <not>
    <in op="Like" match="all">
      <evtMailRecipients />
      <list>
        <string value="%lol.com%" />
        <string value="%lol.org%" />
        <string value="%lol.biz%" />
        <string value="%lol.in%" />
      </list>
    </in>
  </not>
  <equal>
    <evtOperationType />
    <constOpSendMail />
  </equal>
</and>

```

This rule evaluates to True when the recipient's email address domain does not match any of the strings provided.

Match Attributes

The Like and Regular Expression operators can use the optional match attribute to evaluate string values against one or all of the multiple values in a single property. For example, the `evtDestFilePolicyTag` property lists of all of the classification tags associated with a file.

Match All

A Like or Regular Expression operator with a Match All attribute evaluates to True if the string being compared matches **all** of the values contained in the property.

For example, the following property evaluates to True only when all of the recipient addresses of a particular email contain the string “companyone.com” or “companytwo.com”. If any addresses in the property do not match the strings, the property evaluates to False.

```
<regExp expr="(.*@*companyone.com)|(.*@*companytwo.com)"
match="all">
<evtMail Recipients />
</regExp>
```

Match Any

A Like or Regular Expression with a Match Any attribute requires that the string being compared match any of the values in the list. If you do not specify a Match attribute, this is the default behavior for all properties.

For example, the following property would evaluate to True if any of the classification tags associated with a file matched the string “confidential”. The existence of other classification tags would not prevent the property from evaluating to True.

```
<Like expr="confidential" match="any">
  <evtSrcFilePolicyTag />
</Like>
```

Supported Properties

The supported rule properties for the Match attribute include all string properties. Refer to [“Rule Properties” on page 67](#) or the Quick Reference card to see the string properties.

The rule properties clipboardVaultRuleName, curProcessVaultRuleName and prevProcessVaultRuleName are not supported.

Regular Expressions

The RegExp operator evaluates whether the operand matches a given regular expression. For example, to control files whose names contains certain strings of characters, regular expressions enable you to construct an expression that matches those strings. The characters that you include in your regular expressions give you a flexible level of specificity. You can look for a string that appears anywhere in an expression, or you can add detailed instructions to match characters that appear at the beginning or end of a string.

For rule properties that contain multiple values, regular expressions can also include a match attribute. For more information on match attributes, refer to [“Match Attributes” on page 229](#).

Note: Regular expressions work with standard ASCII characters only. Do not include symbols in your regular expressions.

You can include regular expression characters in any rule property that accepts a string value. Use regular expressions only when necessary, as they affect performance on DG Agent computers. Instead, use the `<list>` element in your rule definitions to provide lists of valid values for better performance.

Only string values are valid operands.

Digital Guardian converts properties to lower case before performing a regular expression comparison. As a result, regular expressions are case insensitive.

Character	Description
^	Matches the beginning of a line
.	(period) Matches any character
*	Finds 0 or more of a preceding match
[]	Matches any instance in the bracketed range
	Delimits Boolean OR values within a bracketed set. For example, (app.exe app2.exe).
()	Groups one or more sequences of characters or matches

Character	Description
\$	Matches the end of a line
\	Indicates that the next character should be treated literally, not as a regular expression character

Examples

This example shows a regular expression that checks any product version for the string "microsoft".

```
<regExp expr="mi crosoft">  
<curProcessProductVersi on/>  
</regExp>
```

This example shows a regular expression that checks the beginning of a file path for the directory c: \docrepository. The regular expression includes the ^ character to indicate that the string appears at the beginning of a line. The expression also includes the \ character to indicate that the slash character in the path is to be taken literally, and not as a regular expression operator.

```
<regExp expr="^c: \\docrepository">  
<evtSrcFilePath/>  
</regExp>
```

CHAPTER 8 Using MD5 in Rules

MD5 is a content hash that uniquely identifies executable files. If users move or rename executable files, the MD5 hash remains the same, enabling Digital Guardian to recognize and respond to the application. You can use MD5 in your Digital Guardian rules to identify applications.

The advantage to using MD5 in your rules is that it lets you "fingerprint" a specific application, down to the exact version of that application. By identifying applications so specifically, you guarantee that you permit only acceptable versions of an application in your enterprise. Earlier or later versions of the application, even if they share the same name, will not match the MD5 fingerprint and will be detected by Digital Guardian. In situations where you are running multiple versions of an application, you must generate MD5 for each version.

Alternatively, if the application does not present a security risk, you can identify it by name. The advantage to identifying an application by name is that version numbers are not an issue. As long as the file name matches the name in the rule definition, Digital Guardian accepts the application.

Although the Digital Guardian MD5 hash generation utility is capable of generating GUIDs of MD5 hash for any type of file, you should only use MD5 from executable files in your rules.

Generating GUIDs of MD5 Hash

You must use Digital Guardian's MD5 hash generation utility (MD5uti l . exe) to create GUIDs of MD5 hash. MD5uti l . exe converts the content hash into a GUID to use in your rules. Third-party hash generation tools may not create GUIDs that match MD5uti l . exe-generated GUIDs, so rules based on those third-party GUIDs might not work. The Digital Guardian Server Installer places the hash generation utility file in the following default location:

c:\program files\verdasys\DG Server\bin\MD5uti l . exe

The MD5 hash generation utility can generate rule syntax based on a single executable or a directory. If you select a directory, the MD5 hash generation utility creates MD5 hash GUIDs for all files in the selected directory, including non-executable files. Before you insert the generated hash GUIDs into a rule, make sure that you have remove any MD5 hash GUIDs that reference non-executable files.

1. Using Windows Explorer, navigate to the MD5 hash utility MD5uti l . exe and double-click on it. The MD5 hash utility opens.
2. Click **Select Files**. The Select Files dialog box opens.
3. Select an executable file (*.exe) and click **Open**. The MD5 hash GUID for that executable appears in the MD5 utility.
4. Highlight the generated MD5 hash GUID and press **CTRL+C** to copy it. You can now insert this GUID into Digital Guardian rules.

Adding MD5 Hash to a Rule Definition

You can edit rules that contain MD5 hash GUIDs. For example, if your organization updates its word processing application, you might need to update your rules to cover both the existing and new versions of the word processor.

1. Use the MD5 hash generation utility to generate the MD5 hash GUID for the application to include in your rule.

2. Copy the MD5 hash GUID from the hash generation utility.
3. Using DGMCM, open the rule that you want to edit.
4. Paste the new MD5 GUID into the rule definition. The rule now uses the updated MD5 GUID. You can paste over and replace any old MD5 GUIDs, or you can use the <or> operator in your rule definition to allow both the old and new values in your rule.
5. Click **Save Rule**. The rule is saved and updated.

MD5 Rule Example

This example integrates MD5 hash GUIDs with a typical rule.

```
<and>
  <equal >
    <evtOperati onType/>
    <constOpAdePaste/>
  </equal >
  <or>
    <i n>
      <curProcessI mageName/>
      <!--I denti fi es  three  appli cati ons  by  executabl e  name. -->
      <li st>
        <stri ng  val ue="ymsgr. exe" />
        <stri ng  val ue="msmsg s. exe" />
        <stri ng  val ue="ai m. exe" />
      </li st>
    </i n>
    <i n>
      <curProcessMD5ContentHash/>
      <!--I denti fi es  two  more  appli cati ons  by  MD5. -->
      <li st>
        <md5Hash  val ue="{D27DEA1E-EAF1-FE6E-F380-B99A90228D2F}" />
        <md5Hash  val ue="{033B2A56-6554-3036-7AC4-7FCBOCEADCDE}" />
      </li st>
    </i n>
  </or>
</and>
```

CHAPTER 9 Rules for SharePoint Operations

Digital Guardian supports Microsoft SharePoint using Microsoft Office and through browser access.

When users access documents in Microsoft SharePoint repositories with Microsoft Internet Explorer or other clients, such as Microsoft Office applications, Digital Guardian Agents can monitor, record and control the user actions.

Digital Guardian Features With SharePoint

You can use Digital Guardian to monitor, record, or control files uploaded to or downloaded from SharePoint based on the source path, destination path or both. In addition, you can use DG to monitor, record or control the movement of files to SharePoint based on the classification tags that exist in the document before you upload the document to SharePoint.

Classification

DG enables you to apply DG classification tags to files you download from SharePoint to computers equipped with Agents. You cannot use the Agent to classify files stored in a SharePoint repository.

Supported SharePoint Clients

Digital Guardian Agents can monitor, report and control file events when users access SharePoint repositories with the following clients:

- Microsoft Internet Explorer
- Microsoft Office

SharePoint Terminology

This following documentation assumes that you are familiar with Microsoft SharePoint terminology and administration. Here are some SharePoint terms you should know:

Site — A Web site hosted in a virtual URL location. Sites contain lists and list folders.

Library — A collection of files that users can share on a Web-based SharePoint service. The files might, for example, be documents that a team shares for a project. Team member use Web browsers to access the files.

List — A container in a SharePoint site that stores **list items** and **list folders** that have the same properties.

List Folder — A folder in a SharePoint list. List folders can contain documents or list items that have the same properties. The folder retains the characteristics of items in the list.

List Item — An entry in a SharePoint list. Each list item has properties that map to fields in the list that contains the item, depending on the content type of the item.

Sharepoint Service — A Microsoft Web application that helps manage content and documents in enterprises.

Operations the Agent Can Report, Monitor and Control

This table lists some common file download mechanisms in SharePoint that the Agent can monitor and control. This is not an exhaustive list of available and supported download mechanisms.

Digital Guardian does not support the following download mechanisms from SharePoint repositories or sites		
File Download Process	Applicable SharePoint Version	Comments
Right-click on a file and select Save Target As from the context menu.	All	Supported when the downloaded file is saved in the same format as the file in SharePoint. Event capture results can vary if the user downloads the file as a new file type. For example, downloading a Microsoft PowerPoint presentation in bitmap (.bmp) format.
Click the drop-down arrow next to the name of the file to use. Select Send To > Download a Copy from the drop-down menu.	All	Supported when the downloaded file is saved in the same format as the file in SharePoint. Event capture results can vary if the user downloads the file as a new file type. For example, downloading a Microsoft PowerPoint presentation in bitmap (.bmp) format
Click the drop-down arrow next to the name of the file to use. Select Edit in Application > Save As from the File menu. <i>Application</i> will be an application appropriate to the file type, such as PowerPoint, Excel, or Visual Basic.	All	The user can open the document and read or edit the document. The Agent monitors, reports or controls the movement of the file when the user saves the file from SharePoint using File > Save As . You can control the user's ability to read or edit files in SharePoint by applying appropriate SharePoint permissions.

File Download Process	Applicable SharePoint Version	Comments
<p>Click the drop-down arrow next to the name of the file to use. Select Edit in Application > Save and Send Send as Attachment from the drop-down menu.</p> <p><i>Application</i> will be an application appropriate to the file type, such as PowerPoint, Excel, or Visual Basic.</p>	All	<p>The user can open the document and read or edit the document. The Agent monitors, reports or controls the movement of the file when the user selects Save and Send from the drop-down menu and sends the document via email.</p> <p>You can control the user's ability to read or edit files in SharePoint by applying appropriate SharePoint permissions.</p>
<p>Click Actions > Open with Windows Explorer. In Windows Explorer, Right-click on a file and select Open from the context menu. After the file opens, click File > Save and Send or Send > Send as Attachment.</p>	All	
<p>Click Actions > View in DataSheet. Right-click the file of interest from the Datasheet and select Save As from the context menu.</p>	All	When View in DataSheet is available.
<p>Click Actions > Edit in DataSheet. Right-click the file of interest from the Datasheet and select Save As from the context menu</p>	All	When Edit in DataSheet is available.
<p>Open a PDF file in a browser and save the file.</p>	All	

File Download Process	Applicable SharePoint Version	Comments
Click Actions > Open with Windows Explorer . Right-click a file to use and select Copy from the context menu. Then open a new Windows Explorer window and paste the selected file into the new window.	All	
Print a file from Share-Point.	All	

Digital Guardian does not support the following download mechanisms from SharePoint repositories or sites:

- If you use HTTPS protocol to access SharePoint, opening Windows Explorer view using Microsoft FrontPage® and downloading or copying files.
- Send a file to a document library. With this file operation, the file is moved only within SharePoint. You control file movement within SharePoint with SharePoint controls.
- Synchronize with Microsoft Outlook.
- Synchronize desktop files using Workspace or Groove.
- On SharePoint 2013, synchronize with OneDrive for Business (previously named SkyDrive Pro) or with SYNC functionality that is similar to synchronize workspace.
- Checking out files. Checking out a file does not remove it from the Share-Point repository and does not create a download event.

The next table lists common file upload mechanisms monitored and controlled by the Agent. This is not an exhaustive list of supported upload mechanisms.

File Upload Mechanism Used	Applicable SharePoint Version
Upload > Upload Document	All
Upload > Upload Multiple Documents	All
Actions > Open with Windows Explorer Open new Windows Explorer window. Copy a file from the user system and paste selected file into SharePoint Explorer Window	All
Actions > Open with Windows Explorer Open new Windows Explorer window. Drag a file from the new Windows Explore window to the SharePoint Explorer Window	All

The Agent does not support monitoring and reporting for the following upload mechanisms:

- Send a file to a document library. The file is moved only within SharePoint and file movement can be controlled by SharePoint.
- Synchronize with Microsoft Outlook.
- Synchronize desktop files using Workspace or Groove.
- On SharePoint 2013, synchronize with OneDrive for Business (previously named SkyDrive Pro) or with SYNC functionality that is similar to synchronize workspace.
- Check in. Checking in a document does not add it to the SharePoint repository and it does not create an upload event.

Writing Control Rules for SharePoint Operations

You write control rules to monitor, report and control SharePoint operations the same way you write control rules to manage other operations.

Write Rules for Managing File Download Events

The general form for rules to manage file downloads from a SharePoint site is:

```
<and>
  <or>
    <SharePoint source file path 1 >
      .
      .
      .
    <SharePoint source file path 6 >
  </or>

  <i n>
    <evtOperationType />
    <list>
      <constOpFileCopy />
      <constOpFileMove />
      <constOpFileSaveAs />
      <constOpNetTransferDownload />
      <constOpSendMail />
    </list>
  </i n></and>
```

where the six variations of the SharePoint source file path specification (path 1 through path 6 in the example) ensure that when users select one of the supported file download mechanisms to access files in the SharePoint repository, your rules will fire successfully.

The formats of the six variations are as follows. Note the differences in the specification of the name of the SharePoint Web site and in the use of forward and backward slashes. When to use forward or backward slashes is determined by the type of event.

- Network-based events, such as file downloads, require slashes (/) to specify the source file paths.
- File copy events, such as copying files from SharePoint sites using WebDAV (as employed by downloading files with Internet Explorer), require back slashes (\) to specify the source file paths.

In the example, you are managing downloads from this SharePoint folder—
<http://sharepoint.company.com/Sales/Super Secret/>.

1. Forward slashes:

```
<like expr="%/sharepoint/Sales/Super Secret/%">
  <evtSrcFilePath />
</like>
```

2. Fully qualified URL with forward slashes:

```
<like expr="%sharepoint.company.com/Sales/Super Secret/%">
  <evtSrcFilePath />
</like>
```

3. Backward slashes:

```
<like expr="%\sharepoint\Sales\Super Secret\%">
  <evtSrcFilePath />
</like>
```

4. Fully qualified URL with backward slashes:

```
<like expr="%sharepoint.company.com\Sales\Super Secret\%">
  <evtSrcFilePath />
</like>
```

5. For SSL/WebDAV accesses:

```
<like expr="%\sp2\DavWWWRoot\Sales\Super Secret\%">
  <evtSrcFilePath />
</like>
```

6. For SSL/WebDav access using a fully qualified URL:

```
<like expr="%sharepoint.company.com\DavWWWRoot\Sales\
SuperSecret\%">
  <evtSrcFilePath />
</like>
```

If you are creating rules that refer to SharePoint repository folders that have spaces in the folder names, use spaces in the URL specification as shown immediately above.

If you do not use SSL with your SharePoint sites, you do not need the SSL-related statements (numbers 5 and 6) in your rules.

Unless you encounter special circumstances in your environment, use the following operations in your download rules:

- `constOpFileCopy`
- `constOpFileMove`
- `constOpFileSaveAs`
- `constOpNetTransferDownload`
- `constOpSendMail`

Write Rules for Managing File Upload Events to SharePoint

The general form for rules to manage the uploading of files into SharePoint is:

```
<and>
  <or>
    <SharePoint upload file path 1 >
      .
      .
      .
    <SharePoint upload file path 6 >
  </or>

  <i n>
    <evtOperationType />
    <list>
      <constOpFileCopy />
      <constOpFileMove />
      <constOpFileSaveAs />
      <constOpNetTransferUpload />
    </list>
  </i n></and>
```

where the variations of the destination file paths are the same as for download rules (refer to [“Write Rules for Managing File Download Events” on page 244](#)). For upload rules, use `evtDestFilePath` in place of `evtSrcFilePath`.

Barring special circumstances in your environment, you should use the following operations in your upload rule:

- `constOpFileCopy`
- `constOpFileMove`
- `constOpFileSaveAs`
- `constOpNetTransferUpload`

If you are creating rules that refer to SharePoint repository folders that have spaces in the folder names, use spaces in the URL specification as shown in the previous section.

If you do not use SSL with your SharePoint sites, you do not need the SSL-related statements in your rules.

Example Rule 1 — Control downloads to removable drives

```
<and>
  <userFunction name= "MySharePointSites" />
  <equal >
    <evtDestDriveType />
    <constDriveRemovable />
  </equal >
  <in>
    <evtOperationType />
    <list>
      <constOpFileCopy />
      <constOpFileMove />
      <constOpFileSaveAs />
      <constOpNetTransferDownload />
    </list>
  </in>
</and>
```

Example Rule 2 — Control printing from folders

```
<and>
  <userFunction name= "MySharePointSites" />
  <equal >
    <evtOperationType />
```

```
        <constOpPri nt />
    </equal >
</and>
```

The examples use a component rule called `MySharePointSites` to define common information that you need in each rule—the SharePoint sites in your organization. In the component rule, notice that you use both the file path and URL to define your SharePoint sites in the rule. Also, that you specify the WebDAV paths to support Windows Explorer access to SharePoint sites.

Caution: You must create the component rule before you can save rules that refer to the component rule. For more about component rules, refer to [“Component Rules and Rule Functions” on page 309](#).

Here is the `MySharePointSites` component rule:

```
<or>
<like expr="%\sharepoi nt\Sa les\Super Secret\%">
<evtSrcFile Path />
</like>
<like expr="%sharepoi nt. company. com\Sa les\Super Secret\%">
<evtSrcFile Path />
</like>
<like expr="%/sharepoi nt/Sa les/Super Secret/%">
<evtSrcFile Path />
</like>
<like expr="%sharepoi nt. company. com/Sa les/Super Secret/%">
<evtSrcFile Path />
</like>
</or>
```

If you use SSL to connect to your SharePoint repositories, add this set of path definitions to your component rule:

```
<like expr="%\sp2\DavWWWRoot\Sa les\Super Secret\%">
<evtSrcFile Path />
</like>
<expr="%sharepoi nt. company. com\DavWWWRoot\Sa les\SuperSecret\%">
<evtSrcFile Path />
</like>
```


About Writing Rules for Microsoft Excel File Events Involving SharePoint

If you use Microsoft SharePoint 2007, 2010 or 2013, one consideration applies when you write rules for events that involve Microsoft Excel spreadsheets.

In some circumstances, when users download Excel spreadsheet files from SharePoint sites, SharePoint downloads an Excel viewer file—`xlviewer.htm`—instead of the actual file. When the user double-clicks the viewer file, the viewer opens in the browser and displays the actual requested file.

Whether SharePoint downloads the actual Microsoft Excel file or the viewer file depends on the setting **Default open behavior for browser-enable documents** in SharePoint.

Here is how the setting for **Default open behavior for browser-enable documents** affects file downloads:

- When **Default open behavior for browser-enable documents** is set to **Open in the Client Application**, SharePoint downloads the requested file. For example, `testfile.xlsx`.
- When **Default open behavior for browser-enable documents** is set to either **Open in the browser** or **Use the server default (Open in the browser)**, SharePoint downloads a file called `xlviewer.htm` instead of the requested file `testfile.xlsx`, perhaps `testfile.xlsx`. Double-clicking on `xlviewer` opens the file in the browser, not in Excel.

Writing Rules for SharePoint Excel File Events

To write rules to capture the way SharePoint downloads Excel files, do one of the following tasks:

- If **Default open behavior for browser-enable documents** is not set to **Open in the Client Application**, write your rules to fire when SharePoint downloads the file `xlviewer.htm`.
- Change the setting **Default open behavior for browser-enable documents** to **Open in the Client Application** for the SharePoint page. This configures SharePoint to download the actual requested file, not the Excel viewer file.

Then write your rules to fire on the file name, source, destination or other operation details.

To capture the event that downloads `xlviewer`, apply the following rule to your Agents:

```
<and>
  <equal >
    <evtOperationType />
    <constOpNetTransferDownload />
  </equal >
</and>
<like expr="%xlviewer.htm">
  <evtDestFilePath />
</like>
<like expr="%sp3.verdasys.com/dept/pm/%">
  <evtSrcFilePath />
</like>
<like expr="%xlviewer.aspx">
  <evtSrcFilePath />
</like>
</and>
</and>
```

In this rule, you cannot specify a specific document library. You can specify a sub-folder. This example specifies the `pm` sub-folder on SP3.

To Control Whether SharePoint Downloads `xlviewer`

The SharePoint configuration setting—**Default open behavior for browser-enable documents**—controls whether SharePoint downloads Excel format files (.xls or .xlsx) as the files themselves or as files to be displayed in the Excel viewer (`xlviewer.htm`). This setting applies to each SharePoint site on which you configure it.

Users who use the viewer to view Excel files cannot modify the files directly in the viewer.

Caution: If you configure SharePoint to download the actual Excel spreadsheet file rather than the viewer, users who do not have Excel will not be able to view the file.

1. Access the SharePoint site where your file is stored.
2. From the SharePoint Tools list (the gear icon), select **Site settings**. The Site Settings page opens.
3. Under Site Administration, click **Site Libraries and Lists**. The Site Libraries and Lists page opens.
4. Click **Customize “All Documents”**. The All Documents > Settings page opens.
5. Under General Settings, click **Advanced Settings**.
The SharePoint Settings > Advanced Settings page opens where you see a setting for **Default open behavior for browser-enable documents**.
6. Do one of the following:
 - Select **Open in the Client Application** to ensure the expected file test.xlsx is downloaded and opened in Excel. Best practice suggests this is the option to use for most environments and SharePoint sites.
 - Select either of the other two settings—**Open in the browser** or **Use the server default (Open in the browser)** to make SharePoint download viewer.htm to view the file test.xlsx. Use one of these options if you do not want the file downloaded and opened in Excel, or if your users do not have Excel available on their computers.
7. Click **OK** to save your changes.

Writing Classification Rules for SharePoint Operations

You can write rules that classify files users download from SharePoint. You can then use control rules to determine the allowed egress paths for the classified files. For example, you might want to prevent any file downloaded from the Super Secret folder in the Sales site from being written to a removable device.

You could write a control rule that would prevent the file from being copied or downloaded from the folder directly to a removable device. However, to prevent users from downloading the file to their computer and then copying it to the removable device, you need to use a classification rule to tag the file and a control rule that manages file egress based on the file tag.

The classification rule that you write is similar to the control rule. For the Super Secret folder of the Sales site, the classification rule is written as:

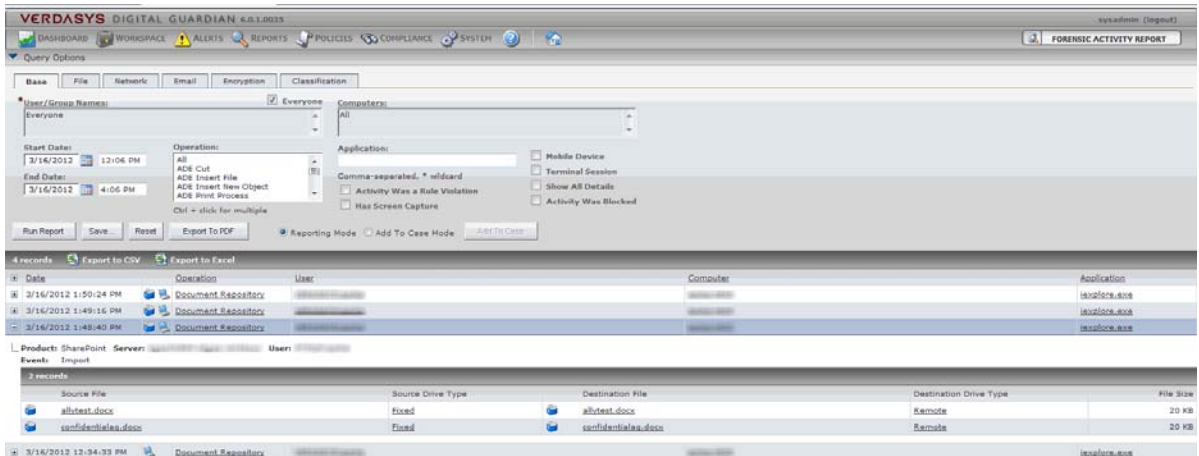
```
<or>
  <like expr="%\sharepoint\Sales\Super Secret%">
    <evtSrcFilePath />
  </like>
  <like expr="%sharepoint.company.com\Sales\Super Secret%">
    <evtSrcFilePath />
  </like>
  <like expr="%/sharepoint/Sales/Super Secret/%">
    <evtSrcFilePath />
  </like>
  <like expr="%sharepoint.company.com/Sales/Super Secret/%">
    <evtSrcFilePath />
  </like>
</or>
  <like expr="%\sp2\DavWWWRoot\Sales\Super Secret%">
    <evtSrcFilePath />
  </like>
  <like expr="%sharepoint.company.com\DavWWWRoot\Sales\
    SuperSecret%">
    <evtSrcFilePath />
  </like>
```

If you do not use SSL to connect to your SharePoint repositories, you may omit the last two entries.

The classification rule is applied whenever users move a file from SharePoint to their computer.

Viewing SharePoint-Related Events in the DGMC

DG reports SharePoint repository events in the forensic reports in the DGMC with an operation type of Document Repository, as shown here.



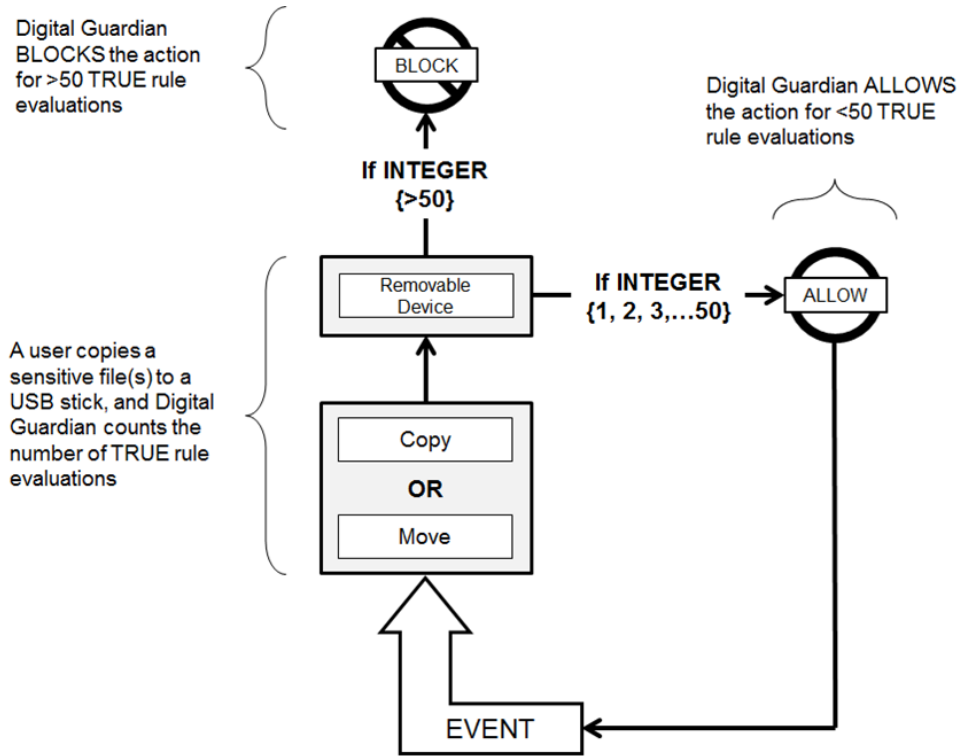
The following information applies to the SharePoint operations reported in forensic reports:

- There are two events for every download—a Network Transfer Download and a File Save As event. This results from the way in which Microsoft SharePoint implements downloads.
- If users upload classified files to SharePoint, the classification is not retained after the upload in SharePoint. This state is reflected by the classification icon on the Network Transfer Upload source file and absence of the icon on the destination file.
- When files are classified on a download from SharePoint, the classification is associated with the File Save As event. The classification icon is shown with the destination file name on the File Save As event.

CHAPTER 10 Rule Variables

Digital Guardian rules cannot respond to criteria such as the passage of time or counts of events (network transfer downloads or printing, for example) because rules do not store information about their evaluations. Rule variables enable you to enhance your current rules by adding rule evaluation information, called state, to your rules.

For example, this figure shows the rule evaluation process for a rule that prevents users from writing more than 50 files to removable drives in one session. To implement this rule, you need to use rule variables to store the file count and trigger the block action when the count exceeds 50 files. The basic rule system in DG cannot accomplish this use case because it cannot store the file count.



Introducing Rule Variables

Rule variables extend the rule programming language with features similar to the variables in advanced programming languages. Using rule variables, you can define information that persists between rules and between executions of a rule. With the addition of variables to rules, you can do things such as:

- Influence the behavior of other rules by the results of evaluating a rule.
- Prevent the same code from running many times during rule evaluation.
- Count user operations and use the count to monitor and limit how many times a user is allowed to perform actions, such as printing files or downloading files.
- Track time to monitor or limit user operations based on the time that has elapsed from a starting point or operation.
- Limit the number of user prompts generated by triggering the same rule for repeated policy infractions.

Rule Variable Use Case Examples

With rule variables, you can fulfill use cases that you cannot accomplish without them. Here are examples of use cases that require rule variables:

- At a defense contractor, printed copies of system designs are a security risk because they cannot be tracked after they have been printed. For that reason, the ability to print design documents is restricted to a few individuals.
- Some employees are permitted to log into the corporate network from their home computers. The company wants to prevent those employees from printing design documents when they are logged onto the network remotely. To provide security for printed documents, the corporate security policy requires that users store all printed design documents in a locked storage cabinet with limited access to the cabinet.

To support the corporate security policy on printing, the organization created a rule variable to track the network adaptor through which the Agent is con-

nected. If the network adaptor is not the fixed LAN card, the Agent blocks print operations.

In another example, an engineering company has a policy to control the amount of data users can copy to removable media, such as USB external drives and CD/DVD drives. They created a rule variable to track the amount of data a user has copied to removable drives in one day. Another rule evaluates the first variable, and when the value copied reaches 50MB, the Agent blocks all further copying of data to the drives.

When the user triggers the rule the next day it resets the variable and the user can start copying files again.

Example 1 — Rule Counting Operations or Events

This rule uses an integer variable—TotalCopiesToRemovable—to count the number of files a user copies to a removable drive. When the count of the number of files reaches 50, the rule prevents further copies.

```
<and>
<greaterThan>
  <vari nt name="Total Copi esToRemovabl e" scope="gl obal " />
  <i nt val ue="50" />
</greaterThan>
<add>
  <vari nt name="Total Copi esToRemovabl e" scope="gl obal " />
  <i nt val ue="1" />
</add>
<i n>
<evtSrcFileExt/>
  <list>
    <stri ng val ue="doc" />
    <stri ng val ue="xl s" />
    <stri ng val ue="ppt" />
    <stri ng val ue="txt" />
    <stri ng val ue="docx" />
    <stri ng val ue="xl sx" />
    <stri ng val ue="pptx" />
    <stri ng val ue="j pg" />
    <stri ng val ue="pdf" />
    <stri ng val ue="rtf" />
    <stri ng val ue="mpp" />
```

```

        <string value="vsd" />
        <string value="tmp"/>
    </list>
</in>

<equal >
    <evtOperationType/>
    <constOpFileCopy/>
</equal >
<equal >
    <evtDestDriveType/>
    <constDriveRemovable/>
</equal >
</and>

```

The rule adds 1 to the total in TotalCopiesToRemovable each time the user copies a file that triggers the rule. When the value of TotalCopiesToRemovable exceeds 50, the rule blocks future file copy operations.

Using the example rule, you can enhance the rule by adding the stipulation that user copy and move operations to a removable device will be blocked only after a specified number of copy or move operations have occurred.

To achieve the operation count requirement, declare an integer variable that tracks the number of file copy and move events. The declaration should appear at the top of the existing control rule.

To enforce the new requirement, create an evaluation statement that measures the value of the variable and includes a greaterThan expression. You want the users' action to be blocked only if they have already exceeded the maximum number of operations, so the evaluation of the variable should appear above the variable declaration in the rule. To restrict the user to 50 file copy events, you want the rule to update the number of copy events before you evaluate whether the count is greater than 50.

Example 2 — Rules Tracking Time Spans

Rule variables enable you to track the passage of time. In this example, the rule, when triggered, identifies classified files that are older than 30 days. Rule vari-

ables enable you to compare the file's last modified date with the current date to determine its age.

After a file operation triggers the rule, four processes comprise the rule process for determining the file's age:

- A. Get the time and date when the file was modified most recently (evtSrcFile-ModifiedTime).
- B. Get the current date and time (agentCurrentDateTime).
- C. Subtract the two dates to determine the difference.
- D. Evaluate the result of the subtraction operation to determine the age of the file and whether it is older than 30 days.

Example 3— Control Rule Application

Control rules can reference variables to regulate user activity after the variable value meets a specified criterion.

For instance, you could create a rule which triggers only when a user copies more than one gigabyte of data to a removable USB device within a 12 hour period. As long as the amount of data tracked by the variable is less than one gigabyte, the user can continue to copy files.

Another application for rule variables might be to reduce the effect of social media Web sites on employee productivity. A company has realized that restricting social media sites entirely increased the time employees spent on mobile devices, such as smart phones and tablets. Instead, the company created a variable to track the number of times an employee connects to a given social Web site URL. The rule and variable allow the first visit an employee makes to a social media site. The rule records subsequent visits to the URL and generate a warning that the employee is exceeding the allowed daily number of visits as set by the corporate network usage policy.

You can also use variables to enhance existing control rules. For instance, suppose you have a rule that limits access to classified files to a few individuals. You could enhance this rule by referencing another variable used in a rule that checks the users' network connections. If the users' network adapters indicate they are connected over VPN, they might not be allowed to access classified file at all.

Example 4 — Classification Rule Application

You can use rule variables to add criteria on context classification that the existing rule properties available in DG cannot measure.

Another challenging use case is to classify data that users save from specific Web pages via ADE copy or print screen operations. You could create a variable to identify and store the specific page as the value. A classification rule might then reference the variable's value, and if it is a location known to store sensitive data, the Agent would classify the pasted or buffered data.

Creating Rule Variables

When you declare a rule variable in a rule, you are instructing the Agent to gather information and store that information as a value until another rule or function evaluates the value.

After you declare variables, reference them like any other rule property. Use any applicable relational operator such as `equal`, `greaterThan`, or `lessThan` to evaluate the variable as long as the value remains in scope.

To define a rule variable in a rule, supply values for these arguments:

- **Variable Name**—an arbitrary text string that might help you remember what the variable does or how you use it. For details about naming variables, refer to [“Rule Variable Names” on page 266](#).
- **Variable Type** — the type of content in the variable, such as integer or string or array. For details about types of variables, refer to [“Rule Variable Types” on page 263](#).
- **Variable Scope** — specifies the reach and lifetime of the variable. For programmers, scope will be familiar from local and global variables in C or C++. For details about applying scopes to variables, refer to [“Rule Variable Scopes” on page 267](#).
- **Variable Initial Value or String** — specifies the starting value for the variable. It might be a numeric value, a Boolean or a text string.
- **Logical or mathematical operation** — the operation to use with the variable

You supply the arguments in a syntax to define the variable or to refer to the variable:

```
<type name="var_name" scope="scope_type" />
<variable_name />
```

or

```
<type name="var_name" scope="scope_type" />
<property />
```

or

```
<type name="var_name" scope="scope_type" />
<type value="value" />
```

When you add these variable definition statements to the top of a rule—the traditional location, though not required—you surround the statements with the operator that applies to the variable, such as Add. The result might resemble the following rule code that defines three variables—one integer variable and two string variables:

```
<and>
  <add>
    <variable name="MyVariable3" scope="process" />
    <int value="1" />
  </add>

  <set>
    <variable name="MyVariable2" scope="process" />
    <string value="true" />
  </set>

  <set>
    <variable name="MyVariable1" scope="event" />
    <eventSrcFilePath />
  </set>
</and>
```

Rule Variable Types

The first part of a rule variable definition specifies the variable type. The type argument is the first entry in the syntax and determines what follows in the definition.

Depending on the data you are trying to capture, you must assign an appropriate type when you declare the variable. For instance, to count the number of file copy events to a removable device a user performs, you would declare an integer variable.

Variable type determines the relational operators that can be used to evaluate the variable. Therefore, in the example of an integer-type variable declared to count file copy events, appropriate operators are equal, greaterThan, lessThan, and list.

Integers are whole numbers, so the relational operators evaluate integer variables as numerical values and their relationship with a value you define, such as greater than 50 or equal to 10.

All rule variables are typed and type conversion is automatic (refer to [“Variable Type Conversion” on page 286](#)) for more information).

Note: When you evaluate a rule property, such as a file path, that contains a wildcard, you must use the like operator instead of equal. The same concept applies to the string-type variables you declare. If the variable value does or might contain a wildcard value, use the like operator instead of equal.

Boolean variables refer to the state of an event, and the operator must evaluate to either true or false. While Boolean values are presented as true or false, computer logic equates true to the integer value 1, and false as the integer value 0. Therefore if the variable value not equal to 1, the Boolean variable evaluates to false.

After a variable has been declared as a certain type, you can convert it to another type without changing the declaration statement.

All of these variable can hold one or more values of their type, so you can handle all of them as arrays.

Note: Any variable with global or persistent scope can be referenced across multiple rules.

Type	Variable Argument	Supported Platforms	Applicable Relational Operators	Description
Integer	varint	Windows, Linux, macOS	Equal, Greater Than, Less Than, List	Count and compare the number of times an event occurs.
8-bit Integer	varint8	Windows, Linux, macOS	Equal, Greater Than, Less Than, List	Count and compare the number of times an event occurs.
16-bit Integer	varint16	Windows, Linux, macOS	Equal, Greater Than, Less Than, List	Count and compare the number of times an event occurs.
64-bit Integer	varint64	Windows, Linux, macOS	Equal, Greater Than, Less Than, List	Count and compare the number of times an event occurs.
String	varstring	Windows, Linux, macOS	Equal, Like, List, Regex	Associate one event evaluation with a different one, as between rules.
Boolean	varbool	Windows, Linux, macOS	Equal, Greater Than, Less Than	Set and test the event state. Either True or False (1 or 0).
IP Address	varipaddress	Windows		32-bit IPv4 address
16-bit Integer	varmd5	Windows		16-bit MD5 hash values
20-bit Integer	varsha1	Windows		20-bit SHA1 hash values
32-bit Integer	varsha256	Windows		32-bit SHA256 hash values

Pairing Rule Variables With Types

One of the most important things to know about rule variables is which variable type to use to store values from DG rule properties, such as `evtSrcFile`, `curProcessSaveAsActive` or `evtDestFile`. In this table, you see some representative rule properties and the suggested variable type to use. Note that the table is not exhaustive and the type is suggested. In many cases with integer variables, you can use a larger or smaller variable type as you require. You can use the table information to make decisions about the variable type to use with rule properties that are not listed in the table.

Example Rule Property	Suggested Variable Type to Use
<code>agentAdapterCount</code>	<code>varint</code>
<code>agentCurrentDateTime</code>	<code>varint64</code>
<code>agentLastServerComm</code>	<code>varint</code>
<code>anyProcessVaulted</code>	<code>varbool</code>
<code>curProcessSaveAsActive</code>	<code>varbool</code>
<code>curProcessVaulted</code>	<code>varbool</code>
<code>evtDestFilePath</code>	<code>varstring</code>
<code>evtDestFilePolicyTag</code>	<code>varstring</code>
<code>evtDestFileSize</code>	<code>varint64</code>
<code>evtDestProduct</code>	<code>varstring</code>
<code>evtDestPProductID</code>	<code>varstring</code>
<code>evtLocalPort</code>	<code>varint16</code>
<code>evtRemotePort</code>	<code>varint16</code>
<code>evtSrcFileModifiedTime</code>	<code>varint64</code>
<code>evtSrcFilePath</code>	<code>varstring</code>
<code>evtSrcFileSize</code>	<code>varint64</code>

Note: Rule variables do not support MAC Address types. You cannot set a variable with the content of that rule property, such as `agentGatewayMac`.

All the rule properties in the Digital Guardian library have a variable type set and have a logical relationship with specific operators.

Integer type variables extend to accommodate varying integer lengths or sizes.

- `varint` variables support integer value lengths from 8 bits to 454 bits. Use `varint` for relatively low numbers such as file count, adapter count, or counts of user operations.
- `varint8` supports 8-bit numbers. This is essentially the same as `varbool`, which is also an 8-bit variable. Many low value numbers, such as file counts or operation counts, work with this variable type.
- `varint16` supports 16-bit numbers such as port numbers. While some frequently-used ports, for example 80, 21 and 22, are supported using `varint`, you might not know the size of the value returned when you are trying to identify a remote port. Using the larger integer variable type supports all possible returned port values.
- `varint64` variables support very large integer values, including file sizes and current time. (The Agent calculates current time in the number of seconds passed since a date in the 1600's).

Rule Variable Names

Variable names are arbitrary. Use names that will be meaningful to you and all others who support DG in your environment.

- The maximum length for rule variable names is 64 characters.
- Variable names are case-sensitive. `MyVariable` is not the same as `myvariable`.
- Rule variable names cannot use any special characters that would interfere with XML. The valid characters are:
 - A-Z, a-z
 - 0-9
 - Dollar sign (\$), underscore (_) and dash (-)

- Rule variable names should not start with a number.
- Rule variable names should not contain spaces.
- Rule variable names should not contain these characters:
 - Colon (:), semicolon (;), hash (#), plus (+)
 - Left and right parentheses (()), percent (%), at sign (@), equal (=)
 - Vertical bar (|), left and right square brackets ([]), slash (/), question mark (?), asterisk (*), backslash (\), tilde (~)
 - Exclamation mark (!), caret (^), period (.), or comma (,)
- Rule variable names cannot contain these characters:
 - Single or double quotes (' or "), ampersand (&)
 - Left or right angle brackets (< >)

You can use the same variable name more than once with different scope values or different type declarations. To limit confusion, best practice suggests that you should not use duplicate variable names. If you use the same variable name with a different scope, DG creates a unique variable with the same variable name so that myVar with global scope is not the same variable as myVar with process scope.

However, DG does not perform error checking to prevent you from using the same variable name more than once. If you create a rule variable with a duplicate name, DG does not return an error. If you use the same variable name within the same scope, you will overwrite your existing variable. Any rules that reference the original variable might start behaving unexpectedly.

Rule Variable Scopes

If you are familiar with programming, the concept of variable scope will not be new. Rule variables have scope much as variables in C programs have scope.

Each scope represents a Digital Guardian object type that holds a list of variables. The lifetime of the variable is thus tied to the lifetime of the object.

In DG, you define rule variables with one of the following scope specifications:

Scope Type	Scope String	Description and Lifetime
Destination File	destFile	The variable is created for the destination file in an event after the file is opened. A short time after the user closes the file, the variable goes out of scope and is deleted.
Event	event	The variable applies to the event. It exists as long as the event is being processed. After the Agent logs the event, the variable and the event are deleted.
Global	global	The variable lives until the end of this computer session and can be accessed from any process or thread and from any event. Rule variables do not persist when you reboot the host computer. Any rule running on the computer can access the variable, performing the specified operation on the contents of the variable. This creates a global list that the rule driver holds. Global rule variables persist until the user the session on the Agent machine. Note that the variable exists until the reboot process, so it can survive termination without reboot. Rebooting the computer removes global variables. This is the default scope.
Parent Process	parentProcess	The variable applies to the parent process of the process that triggered the rule. The variable exists as long as the current child process exists. Thus the variable scope can extend past the lifetime of the parent process.
Persistent	persistent	The variable has global scope and survives rebooting the computer.

Scope Type	Scope String	Description and Lifetime
Process	process	The variable applies per process and exists for the life of the process. Two instances of a process, such as winword.exe, could each hold separate and different values for the same named variable. This is essentially having two variables with the same name and type, but different processes (hence different variables because the definitions are not identical). This might be particularly helpful in Web browsers.
Source File	srcFile	The variable is available after the user opens a source file and triggers a rule. A few minutes after the user closes the file, the variable goes out of scope and is deleted.
Source Process	sourceProcess	The variable is available only to ADE events. The variable persists in the process entry list for the source of the ADE event.
Source Parent Process	sourceParentProcess	The variable is available only to ADE events. The variable persists in the parent process entry list for the source of the ADE event.
Thread	thread	The variable applies to the thread. The variable exists for the life of the thread.

Variables exist only for the time period defined by their scope. For instance, suppose you want to record every URL visited. After a URL is recorded, that information is no longer necessary, so that information would have a shorter lifespan. To record the number of files downloaded from a specific site, you would want a longer term variable to track the site so that you collect the total URL count for the life of the browser process.

You determine the lifespan of a variable when you set the variable scope. The scope you specify defines the lifespan of the declared variable and which rules can access the variable. If you do not provide a scope for a variable, it defaults to global.

For example, for a process scope variable, Digital Guardian constructs a Process Entry object when an executable runs on the system. The object is added to its process cache, and attaches an empty list specifically for variable support.

When an executable is terminated, the corresponding Process Entry is removed from the cache. If no other object is referencing the Process Entry object, Digital Guardian will destroy it.

Lifetimes of Rule Variables

How long a rule variable persists during a rule evaluation or session or event depends on the scope assigned to the variable.

- Global scope is the longest time frame available. For global scope variables, the rule driver holds the list of variable values with global scope. The variables persist until the Agent is rebooted. In fact, variables survive even when you terminate the Agent.
- Process scope variables persist for the lifetime of the current process. When the process or application terminates, Digital Guardian deletes the variable.
- Thread scope can act as a bridge between the process and event scopes. A thread can exist longer than a specific event, but not as long as the process. As long as the target thread is active, the variable data persists.
- Event scope variables exist while the event is being processed. After the event is logged, the variable is deleted with its variable.
- Parent process exists as long as the child process exists. This means the variable's scope can exist beyond the lifetime of the parent process.
- Source process scope is accessible only to ADE events. The variable persists in the process entry list for the source of the ADE event.
- Source parent process is also accessible only to ADE events. The variable persists in the parent process entry list for the source of the ADE event.
- For a source file scope, the variable is available when a source file is opened. Shortly after the file is closed, the variable goes out of scope and is deleted.

- When scope is set to destination file, the variable is created for the destination file of an event, after the file is opened. Shortly after the file is closed, the variable goes out of scope and is deleted.

Changing the scope of a variable can have a dramatic affect on the behavior of the variable and variable evaluation results.

Rule Variable Arrays and Lists

As in most programming languages, DG rule variables can be single values or arrays (lists). Working with arrays is one of the most powerful features of rule variables. With variables as arrays, you can do things like:

- Create lists dynamically
 - Lists of URLs that an employee visits
 - Lists of the addresses to which employees send emails, inside or outside your domain
- Test property values against the entries on lists
 - Block users from visiting specific sites or domain too often
 - Justify sending classified information in emails to more than two addresses

Creating Rule Variable Arrays — AsArray Statement

String variables are arrays automatically. To change this behavior, specify that configuration with the `asArray` statement when you apply the operation to the variable. For example:

```
<add asArray="false">  
  <varstring name="myName" scope="global" />  
  <string value="Sanderson" />  
</add>
```

The `asArray = "False"` statement defines the variable `myName` to be a single value, not an array. If you add a new string to `myName`, it appends the new value to the existing value.

The `asArray` statement takes `true` or `false` as input arguments. Integer types are not arrays unless you add the `asArray = "true"` argument to the operator command syntax.

For more information about working with arrays, refer to the following sections:

- [“Adding Variables” on page 287](#)
- [“Subtracting Variables” on page 293](#)

To create a list of integers, for example, you could use this rule code to list three integers in a variable array:

```
<add asArray="true">
  <varint16 name="MyInt16List" scope="global" />
  <int16 value="32" />
</add>

<add asArray="true">
  <varint16 name="MyInt16List" scope="global" />
  <int16 value="10" />
</add>

<add asArray="true">
  <varint16 name="MyInt16List" scope="global" />
  <int value="5" /> <!-- Note Type Conversion int to int16-->
</add>
```

The `asArray="true"` statement defines `myInt16List` as an array instead of a single value. These commands create a list of three values in `MyInt16List`:

MyInt16List Content

5

10

32

Notice that the order of the values is from the bottom of the rule up—5, 10, 32. 5 is the first entry into the array, 32 is the last. With these entries on the list, you can test for the values in the list.

For an example that uses strings, the following code creates a variable `stringList1` that contains unique values only, no duplicates. Remember that DG treats string variables as arrays automatically. You do not have to specify the `asArray` argument.

```
<add>
  <varstring name="stringList1" scope="global" />
  <string value="mynewstringvalue" />
</add>
```

One more example uses a string to collect email address:

```
<add>
  <varstring name="all Mailed Recipients" scope="global" \>
  <evtMailed Recipients />
</add>
```

When you are working with variable string or integer arrays, keep these points in mind:

- DG processes arrays in last in, first out (LIFO) order. By default it adds new elements to the end of an array. This makes it possible for you to track things like the 20 most recent events in a list of events. You can change this with the `mru` attribute that you use with `<add>`. Refer to [Control How DG Adds Values to Arrays](#) for more information.
- Strings are lowercase unless you include the `preserveCase = "true"` argument.
- DG returns elements starting with the end of the array. The Agent returns the last element in the array first.
- Array size is limited by default to 2000 entries. You can change that each time you add to the array using the `<add size="value">` syntax.
- Pay attention to the scope of your variables and the data sources you use to ensure you do not encounter problems with large arrays. For example, you should not create an array that stores the path to every file that gets opened on a machine.
- By default, arrays hold only one copy of a value. Adding the same value to the array has no effect. To let your variable arrays hold multiple copies of the same value, add the value with the `<add unique="false">` syntax.

Control How DG Adds Values to Arrays

As noted, DG uses LIFO ordering when you add elements to an array. You can change the ordering if your use case requires it. To change this ordering, use the `mru` attribute when you add elements to the array. Attribute `mru` is a Boolean:

- `mru = True` directs DG to add new array elements to the beginning of the array. Also, DG indexes the array from the beginning of the array. This

affects the index value for the array elements. This is the default condition and this is what happens if you leave out the `mru` attribute.

- `mru = False` directs DG to add new array elements to the end of the array, supporting Last In, Last Out (LILO) ordering. Also, DG indexes the array from the end of the array in this configuration, affecting the index values of the array elements.

Example 1 — Attribute `mru = True`

This short rule creates an array `MyPathArray` where DG adds new elements to the end of the array.

```
<add mru="false">
  <varstring scope="global" name="MyPathArray"/>
  <evtSrcFilePath/>
</add>
```

The attribute `mru = "True"` tells the rule engine to add a new string to the end of `MyPathArray`—LIFO processing. By default, the rule engine uses LIFO.

The following set of commands initializes `MyPathArray` as follows:
`MyPathArray = { "t3.txt", "t2.txt", "t1.txt" };`

```
<add>
  <varstring scope="global" name="MyPathArray"/>
  <varstring name="t1.txt"/>
</add>
<add >
  <varstring scope="global" name="MyPathArray"/>
  <varstring name="t2.txt"/>
</add>
<add >
  <varstring scope="global" name="MyPathArray"/>
  <varstring name="t3.txt"/>
</add>
```

These are equivalent to using `mru = "True"` in rules.

Example 2 — Attribute mru = False

In contrast to example 1, these commands initialize the `MyPathArray` as follows, which represents LIFO processing: `MyPathArray = {"t1. txt", "t2. txt", "t3. txt"};`

```
<add mru="false">
  <varstring scope="global" name="MyPathArray"/>
  <varstring name="t1. txt"/>
</add>

<add mru="false">
  <varstring scope="global" name="MyPathArray"/>
  <varstring name="t2. txt"/>
</add>

<add mru="false">
  <varstring scope="global" name="MyPathArray"/>
  <varstring name="t3. txt"/>
</add>
```

Adding Rule Variables to Rules

Rule variables and the operators that you use with them can be used anywhere in a rule. Most of your rules will evaluate a few criteria before declaring or modifying a variable. Variable declarations usually appear toward the top of the rule XML code and are therefore evaluated last.

When you implement rule variables, continue to place the basic rule components that are most likely to return a false evaluation toward the bottom of the rule structure to reduce unnecessary evaluations in your rules.

For example, to count the number of file copy events to removable drives a user performs, you would evaluate whether the user is performing a file copy to a removable drive before you add a count to the variable. Proper placement of the operators can be challenging; it might take a few tests before you get the results you want from your variables.

Rule variables should be declared at the top of the rule, after all other criteria have been met. Rule variable evaluations can be closer to the top of the rule, or from other rules.

When evaluating the value of a rule variable, the evaluation statement can go anywhere in the rule. You cannot evaluate a variable you have not defined, so you need the rule engine to evaluate the variable after it has declared or updated the variable's value.

Referring to Variables in Other Rules

After you have declared a variable, the current variable value can be referenced like any other property. You can reference the variable value from within the same rule, or using another rule. Also, the same variable can be referenced from multiple rules (as long as the scope persists).

For diagnostic purposes, variable operations will appear in the Agent's log file (dg.log) when the Agent is in either debug or verbose logging mode. Placing a test Agent in debug mode while you are creating or modifying rule variables can help you validate the rule operation. After you validate the rule operation, reset the Agent's log level.

You must specify the variable's type in the declaration statement when you create a variable. A variable's type determines which relational operators you can use to evaluate its value.

Rule Evaluation Logic With Variables

Rules are read from the bottom up by the rule engine. For efficient rule processing, express the rule components that are most likely to return a false value closer to the bottom of the rule logic so they are evaluated first.

```
<and>
<add>
  <variable name="MyVariable2" scope="process" />
  <variable value="1" />
```

```
</add>
<! This clause is evaluated if the two previous clauses -->
<! are true -->

<like>
  <varstring name="MyVariable1" scope="process" />
  <varstring="true"/>
</like>
<! This is evaluated only if the first clause is true -->

<equal>
  <varstring name="MyVariable1" scope="event" />
  <varstring="evtSrcFilePath" />
</equal>
<! This is the first clause evaluated -->
</and>
```

In this sample rule, the value of “MyVariable1” is being evaluated to determine whether it is equal to “evtSrcFilePath.” If the variable value is equal to “evtSrcFilePath”, the rule engine moves on (up) to the next operator. If “MyVariable1” is any other value, the test evaluates to false and the rule engine exits the rule. MyVariable1 remains for the duration of the event as specified by having scope = event.

Note: If the first operation returns a null value because there is no value to feed to the variable, the rule engine evaluates the operation as true and continues to the next criterion.

In the second operation, the variable “MyVariable1” is evaluated to determine if it contains the string “true.” With scope = process, MyVariable1 remains for the duration of the process. When the process is terminated, the variable is cleared. Notice that while the variable name is the same, the scope is not, so this is a unique variable.

In the third operation, after the other two criteria have been met, the rule adds one to the variable “MyVariable2.” Like MyVariable1, MyVariable2 remains for the duration of the process.

Operators for Rule Variables

DG provides three general types of operators:

- Logical operators — These specify the relationship between two or more rule properties. For example, suppose you need multiple rule criteria to evaluate to true before you block the user's operation. You would use the `<and>` operator to include all of the criteria.
- Relational operators — These evaluate one or two symbolic constants or user defined values. They evaluate strings, lists of property values, or compare one expression to another. If you wanted to evaluate whether or not the destination drive type is removable, you would use the `<equal>` operator to evaluate the drive type.
- Variable operators — These are language elements that create and modify variables. Any variable operator can declare a variable. Like all language elements, variable operators are case sensitive, so entering operators with the correct spelling and case is critical. However, the values provided for evaluation are not case-sensitive.

Operator	Description
set	Sets a rule variable to a specific value or values
add	Adds or creates an operand to a rule variable
sub	Subtracts an operand from a rule variable

Setting Variables With Set

Using the `set` operator to declare a variable creates the variable and sets the initial value to what you specify in the `<set>` statement. If the rule containing the `set` operator is triggered again, whatever value is presently stored will clear and reset to 0.

You might use the `set` operator to declare a variable that will store the destination drive type of a user's file move operation. The next time a user performs a file move that meets the rule's criterion, the previous value (fixed drive type) will be reset and the new value (removable drive type) will be stored instead.

Adding Variables With Add

The add operator can be used to create and add to the value of the variable. If a rule containing the same <add> statement is triggered again, the value of the variable will not be cleared and reset to 0. Instead, the present variable value remains and will be added to. In other words, the variable to be added to or “operated on” is in the left_operand node.

For example, to count the number of files copied to a removable drive, use the add operator to both declare and increment the variable operand each time the user performs a file copy operation for which the destination drive type is removable.

To continue the example, after you have declared the variable (i.e. fileCount), counting the number of file copy events to a removable device, you can reference that variable in another rule. For example, if fileCount is greaterThan 50, block the file copy operation.

Subtracting Variables With Sub

Sub operates on a previously declared variable by subtracting an operand from the variable. The value to be operated on, or subtracted from, is in the left_Operand node. Sub removes a stored value, typically from an array.

If you have a string variable declared that contains a user’s full email address (for example jfusco@widgets.com), you might use the sub operator to subtract the user’s ID (jfusco), leaving the domain name (@widgets.com) as the value stored in the variable.

Using Sub to Declare a Rule Variable

This code snippet declares a string rule variable named Var-FilesNeedClassification with global scope:

```
<sub>
<varstring name="Var-FilesNeedClassification" scope="global" />
</sub>
```


Using Sub to Remove Information From an Array

Suppose you have an array of web sites from a user’s browsing history. The array might contain operating system software company sites—Microsoft, Apple and Red Hat—among the list of visited web sites:

RemoveFromArray Array Indices	Web Site
0	www. mi crosoft. com
1	support. mi crosoft. com
2	www. appl e. com
3	www. redhat. com
4	www. di gi tal guardi an. com
5	www. ubuntu. com

You could use the sub operator to remove the OS software sites from the RemoveFromArray rule variable with this rule snippet:

```

<sub>
  <varstri ng name=" RemoveFromArray"  scope="gl obal " />
  <stri ng val ue=" www. mi crosoft. com" />
</sub>
<sub>
  <varstri ng name=" RemoveFromArray"  scope="gl obal " />
  <stri ng val ue=" www. appl e. com" />
</sub>
<sub>
  <varstri ng name=" RemoveFromArray"  scope="gl obal " />
  <stri ng val ue=" www. redhat. com" />
</sub>

```

After the rule engine evaluates a rule that has this XML, the OS web sites would no longer appear in the array.

RemoveFromArray Array Indices	Web Site
0	support.microsoft.com
1	www.digitalguardian.com
2	www.ubuntu.com

If your array contains multiple instances of the same item, such as the same URL in our example, sub removes only one instance for each rule evaluation operation.

Rules with Variable Operators and No Other XML

In some cases, you might write an <and>, <or>, or <eval> section in a rule that contains only the <set>, <add>, or <sub> variable operators and no other statements. For example, you might write the following XML to create an array in a component rule:

```
<and>
  <sub>
    <varstring name="Whitelist" scope="event" />
    <string value="world" />
  </sub>
  <add asArray="true">
    <varstring name="Whitelist" scope="event" />
    <string value="hello" />
  </add>
  <set>
    <varstring name="Whitelist" scope="event" />
    <string value="mystring" />
  </set>
</and>
```

Typically, you enclose rule statements in an <and>...</and> pair, as in this component rule example. This component rule is used by another rule, so you want the rule to evaluate to True so that it does not exit immediately. However, none of <set>, <add> or <sub> return True, the <and> statement returns False, and the rule fails.

To ensure that the rule succeeds, add a <return> statement at the top of the rule:

```

<and>
  <return>
    <bool value="true" />
  </return>
  <sub>
    <varstring name="Whitelists" scope="event" />
    <string value="world" />
  </sub>
  <add asArray="true">
    <varstring name="Whitelists" scope="event" />
    <string value="hello" />
  </add>
  <set>
    <varstring name="Whitelists" scope="event" />
    <string value="mystring" />
  </set>
</and>

```

The `<return>` statement causes the component rule to cease evaluating and return True to its calling rule. Note that if the `<return>` statement is placed in a normal rule and the statement is encountered during rule execution, the rule exits immediately and the boolean result that you specify is returned as the rule's result.

As an alternative to using a `<return>` statement, you can use an `<equal>` statement at the top of the rule to make the `<and>` statement return True. Repeating `<bool value="true" />` provides a True-True comparison that allows the rule to continue:

```

<and>
  <equal >
    <bool value="true" />
    <bool value="true" />
  </equal >
  <sub>
    <varstring name="Whitelists" scope="event" />
    <string value="world" />
  </sub>
  <add asArray="true">
    <varstring name="Whitelists" scope="event" />
    <string value="hello" />
  </add>
  <set>
    <varstring name="Whitelists" scope="event" />
    <string value="mystring" />
  </set>

```

</and>

Naming Rule Variables

When you are creating and naming rule variables, best practice is to design a naming convention before you begin. Variable names should indicate the kind of information stored in the variables and be logical for administrators who will be referencing the variables in their rules. Variable names should be meaningful and unique.

- Variable names can be up to 64 characters in length. While the 64-character length offers ample opportunity for descriptive names, you might not want to type a 64-character name numerous times.
- Spaces are not supported in variable names. You can use underscores and dashes to separate words or abbreviations. Although underscores and dashes are technically considered special characters, they are the only special characters that are supported in variable names.
- Variable names are case-sensitive. When you are creating your variable naming convention (and you should have a convention), create a standard which takes case into consideration. For example, all Digital Guardian rule properties use the following convention: they start with all lower case, then each word is capitalized, such as `evtSrcDriveType`. When rule writers are referencing rule properties, they know the naming convention and can therefore type property names without looking up the rule property spelling in the implementation guide.

Because variables are objects, like DG rule properties, consider using the same naming convention as the Digital Guardian rule properties.

Variable Syntax

After you have determined your variable's type, name and scope, you can begin constructing the variable declaration statement. Like the syntax of any logical or relational operation, the variable operation must follow a specific syntax.

```
<set>
  <varint name="MyVariable" scope="process" />
  <int value="5" />
</set>
```

The example uses the `<set>` operator to declare the variable. The statement begins by identifying the variable type as a small integer—`varint`. The variable name is `MyVariable` and the scope is set to `process`.

The set operator initializes `MyVariable` to the integer value 5. This value remains in the process entry list until the process is terminated.

```
<set>
  <varint name="MyVariable" scope="process" />
  <int value="5" />
</set>
```

When the user executes the operations, the rule engine processes the logic from the bottom up. Whatever appears in the value statement will be read and written as the variable's stored value for the duration of the variable's scope.

Using Data in Rule Variables

Variable data can represent a single element or a list of elements of that type. For example, you can provide a single file name to the string variable “`userFileName`”, or you can provide a list of all file names that a user touches while they are using a particular application without overwriting the initial file name.

Property Values in Rule Variables

Content can also be a property or another variable. Instead of explicitly defining the variable value, you can specify a rule property. The event details provide content or data for the rule property object. The rule property's content can be transferred to a variable by specifying the rule property name.

```
<set>
  <varstring name="MyStringValue" scope="event" />
  <evtSrcFilePath />
</set>
```

In the example, user activity has triggered a rule that contains this variable declaration. The source file path for the event is pulled from the event details. Whatever the path is, MyStringVariable is set to that path as a string, such as c:\documents.

Variable Type Conversion

Converting variables between types is provided automatically. For example, when you assign numeric value to a string variable, the rule system converts the numeric to a string. This example does that conversion with the Port Number value:

```
<set>
  <varstring name="PortAsString" scope ="process" />
  <evtRemotePort />
</set>
```

evtRemotePort returns an integer value. Setting varstring to the value returned by evtRemotePort converts the value to a string. In this case, if the port number is 80, the string PortAsString gets the string "80."

This table details the conversions that the rule system does between the source type and destination type.

Destination	varstring	varint	varint16	varint64	varbool	varint8
Source						
varstring	X	varint	varint16	varint64	varbool	varint8
varint	varstring	X	varint16*	varint64	varbool	varint8*
varint16	varstring	varint	X	varint64	varbool	varint8*
varint64	varstring	varint	varint16*	X	varbool	varint8*
varbool	varstring	varint	varint16	varint64	X	varint8
varint8	varstring	varint	varint16	varint64	varbool	X

X indicates that conversion does not apply.
* indicates that the conversion causes significant loss of bits and precision.

Similarly, you can convert numeric values between sizes. This set of commands converts the value contained in `PortAsInt` from a 16-bit integer to a 64-bit integer:

```
<set>
  <varint64 name="PortAsInt" scope="process" />
  <varint16 name="PortAsInt" scope="process" />
</set>
```

After executing this command set, `PortAsInt` contains a 64-bit version of the 16-bit value it held.

Because variables are objects like rule properties, you can transfer a variable object's value to another variable. For instance, suppose variable `evtAdapterName` has a value equal to `agentAdapterCount`. If the Agent has two adapters, the value currently stored for `evtAdapterName` is 2.

The value for `evtAdapterName` can be transferred to another variable.

```
<set>
  <varstring name="AgentAdapterList" scope="global" />
  <evtAdapterName />
</set>
```

In this example, `AgentAdapterList` gets the value from `evtAdapterName`. Digital Guardian reserves objects such as rule properties, preventing you from directly manipulating or assigning data to them. Because DG owns the objects, only the Agent can assign data.

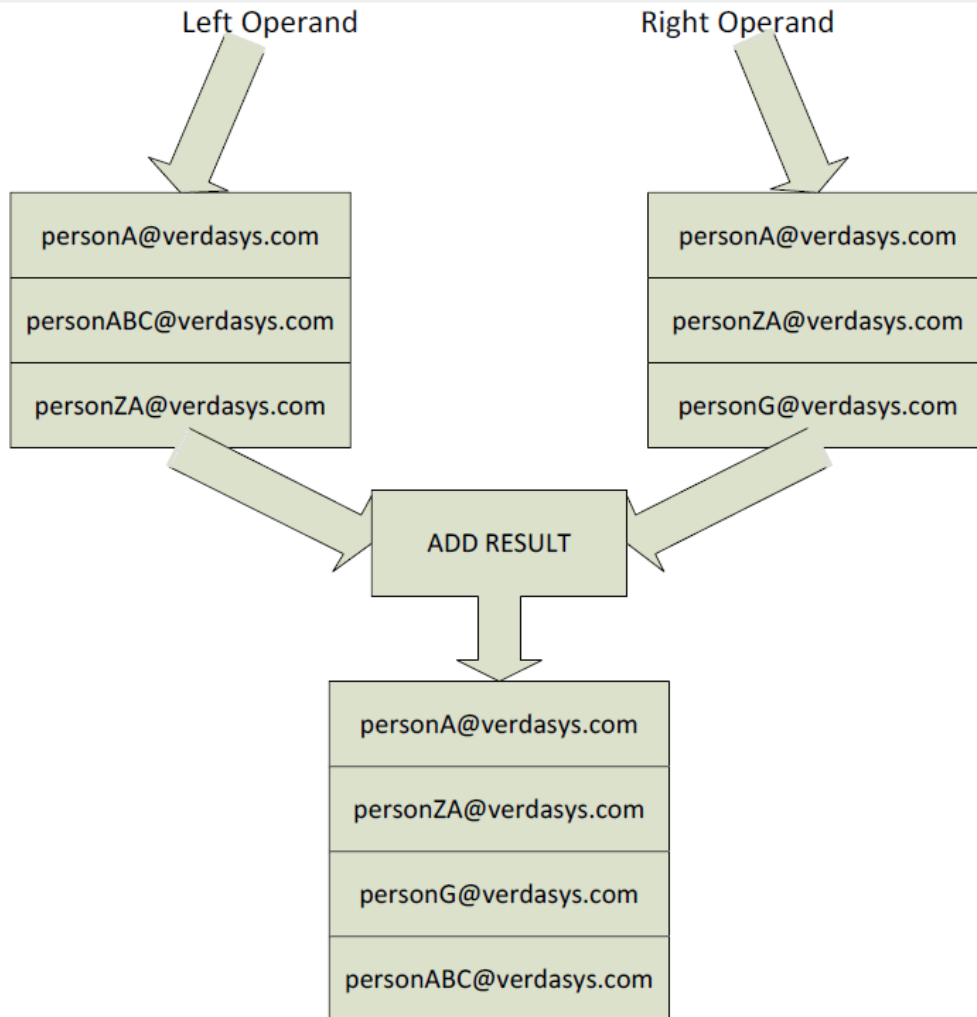
Using rule variables, you can assign data directly, or you can transfer data from other objects, such as rule properties and variables.

Adding Variables

After you declare rule variables, you can perform operations with them such as adding or subtracting them.

Adding Arrays of Strings

When you add arrays together, DG applies this algorithm:



This is the addition process:

1. Place all of the items from the right operand list at the top of the destination array.
2. Add each left operand list to the destination array if the entry is not already in the destination array.
3. Replace the left operand array with the destination array create the result of the addition.

Adding Arrays of Integers or Strings

Integer array addition is similar to adding string arrays, with some differences if you add them as arrays or not.

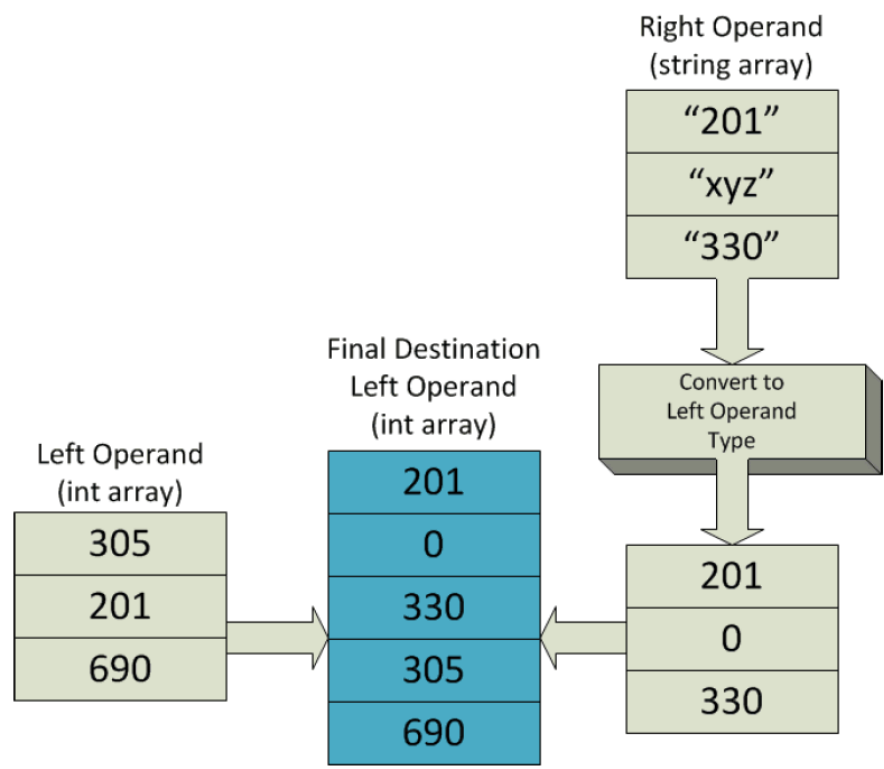
Rule elements that add integers might look like this:

```
<add asArray="true">
  <varint name="myIntArray" scope="global" />
  <varstring name="myStringArray" scope="global" />
</add>
```

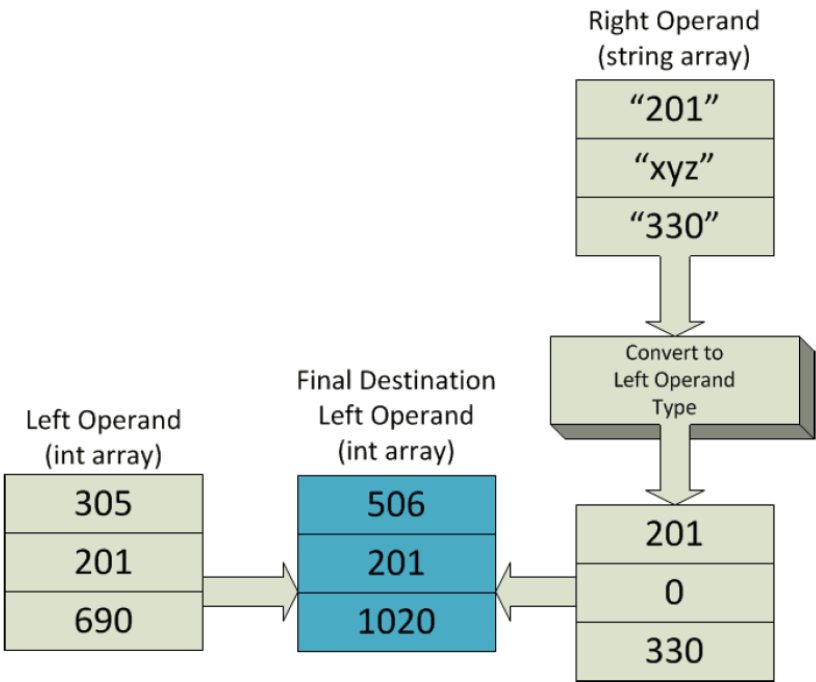
In this code, the idea is to add the array `myStringArray` to the integer array `myIntArray`. And the code specifies to add these as arrays. The left operand is the integer array. The right operand is the string array. The next figure shows what happens when your rule executes the code shown.

The commands shown perform a type conversion of the `myStringArray` to integer type before adding the array elements to `myIntArray`. In particular, the string elements 201, xyz and 330 are converted to the integers 201, 0 and 330. That second element after the conversion—0—results from converting the string xyz to integers because there is no integer equivalent of xyz. So the result of conversion is 0.

As is true with string array addition, the destination array contains only unique entries. Though 201 appears in both `myIntArray` and `myStringArray`, it appears only once in the destination array. And the destination array replaces the `myIntArray` after the operation.



If you set `AsArray` to `false` when you add integer or string arrays, the behavior is somewhat different from `asArray = true`. Recall that integers are not arrays by default, but strings are arrays. So addition looks like this when you mix string types and integer types:



The rule elements that might create this addition are:

```
<add asArray="false">
  <varint name="myIntArray" scope="global" />
  <varstring name="myStringArray" scope="global" />
</add>
```

In this example, the destination array contains the arithmetic sums of the entries in myIntArray and myStringArray:

myIntArray Element	myStringArray Element	Destination Array Element
305	"201"	305 + 201 = 506
201	"xyz"	201 + 0 = 201
690	"330"	690 + 330 = 1020

From the example, you see that when you add integer arrays, the destination array contains the sum of the input arrays

When you add strings with `asArray = false`, the result is to concatenate the strings with the left operand value followed by the right operand value, as shown in the next example that treat strings as if they are not arrays:

```
<add asArray="false">
  <varstring name="myName" scope="global" />
  <string value="Sanderson" />
</add>
<set>
  <varstring name="myName" scope="global" />
  <string value="George" />
</set>
```

Because the `preserveCase` argument is not included, this code generates the string value `georgesanderson` in the variable `myName` after execution. `myName` starts as `george` (the left operand) and the `add` operator appends `sanderson` (the right operand) to `myName`.

Adding Mixed Types

For the final examples of adding with rule variables, these two sets of rule XML add variable values of different types, changing the result:

```
<add asArray="false">
  <varstring name="MyVarString" scope="event" />
  <int16 value="45" />
</add>
```

Notice, the above rule operation is set up as an append operation (`asArray = false`), but the second argument is an `int16`, so the operation converts the value from `int16` to string. If `MyVarString` contained a street address, such as `Main Street`, you would be adding the string `45` to the address—`MyVarString` would become `Main Street45`.

The following operation automatically converts the type of the second argument to that of the first argument—from string to `varint`.

```
<add asArray="false">
  <varint name="ValueA" scope="event" />
```

```
<string value=" 3" />
</add>
```

In this set of XML, your operation would add the value 3 as an integer to the value in ValueA.

Subtracting Variables

You can subtract variables as you can add variables. The algorithm to perform the subtraction is similar to addition.

Subtracting Arrays of Strings

This code example shows DG subtract string arrays from one another

```
<sub>
  <variant name="decrementingVar" scope="process" />
  <int value=" 1" />
</sub>
<sub>
  <varstring name="stringList1" scope="global" />
  <string value="mynewstringvalue" />
</sub>
```

This is how DG performs the subtraction:

1. Copy all left hand items (stringList1), in order, to the beginning of the destination array, but only if they do not exist in the right hand list.
2. Replace the left hand item array (stringList1) with destination array.
3. Remove the new string value (mynewstringvalue) in stringList1 if it is already in the destination array. If it is not in the list, do nothing to stringList1.

Testing Variable Values

Testing variables against property values in rules is the primary aim of rule variables.

You use relational operators to compare rule variables to property values returned by events or property values in other rules. The available operators are:

For Comparing Integers	For Comparing Strings	For Comparing Booleans
equal	equal	equal
greaterThan	list	greaterThan
lessThan	like	lessThan
list	regExp	

With `greaterThan`, you could check whether the value of an integer variable has exceeded some set value:

```
<greaterThan>
  <vari nt name="Vari abl eName" scope="event" />
  <i nt val ue="1" />
</greaterThan>
```

When the value of `VariableName` is greater than 1, the rule evaluates to true.

You could also test to see whether something is on or off (or true or false):

```
<l essThan>
  <varbool name="Vari abl eName" scope="event" />
  <i nt val ue="1" />
</l essThan>
```

With `VariableName` declared as a Boolean, when its value is less than 1, the rule evaluates to false.

For one more use, you could use the regular expression operator to match a variable string:

```
<regExp expr=". *@* .(\. com/] . co\. uk)">
  <varstring name="Var-Mail" scope="global" />
</regExp>
```

This set of commands evaluates to true when Var-Mail includes any string that presents an email address (*@* format) that contains .com, .co or .uk. Alternatively, you could use Like to compare strings:

```
<like expr="%\PAY.xls">
  <varstring name="Var-HRFiles" scope="process" />
</like>
```

Like is a bit more efficient for this task than regExp. This sample code would match any expression in Var-HRFiles that contained the string PAY.xls.

If you have a string variable named VariableName, you can search the string for occurrences of fixed strings, such as "1", "2" or "3" with the list operator:

```
<in>
  <varstring name="VariableName" scope="event" />
  <list>
    <string value="1" />
    <string value="2" />
    <string value="3" />
  </list>
</in>
```

Similarly, the equal operator lets you find exact matches to strings. For example, this XML rule code tries to match the string in Webmail-Addr to the string in curProcessWindowTitle involved in an event that triggered the rule.

```
<equal>
  <varstring name="Webmail-Addr" scope="process" />
  <curProcessWindowTitle />
</equal>
```

When you use the equal operator to test an array of integers, your query compares the test value against every entry on the list. In this example, the array MyInt16List contains three values—5, 10 and 32. When you query to see whether any specific int16 value is in MyInt16List, the rule engine checks 5, 10 and 32 to see if any one matches.

```
<equal>
  <varint16 name="MyInt16List" scope="global" />
```

```
<int16 value=" 10" />
</equal >
```

The rule evaluates to true because MyInt16List includes the specified value 10.

Array Processing Commands

Various elements and properties work with rules to let you process arrays:

- Loop through (sometimes called enumerating) the elements in an array
- Get the length of an array
- Set an element of an array
- Get an element of an array
- Return the index of an array element

To enumerate an array you need to define a component rule that will be the function that will be called for each array element. You then use the element <foreach-Function> to call the component rule with each item in the array.

```
COMPONENT RULE "componentLoopTest"
<and>
  <set>
    <varstring name="NameStartingWithAorB" scope="global" />
    <varstring name="item" scope="event" />
  <set>
    <or>
      <like expr="a%">
        <varstring name="item" scope="event" />
      </like>
      <like expr="b%">
        <varstring name="item" scope="event" />
      </like>
    </or>
  </and>

...XML TO LOOP THROUGH ARRAY
<foreachFunction name="componentLoopTest" "breakOn=true">
  <varstring name="item" scope="event" />
  <evtMailRecipients />
</foreachFunction>
```


The above example loops through each mail recipient and copies the first string in `evtMailRecipients` that begins with an 'a' or 'b' to the global array "NameStarting-WithAorB".

You can also loop through each item in the array from the end of the array to the beginning by adding the "reverse" attribute.

```
...XML TO LOOP THROUGH ARRAY
<foreachFunction name="componentLoopTest" reverse="true" breakOn="true">
  <varstring name="item" scope="event" />
  <evtMailRecipients />
</foreachFunction>
```

Notice the "breakOn" attribute. This makes the `<foreach>` loop exit if the component rule returns the result specified. The default is to break out of the loop on a false result from the component rule.

There are two additional elements you can use in DG rules.

```
<setArrayItem>
<getItem>
```

The first one, `setArrayItem`, sets the value of an array member at the specified index. The second one gets the value of an array member from the specified index.

```
<setArrayItem>
  <varstring name="MyArray" scope="global" />
  <int value="2" />
  <string value="ValueToPutInArray...ABC" />
</setArrayItem>
```

which is equivalent to:

`MyArray[2] = "ValueToPutInArray...ABC"`

and

```
<getItem>
  <varstring name="Item" scope="event" />
  <varstring name="MyArray" scope="global" />
  <int value="1" />
</getItem>
```

which is equivalent to:

```
Item = MyArray[1];
```

Array Variable Processing Elements

There are additional rule elements you can use to work with array rule variables.

- **Array Length**

You can get the length of an array by using the `varArrayLength` property.

```
varArrayLength name="MyArray" scope="process"
```

Note: You cannot change `varArrayLength` directly with `<Set>`. To change the length, add or remove items from the array itself.

Note: If you use component lists, you cannot use the `varArrayLength` property to get the component list size in rules written against a DG 7.3 or later Agent. If you add a DG 7.3 or later Agent, you will need to write new rules using the `componentListSize` property to get the component list size. For more information on component lists, refer to [“Component Lists” on page 371](#).

- **Direct Array Member Access**

There are two elements you can use in your XML rules.

- `setArrayItem`

This one sets the value of an array element at the specified index. For some examples:

```
<setArrayItem>
  <varstring name="MyArray" scope="global">
    <int value="2" />
    <string value="ValueToPutInArray...ABC" />
  </setArrayItem>
```

This produces the same result as

```
MyArray[2] = "ValueToPutInArray...ABC"
```

in JavaScript.

- getItem

This one gets the value of an array member from the specified index.

```
<getItem>
  <varstring name="Item" scope="event" />
  <varstring name="MyArray" scope="global" />
  <int value="1" />
</getItem>
```

This is equivalent to the Java Script construct:

```
Item = MyArray[1];
```

- Function to Loop Through Arrays

A function, `foreachFunction`, enables you to write rules that look at each element in an array. Refer to “Enumerate Arrays with a Loop Function” for a bit more information.

- Enumerate Arrays with a Loop Function

With the `foreachFunction`, you can enumerate an array (loop through the elements). To enumerate arrays, you define a component rule that the loop function calls for each array element. You use the element `<foreachFunction>` to call the component rule with each element in the array.

Here is an example that defines a component rule named `component-LoopTest` and calls the component rule with the `foreachFunction`. The array contains email addresses from a send mail event.

Component Rule — `componentLoopTest`

```
<and>
  <set>
    <varstring name="NamesStartingWithAorB" scope="global" />
    <varstring name="item" scope="event" />
  </set>
  <or>
    <like expr="a%">
      <varstring name="item" scope="event" />
    </like>
    <like expr="b%">
      <varstring name="item" scope="event" />
    </like>
  </or>
</and>
```

Rule XML — loop through the recipient email addresses from a mail event.

```
<foreachFunction name="componentLoopTest" >
  <varstring name="item" scope="event" />
  <evtMailRecipients />
</foreachFunction>
```

The example loops through each mail recipient and copies any recipient that begins with an 'a' or 'b' to the global array "NamesStartingWithAorB". The looping begins with the beginning of the array and continues to the end.

Returning the Index Value of an Array Element

After you create an array of values, you might want to know which array element holds a particular value. That is, you want to know the index of a specific element in your array.

The array operator `GetIndexOfArrayItem` allows you to specify an element in an array and then use the operator to return the array index of the element. It joins the rest of the array operators—`set`, `add`, `sub`, `setArrayItem`, `getArrayItem`, `dispose`.

Using `GetIndexOfArrayItem`

To use the operator, you provide the name of the array to search and the string to find in the array:

```
<getIndexOfArrayItem>
  <varint name="IndexOfArrayItem" scope="global" />
  <varstring name="MyArrayToSearch" scope="global" />
  <string value="ABC" />
</getIndexOfArrayItem>
```

In the code snippet the array to search is `MyArrayToSearch` and the string to find is `ABC`.

If `MyArrayToSearch` contains the following four entries

```
{ABA, ABB, ABC, ABD}
```

- If ABC is the first entry in the array, the function returns a value of 0.
- If ABC is the second entry in the array, the function returns a value of 1.
- If ABC is the third entry in the array, the function returns a value of 2.
- If ABC is the fourth entry in the array, the function returns a value of 3.
- If ABC is not present in the array, the function returns a value of -1.

Note: Array indexes are zero-based. The first entry in an array is 0, not 1.

Here is an example of finding the index of a file path in an array of file paths. This demonstration uses two rules—one to initialize an array with source file paths and the other to search the array of paths and return the index of a specified path.

Rule 1 — Create an Array of File Paths

Use the Command prompt to open or read files in a test folder. Opening or reading the files creates the array `MyPathsArray` containing source file paths.

```
<and>
  <add>
    <varstring name="MyPathsArray" scope="global" />
    <evtSrcFilePath/>
  </add>

  <like expr = "%test%">
    <evtSrcFilePath/>
  </like>

  <equal >
    <curProcessImageName/>
    <string value="cmd.exe" />
  </equal >

  <in>
    <evtOperationType />
    <list>
      <constOpFileOpen />
      <constOpFileRead />
    </list>
  </in>
</and>
```

Rule 2 — Get the Index of a File Path in MyPathsArray

When the user opens or reads a file in the test folder with Notepad, this rule captures and stores the source file path for the operation in MyPathsArray and returns the array index value for the captured path in IndexofSpecifiedString.

```
<and>
  <greaterThan>
    <varint name="IndexofSpecifiedString" scope="event" />
    <int value="2" />
  </greaterThan>

  <not>
    <equal >
      <varint scope="event" name="IndexofSpecifiedString"/>
      <int value="-1"/>
    </equal >
  </not>

  <getIndexOfArrayItem>
    <varint name="IndexofSpecifiedString" scope="event" />
    <varstring name="MyPathsArray" scope="global" />
    <varstring name="MyPathName" scope="event" />
  </getIndexOfArrayItem>

  <add>
    <varstring name="MyPathName" scope="event" />
    <evtSrcFilePath/>
  </add>

  <equal >
    <curProcessImageName/>
    <string value="notepad.exe" />
  </equal >

  <like expr = "%test%">
    <evtSrcFilePath/>
  </like>

  <in>
    <evtOperationType />
    <list>
      <constOpFileOpen />
      <constOpFileRead />
```

```

        </list>
    </in>
</and>

```

Deleting an Array Item With Dispose

When you need to reset an entry from an array, the dispose operator provides the means. With dispose, you specify the array that has the element to reset and the element itself:

```

<dispose>
  <varstring name="MyArray" scope="global" />
  <string value="ABC" />
</dispose>

```

where ABC is the element to reset to its original value (dispose) and the array MyArray holds ABC.

Here is another example of using dispose:

```

<dispose name="atp-9103-g-RelatedRuleNames" scope="global" />
<dispose name="atp-9103-ProcessThreatBusCurrentEvtDetailID" scope="global" />
<greaterThan>
  <varArrayLength name="atp-9103-ProcessThreatBusCurrentEvtDetailID" scope="global" />
  <int value="16" />
</greaterThan>

```

Using the Eval Operator

One of the most important operators you can use with rules and rule variables is Eval. Using eval in a rule forces the rule engine to evaluate all clauses in the rule, even if one of the clauses evaluates to false, which would normally end the evaluation of the rule.

The primary use of eval is to ensure that the rule engine sets or increments rule variable values when the rule would exit before evaluating the variables. If you have variable operators in the rule, these will be evaluated in all cases if you add the eval operator.

Caution: Use eval very carefully. Improper use may cause the rule system to store or accumulate large quantities of data that can badly degrade your system performance. Eval forces the rule engine to evaluate all of the rule, even after one clause is false. If, for example, your rule captures and stores files or images when the user opens them, that capture process would still happen even in the event of a false evaluation in the rule. Storing the captured files may use up a large amount of memory inadvertently.

Adding the Eval Operator to Rules

This example tracks the destination file name of the last file touched by the user:

```
<eval >
<set>
  <varstring name="Var-Last-evtDestFilePath " scope="process"/>
  <evtDestFilePath/>
</set>
<and>
  <like expr="c: %\classified files\%">
    <evtDestFilePath />
  </like>
  <equal >
    <evtOperationType />
    <constOpFileCopy />
  </equal >
</and>
</eval >
```

By placing the <eval>...</eval> commands outside of the <and>...</and> elements, you force the rule engine to capture the destination file path even when the operation is not a file copy. Any file operation that triggered this rule would cause the Agent to capture the path and name of the destination file. With the rule variable declaration inside the eval operator, Var-Last-evtDestFilePath is set to the path and file name of the file used in the operation in every instance of a user triggering this rule.

In some cases, you might want to capture data for a variable, even if no data is available. The following rule code uses eval to capture the property value when no value exists:


```

<eval >
<set>
  <varstring name = "DebugVar-curProcessAddressBar" scope =
"event"/>
  <varstring name="DebugTempVar" scope="event"/>
</set>
<set>
  <varstring name="DebugTempVar" scope="event"/>
  <curProcessAddressBar/>
</set>
<set>
  <varstring name="DebugTempVar" scope="event"/>
  <string value="UNAVAILABLE"/>
</set>
</eval >

```

In process flow form, here is what the rule engine does while evaluating the rule:

- A. Sets the variable DebugTempVar to the string UNAVAILABLE.
- B. Sets the variable DebugTempVar to the property value returned by curProcessAddressBar.
- C. Moves (sets) the value of DebugTempVar into a new variable DebugVar-curProcessAddressBar.

Posting Rule Variables to the DGMC

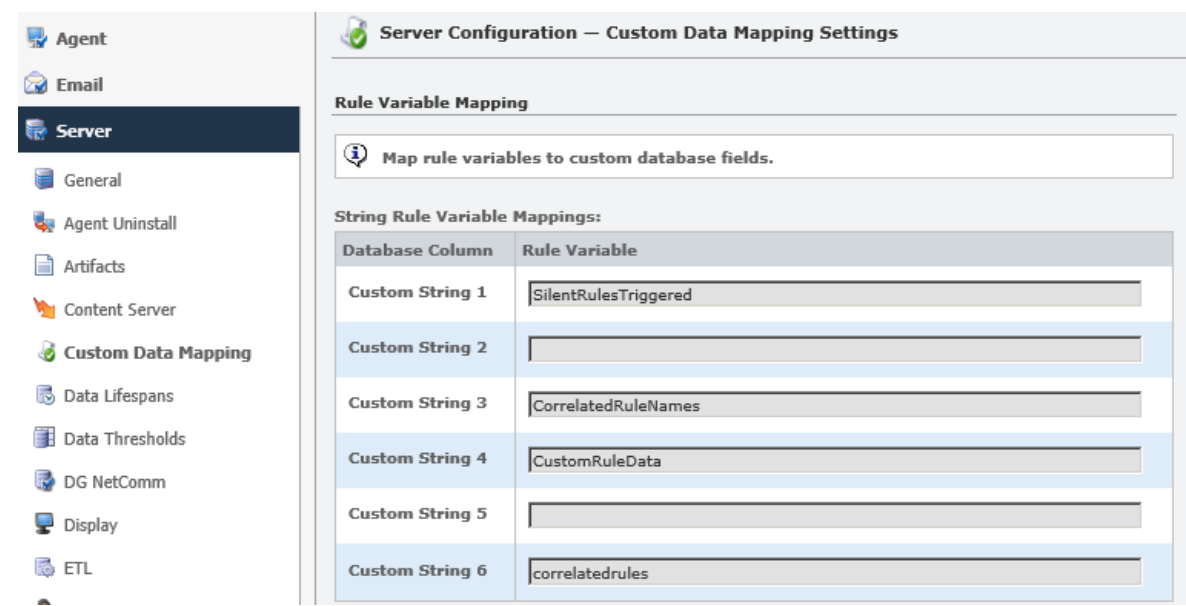
You can map rule variables in a mapping table on the DG Server for custom event data. When you map rule variables, the Agent bundles and sends to the Server only variables that match the names of variables that you mapped in custom data mapping. The variables are entered into the database as event data and you can view the data in reports in the DGMC. You can map up to six string variables, four integer variables, and four date variables.

Network Transfer Upload (NTU) events provide a good use case for using custom rule variables. For NTU events, the window title of the process involved in the event often has useful forensic information. (On Gmail, for example, the window title contains the email account the user is emailing from.) Capturing information contained in the window title requires:

- Mapping a rule variable to a database column field on the Server Configuration — Custom Data Mapping Settings page.
- Writing a rule that populates this variable in NTU events with the window title of the process involved in the event.

Mapping a Rule Variable

On the Custom Data Mapping Settings page, map the CustomRuleData string variable to Custom String 4. (You can use this string to send various types of data. DG reserves strings 1, 3, and 6 for specific types of data.)



Writing Rules With Rule Variables


When writing rules with rule variables, keep the following in mind:

- The variable you choose must be event-scoped. For example:
`<varstring name="CustomRuleData" scope="event" />`

- String variables are limited to 1024 bytes in the DGMC database. The DGMC will store and display only the first 1024 bytes of custom strings sent from the Agent.
- String variables can be arrays.
- Integer variables and date variables sent to the DGMC should be single values and not arrays.

The following rule would populate the CustomRuleData variable declared in the Custom Data Mapping Settings with the current process's window title:

```
<and>
  <return>
    <bool value="false" />
  </return>
  <!-- Populate a variable declared in the Custom Data Mapping settings on DGMC with the current process's window title -->
  <!-- We add to the array variable instead of setting it, so as to preserve its existing contents, in case another rule has already populated it -->
  <add asArray="true">
    <varstring name="CustomRuleData" scope="event" />
    <varstring name="temp-windowtitle" scope="event" />
  </add>
  <add asArray="false">
    <varstring name="temp-windowtitle" scope="event" />
    <curProcessWindowTitle />
  </add>
  <set>
    <varstring name="temp-windowtitle" scope="event" />
    <string value="Window title: " preserveCase="true" />
  </set>
  <equal>
    <evtOperationType />
    <constOpNetTransferUpload />
  </equal>
</and>
```

When this rule is executed, NTU events returned to the DG Server by the Agent display a Custom Data icon () for Network Transfer Upload operations in the Forensic Activity Report:

	Date		Operation
	3/16/2016 12:38:59 AM		Network Transfer Upload

Clicking the icon allows you to view the custom data returned by the Agent:

Custom Data

1 of 1 Export to CSV

Name	Value
CustomRuleData	Window title: inbox (488) - verdasys33@gmail.com - gmail - windows internet explore

You can create a custom report to view the collected custom data in Custom Reports or Data Exports. In the Custom Report Wizard, step 2 of 10: Choose Fields, select **Custom String 4**.

Custom Report Wizard SYSTEM

CUSTOM REPORT WIZARD

Step 2 of 10: Choose Fields

Choose specific fields for display in your report.

Fields:

Custom Int 3

Custom Int 4

Custom String 1

Custom String 2

Custom String 3

Custom String 4

Custom String 5

CHAPTER 11 Component Rules and Rule Functions

Digital Guardian provides two kinds of functions that you use to manage rule development and operation:

- **Component Rules** — Component rules are rules (sometimes called user functions) that you reference from the definitions of other rules, similar to functions in a programming language. By using component rules, you can reduce the time required to write and maintain your rules.

Component rules do not contain action, severity, or status information. That information is supplied by the rule that uses the component rule. For more information, refer to [“Create Component Rules” on page 310](#).

Note: A component rule cannot be the only property in a rule. A rule must contain at least one other property in addition to a component rule.

- **Rule Functions** — Rule functions (sometimes called user functions) are functions that you call in rules to perform operations on the Agent. These functions provide features that you can use to manipulate Agents, Agent configurations and more. For more information about the functions available and using the functions, refer to [“About Rule Functions” on page 314](#).

Create Component Rules

You create component rules from DGMC. As long as other rules reference the component rule, you cannot rename the rule. To rename a component rule, remove all references to that component rule from any other rules.

Component rules use the same syntax as all other Digital Guardian rules.

1. In the DGMC, hover your cursor over **Policies** and select Component Rules. The Control Component Rules page appears. This page displays a list of existing control component rules.
2. In the Rules section, select the component rule type to create. A list of existing component rules of that type appears.
3. From the Actions menu select **Create New Rule**. The Create New Rule page opens.
4. Enter the following information:
 - Name** - The name of the rule. You cannot change this value once you have created the rule. Rules use this value to reference the component rule.
 - Category** - The category to which the rule will belong. You must assign all rules to a category. You can assign a rule to an existing category or create a new category. To create a new category, select **New Category** and enter a new category name.
 - Description** - A brief description of what this rule does. The text you enter here will appear in the Description section of the Rule Listing on the Rules page.
 - Definition** - The Rule text. A rule is only triggered when all of its conditions are met. For information about writing rule definitions, refer to Rule Definitions.
5. Click **Save Rule**. The component rule is created. Now you can call this rule from other rules.

Call a Component Rule

You call a component rule from within other rules using the `<userfunction name=""> element`.

The component rule and the rule that calls it must be the same rule type. For example, a filter rule can only call component filter rules.

Digital Guardian converts all component rule names to lowercase. When you refer to a component rule, be sure to use lowercase characters. When you reference a component rule from another rule, the rule name case must match, as shown in the following example:

```
<userFunction name="data Leakage" />
```

Example

This example shows two rules—a component rule that defines the drive type and operations for the rule, and a control rule that defines source file extensions. The combined rule prevents data from leaking from your computer.

- Component Rule that defines egress paths for data leakage.

```
<or>
  <equal >
    <evtDestDriveType/>
    <constDriveRemovable/>
  </equal >
  <i n>
    <evtOperationType/>
    <l i s t>
      <constOpNetTransferUpload/>
      <constOpCDBurn/>
    </l i s t>
  </i n>
</or>
```

- Control rule that uses the data leakage component rule to stop data leakage of source files on the egress paths.

```
<and>
  <userFunction name="data Leakage" />
  <i n>
```

```
<evtSrcFileExt/>
<list>
  <string value="cpp"/>
  <string value="c"/>
  <string value="h"/>
</list>
</in>
</and>
```

Export and Import Component Rules

You can import component rules that you have exported from Digital Guardian. You might take this step if you want to move rules from an evaluation instance to a production instance, or to restore an earlier version of exported rules.

Users with the role of System Administrator, or Policy Manager can export and save copies of the specific rule type they manage to any location on the network. Exported rules are stored as XML files that you can then import back into Digital Guardian.

When you import multiple items to Digital Guardian, you must import those items in the following order to maintain referential integrity:

1. Import Custom Skins
2. Import Custom Prompts
3. Import Component Rules
4. Import Rules

These items contain references to one another. If you import an item before the object of its reference is present, Digital Guardian interprets the references as an error and rejects the import. For example, rules can refer to component rules. If you import a rule that references a nonexistent component rule, Digital Guardian rejects the import. By importing the component rules first, you allow Digital Guardian to validate the references and maintain its integrity.

Exporting Component Rules

You can export a component rule from Digital Guardian to an XML file.

1. In the DGMC, hover your cursor over **Policies** and select Component Rules. The Control Component Rules page appears. This page lists existing control component rules.
2. Check the rules to export.
3. From the Actions menu select **Export Selected Rules**. A dialog box asks if you want to open or save the exported rules.
4. Click **Save As**. The Save As dialog box opens.
5. Specify a location and filename for the exported rules and click **Save**. The rules are exported to an XML file.

Importing Component Rules

You can import a component rule from an XML file to Digital Guardian.

1. On the DGMC menu bar, hover your cursor over **Policies** and select Component Rules. The Control Component Rules page opens.
2. From the Actions menu select **Import Rules**. The Import Utility dialog box opens.
3. Navigate to the XML file that contains the rules you want to import and click **Import**. The rules are imported to Digital Guardian. A message box appears to confirm a successful import. The imported rules now appear in the listing of rules.

About Rule Functions

Agents can call functions in rules to execute actions or call predefined functions. These function-driven operations occur at the operating system kernel level and are launched from within Digital Guardian rules.

There is a finite list of functions you can execute from rules (refer to the following table). These operations are specific to Digital Guardian. You cannot, for example, create a rule that executes an operating system command or a function you create. The supported functions extend the capabilities of Digital Guardian features only.

Name of Function	Use the Function In a Rule to...
DG_AggregationControl	Indicate how to consolidate redundant events in event cache
DG_CaptureFile	Trigger the file capture process on the Agent
DG_CaptureFileByPath	Trigger the file capture process on the Agent and identify the file to capture by the full file path
DG_ClearCachedData	Clear the event cache
DG_ClearProcessFlags	Remove specific process flags from a process in response to an event
DG_DecryptFile	Decrypt a file based on a triggering event
DG_DeleteFileClassification	Erase the file classification stream from a source file
DG_EnableEventCaptureForCurrent Process	Enable capture of events of the specified type for the specified amount of time on the current process
DG_EnableRegistryEventCaptureFor CurrentProcess	Enable capture of registry events for all keys and values, limited to the specified registry event type and for the specified amount of time on the current process
DG_EncryptFile	Encrypt a file based on a triggering event
DG_GenerateCustomEvent	Create a custom event that rules can pick up
DG_IsProcessActive	Test whether a specific process is still in the process cache (active)

Name of Function	Use the Function In a Rule to...
DG_KillProcess	Stop a process that is running on the Agent, except SYSTEM or any DG process
DG_KillTimerEvent	Use a rule at a future time to cancel a timer that has not fired after you create the time event
DG_NameCustomEvent	Provide names for custom events
DG_RefreshEventClassification	Refresh the classification information for an event in the DG collection database on the server
DG_RemoveFileClassificationTags	Remove selected classification tags from a file, leaving others if any (compare to DG_ResetFileClassification)
DG_RemoveRegistryKeyEvents	Stop monitoring on or more keys that you set up to be monitored
DG_RemoveRegistryValueEvents	Stop monitoring one or more values that you set up for monitoring
DG_RequestRiskAnalysis	Trigger a scan of active memory on the Agent computer
DG_ResetFileClassification	Clear the contents of the file classification stream (remove all of the classification tags) for the source file in an event
DG_SendCachedData	Send events cache contents to the Server in bundles
DG_SetProcessFlags	Add process flags to the existing flags on a process in response to an event
DG_SetRegistryKeyEvents	Set up monitoring of one or more keys and their subkeys
DG_SetRegistryKeyEventsEx	Enable capture of registry events for specified keys and subkeys, limiting event capture to the specified registry event type and for the specified amount of time. Event capture is enabled for all processes.
DG_SetRegistryValueEvents	Set up monitoring of one or more key values

Name of Function	Use the Function In a Rule to...
DG_SetRegistryValueEventsEx	Enable capture of registry events for the specified values, limiting event capture to the specified registry event type and for the specified amount of time. Event capture is enabled for all processes.
DG_SetTimedProcessFlags	Set specific process flags on a process for a specified time in response to an event
DG_SetTimerEvent	Create a timer to fire from a rule
DG_String_SplitPath	Separate a full file path string into four parts: drive letter, file path, file name, and file extension
DG_StrTok	Split a string at locations specified by tokens (characters) in the string
DG_TagFile	Add a specified classification tag to a file classification stream
DgMaster_IsAgentRunning	Test the status of the Agent
DgMaster_IsFeatureEnabled	Test whether a feature is enabled on an Agent
DgMaster_IsStealthOn	Test whether the Agent is in stealth mode
DgMaster_IsTamperResistOn	Test whether the Agent is in tamper-resistance mode
foreachFunction	Enumerate the elements in an array

How Functions Fit in the Workflow

In a rule, you specify the action to be taken in the rule metadata—the options you select when you create the rule, such as **Action**, **Send Alert** or **Continue Rule Evaluation**. Functions are executed in-line with rule logic.

Functions consist of a set of instructions to the Agent operating system kernel. Most functions are synchronous and are completed before the rule engine executes more rule logic.

Here is what happens when the user performs an operation when a rule is in place and includes a rule function:

- A. Agent sees the user activity.
- B. The Agent sends the event from the activity to the rule engine.
- C. The rule with the functions starts execution.
- D. The rule engine calls the rule function.
- E. The function executes to completion, returning to the rule execution.
- F. The rule end execution.

Using functions can allow you to extend the capabilities of rules beyond the default actions of block, warn, justify or encrypt. Because rule logic drives the functions, you can direct the response based on your use case and specific criteria.

What Do Functions Look Like?

Rules call functions with the `<function>...</function>` element. In any programming language, functions are program fragments that perform defined tasks. DG rule functions behave the same way. When called, they operate within the provided constraints (called **parameters** in functions) to perform a defined task, then might return a code to the rule upon completion. Some functions do not have return values.

Within the elements, the specific named function is called. Each function has an approved name that the Agent evaluates. The Agent validates the function name when it receives the rule as well as all parameters. If anything in the function is incorrect, the rule containing the function is ignored. The rule that contains the invalid function call is dropped and the Agent sends an operational alert (Op Alert) to the Server.

Each function has parameters to provide constraints (input arguments) for the task. The rule author specifies the values passed to the function parameters in the rule logic. Therefore the parameters differ for each function, and the values for each parameter can differ between control rules or function calls.

After a function completes the function task, it has a return code. The return codes are typically the same standard code, however a few supported variations exist.

In the sample code below, the function element calls “*function*,” providing a parameter that uses the ID of the rule that fired (*currentEventID*). After the function completes, it returns a value in *fc_result*. You can use any name for the integer variable that receives the returned result from the function call.

```
<function name="function">
  <parameters>
    <currentEventID/>
  </parameters>
</return>
  <variant name="fc_result" scope="event" />
</return>
</function>
```

For example, when your rule calls the function *DG_CaptureFile*, the *int* values in the function specify file hashing, capture and transfer configuration parameters. The values assigned to the parameters direct the function behavior. The function closes with a return code, in this case “*fc_result*” for the scope of the current event. The return is not always an *int* value.

The following sections provide details about the functions provided in DG:

- [“String Management Functions” on page 319](#)
- [“Array Management Functions” on page 320](#)
- [“Timer Functions” on page 322](#)
- [“Custom Events Functions” on page 325](#)
- [“Function To Kill Processes” on page 328](#)
- [“Function for Resetting the Classification Stream” on page 329](#)
- [“Function for Erasing the Classification Stream” on page 329](#)
- [“Function for Removing Tags From the Classification Stream” on page 332](#)
- [“Registry Key Functions” on page 341](#)
- [“Agent Configuration Information Functions” on page 355](#)
- [“Event Caching Functions” on page 359](#)

String Management Functions

DG_String_SplitPath — Takes a full file path string—with drive letter, UNC path or relative path—as an input argument and returns the path separated into four strings:

- Drive letter (where available in the input string)
- File path
- File name
- File extension

The function is identical to the C programming function `split_path`. The `out="1"` attribute is required to indicate that the variable is an output of the function. The default is `input`, indicating that the parameter is an input to the function.

```
<function name="DG_String_SplitPath" >
  <parameters>
    <varstring name="path" out="1" />
    <varstring name="drive" out="1" />
    <varstring name="directory" out="1" />
    <varstring name="fileName" out="1" />
    <varstring name="extension" out="1" />
  </parameters>
  <return>
    <varbool name="result" scope="event" />
  </return>
</function>
```

DG_StrTok — Function to split a string at locations identified by tokens, such as periods (.). For example, this rule splits an IP address into four parts, separated at the period.

```
<function name="DG_StrTok">
  <parameters>
    <evtRemoteAddress/>
    <string value="." />
    <varstring name="parts" scope="global" out="1" />
  </parameters>
</function>
```

This rule splits the Remote Address (for example 255.255.194.3) into four strings—"255", "255", "194", "3". The period is the token as specified by the

string value statement in the function. Similar to `DG_String_SplitPath`, the `out="1"` attribute is required to indicate that the variable is an output of the function.

Example

The following function configuration would separate a phone number, such as 888-555-1234 into three strings—"888","555","1234" using the hyphen as the token.

```
<function name="DG_StrTok">
  <parameters>
    <varstring name="phone_number" />
    <string value="-" />
    <varstring name="parts" scope="global" out="1" />
  </parameters>
</function>
```

Array Management Functions

To enumerate an array, you define a component rule that will be the function that will be called for each array element. You then use the element `<foreachFunction>` to call the component rule with each item in the array.

```
<!-- Component Rule "componentLoopTest" -->
<and>
  <set>
    <varstring name="NameStartingWithAorB" scope="global" />
    <varstring name="item" scope="event" />
  <set>
    <or>
      <like expr="a%">
        <varstring name="item" scope="event" />
      </like>
      <like expr="b%">
        <varstring name="item" scope="event" />
      </like>
    </or>
  </and>
```


You use the code in the next example as part of a separate rule that uses arrays, such as an array of email recipients.

```
<!-- XML to loop through array -->
<foreachFunction name="componentLoopTest" breakOn="true">
  <varstring name="item" scope="event" />
  <evtMailRecipients />
</foreachFunction>
```

The above example loops through each mail recipient and copies the first string in `evtMailRecipients` that begin with an 'a' or 'b' to the global array `"NameStartingWithAorB"`.

You can also loop through each item in the array from the end of the array to the beginning by adding the `"reverse"` attribute.

```
<foreachFunction name="componentLoopTest" reverse="true" breakOn="true">
  <varstring name="item" scope="event" />
  <evtMailRecipients />
</foreachFunction>
```

Notice the `"breakOn"` attribute. This causes the `<foreach>` loop to exit when the component rule returns the specified result (in this example `true`). The default is to break out of the loop on a false result from the component rule.

There are two additional elements you can use in DG rules.

- `<setArrayItem>`
- `<getArrayItem>`

The first one, `setArrayItem`, sets the value of an array member at the specified index. The second one gets the value of an array member from the specified index.

Note: Arrays are zero-based.

Example

This code sets the value of the third array element at index 2 to a string `ValueToPutInArray_ABC`.

```
<setArrayItem>
  <varstring name="MyArray" scope="global">
    <int value="2" />
    <string value="ValueToPutInArray...ABC" />
  </setArrayItem>
```

which is equivalent to this C programming statement:

```
MyArray[2] = "ValueToPutInArray...ABC"
```

Caution: If you have not assigned MyArray[0] and MyArray[1] yet, assigning MyArray[2] automatically assign MyArray[0] and MyArray[1] to empty strings ("").

This code returns the value in element 1 from MyArray.

```
<getItem>
  <varstring name="Item" scope="event" />
  <varstring name="MyArray" scope="global">
    <int value="1" />
  </getItem>
```

which is equivalent to the C programming statement:

```
Item = MyArray[1];
```

You can get the length of an array by using the *varArrayLength* property. For example, this code returns the length of array MyArray.

```
<varArrayLength name="MyArray" scope="process">
```

Note: You cannot use <Set> to change varArrayLength directly. To change the array length, add or remove items from the array itself.

Timer Functions

You can set up a timer to fire via rules and you can kill a timer you set up. Call the following function to configure a timer:

DG_SetTimerEvent

```
<function name="DG_SetTimerEvent">
  <parameters>
    <variant32 name="processId" />
    <variant64 name="timeoutMS" />
  </parameters>
  <return>
    <variant name="MyTimerID" scope="event" />
  </return>
</function>
```

The return result is the timer ID for the timer event you constructed. Often, you would use `curProcessID` to get the process ID, rather than specifying the ID explicitly.

To catch the timer event use a rule similar to this:

```
<and>

<!--Do something like clear states or generate custom events. -->

<equal >
  <evtTimerID />
  <variant name="MyTimerID" scope="global" --ID from SetTimer. />
</equal >
<equal >
  <evtOperationType />
  <constOpTimer />
</equal >
</and>
```

Here is a sample rule that sets a timer to create a timer event every time a user copies a Microsoft Work format file. The timer is set up to fire in 20 seconds (20000 ms) and the ID of the timer event is stored in a global array "MyTimerIDs".

```
<and>
<add asArray="true">
  <variant name="MyTimerIDs" scope="global" />
  <variant name="TempMyTimerID" scope="event" />
</add>
<function name="DG_SetTimerEvent">
  <parameters>
    <curProcessID />
    <int64 value="20000" />
```

```
</parameters>
<return>
    <variant name="TempMyTimerId" scope="event" />
</return>
</function>

<equal >
<evtSrcFileExt />
<stringValue="docx" />
</equal >
<equal >
<evtOperationType />
<constOpFileCopy />
</equal >
</and>
```

Timer Events have two properties:

- <evtTimerID/> to identify the timer involved in the event.
- <evtTimerTimeoutMS/> to specify when the timer fires after your rule sets it, in milliseconds.

Best practice suggests that you always use <evtTimerID> to verify the timer ID in your rules that handle timer events. Verifying the timer ID ensures that you are receiving your correct timer event.

DG_KillTimerEvent

After you create a timer event, you can use a rule at a future time to cancel a timer that has not fired. The DG_KillTimerEvent function cancels timers that you set with DG_SetTimerEvent.

By using <curProcessID> in the example for DG_SetTimerEvent, the timer event is associated with the current process. If the process terminates, the timer event is cancelled automatically. To configure the timer event to live beyond the lifetime of a process, specify 0 (zero) for the process ID parameter when you call DG_SetTimerEvent. However, when you use 0 for the process ID, the process parameters available during handling of the timer event when it fires are the parameters of the system process.

Cancelling a Timer

You can kill any timer you started by calling `DG_KillTimerEvent` to cancel the timer. The event related to the timer will not fire after you cancel the timer. To cancel a timer, you need the timer ID the rule received when it called `DG_SetTimerEvent`. Use the received ID as the timer ID when you call `DG_KillTimerEvent` to kill the timer. In the example, `MyTimerId` is the timer ID from calling `DG_SetTimerEvent`.

```
<function name="DG_KillTimerEvent">
  <parameters>
    <variant name="MyTimerId" scope="global" />
  </parameters>
  <return>
    <varbool name="killresult" scope="event" />
  </return>
</function>
```

Custom Events Functions

A pair of rule functions—`DG_GenerateCustomEvent` and `DG_SetEventDisplayName`—enable you to create and name custom events. Custom Events are generated on demand. In a rule, you request a custom event with the function `DG_GenerateCustomEvent`, and the event can be picked up by other rules during the call to `DG_GenerateCustomEvent`.

`DG_GenerateCustomEvent`

Here is a sample rule to generate a custom event:

```
<and>
<function name="DG_GenerateCustomEvent">
  <parameters>
    <curProcessID />
  </parameters>
  <return>
    <varbool name="result" scope="thread" />
  </return>
</function>
<set>
  <varbool name="DataCopiedOut" scope="thread" />
  <bool value="true" />
</set>
<set>
```

```
    <varstring name="StolenFileName" scope="thread" />
    <evtSrcFilePath />
</set>
<not>
    <like expr="%\mydocs\%">
        <evtDestFilePath />
    </like>
</not>
<like expr="%\mydocs\%">
    <evtSrcFilePath />
</like>
<equal>
    <evtOperationType />
    <constOpFileCopy />
</equal>
</and>
```

For the custom event generated by the rule above, you can apply custom data to it with a separate rule that watches for the event:

As usual, read the rule from the bottom:

```
<and>
    <add>
        <varstring name="customData1" scope="event" />
        <varstring name="StolenFileName" scope="thread" />
    </add>
    <set>
        <varstring name="customData1" scope="event" />
        <string value="Process Stole Data: " />
    </set>
    <set>
        <varbool name="DataCopiedOut" scope="thread" />
        <bool value="false" />
    </set>
    <equal>
        <varbool name="DataCopiedOut" scope="thread" />
        <bool value="true" />
    </equal>
    <equal>
        <evtOperationType />
        <constOpCustomEvent />
    </equal>
</and>
```

The above rule does not alert or send the custom event until the `DataCopiedOut` variable is set to true by the first rule. The above rule resets the state of `DataCopiedOut` so other generated custom events will not trip it. The call `DG_GenerateCustomEvent` generates the custom event. The rule engine runs the event through the rules before the function returns to generate the custom event. The rule engine preserves the time the event was created, so the triggering event that tripped the first rule appears first in forensic reports.

Note: The custom event is generated to be in the context of the current process when the rule fires. The rule engine does this by passing the process ID into the `DG_GenerateCustomEvent` function.

You can save the process ID in an integer rule variable for later use. Because systems recycle and reuse process IDs, they are reliable and valuable only in short-term use.

DG_SetEventDisplayName

After you create a custom event, you can define a display name for the event. When you define a name, that is the name you see for the event in reports in the DGMC.

```
<function name="DG_SetEventDisplayName" >
  <parameters>
    <currentEvent />
    <varstring value="Your Display Name" />
  </parameters>
  <return>
    <varbool name="Result" scope="event" />
  </return>
</function>
```

Note: The custom event will still appear as “Custom Event” in the Forensic reports. The custom display name will be set in `EVENT_DISPLAY` for use in custom reports created under Workspace.

Function To Kill Processes

DG provides a rule function that can kill a running process. Killing a process on demand can be a powerful tool to prevent malware attacks and distribution. You cannot use `DG_KillProcess` to kill the `SYSTEM` process or any DG processes. You can kill any other processes, including services.

With this function you can kill the current process:

```
<function name="DG_KillProcess">
  <parameters>
    <curProcessID />
  </parameters>
  <return>
    <varbool name="killResult" scope="event" />
  </return>
</function>
```

The function returns 0 (failed) or 1 (succeeded) to indicate the status of the attempt to kill a process.

Similarly, you could save the ID of a process in a rule variable and kill the process at a later time:

```
<function name="DG_KillProcess">
  <parameters>
    <variant name="processID" scope="global" />
  </parameters>
  <return>
    <varbool name="killResult" scope="event" />
  </return>
</function>
```

In this example, you first see the process ID by catching an event in another rule and using `set` to capture the ID:

```
<set>
  <variant name="processID" scope="global" />
  <curProcessID/>
</set>
```


Classification Functions

DG provides user rule functions that you can use to control or change the classification stream or classification tags on a file. With these functions, you can

- Reset the classification stream with function `DG_ResetFileClassification`
- Erase the classification stream with function `DG_DeleteFileClassification`
- Remove tags from the classification stream with function `DG_RemoveFileClassificationTags`
- Refresh the classification information for an event in the DG database with the function `DG_RefreshEventClassification`

Function for Resetting the Classification Stream

Using the function `DG_ResetFileClassification` with a file control rule clears the contents of the file classification stream for the source file in an event. This leaves an empty stream on the file. The syntax for `DG_ResetFileClassification` appears in the following example:

```
<function name="DG_ResetFileClassification">
  <parameters>
    <evtSrcFilePath />
  </parameters>
  <return>
    <variant name="statusResult" scope="event" />
  </return>
</function>
```

`DG_ResetFileClassification` takes a string as the input parameter and returns zero (0) in `statusResult` when the reset operation succeeds.

Function for Erasing the Classification Stream

Using the function `DG_DeleteFileClassification` with a source file erases the file classification stream. This leaves the file without the classification stream. The syntax for `DG_DeleteFileClassification` appears in the following example:

```
<function name="DG_DeleteFileClassification">
  <parameters>
```

```
        <evtSrcFilePath />
    </parameters>
    <return>
        <variant name="statusResult" scope="event" />
    </return>
</function>
```

DG_DeleteFileClassification takes a string as the input parameter and returns zero (0) in statusResult when the delete operation succeeds.

When you create rules that remove classification from files, best practice is to select **Execute Rule After Operation** for the rule. This ensures that the rule removes the classification from the file even if other rules classify the file on the same operation.

When you use this function to declassify a file, the DGMCM may report that the destination file is classified when it is not. This happens because the Agent sends the message about the event while the destination file is still classified. After that, the Agent removes the classification on the file as a separate operation. Both operations occur after the event. Removing classification is a separate event that does not yet have its own event on the Agent.

You can use DG_RefreshEventClassification to update the file information associated with the current event with the actual current classification for the files.

Here is an example rule that uses DG_DeleteFileClassification. This rule calls the declassification function twice to remove the classification stream from the source and destination files in the copy operation.

```
<and>
    <function name="DG_DeleteFileClassification">
        <parameters>
            <evtSrcFilePath />
        </parameters>
        <return>
            <variant name="retStatus" scope="event" />
        </return>
    </function>
    <function name="DG_RefreshEventClassification">
        <parameters>
            <currentEvent/>
```

```

        </parameters>
        <return>
            <varbool name="retStatus" scope="event" />
        </return>
    </function>
    <function name="DG_DeleteFileClassification">
        <parameters>
            <evtDestFilePath />
        </parameters>
        <return>
            <varint name="retStatus" scope="event" />
        </return>
    </function>
    <like expr="%\declassfy%">
        <evtDestFilePath />
    </like>
    <equal>
        <evtOperationType />
        <constOpFileCopy />
    </equal>
</and>

```

The declassification trigger is the user copying the file into a special declassification folder:

```

    <like expr="%\declassfy%">
        <evtDestFilePath />
    </like>
    <equal>
        <evtOperationType />
        <constOpFileCopy />
    </equal>

```

After the trigger part is satisfied, the rule engine executes the delete classification part of the rule:

```

    <function name="DG_DeleteFileClassification">
        <parameters>
            <evtSrcFilePath />
        </parameters>
        <return>
            <varint name="retStatus" scope="event" />
        </return>
    </function>
    <function name="DG_DeleteFileClassification">

```

```
<parameters>
  <evtDestFilePath />
</parameters>
<return>
  <variant name="retStatus" scope="event" />
</return>
</function>
```

In this rule, you should set **Execute Rule After Operation** Yes. This ensures that when a classification policy exists that tags the file based on the copy operation, the declassification rule is executed after the classification to remove the tags from the file.

You can combine this rule with any prompt, such as a Justification Prompt.

Note: To use a **Block** button on the prompt, you must set **Execute Rule After Operation** to No.

Function for Removing Tags From the Classification Stream

Using the function `DG_RemoveFileClassificationTags` with a source file removes selected tags from the file classification stream. This leaves the remaining tags in the stream on the file. The syntax for `DG_RemoveFileClassificationTags` appears in this example that removes the tag `tag_one` from the classification stream:

```
<function name = "DG_RemoveFileClassificationTags">
<parameters>
  <evtSrcFilePath/>
  <string value = "tag_one"/>
</parameters>
<return>
  <variant name = "statusResult" scope = "event"/>
</return>
</function>
```

To use `DG_RemoveFileClassificationTags`, specify the files to access and the tags to be removed in the body of the rule (refer to [Remove Selected Tags From the Stream](#)). In the example, the first function parameter points to a file. This could be source file (`evtSrcFilePath`), a destination file (`evtDestFilePath`), a static variable or

a string variable. The second parameter specifies the classification tags to remove. You can use a static string or a varstring variable array.

Remove Selected Tags From the Stream

This example defines two tags to remove from the stream—"top secret" and "private" for the current event. Note that the variable string name (varstring name) the function uses to get the list of tags to remove is fileTagsToRemove.

```
<and>
  <function name="DG_RemoveFileClassificationTags">
    <parameters>
      <evtSrcFilePath/>
      <varstring name="fileTagsToRemove" scope="event"/>
    </parameters>
    <return>
      <variant name="statusResult" scope="event"/>
    </return>
  </function>
  <add asArray="true">
    <varstring name="fileTagsToRemove" scope="event"/>
    <string value="top secret"/>
  </add>
  <set>
    <varstring name="fileTagsToRemove" scope="event"/>
    <string value="private"/>
  </set>
  <like expr="c:\dg\%">
    <evtSrcFilePath/>
  </like>
  <in>
    <evtOperationType/>
    <list>
      <constOpFileCopy/>
    </list>
  </in>
</and>
```

By placing this set of rule elements below the call to DG_RemoveFileClassificationTags, the elements create a string array named fileTagsToRemove that contains the tags "top secret" and "private." When the rule fires, DG_RemoveFileClassificationTags removes the tags for the event.

To remove specific tags, `DG_RemoveFileClassificationTags` requires the list of the tags to remove from the stream. When the operation succeeds, `DG_RemoveFileClassificationTags` returns zero (0) in `statusResult`.

Remove All Tags From the Stream (Declassify a File)

To remove all of the tags in a classification stream, use the function `DG_ResetFileClassification` and specify the path to files to declassify. You do not have to specify the tags. Use rule properties in the body of the rule, such as `evtSrcFilePath`, to define the file path from which to remove tags.

```
<function name="DG_ResetFileClassification">
  <parameters>
    <string value="pathToFileToDeclassify" />
  </parameters>
  <return>
    <variant name="statusResult" scope="event" />
  </return>
</function>
```

Remove One Tag

You can remove a single tag by specifying the tag name with a string value. For example, if a file stream includes one tag named `tag_one`, you could use parameters to identify the file from which to remove the tag, and to specify the tag to remove:

```
<function name="DG_RemoveFileClassificationTags">
  <parameters>
    <string value="pathToFileToDeclassify" />
    <string value="tag_one" />
  </parameters>
  <return>
    <variant name="statusResult" scope="event" />
  </return>
</function>
```

Use Classification Functions in Rules

To use a classification rule function, you add it to a rule, such as a control rule. When the rule triggers, the rule engine runs the function in line with the rule

logic. For example, suppose you have the following rule that fires when a user copies a Microsoft Word document:

```
<and>
<equal >
  <evtSrcFileExt/>
  <string value="docx" />
</equal >
<in>
  <evtOperationType/>
  <list>
    <constOpFileCopy/>
  </list>
</in>
</and>
```

Now add the function DG_ResetFileClassification to the rule, as shown here:

```
<and>
<function name="DG_ResetFileClassification">
  <parameters>
    <evtSrcFilePath />
  </parameters>
  <return>
    <variant name="statusResult" scope="event" />
  </return>
</function>
<equal >
  <evtSrcFileExt/>
  <string value="docx" />
</equal >
<in>
  <evtOperationType/>
  <list>
    <constOpFileCopy/>
  </list>
</in>
</and>
```

When the rule fires, the rule engine tests the event against the operation type and the source file extension. If both test evaluate to true, the function DG_ResetFileClassification resets the classification stream on the source file.

Similarly, if the you added the rule function `DG_RemoveFileClassificationTags` to the previous file copy rule, your rule might look like this:

```
<and>
<function name="DG_RemoveFileClassificationTags">
  <parameters>
    <evtSrcFilePath/>
    <varstring name="fileTagsToRemove" scope="global" />
  </parameters>
  <return>
    <variant name="statusResult" scope="event" />
  </return>
</function>
<!-- Define the tags to remove from the stream. -->
<add>
  <varstring name="fileTagsToRemove" scope="global" />
  <string value="top secret" />
</add>
<set>
  <varstring name="fileTagsToRemove" scope="global" />
  <string value="private" />
</set>
<equal>
  <evtSrcFileExt/>
  <string value="docx" />
</equal>
<in>
  <evtOperationType/>
  <list>
    <constOpFileCopy/>
  </list>
</in>
</and>
```

When this rule fires on the file copy, the rule engine tests the event against the source file extension. If that test is also true, `DG_RemoveFileClassificationTags` removes the tags “private” and “top secret” from the classification stream. Notice the rule elements that define the tags to remove.

Refresh the Classification Information for An Event

Use this function to ensure that the classification for an event that you changed after the event is reported correctly in reports in the DGM. “Change” means that

you might have used a function like `DG_TagFile` or `DG_ResetFileClassification` in a rule that modified the file classification information after the event occurred. In that case, the information in the event details reported by the Agent and stored on the server might not be correct. Adding the function `DG_RefreshEventClassification` to a rule updates the database information to update the classification data for the files in the event.

For example, suppose you copy a file that is not tagged with classification information (it is not classified) and you copy it.

Now, suppose you have a rule that runs after the file copy operation (a post operation), meaning you selected **Yes** for **Execute Rule After Operation** when you created your rule. In the rule you call `DG_TagFile` and add a tag to the destination file. Perhaps you intentionally used the post-operation file copy because you wanted to tag the destination file and the destination file would be available until after the file copy event was complete. When you copy the file, the Agent reports the file copy event with the unclassified file and then adds the new classification information to the file.

In this situation, the file in the event has no classification information, and that is what the Agent reports to the DG Server. However, you want to report the tags that you added using `DG_TagFile` as well. To do that, you add a call to `DG_RefreshEventClassification` to the rule. This updates the reported event file references to have the new classifications which includes the ones you added with `DG_TagFile`, and reports those to the DG Server. Now you have refreshed the event classification to reflect changes that occurred after the event was completed and reported by the Agent.

Process Flag Functions

Three functions enable you to write rules to modify the process flags for a process. The functions are:

- `DG_SetProcessFlags` — add process flags to the existing flags on a process in response to an event
- `DG_SetTimedProcessFlags` — set specific process flags on a process for a specified time in response to an event

- **DG_ClearProcessFlags** — remove specific process flags from a process in response to an event

The forms and syntax of the functions are:

DG_SetProcessFlags

```
<function name="DG_SetProcessFlags">
  <parameters>
    <currentEvent />
    <string value="flags_to_set_on_process" />
  </parameters>
  <return>
    <varbool name="statusResult" scope="event" />
  </return>
</function>
```

DG_SetTimedProcessFlags

```
<function name="DG_SetTimedProcessFlags">
  <parameters>
    <currentEvent />
    <string value="flags_to_apply_to_process" />
    <int64 value="timeout_in_msecs" />
  </parameters>
  <return>
    <varbool name="statusResult" scope="event" />
  </return>
</function>
```

DG_ClearProcessFlags

```
<function name="DG_ClearProcessFlags">
  <parameters>
    <currentEvent />
    <string value="flags_to_remove_from_process" />
  </parameters>
  <return>
    <varbool name="statusResult" scope="event" />
  </return>
</function>
```

A few restrictions apply to removing or adding flags to a process:

- After DG injects a process, adding the NI flag will not cause DG to remove its injected code from the process.
- The No Inject (NI) flag must be present when the process is starting to prevent the application from being injected.
- You can use `DG_SetProcessFlags` to add the NI flag to an application when the application starts.

These limitations mean rules can apply or remove the NI flag only when the process starts. Therefore, rules that add or remove the NI flag must use the property `evtOperationType` with the symbolic constant `constOpAppStart` to trigger the function that adds or removes the NI flag.

Caution: You can remove the NI flag from a process at anytime. The Agent will not at that time inject the process for monitoring. The process remains not injected and monitored. This detail is important when you are using the Propagate (PR) flag. If you remove the NI flag from a process that is not injected and that has the PR flag, and process starts a child process, the new (child) process will be injected. The Agent will continue not to monitor the parent process but will monitor the child process.

Example — Setting Process Flags

The following rule example uses `DG_SetProcessFlags` to apply the process flags NI and SK on `word.exe` when it is started.

```
<and>
<function name="DG_SetProcessFlags" >
  <parameters>
    <currentEvent />
    <string value="NI+SK" />
  </parameters>
  <return>
    <varbool name="result" scope="event"/>
  </return>
</function>
<equal>
  <curProcessImageName />
```

```
        <string value="winword.exe" />
    </equal >
    <equal >
        <evtOperationType />
        <constOpAppStart />
    </equal >
</and>
```

Example — Setting Timed Process Flags

The `DG_SetTimedProcessFlags` function sets the specified flag NFLT (no filtering) on the process `rundll.exe` for the length of time specified by `timeoutMS` (1000msec). After this rule runs, the Agent reports everything `rundll.exe` does for the next 1000msec. At that point, the NFLT flag is removed.

```
<and>
<function name="DG_SetTimedProcessFlags" >
    <parameters>
        <currentEvent />
        <string name="NFLT" />
        <int64 value="1000" />
    </parameters>
    <return>
    <bool name="result" scope="event" />
    <return />
</function>
    <equal >
        <curProcessImageName />
        <string value="rundll.exe" />
    </equal >
    <equal >
        <evtOperationType />
        <constOpAppStart />
    </equal >
</and>
```

Example — Clearing Process Flags

The `DG_ClearProcessFlags` function removes the specified flags from the flags assigned to the process in the event.

```
<function name="DG_ClearProcessFlags" >
    <parameters>
```

```

        <currentEvent />
        <string name="flags_to_remove" />
    </parameters>
    <return>
    <bool name="result" scope="event" />
    <return />
</function>

```

This example rule modifies the flags applied to the process `rundll.exe`.

```

<and>
<function name="DG_ClearProcessFlags" >
    <parameters>
        <currentEvent />
        <string value="NI+SK" />
    </parameters>
    <return>
        <bool name="result" scope="event" />
    </return>
</function>
<equal >
    <currentProcessImageName />
    <string value="rundll.exe" />
</equal >
<equal >
    <eventOperationType />
    <constOpAppStart />
</equal >
</and>

```

The rule clears the NI and SK (no inject and skipped) flags for `rundll.exe` whenever it is launched.

For information about process flags, refer to “Working with Process Flags” in *Digital Guardian Management Console User’s Guide*.

Registry Key Functions

You can write rules that tell the Agent to send a Registry Event when one of the following registry operations occurs:

- Key created
- Key renamed
- Key deleted
- Key value set
- Key value deleted

The Microsoft Windows registry is used heavily during computer operation so it is not reasonable to send a registry event for every operation of the types above to the rule system or to the DG Server. To make it feasible to monitor changes in the registry, you write rules that call a rule function to direct the Agent to monitor and report about particular keys and their subkeys. The following rule functions provide the ability to manage registry keys and values:

- `DG_EnableRegistryEventCaptureForCurrentProcess` — Enables capturing registry events for all keys and values, limiting event capture to the specified registry event type and for the specified amount of time on the current process.
- `DG_SetRegistryKeyEvents` — Sets up monitoring of one or more keys and their subkeys. Calls to this function add the given keys to the list of keys to monitor.
- `DG_SetRegistryKeyEventsEx` — Sets up monitoring of one or more keys and their subkeys, limiting event capture to the specified registry event type and for the specified amount of time. Applies to all running processes.
- `DG_RemoveRegistryKeyEvents` — Discontinues monitoring one or more keys that you set up to be monitored using `DG_SetRegistryKeyEvents`.
- `DG_SetRegistryValueEvents` — Sets up monitoring of one or more values.
- `DG_SetRegistryValueEventsEx` — Sets up monitoring of one or more values, limited to the specified registry event type and for the specified amount of time. Applies to all running processes.
- `DG_RemoveRegistryValueEvents` — Discontinues monitoring one or more values that you set up to be monitored using `DG_SetRegistryValueEvents`.

Support for Monitoring Registry Keys

Because changes to the Microsoft Windows registry might indicate the presence of unwanted processes or operations on the Agent computer, the Agent can be configured to monitor keys in the registry.

Caution: Use registry key monitoring with care. Monitoring all registry changes will affect performance on the Agent. Also, Agents send a registry event to the DGMC when you select Registry Event action in **User Actions to Log**. The Microsoft Windows OS makes frequent registry changes. Reporting all of them to the DGMC would flood the DGMC.

You can monitor important keys that relate to:

- Automatic start-type processes
- Installations
- User Access Control (UAC)
- Microsoft Internet Explorer
- Network operations

Note: Agents do not report registry events to the Server unless you have a rule configured to send an event or an event and alert.

Registry monitoring allows you to perform the following tasks and more:

- Control and profile applications
- Profile how an application uses the registry and note changes
- Monitor key areas of the registry for changes
- Identify DLL injection by looking for registry changes
- Search for indicators of compromise that appear in the registry
- Monitor and alert on IOCs, as indicated by the registry

Setting Up Monitoring

To capture a registry event, use `constOpRegistry` with `evtOperationType` and add one of the registry-related user functions to your registry rules:

- `DG_SetRegistryKeyEvents`
- `DG_SetRegistryKeyEventsEx`
- `DG_RemoveRegistryKeyEvents`
- `DG_SetRegistryValueEvents`
- `DG_SetRegistryValueEventsEx`
- `DG_RemoveRegistryValueEvents`

The following sample rule uses the `DG_SetRegistryValueEvents` function and monitors the registry only when a user logs on. This prevents the rule from setting up registry monitoring on every event.

You only need to set up the process once.

```
<and>
<!-- Calls to registry functions go here. -->
<!-- For example: -->
<function name="DG_SetRegistryValueEvents">
  <parameters>
    <string value="\registry\machine\SOFTWARE\Microsoft\
      WindowsNT\CurrentVersion\AeDebug\Debugger"/>
  </parameters>
</function>
<function name="DG_SetRegistryValueEvents">
  <parameters>
    <string value="\registry\user\*\Software\Microsoft\Office\*\
      Common\Internet\UseRWHLinkNavigation"/>
  </parameters>
</function>
<!-- Setup to monitor registry after logon event. -->
  <equal>
    <evtOperationType/>
    <constOpLogon/>
  </equal>
</and>
```


Registry Key Monitoring Operation

The Agent monitors the list of registry keys that you specify by one or more calls to `DG_SetRegistryKeyEvents`.

When one of the following actions affects a registry key on the list of keys to monitor, the Agent generates a registry event and sends the event to the rule system for evaluation by the control rules.

Note: Classification rules, trusted process, or process blocking rules are not evaluated for registry events.

- A key or one of its child keys is deleted
- A child key is created for a key
- A key or a child key is renamed

Registry Value Monitoring Operation

Similar to the registry key support, one can receive an event whenever one or more registry values are added/modified (set), or deleted.

Registry events have a few of their own properties (in addition to the common properties) you can use in rules.

Property Name	Type	Description
evtRegistryOperationType	INT8	The operation type that occurred. This is defined by an enumeration, refer to the tables in “evtRegistryOperationType” on page 346 .
evtRegistrySrcPath	WSTRING	The source path for the operation. This always contains the full path to the registry item the operation was performed on.

Property Name	Type	Description
evtRegistryDestPath	WSTRING	The destination path for the operation. This is an empty string except for “rename” operations.
evtRegistryValueType	INT8	Specifies the value type the operation was on. This is an enumeration, see “Registry Value Types”.
evtRegistryValue DWORD	INT32	The actual DWORD of the value involved in the operation. This is the final value in the case of “set” operations.
evtRegistryValue QWORD	INT64	The quad word of the value involved in the operations.
evtRegistryValue StringSZ	WSTRING	The string value involved in the operation.
evtRegistryValue MultiStringSZ	WSTRING VECTOR	List of strings involved in the multi-string operation. This can be quite long. The DG Server will hold roughly the first 2048 bytes of data in the DB.

evtRegistryOperationType

Operation	Value	Rule Property/Constant Name
Registry Create Key	1	constRegistryCreateKey
Registry Rename Key	2	constRegistryRenameKey
Registry Delete Key	3	constRegistryDeleteKey
Registry Delete Value	4	constRegistryDeleteValue
Registry Set Value	5	constRegistrySetValue

Example — Activate Registry Events for a Specified Value

This rule monitors the key `\registry\machine\software\google\` after Google Chrome browser starts.

```
<and>
  <function name="DG_SetRegistryKeyEvents">
    <parameters>
      <string value="\registry\machine\software\google\" />
    </parameters>
  </function>
  <equal>
    <currentProcessImageName>
      <string value="chrome.exe" />
    </equal>
  <equal>
    <eventOperationType/>
    <constantOperationStart/>
  </equal>
</and>
```

Any time the user starts Chrome browser, the example rule tells DG to start to monitor the key `\registry\machine\software\google\`. After the `DG_SetRegistryKeyEvents` call above, anytime the key is modified, a registry event is generated and processed by any existing control rules. Generally, you will have a separate rule set up to process the registry event and perhaps block the operation.

Best practices suggest that you create a component list that contains the registry keys to monitor.

Example — Rule to Process Registry Event

Based on the rule in section [“Example — Activate Registry Events for a Specified Value” on page 347](#), the below rule evaluates to True when something or someone changes or deletes the test value.

```
<and>
  <like expr="%\test" >
    <eventRegistrySrcPath>
  </like>
</and>
```

```
        <evtOperationType />  
        <constOpRegistry />  
    </equal >  
</and>
```

Incident Remediation Function

It is no longer sufficient to identify risks and problems across your enterprise. Now it is necessary to respond to identified problems or risks to mitigate or remove them. One way to do this is to quarantine (isolate) files that you identify as putting your data at risk or that need to be controlled to prevent data loss. DG provides a user function to use to quarantine any file you need to manage.

Function for Quarantining Files

Being able to quarantine a file represents one form of incident remediation or compliance. Quarantine can range from isolating a file into a secure location (moving the file) to deleting the file altogether. DG provides the ability to quarantine files by moving them to a secure location triggered by rules.

File quarantine can provide a solution for a number of common data loss use cases:

- Allow compliance or information security officers to move their organizations closer to compliance with regulations like PCI-DSS, HIPAA or GDPR.
- Allow information security officers to protect intellectual property that may have propagated out of secure repositories.
- Allow information security officers to quarantine potentially malicious files.

File quarantine supports this by enabling the officers to identify files containing confidential data and to move the files to a location with more stringent controls or apply controls to the file in place, such as applying encryption.

Potential Use Cases

These are a few examples of where you might use file quarantine.

- You want to audit your organization workstations for PCI-DSS compliance, for example.
- You run a discovery scan and the scan returns a list of workstations and files that contain PCI data. You examine the list and perform manual reme-

diation on files. You might create a report based on gathered data and submit it to your auditor.

- You run a discovery scan and copy all files that are potentially out of compliance to a secure location. This is largely used by organizations that want to review the accuracy of their DLP policies before moving to a more drastic remediation step.
- You run a discovery scan and move any file that matches the DLP policy to a secure location. As part of the process, the system leaves a breadcrumb file in place of the file that was moved.
- You run a discovery scan and use a script to call an application to encrypt files containing confidential data in place.
- A user in your enterprise creates a file that corporate policy requires be stored in a secure location.

Using the scanner is an ideal way to run this function. The scanner can scan all files and quarantines any file it finds that meet the quarantine criteria. If you use the scanner, best practice is to look for File Create (constOpFileCreate) operations to trigger your quarantine rule. Using File Create reduces the number of duplicate events captured and reported by the Agent during scanning.

However, you can use the rule function in response to a user operation as well. In that instance, the rule with the function should run as a POST operation (select **Yes** for **Run Rule After Operation** when you create your rule). **Run Rule After Operation** is available as an option for control rules. It is not available for classification rules. To make classification rules run as post operations, add `evtIsAfterOperation` to the XML for your rule. For more about post operations, refer to [“Using evtIsAfterOperation” on page 108](#).

In the above use cases use two new terms—**secure location** and **breadcrumb file**.

Secure Location

Any location in your enterprise that you identify as secure for storing quarantined files.

Bread Crumb File

Bread crumb files are left in place of a source file when a quarantine action relocates the source file so the user cannot access it.

Bread crumb files are text (.txt) files with the same name as the source file that was moved, with . quarantine. txt appended. For example, a file named acmesecret. docx generates a breadcrumb file with the name acmesecret. docx. quarantined. txt.

It is very important that your system not quarantine breadcrumb files. That is the reason breadcrumb files have the unique naming convention and the quarantine process checks the file name to quarantine. If it ends in quarantine.txt, the quarantine process stops—the file is not quarantined.

This code example skips quarantine.txt files in your rules:

```
<not>
  <like expr = "%. quarantine. txt">
    <evtSrcFilePath/>
  </like>
</not>
```

Rule Function Syntax

The user function for quarantining a file is named DG_QuarantineEX. For input parameters it takes:

Input Parameters	Description
Current rule ID	Rule ID of the rule that contains this function.
Source file path	Path to identify the file to quarantine. Use a UNC path or a path to the file with a drive letter. Enter the path with mixed case to represent the desired path accurately. Although DG uses lower case paths, entering the mixed case path makes it easier to read the path and file name.
Destination file path	Path to location to quarantine the file. You provide the full path to use for the destination file.

Input Parameters	Description
Bread Crumb Text	<p>This is a string that you specify as the content in the breadcrumb file, replacing the original file content. For example, you might use this:</p> <pre><set> varstring name="breadcrumbText" scope="local" /> <string value="This file has been quarantined. You can find the original file on the quarantine server at &lt;QuarantineDestPath>"/> </set></pre> <p>where QuarantineDestPath is a predefined expandable name that contains the path to the destination file from the function.</p>
Username	User to impersonate to perform the action on the file in the quarantine folder. The user has to have permission to write to the quarantine folder. Use the format for username that your enterprise requires, often username@domain.
Password	Password for username.

This is the function:

```
<function name = "DG_QuarantineEx">
  <parameters>
    <currentRuleId/>
    <evtSrcFilePath/>
    <string value = "Path to quarantine location"/><!--Dest. file-->
    <string value = "String that explains the file was quarantined"/>
    <string value = "user_name"/><!--User to impersonate-->
    <string value = "user_password"/><!--User password-->
  </parameters>
  <return>
    <varint name = "Var-Quarantine_Result" scope = "event"/>
  </return>
</function>
```

DG_QuarantineEX uses the standard file system MoveFile command to relocate the quarantined file. Quarantining a file is a synchronous operation. If the

destination file path exceeds 259 characters, the function drops elements from right to left until the path is less than 259 characters.

For flexibility, you could use rule variables to define the quarantine directory and the breadcrumb text to use.

Example—Quarantine Classified Files by Extension

This example quarantines files that are created, copied or saved with classified content and selected extensions (.docx, .doc and .txt) to a repository whose location is specified in the function. For this rule to run effectively, it should be a POST operation.

The repository path is stored in Var-Quarantine-Server-Target-Share. The file to quarantine is in evtSrcFilePath. The source file is captured by the constOpFile... operations.

```
<and>
<function name = "DG_QuarantineEx">
<parameters>
<currentRuleId/>
<evtSrcFilePath/>
<string value = "path to quarantine directory" preserveCase = "1"/>
<string value = "This is breadcrumb text."/>
<string value = "admin@domain" preserveCase = "1"/>
<string value = "password" preserveCase = "1"/>
</parameters>
<return>
    <varint name = "Var-Quarantine_Result" scope = "event"/>
</return>
</function>

<i n>
    <evtSrcFilePolicyTag />
    <list>
        <string value="CONFIDENTIAL" />
        <string value="context_tag" />
    </list>
</i n>

<not>
    <like expr = "%.quarantined.txt">
        <evtSrcFilePath/>
    </like>
</not>

<i n>
    <evtOperationType/>
<list>
    <constOpFileCreate/>
    <constOpFileSaveAs/>
    <constOpFileCopy/>
</list>
</i n>
</and>
```

The result—success (1) or failure (0)—of the quarantine process is returned in Var-Quarantine_Result.

Agent Configuration Information Functions

A set of rule functions enables you to determine or test the status of an Agent in a rule:

- DgMaster_IsAgentRunning
- DgMaster_IsFeatureEnabled
- DgMaster_IsStealthOn
- DgMaster_IsTamperResistOn

Test if the Agent is Running. If user mode DgAgent.exe is running, this returns 1. This is a kernel mode test, not user mode.

```
<function name="DgMaster_IsAgentRunning">
  <return>
    <varbool name="IsAgentRunning" scope="global" />
  </return>
</function>
```

Test Agent Feature Status in Rules

DG provides a rule function that lets you use rules to determine whether a feature is enabled on an Agent. The function is DgMaster_isFeatureEnabled.

To use this function, add the function to your rule with the appropriate feature code from the following table. It tells you if the feature that matches the integer value is enabled. It will not tell you whether a reboot is required to enable the feature or module.

This table lists the features and modules available in DG (both current and older) and the integer values you use to specify the feature of interest in the function. The hexadecimal value does not apply in the function. The hexadecimal value is there for reference purposes only.

Feature	Integer Value	Hexadecimal Value
Adaptive Content Inspection	16	0x00000010
Adaptive Data Inspection	262144	0X00040000

Feature	Integer Value	Hexadecimal Value
Advanced Persistent Threat	1024	0x00000400
Application Remediation	1	0x00000001
Exchange Active Server	131072	0X00020000
Investigation Module	65536	0X00010000
Microsoft SharePoint	32768	0X00008000
Removable Media Encryption	512	0x00000200
User Classification for Desktop	8192	0x00002000
User Classification for Message	2048	0x00000800
User Classification for Office	4096	0x00001000

For example, executing the following code tells you whether RME is enabled (the integer value of 512 corresponds to RME, hexadecimal value 0x00000200).

```
<function name="DgMaster_IsFeatureEnabled">
  <parameters>
    <int value="512" />
  </parameters>
  <return>
    <varbool name="RMEEnabled" scope="global" />
  </return>
</function>
```

The function code assigns the Boolean 1 (true) to variable RMEEnabled when the RME feature is enabled or on.

To test for more than one feature in the function, add the integer values together. Then you can tell whether any of the features is enabled. For example, to test for both ACI and RME, add the values 16 and 512 = and use 528 in the function:

```
<function name="DgMaster_IsFeatureEnabled">
  <parameters>
    <int value="528" />
  </parameters>
  <return>
    <varbool name="MyFeaturesEnabled" scope="global" />
  </return>
```

```
</function>
```

This function assigns `MyFeaturesEnabled = 1` if either ACI or RME is enabled on the Agent computer.

This function tests whether the Agent is in stealth mode.

```
<function name="DgMaster_IsStealthOn">
  <return>
    <varbool name="StealthEnabled" scope="event" />
  </return>
</function>
```

You can test whether the Agent is in tamper resist mode.

```
<function name="DgMaster_IsTamperResistOn">
  <return>
    <varbool name="TamperEnabled" scope="event" />
  </return>
</function>
```

Restore a Rule Variable to Unused State

To restore any rule variable immediately to the state you set initially, use the `dispose` element. In particular, this can be very useful for resetting persistent variables that otherwise are difficult to remove.

Here is an example that uses `dispose` to reset the global scope variable `varName` to the unused state, freeing the data associated with the variable. That makes the variable empty:

```
<dispose name="varName" scope="global" />
```

You can immediately return a rule variable to its original unused state at any point by adding a `<dispose>` element.

Check for an Active Process

The function `DG_IsProcessActive` enables you to determine whether a specific process is active by checking whether the ID for the process is still in the process cache.

This following function checks if a process ID is active.

```
<function name="DG_IsProcessActive">
  <parameters>
    <varint name="processId" scope="global" />
  </parameters>
  <return>
    <varbool name="fn_result" />
  </return>
</function>
```

Event Caching Functions

A 5-minute cache collects all non-alerted event and non-selected user data leading to a rule firing and forwards it to the DG Server.

DG_SendCachedData

The function DG_SendCachedData causes the event cache to flush to the server. It takes an input variable eventsToSendMask. Instead of sending all events in the AggregateEventCache to the server, it sends only those that match the mask.

To see all events, pass in a -1. The following example function returns the number of events flushed to the server.

```
<function name="DG_SendCachedData">
  <!-- send all events in the cache to the server -->
  <parameters>
    <string value="-1"/>
  </parameters>
  <return>
    <variant name="count_sent" scope="event"/>
  </return>
</function>
```

To see specific events that the Agent has recorded, you can pass in activity logging values as a string value. For a list of the activity logging values and their associated events, refer to “Digital Guardian Agent Configuration Settings” in the *Digital Guardian Management Console User’s Guide*. The following example specifies sending several types of file-related events.

```
<function name="DG_SendCachedData">
  <parameters>
    <string value="11, 17, 18"/>
  </parameters>
  <return>
    <variant name="count_sent" scope="event"/>
  </return>
</function>
```

To see only File Copy events, you would pass in 11 in the eventsToSendMask.

```
<function name="DG_SendCachedData">
```

```
<!-- send all events in the cache to the server -->
  <parameters>
    <string value="11"/>
  </parameters>
  <return>
    <variant name="count_sent" scope="event"/>
  </return>
</function>
```

DG_ClearCachedData

The function DG_ClearCachedData clears the cache before running a particular test.

```
<function name=" DG_ClearCachedData ">
  <!-- clear all events in the event mask -->
  <parameters>
  </parameters>
  <return>
    <variant name="count_cleared" scope="event"/>
  </return>
</function>
```

DG_AggregationControl

The function DG_AggregationControl allows the rule function to indicate how the custom data of the event should be aggregated if there are redundant events or if no aggregation should be performed.

The function takes two parameters, `currentEvent` and one of the following enumerated values:

- `KeepCurrent = 0` // the custom data taken from the newest event. This is the default value.
- `NoAggregate = 3` // performs no aggregation, just adds the event to the cache.

The following example adds the current event to the cache without performing aggregation.

```
<function name="DG_AggregationControl ">
```



```
<parameters>
  <currentEvent/>
  <i nt value="3"/>
</parameters>
<return>
  <varbool name="resul t" scope="event"/>
</return>
</functi on>
```

Functions To Capture Files

DG provides two functions to capture files, `DG_CaptureFile` and `DG_CaptureFileByPath`.

DG_CaptureFile Function

The function `DG_CaptureFile` triggers the file capture process. `DG_CaptureFile` requires seven input arguments:

- **Enable Hashing** — Specifies whether to apply hashing to the captured file to enhance security.
- **Hash Type** — Specifies the type of hash to apply to the captured file.
- **Disposition** — Specify what to capture—source, destination, or both. (Both [4] requires both source and destination files to be available to the rule.)
- **Capture Alg** — Specify whether to capture the file synchronously or asynchronously.
 - In synchronous mode, the Agent temporarily suspends the user operation while capturing the file. This can be critical when you are capturing files that may not be available after an operation, such as when the user is deleting a file.
 - Asynchronous mode lets the Agent queue the request to capture the file when the file is available. For example, if a file is locked when the Agent tries to capture it, asynchronous mode captures the file when the lock is no longer in place.
- **Capture Mode** — Specify whether to capture the file record only or the file and file record.
- **Transfer Mode** — Specify whether the Agent uploads the file to the Server automatically or waits for a request for the file. This option requires a valid license key for the Investigation Module.
- **Max Capture File Size in MB** — Specify the maximum size of the file to capture into the Agents file capture buffer.

The following example rule shows the input arguments and their required order in the file capture function.

Example Rule

Here's a rule that captures the source files that have the .doc extension involved in recycle or delete events. The first part of the example is the file capture function `DG_CaptureFile`.

```
<and>
<function name="DG_CaptureFile">
  <parameters>
    <currentEvent/>
    <currentAssociatedFileInfo/>
    <currentRuleId/>
    <int value="2"/> <!-- Enable Hashing: Def=0, Enabled=1, Disabled=2 -->
    <int value="0"/> <!-- Hash Type: Def=0, MD5=104, SHA1=103, SHA256=102, SHA512=101
-->
    <int value="2"/> <!-- Disposition: Def=0, Src=2, Dst=3, Both=4 -->
    <int value="1"/> <!-- Capture Alg: Def=0, Synch=1, Async=2 -->
    <int value="2"/> <!-- Capture Mode: Def=0, Record_Only=1, Record_And_File=2 -
->
    <int value="2"/> <!-- Transfer Mode: Def=0, Auto_Upload=1, On_Demand=2 -->
    <int value="0"/> <!-- Max Capture File Size in MB: Def=0 -->
  </parameters>
  <return>
    <varint name="fc_result" scope="event" />
  </return>
</function>
<in>
<evtSrcFileExt/>
<list>
  <string value="doc"/>
</list>
</in>

<in>
  <evtOperationType/>
  <list>
    <constOpFileRecycle/>
    <constOpFileDelete/>
  </list>
</in>
</and>
```

As usual, the rule engine executes the rule from the bottom up. To trigger the file capture function and capture the file, all of the rule elements must evaluate to true.

Note: You must list the values in the parameters section in the order shown.

Function `DC_CaptureFile` requires seven input arguments, detailed in the following table:

Input Argument	Valid Capture Settings	Description
Enable File Hashing	0 - Default 1 - Enabled 2 - Disabled	Specifies whether to apply hashing to the captured file to enhance security
Hash Type	0 - Default 101 - SHA512 102 - SHA256 103 - SHA1 104 - MD5	Specifies the type of hash to apply. MD5 is not available in FIPS mode.
Disposition	0 - Default 2 - Source File 3 - Destination File 4 - Source and Destination files	Specify what to capture—source, destination, both. If both (4) is requested, the function will work only if both source and destination files are available to the rule.
Capture Alg	0 - Default 1 - Synchronous 2 - Asynchronous	Specify whether to capture the file synchronously or asynchronously.
Capture Mode	0 - Default 1 - Record Only 2 - Record and File	Specify whether to capture only the file record or the file and file record.
Transfer Mode	0 - Default 1 - Automatic Upload 2 - Wait for Request	Specify whether the Agent uploads the file to the Server automatically or waits for a request for the file.
Max Capture File Size	0 - Default Value in MB for Max File Size	Specify the maximum size of the file to capture.

Set the defaults for these configuration values in the File Capture settings in the Core Settings configuration resource for the workstation. The example function

shown sets capture values that may be different from the defaults. In the “int val ue=” XML elements, the Def=0 comment indicates that when you set “int val ue” to 0, file capture uses the default value in the File Capture settings in the Core Settings configuration resource applied to the workstation.

Two notes apply to the file capture function:

- You do not need to include the comment text for each input argument. Best practices suggest that you keep the text to remind you of the argument values and their meanings.
- If you set Transfer Mode to Automatic Upload, but you do not have the Investigation Module enabled, Transfer Mode defaults to Wait for Request.

The order and number of arguments must be as shown in the example rule on page 317. If you change the order, your rule will not work as intended.

Rules you develop to capture files must have the function `DG_CaptureFile`, either as part of the rule or available as a component rule.

DG_CaptureFilePath Function

`DG_CaptureFilePath` triggers the file capture process for a specific file, using the full file path. This function operates similarly to `DG_FileCapture`, but where `DG_FileCapture` requires an event associated with a file, `DG_CaptureFilePath` provides a means of using events that do not have associated files, such as app starts.

`DG_CaptureFilePath` is especially useful when writing ATP rules. When a specific Advanced Threat Protection (ATP) rule fires, `DG_CaptureFilePath` allows you to capture the running executable and pass the file to the DGMC for review. You can capture PE, BAT, DLL, SYS, and EXE files and store them in a password-protected, zipped/encrypted format on the DG Server. You can specify the password in the DGMC as a setting.

`DG_CaptureFilePath` takes the same input arguments as `DG_CaptureFile`, with these exceptions:

- `DG_CaptureFilePath` does not use the `Disposition` argument, because no source or destination files are involved.

- DG_CaptureFileByPath takes the following additional input arguments:
 - Current Event
 - File Path To Capture
 - Current Rule ID

The following table shows the arguments that the DG Agent uses when it captures a file by path. The Valid Capture Settings column lists the capture settings that the Agent uses. You must specify Max Capture File Size in rules. All other settings are used by default and do not need to be specified in rules.

Input Argument	Type	Capture Settings	Description
Enable Hashing	integer	1 - Enabled	Applies hashing to the captured file to enhance security
Hash Type	integer	102 - SHA256	Specifies the type of hash to apply.
Capture Algorithm	integer	1 - Synchronous	Captures the file immediately.
Capture Mode	integer	2 - Record and File	Captures the file and the file record.
Current Event	object		Gets the event that triggers the file capture.
Current Rule ID	GUID		ID of the rule that triggers the file capture.
File Path To Capture	string		Path of the file to capture.
Transfer Mode	integer	1 - Automatic Upload	Uploads the file to the Server automatically.
Max Capture File Size	integer	0 - Default Value in MB for Max File Size	Maximum size of the file to capture into the Agent's file capture buffer.

Example Rule

```
<and>
  <function name="DG_CaptureFileByPath">
    <parameters>
      <currentEvent/>
      <curProcessFilePath/> <!-- path for current process -->
      <currentRuleId/>
      <int value="0"/> <!-- Max Capture File Size in MB: Def=0 -->
    </parameters>
    <return>
      <variant name="fc_result" scope="event" />
    </return>
  </function>
</in>
  <evtOperationType/>
  <list>
    <constOpAppStart/>
  </list>
</in>
</and>
```

Encrypting and Decrypting Files With Functions

Digital Guardian provides functions for encrypting and decrypting files based on rules. In most cases, DG file encryption and decryption take place transparently. With these functions, you can write rules that decrypt or encrypt files based on an event that triggers the rules. The functions are:

- **DG_EncryptFile** — encrypt a file when an event triggers a rule that includes this function. The function takes the file path and file name as an input argument.
- **DG_DecryptFile** — decrypt a file when an event triggers a rule that includes this function. The function takes the file path and file name as an input argument.

These are the function forms:

```
<function name="DG_DecryptFile" >
  <parameters>
    <string value="filePath" />
  </parameters>
```



```

    <return
      <varbool name="result" scope="event" />
    </return>
  </function>

```

and

```

<function name="DG_EncryptFile" >
  <parameters>
    <string value="filePath" />
  </parameters>
  <return
    <varbool name="result" scope="event" />
  </return>
</function>

```

Here is a rule that encrypts an unencrypted classified file when the user closes the file. For this rule, you should set Continue Rule Evaluation to No, set Execute Rule After Operation to Yes and set the rule action to Continue.

```

<and>
  <function name = "DG_EncryptFile">
    <parameters>
      <evtSrcFilePath />
    </parameters>
    <return>
  <
    <varint name = "fc_result" scope = "event" />
  </return>
</function>
  <equal >
    <evtSrcFileIsEncrypted />
    <bool value = "false" />
  </equal >
  <equal >
    <evtSrcFileIsClassified />
    <bool value = "true" />
  </equal >
  <in>
    <evtOperationType />
    <list>
      <constOpFileClose />
    </list>
  </in>
</and>

```

This example rule decrypts encrypted classified files when the user opens the files.

```
<and>
  <function name = "DG_DecryptFile">
    <parameters>
      <evtSrcFilePath/>
    </parameters>
  <return>
    <variant name = "fc_result" scope = "event"/>
  </return>
</function>
<equal >
  <evtSrcFileIsEncrypted/>
  <bool value = "true"/>
</equal >
<equal >
  <evtSrcFileIsClassified/>
  <bool value = "true"/>
</equal >
<in>
  <evtOperationType/>
  <list>
    <constOpFileOpen/>
  </list>
</in>
</and>
```

CHAPTER 12 Component Lists

Some rule use cases require you to manage potentially large and dynamic lists of values that can be referenced by rules. This is especially the case for advanced threat protection where you might want your Agents to act on lists of Indicators of Compromise (IOCs) received from network appliances or Security Information and Event Management software (SIEM) within your environment, from feeds, or from lists generated as part of an active incident response effort.

To make it easier to write rules that allow your Agents to act on lists of IOCs, DG provides component lists. Component lists are lists that you reference from rules, similar to arrays in a programming language. By using component lists, you can reduce the time required to write and maintain your rules and protect your data more successfully. For example, you can create component lists that contain IOCs to reference in your rules to alert on or block activities to prevent, detect, or contain a cyber attack.

Terminology

When you work with component lists, you should be familiar with the following terms:

Compile — The process by which the DG Server converts a component list to XML so that the list can be downloaded to and used by the Agent.

Component List — A dynamic list of strings, often associated with IOCs. DG rules can use component lists as rule variables in rules. Multiple rules can refer to the same component list. Component lists are dynamic: Your DG Server can distribute updated lists to your Agents on a schedule or as needed so that the list content stays current as you gather or retire IOCs, without having to modify rules.

Component List Application Programming Interface (API) — A set of stored procedures that allow you to manage lists of values from external sources and construct component lists that contain the value information.

List Compile Job — The DG Server job that compiles your component lists into a format that rules can use. The job also prepares the list to be deployed.

Member — An entry on a component list. List members have values, an optional description, and attributes to manage their life cycle.

Feed — An external source of IOCs that you might add to a component list. The content of the feed is information that identifies possible risks to your information or network. In some cases, feeds create component lists automatically. For example, integration with FireEye network device allows the FireEye information to be part of a component list automatically. For more information about feeds, refer to “Advanced Threat Protection” in the *Digital Guardian Management Console User’s Guide*.

Types of Component Lists

DG supports a variety of component list types, based on the content you expect to use in your rules. Because the lists contains strings that can become rule variables, the rules that will use the lists determine the type of content in the lists. For example, rules that trigger on email addresses would expect component lists that contain email address strings.

You can create the following types of lists:

- Domain — Contains strings that represent domains. You would use domain lists in rules that monitor and control access to domains.
- Email — Contains email address strings that your email-based rules could use to monitor and control email messages.
- Generic — Contains any types of strings. You might use a generic list when you have to provide a variety of string formats to a rule. For example, writing rules for Microsoft Sharepoint requires at least two different file path formats. A generic list could contain both types of file path strings to support your SharePoint rules.
- IP Address — Contains IP address strings that your network-based rules use to monitor and control access to internet addresses defined by their IP addresses.
- MD5 — Contains MD5 strings that your process or application-based rules could use to monitor and control the operation of processes or applications on the Agent computer.
- SHA1 Hash — Contains SHA1 hash values that your advanced threat protection rules should trigger on. These represent Digital Guardian hash values of processes or applications that may perform operations that put your enterprise at risk.
- SHA256 Hash — Contains SHA256 hash values that your advanced threat protection rules should trigger on. These represent Digital Guardian hash values of processes or applications that may perform operations that put your enterprise at risk.

Sources of Component Lists

You can create component lists from any of the following sources:

- DGMC
- Feeds
- Component list API
- Third parties products, including FireEye, ArcSight, Palo Alto, and Blue-Coat. For more information on third-party sources, refer to “Advanced Threat Protection” in the *Digital Guardian Management Console User’s Guide*.

Component List Size

Digital Guardian 7.3 and later Agents support large component lists — component lists that contain over 1,000 entries. Large component lists can contain up to 1,000,000 entries. DG 7.3 and later Agents support checking up to 5,000,000 entries per operation. You can use any combination of lists to reach the 5,000,000 maximum, such as 5 lists of 1,000,000 entries each, 500 lists of 10,000 entries each, and so on.

The DG Server sends only unique component list entries to the Agent. If you have component lists that share some entries, each shared entry is counted only once.

Note: DG supports large component lists on the Agent for Windows. The Agent for Linux and Agent for macOS do not currently support large component lists.

Limitations When Using Older DG Agents

- Pre-DG v7.3 Agents do not support policies containing component lists of over 1,000 entries. The DG Agent will receive policies that reference those lists, but the policies will not have the component list information and will not function correctly.
- If a component list that contains the pre-DG 7.3 limit of 1,000 entries or fewer has already been applied to a policy on computers running pre-DG 7.3 Agents, you cannot modify the list to increase its size beyond the 1,000-item limit.
- If you attempt to apply a policy that contains large component lists to a dynamic machine group or user group with pre-DG 7.3 Agents, the DG Server prevents the action and presents an error message.

Note: The Dynamic Group Sync job could cause computers running an older Agent version to become associated with dynamic groups that have policies with rules referencing large component lists. In such cases, DGMC sends an email message to the administrator. The administrator must have subscribed to receive notifications.

DG License Requirements for Component Lists

Your ability to work with component lists depends on DG product license.

- If you have a Core license only, you cannot edit or create component lists. The Actions option on the Component Lists page in the DGMC is not available. With a Core license, you get component lists from the Content Service.
- If you have a DLP license, you have full access to components lists. You can create and edit component lists unless you downloaded the lists from the Content Service that are not marked Customer Can Edit.
- If you have an ATP license, your component list capabilities depend on the source of the list, as shown in this table:

Component List Source	Actions Menu Available?	Component List Editable?	Detail on how list was created
You created the component list	Yes	Yes	You created the component list using Actions > Create List
DG created the component list with read and write permissions	Yes	Yes	DG created the component list and marked it "Customer Can Edit"
DG created the component list with read-only permission	Yes	No	DG created the component list and did not mark it "Customer Can Edit"
You created the list from a feed	No	Yes	Customer created a feed that created the component list
DG created the threat feed and thus the list	No	Yes	DG created a threat feed that created the component list.

Format of Component Lists

Component lists are XML files that contain members as XML elements. The general form of the compiled lists is:

```
<?xml version="1.0" encoding="UTF-8"?>
<componentlists>
```

```
<componentlist Id="0593bbf4-6122-4a73-864d-cbaf5b1ad7e8" Name="Bad IP addresses"
Description="" ContentType="IPAddress" Status="Active" MaxSize="1000" Create-
Date="2/6/2014 9:39:23 PM" ModifiedDate="2/6/2014 9:39:23 PM" ScheduleType="OnDe-
mand" ScheduleInterval="0" ScheduleTime="-1" VersionNumber="1"
PolicyContainerId="00000000-0000-0000-0000-000000000001">
<Items>
<Item Id="" Value="" CreateDate="" ModifiedDate="" Status="" SourceType="" />
<Item Id="" Value="" CreateDate="" ModifiedDate="" Status="" SourceType="" />
<Item Id="" Value="" CreateDate="" ModifiedDate="" Status="" SourceType="" />
...
</Items>
</componentlist>
</componentlists>
```

where each `<Item . . . />` XML entry is one member of the list—an email address, a domain name, an IP address, Digital Guardian MD5 hash value, or other hash value or string.

The following sample compiled list file shows three IP addresses in an IP Address component list:

```
<?xml version="1.0" encoding="UTF-8"?>
<componentlists>
<componentlist Id="0593bbf4-6122-4a73-864d-cbaf5b1ad7e8" Name="Bad IP addresses"
Description="" ContentType="IPAddress" Status="Active" MaxSize="1000" Create-
Date="2/6/2014 9:39:23 PM" ModifiedDate="2/6/2014 9:39:23 PM" ScheduleType="OnDe-
mand" ScheduleInterval="0" ScheduleTime="-1" VersionNumber="1"
PolicyContainerId="00000000-0000-0000-0000-000000000001">
<Items>
<Item Id="00000000-0000-0000-0000-000000000000" Value="10.11.200.11" CreateDate="2/
6/2014 9:39:23 PM" ModifiedDate="2/6/2014 9:39:23 PM" Status="Active" Source-
Type="DGMC" />
<Item Id="00000000-0000-0000-0000-000000000000" Value="192.168.1.1" CreateDate="2/
6/2014 9:39:23 PM" ModifiedDate="2/6/2014 9:39:23 PM" Status="Active" Source-
Type="DGMC" />
<Item Id="00000000-0000-0000-0000-000000000000" Value="10.1.20.1" CreateDate="2/6/
2014 9:39:23 PM" ModifiedDate="2/6/2014 9:39:23 PM" Status="Active" Source-
Type="DGMC" />
</Items>
</componentlist>
</componentlists>
```


Component List Storage on the DG Agent

To save component lists on the DG Agent computer, you need to add a `<componentListsSaveToDisk>1</componentListsSaveToDisk>` entry to your DG Agent `config.xml` file. You can supply this entry either at Agent installation time or by using a Custom Configuration Resource applied to a Dynamic Group that your Agent belongs to. This entry instructs the Agent to save an XML file for each component list that you create to the `C:\Program Files\dgagent\LISTS` directory.


Creating and Updating Component Lists

You create component lists from the DGMC, feeds, third-party sources, and component list APIs. This section describes creating and updating component lists in the DGMC. For information about using component list APIs, refer to [“Using the Component List API” on page 395](#).

The Create List option in the DGMC provides an interface for creating any type of component list.

Create Component Lists

To create small lists with only a few entries, use the DGMC component lists feature.

1. Hover over Policies and, under Lists, select **Component Lists**. The Component Lists page opens, showing your existing Component Lists.
2. To create a new list, hover over  in the menu bar and select the type of list to create from Create List. Choose one of the following:
 - Domain
 - Email
 - Generic
 - IP Address

- MD5 Hash
- SHA1 Hash
- SHA256 Hash

3. Enter the following information as needed to create your list:


- **Name** — (required) Enter a name for your list. The name must be unique on this DGMC instance. Do not use special characters, such as & or #, in the name.
- **Description** — Enter a brief description of the list and its content.
- **Content Type** — (Read-only setting unless you are creating a generic component list). Shows the type of list you are creating.
- **Status** — Select the status for your list, either Active or Inactive.
- **Max List Size** — (required) Specify how large the list can grow on the Agent (1- 1000000). When the list reaches the maximum size, you cannot add more members until you change one or more existing members' status from Active to Inactive or Deleted. Limiting the size on the Server controls the size of the list on the Agent.
- **Schedule** — (required) Specify when the Server compiles the list. Choose one of the following scheduling options:

Every Day — Compile the list every day. Selecting **Every Day** enables **At** to let you set the time the list should be compiled. Enter the time in HH:MM format, such as "2:00 AM" (0 to 24 hours). This is the default setting.

Minutes — Specify the interval, in minutes, at which the Server compiles lists. Use the control to set the number of minutes or enter the number of minutes (1-1440). The default is 60 minutes.

Specific Day — Select a specific day of the month to compile lists. Selecting **Specific Day** enables the list of days and **At** options. Select the day of the month from the list and enter the time in HH:MM format (0 to 24 hours). The scheduled job does not run in months that do not have the selected day. For example, if you select 31st, your job does not run in February, April, June, September or November. To ensure that your job runs every month, select a day between First and 28th, or select Last.

Day of Week — Select the day of the week to compile lists. Selecting **Day of Week** enables the list of days and the **At** options. Select the day and enter the time in HH:MM format, for example Sunday At 12:00 AM (0 to 24 hours).

On Demand — The list is compiled only when you click  (Submit list for compilation) while viewing the list details. Best practice suggests that you schedule the compile process.

Add List Members

After you define your list, add members to your list. You must add at least one member.

Note: If you are creating a generic component list, you can use the following special characters:




]	Right square bracket
^	Caret
`	Grave accent
\	Left backslash
_	Underscore

Caution: If you are creating an MD5 Hash list, enter the values as DG MD5 Hash values, not native MD5 values. The DGMCM provides a utility to convert the values for you.

1. Under Add List Member, enter a **Value** and **Description** (optional) and select the **Status** of the new member, as follows:
 - **Value** — Enter a value of the list type, such as an IP address or MD5 hash value.
 - **Description** — (Optional) Enter a brief description of the member.
 - **Status** — Select the status to assign to the member:
 - Active** — The member should be active on the list.
 - Inactive** — Keeps a value on the list to prevent a user or external system from re-entering a value that you deem undesirable. Use Inactive to identify a value as safe even when the source of the member identifies the value as an IOC. Inactive status reduces false positive triggers. Mark-


ing a member Inactive allows you to add an active member to a list that reaches the maximum size.

Keep Always — Specifies that the value should remain on the list permanently, independent of list updates or changes.

2. (Optional) If you are creating an MD5 Hash list, click  to open a utility to use to convert the MD5 values from your external sources to DG MD5 Hash values. After converting the value, copy the DG MD5, including the braces ({ }) as the value for your MD5 Hash list. If you enter the member value as a DG MD5 value, you see the corresponding native MD5 value after you click **Save**.
3. Click  to add the new member. Click  to reset the fields to default values.

Using Search Filters

After you have created component list members, you can use filter (search) capabilities whenever you view component lists. To apply filters:

1. Click , then fill in the following information to define your search:
 - **Search for Value** — Enter the values you want to search on as comma-separated values, or use wildcards.
 - **Status** — Select the status filters you want to use:
 - All — Display list members in all status categories.
 - Active — Display list members marked as active.
 - Inactive — Display list members marked as inactive.
 - Keep Always — Display list members marked as Keep Active.
 - Deleted — Display list members marked as deleted.
 - **Source** — Select the sources of the component list that you want to filter on.
 - All — All of the following sources.
 - DGMC — Lists created in the DGMC.
 - Feed — Feeds of IOC data or other types of feeds, such as a feed of blacklisted email sites.


API — Component list API (refer to [“Using the Component List API” on page 395](#)).

FireEye — Threat lists provided by FireEye.

ArcSight — Data collected by the ArcSight SIEM.



Palo Alto — Malware and advanced persistent threat information provided by the Palo Alto API.

BlueCoat — Malware and advanced persistent threat information provided by BlueCoat.

- **Use Date Range** — Select this to enable Date Type options. You can filter using the date the component list was created (Created Date) or the date the component list was modified (Modified Date). You can then select a Start Date and an End Date.
2. Click  to apply your filter to the list.
 3. To view component list details, go to the next section, [View Component List Details](#).

View Component List Details

To view the details of a component list, such as the list members, where the list came from and its status, use the Component Lists page in the DGMC.

1. Hover over Policies and, under Lists, select **Component Lists**. The Component List page opens.
2. Click  next to the list to view. The page changes to show the details of the list and its members.
3. (Optional) Apply filters as needed to display the items you want to see, such as all list members whose source is DGMC, and click .

Component List - dlp-chat_apps

General Settings

Name:

dlp-chat_apps

Description:

General list of communication applications for Windows.

Content Type:

Generic

Status:

Active

Version:

3

Max List Size:

1000

Modified:

3/29/2018 11:53:59 AM

Container:

Default (Tier 0)

Last Compiled On:

3/29/2018 11:53:58 AM

Schedule:

Every Day At 12:00 AM

Component List Members

1-8 of 8

VALUE	SOURCE	CREATED	MODIFIED	STATUS
discordapp.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active
facebookmessenger.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active
googletalk.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active
oovoo.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active
pidgin.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active
skype.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active
trillian.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active
yahoomessenger.exe	DGMC	1/30/2017 9:29:31 AM	3/29/2018 11:52:51 AM	Active

The list details include the name of the list, content type, status, maximum list size, schedule and list members.

The member information displays:

- The value of the member
- The source used to create the list, such as DGMC or Feed
- The date the member was created and the date the member was last modified
- The status of the member

Additional sections show rules and policies associated with the list:








- Rules Referencing This List displays the rules that use the component list. The information includes the rule class, type, status, category, platforms, version, date modified, and whether the list is Read Only.
- Policies Referencing This List displays the policies referencing rules that use the component list. The information includes the policy version, date modified, category, platform support, and status.

If you are reviewing the details of an MD5 list, the report shows the DG MD5 Hash (Value column) and native MD5 values for each list member.

382




Edit Component Lists

For making a few changes to lists that are not too large, use the DGMC component lists feature. You cannot change the **Name** or **Content Type** values of an existing list. **Version**, **Modified** and **Last Compiled On** are read-only values.

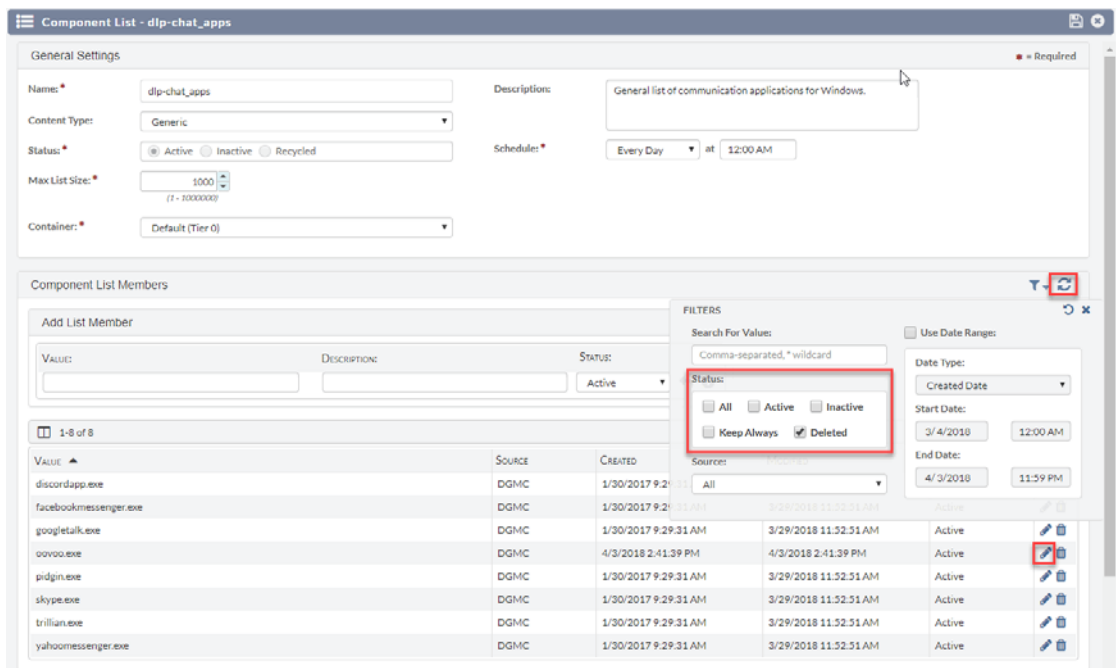
1. Hover over Policies and, under Lists, select **Component Lists**. The Component List page opens.
2. Click  next to the list to edit. The page changes to show the details of the list and its members.
3. Click .
4. Perform one of the following tasks:
 - To change general features of the list, such as the **Max List Size** or **Schedule**, make changes to the options as needed.
 - To change the members on the list, click  next to the list member to change. The member row changes to enable you to edit the member values. Make your changes to the member values and click  to save the new values.
 - To add a list member, enter values for **Value**, **Description**, and **Status** in Add List Member and click  to add the new member.
 - To delete a list member, click  on the row that contains the member to delete.
5. Click  to save your changes.




Restore a Deleted Entry

You cannot delete and then add the same member to a component list. This prevents duplicate entries in lists. You can restore the deleted entry by changing its status from deleted to active.

1. Hover over Policies and, under Lists, select **Component Lists**. The Component List page opens.
2. Click  next to the list to change by restoring a deleted member. The page changes to show the details of the list and its members.
3. Click .
4. Click  to reveal the filter options.

The following figure shows the remaining steps.



5. Under Status, select **Deleted** or **All** to display deleted list members.
6. Click  to refresh the member list.
7. On the row that now displays the deleted member, click  to edit the member entry.
8. In the Status column, select **Active** to change the status of the member from Deleted to Active.
9. Click  to save the new status. You have restored the deleted entry to the list.

Managing Component Lists

DG provides options that allow you to import, export, and delete component lists. You can import component lists that you have exported from Digital Guardian. Exporting lists lets you move them from place to place as files. You might export lists to move them from an evaluation system to a production system.

Users with the role of System Administrator or Policy Manager can export and save copies of the lists they manage to any location on the network. Exported lists are stored as XML files or CSV zip files.


Managing Very Large Component Lists

The file size of very large component lists that can be imported into DG is limited by the IIS `HttpRuntimeSection.maxRequestLength` attribute, which DG has set to 500 MB. If you want to import files larger than 500 MB, you must update the IIS attribute. For more information, refer to the Microsoft IIS documentation.

Import Component Lists

If you have existing lists, either new or updated, you use the Import feature to import the list into DG.

Note: You cannot import the same version or earlier versions of a list into the DGMC. To import a copy of an existing list, you must change either the list contents of the copy or the name of the list.


1. Hover over Policies and, under Lists, select **Component Lists**. The Component Lists page opens, showing your existing Component Lists.
2. To import a list, hover over  in the menu bar and select, select **Import Component Lists**. The Import Component Lists dialog box opens.
3. Click **Choose File** to locate the list to import.

4. Click **Import** to import the list. The list is now available to be deployed to Agents.

Export Component Lists

Exporting your lists can help you manage the list content—review the entries, verify the content, and more.

Note: After you export the list, you cannot re-import the list into the DGMC unless you modify the list contents or name. You cannot import the same version or earlier versions of a list.


1. Hover over Policies and, under Lists, select **Component Lists**. The Component Lists page opens, showing your existing Component Lists.
2. Select the lists to export.
3. To export the lists, hover over  in the menu bar and select one of the following:
 - Export Component Lists — The list is exported as an XML file.
 - Export Component Lists to CSV — The list is exported as a CSV zip file.

The Save dialog box opens.

4. Locate the directory to which to download your lists. Name the list as desired.
5. Click **Save** to save the list in your selected download location. Clicking **Save** saves the list.

Delete Component Lists

You can delete component lists that you no longer need or that are out of date. You cannot delete a list that is being used by any rule.



1. Hover over Policies and, under Lists, select **Component Lists**. The Component Lists page opens, showing your existing Component Lists.
2. Select the lists to delete.
3. To delete the lists, hover over  in the menu bar and select Delete List. The Confirm Deletion dialog box opens.
4. Click **OK** to delete the lists you selected.

Component List Rule Event Reporting

Because the rules you write that use component lists also leverage normal rule properties and symbolic constants, the events and alerts generated appear in the local and enterprise forensic reports.


View Component List Deployment

To review the deployment state of your lists, use the List Deployment report.

1. Hover your cursor over Policies and select **List Deployment**. The List Deployment page opens.
2. Click  to use the Finder to select one computer or group to report about.
3. Click .

The List Deployment report opens displaying all the lists deployed to the computer or group. From the report you get the list name, the version of

the list on the server and Agent, compiled time, and the send and receive times for each deployed list.

To view the details of a list, click  next to the list.

About Rules for Component Lists

Component lists are lists that you reference from rules, similar to arrays in a programming language. By using component lists, you can reduce the time required to write and maintain your rules and protect your data more successfully.

- As long as rules reference the component list, you cannot rename the list. To rename a component list, you must remove references to that list from all rules.
- As with component rules, you cannot write a rule that references a component list until after you create the list.
- Component lists do not contain action, severity, or status information. That information is supplied by the rule that uses the component list.
- Environment variables are supported as items in a string component list. For example, an entry like:

```
##appdata##\folder\malware.exe
```

will be resolved by the rule engine during rule evaluation to a path like:

```
c:\users\user1\appdata\roaming\folder\malware.exe
```

- Wildcard characters "*" and "?" are supported in component list items. For example, during rule evaluation an entry like:

```
c:\users\user1\appdata\roaming\*\malware.exe
```

will match a path like:

```
c:\users\user1\appdata\roaming\folder\malware.exe
```

A component list item can have both environment variables and wildcards at the same time.

- A component list cannot be the only property in a rule. A rule must contain at least one other property in addition to a component list.

- You cannot use the `varArrayLength` property to get the component list size in rules written against a DG 7.3 or later Agent. If you add a DG 7.3 or later Agent, you will need to write new rules using the `componentListSize` property to get the component list size.
- Component lists are not supported in the following rule constructs:
 - `forEachFunction`
 - `getIndexOfArrayItem`
 - `getArrayItem`
 - `setArrayItem`
 - `varArrayLength` (Use the `componentListSize` property to get the component list size, as described in the previous item.)

Calling a Component List

You call a component list from within a rule using the rule element that matches the list content:

- `<componentListString name= "list_name" />` — use this to refer to lists that contain domains or email addresses
- `<componentListMD5 name= "list_name" />` — use this to refer to lists that contain DG MD5 hash values
- `<componentListIPAddress name= "list_name" />` — use this to refer to lists that contain IP addresses
- `<componentListSHA1 name= "list_name" />` — use this to refer to lists that contain SHA1 hash values
- `<componentListSHA256 name= "list_name" />` — use this to refer to lists that contain SHA256 hash values
- `<componentListVersion name= "list_name" />` — use this to refer to the current version of a component list
- `<componentListSize name= "list_name" />` — use this to refer to the current number of items in a component list

list_name is the name you gave to the list when you created it.

When you refer to a component list, be sure to use the correct list name. When you reference a component list from a rule, the list name case must match, as shown in the following example that calls the list `ip_addresses`:

```
<componentListIpAddress name="ip_addresses" />
```

Writing Rules That Use Like for Matching List Members

When you write rules that intend to match member strings on a component list, such as a domain list, you might use the `<in op=like...>` element for the string matching operation. When you use that syntax, add the `like` attribute to the syntax:

```
in op=like like="option">
```

where *option* is replaced by `left`, `right`, or `both`. The `like` attribute specifies where the % matching character gets applied to the member strings to match. For more information about the `like` attribute, refer to the `Like` relational operator in this document.

In this syntax, do not add the % character to the members of the list you are using for matching.

Example Control Rule


This is an example control rule to stop data transfers from IP addresses specified in the component list `ip_addresses`:

```
<and>
<or>
<in>
<parentProcessTopUrlIpAddress />
<list>
<componentListIpAddress name="ip_addresses" />
</list>
</in>
<equal>
<currentProcessImageName />
<stringValue="plugincontainer.exe" />
</equal>
</or>
```

```
<equal >  
<evtOperati onType />  
<constOpNetTransferDownl oad />  
</equal >  
</and>
```

List Compile Job

After you create, import, or modify a component list, you must compile the list.

You can compile the list immediately with the **Compile** option () when you view the details of the list. When you compile a list, the version of the list increases by one.

In addition, DG provides a job that automatically compiles lists that have changed. The List Compile job runs every five minutes by default. When the Agents contact the Server, the Server sends the Agents a list of the lists. After Agents receive the list of lists, they can request updated versions of the lists they have from the Server.

List Job Processing

The list deployment job works by sending the full list to Agents that request the list for the first time after they get a new policy that uses a component list. After the first deployment of a list, future deployments for the list send only changes (deltas) to existing lists, not the full lists. This limits the traffic between Agents and Servers.

List Compilation Logic

To qualify to be compiled by the list job, a list must meet the following conditions:

- The list has to be active and referenced by at least one rule or rule comment. The job does not compile lists that are not referenced in rules.
- The list has to have a notification based on the schedule for the list.

- The existing list has to have new content or the content has to have been updated (perhaps a list member changed from active or keep always to inactive or delete.).
- Any list that is referenced by a rule and has not been compiled is automatically queued for compilation. The job compiles the list the next time it runs.

List Compilation Job Status

After the list job runs, it returns one of the following status conditions:

- Job was successful. This indicates either:
 - There were no lists to process—none of the lists met the conditions to be compiled.
 - All lists that required compilation were processed successfully.
- Job has a warning. This status indicates that one or most lists failed to compile and at least one list compiled successfully
- Job has failed. This status indicates that all the lists that were available for compilation failed to compile.

List Delivery Requirements

To be made available to Agents by the server, a list must meet the following conditions:

- The list has to be active
- The list has to be referenced by a rule
- The list has to have been compiled successfully independent of any list changes

The Agent removes any list that the Server stops listing as available.

About the Component List API

To enable you to create lists from external threat data sources, without creating the lists entry-by-entry in the DGMC, DG provides an API composed of SQL stored procedures. The API enables you to convert externally-sourced data to DG component list format and add the data to component lists. You can then add those lists to your console. The API also includes stored procedures to manage component lists.

To use the API, you need the following:

- Database tables that contain the data for your lists. Generally, each table would contain one of the following types of data that could present a risk to your corporate data or IP, except the generic list which can contain a mix of data types:
 - A list of domains that should trigger your advanced threat protection rules for stopping malware.
 - A list of email address that should trigger your advanced threat protection rules for stopping malware. These represent addresses that might send mail with content that puts your enterprise at risk.
 - A list of items that should trigger your advanced threat protection rules for stopping malware. These represent any combination of entries such as email addresses, domains or other items.
 - A list of IP address that should trigger your advanced threat protection rules for stopping malware.
 - A list of DG MD5 hash values that should trigger your advanced threat protection rules for stopping malware. These represent hash values of processes or applications that may perform operations. The MD5 hash values must be the Digital Guardian hash values, not values generated by MD5 hash value tools.
- Software that you use to access the database tables using SQL stored procedures.
- A connection to the DG Server.

The information in the database tables likely does not come directly from third-party feeds.

Using the Component List API

To enable you to create lists from your threat intelligence data or Indicators of Compromise (IOCs), DG includes an API composed of SQL stored procedures (stored procs or SP) to create and manage lists. The API allows you to convert externally-sourced threat intelligence data to DG component list format. The available SPs are:

Stored Procedure Name (Link to Reference Page)	Description of Operation
“SP_API_ADD_VALUE_TO_COMP_LIST” on page 400	Add content value, such as a domain or email address, to specified list.
“SP_API_CREATE_COMP_LIST” on page 402	Create new component list
“SP_API_DELETE_COMP_LIST” on page 407	(Not Implemented) Delete specified component list and associated content.
“SP_API_DELETE_VALUE_FROM_COMP_LIST” on page 408	Delete specified value from component list.
“SP_API_DEPLOY_COMP_LIST” on page 410	Deploy (advertises) a component list.
“SP_API_GET_COMP_LIST_CONTENT” on page 411	Return a table of all values for specified list.
“SP_API_UPDATE_COMP_LIST” on page 413	Update new component list.
“SP_API_UPDATE_VALUE_IN_COMP_LIST” on page 417	Update metadata associated with specified component list value.

To use the API, you need the following:

- Sources of one the following types of data:
 - Domains that should trigger your advanced threat protection rule.
 - Email address that should trigger your advanced threat protection rule.
 - Items that should trigger your advanced threat protection rule. These represent any combination of entries such as email addresses or domains or other items.
 - IP addresses that should trigger your advanced threat protection rule.

- DG MD5 hash values that should trigger your advanced threat protection rule. These represent hash values of processes or applications that may perform operations that could compromise your network. These must be DG MD5 hash values.
- Software that you use to run stored procedures, such as C# or other ODBC- or JDBC-enabled software. Microsoft SQL Server is one example.
- A connection to the DG Server.

To use the stored procedures, provide the arguments to the procedures that they require. You can call the SPs from another language, such as C#, providing the input arguments in the call to the SP. Because the SPs run SQL queries, your threat intelligence data source files must contain the required input arguments in the order shown on the stored procedure reference pages.

Creating and Deploying Component Lists With the API

The most important part of using the stored procedures to create component lists is the order in which you use them.

Component List API Workflow

Perform these task by running the SP for each step.

1. Create the file that contains the threat data intelligence to add to your component list. You do this outside of DG to create the data tables for the component list API to access.
2. Create the List — [“SP_API_CREATE_COMP_LIST” on page 402](#).
3. Add members to the list from the threat intelligence data tables — [“SP_API_ADD_VALUE_TO_COMP_LIST” on page 400](#).
4. Add the list to the console— [“SP_API_UPDATE_COMP_LIST” on page 413](#)
5. Deploy the list — [“SP_API_DEPLOY_COMP_LIST” on page 410](#).
6. Add the list to the console — [“SP_API_UPDATE_COMP_LIST” on page 413](#).

Managing Component Lists With the API

After you complete these tasks, you can manage your lists in the console or with the other component list SPs:

- [“SP_API_ADD_VALUE_TO_COMP_LIST” on page 400](#) — Add content value to specified list.
- [“SP_API_DELETE_COMP_LIST” on page 407](#) — (Not Implemented) Delete specified component list and associated content.
- [“SP_API_DELETE_VALUE_FROM_COMP_LIST” on page 408](#) — Delete specified value from component list.

- [“SP_API_GET_COMP_LIST_CONTENT” on page 411](#) — Return a table containing all values for specified list.
- [“SP_API_UPDATE_COMP_LIST” on page 413](#) — Update new component list.
- [“SP_API_UPDATE_VALUE_IN_COMP_LIST” on page 417](#) — Update metadata associated with specified component list value.

Stored Procedure Reference Pages

The following pages provide details about the stored procedures available to manage component lists in your enterprise. The procedures are listed in alphabetical order. Each page provides the following information about the referenced procedure:

- Name — The name of the stored procedure
- Function — What the stored procedure does
- Syntax — How to use the input arguments in the procedure
- Arguments — The arguments you provide to the procedure to run it
- Examples — Some examples of SP execution with arguments
- Messages — The messages the procedure returns when it runs
- Return Status — The result of running the procedure
- Effects — Any effects that running the procedure might cause, such as writing information to the console log

Name	SP_API_ADD_VALUE_TO_COMP_LIST
Function	Add content value to specified list
Syntax	<code>sp_api_add_value_to_comp_list 'list_name', source_type, content_value, content_status</code>
Arguments	<p>LIST_NAME — Specify the name of the list to which to add a member (value).</p> <p>SOURCE_TYPE — Specifies the source of the list value. Valid values are 0 = None, 1 = DGMC, 2 = API, 3 = FireEye, 4 = ArcSight, 5 = Palo Alto. The source type appears in the listing of members of the list in the console. You can sort the members by source type. You cannot filter the list by source type.</p> <p>CONTENT_DESCRIPTION — (Optional) Provides a brief description of the added member.</p> <p>CONTENT_VALUE — Provide the member (value) to add to the component list specified by LIST_NAME. The console displays this as the list value. If you are adding an MD5 hash value to an MD5 Hash list, use the native MD5 hash values.</p> <p>CONTENT_STATUS — (Optional) Defines the status assigned to the new list value. Defaults to 1 (active). The valid values are 0 = inactive, 1 = active, 2 = deleted, 3 = keep always.</p> <p>IS_NATIVE_MD5 — Set to 1 (default) for native MD5 hash values. Set to 0 for DG MD5 hash value.</p> <p>STATUS_MESSAGE — output</p>
Examples	<code>sp_api_add_value_to_comp_list 'blocked_email_addresses', 2 'Email address for a spam site annoyingspam.com'</code> — Adds the value annoyingspam.com to the list blocked email address as an inactive member. The source type is 2 (API) so you see API in the member listing in the console.

`sp_api_add_value_to_comp_list 'blocked_email_addresses', 0, 'annoyingspam.com'` — Adds the value annoyingspam.com to the list blocked_email_addresses as an inactive member. The source type is 0 (none) so you see none in the member listing in the console.

`sp_api_add_value_to_comp_list 'blocked_email_addresses', 3, 'Email address for a spam site annoyingspam.com', 3` — Adds the value annoyingspam.com to the list blocked_email_addresses. The member is immortal (keep always) by specifying 3 for CONTENT_STATUS. The SOURCE_TYPE input argument = 3 specifies that a FireEye sensor is the source of the member.

Messages

SUCCESS — The value CONTENT_VALUE was added to list LIST_NAME.

FAILURE — A component list with name LIST_NAME does not exist.

FAILURE — A component list with value CONTENT_VALUE already exists.

FAILURE — Maximum list size MAX_LIST_SIZE has been reached.

Status

0 = Success

1 = Failure — list does not exist

2 = Failure — value already exists

3 = Failure — max list size reached

Other = failure — SQL error

Effects

No effects beyond the messages provided.



Name	SP_API_CREATE_COMP_LIST
Function	Create new component list
Syntax	<code>sp_api_create_comp_list 'list_name', list_type</code>
Arguments	<p>LIST_NAME — Enter a name for your list. The name must be unique on this DGMCA instance. Do not use special characters, such as & or #, in the name.</p> <p>LIST_DESCRIPTION — (Optional) Provide a brief description of the list and its content.</p> <p>LIST_TYPE — Specify the content type of the list by providing the type value. LIST_TYPE is read only unless you create a generic component list (LIST_TYPE = 4). You cannot change the type of list after you create it if it is not a generic list.</p>

To Create this Type of List	Use This Type Value
Domain	1
Email	3
Generic	4
IP Address	2
MD5	0

LIST_STATUS — (Optional) Specify the list status as 1 = Active Status (the default) or 0 = Inactive Status.

MAX_LIST_SIZE — (Optional) Specify how large the list can grow on the Agent. When the list reaches the maximum size, you cannot add more members until you change one or more existing members' status from Active to Inactive or Deleted. Limiting the size here on the server controls the size of the list on the Agent effectively.

SCHEDULE_TYPE — (Optional) Provide the type of schedule for when the server compiles and deploys the list):

For Schedule Type Of...	Specify The Value...
Day of Week	0. Specify the day in Schedule_Interval. If you leave Schedule_Interval empty, the job runs on Sunday.
Every Day	3. Specify the time of day in Schedule_Time. If you leave this empty, the job runs daily at 1:00am.
Minutes	2. Specify the number of minutes between running the list component job in Schedule_Time. If you leave this empty, the job runs every five minutes.
On Demand	255. (The default schedule option). You compile the list by clicking  (Submit list for compilation) on the page that shows the list details. Alternatively, you can compile the list by selecting the list on the Component Lists page, hovering over  and selecting Compile List.
Specific Day	1. Specify the day of the month on which to run the job. If you leave this empty, the job runs on the first day of every month.

SCHEDULE_INTERVAL — (Optional) Provide the schedule interval for your list (when to compile and deploy the list):

For Schedule Type Of...	Specify The Value...
Day of Week	1-7 to specify the day on which to run the job. 1 = Sunday and 7 = Saturday. If you leave Schedule Interval empty, the job runs on Sunday.
Every Day	0-1440 minutes, where 0 is 12:00am and 1439 is 11:59pm. Specify the time of day to run the job. If you leave this empty, the job runs daily at 1:00am.
Minutes	1-1440 minutes (24 hours). Specify the number of minutes between running the list component job. If you leave this empty, the job runs every 5 minutes.
On Demand	255 (the default schedule option)
Specific Day	1-32 to specify the day of the month on which to run the job. 32 is the last day of the month. If you leave this empty, the job runs on the first day of every month.

If you do not provide a value for SCHEDULE_INTERVAL, the job runs at the default interval depending on the type of interval.

SCHEDULE_TIME — (Optional) Provide the schedule timing for your list (when to compile and deploy the list):

For Schedule Type Of...	Specify The Value...
Day of Week	0-1440 minutes, where 0 is 12:00am and 1439 is 11:59pm. Specify the day in Schedule_Time. If you leave Schedule Time empty, the job runs at 1:00am on the scheduled day.
Every Day	0-1440 minutes, where 0 is 12:00am and 1439 is 11:59pm. If you leave this empty, the job runs daily at 1:00am.

For Schedule Type Of...	Specify The Value...
Minutes	1-1440 where 1 is one minute and 1439 is 24 hours. Specifies the number of minutes between executing the list compile job. If you leave this empty, the job runs every five minutes.
On Demand	255 (the default schedule option)
Specific Day	0-1439 where 0 is 12:00am and 1439 is 11:59pm. Specify the day of the month on which to run the job. If you leave this empty, the job runs on the first day of every month.

If you do not provide a value for `SCHEDULE_TIME`, the job runs at the default time, depending on the type of schedule.

`CONTAINER_ID`— (Optional) Provide the policy container for this component list. The default container value is 0 (Tier 0).

`STATUS_MESSAGE` — Output

Examples

`sp_api_create_comp_list 'blocked_md5_hashes', 0, 1, 500` creates an active list named `blocked_md5_hashes` that can have up to 500 members.

`sp_api_create_comp_list 'blocked_md5_hashes', 0, 1, 750, 0` creates an active list named `blocked_md5_hashes` that can have up to 750 members and builds and compiles on Sunday.

`SP_API_CREATE_COMP_LIST 'blocked_md5_hashes', 0, 1, 15` creates an active list named `blocked_md5_hashes` that can have up to 1000 members and builds and compiles on the 15th of every month.

`SP_API_CREATE_COMP_LIST 'blocked_md5_hashes', 0, 3, 600` creates an active list named `blocked_md5_hashes` that can have up to 1000 members and builds and compiles every day at about 10:00am.

Messages

SUCCESS — A component list LIST_NAME was created.

FAILURE — A component list with name LIST_NAME already exists.

FAILURE — Cannot create a list with size greater than MAX_LIST_SIZE.

Status

0 — success

1 — Failure the list already exists.

2 — Failure the list size specified is larger than 1000.

Other — Failure list creation encountered a SQL error.

Effects

No effects beyond the messages and status.

Name	SP_API_DELETE_COMP_LIST
Function	(For future use) Delete specified component list and content
	Caution: You cannot remove a list that is referenced by any rule. To remove a list, remove all rule references to the list.
Syntax	Not implemented.
Arguments	Not implemented.
Examples	Not implemented.
Messages	Not implemented.
Status	Not implemented.
Effects	Not implemented.

Name	SP_API_DELETE_VALUE_FROM_COMP_LIST
Function	Delete specified value (list member) from component list
Syntax	<code>sp_api_delete_value_from_comp_list 'list_name', 'content_value', 'status_message'</code>
Arguments	<p>LIST_NAME — Specify the name of the list from which to remove a member (value).</p> <p>CONTENT_VALUE — Specify the list member (value) to remove from list.</p> <p>STATUS_MESSAGE— (Optional) Specify an output message.</p>
Examples	<p><code>sp_api_delete_value_from_comp_list 'blocked_email_list', 'j fusco@digitalguardian.com'</code> removes the email address <code>jfusco@digitalguardian.com</code> from the list named <code>blocked_email_list</code>.</p> <p><code>sp_api_delete_value_from_comp_list 'blocked_email_list', 'j fusco@digitalguardian.com'</code> Removed value from <code>list</code> removes the email address <code>jfusco@digitalguardian.com</code> from the list named <code>blocked_email_list</code> and writes the message <code>Removed value from list</code> if successful.</p>
Messages	<p>SUCCESS — The value <code>CONTENT_VALUE</code> was remove from list <code>LIST_NAME</code>.</p> <p>FAILURE — A component list with name <code>LIST_NAME</code> does not exist.</p> <p>FAILURE — <code>CONTENT_VALUE</code> does not exist on the specified list.</p>
Status	<p>0 = Success</p> <p>1 = Failure — list does not exist</p>

2 = Failure — value does not exist

Other = failure — SQL error

Effects

No effects expected beyond the messages and status.

Name	SP_API_DEPLOY_COMP_LIST
Function	Mark list for deployment so list compile job compiles and advertises list
Syntax	<code>sp_api_deploy_comp_list 'list_name', 'status_message'</code>
Arguments	<p>LIST_NAME — Specify the name of the list to mark for deployment.</p> <p>STATUS_MESSAGE — (Optional) Output.</p>
Examples	<p><code>sp_api_deploy_comp_list 'blocked_md5_list'</code> marks the list Blocked_MD5_List so the list compile job processes the list and makes it available to Agents.</p>
Messages	No message returned.
Status	No status information returned.
Effects	No effects expected beyond the messages and status.

Name	SP_API_GET_COMP_LIST_CONTENT
Function	Returns table of all values for specified list
Syntax	<code>sp_api_get_comp_list_contents <i>list_name</i></code>
Arguments	<p>LIST_NAME — Specifies the name of the component list from which to return the members (values).</p> <p>SORT_BY — (Optional) Use this argument to order the returned fields in the table. Append ASC to the argument to sort in ascending order (A-Z, 0-9, earlier to later date and time) or append DESC to sort in descending order (Z-A, 9-0, later to earlier date and time). ASC (ascending) is the default sort order.</p> <p>For example, you could use CONTENT_VALUE ASC to sort by the entry values in ascending order, or CREATED_DTTM DESC (sorts to show the oldest entry first) or MODIFIED_DTTM DESC (sorts to show the most recently modified entry).</p> <p>Executing this procedure returns a table that contains the following fields for the values from the list:</p> <ul style="list-style-type: none">• CONTENT_VALUE — The value for the list entry• CONTENT_DESCRIPTION — When provided, a brief description of the content.• SOURCE_TYPE — Reports the type of source that provided the content in the list. This may be empty (null).• CONTENT_STATUS — may be null• MODIFIED_DTTM — Time and date that someone last modified the list entry• CREATED_DTTM — Time and date that someone created the entry.• LIST_NAME — Name of the component list from which the member information was taken.

Examples

`sp_api_get_comp_list_contents 'blocked_email_domains'` returns the content of the list `Blocked_Email_Domains`.

`sp_api_get_comp_list_contents 'blocked_md5_list' content_value asc` returns the contents of the list `Blocked_MD5_List` sorted in ascending order by content values.

Messages

SUCCESS — Values from list `LIST_NAME` were returned.

FAILURE — A component list with name `LIST_NAME` does not exist.

OTHER + Failure — SQL error

Status

0 = Success

1 = Failure — List does not exist

Other = Failure — SQL error

Other = Failure — SQL error

Effects

There are no effects expected from running this procedure beyond returning the table of list contents.

Name **SP_API_UPDATE_COMP_LIST**

Function Update new component list

Syntax `sp_api_update_comp_list 'list_name', 'list_description',
list_status, max_list_size, schedule_type, schedule_interval,
schedule_time, container_ID`

Arguments **LIST_NAME** — Specify the name of the list to which to add a member (value).



LIST_DESCRIPTION — (Optional) Provide a brief description for the list update, such as why you are updating the list or the values added.

LIST_STATUS — (Optional) Specify the list status as 1 = Active Status (the default) or 0 = Inactive Status.

MAX_LIST_SIZE — (Optional) Specify how large the list can grow on the Agent. When the list reaches the maximum size, you cannot add more members until you change one or more existing members' status from Active to Inactive or Deleted. Limiting the size here on the server controls the size of the list on the Agent effectively.

SCHEDULE_TYPE — (Optional) Provide the type of schedule for when the server compiles and deploys the list):

For Schedule Type Of...	Specify The Value...
Day of Week	0. Specify the day in Schedule_Interval. If you leave Schedule_Interval empty, the job runs on Sunday.
Every Day	3. Specify the time of day in Schedule_Time. If you leave this empty, the job runs daily at 1:00am.
Minutes	2. Specify the number of minutes between running the list component job in Schedule_Time. If you leave this empty, the job runs every five minutes.

For Schedule Type Of...	Specify The Value...
On Demand	255. (The default schedule option). You compile the list by clicking  (Submit list for compilation) on the page that shows the list details. Alternatively, you can compile the list by selecting the list on the Component Lists page, hovering over  and selecting Compile List.
Specific Day	1. Specify the day of the month on which to run the job. If you leave this empty, the job runs on the first day of every month.

SCHEDULE_INTERVAL — (Optional) Provide the schedule interval for your list (when to compile and deploy the list):

For Schedule Type Of...	Specify The Value...
Day of Week	1-7 to specify the day on which to run the job. 1 = Sunday and 7 = Saturday. If you leave Schedule Interval empty, the job runs on Sunday.
Every Day	0-1440 minutes, where 0 is 12:00am and 1439 is 11:59pm. Specify the time of day to run the job. If you leave this empty, the job runs daily at 1:00am.
Minutes	1-1440 minutes (24 hours). Specify the number of minutes between running the list component job. If you leave this empty, the job runs every 5 minutes.
On Demand	255 (the default schedule option)
Specific Day	1-32 to specify the day of the month on which to run the job. 32 is the last day of the month. If you leave this empty, the job runs on the first day of every month.

If you do not provide a value for SCHEDULE_INTERVAL, the job runs at the default interval depending on the type of interval.

SCHEDULE_TIME — (Optional) Provide the schedule timing for your list (when to compile and deploy the list):

For Schedule Type Of...	Specify The Value...
Day of Week	0-1440 minutes, where 0 is 12:00am and 1439 is 11:59pm. Specify the day in Schedule_Time. If you leave Schedule Time empty, the job runs at 1:00am on the scheduled day.
Every Day	0-1440 minutes, where 0 is 12:00am and 1439 is 11:59pm. If you leave this empty, the job runs daily at 1:00am.
Minutes	1-1440 where 1 is one minute and 1439 is 24 hours. Specifies the number of minutes between executing the list compile job. If you leave this empty, the job runs every five minutes.
On Demand	255 (the default schedule option)
Specific Day	0-1439 where 0 is 12:00am and 1439 is 11:59pm. Specify the day of the month on which to run the job. If you leave this empty, the job runs on the first day of every month.

If you do not provide a value for SCHEDULE_TIME, the job runs at the default time, depending on the type of schedule.

CONTAINER_ID— (Optional) Provide the policy container for this component list. The default container value is 0 (Tier 0).

STATUS_MESSAGE — Output

Examples

```
sp_api_update_comp_list 'list_name'
```

```
sp_api_update_comp_list 'list_name', list_status,  
max_list_size, container_ID
```

```
sp_api_update_comp_list 'list_name', 'list_description',  
schedule_type, schedule_interval, schedule_time, container_ID
```

Messages

SUCCESS — A component list LIST_NAME was created.

FAILURE — A component list with name LIST_NAME does not exist.

FAILURE — Cannot update list because the size is greater than the limit set by MAX_LIST_SIZE.

FAILURE — There are more entries in the list than the new maximum size set by MAX_LIST_SIZE.

Status

0 = Success

1 = Failure — List does not exist

2 = Failure — List size too big

3 = Failure — List bigger than new list size

Other = Failure — SQL error

Effects

There are no effects expected from running this procedure.

Name	SP_API_UPDATE_VALUE_IN_COMP_LIST
Function	Updates metadata associated with specified component list value
Syntax	<code>SP_API_UPDATE_VALUE_IN_COMP_LIST LIST_NAME CONTENT_VALUE</code>
Arguments	<p>LIST_NAME — Specify the name of the list on which to modify the specified member (value).</p> <p>CONTENT_VALUE — Provide the member (value) to update on the component list specified by LIST_NAME. The console displays this as the list value. If you are updating an MD5 hash value, provide DG MD5 hash values, not native MD5 hash values.</p> <p>SOURCE_TYPE — (Optional) Specifies the source of the list value. Valid values are 0 = None, 1 = DGMC, 2 = API, 3 = FireEye, 4 = ArcSight, 5 = Palo Alto. The source type appears in the listing of members of the list in the console. You can sort the members by source type. You cannot filter the list by source type.</p> <p>CONTENT_DESCRIPTION — (Optional) Provides a brief description of the added member.</p> <p>CONTENT_STATUS — (Optional) 0 = inactive, 1 = active, 2 = purged, 3 = immortal</p> <p>STATUS_MESSAGE — (Optional) output</p>
Examples	<p><code>SP_API_UPDATE_VALUE_IN_COMP_LIST blocked_md5_list CONTENT_VALUE —</code></p> <p><code>SP_API_UPDATE_VALUE_IN_COMP_LIST blocked_md5_list CONTENT_VALUE 3 —</code></p> <p><code>SP_API_UPDATE_VALUE_IN_COMP_LIST blocked_md5_list CONTENT_VALUE 5 MD5_hash value for Microsoft Word —</code></p>

Messages

SUCCESS — The value CONTENT_VALUE was updated in list LIST_NAME. This message is augmented with which field were updated.

FAILURE — A component list with name LIST_NAME does not exist.

FAILURE — A component list value with value CONTENT_VALUE does not exist.

FAILURE — Nothing to update for component list value CONTENT_VALUE.

Status

0 = Success

1 = Failure — list does not exist

2 = Failure — value does not exist

3 = Failure — nothing to update. All arguments that can be null are already null.

Other = Failure — SQL error

Effects

There are no effects expected from running this procedure.

CHAPTER 13 Data Vault Rules

A data vault, also known as a sandbox, is a set of additional rules that you can apply to an application process. Usually you create a data vault to apply an additional layer of security while a user is working with particularly sensitive files. For example, you might normally allow users to send file attachments without restriction. While a user is working with files in a confidential folder, you might want to block that ability to send file attachments.

Data vault rules give you the flexibility to implement tighter security when it is warranted and the ability to relax that security during less sensitive activity.

A data vault consists of at least two rules:

- Trigger rule
- Data vault rule

Trigger Rules

A trigger rule is a type of Control rule that contains the conditions that a user must meet to flag the current process as vaulted. Generally, a trigger rule contains

a location, one or more file extensions or file types, and an application. For example, your trigger rule could create a data vault when a user starts Microsoft Excel to open a spreadsheet file in the c:\HRfiles\directory. After you identify the conditions that meet the trigger in the XML rule, you set the rule action to Vault t.

Rule Settings

Name: *

Vault Trigger Rule

Status: *

Active

Inactive

Category: *

Unassigned

X

Action: *

Vault

Send Alert: *

Event Only

Continue Rule Evaluation: *

No

Yes

Run Rule After Operation: *

No

Yes

Container: *

Default (Tier 0)

Platforms:

None

Description:

Expedited Rule: *

No

Yes

When the conditions in the trigger rule are met, the application that met the conditions is flagged as data vaulted, and additional data vault rules take effect. These rules remain in effect until the user closes the application that triggered the data vault.

Note: After DG flags a process as data vaulted, every file the user opens in that process is affected by the vault status. The process remains vaulted until the user closes the process. For example, if DG vaults Microsoft Word when a user opens a .docx file, every subsequent file the user opens in Word will be vaulted until the user closes that Microsoft Word instance.

Data Vault Rules

Data Vault rules are the additional control rules that come into effect once the requirements in the trigger rule have been met. Data Vault rules contain one or more of the following data vault rule properties:

- activeVaultRuleNames
- anyProcessVaulted
- clipboardVaultRuleName

- `curProcessSaveAsActive`
- `curProcessVaulted`
- `curProcessVaultRuleName`
- `parentProcessVaulted`
- `parentProcessVaultRuleName`
- `prevProcessVaulted`
- `prevProcessVaultRuleName`
- `similarProcessVaulted`

For more information about these properties, refer to [“Data Vault Rule Properties” on page 126](#).

Data Vault rules can be any type of Digital Guardian rule. Generally, data vault rules are more restrictive, and add an extra layer of security to user activity for as long as the data vault is in effect. These rules might restrict the user’s ability to copy and paste content, or to rename and move files. After the user closes the data vaulted application instance, the extra security stops and the user can resume normal activity.

You can fine-tune your data vault rules by requiring that a specific application process be data vaulted, or requiring that a specific trigger rule be in effect. Alternatively, you can create data vault rules that do not include process or rule names. In this case, Digital Guardian applies the rule to all data vaulted processes.

Caution: Do not write data vault rules that apply vaulting to Microsoft Windows Explorer (`explorer.exe`). If you vault Windows Explorer, the vault never closes because the process `explorer.exe` never closes. Windows Explorer remains vaulted until you reboot the computer.

Like all Digital Guardian rules, Data Vault rules cannot take effect until you have included them in a policy and applied that policy to a user, computer, or group. For more information on policies, refer to *Digital Guardian Management Console User’s Guide*.

Using File Save As

Applying a data vault to an application is the only way to prevent that application from using the Save As feature to save copies of files to new locations. When a process is data vaulted, users cannot save a file to any directory other than the one where the file originated. This restriction exists independent of any additional data vault rules that you may have created.

Data vault protection against Save As activity applies only to the file being edited. It does not affect internal application activity such as log files.

Multiple Data Vaults

Digital Guardian supports multiple data vaults. You might want multiple data vaults for situations where the restrictions on one group of files, printers, or network connections differs from the restrictions on another group. Even if both data vaults are in effect, Digital Guardian can recognize the differences, and respond appropriately.

You can use any of the the `VaultRuleName` properties to identify the trigger rule for a data vault. By including these properties with a data vault rule, you can ensure that the rule takes effect only when the appropriate trigger rule has marked a process as data vaulted.

If you do not include the `curProcessVaultRuleName` property in a rule definition, the data vault rule applies to all data vaulted applications.

Note: When referring to another rule, always refer to that rule in lowercase text.

Creating Data Vault Rules

To create a data vault, you must create a trigger rule and a data vault rule.

Creating a Trigger Rule

You create trigger rules from the Digital Guardian Management Console (DGMC). Trigger rules use the same syntax as all other Digital Guardian rules. The process described here applies only to the trigger rules themselves - the rules that create the data vault. Trigger rules have no effect beyond setting the conditions necessary for Data Vault rules to become active.

1. Hover over Policies in the DGMC and select Rules in the Control category. The Control Rules page appears. This page displays a list of existing control rules.

2. Click  and select **Create New Rule**. The Create New Rule page opens.

3. Enter the following information:

Name - The name of the rule. You cannot change this value after you have saved the rule.


Status - Select Active to enable the Rule. Select Inactive to disable the rule.

Category - The category to which the rule will belong. You must assign all rules to a category. You can assign a rule to an existing category or create a new category. To create a new category, select New Category and enter a new category name.

Action - Set to Vault.


Description - A brief description of what this rule does. The text you enter here will appear in the Description section of the Rule Listing on the Rules page.

Definition - The Rule text. A rule is triggered only when all of its conditions are met. For information about writing rule definitions, refer to “Rule Definitions.”

4. Click . The trigger rule is created.

Creating a Data Vault Rule

You create data vault rules from the Digital Guardian Management Console (DGMC). Data Vault rules use the same syntax as all other Digital Guardian rules. The process described here applies only to the data vault rules themselves—the rules that take effect when the conditions in the trigger rule have been met.

1. Hover over Policies in the DGMC and select Rules in the Control category. The Control Rules page appears. This page displays a list of existing control rules.
2. Click  and select **Create New Rule**. The Create New Rule page opens.
3. Enter the following information:

Name - The name of the rule. You cannot change this value after you have saved the rule.

Status - Select **Active** to enable the Rule. Select **Inactive** to disable the rule.

Category - The category to which the rule will belong. You must assign all rules to a category. You can assign a rule to an existing category or create a new category. To create a new category, select **New Category** and enter a new category name.

Action - The type of action that the rule performs. You can specify any type of action for the data vault rules. For example, you might have additional rules to block file uploads as long as the data vault is in effect.


Send Alert- Allows you to prevent alerts and alert email notifications from being generated when this rule is violated. Select:

- **Immediately** to allow alert generation and notification.
- **Never** to prevent alert generation and notification. If you select this option, you will not receive alerts when a user violates a control rule. This option is only available for the Block and Prompt rule actions.

Severity - The seriousness of a violation of the rule. Note that the Severity Level you assign a Rule translates into the Severity Level of its Alert, which is sent by the Agent to the DG Server.

Description - A brief description of what this rule does. The text you enter here will appear in the Description section of the Rule Listing on the Rules page.

Definition - The Rule text. A rule is only triggered when all of its conditions are met. For information about writing rule definitions, refer to Rule Definitions. Data vault rules must contain one of the rule properties listed in [“Data Vault Rules” on page 420](#).

4. Click . The data vault rule is created.

Examples of Data Vault Rules

The examples presented in this section show how to create a data vault, as well as a rule that acts once the data vault is created.

Trigger Rule To Flag Notepad Process as Data Vaulted

This rule lists a number of document file extensions. When a user opens a file with a listed extension that is stored in the c: \docrepository directory using Notepad, the Notepad process is flagged as data vaulted.

```
<and>
  <i n>
    <evtSrcFileExt/>
    <list>
      <string value="txt" />
      <string value="doc" />
      <string value="rtf" />
      <string value="dot" />
      <string value="htm" />
    </list>
  </i n>
  <regExp expr="^c: \\docrepository">
    <evtSrcFilePath/>
  </regExp>
  <equal >
    <evtOperationType/>
    <constOpFileOpen/>
  </equal >
  <equal >
    <curProcessImageName/>
    <string value="notepad. exe" />
  </equal >
</and>
```

Trigger Rule To Create Data Vault in Response To On-Screen Content

This rule examines on-screen text in the current application using Digital Guardian Adaptive Content Inspection. If the text matches any of the criteria specified in the PCI policy, the rule classifies the application as PCI and data vaults the application.

This rule uses the constant `constOpAppScreenBuffer` that applies only to 3270 applications and Microsoft Internet Explorer.

Note: This rule requires functionality provided by the optional add-on feature Digital Guardian Adaptive Content Inspection.

```
<and>
  <equal >
    <evtBufferPolicyTag/>
    <string value="pci" />
  </equal >
  <equal >
    <evtOperationType/>
    <constOpAppScreenBuffer/>
  </equal >
</and>
```

Prevent User From Saving Files Outside of c:\DocRepository Using a Data Vaulted Notepad Process

This rule takes effect only when the Notepad process is data vaulted. If the process notepad.exe is not data vaulted, none of the conditions in this rule take effect.

```
<and>
  <not>
    <regExp expr="^c: \\docrepository">
      <evtSrcFilePath/>
    </regExp>
  </not>
  <equal >
    <curProcessVaulted/>
    <bool value="true"/>
  </equal >
  <equal >
    <curProcessImageName/>
    <string value="notepad.exe"/>
  </equal >
  <equal >
    <evtOperationType/>
    <constOpFileWrite/>
  </equal >
</and>
```

Prevent User From Performing Network Uploads When a Specific Data Vault Is Active

This rule takes effect only when any process has been data vaulted by a trigger rule named PreventUploads. You could use this rule to prevent users from uploading files while the data vault is in effect. This rule prevents all uploads, regardless of application or the location of the file.

```
<and>
  <equal >
    <anyProcessVaulted/>
    <bool value="true"/>
  </equal >
  <equal >
    <currentProcessVaultRuleName/>
    <string value="PreventUploads"/>
  </equal >
  <equal >
    <eventOperationType/>
    <constantNetTransferUpload/>
  </equal >
</and>
```

Block Application Data Exchange From Application Screen Content

This rule takes effect when a user has violated the trigger rule Accessing PCI Screen. It prevents users from performing any type of application data exchange event.

```
<and>
  <equal >
    <evtOperationType/>
    <constOpAdePaste/>
  </equal >
  <equal >
    <clipboardVaultRuleName/>
    <string value="Accessing PCI Screen" />
  </equal >
</and>
```

Prevent Uploads and CD Burning When Any Data Vault Is Active

This rule takes effect when any process is data vaulted. It prevents users from uploading files or burning CDs. You might create a rule like this one to increase security regardless of the specific rule that triggers the data vault.

```
<and>
  <equal >
    <anyProcessVaulted/>
    <bool value="true" />
  </equal >
  <equal >
    <evtOperationType/>
    <list>
      <constOpNetTransferUpload/>
      <constOpCDBurn/>
    </list>
  </equal >
</and>
```

Prevent Screen Prints When a Specific Rule Is Active

This rule takes effect when any of the three listed trigger rules have activated. It prevents users from copying or printing screen content.

```
<and>
  <i n>
    <evtOperati onType/>
    <l i s t>
      <constOpAdePri ntScreen/>
      <constOpAdePri ntProcess/>
    </l i s t>
  </i n>
  <i n>
    <acti veVaul tRul eNames/>
    <l i s t>
      <stri ng val ue="VAULT - personal  data"/>
      <stri ng val ue="VAULT - confi denti al  data"/>
      <stri ng val ue="VAULT - fi nanci al  data"/>
    </l i s t>
  </i n>
</and>
```


Block File Writes Outside Specific Directory

This rule blocks any file write or rename events outside the c: \datavault\confidential\ directory when a data vault triggered by the rule r1-trigger is present. It uses the source or destination file properties to prevent the rule from blocking file write events performed by normal internal application processes. For more information, refer to [“curProcessSave AsActive” on page 127](#)

```

<and>
  <not>
    <regExp expr="c: \datavault\confidential\\. *">
      <evtSrcFilePath/>
    </regExp>
  </not>
  <and>
    <not>
      <regExp expr="c: \datavault\confidential\\. *">
        <evtDestFilePath/>
      </regExp>
    </not>
    <equal >
      <evtOperationType/>
      <constOpFileRename/>
    </equal >
  </and>
  <equal >
    <evtOperationType/>
    <constOpFileWrite/>
  </equal >
  <equal >
    <curProcessSaveAsActive/>
    <bool value="true"/>
  </equal >
  <equal >
    <curProcessVaultRuleName/>
    <string value="r1-trigger"/>
  </equal >
</and>

```

APPENDIX A Common Use Cases

This appendix provides examples of common use cases for rules requested by current and potential customers. These use cases demonstrate the syntax of the rule language and highlight its capabilities.

You can find examples in the following categories:

- [“Examples of Classification Rules” on page 436](#)
- [“Examples of File System Rules” on page 446](#)
- [“Examples of Network Rules” on page 470](#)
- [“Examples of Print Rules” on page 480](#)
- [“Examples of CD/DVD Rules” on page 486](#)
- [“Examples of Application Data Exchange Rules” on page 491](#)
- [“Examples of Email Rules” on page 498](#)
- [“Examples of Filter Rules” on page 503](#)
- [“Examples of Trusted Process Rules” on page 508](#)
- [“Examples of File Capture Rules” on page 510](#)
- [“Examples of Virtualization and RDP Rules” on page 516](#)

Examples of Classification Rules

The examples presented in this section deal with user activities involving file classification. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

These examples use a combination of basic classification and Adaptive Content Inspection, the last being an add-on feature. For more information about classification, refer to *Digital Guardian Management Console User's Guide*.

Classify Files Containing More Than One Credit Card Number

This rule classifies a file when it finds more than one instance of the cc_number content pattern in the content buffer. This rule would govern attempts to copy or paste credit card numbers into an instant messenger application or from another document.

Note: This rule requires functionality provided by the add-on feature Digital Guardian Adaptive Content Inspection.

The rule syntax:

```
<greaterThan>  
  <evtBufferEntropyFrequency name="cc_any" />  
  <int value="1" />  
</greaterThan>
```

Classify Word, Excel, and Text Files Containing an Employee Name

This rule classifies a Word, Excel, or text file when it finds at least one instance of the `firstname` content pattern in the content buffer.

This rule assumes the `'firstname'` content pattern contains a reference to the `'employee'` dictionary using the following syntax:

```
(?A: empl oyee/preferredfi rstname)
```

Note: This rule requires functionality provided by the add-on feature Digital Guardian Adaptive Content Inspection.

The rule syntax:

```
<and>
  <greaterThan>
    <evtSrcFileEntityTypeFrequency name="fi rstname"/>
    <int value="0"/>
  </greaterThan>
  <in>
    <evtSrcFileExt/>
    <list>
      <string value="doc"/>
      <string value="txt"/>
      <string value="ppt"/>
    </list>
  </in>
  <greaterThan>
    <evtDestFileEntityTypeFrequency name="fi rstname"/>
    <int value="0"/>
  </greaterThan>
  <in>
    <evtDestFileExt/>
    <list>
      <string value="doc"/>
      <string value="txt"/>
      <string value="ppt"/>
    </list>
  </in>
</and>
```

Classify Files Downloaded From a Particular Domain

This rule classifies files downloaded from widgets.example.com.

The rule syntax:

```
<and>
  <regExp expr="wi dgets\ . exampl e\ . com" >
    <evtDomai n/>
  </regExp>
  <equal >
    <evtOperati onType/>
    <constOpNetTransferDownl oad/>
  </equal >
</and>
```

Classify Excel and Text Files Containing a User Name

This rule classifies Excel and text files that contain at least one instance of the `full_name` content pattern.

Note: This rule requires functionality provided by the add-on feature Digital Guardian Adaptive Content Inspection.

The rule syntax:

```
<and>
<i n>
  <evtSrcFileExt/>
  <list>
    <string value="xls"/>
    <string value="txt"/>
  </list>
</i n>
<greaterThan>
  <evtSrcFileEntityFrequency name="full_name"/>
  <int value="0"/>
</greaterThan>

<i n>
  <evtDestFileExt/>
  <list>
    <string value="xls"/>
    <string value="txt"/>
  </list>
</i n>
<greaterThan>
  <evtDestFileEntityFrequency name="full_name"/>
  <int value="0"/>
</greaterThan>
</and>
```

Classify Excel, Word and Text Files Containing Widgets

This rule classifies Excel, Word, and text files that contain at least one instance of the “widgets” content pattern.

Note: This rule requires functionality provided by the add-on feature Digital Guardian Adaptive Content Inspection.

The rule syntax:

```
<and>
<i n>
<evtSrcFileExt/>
<list>
<string value="doc"/>
<string value="xls"/>
<string value="txt"/>
</list>
</i n>
<greaterThan>
<evtSrcFileEntityTypeFrequency name="widgets"/>
<int value="0"/>
</greaterThan>

<i n>
<evtDestFileExt/>
<list>
<string value="doc"/>
<string value="xls"/>
<string value="txt"/>
</list>
</i n>
<greaterThan>
<evtDestFileEntityTypeFrequency name="widgets"/>
<int value="0"/>
</greaterThan>

</and>
```


Classify Files That Contain More Than 9 Social Security Numbers and 19 Credit Card Numbers

This rule classifies Microsoft Excel, Word, and text files that contain at least 10 instances of the `ss_number` content pattern and at least 20 instances of the `cc_number` content pattern.

Note: This rule requires functionality provided by the add-on feature Digital Guardian Adaptive Content Inspection.

The rule syntax:

```
<and>
<i n>
  <evtDesFileExt/>
  <list>
    <string value="doc" />
    <string value="xls" />
    <string value="txt" />
  </list>
</i n>
<i n>
  <evtSrcFileExt/>
  <list>
    <string value="doc" />
    <string value="xls" />
    <string value="txt" />
  </list>
</i n>
  <greaterThan>
    <evtDestFileEntityFrequency name="ss_number" />
    <int value="10" />
  </greaterThan>
  <greaterThan>
    <evtDestFileEntityFrequency name="cc_number" />
    <int value="20" />
  </greaterThan>
  <greaterThan>
    <evtSrcFileEntityFrequency name="ss_number" />
    <int value="10" />
  </greaterThan>
  <greaterThan>
    <evtSrcFileEntityFrequency name="cc_number" />
```

```
        <i nt val ue="20" />  
    </greaterThan>  
</and>
```

Classify Word, Excel and Text Files

This rule classifies files with .doc, .xls, and .txt file extensions.

The rule syntax:

```
<and>
<i n>
  <evtSrcFileExt/>
  <list>
    <string value="doc"/>
    <string value="xls"/>
    <string value="txt"/>
  </list>
</i n>
<i n>
  <evtSrcFileExt/>
  <list>
    <string value="doc"/>
    <string value="xls"/>
    <string value="txt"/>
  </list>
</i n>
</and>
```

Erasing the Classification Stream on a File

This rule uses DG_DeleteFileClassification to erase the classification tags in the classification stream. This leaves the file without the classification stream. The declassification trigger is the user copying the file into a special declassification folder:

The example calls the declassification function twice to remove the classification stream from both the source and destination files in the copy operation.

```
<like expr="%\declassi fy%">
<evtDestFilePath />
</like>
<equal >
<evtOperationType />
<constOpFileCopy />
</equal >
```

After the trigger part is satisfied, the rule engine executes the declassification part of the rule:

```
<and>
<function name="DG_DeleteFileClassification">
<parameters>
<evtSrcFilePath />
</parameters>
<return>
<varint name="retStatus" scope="event" />
</return>
</function>
<function name="DG_DeleteFileClassification">
<parameters>
<evtDestFilePath />
</parameters>
<return>
<varint name="retStatus" scope="event" />
</return>
</function>
<like expr="%\declassi fy%">
<evtDestFilePath />
</like>
<equal >
<evtOperationType />
<constOpFileCopy />
```

```
</equal >  
</and>
```

In this example, **Execute Rule After Operation** is Yes. This ensures that if a classification policy exists that tags the file based on the copy operation, the declassification rule is executed after the classification to remove the tags from the file.

You can combine this rule with any prompt, such as a Justification Prompt. Note that to use a Block button on the prompt, you must set **Execute Rule After Operation** to No.

Examples of File System Rules

The examples presented in this section deal with user activities involving file activity. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Reading Files on a Specified File Server Generates an Alert

This rule activates whenever a user attempts to read files contained in a network share named 'ShareName' located on 'FileServer'.

The rule syntax:

```
<and>  
<regExp expr="\\\\FileServerName\\ShareName\\.*">  
<evtSrcFilePath/>  
</regExp>  
<equal >  
<evtOperationType/>  
<constOpFileRead/>  
</equal >  
</and>
```

Block File Delete Events for Non-owner of File

This rule activates when a user who is not the owner of a file attempts to delete a file. This rule requires the Personal Removable Media Encryption (RME) feature of Digital Guardian. Use `evtCurrentUserId` to compare the user SID with `evtSrcFileOwnerId` to determine if the current user owns the file.

The rule syntax:

```
<and>
  <not>
    <equal >
      <evtCurrentUserId/>
      <evtSrcFileOwnerId/>
    </equal >
  </not>
  <equal >
    <evtSrcFileHasOwner />
    <bool value="true" />
  </equal >
  <equal >
    <evtOperationType />
    <constOpFileDelete />
  </equal >
</and>
```

Test a User Security ID

The following rule snippet shows you one way to use `evtCurrentUserId` to test the security ID (SID) of the user logged on to the Agent computer.

```
<equal >
  <evtCurrentUserId />
  <string value="userSID" />
</equal >
```

With this snippet you can build personal RME control rules that apply to specific users. Replace `userSID` with the security ID of the user to test.

Note: *userSID* is the user security ID, not the user ID.

Use any third-party tool to determine a user SID. One example tool is PsGetSid which is part of the PsTools suite. You can download this from the Microsoft Web site. Search for PsGetSid.

Microsoft Windows stores SIDs in the Registry. Here is an example user SID:

S-1-5-21-1454471165-1004336348-1606980848-5555

Only Approved Applications Can Open Specified Extensions

This rule activates when an unapproved application attempts to read files with the specified extensions.

The rule syntax:

```
<and>
  <i n>
    <evtSrcFileExt/>
    <l i s t>
      <string value="c"/>
      <string value="cpp"/>
      <string value="h"/>
      <string value="id"/>
      <string value="rgs"/>
      <string value="def"/>
      <string value="rc"/>
    </l i s t>
  </i n>
  <not>
    <i n>
      <curProcessImageName/>
      <l i s t>
        <string value="cl.exe"/>
        <string value="link.exe"/>
        <string value="nmake.exe"/>
        <string value="rc.exe"/>
        <string value="devenv.exe"/>
        <string value="build.exe"/>
        <string value="ssexp.exe"/>
        <string value="ss.exe"/>
        <string value="devenv.com"/>
      </l i s t>
    </i n>
  </not>
</and>
```

Block File Copy or Move on Linux Computer

This rule activates when a user attempts to move or copy a file outside of the specified directory. This example demonstrates the single slash used in Linux directory paths.

The rule syntax:

```
<and>
<regExp expr="/mnt/smbserver/Designs/. *">
<evtSrcFilePath/>
</regExp>
<not>
<regExp expr="/home/j fusco/Designs/. *">
<evtDestFilePath/>
</regExp>
</not>
<in>
<evtOperationType/>
<list>
<constOpFileCopy/>
<constOpFileMove/>
</list>
</in>
</and>
```

Block File or Folder Writes to Removable Drives

This rule would prevent all attempts to write files or folders to removable media including floppy disks and USB drives.

The rule syntax:

```
<and>
<i n>
<evtOperati onType/>
<l i st>
<constOpFi leWri te/>
<constOpFi leCopy/>
<constOpFi leMove/>
</l i st>
</i n>
<equal >
<evtDestDri veType/>
<constDri veRemovabl e/>
</equal >
</and>
```

Block File or Folder Activity on Removable Media

This rule would prevent users from writing to, deleting, or renaming a file or folder on removable media.

```
<and>
  <or>
    <equal >
      <evtDestDriveType />
      <constDriveRemovable />
    </equal >
    <equal >
      <evtSrcDriveType />
      <constDriveRemovable />
    </equal >
  </or>
  <i n>
    <evtOperationType />
    <list>
      <constOpFileWrite />
      <constOpFileDelete />
      <constOpFileRename />
    </list>
  </i n>
</and>
```

Block Starting Instant Messenger Based on Original File Name

This rule activates whenever a user attempts to start an instant messenger application where the original file name property of the executable matches the files listed. This rule would prevent a user from starting a listed application even if the user renamed the executable file.

The rule syntax:

```
<and>
  <equal >
    <evtOperati onType/>
    <constOpAppStart/>
  </equal >
  <i n>
    <curProcessOri gi nal Name/>
    <l i st>
      <stri ng val ue="ai m. exe" />
      <stri ng val ue="ypager. exe" />
      <stri ng val ue="msnmsgr. exe" />
    </l i st>
  </i n>
</and>
```

Block File Overwrites on Removable Drives

This rule prevents attempts to use Microsoft Windows Explorer to overwrite a file on a removable drive.

```
<and>
  <equal >
    <evtDestDriveType />
    <constDriveRemovable />
  </equal >
  <equal >
    <evtIsOverwrite />
    <bool value="true" />
  </equal >
  <equal >
    <curProcessImageName />
    <string value="explorer.exe" />
  </equal >
  <or>
    <equal >
      <evtOperationType />
      <constOpFileMove />
    </equal >
    <equal >
      <evtOperationType />
      <constOpFileCopy />
    </equal >
  </or>
</and>
```

Restrict File Writes, Copies and Moves to List of Approved Removable Drives

This rule would limit attempts to write copy and move files to removable media to a predetermined list of approved drives. In this example, only the PNY (represented by vendor ID 0930) one gigabyte Attache drive (represented by product ID 653d) is permitted. This rule would block file events on all other removable drives.

The rule syntax:

```
<and>
<not>
<and>
<equal >
<evtDestVendorId/>
<string value="0930" />
</equal >
<equal >
<evtDestProductId/>
<string value="653d" />
</equal >
</and>
</not>
<i n>
<evtOperationType/>
<list>
<constOpFileWrite/>
<constOpFileCopy/>
<constOpFileMove/>
</list>
</i n>
<equal >
<evtDestDriveType/>
<constDriveRemovable/>
</equal >
</and>
```

Block File Activity Based on Document Properties

This rule prevents file write, copy, move, and rename events when the file contains more than 100 words, the document author property is Digital Guardian, and the document keywords property starts with "Proprietary".

The rules syntax:

```
<and>
<greaterThan>
<int value="100" />
<evtSrcDocPropertyInt name="number of words" />
</greaterThan>
<regExp expr="proprietary">
<evtSrcDocPropertyString name="keywords" />
</regExp>
<equal>
<string value="digital_guardian" />
<evtSrcDocPropertyString name="author" />
</equal>
<regExp expr="\\financial s\\. *>
<evtSrcFilePath />
</regExp>
<or>
<equal>
<evtOperationType />
<constOpFileWrite />
</equal>
<equal>
<evtOperationType />
<constOpFileCopy />
</equal>
<equal>
<evtOperationType />
<constOpFileMove />
</equal>
<equal>
<evtOperationType />
<constOpFileRename />
</equal>
</or>
</and>
```


Detect File Writes to Unknown or Removable Drives

This rule activates when Microsoft Word format, Microsoft Power Point format or Acrobat documents are written to an unknown or removable drive, such as a USB device.

The rules syntax:

```
<and>
  <i n>
    <evtSrcFileExt/>
    <list>
      <string value="doc"/>
      <string value="ppt"/>
      <string value="pdf"/>
    </list>
  </i n>
  <equal >
    <evtOperationType/>
    <constOpFileWrite/>
  </equal >
  <i n>
    <evtSrcDriveType/>
    <list>
      <constDriveUnknown/>
      <constDriveRemovable/>
    </list>
  </i n>
</and>
```

Block Attempts To Delete or Rename Files Contained in Specified Directory

This rule detects attempts to delete, rename or recycle files that are in a folder named Design Documents.

The rule syntax:

```
<and>
<regExp expr="\Desi gnDocuments\\">
<evtSrcFi lePath/>
</regExp>
<i n>
<evtOperati onType/>
<l i st>
<constOpFi leRename/>
<constOpFi leDel ete/>
<constOpFi leRecycl e/>
</l i st>
</i n>
</and>
```

Allow Users To Write or Copy File Only to Specified Folders on the File Server

This rule activates when a user tries to read or write files on a share other than 'Users' located on the file server named 'EngServer'.

The rule syntax:

```
<and>
<not>
<regExp expr="\\\\EngServer\\Users\\. *">
<evtSrcFilePath/>
</regExp>
</not>
<not>
<regExp expr="\\\\EngServer\\Users\\. *">
<evtDestFilePath/>
</regExp>
</not>
<equal >
<evtSrcDriveType/>
<constDriveRemote/>
</equal >
<i n>
<evtOperationType/>
<l i s t>
<constOpFileRead/>
<constOpFileWrite/>
<constOpFileRename/>
</l i s t>
</i n>
</and>
```

Prevent All File Operations Within A Specified Folder if the Laptop Is Outside the Corporate Network for Over 12 Hours

This rule activates if the user's laptop has not communicated with the DG Server in over 12 hours and the user attempts to read or write to the folder named 'Source Code'.

The rule syntax:

```
<and>
<regExp expr="//SourceCode\\">
<evtSrcFilePath/>
</regExp>
<greaterThan>
<agentLastServerComm/>
<int value=" 720" />
</greaterThan>
<equal >
<agentIsLaptop/>
<bool value=" true" />
</equal >
</and>
```

User Cannot Compress or Zip Source Files

This rule activates when a user attempts to compress the specified files using WinZip32.exe.

The rule syntax:

```
<and>
<i n>
  <evtSrcFileExt/>
  <list>
    <string value="c"/>
    <string value="cpp"/>
    <string value="h"/>
    <string value="id"/>
    <string value="rgs"/>
    <string value="def"/>
    <string value="rc"/>
  </list>
</i n>
<equal >
  <evtOperationType/>
  <constOpFileRead/>
</equal >
<equal >
  <curProcessImageName/>
  <string value=" winzip32.exe " />
</equal >
</and>
```

User Cannot Send Specified File Extensions Using IM Applications

This rule blocks users from attaching files with specified extensions to email, Web-based email or chat sessions.

The rule syntax:

```
<and>
  <i n>
    <evtSrcFileExt/>
    <l i s t>
      <string value="c" />
      <string value="cpp" />
      <string value="h" />
      <string value="id" />
      <string value="rgs" />
      <string value="def" />
      <string value="rc" />
    </l i s t>
  </i n>
  <i n>
    <curProcessImageName/>
    <l i s t>
      <string value="iexpl ore. exe" />
      <string value="msmsgs. exe" />
      <string value="ymsgr. exe" />
      <string value="outlook. exe" />
    </l i s t>
  </i n>
</equal >
<evtOperati onType/>
<constOpNetTransferUpl oad/>
</equal >
</and>
```

Do Not Allow Kazaa To Share Music Files

This rule prevents an application identified as kazaa.exe from sharing .mp3 or .wma files. You could modify this rule to prevent kazaa.exe from starting at all, or you could edit the file extensions that kazaa.exe is allowed to work with.

The rule syntax:

```
<and>
<regExp expr="kazaa">
<curProcessCompanyName/>
</regExp>
<regExp expr="kazaa\.exe">
<curProcessImageName/>
</regExp>
<in>
<evtSrcFileExt/>
<list>
<string value="mp3"/>
<string value="wma"/>
</list>
</in>
</and>
```

Prevent Users From Changing the Extensions of Source Code Files

This rule prevents users from changing the file extensions of source code files. You could use this rule in conjunction with another rule that prevents users from copying or deleting files with the source code extension.

The rule syntax:

```
<and>
<not>
<equal >
<evtSrcFileExt/>
<evtDestFileExt/>
</equal >
</not>
<i n>
<evtSrcFileExt/>
<l i s t>
<string value="cpp"/>
<string value="c"/>
<string value="h"/>
</l i s t>
</i n>
<i n>
<evtOperationType/>
<l i s t>
<constOpFileRename/>
<constOpFileCopy/>
<constOpFileMove/>
</l i s t>
</i n>
</and>
```


Prevent Visitor Account Users From Copying Files to Removable Drives

This rule prevents users logged on under a Visitor account from copying local files to remote or removable drives. This rule takes advantage of Windows environment variables to identify the domain and account of the user.

The rule syntax:

```
<and>
<and>
<i n>
<evtDestDriveType />
<l i s t>
<constDriveRemote />
<constDriveRemovable />
<constDriveUnknown />
</l i s t>
</i n>
<equal >
<environmentVariable name="USERDOMAIN" />
<string value="DGUIARDIAN" />
</equal >
<equal >
<environmentVariable name="USERNAME" />
<string value="Visitor" />
</equal >
</and>
<equal >
<evtOperationType />
<constOpFileCopy />
</equal >
</and>
```

Block Registry Editor From Starting if User Name Is "tester" and Agent Driver Is Running

This retrieves the Logon User Name from the Windows Registry and verifies that the agent driver is running. As a Control rule with an Action type of Block, it would prevent specific users from opening the Windows Registry Editor.

The rule syntax:

```
<and>
<equal >
<string value = "tester"/>
<registryString key= "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer" value ="Logon User Name"/>
</equal >
<equal >
<bool value= "true"/>
<driverRunning name="dgtmon"/>
</equal >
<equal >
<evtOperationType/>
<constOpAppStart/>
</equal >
<equal >
<curProcessImageName/>
<string value="regedit.exe"/>
</equal >
</and>
```

Prevent Users Who Are not Administrators From Renaming or Using Save As To Save Executable Files to New Filenames

This blocks users who do not have administrator rights from using the Save As and file rename operations for executable files.

The Rule Syntax:

```
<and>
  <equal >
    <evtIsLocalAdmin/>
    <bool value="false" />
  </equal >
  <like expr="*.exe">
    <evtDestFileExt />
  </like>
  <in>
    <evtOperationType />
    <list>
      <constOpFileRename />
      <constOpFileSaveAs />
    </list>
  </in>
</and>
```

Prevent RME From Attaching to Specified USB Drives

This rule blocks RME from attaching to the USB device specified by the `afedevice` expression. When the rule evaluates to true, RME does not attach to the specified device and does not encrypt files. This rule should be executed only if the target drive is removable.

Rule Action: Block

Rule Syntax:

```
<and>
  <like expr="afedevice\usb#vid(178c)%">
    <evtSrcFilePath/>
  </like>
  <equal>
    <evtOperationType />
    <constOpFileOpen />
  </equal>
  <equal>
    <evtSrcDriveType/>
    <constDriveRemovable/>
  </equal>
</and>
```

The format of the device name in the expression for the source file path is the following:

```
afedevice\usb#vid(%s)pid(%s)vendor(%s)sn(%s)
```

where:

- `usb#vid` — A string that specifies the device vendor ID
- `pid` — A string that specifies the product ID for the device
- `vendor` — A string that specifies the name of the vendor
- `sn` — A string that specifies the device serial number

Encrypt Files Based on Rule Evaluation

For information on encrypting files based on rule evaluation, refer to “Removable Media Encryption” in the *Digital Guardian Management Console User’s Guide*.

Examples of Network Rules

The examples presented in this section deal with user activities involving network activity. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Certain Subnets Cannot Connect to Sensitive Computers

This rule activates when a user on subnet 10.10.10.0/24 attempts to access servers with the IP address of 10.10.10.130 or 10.10.10.13.

The rule syntax:

```
<and>
<i n>
<evtRemoteAddress/>
<l i s t>
<i pAddress value="10. 10. 10. 130"/>
<i pAddress value="10. 10. 10. 13"/>
</l i s t>
</i n>
<i pMask mask="10. 10. 10. 0/24">
<evtRemoteAddress/>
</i pMask>
</and>
```

Allow Remote Connections Only to Authorized Servers

This rule activates when users on a particular subnet attempt to access a computer whose IP address is not listed in the rule.

The rule syntax:

```
<and>
<not>
<i n>
<evtRemoteAddress/>
<l i s t>
<i pAddress val ue="10. 10. 10. 4"/>
<i pAddress val ue="10. 10. 10. 2"/>
<i pAddress val ue="10. 10. 10. 161"/>
</l i s t>
</i n>
</not>
<i pMask mask="10. 10. 10. 0/24">
<evtRemoteAddress/>
</i pMask>
</and>
```

Block Connections to Computers Outside the Subnet

This rule prevents a computer located in a particular subnet from accessing any computer that is outside its own subnet.

The rule syntax:

```
<and>
<i pMask mask="10. 10. 11. 0/24" >
<agentIPAddress/>
</i pMask>
<not>
<i pMask mask="10. 10. 11. 0/24" >
<evtRemoteAddress/>
</i pMask>
</not>
<equal >
<evtOperationType/>
<constOpNetwork/>
</equal >
</and>
```


Block Downloads of Mp3 Files

This rule prevents users from downloading files with an mp3 extension.

The rule syntax:

```
<and>  
<equal >  
<evtSrcFileExt/>  
<string value="mp3" />  
</equal >  
<equal >  
<evtOperationType/>  
<constOpNetTransferDownload/>  
</equal >  
</and>
```

Block Access to Specified Ports on Remote Computers

This rule activates when the user attempts to access the listed ports on the specified computers.

The rule syntax:

```
<and>
<i n>
<evtRemotePort/>
<l i s t>
<i n t   v a l u e=" 445" />
<i n t   v a l u e=" 3567" />
<i n t   v a l u e=" 2990" />
</l i s t>
</i n>
<i n>
<evtRemoteAddress/>
<l i s t>
<i p A d d r e s s   v a l u e=" 10. 10. 10. 4" />
<i p A d d r e s s   v a l u e=" 10. 10. 10. 2" />
</l i s t>
</i n>
</and>
```

Block Uploads if too Many Adapters Connect to Known Network

This rule would block uploads if you are on a known network and have more than two active network adapters.

The rule syntax:

```
<and>
  <equal >
    <dnsHostAvai l a b l e  name="10. 10. 10. 1"/>
    <bool  val ue=" true"/>
  </equal >
  <greaterThan>
    <agentAdapterCount/>
    <i n t  val ue="2"/>
  </greaterThan>
  <equal >
    <evtOperati onType/>
    <constOpNetTransferUpl oad/>
  </equal >
</and>
```

Detect FTP Connections to Unauthorized Computers

This rule activates when users attempt to connect to unauthorized FTP sites using the applications listed in the rule.

The rule syntax:

```
<and>
<i n>
<evtRemotePort/>
<l i s t>
<i n t   v a l u e=" 445" />
<i n t   v a l u e=" 3567" />
<i n t   v a l u e=" 2990" />
</l i s t>
</i n>
<i n>
<curProcessImageName/>
<l i s t>
<s t r i n g   v a l u e=" f t p . e x e" />
<s t r i n g   v a l u e=" w s f t p . e x e" />
<s t r i n g   v a l u e=" i e x p l o r e . e x e" />
</l i s t>
</i n>
<not>
<i n>
<evtRemoteAddress/>
<l i s t>
<i p A d d r e s s   v a l u e=" 10 . 10 . 10 . 4" />
<i p A d d r e s s   v a l u e=" 10 . 10 . 10 . 2" />
</l i s t>
</i n>
</not>
</and>
```

Allow Only Web Browsers and Web Services To Make Network Connections Using Ports 80 and 443

This rule will prevent unauthorized applications from connecting to the internet. In addition, it requires authorized applications to use the HTTP or HTTPS protocols to connect to the internet. This rule is effective in blocking spyware and Trojan Horse programs from communicating using ports 80 and 443.

The rule syntax:

```
<and>
<i n>
<evtRemotePort/>
<l i s t>
<i n t v a l u e="80" />
<i n t v a l u e="443" />
</l i s t>
</i n>
<not>
<i n>
<curProcessMD5ContentHash/>
<l i s t>
<!-- [MD5 Hash for IE - add appropriate DG MD5 hash values] -->
<md5Hash v a l u e="{ dg_hash_val ue}" />
<md5Hash v a l u e="{ dg_hash_val ue}" />
<!-- [MD5 Hash for servi cehost.exe - add appropriate DG MD5 hash
v a l u e] -->
<md5Hash v a l u e="{ dg_hash_val ue}" />
</l i s t>
</i n>
</not>
</and>
```

Block Network Uploads From the Specified Directory

This rule blocks upload events from the specified directory.

The rule syntax.

```
<and>
<!-- Specifies directory from which to prevent upload. -->
<regExp expr="//SourceCode\\"*>
<evtSrcFilePath/>
</regExp>
<equal >
<!-- Defines operation as network upload -->
<evtOperati onType/>
<constOpNetTransferUpl oad/>
</equal >
</and>
```

Only Allow NETBIOS Calls on Port 139 in the Corporate Subnet

This rule allows only the corporate subnet to connect to port 139. All other computers are blocked.

The rule syntax:

```
<and>
<equal >
<evtLocal Port/>
<i nt val ue=" 139"/>
</equal >
<not>
<i pMask mask="10. 10. 10. 0/24">
<evtRemoteAddress/>
</i pMask>
</not>
<equal >
<evtOperati onType/>
<constOpNetwork/>
</equal >
</and>
```

Limit Browser Access to Company Internal Web Sites

This rule limits user access to URLs that contain example.com. Other URLs are blocked.

The rule syntax:

```
<and>
<not>
<like expr="%.example.com%">
<evtUrlPath/>
</like>
</not>
<equal>
<evtProtocolType/>
<constProtocol HTTP/>
</equal>
<equal>
<evtOperationType/>
<constOpNetwork/>
</equal>
</and>
```

Block Network Operations Over WiFi Connections

Prevent users from performing network operations over wireless local area network (WiFi) connections.

```
<and>
<in>
<evtWirelessEncryption />
<list>
<constAES />
<constNone />
<constTKIP />
<constWEP />
</list>
</in>
<equal>
<evtOperationType />
<constOpNetwork />
</equal>
</and>
```

Examples of Print Rules

The examples presented in this section deal with user activities involving printers and printing. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Note: You must add a rule to a policy for it to take effect.

Block Printing Unless the DG Server Is Available

The following rule uses the property `serverAvailable` to prevent users from printing unless the DG Server is available.

The rule syntax:

```
<and>
  <equal >
    <serverAvailable />
    <boolean value="true" />
  </equal >
  <equal >
    <eventOperationType />
    <constantPrint />
  </equal >
</and>
```

Note: In earlier releases, you would have used `isAgentRegistered` for this agent status test. `isAgentRegistered` cannot be reliably used for this purpose with 5.3.2 unless you set `<registerOnPropertyChanged>` to 1 in the `CONFIG.XML` file:

```
<registerOnPropertyChanged pushDurationUpdate="1">1</registerOnPropertyChanged>
```

Generate Alert When Printing Specified Files

The following rule, if specified as an alert, sends a notification to DG Server every time files with the specified extensions are printed.

The rule syntax:

```
<and>
<i n>
<evtSrcFileExt/>
<l i s t>
<stri ng val ue="doc" />
<stri ng val ue="pdf" />
<stri ng val ue="xl s" />
<stri ng val ue="ppt" />
</l i s t>
</i n>
<equal >
<evtOperati onType/>
<constOpPri nt/>
</equal >
</and>
```

Allow Printing Only on Authorized Printers

The below rule could be used to block printing of documents to any printers on the corporate network other than the ones listed in the rule.

The rule syntax:

```
<and>  
<not>  
<regExp expr="mi crosoft. *" >  
<evtPri nterName/>  
</regExp>  
</not>  
<equal >  
<evtOperati onType/>  
<constOpPri nt/>  
</equal >  
</and>
```

Block Printing When Laptop Computers Are Not on the Corporate Network

The rule would prevent a laptop from being able to print if it is not on the corporate network. Even when the laptop is on the corporate network, printing is allowed only to specified printers.

The rule syntax:

```
<and>
<not>
<regExp expr="mi crosoft. *" >
<evtPri nterName/>
</regExp>
</not>
<equal >
<agentGatewayMac/>
<macAddress val ue="00-09-6B-E3-78-4A" />
</equal >
<equal >
<evtOperati onType/>
<constOpPri nt/>
</equal >
<equal >
<agentIsLaptop/>
<bool val ue=" true" />
</equal >
</and>
```

Block Printing From Specified Computers

This rule would prevent all print actions from the specified computers.

The rule syntax:

```
<and>
<i n>
<agentIPAddress/>
<l i s t>
<i pAddress  val ue="10. 10. 10. 121"/>
<i pAddress  val ue="10. 10. 10. 130"/>
<i pAddress  val ue="10. 10. 10. 186"/>
</l i s t>
</i n>
<equal >
<evtOperati onType/>
<constOpPri nt/>
</equal >
</and>
```

Block Printing of Certain File Extensions After 6:00 PM and Before 8:00 AM

This rule stops users from printing files that have specified extensions after business hours.

```
<and>
  <i n>
    <evtSrcFileExt/>
    <list>
      <string value="xlsx"/>
      <string value="docx"/>
      <string value="c"/>
      <string value="cpp"/>
      <string value="h"/>
      <string value="pptx"/>
      <string value="txt"/>
      <string value="rtf"/>
    </list>
  </i n>
  <or>
    <greaterThan>
      <agentCurrentTime/>
      <time value="06:00 pm"/>
    </greaterThan>
    <lessThan>
      <agentCurrentTime/>
      <time value="08:00 am"/>
    </lessThan>
  </or>
  <equal>
    <evtOperationType/>
    <constOpPrint/>
  </equal>
</and>
```

Examples of CD/DVD Rules

The examples presented in this section deal with user activities involving CD/DVD drives. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Note: You must add a rule to a policy for it to take effect.

Allow CD/DVD Writing Using Only Authorized Applications

This rule prevents unauthorized applications from writing to CD/DVD drives. In this example, only nero and rxmon are allowed to burn CD or DVDs.

The rule syntax:

```
<and>
<not>
  <i n>
    <curProcessImageName/>
    <l i s t>
      <stri ng val ue="nero. exe" />
      <stri ng val ue="rxmon. exe" />
    </l i s t>
  </i n>
</not>
<equal >
<evtOperati onType/>
<constOpCDBurn/>
</equal >
</and>
```

Allow CD/DVD Writing Only During Office Hours

This rule prevents CD/DVD writing before 10:00 AM and after 5:00 PM.

The rule syntax:

```
<and>
<lessThan>
  <agentCurrentTime/>
  <time value=" 10: 00 AM" />
</lessThan>
<greaterThan>
  <agentCurrentTime/>
  <time value=" 05: 00 PM" />
</greaterThan>
<equal >
  <evtOperationType/>
  <constOpCDBurn/>
</equal >
</and>
```

Prevent Laptops From Writing to CD/DVD Drives

This rule prevents laptops from writing to CD/DVD drives.

The rule syntax:

```
<and>
<equal >
  <evtOperationType/>
  <constOpCDBurn/>
</equal >
<equal >
  <agentIsLaptop/>
  <bool value=" true" />
</equal >
</and>
```

Block File Reads From CD Drives

This rule prevents users from reading files stored on CD Drives.

```
<and>
<equal >
<!--Defines the event as File Read -->
<evtOperationType/>
<constOpFileRead/>
</equal >
<i n>
<evtSrcDriveType/>
<list>
<!--evtSrcDriveType property defines the source drive type as
Removable or CDRom. -->
<constDriveRemovable/>
<constDriveCDRom/>
</list>
</i n>
</and>
```


Block File or Folder Activity on Removable Media

This rule prevents users from writing to, deleting, or renaming a file or folder on removable media.

```
<and>
  <or>
    <equal >
      <evtDestDriveType />
      <constDriveRemovable />
    </equal >
    <equal >
      <evtSrcDriveType />
      <constDriveRemovable />
    </equal >
  </or>
  <i n>
    <evtOperationType />
    <list>
      <constOpFileWrite />
      <constOpFileDelete />
      <constOpFileRename />
    </list>
  </i n>
</and>
```

Examples of Device Rules

The examples presented in this section deal with user activities involving devices. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Note: You must add rules to a policy for them to take effect.

Control Web Camera Video Stream Use

The following rule allows Skype, Chrome and launcher_main to open the video camera for video. The audio portion is not affected. Replace the curProcessImageName string values with the names of the programs you allow to use video. Programs that you do not list cannot open cameras to use video streams.

```
<and>
  <not>
    <i n>
      <curProcessImageName />
      <list>
        <string value="skype.exe" />
        <string value="launcher_main.exe" />
        <string value="chrome.exe" />
      </list>
    </i n>
  </not>
  <equal >
    <evtSrcDeviceClass />
    <string value="image" />
  </equal >
  <equal >
    <evtOperationType />
    <constOpDeviceOpen />
  </equal >
</and>
```

Examples of Application Data Exchange Rules

The examples presented in this section deal with user activities involving application data exchange events. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Note: You must add a rule to a policy for it to take effect.

Generate an Alert When Specified Applications Perform Application Data Exchange Actions

This rule activates when a user performs an application data exchange activity using one of the applications listed in the rule:

The rule syntax:

```
<and>
<i n>
<curProcessImageName/>
<l i s t>
<string value="textpad.exe"/>
<string value="notepad.exe"/>
<string value="iexplorer.exe"/>
<string value="unipad.exe"/>
</l i s t>
</i n>
<equal >
<evtOperationType/>
<constOpAdePaste/>
</equal >
</and>
```

Block Paste Actions Into IM Applications

This rule activates if the user attempts to paste into the applications listed in the rule. In the example applications are identified using either the executable name or the MD5 hash of the application.

The rule syntax:

```
<and>
<or>
<i n>
<curProcessMD5ContentHash/>
<l i s t>
<md5Hash val ue="{eaf0a7c3-381f-42eb-fdf6-dd7d87266158}" />
<md5Hash val ue="{eaf0a7c3-381f-42eb-fdf6-dd7d82766159}" />
</l i s t>
</i n>
<i n>
<curProcessI mageName/>
<l i s t>
<stri ng val ue="ymsgr. exe" />
<stri ng val ue="msmsg. exe" />
<stri ng val ue="ai m. exe" />
</l i s t>
</i n>
</or>
<equal >
<evt0perati onType/>
<const0pAdePaste/>
</equal >
</and>
```

Block Application Data Exchange Using Unauthorized Applications

This rule activates when the user attempts to perform clipboard cut/copy or paste operations using applications not listed in the rule. In this example, Word, Excel, and Acrobat Reader are listed as authorized applications.

The rule syntax:

```
<and>
<or>
<not>
<i n>
<curProcessMD5ContentHash/>
<l i s t>
<md5Hash val ue="{eaf0a7c3-381f-42eb-fdf6-dd7d87266158}" />
<md5Hash val ue="{eaf0a7c3-381f-42eb-fdf6-dd7d87266159}" />
</l i s t>
</i n>
</not>
<not>
<i n>
<curProcessI mageName/>
<l i s t>
<stri ng val ue="wi nword. exe" />
<stri ng val ue="excel . exe" />
<stri ng val ue="acroread32. exe" />
</l i s t>
</i n>
</not>
</or>
<equal >
<evtOperati onType/>
<constOpAdePaste/>
</equal >
</and>
```

Prevent Users From Exchanging Content From Microsoft Word Into AOL Instant Messenger

This rule activates when the user attempts to perform an application data exchange from Microsoft Word into AOL Instant Messenger.

The rule syntax:

```
<and>
<regExp expr="ameri ca onl i ne">
<curProcessCompanyName/>
</regExp>
<regExp expr="i nstant messenger">
<curProcessProductName/>
</regExp>
<equal >
<evtSrcFileExt/>
<string value="doc" />
</equal >
<equal >
<evtOperati onType/>
<constOpAdePaste/>
</equal >
</and>
```

Prevent Users From Cutting or Dragging and Dropping From any Process Containing the String "acro"

This rule blocks any attempt to cut or drag and drop from any process named with the string "acro". In this case, the rule would prevent users from copying or cutting content from a range of Adobe Acrobat® products, including Acrobat Professional (acrobat.exe) and Acrobat Reader® (acrord32.exe).

Note: This rule does allow users to perform print screen actions.

The rule syntax:

```
<and>
<equal >
<evtOperationType/>
<constOpAdeCut/>
</equal >
<like expr="%acro%">
<evtSrcProcessImageName/>
</like>
</and>
```

Block Screen Capture of File Content in Selected Processes

Prevent users from using supported screen capture software (static and video) and the Microsoft Windows print screen options to capture data from specified processes. To block other processes, add them to the list. You can use a component rule to list the processes.

```
<or>
<and>
  <i n>
    <evtOperati onType />
    <l i s t>
<!-- Block Al t+Print Screen keys -->
    <constOpAdePri ntProcess />
<!-- Block Print Screen key -->
    <constOpAdePri ntScreen />
  </l i s t>
</i n>
<not>
  <i n>
    <curProcessI mageName />
    <l i s t>
      <string val ue="whi tel i stedproc2. exe" />
    </l i s t>
  </i n>
</not>
</and>
<and>

  <equal >
    <!-- Block i mage capture or vi deo capture by capturi ng tool s such
as Windows snippi ng tool , Snagl t, Ji ng, Cantasia Recorder, Camstu-
dio, Capti vate. Recommend Putti ng +SR fl ag on process fl ag fi le. -->
    <evtOperati onType />
    <constOpAdeScreenCapture />
  </equal >

<not>
<i n>
  <curProcessI mageName />
  <l i s t>
    <string val ue="excel . exe" />
    <string val ue="acrord32. exe" />
    <string val ue="fi refox. exe" />
```



```
<string value="winnvnc4.exe" />
<string value="orpl at.exe" />
<string value="acrobat.exe" />
<string value="iexpl ore.exe" />
<string value="winproj.exe" />
<string value="win scp.exe" />
<string value="superputty.exe" />
<string value="chrome.exe" />
<string value="regsvr32.exe" />
<string value="fil ezi l l a.exe" />
<string value="winword.exe" />
<string value="powerpnt.exe" />
</list>
</in>
</not>
</and>
</or>
```

Examples of Email Rules

The examples presented in this section deal with user activities involving email messages and attachments. These examples represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

In Digital Guardian, you can block and prompt for email events, just as you can for any other event. For more information on encryption, refer to *Digital Guardian Management Console User's Guide*.

Block Emails Sent to BCC Recipients

This rule would block any emails sent to a BCC recipient, effectively preventing any BCC messages. In the case of an email sent to multiple recipients, only BCC recipients would be blocked. Recipients in the To: and CC: fields would be sent normally.

```
<and>
<equal >
<evtMailRecipientTypes/>
<constMailBCC/>
</equal >
<equal >
<evtOperationType/>
<constOpSendMail/>
</equal >
</and>
```

Block Excel Files Sent to External Recipients

This rule prevents Excel files from being mailed to recipients outside of your organization.

```
<and>
<not>
<like expr="%yourenterprise.com" match="all">
<evtMailRecipients/>
</like>
</not>
<like expr="*.xls">
<evtSrcFilePath/>
</like>
<equal>
<evtOperationType/>
<constOpSendMail/>
</equal>
</and>
```

Block All Emailed .doc File Attachments

This blocks all .doc files from being sent as email attachments.

```
<and>
<like expr="%.doc">
<evtSrcFilePath/>
</like>
<equal>
<evtOperationType/>
<constOpSendMail/>
</equal>
</and>
```

Block Email Attachments Greater Than 1KB in Size

This rule would block email attachments larger than one kilobyte. Note that the file size is given in bytes.

```
<and>
<greaterThan>
<evtMailAttachmentSize/>
<intValue="1024"/>
</greaterThan>
<equal>
<evtOperationType/>
<constOpSendMail/>
</equal>
</and>
```

Block Emails Larger Than 1Kb in Size

This rule would block emails larger than one kilobyte. Unlike the rule governing attachment sizes, this rule governs the total size of the email, including the body of the message.

```
<and>
  <greaterThan>
    <evtMailTotalSize/>
    <intValue="1000"/>
  </greaterThan>
  <equal>
    <evtOperationType/>
    <constOpSendMail/>
  </equal>
</and>
```

Test Email Address Against List of Addresses

The following SendMail block rule uses Op with Like to evaluate email recipients against a list of domains for the company Lol. This rule prevents users from sending email to addresses outside of the Lol corporate network.

Note: The % wildcard character at the end of the domain names (.com, .org and the others in this example) is required to ensure that DG matches the domains correctly. Because some email systems treat addresses differently, the % character makes your Send Mail rules work across all protocols and clients.

```
<not>
  <in op="Like" match="all">>
    <evtMailRecipients />
    <list>
      <stringValue="%lol.com"/>
      <stringValue="%lol.org"/>
      <stringValue="%lol.biz"/>
      <stringValue="%lol.in"/>
    </list>
  </in>
</not>
<equal>
```

```
<evtOperationType />  
<constOpSendMail />  
</equal >
```

This rule evaluates to True when one or more of the recipient email addresses in the message does not match any of the strings provided. The True result blocks the email. In general, a better way to solve this problem is with a regular expression. For information about regular expressions, refer to [“Regular Expressions” on page 231](#).

Examples of Filter Rules

You can create rules that define which activities Digital Guardian should filter out of reports. For example, you could choose to filter File Read events from some common applications. If you want to filter all activity from an application, use a Trusted Process rule.

Note: You must add a rule to a policy for the rule to take effect.

Filter File Read Events From Specified Applications

This rule filters File Read events from the listed applications.

The rule syntax:

```
<and>
<i n>
  <curProcessImageName/>
  <!-- Lists applications to filter out -->
  <list>
    <string value="cl.exe" />
    <string value="link.exe" />
    <string value="make.exe" />
    <string value="nmake.exe" />
    <string value="rc.exe" />
    <string value="devenv.exe" />
    <string value="cvtres.exe" />
    <string value="buid.exe" />
    <string value="ssexp.exe" />
    <string value="winword.exe" />
    <string value="excel.exe" />
    <string value="notepad.exe" />
    <string value="explorer.exe" />
  </list>
</i n>
<equal>
  <!-- Specifies an operation type of File Read -->
  <evtOperationType/>
  <constOpFileRead/>
</equal>
</and>
```

Filter Microsoft Outlook Express Noise

This rule filters events generated when Microsoft Outlook Express accesses the Windows Address Book (wab) file.

The rule syntax:

```
<and>
<equal >
<curProcessImageName/>
<string value="msimn.exe"/>
</equal >
<equal >
<evtSrcFileExt/>
<string value="wab"/>
</equal >
</and>
```

Filter MSN Toolbar Update Noise

This rule filters events generated for xml and log files used by the MSN Toolbar update utility.

The rule syntax:

```
<and>
<in>
<evtSrcFileExt/>
<list>
<string value="xml"/>
<string value="log"/>
</list>
</in>
<equal >
<curProcessImageName/>
<string value="msnappau.exe"/>
</equal >
</and>
```


Filter Noisy File Extensions

This rule filters events generated for the following file extensions:

- oab - Outlook address book
- lnk - Windows shortcuts
- tmp - temporary files or folders
- dic - dictionary files used by various applications

Because the rule specifies that the source file and the destination file have the same extension, this rule does not filter attempts to rename one file to another file that has one of these extensions.

The rule syntax:

```
<or>
<i n>
<evtSrcFileExt/>
<list>
<string value="lnk" />
<string value="oab" />
<string value="tmp" />
<string value="dic" />
</list>
</i n>
<i n>
<evtDestFileExt/>
<list>
<string value="lnk" />
<string value="oab" />
<string value="tmp" />
<string value="dic" />
</list>
</i n>
</or>
```

Filter Noisy Microsoft Office Application Files

This rule filters events generated by Microsoft Office applications using the file `index.dat`. This file stores log information for various Microsoft applications.

The rule syntax:

```
<and>
<regExp expr=". *\\i ndex\\. dat">
<evtSrcFi lePath/>
</regExp>
<i n>
<curProcessI mageName/>
<l i st>
<stri ng val ue="outl ook. exe" />
<stri ng val ue="wi nword. exe" />
<stri ng val ue="excel . exe" />
<stri ng val ue="powerpoi nt. exe" />
<stri ng val ue="i expl ore. exe" />
</l i st>
</i n>
</and>
```

Filter Microsoft Outlook Noise

This rule filters file events generated by Microsoft Outlook when using data files, log files, and offline content files.

The rule syntax:

```
<and>
  <i n>
    <evtSrcFileExt/>
    <l i s t>
      <string value="dat"/>
      <string value="log"/>
      <string value="ost"/>
    </l i s t>
  </i n>
  <equal >
    <curProcessImageName/>
    <string value="outlook.exe"/>
  </equal >
</and>
```

Filter X1 File Activity Noise

This rule filters data file activity generated by the X1 desktop search utility.

The rule syntax:

```
<and>
  <equal >
    <curProcessImageName/>
    <string value="x1.exe"/>
  </equal >
  <equal >
    <evtSrcFileExt/>
    <string value="dat"/>
  </equal >
</and>
```

Examples of Trusted Process Rules

You can create rules that define which processes Digital Guardian should ignore. For example, you could choose to make an anti-virus application a trusted process. In a trusted process, Digital Guardian identifies the trusted process when the process starts, and ignores any further activity from that process.

To filter only selected activity from an application, use a filter rule.

Note: You must add a rule to a policy for it to take effect.

Identify Notepad as a Trusted Process

This rule uses a regular expression to identify Notepad as a trusted process.

The rule syntax:

```
<regExp expr="notepad\.exe">  
<curProcessImageName/>  
</regExp>
```

Identify a List of Trusted Applications

This rule creates a list of trusted process applications.

The rule syntax:

```
<i n>  
<curProcessImageName/>  
<l i s t>  
<string value="cl.exe" />  
<string value="link.exe" />  
<string value="make.exe" />  
<string value="nmake.exe" />  
<string value="rc.exe" />  
<string value="devenv.exe" />  
<string value="cvtres.exe" />  
<string value="build.exe" />  
<string value="ssexp.exe" />  
</l i s t>  
</i n>
```

Examples of File Capture Rules

The examples presented in this section deal with using file capture while monitoring and controlling user file operations. They represent typical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Note: You must add a rule to a policy for it to take effect.

Capture Source Files in File Recycle or Delete Events

This rule captures source files involved in recycle or delete events.

The rule syntax:

```
<and>
<function name="DG_CaptureFile">
  <parameters>
    <currentEvent/>
    <currentAssociatedFileInfo/>
    <currentRuleId/>
    <int value="2"/> <!-- Enable Hashing: Def=0, Enabled=1, Disabled=2 -->
    <int value="0"/> <!-- Hash Type: Def=0, MD5=104, SHA1=103, SHA256=102, SHA512=101 -->
    <int value="2"/> <!-- Disposition: Def=0, Auto=1, Src=2, Dst=3, Both=4 -->
    <int value="1"/> <!-- Capture Alg: Def=0, Synchronous=1, Asynchronous=2 -->
    <int value="2"/> <!-- Capture Mode: Def=0, Record_Only=1, Record_And_File=2 -->
    <int value="2"/> <!-- Transfer Mode: Def=0, Auto_Upload=1, On_Demand=2 -->
    <int value="0"/> <!-- Max Capture File Size in MB: Def=0 -->
  </parameters>
  <return>
    <variant name="fc_result" scope="event" />
  </return>
</function>
<in>
<evtSrcFileExt/>
<list>
  <string value="doc"/>
  <string value="docx"/>
  <string value="xls"/>
  <string value="xlsx"/>
  <string value="ppt"/>
  <string value="pptx"/>
  <string value="txt"/>
  <string value="rtf"/>
  <string value="pdf"/>
```

```

</list>
</in>

<in>
  <evtOperationType/>
  <list>
    <constOpFileRecycle/>
    <constOpFileDelete/>
  </list>
</in>
</and>

```

Capture Files That Users Write to Removable Media

This rule prevents all attempts to write files to removable media such as USB drives. It captures the source files involved in the operation.

The rule syntax:

```

<and>
<function name="DG_CaptureFile">
  <parameters>
    <currentEvent/>
    <currentAssociatedFileInfo/>
    <currentRuleId/>
    <int value="2"/> <!-- Enable Hashing: Def=0, Enabled=1, Disabled=2 -->
    <int value="0"/> <!-- Hash Type: Def=0, MD5=104, SHA1=103, SHA256=102, SHA512=101 -->
    <int value="2"/> <!-- Disposition: Def=0, Auto=1, Src=2, Dst=3, Both=4 -->
    <int value="1"/> <!-- Capture Alg: Def=0, Synchronous=1, Asynchronous=2 -->
    <int value="2"/> <!-- Capture Mode: Def=0, Record_Only=1, Record_And_File=2 -->
    <int value="2"/> <!-- Transfer Mode: Def=0, Auto_Upload=1, On_Demand=2 -->
    <int value="0"/> <!-- Max Capture File Size in MB: Def=0 -->
  </parameters>
  <return>
    <variant name="fc_result" scope="event" />
  </return>
</function>

<in>
  <evtOperationType/>
  <list>
    <constOpFileWrite/>
    <constOpFileCopy/>
    <constOpFileMove/>
  </list>
</in>
<equal>
  <evtDestDriveType/>

```

```
<constDriveRemovable/>
</equal>
</and>
```

Capture and Hash the Source File in a File Copy Operation

This rule captures the source file and file record when a user performs a file copy. After the capture, the Agent applies the MD5 hash to the captured file because Hash Type is set to 104.

The rule syntax:

```
<and>
<function name="DG_CaptureFile">
  <parameters>
    <currentEvent/>
    <currentAssociatedFileInfo/>
    <currentRuleId/>
    <int value="1"/> <!-- Enable Hashing: Def=0, Enabled=1, Disabled=2 -->
    <int value="104"/> <!-- Hash Type: Def=0, MD5=104, SHA1=103, SHA256=102, SHA512=101 -->
    <int value="2"/> <!-- Disposition: Def=0, Auto=1, Src=2, Dst=3, Both=4 -->
    <int value="1"/> <!-- Capture Alg: Def=0, Synchronous=1, Asynchronous=2 -->
    <int value="2"/> <!-- Capture Mode: Def=0, Record_Only=1, Record_And_File=2 -->
    <int value="0"/> <!-- Transfer Mode: Def=0, Auto_Upload=1, On_Demand=2 -->
    <int value="0"/> <!-- Max Capture File Size in MB: Def=0 -->
  </parameters>
  <return>
    <variant name="fc_result" scope="event" />
  </return>
</function>
<in>
  <evtSrcFileExt/>
<list>
  <string value="doc"/>
  <string value="docx"/>
  <string value="xls"/>
  <string value="xlsx"/>
  <string value="ppt"/>
  <string value="pptx"/>
  <string value="txt"/>
  <string value="rtf"/>
  <string value="pdf"/>
</list>
</in>
<equal>
  <evtOperationType/>
  <constOpFileCopy/>
```



```
</equal >
</and>
```

Capture the Destination Files When a User Moves a File

This rule captures the source and destination files and file records when a user performs a file copy. After the capture, the Agent applies the MD5 hash to the captured file because Hash Type is set to 104. To enable the rule to capture the destination file, you must select **Execute Rule After Operation** in the Rule Wizard when you create or edit the rule. If you write the rule XML yourself, add the run-PostOp property to the dgrule element in the rule.:

```
<dgrule>runPostOp = "1" </dgrule>
```

The rule syntax:

```
<and>
  <function name="DG_CaptureFile">
    <parameters>
      <currentEvent/>
      <currentAssociatedFileInfo/>
      <currentRuleId/>
      <int value="1"/> <!-- Enable Hashing: Def=0, Enabled=1, Disabled=2 -->
      <int value="104"/> <!-- Hash Type: Def=0, MD5=104, SHA1=103, SHA256=102, SHA512=101 -->
      <int value="4"/> <!-- Disposition: Def=0, Auto=1, Src=2, Dst=3, Both=4 -->
      <int value="1"/> <!-- Capture Alg: Def=0, Synchronous=1, Asynchronous=2 -->
      <int value="2"/> <!-- Capture Mode: Def=0, Record_Only=1, Record_And_File=2 -->
      <int value="0"/> <!-- Transfer Mode: Def=0, Auto_Upload=1, On_Demand=2 -->
      <int value="0"/> <!-- Max Capture File Size in MB: Def=0 -->
    </parameters>
    <return>
      <variant name="fc_result" scope="event" />
    </return>
  </function>
</in>
  <evtSrcFileExt/>
</list>
  <string value="doc"/>
  <string value="docx"/>
  <string value="xls"/>
  <string value="xlsx"/>
  <string value="ppt"/>
  <string value="pptx"/>
  <string value="txt"/>
  <string value="rtf"/>
  <string value="pdf"/>
```

```
</list>
</in>
<equal>
  <evtOperati onType/>
  <constOpFil eMove/>
</equal>
</and>
```

Capture Source Files in Network Upload Events

This rule captures source files and associated records for files involved in network transfer upload events. The Agent applies the SHA-256 hash to the captured files.

The rule syntax:

```
<and>
<function name="DG_CaptureFile">
  <parameters>
    <currentEvent/>
    <currentAssociatedFilesl nfo/>
    <currentRuleId/>
    <int value="1"/> <!-- Enable Hashing: Def=0, Enabl ed=1, Di sabl ed=2 -->
    <int value="102"/> <!-- Hash Type: Def=0, MD5=104, SHA1=103, SHA256=102, SHA512=101 -->
    <int value="2"/> <!-- Di spositi on: Def=0, Auto=1, Src=2, Dst=3, Both=4 -->
    <int value="1"/> <!-- Capture Alg: Def=0, Synchronou s=1, Asynchronou s=2 -->
    <int value="2"/> <!-- Capture Mode: Def=0, Record_Onl y=1, Record_And_Fil e=2 -->
    <int value="2"/> <!-- Transfer Mode: Def=0, Auto_Upl oad=1, On_Demand=2 -->
    <int value="0"/> <!-- Max Capture File Si ze in MB: Def=0 -->
  </parameters>
  <return>
    <vari nt name="fc_resul t" scope="event" />
  </return>
</function>
<in>
<evtSrcFil eExt/>
<list>
  <stri ng val ue="doc"/>
  <stri ng val ue="docx"/>
  <stri ng val ue="xl s"/>
  <stri ng val ue="xl sx"/>
  <stri ng val ue="ppt"/>
  <stri ng val ue="pptx"/>
  <stri ng val ue="txt"/>
  <stri ng val ue="rtf"/>
  <stri ng val ue="pdf"/>
</list>
</in>
```

```

<equal >
  <evtOperationType/>
  <constOpNetTransferUpload/>
</equal >
</and>

```

Upload Captured Files to the Server Automatically

This rule prevents all attempts to write files to removable media such as USB drives. It captures the source files involved in the operation and uploads the files to the server automatically. You must have the Investigation Module enabled to use the Auto_Upload option.

The rule syntax:

```

<and>
<function name="DG_CaptureFile">
  <parameters>
    <currentEvent/>
    <currentAssociatedFileInfo/>
    <currentRuleId/>
    <int value="2"/> <!-- Enable Hashing: Def=0, Enabled=1, Disabled=2 -->
    <int value="0"/> <!-- Hash Type: Def=0, MD5=104, SHA1=103, SHA256=102, SHA512=101 -->
    <int value="2"/> <!-- Disposition: Def=0, Auto=1, Src=2, Dst=3, Both=4 -->
    <int value="1"/> <!-- Capture Alg: Def=0, Synchronous=1, Asynchronous=2 -->
    <int value="2"/> <!-- Capture Mode: Def=0, Record_Only=1, Record_And_File=2 -->
    <int value="1"/> <!-- Transfer Mode: Def=0, Auto_Upload=1, On_Demand=2 -->
    <int value="0"/> <!-- Max Capture File Size in MB: Def=0 -->
  </parameters>
  <return>
    <varint name="fc_result" scope="event" />
  </return>
</function>
<in>
  <evtOperationType/>
  <list>
    <constOpFileWrite/>
    <constOpFileCopy/>
    <constOpFileMove/>
  </list>
</in>
  <equal >
    <evtDestDriveType/>
    <constDriveRemovable/>
  </equal >
</and>

```

Examples of Virtualization and RDP Rules

These rules might help you protect your intellectual property or sensitive information by preventing users from copying data in a variety of configurations. The rules depend on files that have classification tags to identify the sensitive content.

Copying Tagged Files Over Remote Desktop Connection

This rule triggers when a user copies sensitive file to a local disk in a remote or virtual session.

```
<and>
  <and>
    <equal >
      <evtSrcFilePolicyTag />
      <string value="tag_name" />
    </equal >
    <like expr="//tsclient\\">
      <evtDestFilePath />
    </like>
    <equal >
      <agentIsVirtualSession />
      <bool value="true" />
    </equal >
  </and>
  <equal >
    <evtOperationType />
    <constOpFileCopy />
  </equal >
</and>
```

Capture Remote Desktop Actions With Tagged Files

This rule captures remote desktop software operations with tagged files. In this version, the rule monitors file copy events.

```
<and>
  <and>
    <equal >
      <evtSrcFilePolicyTag />
      <string value="tag_name" />
    </equal >
  </and>
```

```

    </equal >
    <equal >
      <agentIsVirtualSession />
      <bool value="true" />
    </equal >
  </and>
  <equal >
    <evtOperationType />
    <constOpFileCopy />
  </equal >
</and>

```

Capture Microsoft TeamViewer Software Accessing Tagged Files

Users who try to open classified files with Microsoft TeamViewer trigger this rule.

```

<and>
  <equal >
    <evtSrcFilePolicyTag />
    <string value="tag_name" />
  </equal >
  <equal >
    <curProcessImageName />
    <string value="teamviewer.exe" />
  </equal >
  <equal >
    <evtOperationType />
    <constOpFileOpen />
  </equal >
</and>

```

Capture Tagged File Access From Virtual Machines

If users try to access classified files from VMWare virtual machines, this rule captures the event.

```

<or>
<and>

```

```
<equal >
  <evtSrcFilePolicyTag />
  <string value="tag_name" />
</equal >
<equal >
  <curProcessImageName />
  <string value="vmware-vmx.exe" />
</equal >
<equal >
  <evtOperationType />
  <constOpFileOpen />
</equal >
</and>

<and>
  <equal >
    <evtSrcFilePolicyTag />
    <string value="tag_name" />
  </equal >
  <equal >
    <curProcessImageName />
    <string value="vmware.exe" />
  </equal >
  <equal >
    <evtOperationType />
    <constOpAdePaste/>
  </equal >
</and>

</or>
```

APPENDIX B Rule Optimization

To achieve the highest possible performance from your Digital Guardian implementation, follow these suggestions when you create your rule definitions. These can help you create rules that run with the lowest possible overhead.

- Minimize your use of regular expressions.
- Place more restrictive rule conditions at the bottom of the rule definition.
- Use Trusted Process rules.

Minimize Use of Regular Expressions

Although they are very flexible, regular expressions require more computing resources on the DG Agent. Best practice suggests the following strategies as alternatives to regular expressions:

- Join multiple rule properties using the <list> operator rather than using a regular expression.
- Use file extension or file type rule properties rather than file path regular expressions.

- When you use regular expressions, make the regular expressions the last rule conditions to evaluate by placing them at the top of the rule.

For more information about regular expressions, refer to [“Regular Expressions” on page 231](#).

Place More Restrictive Rule Conditions at the Bottom of the Rule Definition

Digital Guardian evaluates rules from the bottom to the top. As soon as the DG Agent encounters a rule condition that does not match the current activity, it stops evaluating that rule. By placing the most restrictive rule conditions at the bottom of a rule definition, Digital Guardian can more quickly determine if the current user activity is subject to a particular rule.

For example, if you have a Control rule that prevents users from writing files that have certain extensions to removable drives, use the following model:

```
<and>
<i n>
<evtSrcFileExt/>
<l i s t>
<stri ng val ue=" doc" />
<stri ng val ue=" ppt" />
<stri ng val ue=" pdf" />
</l i s t>
</i n>
<equal >
<evtOperati onType/>
<constOpFileWrite/>
</equal >
<i n>
<evtSrcDriveType/>
<l i s t>
<constDriveUnknown/>
<constDriveRemovable/>
</l i s t>
</i n>
</and>
```


Evaluating from the bottom up, the first condition the rule engine tests against is the drive type. If the drive type in the activity is not unknown or removable, the evaluation stops and the rule does not take effect. In most enterprises, relatively little day-to-day activity involves removable or unknown drives. Placing this condition first makes it easier for Digital Guardian to determine if this rule is relevant to the current activity.

The second condition the rule tests is the type of activity. Because the rule engine has already determined that a removable or unknown drive type is involved, it now evaluates the type of user activity. In this case, the rule governs file write events. If the user is performing a file write, Digital Guardian continues to evaluate the current activity against the rule.

The last rule condition the rule engine evaluates is the file extension. In most enterprises, users are frequently working with Microsoft Word, PowerPoint and Adobe Acrobat® files. If this rule condition appeared first, the DG Agent would likely have to evaluate several more conditions of the rule before determining that the user activity was a rule violation or that the rule did not apply. By placing the least restrictive condition of the rule last in the evaluation order (at the top of the rule), you increased the likelihood that the rule applies to the current user activity.

Although the examples provided in [“Common Use Cases” on page 435](#) use this specific-to-general design, you should consider the activity and needs of your own organization when you write rule definitions.

Use Trusted Process Rules

Trusted process rules identify a process when that process starts. If the process is a trusted process, Digital Guardian recognizes the process and ignores all further activity from that process. The DG Agent does not execute any rules on activity generated by trusted processes.

Trusted process rules execute before filter and control rules. By identifying trusted processes, you can reduce the number of processes that the DG Agent must monitor.

Examples of typical trusted processes include the following:

- Virus scanners
- Host-based firewalls and intrusion detection systems
- Compilers and linkers

APPENDIX C Common Rule Variable Uses

This appendix provides examples of common uses for variables in rules. These use cases demonstrate the syntax of the rule variable language and highlight its capabilities.

You can find examples in the following categories:

- Counting
- Tracking the Passage of Time
- Running a Process

Counting With Rule Variables

The examples in this section deal with counting various items, such as files, while users interact with them.

Prevent User From Copying ZIP Archives to Removable Devices

This rule prevents users from copying more than four ZIP archives to a removable device.

The rule syntax:

```
<and>
  <greaterThan>
    <vari nt name="total Copi esToRemovabl e" scope="gl obal " />
    <i nt val ue="4" />
  </greaterThan>

  <add>
    <vari nt name="total Copi esToRemovabl e" scope="gl obal " />
    <i nt val ue="1" />
  </add>

  <!--wi nzi p-->
  <regExp expr=". zi p">
    <evtSrcFi l ePath/>
  </regExp>

  <equal >
    <evtDestDri veType/>
    <constDri veRemovabl e/>
  </equal >
  <equal >
    <evtOperati onType />
    <constOpFi l eCopy />
  </equal >
</and>
```

Tracking Time

The examples presented in this section deal with user activities involving operations and time. They represent hypothetical Digital Guardian implementations. Use these rules as templates for creating your own rules.

Limit File Copying to 1GB per Day

This set of rules tracks both the amount of files the user copies to removable devices in a day.

Implementing this process requires four rules, six variables and a counting/accumulating process:

- One rule to start the 24 hour time period
- One rule to reset the 24 hour time period
- One rule to track and accumulate the amount data copied to USB
- One rule to present the prompt when the amount copied to USB exceeds the threshold

Time-based rules like these require you to set Continue Rule Evaluation to Yes because the entire rule set needs to be evaluated by the rule engine for each event to ensure the proper time tracking.

You also need to set the rule priorities correctly—set the reset rule and the tracking/accumulating rule to lower priority than the other rules.

The rule syntaxes are:

Start the Timer

This rule starts a clock to time 24 hours. Set Continue Rule Evaluation to Yes and do not send alerts or events.

```
<and>
  <set>
    <varbool  name="timeoutActive"  scope="global"  />
    <bool  value="true"  />
```

```
</set>
<set>
  <vari nt name="num30Mi nuteTi meoutsLeft" scope="gl obal " />
  <i nt val ue="48" />
</set>
<add>
  <vari nt64 name="ti meout" scope="gl obal " />
  <ti me val ue="00: 30: 00" />
</add>
<set>
  <vari nt64 name="ti meout" scope="gl obal " />
  <agentCurrentTi me />
</set>
<equal >
  <varbool name="ti meoutActi ve" scope="gl obal " />
  <bool val ue="fal se" />
</equal >
</and>
```

Reset the Timer

This rule resets the 24 hour clock in the previous rule. Set Continue Rule Evaluation to Yes and do not send alerts or events. Set the priority for this rule to high with low value.

```
<and>
  <set>
    <vari nt name="num30Mi nuteTi meoutsLeft" scope="gl obal " />
    <i nt val ue="48" />
  </set>
  <set>
    <vari nt64 name="Total Copi edToRemovabl e" scope="gl obal " />
    <i nt64 val ue="0" />
  </set>
  <equal >
    <vari nt name="num30Mi nuteTi meoutsLeft" scope="gl obal " />
    <i nt val ue="0" />
  </equal >
  <add>
    <vari nt64 name="ti meout" scope="gl obal " />
    <ti me val ue="00: 30: 00" />
  </add>
</set>
```

```

        <vari nt64 name="ti meout" scope="gl obal " />
        <agentCurrentTi me />
    </set>
    <sub>
        <vari nt name="num30Mi nuteTi meoutsLeft" scope="gl obal " />
        <i nt val ue="1" />
    </sub>
    <l essThan>
        <vari nt64 name="ti meout" scope="gl obal " />
        <agentCurrentTi me />
    </l essThan>
    <greaterThan>
        <vari nt name="num30Mi nuteTi meoutsLeft" scope="gl obal " />
        <i nt val ue="0" />
    </greaterThan>
    <equal >
        <varbool name="ti meoutActi ve" scope="gl obal " />
        <bool val ue="true" />
    </equal >
</and>

```

Calculate the Amount of Data Transferred to USB

This rule calculates the quantity of data transferred to USB. Set Continue Rule Evaluation to Yes and do not send alerts or events. Set the priority for this rule to high with low value.

```

<and>
    <add>
        <vari nt64 name="Total Copi edToRemovabl e" scope="gl obal " />
        <evtSrcFi l eSi ze />
    </add>
    <equal >
        <evtDestDri veType />
        <constDri veRemovabl e/>
    </equal >
    <equal >
        <evtOperati onType />
        <constOpFi l eCopy />
    </equal >
</and>

```

Prompt When the Amount Transferred in 24 Hours Exceeds the 1GB Threshold

This rule calculates when the quantity of data transferred to USB exceeds the 1GB threshold. Every 24 hours the Reset rule clears the variable TotalCopiedToRemovable.

```
<and>
  <greaterThan>
    <variant64 name="Total CopiedToRemovable" scope="global" />
    <int64 value="1073741824" />
  </greaterThan>
  <equal >
    <evtDestDriveType />
    <constDriveRemovable/>
  </equal >
  <equal >
    <evtOperationType />
    <constOpFileCopy />
  </equal >
</and>
```


Running Processes

The rule examples in this section use rule variables to run or monitor processes.

Transfer Events Immediately From Agents

This example runs DGet to make the Agent send the event that triggered the rule to the Server immediately. Without this rule, the Agent sends the event the next time it contacts the Server.

To make this example work, the triggered rule needs to exit and let the event get written into a bundle. Making that happen requires two rules. Rule 1 tracks the event (copies of documents to removable drives in this example), and then sets a global Boolean rule variable Var-RunDGETNow at the top of the rule to true. Then the rule exits.

Rule 2 fires only when the global Boolean variable Var-RunDGETNow is set to true by rule 1. It runs the process dget.exe to execute DGet, resets Var-RunDGETNow to false and exits.

Rule 1 — Sets the variable Var-RunDGETNow to true when the user moves a file to a removable drive.

```
<and>
<set>
<varbool name="Var-RunDGETNow" scope="global" />
<bool value="true" />
</set>
<in>
<evtOperationType/>
<list>
<constOpFileWrite/>
<constOpFileCopy/>
<constOpFileMove/>
</list>
</in>
<equal>
<evtDestDriveType/>
<constDriveRemovable/>
</equal>
</and>
```

Rule 2 — Runs DGet based on the state of Var-RunDGETNow. Rule 2 has lower priority than rule 1 because it has no operation associated with it, so it runs after rule 1 completes. This allows you to chain multiple rules.

```
<and>
<set>
<varbool name="Var-RunDGETNow" scope="gl obal " />
<bool val ue="fal se"/>
</set>
<process name="dget.exe" path="C:\DG Tool s\" cmdLi ne="" />
<equal >
<varbool name="Var-RunDGETNow" scope="gl obal " />
<bool val ue="true"/>
</equal >
</and>
```

Capture Executable File (.exe) Exit Codes

You can capture the exit codes for executable files into a rule variable using a rule that captures process launches. Note the custom rule data assignment for the exit code.

```
CustomRuleData = 501
```

```
<and>
<set>
    <varstring name = "CustomRuleData" scope = "event"/>
    <varint name = "Var-Temp1" scope = "event"/>
</set>
<process name = "cmd.exe" path = "c:\wi ndows\system32\" cmd
Li ne = "/c EXIT 501" wai tForTerminati on = "true" />
<return>
    <varint name = "Var-Temp1" scope = "event"/>
</return>
</process>
<equal >
    <curProcessI mageName/>
    <stri ng val ue = "cal c. exe" />
</equal >
<equal >
    <evt0perati onType/>
    <constOpAppStart/>
</equal >
```

</and>

Index

A

Action Types 21
activeVaultRuleNames 126
Adding
 MD5 Hash to a Rule Definition 234
Address Properties 110
agentAdapterCount 72
agentCurrentDayOfWeek 72
agentCurrentTime 73
agentFileVersion 73
agentGatewayMac 73
agentIPAddress 73
agentIsLaptop 73
agentIsRegistered 74
agentIsVirtualSession 74
agentLastServerComm 74
And Operator 218
anyProcessVaulted 126
Application Data Exchange Properties 113
Application Data Exchange Rules

Examples 491

Application Data Exchange Types 190
Audience for This Book 2

B

bluetoothAddress 183
Books in Documentation Set 3
bool 183

C

CD/DVD Rules
 Examples 486
Classification Properties 119
Classification Rules 4
 Examples 436, 490
clipboardVaultRuleName 126

Component List 371, 389
Component Rules 7, 309, 371, 389
Component rules 309, 371, 389, 419
const1394 198
constAccessPoint 216
constAdHoc 216
constAES 216
constant 184
Constants 189
constBluetooth 198
constBus1394 191
constBusATA 191
constBusATAPI 191
constBusFibre 191
constBusiSCSI 191
constBusRAID 191
constBusSAS 191
constBusSATA 191
constBusSCSI 191
constBusSD 191
constBusSSA 191
constBusUnknown 191
constBusUSB 192
constCutPaste 190
constDocRepositoryWebtop 210
constDragDrop 190
constDriveCDRom 194
constDriveFixed 194
constDriveRamDisk 194
constDriveRemote 194
constDriveUnknown 194
constEthernet 198
constFriday 193
constIrDA 198
constMailBCC 195
constMailCC 195
constMailTo 195
constMonday 193
constNone 216
constOpAdeCut 199
constOpADEInsertFile 199
constOpADEInsertNewObject 199
constOpAdePaste 199
constOpAdePrintProcess 199
constOpAdePrintScreen 199
constOpAdeScreenCapture 200
constOpAppScreenBuffer 200
constOpAppStart 201
constOpCDBurn 202
constOpen 215
constOpFileArchive 203
constOpFileCopy 203
constOpFileCreate 203
constOpFileDecrypt 204
constOpFileDelete 204
constOpFileMove 204
constOpFileOpen 204
constOpFileRead 204
constOpFileRecycle 204
constOpFileRename 204
constOpFileRestore 204
constOpFileSaveAs 204
constOpFileWrite 205
constOpMailAttach 208
constOpNetTransferDownload 208
constOpNetTransferUpload 208
constOpNetwork 209
constOpNetworkEx 209
constOpOperation 209
constOpPrint 209
constOpSendMail 209
constPrintScreen 190
constProtocolBluetooth 211
constProtocolDNS 211

- constProtocolHTTP 211
- constProtocolIrDA 211
- constProtocolTCP 211
- constProtocolUDP 211
- constSaturday 193
- constShared 215
- constSunday 193
- constThursday 193
- constTKIP 216
- constTuesday 193
- constVPN 198
- constWednesday 193
- constWEP 216
- constWPA 215
- constWPA2 215
- constWPA2PSK 215
- constWPANone 215
- constWPAPSK 215
- Control Rules 4
- Copying 38
 - Rules 38
- Create 310
- Creating 20, 424
 - Data Vault Rules 424
 - Rules 20
 - Trigger Rules 423
- curProcessAddressBar 110
- curProcessCmdLine 75
- curProcessCompanyName 75
- curProcessFileDescription 76
- curProcessFilePath 76
- curProcessFileVersion 76
- curProcessImageName 77
- curProcessLegalCopyright 77
- curProcessMD5ContentHash 77
- curProcessNoPrompting 78
- curProcessOriginalFileName 78

- curProcessProductName 79
- curProcessProductVersion 79
- curProcessSaveAsActive 127
- curProcessTopUrlIPAddress 79
- curProcessURLList 112
- curProcessVaulted 127
- curProcessVaultRuleName 127
- curProcessWindowTitle 80
- Current Processes 69
- Custom Document Properties 148

D

- Data Vault 419
- Data Vault Properties 126
- Data Vault Rules 7, 419
 - Creating 424
 - Examples 426
- dateTime 183
- Days of the Week 193
- Default Document Properties 146
- define rule variables 261
- Device Properties 129
- Digital 309
- Digital Guardian
 - Rules 9
- dnsHostAvailable 80
- Document Properties
 - Custom 148
 - Microsoft Office 146
 - Windows Explorer 145
- Document Properties Rule Properties 142
- Drive Types 194
- driverRunning 81

E

- Email Address Formats 187
 - SMTP 187
 - X.400 187
- Email Recipient Types 195
- Email Rule Properties 152
- Email Rules
 - Examples 498
- environmentVariable 176
- Equal Relational Operator 223
- evtADEType 113
- evtBufferEntityFrequency 119
- evtBufferPolicyTag 120
- evtCurrentRuleAction 82
- evtCurrentUserId 83
- evtDestBusType 83
- evtDestCustomID 129
- evtDestDocPropertyDate 142
- evtDestDocPropertyInt 142
- evtDestDocPropertyString 143
- evtDestDriveType 83
- evtDestFileCanContentInspect 121
- evtDestFileEntityFrequency 121
- evtDestFileExt 84
- evtDestFileIsClassified 122
- evtDestFileIsEncrypted 85
- evtDestFileModifiedTime 85
- evtDestFilePath 85
- evtDestFilePolicyTag 122
- evtDestFileSize 85
- evtDestIsWindowsCdBurnFolder 87
- evtDestProduct 130
- evtDestProductID 130
- evtDestSerialNumber 131
- evtDestVendor 132
- evtDestVendorID 132
- evtDomain 160, 164
- evtIsAfterOperation 90
- evtIsDomainUser 90
- evtIsLocalAdmin 91
- evtIsOutboundConn 160, 165
- evtIsOverwrite 91
- evtIsPrivate 165
- evtLocalPort 160, 165
- evtMailAttachmentSize 153
- evtMailRecipients 153
- evtMailRecipientTypes 153
- evtMailSubject 154
- evtMailTotalSize 154
- evtOperationType 92
- evtParentOperationType 92
- evtPhysicalMedium 166
- evtPrinterName 174
- evtProtocolType 160, 166
- evtRemoteAddress 161, 166
- evtRemotePort 161, 167
- evtSrcBusType 93
- evtSrcCustomID 132
- evtSrcDocPropertyDate 143
- evtSrcDocPropertyInt 144
- evtSrcDocPropertyString 144
- evtSrcDriveMatchesDest 93
- evtSrcDriveType 94
- evtSrcFileCanContentInspect 123
- evtSrcFileEntityFrequency 123
- evtSrcFileExt 94, 115, 161
- evtSrcFileIsClassified 124
- evtSrcFileIsEncrypted 94
- evtSrcFileModifiedTime 95
- evtSrcFilePath 95, 115, 161
- evtSrcFilePolicyTag 124
- evtSrcFileSize 96
- evtSrcIsWindowsCdBurnFolder 98

evtSrcProcessCompanyName 116
evtSrcProcessFileDescription 116
evtSrcProcessFilePath 116
evtSrcProcessFileVersion 117
evtSrcProcessImageName 117
evtSrcProcessLegalCopyright 117
evtSrcProcessMD5ContentHash 118
evtSrcProcessProductName 118
evtSrcProcessProductVersion 118
evtSrcProcessWindowTitle 118
evtSrcProduct 133
evtSrcProductID 133
evtSrcSerialNumber 134
evtSrcVendor 135
evtSrcVendorID 135
evtUrlPath 162, 168
evtWirelessAuthenticationMode 170
evtWirelessEncryption 171
evtWirelessInfrastructureMode 172
Exporting
 Rules 40

F

File Save As 422
File System Rules
 Examples 446
Filter Rules 5, 17
 Examples 503

G

Global Properties 72
Greater Than Relational Operator 223

I

Importing
 Rules 41
In Relational Operator 223
int 184
IP Mask Relational Operator 224
ipAddress 183
irdaAddress 184

L

Less Than Relational Operator 224
Like Relational Operator 225
Logical Operators 217
 And 218
 Nested 219
 Not 218
 Or 218

M

macAddress 185
Manuals, List of 3
Match Attribute 229
 All 230
 Any 230
MD5 Hash 233
 Generating 234
 Using 234
md5Hash 185
Message Recipients
 Rule Evaluation 152

N

Nested Logical Operators 219
Network Adapter Types 198
Network Operation Properties 163
Network Rules
 Examples 470
Not Operator 218

O

Office Document Properties 146
OLE Insertion Properties 174
Operation Types 199
Optimization
 Rules 519
Or Operator 218

P

Parent Processes 69
parentProcessAddressBar 111
parentProcessCmdLine 102
parentProcessCompanyName 103
parentProcessFileDescription 103
parentProcessFilePath 103
parentProcessFileVersion 104
parentProcessImageName 104
parentProcessLegalCopyright 105
parentProcessMD5ContentHash 105
parentProcessProductName 105
parentProcessProductVersion 105
parentProcessTopUrlIpAddress 106
parentProcessURLList 112
parentProcessVaulted 127

parentProcessVaultRuleName 128
parentProcessWindowTitle 107
prevProcessAddressBar 111
prevProcessVaulted 128
prevProcessVaultRuleName 128
Print Properties 174
Print Rules
 Examples 480
Properties 67
 Application Data Exchange 113
 Classification 119
 Data Vault 126
 Device 129
 Document Properties 142
 Email Operations 152
 Global 72
 Network Operations 163
 OLE Insertion 174
 Print 174
 URL/Address 110
 Value Types 183
 Windows Registry 178
Protocol Types 211

R

registryInt 178
registryString 178
Regular Expressions 231
Relational Operators 221
 Equal 223
 Greater Than 223
 In 223
 IP Mask 224
 Less Than 224
 Like 225

- Rule 18
- Rule Definitions 61
- Rule Optimization 519
- Rule Syntax 64
- rule variable
 - about 261
 - arrays 272
 - classification rule use case examples 261
 - control rule use case examples 277
 - names 266
 - scope 267
 - syntax 261
- Rules 20, 38, 39
 - About 9
 - Activating 39
 - Copy 38
 - Copying 38
 - Create 20
 - Creating 20
 - Edit 36

S

- Save As 422
- serverAvailable 108
- Severity Levels 18
 - Alerts 18
 - Rules 18
- similarProcessRunning 108
- similarProcessVaulted 128
- SMTP Format 187
- Source Processes 70
- srcProcessCmdLine 118
- string 185
- Symbolic Constants 189

T

- time 186
- Trigger Rules 419
 - Creating 423
- Trusted Process Rules 6, 17, 521
 - Examples 508

U

- URL List Properties 111
- URL/Address Properties 110

V

- Value Types 183
 - Bluetooth Address 183
 - Bool 183
 - Constant 184
 - dateTime 183
 - Int 184
 - IP Address 183
 - IrDA Address 184
 - MAC Address 185
 - MD5 Hash 185
 - String 185
 - Time 186
 - Valid Relational Operators 186

W

- Who Should Read This Book 2
- Window Explorer Document Properties 145
- Windows Registry Properties 178

Wireless Authentication Types 215
Wireless Encryption Types 216
Wireless Infrastructure Mode Types 216

X

X.400 Format 187