IBM® Security

How BigFix Helps Investigate a Threat in Forensic Activities

IBM
| AUTHOR: Cristina Bonanni |
| CONTRIBUTORS: Bernardo Pastorelli, Bradford Fisher, Rosario Gangemi |
| REVIEWERS: Alex Donatelli, Daniel P Paquette, James Murphy, John Golembiewski, Miles Hamill, Rich Caponigro, Wei Lee |
## Contents

**Detecting malware and remediating with BigFix** ..... 1  
  Detecting exposures using the Relevance language ..... 1  
  An example of computer attack ..... 2  
  Creating relevance expressions ..... 2  
  Checking a computer status ..... 3  
  Checking the enterprise ..... 4  
  Periodically checking computer compliance ..... 7  
  Checking and remediating the environment ..... 9  
Detecting exposures by mapping IOC ..... 11  

Mapping an IOC document to a Fixlet ..... 11  
BigFix and Yara tool ..... 14  
Yara Rules ..... 15  
Using Yara from BigFix ..... 15  
Some useful relevance examples for malware detection checks ..... 15  
Mapping between OpenIOC terms and Relevance language ..... 17  
Using WMI Queries ..... 18

© Copyright IBM Corp. 2015
Detecting malware and remediating with BigFix

Organizations are focusing their attention on how to defend themselves against cybersecurity threats by detecting potentially malicious applications when they run on their computers. The objective of incident handling is to manage and control faults and disruptions to IT services in a timely manner. In this battle, BigFix is a fundamental tool to identify malware on computers by scanning them. With BigFix and the applications that run on top of it, security teams can query and analyze information for a potentially compromised computer. They can detect missing patches and install them, uninstall or update software containing vulnerabilities, manage the running and well-being of security tools such as virus scanners and firewalls that are misconfigured, disabled, or running with out-of-date signature data.

BigFix is a reliable and fast mechanism for detecting security incidents by being:
• Timely: it manages the current situation, not how it was a few minutes ago.
• Scalable: it deploys to all the company computing assets.
• Efficient: it minimally impacts the network and computing resources.
• Role-based: it grants permissions to perform particular activities on particular computers.
• Content driven: it shares known and emerging insights efficiently and with flexibility to address situations never before encountered.
• Secure: it transfers instructions and results by using secure, authenticated, and compressed data streams.
• Actionable on the endpoint: the endpoint runs authenticated instructions.
• Lightweight: it minimally impacts the user while these activities are taking place.

Using BigFix, you can monitor large networks of remote computers, detect security exposures, incorrect configurations, and other vulnerabilities, and resolve issues in real time.

As with any security situation, shortening the time interval from detection to protection is key to surviving an attempted attack. Some of the most common activities that attackers perform on systems include password-dumping, reconnaissance, and data theft.

In this document, you can find information on how to detect malware and remediate by using BigFix.

Detecting exposures using the Relevance language

Using the Relevance language, you can collect a large set of data from a computer, such as files, services, processes, device drivers, operating system configuration, and network configuration. The BigFix agent already provides thousands of key phrases (known as inspectors) of the Relevance language. As the name implies, these phrases are used to inspect the properties of the clients on which the BigFix agent is installed. The values returned by the inspectors can also be used for calculations and comparisons.
You can inspect remote client computers and summarize their properties by running a BigFix analysis. The analysis retrieves the various computer properties, analyzes and reports them to central servers.

You can also run a Fixlet to take corrective actions. Use Fixlet Debugger, a BigFix tool, to write and debug Relevance language expressions that are then evaluated on every computer where analyses or Fixlets are run.

You can use Fixlets and analyses to detect malware and support forensic company activities.

The [Bigfix.me](https://bigfix.me) community includes many examples of custom content, including single relevance expressions, analyses, and Fixlets.

In this community all users can share custom content such as single relevance expressions, analyses, and Fixlets. You can also participate in the discussion about BigFix topics and issues by accessing the [BigFix forum](https://community.bigfix.com).

### An example of computer attack

With the BigFix Relevance language, you can implement and perform many checks to detect a computer attack or security exposure.

An example of an attack that might compromise a computer is the combination of the following actions:
- Create a registry key whose value contains an IP address, such as 192.168.100.*
- Start a process that listens on port 1234.
- Install a file in the Windows system folder that has a well-defined md5 hash and a name that can be represented by the regular expression ^hack.*

The following topics provide step-by-step instructions about how you can write and run the Relevance expressions to detect the three attack actions.

### Creating relevance expressions

A security operator can create relevance expressions to verify any of the three attack actions described in "An example of computer attack" as follows:

- To check the existence of a specific registry, use the following expression:
  
  ```relevance
  exists key "HKEY_LOCAL_MACHINE\SOFTWARE\Hack\HackedValue" whose
  (value of it as string = regex "^192\.168\.100\.\d{1,3}" ) of registry
  ```

- To check if a listening mode socket exists and that the listening port is 1234, use the following relevance:
  
  ```relevance
  exists sockets whose (listening of tcp state of it and local port of it = 1234) of network
  ```

- To check the existence of a file and its hash, use the following relevance expression:
  
  ```relevance
  exists descendants whose ( name of it as lowercase = regex "^hack.*" and md5 of it
  as lowercase = "54a47f66e09a77e61649109c6a08866" as lowercase) of system folder
  ```

The relevance expression counts the files (descendants) in the Windows system folder whose name matches your chosen regular expression and whose md5 matches the hacked value.

To find other useful Relevance expressions, see "Some useful relevance examples for malware detection checks" on page 15.
Checking a computer status

To run the previously defined relevance expressions on a single computer, use Fixlet Debugger.

When you enter the relevance expressions in the (qna) tab of the Fixlet Debugger tool, ensure that each one is preceded by the prefix Q: to identify it as a question. You can also use the single clause tab, which does not require you to prefix your expression with Q:.

Press Ctrl-A or select Edit > Question/Answer from the menu to get an answer to the questions entered in the Fixlet Debugger tool.

The results are shown in the following screen capture:
The answer to the second question reveals that the registry key was matched and this means that the computer was compromised.

**Checking the enterprise**

Using BigFix you can check and continuously monitor all the computers of an enterprise by defining an analysis or a Fixlet. More specifically, you can run an analysis to get true or false answers and to collect data.

As a first step, you can define an analysis that collects true or false answers from all the endpoints in your enterprise. An analysis can collect multiple properties. For each of the three relevance expressions, you can define a property.

The following screen capture shows a custom analysis created with these properties related to the three attack actions:

- HackedFile
- HackedRegistry
- HackedNetwork
The resulting analysis is displayed as follows in the BigFix console:

The analysis evaluates the property on the endpoint periodically and, if any of the properties change, the new property values are uploaded to the server.
You can monitor the results from the console by looking at the collected data:

In the Results table, you can assess the environment and check for exposures. The columns in the Results table can be ordered to search for computers with specific results. You can see that just one computer seems to be affected by an issue.

In addition, to get a true or false answer, you can use an analysis to collect information from a computer. For example, assume that you want to change the analysis to collect the IP addresses that is defined in the registry key, if the registry key exists. This information might be helpful to get more details about the hack, regardless of whether or not the computer is compromised.

You can change the second relevance example by using the following expression:

values of keys "HKEY_LOCAL_MACHINE\SOFTWARE\Hack\HackedValue" whose (value of it as string = regex "^[192]\.[168]\.[100].\d{1,3}" ) of registry

The result shows that the HackedRegistry property has no value for computers that are not affected by the hack, but instead contains the IP address, specified in the registry key, for computers that are affected. In this way BigFix is collecting additional information and not just a simple Boolean answer.
Periodically checking computer compliance

After an analysis is configured, an administrator can set up BigFix to deliver a periodic report about the status of that analysis on the computers in the environment. It means that a daily report about the status of the environment is sent directly to an email inbox, without you having to check the BigFix console every day.

You can use BigFix Web Reports to create reports that filter computers by specific properties.

The following screen capture shows a configuration example in which you can select all the computers for which at least one of the properties is true:

![Web Reports screenshot](image)

This report contains the basic computer properties, in addition to the three new properties that are defined by the analysis. It includes a filter on the three properties that matches computers for which any of the properties is true.

After you create the report, save it with the name HackReport.

To receive a daily email with the outcome of the report, create a scheduled activity by selecting the Administration entry in the top navigation bar. The following page opens:
Click **Create Scheduled Activity** to create a new scheduled activity and define it to generate and email the report daily:

Enter the email address (administrator@example.com) to which you want to email the report and save the activity. This setting save the administrator from having to frequently check the BigFix console.

The report content might display as follows:
Checking and remediating the environment

Administrators can use BigFix Fixlets to check the environment and remediate suspicious behavior. A Fixlet contains a set of relevance conditions that assess the status of the target system, as well as an action that can be run to remediate the target system.

Considering the attack scenario described in “An example of computer attack” on page 2, you can define a Fixlet that has a relevance condition that is met when any of the three conditions, described in “Creating relevance expressions” on page 2, is met.

The following screen capture shows you how to create the relevance section of a Fixlet:
The Fixlet also has an action that you can use to clean up any computer that is affected by the hack. For example, consider an action that deletes the registry key, as shown in the following screen capture:

```
// clear previously used files
1 delete _appendfile
2 delete Temp.reg
3 // create .reg file
4 appendfile REGEDIT4
5 appendfile -HKEY_LOCAL_MACHINE\SOFTWARE\Hack\HackValue
6 move _appendfile temp.reg
7 // run .reg file
8 regedit -s temp.reg
```

The Fixlet is shown to be applicable to all computers that have been hacked; in this scenario, one computer.
When you click Take Action, you activate the action on all the computers where the Fixlet is relevant. The action cleans up the registry and remediates the exposure.

Detecting exposures by mapping IOC

Forensics identifies checks that detect exposures through Indicators Of Compromise (IOC).

BigFix uses the Relevance language to map these IOC checks.

One of the most common ways to code Indicators Of Compromise is the OpenIOC format, which defines the checks in xml documents.

The following topic shows how to convert an OpenIOC document into a Fixlet.

Mapping an IOC document to a Fixlet

For example, see the document unusual files in recycler, retrieved from the http://www.iocbucket.com repository.

This document checks a typical attack technique that uploads files to C:\$RECYCLE.BIN\ and then runs them.

This is the content of the IOC example:

```xml
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     id="8ada2a7f-b5f4-43b6-b6cc-5d255cedc814"
     last-modified="2013-12-23T20:19:25"
     xmlns="http://schemas.mandiant.com/2010/ioc">

    <short_description>Unusual Files in Recycler</short_description>
    <description>This IOC will detect unusual files executing out of the C:\Recycler\ folder typically indicative of malicious activity.</description>
    <authored_by>@iocbucket</authored_by>
    <authored_date>2013-12-23T20:15:13</authored_date>
    <definition>
        <Indicator operator="OR" id="b9482ca1-930f-4113-8c65-b9c35d0c2333">
            <Indicator operator="AND" id="df0e5af2-d911-4a12-ab2c-a9f174eb9d03">
                <Indicator operator="OR" id="a2f92566-445c-479e-8401-8c3a6c87e7fa">
                    <IndicatorItem id="f54f769b-e4b1-4047-b57d-2499128c5c10" condition="is">
                        <Context document="FileItem" search="FileItem/FilePath" type="mir" />
                        <Content type="string">Recycler</Content>
                    </IndicatorItem>
                    <IndicatorItem id="178be4f2-1d22-4b5a-bed7-dba72cf93766" condition="is">
                        <Context document="FileItem" search="FileItem/FilePath" type="mir" />
                        <Content type="string">Recycle.bin</Content>
                    </IndicatorItem>
                </Indicator>
            </Indicator>
        </Indicator>
    </definition>
</ioc>
```
This IOC document can be translated into a Relevance language expression. The first step of the translation process is to analyze the operators in the IOC document. The example performs the OR of two conditions, one on files, the other one on the registry content.

The first condition searches all the files whose path (FileItem/FilePath) contains either the word recycler or the word recycle.bin. The check on the paths is ANDed with a set of four checks on the file extension (FileItem/FileExtension). This condition thus matches all the files on disk that have:
1. The path containing either recycler or recycle.bin
2. The file extension is txt, dll, exe, or rar

This check can be encoded in the Relevance language, but a straightforward translation would be very resource-intensive and should be avoided. Instead of searching all the files on the disk, the relevance expression can be optimized to search for the recycle bin in the root folder of the disk, and then search for files inside the recycle bin. This approach greatly reduces the number of files analyzed.

The following example shows the resulting relevance expression:
exists descendants whose (name of it as lowercase = regex "(.*txt|.*dll|.*exe|.*rar)") of folders whose (name of it as lowercase contains "recycler") or name of it as lowercase contains "recycle.bin") of folder "C:"
It maps the following section of the IOC document:

```
<Indicator operator="AND" id="5537bbaa-3d93-4579-9615-51828b964974">
  <IndicatorItem id="4e38893a-6d64-417c-a2a3-e1b1396579e4" condition="contains">
    <Context document="RegistryItem" search="RegistryItem/Path" type="mir"/>
    <Content type="string">TypedPaths</Content>
  </IndicatorItem>
  <Indicator operator="OR" id="af61d88f-9b1f-4895-b7f7-9ba3fba348fd">
    <IndicatorItem id="5b6c7800-871c-4805-b323-ed631d547f8" condition="contains">
      <Context document="RegistryItem" search="RegistryItem/Text" type="mir"/>
      <Content type="string">recycler</Content>
    </IndicatorItem>
    <IndicatorItem id="754aab92-35f4-4d5d-803f-2296ce5cc405" condition="contains">
      <Context document="RegistryItem" search="RegistryItem/Text" type="mir"/>
      <Content type="string">recycle.bin</Content>
    </IndicatorItem>
  </Indicator>
</Indicator>
```

The second condition in the IOC document checks if the recycle bin path was entered manually in Windows Explorer. The condition analyzes the entire registry, searching for a key that contains the string TypedPaths and for a value that contains either the string recycler; or recycle.bin.

Again, searching the entire registry is an expensive operation, so try to optimize the check operations when you translate to the Relevance language. In this case the TypedPath registry key is always located at:

```
HKEY_USERS\user_SID\Software\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths
```

The following example shows the resulting relevance expression:

```
exists values whose (it as string as lowercase contains "recycler" or it as string as lowercase contains "recycle.bin") of keys "Software\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths" of keys of key "HKEY_USERS" of registry
```

It maps the following section of the IOC document:

```
<Indicator operator="AND" id="5537bbaa-3d93-4579-9615-51828b964974">
  <IndicatorItem id="4e38893a-6d64-417c-a2a3-e1b1396579e4" condition="contains">
    <Context document="RegistryItem" search="RegistryItem/Path" type="mir"/>
    <Content type="string">TypedPaths</Content>
  </IndicatorItem>
  <Indicator operator="OR" id="af61d88f-9b1f-4895-b7f7-9ba3fba348fd">
    <IndicatorItem id="5b6c7800-871c-4805-b323-ed631d547f8" condition="contains">
      <Context document="RegistryItem" search="RegistryItem/Text" type="mir"/>
      <Content type="string">recycler</Content>
    </IndicatorItem>
    <IndicatorItem id="754aab92-35f4-4d5d-803f-2296ce5cc405" condition="contains">
      <Context document="RegistryItem" search="RegistryItem/Text" type="mir"/>
      <Content type="string">recycle.bin</Content>
    </IndicatorItem>
  </Indicator>
</Indicator>
```

Detecting malware and remediating with BigFix
Concatenating these two Relevance expressions by using the OR operator generates an optimized Relevance expression corresponding to the IOC document. This expression can be run inside mFixlet Debugger for validation and then used in an analysis or Fixlet, as described in the scenario “Detecting exposures using the Relevance language” on page 1.

The following screen capture shows that the execution of the optimized relevance requires a very short amount of time and can therefore be run periodically by an analysis or Fixlet without affecting the overall performance of the computer on which the checks are run.

**Important:** The Fixlets obtained by translating OpenIOC documents can be scheduled to run periodically to continuously monitor known threats.

**BigFix and Yara tool**

In addition to mapping IOC checks, you can also map Yara rules by using BigFix.

Yara is an open source malware identification tool that uses rules based on text or binary patterns to look for malware signatures in files. It also scans compressed files using yextend.

This tool runs on Windows, Linux, and Mac OS X and is used to help malware researchers identify and classify malware, but it is also useful for real-time forensic analysis.
Anti-malware detection tools find potential threats on your computer systems. They work from rule files downloaded from the vendors website.

While these vendors quickly release updates to their rules for new malware definitions, there are times when you need to look for threats (such as zero day vulnerabilities) as soon as possible before these vendors have released updates to the definition files.

### Yara Rules

The website [http://yararules.com](http://yararules.com) provides a centralized location on GitHub and includes content that is published by the forensic community.

Rules are divided into five categories: AntiDebug, Crypto, Malicious Document, Packer, and Malware

The website [https://www.iocbucket.com/](https://www.iocbucket.com/) has 11 IOC documents in Yara format.

### Using Yara from BigFix

BigFix does not currently provide content to install Yara and run Yara rules, but it is relatively simple to produce BigFix tasks for your environment that install Yara on target endpoints and run Yara rules. These tasks could indicate success if no infection was found on the target endpoints, or you could code them to generate a simple report of infected computers and send it to the BigFix server for reporting through the BES Upload Manager.

If you have experienced situations where you are concerned about your endpoints being vulnerable to a new malware before anti-malware vendors have released an updated set of rules, then using a tool like Yara and running it from BigFix could be just what you need to ensure the security of your IT deployment.

### Some useful relevance examples for malware detection checks

These are some examples of what you can query on your systems by using the Relevance language:

Check if a file exists:

```relevance
exists file "svchost.exe" of system folder
```

Calculate the md5 hash of a file:

```relevance
md5 of file "svchost.exe" of system folder
```

Calculate the sha1 hash of a file:

```relevance
sha1 of file "svchost.exe" of system folder
```

Check if a file has a specific sha1 hash:

```relevance
exists file "svchost.exe" whose ( sha1 of it as lowercase = "4af001b3c3816b86060cf2de2c0fd3c1dfb4878" as lowercase) of system folder
```

Look for a file with a specific sha1 and name in the Windows system folder and its subfolders (this relevance might take long time):

```relevance
exists descendant whose( name of it as lowercase = "svchost.exe" and sha1 of it as lowercase = "4af001b3c3816b86060cf2de2c0fd3c1dfb4878" as lowercase) of system folder
```
Check if a registry key exists:
exists key
"HKLM\Software\Microsoft\Windows\CurrentVersion\Run\DaemonDisk" of registry

Check if a registry key has a specific value:
exists key
"HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\BigFix\EnterpriseClient\Settings\Client\_GatherService_HTTPServer_HostName" whose (value of it = "127.0.0.1") of registry

Check if a file and a registry key exist:
exists descendant whose (name of it as lowercase = "svchost.exe" and shal of it as lowercase = "4af001b3c3816b960660cf2de2c0fd3c1dfb4878" as lowercase) of system folder OR exists key "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\BigFix\EnterpriseClient\Settings\Client\_GatherService_HTTPServer_HostName" whose (value of it = "127.0.0.1") of registry

Check if a file with a specific hash exists and its name satisfies a regular expression:
exists descendants whose (name of it as lowercase = regex "^svc.*" and shal of it as lowercase = "4af001b3c3816b960660cf2de2c0fd3c1dfb4878" as lowercase) of system folder

List all the files contained in the Windows system folder, whose name matches a regular expression:
matches (case insensitive regex (".*kernel.*") of names of descendants of system folder

Check if a registry key exists with a value satisfying a regular expression:
exists key "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\BigFix\EnterpriseClient\Settings\Client\_GatherService_HTTPServer_HostName" whose (value of it as string = regex "127.0.*") of registry

Use a regular expression to find the letter of the Windows disks defined by using a WMI query:
(parenthesized part 1 of (matches (regex "^Win32_LogicalDisk\.DeviceID=\%22(.*)\%22") of string values of it)) of properties whose(string value of it contains "DeviceID=") of select objects ("DeviceID from Win32_LogicalDisk") of WMI

List the wireless connections:
(ssids of it, secured of it, encryption of it) of wifi of adapters of network

List the storage devices whose name contains usb:
friendly names whose (it as lowercase contains "usb") of active devices

List the Windows autostart locations:
(values of keys "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" of it; values of keys "HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run" of it; values of keys "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" of it) of registry

Lists the files opened in Windows Explorer more recently:
files of folder ("C:\Users\" & name of current user & "\AppData\Roaming\Microsoft\Windows\Recent")

Checks if Internet Explorer stores the passwords:
exists values of key "HKCU\Software\Microsoft\Internet Explorer\IntelliForms\Storage2" of registry
### Mapping between OpenIOC terms and Relevance language

The following table describes the mapping between the most commonly used OpenIOC terms and the Relevance language.

**Table 1. Mapping between IOC terms and Relevance language**

<table>
<thead>
<tr>
<th>OpenIOC term</th>
<th>Relevance language</th>
</tr>
</thead>
<tbody>
<tr>
<td>F(shared by multiple parts)</td>
<td>name of &lt;file&gt;</td>
</tr>
<tr>
<td>F</td>
<td>md5 of &lt;file&gt;</td>
</tr>
<tr>
<td>F</td>
<td>parent folder of &lt;file&gt;</td>
</tr>
<tr>
<td>F</td>
<td>size of &lt;file&gt;</td>
</tr>
<tr>
<td>R</td>
<td>key &lt;string&gt; of registry</td>
</tr>
<tr>
<td>R</td>
<td>parent key of &lt;key&gt;</td>
</tr>
<tr>
<td>R</td>
<td>(it as string) of values of key &quot;&lt;key path&gt;&quot; of registry</td>
</tr>
<tr>
<td>P</td>
<td>remote address of &lt;socket&gt;</td>
</tr>
<tr>
<td>F</td>
<td>pathname of &lt;file&gt;</td>
</tr>
<tr>
<td>F</td>
<td>(following text of last &quot;,&quot; of name of it) of file &quot;C:\Windows\System32\fc.exe&quot;</td>
</tr>
<tr>
<td>P</td>
<td>name of &lt;process&gt;</td>
</tr>
<tr>
<td>R</td>
<td>(it as string) of values of key &quot;&lt;key path&gt;&quot; of registry</td>
</tr>
<tr>
<td>S</td>
<td>(service name of it) of services</td>
</tr>
<tr>
<td>P</td>
<td>remote port of &lt;socket&gt;</td>
</tr>
<tr>
<td>P</td>
<td>local port of &lt;socket&gt;</td>
</tr>
<tr>
<td>F</td>
<td>sha1 of &lt;file&gt;</td>
</tr>
<tr>
<td>P</td>
<td>tcp of &lt;socket&gt;</td>
</tr>
<tr>
<td>F</td>
<td>values &quot;ProductName&quot; of version blocks of file &quot;C:\Windows\System32\fc.exe&quot;</td>
</tr>
<tr>
<td>S</td>
<td>display name of services</td>
</tr>
<tr>
<td>P</td>
<td>image file of &lt;process&gt;</td>
</tr>
<tr>
<td>S</td>
<td>friendly name of services</td>
</tr>
<tr>
<td>F</td>
<td>language of version blocks of file &quot;C:\Windows\System32\fc.exe&quot;</td>
</tr>
<tr>
<td>E</td>
<td>(event id of it as string</td>
</tr>
<tr>
<td>S</td>
<td>image paths of services</td>
</tr>
<tr>
<td>R</td>
<td>(type of it as string) of values of key &quot;&lt;key path&gt;&quot; of registry</td>
</tr>
<tr>
<td>D</td>
<td>service key value name of active devices</td>
</tr>
<tr>
<td>E</td>
<td>source of &lt;event log record&gt; (</td>
</tr>
<tr>
<td>D</td>
<td>friendly name of it</td>
</tr>
<tr>
<td>U</td>
<td>names of logged on users</td>
</tr>
<tr>
<td>F</td>
<td>sha256 of &lt;file&gt;</td>
</tr>
<tr>
<td>E</td>
<td>event type of &lt;event log record&gt;</td>
</tr>
<tr>
<td>P</td>
<td>tcp state of &lt;socket&gt;</td>
</tr>
<tr>
<td>E</td>
<td>description of &lt;event log record&gt;</td>
</tr>
</tbody>
</table>
Table 1. Mapping between IOC terms and Relevance language (continued)

<table>
<thead>
<tr>
<th>OpenIOC term</th>
<th>Relevance language</th>
</tr>
</thead>
<tbody>
<tr>
<td>VolumeItem/DriveLetter</td>
<td>root folders of drives</td>
</tr>
<tr>
<td>FileItem/Created</td>
<td>creation time of file &quot;C:\Windows\System32\fc.exe&quot;</td>
</tr>
<tr>
<td>ProcessItem/PortList/PortItem/remoteIP</td>
<td>((remote address of it) as string</td>
</tr>
<tr>
<td>TaskItem/Name</td>
<td>names of scheduled tasks</td>
</tr>
<tr>
<td>DriverItem/DeviceItem/AttachedToDeviceName</td>
<td>names of active device files</td>
</tr>
<tr>
<td>PortItem/localIP</td>
<td>local address of &lt;socket&gt;</td>
</tr>
<tr>
<td>PortItem/process</td>
<td>process of &lt;socket&gt;</td>
</tr>
<tr>
<td>EventLogItem/writeTime</td>
<td>(time written of it) of records of security event log</td>
</tr>
<tr>
<td>ProcessItem/Username</td>
<td>user sid of &lt;event log record&gt;</td>
</tr>
<tr>
<td>ProcessItem/parentpid</td>
<td>ppid of &lt;process&gt;</td>
</tr>
<tr>
<td>EventLogItem/machine</td>
<td>computer of &lt;event log record&gt;</td>
</tr>
<tr>
<td>EventLogItem/log</td>
<td>source of &lt;event log record&gt;</td>
</tr>
<tr>
<td>EventLogItem/genTime</td>
<td>(time generated of it) of records of security event log</td>
</tr>
<tr>
<td>EventLogItem/categoryNum</td>
<td>(category of it) of records of security event log</td>
</tr>
<tr>
<td>ServiceItem/status</td>
<td>(running of it) of services</td>
</tr>
<tr>
<td>TaskItem/TriggerList/Trigger/TriggerFrequency</td>
<td>trigger of &lt;task definition&gt;</td>
</tr>
<tr>
<td>ProcessItem/PortList/PortItem/localPort</td>
<td>((local port of it) as string</td>
</tr>
<tr>
<td>VolumeItem/FileSystemName</td>
<td>name of &lt;drive&gt;</td>
</tr>
<tr>
<td>VolumeItem/Type</td>
<td>file system type of &lt;drive&gt;</td>
</tr>
</tbody>
</table>

Using WMI Queries

You can use the WMI queries supported by the Relevance language to map additional IOC terms for which a matching Relevance term does not exist.

This table describes the mapping between IOC terms and WMI queries.

Table 2. Mapping between IOC terms and WMI queries

<table>
<thead>
<tr>
<th>IOC term</th>
<th>WMI query</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiskItem/DiskName</td>
<td>diskdrive get name, diskdrive where(name=&quot;&quot;)</td>
<td></td>
</tr>
<tr>
<td>EventLogItem/category</td>
<td>ntevent get category</td>
<td></td>
</tr>
<tr>
<td>ProcessItem/arguments</td>
<td>process where(name=&quot;&quot;) get commandline</td>
<td></td>
</tr>
<tr>
<td>UserItem/disabled</td>
<td>useraccount where(disabled=&quot;true&quot;)</td>
<td></td>
</tr>
<tr>
<td>UserItem/fullname</td>
<td>useraccount get caption, useraccount where(caption=&quot;&quot;)</td>
<td></td>
</tr>
<tr>
<td>VolumeItem/BytesPerSector</td>
<td>diskdrive get BytesPerSector</td>
<td></td>
</tr>
<tr>
<td>VolumeItem/CreationTime</td>
<td>volume where(deviceID,caption..) get installdate</td>
<td></td>
</tr>
</tbody>
</table>
Table 2. Mapping between IOC terms and WMI queries (continued)

<table>
<thead>
<tr>
<th>IOC term</th>
<th>WMI query</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volumeltem/SerialNumber</td>
<td>volume get serialnumber volume</td>
<td></td>
</tr>
<tr>
<td>Volumeltem/VolumeName</td>
<td>get DeviceID datafile where(name=&quot;fpath&quot;) get Manufacturer</td>
<td></td>
</tr>
<tr>
<td>FileItem/PEInfo/VersionInfoList/VersionInfoItem/CompanyName</td>
<td>datafile where(name=&quot;fpath&quot;) get FileName</td>
<td></td>
</tr>
<tr>
<td>FileItem/PEInfo/VersionInfoList/VersionInfoItem/InternalName</td>
<td>datafile where(name=&quot;fpath&quot;) get FileName,Extension</td>
<td>FileName.Extension</td>
</tr>
<tr>
<td>FileItem/PEInfo/VersionInfoList/VersionInfoItem/OriginalFilename</td>
<td>datafile where(name=&quot;fpath&quot;) get Version</td>
<td></td>
</tr>
<tr>
<td>FileItem/PEInfo/VersionInfoList/VersionInfoItem/ProductVersion</td>
<td>datafile where(name=&quot;fpath&quot;) get FileName,Extension</td>
<td>FileName.Extension</td>
</tr>
<tr>
<td>FileItem/PEInfo/VersionInfoList/VersionInfoItem/OriginalFilename</td>
<td>datafile where(name=&quot;fpath&quot;) get Version</td>
<td></td>
</tr>
<tr>
<td>FileItem/PEInfo/VersionInfoList/VersionInfoItem/ProductVersion</td>
<td>datafile where(name=&quot;fpath&quot;) get FileName,Extension</td>
<td>FileName.Extension</td>
</tr>
<tr>
<td>FileItem/PEInfo/VersionInfoList/VersionInfoItem/OriginalFilename</td>
<td>datafile where(name=&quot;fpath&quot;) get Version</td>
<td></td>
</tr>
<tr>
<td>Partition/PartitionLength</td>
<td>partition get size</td>
<td></td>
</tr>
<tr>
<td>ProcessItem/description</td>
<td>service where(name=&quot;&quot;) get PathName</td>
<td></td>
</tr>
<tr>
<td>PrefetchItem/FullPath</td>
<td>datafile where(path=&quot;\windows\Prefetch&quot; and filename=&quot;&quot;) get name</td>
<td></td>
</tr>
<tr>
<td>PrefetchItem/Created</td>
<td>datafile where(path=&quot;\windows\Prefetch&quot; and filename=&quot;&quot;) get creationdate</td>
<td></td>
</tr>
<tr>
<td>PrefetchItem/SizeInBytes</td>
<td>datafile where(path=&quot;\windows\Prefetch&quot; and filename=&quot;&quot;) get filesize</td>
<td></td>
</tr>
<tr>
<td>EventLogItem/genTime</td>
<td>ntevent get TimeGenerated</td>
<td></td>
</tr>
<tr>
<td>EventLogItem/writeTime</td>
<td>ntevent get TimeWritten</td>
<td></td>
</tr>
<tr>
<td>ServiceItem/description</td>
<td>service where(name=&quot;&quot;) get description</td>
<td></td>
</tr>
<tr>
<td>ProcessItem/description</td>
<td>process where(name=&quot;&quot;)</td>
<td></td>
</tr>
</tbody>
</table>