# Contents

## Introduction

Today's cyber defenders can choose from among many cyber-threat information sources and information-retrieval mechanisms. However, information retrieval alone is not helpful; defenders must be able to use the information. Once cyber defenders have obtained the information, funneling it into analysis and protection tools can require significant effort.

The Last Quarter Mile Toolset (LQMToolset), a modular toolset to address the problem of information usability. The toolset consists of a controller application, parsers, and tool chains. The controller application routes incoming cyber threat information through the appropriate parsers and tool chains to reformat the data as needed and feed it into endpoint tools. Both the input (parsers) and output (tool chains) are modular, allowing the toolset to be easily extended as new data formats and endpoint tools are encountered.

Figure 1 shows an example data flow through the LQMToolset. In this configuration, the Cyber Fed Model (CFM) client provides threat information in a configured directory. When the tool is run, the controller traverses the directory tree and uses the configured parsers to parse the data into an intermediate data format. Then each alert is passed into configured tool chains (one for each device, analysis tool, etc.), which process the alert and send it to the appropriate device.

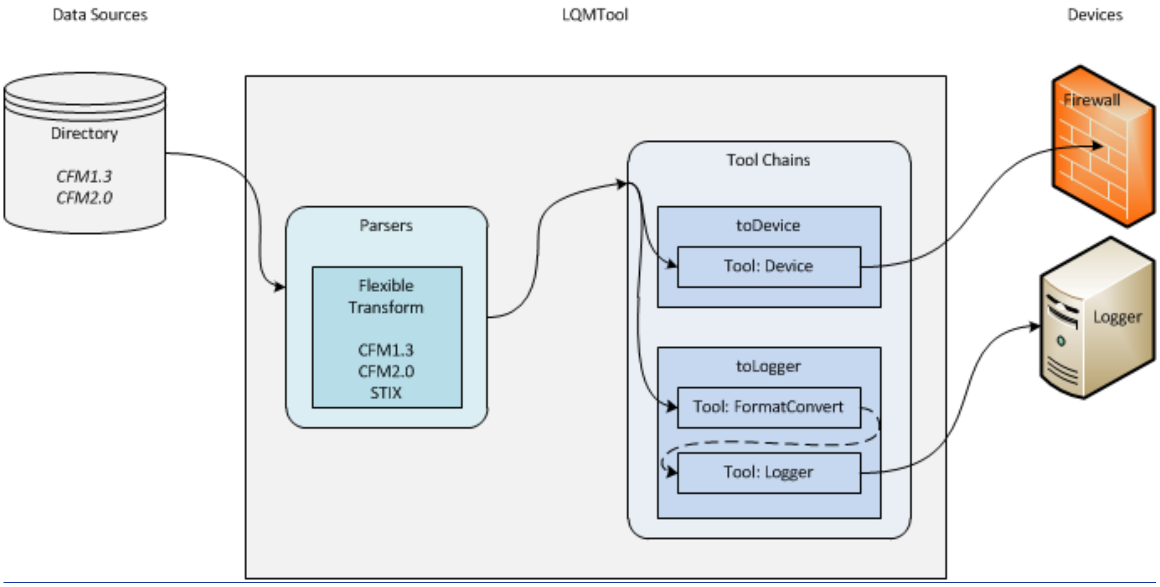

Figure 1 - LQMToolset Data Flow

# Installation

## Prerequisites

Python 3.2

Python packages

It is recommended that if any of the following packages are available via a linux package manger (yum,zipper, etc) for your distribution, those should be used.

| **Package name** | **Pypi installation command** |
|---|---|
| python-dateutil | `pip3 install python-dateutil` |
| lxml | `STATIC_DEPS=true pip3 install lxml` |
| pytz | `pip3 install pytz` |
| Dumper | `pip3 install Dumper` |
| rdflib | `pip3 install rdflib` |
| netaddr | `pip3 install netaddr` |
| pan | `pip3 install pan-python` |
| toml | `pip3 install toml` |

## Downloading

To obtain the installation package, go to https://github.com/anl-cyberscience/LQMToolset and select the release desired.

## Install

To install the LQM Toolset, simply extract the LQMTool.tgz file into the desired directory.

## Executing

After configuring for your installation, to execute the LQMToolset, run the following command from the installation directory:

```
python3 LQMTool.py <user-level-config-file>
```

If desired, create a cron job to run the tools periodically.

# Configuration

The configuration is defined in two configuration files, a system-level configuration and user-level configuration. The system level configuration file, config.toml, contains the definitions for the available parsers and tools. This file does not usually need to be modified by the end-user. The user-level configuration file or files contain the local configuration options including data source definitions, tool instances, and tool chains.

All configuration files are TOML format.  A complete description of the format can be found at https://github.com/toml-lang/toml.

TOML format is designed to be mapped to a muti-level table format.  The data in TOML files are case-sensitive.  They are broken up into sections with the section names in square brackets.  Within sections are key-value pairs.  Values can be strings (surrounded by double-quote characters), integer, floating point, date/time stamps, and arrays of any valid type.

Example configuration files are shown in Appendix A: Sample configuration files.

## Data source configuration

The data sources are specified in the user configuration file.

The only currently supported data source is the directory-based source.  This source specifies root directories and what to do with the files once they have been processed.  This source will traverse the entire directory structure rooted at each entry in the "dirs" key of the Source.Directory table looking for file pairs of the format FILENAME and .FILENAME, where the .FILENAME file contains the metadata for the FILENAME file.  This is the structure of the CFM client downloads.  If you wish to use the LQMToolset with a data source other than the CFM3 client's download directory, please see Appendix C: Metadata files for information on creating the .FILENAME metadata files.

### *Table: [[Source.Directory]]*

| **Key** | **Value** |
|---|---|
| dirs | Array of root-level directories to search for files to process |
| post_process | What to do with the files after they have been processed. Allowable values are:<br>delete – deletes the file<br>move – moves the file to a directory named "processing in the directory being processed<br>track – keeps track of the files in a file containing a list of filenames |

## Tool and Tool Chain configuration

Tool chains are built from tool instances, which are created from the tools available in the installation.  Then chaining these instances creates the tool chains.

To configure a new device, an entry must be added to the local configuration file. There can be many tools of a specific type – one for each physical device, but they must have distinct names.

## Tool Chains

Tool chains are named “chains” of tool instances and are defined in the user-level configuration file. Since there can be multiple tool chains, they are defined as arrays (using the ‘[[‘ ‘]]’ notation)

### *ToolChains*

| **Key** | **Value** |
|---|---|
| name | A unique name identifying this tool chain. |
| chain | A list of tools to chain together to perform processing on each record |
| active | Whether or not this tool chain is currently active. This parameter is not required ad defaults to true. Valid values are true and false and are case sensitive |

## Current Tools

### *Tool: PaloAlto*

| | |
|---|---|
| Purpose | Place or remove blocks on Palo Alto Networks firewall devices |
| Input | Intermediate format |
| Output | None |

| **Key** | **Value** |
|---|---|
| name | A unique name identifying this device. |
| api_key | The api key retrieved from the Palo Alto device for remote access. |
| api_username | The username that has API access/privileges to the Palo Alto device |
| api_password | The user’s password<br>Either the api_key or api_username/api_password must be specified |
| hostname | The Palo Alto device’s hostname or IP |
| badIPFiles | List of dynamic block list files to use. Each file can hold 300 less than the maximum number of IP addresses the Palo Alto device supports. |
| block_lists | The named block lists configured on the Palo Alto device |
| db_location | The directory that will hold the local database of blocked IP |

addresses

cafile — The location of your CA certificate file for the Palo Alto device

prune_method — The method used to prune IP addresses from the database if there are more IP addresses than the Palo Alto supports. IP addresses are pruned after removing all expired IP blocks.
The valid values are:
- Expiration – the blocks with the earliest expiration time are removed
- Added – the blocks with the earliest add time are removed
- Detected – the blocks with the earliest detection time are removed

default_duration — The default time a block should be in place for if the duration is not specified in the alert

unprocessed_file — A file that will hold all unprocessed blocks. This file will be a CSV file and will have the creation timestamp (YYMMDD-HHMMSS) embedded in the filename before the extension, or at the end if no extension is specified For example, if the filename is dir/file.txt, the file created, if necessary, would be dir/file.20150401-113524.txt.

actions_to_process — Specify the list of actions this tool can/will process.
Valid values:
Block, Revoke, Notify, Watch, SendReport, OtherAction, All
All is a shortcut that allows specification of all action types

### *Device Setup & Configuration*

LQMT uses Palo Alto’s Dynamic Block Lists (also called External Block Lists or EBLs) to block IPs. There is a limit to the number of IP addresses that can be blocked by a Palo Alto device. See your devices documentation for these limits. Each Palo Alto device can have up to 10 block lists and each block list is limited to 300 less than the device’s limit. The Alto accesses the EBLs via http requests to a web server. Configuring one is beyond the scope of this document.

Requirements:
- Web server to host the Dynamic Block List files
  - only needs to be accessible from the LQMT computer
  - The LQMT computer also needs to have write access to the location it reads the EBLs from as it will write the files to that location
- The LQMT machine needs to trust the root CA for the Palo Alto web server
  - One will need to be created if one hasn’t already been imported
    - The default certificate created by the device will not work
  - To create one,
    - log in to the web UI as an administrator
    - select the Device tab

    - select Certificates under Certificate management on the menu
    - click on Generate at the bottom of the window
    - Fill in the fields as appropriate
      - Common name should be the machine name
      - Ensure Certificate Authority is checked
      - Add any Certificate Attributes you may want
      - Click Generate
    - Then click on the newly generated certificate and check the following:
      - Forward Trust Certificate
      - Forward Untrust Certificate
      - Certificate for Secure Web GUI
  - After creating the certificate, you will need to export it
    - click the export button at the bottom of the window and accept the defaults
    - store it in a location accessible to LQMT
    - specify the location in the cafile configuration parameter for the specific Palo Alto device
- Create the block list objects
  - Log into the Web GUI as an administrator
  - Select the Objects tab
  - Select the Dynamic Block Lists item from the menu
    - Click the Add button at the bottom of the window
    - Enter a name
      - This name will be in the block_lists configuration parameter
    - Optionally enter a description
    - Enter the source location of the file backing this block list
      - The physical location of the file will be added to the badIPFiles configuration parameter
    - Set the repeat to Monthly at 00:00
      - This is sufficient because LQMT will perform a refresh as necessary and doesn't need to rely on a regularly scheduled refresh
- Create the Policies that use the block lists to restrict network traffic
  - Due to the unique nature of each installation, this is beyond the scope of this document
  - Some useful links related to using EBLs
    - https://live.paloaltonetworks.com/docs/DOC-4724
    - https://live.paloaltonetworks.com/docs/DOC-5850

### *Limitations*

The Palo Alto module currently only blocks IP addresses and network ranges (CIDR). Any other blocks that are not supported will be output to the file specified in the unhandled_blocks configuration parameter

### Tool: Checkpoint

| | |
|---|---|
| Purpose | Place or remove blocks on Checkpoint devices.<br>The connection to Checkpoint devices is done through a shared SSL key which needs to be generated on the machine LQMTools is installed on and copied to the Checkpoint device |
| Input | Intermediate format |
| Output | None |

| **Key** | **Value** |
|---|---|
| name | A unique name identifying this device. |
| hostname | The Checkpoint device's hostname or IP |
| port | The port to connect to via ssh |
| username | The username to use to log into the Checkpoint device |
| originator | The originator that will be stored with the blocks put into the Checkpoint device |
| default_duration | The default time a block should be in place for if the duration is not specified in the alert |
| unprocessed_file | A file that will hold all unprocessed blocks. This file will be a CSV file and will have the creation timestamp (YYMMDD-HHMMSS) embedded in the filename before the extension, or at the end if no extension is specified For example, if the filename is dir/file.txt, the file created, if necessary, would be dir/file.20150401-113524.txt. |
| actions_to_process | Specify the list of actions this tool can/will process.<br>Valid values:<br>Block, Revoke, Notify, Watch, SendReport, OtherAction, All<br>All is a shortcut that allows specification of all action types |

### ***Device Setup & Configuration***

The LQMTool plugin for checkpoint devices uses the checkpoint firewall's Suspicious Activities Monitoring Protocol (samp) to block IP addresses via the command line interface and ssh from the LQMT machine. The following outlines the steps necessary to configure the device for use with LQMT. The following assumes the computer that is running the LQMT software is named lqmt.domain.com, the checkpoint computer is named cp.domain.com.

- Use the web UI to create a new user with an adminRole, set the shell to /bin/bash, and ensure that command line access is granted. For purposes of this document, the user name `cfm` will be used

- Set up password-less ssh access for the cfm user just created. The exact steps will depend on the machine running LQMT, but the basics:
    - Create an RSA key pair if not already done. This can be done by runnig the command: `ssh-keygen –t rsa` on lqmt.domain.com and accept all defaults. For these purposes, it is best to not have a passphrase for this key pair.
    - Copy the public key to the admin user's account on the checkpoint device. Depending on the open-ssl version on the LQMT computer, you could use the command: `ssh` –copy-id cfm@cp.domain.com. If that command isn't available, you can run the following on lqmt.domain.com as the user that will be running LQMT:
        - `cat ~/.ssh/id_rsa.pub | ssh user@123.45.56.78 "mkdir -p ~/.ssh && cat >>  ~/.ssh/authorized_keys"`
    - Verify password-less access by trying the ssh to the checkpoint machine:
        - `ssh cfm@cp.domain.com`
        - You should be able to login without using a password.
- In addition, the checkpoint device needs to be configured as a firewall in order for the LQMT to be effective. This is beyond the scope of this document

### *Limitations*

The Checkpoint module currently only blocks IP addresses and network ranges (CIDR). Any other blocks that are not supported will be output to the file specified in the unhandled_blocks configuration parameter

## *Tool: ArcSight*

| | |
|---|---|
| Purpose | Place or remove blocks on Checkpoint devices |
| Input | CEF format |
| Output | None |

| **Key** | **Value** |
|---|---|
| name | A unique name identifying this device. |
| host | the ArcSight hostname or IP |
| port | The port that ArcSight is listening on |
| protocol | The protocol to be used (tcp or udp) |
| actions_to_process | Specify the list of actions this tool can/will process.<br>Valid values:<br>Block, Revoke, Notify, Watch, SendReport, OtherAction, All<br>All is a shortcut that allows specification of all action types |

### *Device Setup & Configuration*

To configure the ArcSight Logger to receive data from LQMT, a new receiver needs to be configured. To do that, use the web UI to log in to the ArcSight Logger and navigate to Configuration -> Receivers. Then click the Add button, enter a name for the receiver and select the CEF TCP Receiver and click next. Modify any parameters necessary for your site. Do not change the source type and, note the port number (change it if you need to). That is the port number that needs to be entered into the configuration file. By default, newly added receivers aren't enabled. Enable the receiver by clicking the box on the far right of the receiver list for the new receiver just added.

### *Tool: CEF*

| | |
|---|---|
| Purpose | Converts data from intermediate format to CEF format |
| Input | Intermediate format |
| Output | CEF format |

| **Key** | **Value** |
|---|---|
| name | A unique name identifying this device. |

## Logging

LQMTool allows for basic logging using the parameters specified below.

[Logging]

| **Key** | **Value** |
|---|---|
| LogFileBase | The base logging file name and can include a path. This is used as the base for log file creation. The following log files are created:<br>{LogFileBase}.err.log – logs error conditions<br>{LogFileBase}.info.log – Logs general information<br>{LogFileBase}.debug.log – logs debug information only if Debug is true. |
| Debug | Whether or not debug logging is enabled. Allowable values are true or false and are case-sensitive. This parameter is optional and defaults to false. |

## Whitelists

LQMTool allows for whitelisting indicators and is specified in the user-level configuration file. A whitelist file is specified in the configuration and every time the LQMTool script is run, the file is checked for modification and the internal database is updated.

[Whitelist]

| **Key** | **Value** |
|---|---|
| whitelist | the file containing the whitelisted indicators |

| | |
|---|---|
| dbfile | The location of the SQLite database that holds the whitelist information |

### Whitelist File Format

The whitelist file format is a plain text file with section headers (identified by the delimiters ‘[‘ and ‘]’) and entries in each section. The allowed sections are described below.

| **Section Name** | **Description** |
|---|---|
| [IPv4Address] | List of whitelisted IPv4 address, one per line |
| [IPv4Subnet] | List of whitelisted IPv4 subnet specifications, one per line |
| [IPv6Address] | List of whitelisted IPv6 address, one per line |
| [IPv6Subnet] | List of whitelisted IPv6 subnet specifications, one per line |
| [Domain] | List of whitelisted domains, one per line |
| [Host] | List of whitelisted hosts, one per line |
| [URL] | List of whitelisted URLs, one per line |

Example whitelist file:

```
[IPv4Address]
192.168.1.1

[IPv4Subnet]
192.168.1/24
192.168.2.0/24
192.168.3.0/24

[IPv6Address]
ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

[IPv6Subnet]
ffff:ffff:ffff:ffff/90

[Domain]
example.com
example2.com

[Host]
badguy.example.com

[URL]
http://www.somesite.com/blah
```

## Appendix A: Sample configuration files

### The system-level configuration file: config.toml

The config.toml file is the system-level configuration and will not likely need to be modified. It contains information to tell the controller which parsers and tools are available and where they can be found. It also contains the tools available for use.

```
#parsers
[parsers]
  path = "lqm"
  additional_paths = [ "tpl/pyxb","tpl/xmltodict","ft" ]

#the CFM 1.3 format parser uses Flexible Transform
[parsers.cfm13]
  module = "parsers.FlexTransform"
  parser_class = "FlexTransformParser"
  format = "Cfm13Alert"

#the Flexible Transform configuration for CFM 1.3
[parsers.cfm13.configs]
  Cfm13Alert = "resources/sampleConfigurations/cfm13.cfg"
  LQMTools = "resources/sampleConfigurations/lqmtools.cfg"

#the CFM 2.0 format parser uses Flexible Transform
[parsers.cfm20]
  module = "parsers.FlexTransform"
  parser_class = "FlexTransformParser"
  format = "Cfm20Alert"

#the Flexible Transform configuration for CFM 2.0
[parsers.cfm20.configs]
  Cfm20Alert = "resources/sampleConfigurations/cfm20alert.cfg"
  LQMTools = "resources/sampleConfigurations/lqmtools.cfg"

#global tool config
[tools]
  path = "tools"

#The toPaloAlto tool specification
[tools.PaloAlto]
  module = "to_paloalto"
  tool_class = "ToPaloAlto"
  config_class = "PaloAltoConfig"
  additional_paths = [ "tpl/pan" ]

#The toCheckpoint tool specification
[tools.Checkpoint]
  module = "to_checkpoint"
  tool_class = "ToCheckpoint"
  config_class = "CheckpointConfig"

#The toCEF tool specification
[tools.CEF]
  module = "to_cef"
  tool_class = "ToCEF"
  config_class = "CEFConfig"

#The toArcSight tool specification
[tools.ArcSight]
  module = "to_arcsight"
  tool_class = "ToArcSight"
  config_class = "ArcSightConfig"
```

## Example User-level Configuration File

```
#includes each of the files specified in the array
includes = [
  "devices.toml"
]

#specify the whitelist file to use
[Whitelist]
  whitelist = "testdata/whitelist.txt"
  dbfile = "testdata/whitelist.db"

#main program config
[[Source.Directory]]
  dirs = [ "/Data/CFM/Alert" ]
# post_process indicates what to do with the file after processing
# Allowable values:
#  delete - deletes the file
#  move - moves the file to a directory named "processing" in the
#         directory being processed
#  track - keeps track of the files processed in a file containing a
#          list of filenames
 post_process = "move"

#tool instances specification

# create an instance of the CEF tool
[[Tools.CEF]]
  name = "cef-mapping"

#create an instance of the toArcSight tool
[[Tools.ArcSight]]
  name = "arcsight-1"
  host = "arcsight1.yourdomain.com"
  port = 598
  protocol = "tcp"
  actions_to_process = [ “All” ]

#create another instance of the toArcSight tool
[[Tools.ArcSight]]
  name = "arcsight-2"
  host = "arcsight2.yourdomain.com"
  port = 598
  protocol = "tcp"

#Chain the cef-mapping and toArcSight tool instances to process alerts and
send them to the specific arcsight machine defined previously
[[ToolChains]]
  name = "arcsight-1"
  chain = ["cef-mapping","arcsight-1"]

#Chain the cef-mapping and the second toArcSight tool instances to process
alerts and send them to the specific arcsight machine defined previously
[[ToolChains]]
  name = "arcsight-2"
  chain = ["cef-mapping","arcsight-2"]

#create an instance of the toPaloAlto tool
[[Tools.PaloAlto]]
  name = "my-pa200"
  api_key = "API_KEY_OBTAINED_FROM_DEVICE"
  #api_username = "username"
  #api_password = "password"
  hostname = "paloalto.yourdomain.com
  badIPFiles = [ "BlockedIPs-01.txt", "BlockedIPs-02.txt" ]
  block_lists = [ "CFM_EBL-01", "CFM-EBL-02" ]
  db_location = "paloalto"
  cafile = "pa-ames/pa.crt"
  prune_method = "Expiration"
  default_duration = 259200
  unprocessed_file = "unprocessed/pa-ames.csv"

 [[ToolChains]]
  active = true
  name = "ames-pa200"
  chain = ["ames-pa200"]
```

## Appendix B: Intermediate Data Format

The intermediate data format is a subset/simplification of the complete threat data that can be directly used by a variety of systems.

| **Field Name** | **Description** |
|---|---|
| dataItemID | UUID for data item |
| fileID | UUID for file |
| detectedTime | UTC epoch of time in the alert itself (if present, filled with reportedTime if not) |
| reportedTime | UTC epoch of time reported to CFM (or other system) |
| processedTime | UTC epoch of time processed locally on client (i.e. when THIS parsed record was created) |
| indicator | The value to be acted upon (e.g. ip, domain name, URL) |
| indicatorType | A type name that informs how to interpret the indicator (e.g. ipv4, emailAddress) (enum) |
| indicatorDirection | enum { source, destination } |
| secondaryIndicator | A secondary indicator that restricts (logical AND) the indicator (e.g. a port number) (when appropriate) |
| secondaryIndicatorType | A type name that informs how to interpret the secondaryIndicator (e.g. tcpport, udpport) (enum) |
| secondaryIndicatorDirection | enum { source, destination } |
| directSource | The CFM site abbr (or other applicable identifier) that uploaded the data |
| secondarySource | String representing where the CFM site got it from (when appropriate) |
| action1 | An action to be performed when the indicator is seen (semi-enum) |
| duration1 | How long the action is to be performed |
| action2 | An action to be performed when the indicator is seen (semi-enum) |
| duration2 | How long the action is to be performed |
| reason1 | First reason for this alert |

| | |
|---|---|
| reference1 | Reference for info for reason |
| reason2 | Additional reason for this alert |
| reference2 | Reference for info for reason |
| majorTags | A string containing a comma separated list of tags. These are high-level concepts that tend to remain "tag worthy" over time (e.g. Ransomware) |
| minorTags | A string containing a comma separated list of tags. These are highly detailed and/or only important for a short time (e.g. Cryptolocker) |
| restriction | TLP level |
| sensitivity | OUO marking |
| reconAllowed | Boolean: true (default) = recon is ok, false = recon is not ok |
| priors | number of previous (known) reports |
| confidence | Confidence in accuracy: 0 = none, 100 = full |
| severity | How significant is the impact (assuming relevant): 0 = no impact, 100 = catastrophic |
| relevancy | How relevant is this alert: 0 = not applicable, 100 = perfectly targeted |
| relatedID | UUID of a related indicator |
| relationType | enum { supersedes, extends, superseded by, extended by } |
| comment | A string that represents a comment that was included with the alert. |
| fileHasMore | boolean: false (default) = data item was fully translated, true = More details on the data item are present in file) |

## Appendix C: Metadata files

The metadata file contains a JSON object with key-value pairs described below.  All keys are required unless noted.

| **Key** | **Description** |
|---|---|
| DataSensitivity | OUO marking.  Valid values are “ouo” and “noSensitivity” |
| FileName | File name this metadata is attached to |
| PayloadFormat | Format of the data. Valid values are “STIX”, “Cfm13Alert”,”Cfm20Alert” |
| PayloadType | CFM Payload type marking.  Valid value: “Alert” |
| ReconPolicy | Whether or not additional recon is allowed on the indicator.  Valid values:<br>Touch – recon allowed<br>NoTouch – recon not allowed |
| SendingSite | Site name that submitted the report |

| | |
|---|---|
| SentTimestamp | The timestamp when the file was uploaded |
| UploadID | The UUID for the uploaded document |
| DownloadElementExtendedAttribute | Extended information (not required). This is a JSON object that contains |
| Field | Valid values:<br>origFilename<br>orig1.3Filename<br>comment |
| Value | The value of the field specified in the Field attribute |

An example metadata file:

```
{
    "DataSensitivity": "noSensitivity",
    "DownloadElementExtendedAttribute": {
        "Field": "orig1.3Filename",
        "Value": "Alerts.Prod.201207230519.xml.gpg"
    },
    "FileName": "YourAlertFilename",
    "PayloadFormat": "Cfm13Alert",
    "PayloadType": "Alert",
    "ReconPolicy": "Touch",
    "SendingSite": "SomeSite",
    "SentTimestamp": "1409490943",
    "UploadID": "012ABC89-ABCD-EF01-2345-ADAD73739104"
}
```

The JSON schema that describes the metadata file:

```
{
       "DocumentMetaData": {
              "fields": {
                     "DataSensitivity": {
                            "description": "OUO marking",
                            "datatype": "enum",
                            "required": true,
                            "defaultValue": "noSensitivity",
                            "ontologyMappingType": "enum",
                            "enumValues": {
                                   "ouo": {
                                          "ontologyMapping":
"http://www.anl.gov/cfm/transform.owl#OUOSemanticConcept"
                                   },
                                   "noSensitivity": {
                                          "ontologyMapping":
"http://www.anl.gov/cfm/transform.owl#NotOUOSemanticConcept"
                                   }
                            }
                     },
                     "FileName": {
                            "description": "File name the metadata is attached to",
                            "datatype": "string",
                            "required": true,
                            "ontologyMappingType": "simple",
                            "ontologyMapping": ""
                     },
                     "PayloadFormat": {
                            "description": "Schema format of the data",
                            "datatype": "enum",
                            "required": true,
                            "ontologyMappingType": "enum",
                                "enumValues": {
                                        "STIX": {
                                                "ontologyMapping": ""
                                        },
                                        "Cfm13Alert": {
                                                "ontologyMapping": ""
                                        },
                                        "Cfm20Alert": {
                                                "ontologyMapping": ""
                                        }
                                }
                        },
                        "PayloadType": {
                                "description": "CFM Payload type marking",
                                "datatype": "enum",
                                "required": true,
                                "defaultValue": "Alert",
                                "ontologyMappingType": "enum",
                                "enumValues": {
                                        "Alert": {
                                                "ontologyMapping": ""
                                        }
                                }
                        },
                        "ReconPolicy": {
                                "description": "Is additional recon allowed on the inidicator",
                                "datatype": "enum",
                                "required": true,
                                "defaultValue": "Touch",
                                "ontologyMappingType": "enum",
                                "enumValues": {
                                        "Touch": {
                                                "ontologyMapping":
"http://www.anl.gov/cfm/transform.owl#ReconAllowedSemanticConcept"
                                        },
                                        "NoTouch": {
                                                "ontologyMapping":
"http://www.anl.gov/cfm/transform.owl#ReconNotAllowedSemanticConceptActionDescriptionSemanticConcept
"

                                        }
                                }
                        },
                        "SendingSite": {
                                "description": "Site name that submitted the report",
                                "datatype": "string",
                                "required": true,
                                "ontologyMappingType": "simple",
                                "ontologyMapping":
"http://www.anl.gov/cfm/transform.owl#SiteAbbreviationSemanticConcept"
                        },
                        "SentTimestamp": {
                                "description": "The timestamp when the file was uploaded",
                                "datatype": "datetime",
                                "dateTimeFormat": "unixtime",
                                "required": true,
                                "ontologyMappingType": "simple",
                                "ontologyMapping": ""
                        },
                        "UploadID": {
                                "description": "The UUID for the uploaded document",
                                "datatype": "string",
                                "required": true,
                                "ontologyMappingType": "simple",
                                "ontologyMapping":
"http://www.anl.gov/cfm/transform.owl#UniqueFileIdentifierSemanticConcept"
                        },
                        "DownloadElementExtendedAttribute_Field": {
                                "description": "Extended information type",
                                "valuemap": "DownloadElementExtendedAttribute;Field",
                                "datatype": "enum",
                                "required": false,
                                "ontologyMappingType": "enum",
                                "enumValues": {
                                        "origFileName": {
                                                "ontologyMapping": ""
                                        },
                                        "orig1.3Filename": {
                                                "ontologyMapping": ""
                                        },
                                        "comment": {
                                                "ontologyMapping": ""
                                        }
                                }
                        },
                        "DownloadElementExtendedAttribute_Value": {
                                "description": "The value for the extended data",
                                "valuemap": "DownloadElementExtendedAttribute;Value",
                                "datatype": "string",
                                "defaultValue": "NoValue",
                                "requiredIfReferenceField": "DownloadElementExtendedAttribute_Field",
                                "requiredIfReferenceValuesMatch": [ "*" ],
                                "ontologyMappingType": "referencedEnum",
                                "ontologyEnumField": "DownloadElementExtendedAttribute_Field",
                                "ontologyMappingEnumValues": {
                                        "origFileName": {
                                                "ontologyMapping": ""
                                        },
                                        "orig1.3Filename": {
                                                "ontologyMapping": ""
                                        },
                                        "comment": {
                                                "ontologyMapping": ""
                                        }
                                }
                        }
                }
        }
}
```