# Common Event Format

**ArcSight, Inc.**

July 17, 2009

Revision 15

**Common Event Format**

July 17, 2009



## Revision History

| Date | Description |
|---|---|
| 07/17/09 | Made corrections to Extension Dictionary table. |
| 05/18/2009 | Combined CEF, extension dictionary, custom dictionary extensions and data format information into this single document. |
| 11/12/2007 | Corrected errors in examples for backslash and equal sign. |
| 06/07/2006 | First external draft. |

# Contents

# An Event Interoperability Standard

In the realm of security event management, a myriad of event formats streaming from disparate devices makes for a complex integration. The following pages detail the ArcSight standard for promoting interoperability between various event- or log-generating devices.

Although each vendor has its own format for reporting event information, these event formats often lack the key information necessary to integrate the events from their devices.

The ArcSight standard attempts to improve the interoperability of infrastructure devices by aligning the logging output from various technology vendors.

# Common Event Format (CEF)

The format called Common Event Format (CEF) can be readily adopted by vendors of both security and non-security devices. This format contains the most relevant event information, making it easy for event consumers to parse and use them.

To simplify integration, the syslog message format is used as a transport mechanism. This applies a common prefix to each message, containing the date and hostname, as shown below.

`Jan 18 11:07:53 host` *`message`*

If an event producer is unable to write syslog messages, it is still possible to write the events to a file. To do so:

1. Omit the syslog header (shown above)
2. Begin the message with the format shown below

`CEF:`*`Version`*`|`*`Device Vendor`*`|`*`Device Product`*`|`*`Device Version`*`|`*`Signature ID`*`|`*`Name`*`|`*`Severity`*`|`*`Extension`*

After the mandatory **CEF:** prefix, the remainder of the message is formatted using a common prefix composed of fields delimited by a bar ("|") character. All of the fields specified above should be present and are defined under “Definitions of Prefix Fields” on page 2.

The *Extension* part of the message is a placeholder for additional fields. These additional fields are documented under “The Extension Dictionary” on page 4, and are logged as key-value pairs.

## Definitions of Prefix Fields

***Version*** is an integer and identifies the version of the CEF format. Event consumers use this information to determine what the following fields represent.

***Device Vendor***, ***Device Product*** and ***Device Version*** are strings that uniquely identify the type of sending device. No two products may use the same device-vendor and device-product pair. There is no central authority managing these pairs. Event producers have to ensure that they assign unique name pairs.

***Signature ID*** is a unique identifier per event-type. This can be a string or an integer. Signature ID identifies the type of event reported. In the intrusion detection system (IDS) world, each signature or rule that detects certain activity has a unique signature ID assigned. This is a requirement for other types of devices as well, and helps correlation engines deal with the events.

***Name*** is a string representing a human-readable and understandable description of the event. The event name should not contain information that is specifically mentioned in other fields. For example: “Port scan from 10.0.0.1 targeting 20.1.1.1” is not a good event name. It should be: “Port scan”. The other information is redundant and can be picked up from the other fields.

***Severity*** is an integer and reflects the importance of the event. Only numbers from 0 to 10 are allowed, where 10 indicates the most important event.

***Extension*** is a collection of key-value pairs. The keys are part of a predefined set. The standard allows for including additional keys as outlined under “The Extension Dictionary” on page 4. An event can contain any number of key-value pairs in any order, separated by spaces (“ “). If a field contains a space, such as a file name, this is valid and can be logged in exactly that manner, as shown below:

```
fileName=c:\Program<space>Files\ArcSight is a valid token.
```

The following example illustrates a CEF message using Syslog transport:

```
Sep 19 08:26:10 host CEF:0|security|threatmanager|1.0|100|worm
successfully stopped|10|src=10.0.0.1 dst=2.1.2.2 spt=1232
```

## Character Encoding

Because CEF uses the UTF-8 Unicode encoding method, please note the following

- The entire message has to be **UTF-8** encoded.

- If a **pipe** (|) is used in the prefix, it has to be escaped with a backslash (\). But note that pipes in the extension do not need escaping. For example:

  ```
  Sep 19 08:26:10 host
  CEF:0|security|threatmanager|1.0|100|detected a \| in
  message|10|src=10.0.0.1 act=blocked a | dst=1.1.1.1
  ```

- If a **backslash** (\) is used in the prefix or the extension, it has to be escaped with another backslash (\). For example:

  ```
  Sep 19 08:26:10 host
  CEF:0|security|threatmanager|1.0|100|detected a \\ in
  packet|10|src=10.0.0.1 act=blocked a \\ dst=1.1.1.1
  ```

- If an **equal sign** (=) is used in the extensions, it has to be escaped with a backslash (\). Equal signs in the prefix need no escaping. For example:

  ```
  Sep 19 08:26:10 host
  CEF:0|security|threatmanager|1.0|100|detected a = in
  message|10|src=10.0.0.1 act=blocked a \= dst=1.1.1.1
  ```

- **Multi-line** fields can be sent by CEF by encoding the newline character as **\n** or **\r**. Note that multiple lines are only allowed in the value part of the extensions. For example:

  ```
  Sep 19 08:26:10 host
  CEF:0|security|threatmanager|1.0|100|Detected a threat. No
  action needed.|10|src=10.0.0.1 msg=Detected a threat.\n No
  action needed.
  ```

# The Extension Dictionary

The following tables contain predefined keys that establish usages for both event producers and consumers. They display **key names** as well as **full names** for each key. It is the **key name** that is required in events.

| Key Name | Full Name | Data Type | Length | Meaning |
|---|---|---|---|---|
| act | deviceAction | String | 63 | Action mentioned in the event. |
| app | ApplicationProtocol | String | 31 | Application level protocol, example values are: HTTP, HTTPS, SSHv2, Telnet, POP, IMAP, IMAPS, etc. |
| cnt | baseEventCount | Integer | | A count associated with this event. How many times was this same event observed? |
| dvc | deviceAddress | IPV4 Address | 16 | Identifies the device that an event refers to in an IP network. The format is an IPv4 address. **Example**: “192.168.10.1” |
| dvchost | deviceHostName | String | 100 | The format should be a fully qualified domain name associated with the device node, when a node is available. **Examples**: “host.domain.com” or “host”. |
| dst | destinationAddress | IPv4 Address | | Identifies destination that the event refers to in an IP network. The format is an IPv4 address. **Example**: “192.168.10.1” |
| dhost | destinationHostName | String | 1023 | Identifies the destination that an event refers to in an IP network. The format should be a fully qualified domain name associated with the destination node, when a node is available. **Examples**: “host.domain.com” or “host”. |
| dmac | destinationMac Address | MAC Address | | Six colon-separated hexadecimal numbers. **Example**: “00:0D:60:AF:1B:61” |
| dntdom | destinationNtDomain | String | 255 | The Windows domain name of the destination address. |
| dpt | destinationPort | Integer | | The valid port numbers are between 0 and 65535. |
| dproc | destinationProcess Name | String | 1023 | The name of the process which is the event's destination. For example: "telnetd", or "sshd". |

| Key Name | Full Name | Data Type | Length | Meaning |
|---|---|---|---|---|
| duid | destination UserId | String | 1023 | Identifies the destination user by ID. For example, in UNIX, the root user is generally associated with user ID 0. |
| dpriv | destination UserPrivileges | String | 1023 | The allowed values are: “Administrator”, “User”, and “Guest”. This identifies the destination user's privileges. In UNIX, for example, activity executed on the root user would be identified with destinationUserPrivileges of “Administrator”.<br>This is an idealized and simplified view on privileges and can be extended in the future. |
| duser | destination UserName | String | 1023 | Identifies the destination user by name. This is the user associated with the event's destination. E-mail addresses are also mapped into the UserName fields. The recipient is a candidate to put into destinationUserName. |
| end | endTime | Time Stamp | | The time at which the activity related to the event ended. The format is MMM dd yyyy HH:mm:ss or milliseconds since epoch (Jan 1st 1970). An example would be reporting the end of a session. |
| fname | fileName | String | 1023 | Name of the file. |
| fsize | fileSize | Integer | | Size of the file. |
| in | bytesIn | Integer | | Number of bytes transferred inbound. Inbound relative to the source to destination relationship, meaning that data was flowing from source to destination. |
| msg | message | String | 1023 | An arbitrary message giving more details about the event. Multi-line entries can be produced by using \n as the new-line separator. |
| out | bytesOut | Integer | | Number of bytes transferred outbound. Outbound relative to the source to destination relationship, meaning that data was flowing from destination to source. |
| proto | transport Protocol | String | 31 | Identifies the Layer-4 protocol used. The possible values are protocol names such as TCP or UDP. |

| Key Name | Full Name | Data Type | Length | Meaning |
|---|---|---|---|---|
| rt | receiptTime | Time Stamp | | The time at which the event related to the activity was received. The format is MMM dd yyyy HH:mm:ss or milliseconds since epoch (Jan 1st 1970). |
| request | requestURL | String | 1023 | In the case of an HTTP request, this field contains the URL accessed. The URL should contain the protocol as well, e.g., "http://www.security.com" |
| src | SourceAddress | IPv4 Address | | Identifies the source that an event refers to in an IP network. The format is an IPv4 address. **Example**: "192.168.10.1" |
| shost | sourceHostName | String | 1023 | Identifies the source that an event refers to in an IP network. The format should be a fully qualified domain name associated with the source node, when a node is available. **Examples**: "host.domain.com" or "host". |
| smac | sourceMacAddress | MAC Address | | Six colon-separated hexadecimal numbers. **Example**: "00:0D:60:AF:1B:61" |
| sntdom | sourceNtDomain | String | 255 | The Windows domain name for the source address. |
| spt | sourcePort | Integer | | The valid port numbers are 0 to 65535. |
| spriv | sourceUser Privileges | String | 1023 | The allowed values are: "Administrator", "User", and "Guest". It identifies the source user's privileges. In UNIX, for example, activity executed by the root user would be identified with sourceUserPrivileges of "Administrator".<br>This is an idealized and simplified view on privileges and can be extended in the future. |
| suid | sourceUserId | String | 1023 | Identifies the source user by ID. This is the user associated with the source of the event. For example, in UNIX, the root user is generally associated with user ID 0. |
| suser | sourceUserName | String | 1023 | Identifies the source user by name. E-mail addresses are also mapped into the UserName fields. The sender is a candidate to put into sourceUserName. |

| Key Name | Full Name | Data Type | Length | Meaning |
|---|---|---|---|---|
| start | startTime | Time Stamp | | The time when the activity the event referred to started. The format is MMM dd yyyy HH:mm:ss or milliseconds since epoch (Jan 1st 1970). |

| Full and/or Key Names | | Data Type | Length | Meaning |
|---|---|---|---|---|
| cat | deviceEvent Category | String | 1023 | Represents the category assigned by the originating device. Devices oftentimes use their own categorization schema to classify events. |
| cs1Label<br>cs2Label<br>cs3Label<br>cs4Labelcs5Label<br>cs6Label | deviceCustom String1 Label<br>... | String | 1023 | All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. |
| cn1Label<br>cn2Label<br>cn3Label<br>deviceCustomDate1 Label<br>deviceCustomDate2 Label | deviceCustom Number1Label<br>...<br>deviceCustom Date1 Label<br>... | String | 1023 | All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. |
| cs1<br>cs2<br>cs3<br>cs4<br>cs5<br>cs6 | deviceCustomString1<br>... | String | 1023 | There are six *strings* available which can be used to map fields which do not fit into any other field of this dictionary. If possible, these fields should not be used, but a more specific field from the dictionary. Also check the guidelines later in this document for hints about utilizing these fields. |
| cn1<br>cn2<br>cn3 | deviceCustom Number1<br>... | Long | | There are three *number* fields available which can be used to map fields which do not fit into any other field of this dictionary. If possible, these fields should not be used, but a more specific field from the dictionary. Also check the guidelines hereafter for hints on how to utilize these fields. |
| deviceNtDomain | | String | 255 | The Windows domain name of the device address. |

| Full and/or Key Names | | Data Type | Length | Meaning |
|---|---|---|---|---|
| deviceDnsDomain | | String | 255 | The DNS domain part of the complete fully qualified domain name (FQDN). |
| deviceTranslatedAddress | | IPv4 Address | | Identifies the translated device address that the event refers to in an IP network. The format is an IPv4 address. **Example**: "192.168.10.1" |
| deviceMacAddress | | MAC Address | | Six colon-separated hexadecimal numbers. **Example**: "00:0D:60:AF:1B:61" |
| deviceCustomDate1<br>deviceCustomDate2 | | Time Stamp | | There are two timestamp fields available which can be used to map fields which do not fit into any other field of this dictionary. If possible, these fields should not be used, but a more specific field from the dictionary. Also check the guidelines later in this document for hints about utilizing these fields. |
| destinationDnsDomain | | String | 255 | The DNS domain part of the complete fully qualified domain name (FQDN). |
| dntdom | destinationNtDomain | String | 255 | The Windows domain name of the destination address. |
| dhost | DestinationHostName | String | 1023 | Identifies the source that an event refers to in an IP network. The format should be a fully qualified domain name associated with the destination node, when a node is available. **Examples**: "host.domain.com" or "host". |
| dpt | destinationPort | Integer | | The valid port numbers are between 0 and 65535. |
| dproc | destinationProcessName | String | 1023 | The name of the process which is the event's destination. For example: "telnetd", or "sshd". |
| destinationServiceName | | String | 1023 | The service which is targeted by this event. |
| duid | destinationUserId | String | 1023 | Identifies the destination user by ID. For example, in UNIX, the root user is generally associated with user ID 0. |

| Full and/or Key Names | | Data Type | Length | Meaning |
|---|---|---|---|---|
| dpriv | destinationUserPrivileges | String | 1023 | The allowed values are: “Administrator”, “User”, and “Guest”. This identifies the destination user's privileges. In UNIX, for example, activity executed on the root user would be identified with destinationUserPrivileges of “Administrator”.<br>This is an idealized and simplified view on privileges and can be extended in the future. |
| duser | destinationUserName | String | 1023 | Identifies the destination user by name. This is the user associated with the event's destination. E-mail addresses are also mapped into the UserName fields. The recipient is a candidate to put into destinationUserName. |
| destinationTranslatedAddress | | IPv4 Address | | Identifies the translated destination that the event refers to in an IP network. The format is an IPv4 address. **Example**: “192.168.10.1” |
| destinationTranslatedPort | | Integer | | Port after it was translated; for example, a firewall. Valid port numbers are 0 to 65535. |
| deviceDirection | | String | | Any information about what direction the communication that was observed has taken. |
| deviceExternalId | | String | 255 | A name that uniquely identifies the device generating this event. |
| deviceFacility | | String | 1023 | The facility generating this event. Syslog for example has an explicit facility associated with every event. |
| deviceInboundInterface | | String | 15 | Interface on which the packet or data entered the device. |
| deviceOutboundInterface | | String | 15 | Interface on which the packet or data left the device. |
| deviceProcessName | | String | 1023 | Process name associated to the event. In UNIX, the process generating the syslog entry for example. |
| end | endTime | Time Stamp | | The time at which the activity related to the event ended. The format is MMM dd yyyy HH:mm:ss or milliseconds since epoch (Jan 1st 1970). An example would be reporting the end of a session. |
| externalId | | Integer | | An ID used by the originating device. Usually these are increasing numbers associated with events. |

| Full and/or Key Names | | Data Type | Length | Meaning |
|---|---|---|---|---|
| fileCreateTime | | Time Stamp | | Time when file was created. |
| fileHash | | String | 255 | Hash of a file. |
| fileId | | String | 1023 | An ID associated with a file could be the inode. |
| fileModificationTime | | Time Stamp | | Time when file was last modified. |
| filePath | | String | 1023 | Full path to the file, including file name itself. |
| filePermission | | String | 1023 | Permissions of the file. |
| fileType | | String | 1023 | Type of file (pipe, socket, etc.) |
| oldfileCreateTime | | Time Stamp | | Time when old file was created. |
| oldfileHash | | String | 255 | Hash of the old file. |
| oldfileId | | String | 1023 | An ID associated with the old file could be the inode. |
| oldfileModificationTime | | Time Stamp | | Time when old file was last modified. |
| oldFilename | | String | 1023 | Name of the old file. |
| oldfilePath | | String | 1023 | Full path to the old file, including file name itself. |
| oldfilePermission | | String | 1023 | Permissions of the old file. |
| oldfsize | | Integer | | Size of the old file. |
| oldfileType | | String | 1023 | Type of the old file (pipe, socket, etc.) |
| rt | receiptTime | Time Stamp | | The time at which the event related to the activity was received. The format is MMM dd yyyy HH:mm:ss or milliseconds since epoch (Jan 1st 1970). |
| request | requestURL | String | 1023 | In the case of an HTTP request, this field contains the URL accessed. The URL should contain the protocol as well, e.g., http://www.security.com" |
| requestClientApplication | | String | 1023 | The User-Agent associated with the request. |
| requestCookies | | String | 1023 | Cookies associated with the request. |
| requestMethod | | String | 1023 | The method used to access a URL. Possible values: "POST", "GET", ... |

| Full and/or Key Names | | Data Type | Length | Meaning |
|---|---|---|---|---|
| sourceDnsDomain | | String | 255 | The DNS domain part of the complete fully qualified domain name (FQDN). |
| sourceServiceName | | String | 1023 | The service which is responsible for generating this event. |
| sourceTranslatedAddress | | IPv4 Address | | Identifies the translated source that the event refers to in an IP network. The format is an IPv4 address. **Example**: “192.168.10.1” |
| sourceTranslatedPort | | Integer | | Port after it was translated by for example a firewall. Valid port numbers are 0 to 65535. |
| spriv | sourceUser Privileges | String | 1023 | The allowed values are: “Administrator”, “User”, and “Guest”. It identifies the source user's privileges. In UNIX, for example, activity executed by the root user would be identified with sourceUserPrivileges of “Administrator”.<br>This is an idealized and simplified view on privileges and can be extended in the future. |
| start | startTime | Time Stamp | | The time when the activity the event referred to started. The format is MMM dd yyyy HH:mm:ss or milliseconds since epoch (Jan 1st 1970). |

# Special Mappings

In some cases, the mappings between fields of the original device and those of the ArcSight dictionary are not straightforward. The following tables provide examples that should help in such cases.

## Firewall

| Original Field | Mapped to Key Name | Mapped to Full Name |
|---|---|---|
| Rule Number / ACL Number | cs1 | deviceCustomString1 |

The *severity* for a blocked connection is **Medium**.

The *severity* for a passed connection is **Low**.

## Antivirus

| Original Field | Mapped to Key Name | Mapped to Full Name |
|---|---|---|
| Virus name | cs1 | deviceCustomString1 |
| Signature / Engine Version | cs2 | deviceCustomString2 |
| Action (Quarantine, Cleaned, Deleted, ...) | act | deviceAction |

## Email

| Original Field | Mapped to Key Name | Mapped to Full Name |
|---|---|---|
| Recipient (e.g. user@company.com) | duser | destinationUserName |
| Sender (e.g. user@company.com) | suser | sourceUserName |
| Relay | cs1 | deviceCustomString1 |

## Wireless

| Original Field | Mapped to Key Name | Mapped to Full Name |
|---|---|---|
| SSID | cs2 | deviceCustomString2 |
| Channel | cn1 | deviceCustomNumber1 |

# Custom Dictionary Extensions

The Extension Dictionary provides a broad set of predefined extension keys that should cover most event log requirements. In some cases, vendors' devices may generate more information than can be appropriately mapped into the predefined extensions or may generate information that does not fit the orientation of the predefined extensions. In such a case, vendors may define their own custom extensions to the standard extension dictionary.

## Custom Extension Naming Guidelines

Custom extension keys should take the following form

`VendornameProductnameExplanatoryKeyName`

Custom extension keys must also meet the following requirements. Custom extension key(s):

- may not be named the same as any key listed in the common or extended dictionaries.
- must be made up of a single word, with no spaces.
- must be alphanumeric.
- names should be as clear and concise as possible.

## Limitations of Custom Extensions

Custom extension keys are recommended for use only when no reasonable mapping of the information can be established for a predefined CEF key. While the custom extension key mechanism can be used to safely send information to CEF consumers for persistence, there are certain limitations as to when and how to access the data mapped into them.

Custom extension keys also have certain significant limitations that anyone implementing them should be aware of. These limitations fundamentally affect the experience of users of ArcSight products.

### Limitations Affecting ArcSight ESM

- Data submitted to ArcSight ESM using custom key extensions is retained, however it is largely inaccessible except when directly viewing events. This data shows up in a section called "Additional Data".

- Data submitted to ArcSight ESM using custom key extensions cannot be used directly for reporting, as these “Additional Data” fields are not made available in the reporting schema. Thus, any data in the “Additional Data” section of events is not available in reports.

- Data submitted to ArcSight ESM using custom key extensions cannot be used directly for event correlation (as within Rules, Data Monitors, etc.). Thus, any data in the “Additional Data” section is not available as output for correlation activities within the ESM system.

### Limitations Affecting ArcSight Logger

- Data submitted to ArcSight Logger using custom key extensions is retained in the system; however, it is not available for use in the Logger reporting infrastructure.

- Data submitted to ArcSight Logger using custom key extensions is available for viewing by the customer using string-based search. Event export is also be available for this purpose.

# Appendix: Data Formats

## Date Formats

CEF supports several variations on time/date formats to identify the time an event occurred accurately. These formats are detailed below.

1. Milliseconds since January 1, 1970 (integer)—This time format supplies an integer with the count in milliseconds from January 1, 1970 to the time the event occurred.

2. MMM dd HH:mm:ss

3. MMM dd HH:mm:ss.SSS zzz

4. MMM dd HH:mm:ss.SSS

5. MMM dd HH:mm:ss zzz

6. MMM dd yyyy HH:mm:ss

7. MMM dd yyyy HH:mm:ss.SSS zzz

8. MMM dd yyyy HH:mm:ss.SSS

9. MMM dd yyyy HH:mm:ss zzz

For a key to the date formats shown above, visit the SimpleDateFormat page at: java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html.