LOGPOINT

Integrations

Threat Intelligence For Director Console API

V6.3.0

CONTENTS

CHAPTER

ONE

THREAT INTELLIGENCE

Threat Intelligence (TI) fetches information and insights about existing or potential cyber threats and risks from various sources. It then assembles, processes and analyzes the fetched information and uses it to predict data breaches, vulnerable attacks and any evidence of pre-planned attacks or threats.

Supported Sources

- Emerging Threats
- Critical Stack
- CSIS
- Custom CSV
- MISP
- Blueliv
- Recorded Future
- StixTaxii

Threat Intelligence Components

1. **Enrichment Source**
    - ThreatIntelligence
2. **Process Command**
    - ti

CHAPTER

TWO

INSTALLING THREAT INTELLIGENCE

2.1 Uploading Threat Intelligence to the Fabric Storage

2.1.1 Private Location

You can upload Threat Intelligence (TI) to a private location in the Fabric Storage using the **Upload - Upload** API.

Endpoint URL:

```
https://api-server-host-name/configapi/Uploads/{pool_UUID}/
```

Method:

```
POST
```

Header:

FIELD	DESCRIPTION
file_name	Name of the Threat Intelligence .pak file
Content-Type	Mandatory value: *(application/octet-stream)*.

Parameters:

FIELD	TYPE	DESCRIPTION	Required
file	[Object]	The Threat Intelligence .pak file to upload.	Mandatory

Success Response:

```
{
   "status": "Success",
```

(continues on next page)

(continued from previous page)

```
    "message": "Threat Intelligence.pak successfully uploaded in the private storage. "
}
```

2.1.2 Public Location

You can upload Threat Intelligence to a public location in the Fabric Storage using the **Upload - UploadPublic files** API.

Endpoint URL:

```
https://api-server-host-name/configapi/Uploads/PublicUpload
```

Method:

```
POST
```

Header:

FIELD	DESCRIPTION
file_name	Name of the Threat Intelligence .pak file
Content-Type	Mandatory value: *(application/octet-stream)*.

Parameters:

FIELD	TYPE	DESCRIPTION	Required
file	[Object]	The Threat Intelligence .pak file to upload.	Mandatory

Success Response:

```
{
    "status": "Success",
    "message": "Threat Intelligence.pak successfully uploaded in the public storage. "
}
```

2.2 Installing Threat Intelligence

You can install Threat Intelligence in Logpoint using the **Upload - Install** API. It must be uploaded to the Fabric Storage before the installation.

Endpoint URL:

https://api-server-host-name/configapi/Uploads/{pool_UUID}/{logpoint_identifier}/install

Method:

POST

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
application_type		String	Mandatory value: *application*.	Mandatory
file_name	File	String	Name of the Threat Intelligence file.	Mandatory
file_location		String	Location of the Threat Intelligence file. Can be either *private* or *public*.	Mandatory

Request Example:

```
{
   "data": {
      "application_type": "Application",
      "file_name": "TI.pak",
      "file_location": "private"
   }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

CHAPTER

THREE

CONFIGURING THREAT INTELLIGENCE

3.1 General Settings

General Settings consists of all the details about the fetched data. The details of whether the data was successfully fetched or not is in **fetch_status**. The most recent attempt made to fetch data is in **last_fetch_attempt** and the last date and time when data was successfully fetched in **last_fetch_date**. The information of a disabled Threat Intelligence source is not displayed.

You can access the information of General Setting either by listing all the Threat Intelligence configurations or *getting* a configuration using ID.

3.1.1 Listing all the Threat Intelligence Configurations

You can list all the Threat Intelligence configurations using the **PluginConfiguration - List** API.

Note: Changes made to threat intelligence sources are not immediately reflected in the **PluginConfiguration - List** API. To update it, you must execute the *PluginConfiguration - RefreshList API*.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_general
```

Method:

```
GET
```

Success Response:

```
[
  {
    "services": {
      "critical_stack": {
        "last_successful_fetch": 0,
        "name": "Critical Stack",
        "last_fetch_attempt": 1552029459,
        "fetch_status": "Error",
        "error": "error occured"
      }
    },
    "no_of_entries": 0,
    "type": "ti_general",
    "tid": "",
    "id": "5c81f48e10959135395cabea"
  }
]
```


3.1.2 Getting a Threat Intelligence Configuration by ID

You can retrieve a Threat Intelligence configuration using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_general/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing general configuration to fetch.	Mandatory

Success Response:


```
{
  "services": {
    "critical_stack": {
      "last_successful_fetch": 0,
```


(continues on next page)

(continued from previous page)


```
        "name": "Critical Stack",
        "last_fetch_attempt": 1552029459,
        "fetch_status": "Error",
        "error": "error occured"
      }
    },
    "no_of_entries": 0,
    "type": "ti_general",
    "tid": "",
    "id": "5c81f48e10959135395cabea"
}
```


3.2 Emerging Threats

3.2.1 Configuring Emerging Threats

You can configure Emerging Threat using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
et_enable_source	Enable Source	boolean	Parameter to enable or disable Emerging Threats.	Mandatory
et_proxy	Proxy Configuration	JSON	Proxy configuration of the Emerging Threats source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
base_url	Base URL	String	Base URL of the Emerging Threats source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval	Fetch Interval	int	An interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_emergingthreat*.	Mandatory
api_key	API Key	String	API key of the Emerging Threats source.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory

Request Example:

```
{
    "data": {
        "et_enable_source": true,
        "et_proxy": {
            "status": true,
            "ip": "192.168.1.1",
            "protocol": "http",
            "port": 5555
        },
        "base_url": "https://example.com",
        "age_limit_unit": "Days",
        "fetch_interval": 1,
        "action": "ti_emergingthreat",
        "api_key": "1234xx234",
        "age_limit": 1,
        "fetch_interval_unit": "Days"
    }
}
```


Success Response:


```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```


3.2.2 Editing the Emerging Threats Configuration

You can edit the Emerging Threat configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
et_enable_source	Enable Source	boolean	Parameter to enable or disable Emerging Threats.	Mandatory
et_proxy	Proxy Configuration	JSON	Proxy configuration of the Emerging Threats source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
base_url	Base URL	String	Base URL of the Emerging Threats source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval	Fetch Interval	int	An interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_emergingthreat*.	Mandatory
api_key	API Key	String	API key of the Emerging Threats source.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
id		String	The ID of the existing Emerging Threats source configuration.	Mandatory

Request Example:

```
{
    "data": {
        "et_enable_source": true,
        "et_proxy": {
            "status": true,
            "ip": "192.168.1.10",
            "protocol": "http",
            "port": 5555
        },
        "base_url": "https://example.com",
        "age_limit_unit": "Days",
        "fetch_interval": 1,
        "action": "ti_emergingthreat",
        "api_key": "1234xx234",
        "age_limit": 1,
        "fetch_interval_unit": "Days",
    }
}
```

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.2.3 Listing the Emerging Threats Configurations

You can list the Emerging Threat configuration using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_emergingthreat
```

Method:

```
GET
```

Success Response:

```
[
    {
        "et_enable_source": "true",
        "base_url": "https://rules.emergingthreatspro.com",
```

(continues on next page)

(continued from previous page)


```
    "age_limit_unit": "Hours",
    "et_enable_source_confirmed": null,
    "et_proxy": {
      "status": false,
      "ip": "10.xx.x.xx",
      "protocol": "http",
      "port": 22
    },
    "fetch_interval": 3,
    "api_key": "1234xx234s",
    "age_limit": 2,
    "tid": "",
    "id": "5c82360f1095913d168c80c4",
    "fetch_interval_unit": "Days"
  }
]
```


Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.2.4 Getting an Emerging Threats Configuration by ID

You can fetch the Emerging Threats configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_emergingthreat/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing Emerging Threats source configuration to fetch.	Mandatory

Success Response:


```
{
    "et_enable_source": "true",
    "base_url": "https://rules.emergingthreatspro.com",
    "age_limit_unit": "Hours",
    "et_enable_source_confirmed": null,
    "et_proxy": {
        "status": false,
        "ip": "10.xx.x.xx",
        "protocol": "http",
        "port": 22
    },
    "fetch_interval": 3,
    "api_key": "1234xx234",
    "age_limit": 2,
    "tid": "",
    "id": "5c82360f1095913d168c80424",
    "fetch_interval_unit": "Days"
}
```


3.2.5 Removing an Emerging Threats Configuration by ID

You can delete the Emerging Threats configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *ti_emergingthreat*.	Mandatory
id	String	The ID of the existing Emerging Threats source configuration to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.3 Critical Stack

Important: We will be removing the critical stack threat source from the upcoming version, so it is recommended to use the MISP threat source.

3.3.1 Adding a Critical Stack Source Configuration

You can configure Critical Stack using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
api_name	API Name	String	Name of the Critical Stack source configuration.	Mandatory
action		String	Must be *ti_criticalstack*.	Mandatory
api_key	API Key	String	API key of the Critical Stack source.	Mandatory

Request Example:

```
{
   "data": {
      "api_name": "Ram",
```

(continues on next page)

(continued from previous page)

```
    "action": "ti_criticalstack",
    "api_key": "1234xx234"
  }
}
```

Success Response:

```
{
  "status": "Success",
  "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.3.2 Adding a Critical Stack Configuration

You can configure the Critical Stack using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
cs_proxy	Proxy Configuration	JSON	Proxy configuration of the Critical Stack source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
action		String	Must be *ti_criticalstacksettings*.	Mandatory
cs_enable_source	Enable Source	boolean	Parameter to enable or disable the Critical Stack source.	Mandatory

Request Example:

```
{
   "data": {
      "age_limit": 1,
      "age_limit_unit": "Days",
      "fetch_interval_unit": "Days",
      "fetch_interval": 1,
      "cs_proxy": {
         "status": true,
         "ip": "192.168.1.1",
         "protocol": "http",
         "port": 5555
      },
```

(continues on next page)

(continued from previous page)

```
    "action": "ti_criticalstacksettings",
    "cs_enable_source": true
  }
}
```

Success Response:

```
{
  "status": "Success",
  "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.3.3 Editing the Critical Stack Source Configuration

You can edit the Critical Stack configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
api_name	API Name	String	Name of the Critical Stack source configuration.	Mandatory
action		String	Must be *ti_criticalstack*.	Mandatory
api_key	API Key	String	API key of the Critical Stack source.	Mandatory
id		String	The ID of the existing Critical Stack source configuration.	Mandatory

Request Example:

```
{
   "data": {
      "api_name": "Ram",
      "action": "ti_criticalstack",
      "api_key": "1234xx234xs",
   }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.3.4 Editing the Critical Stack Configuration

You can edit the Critical Stack configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
cs_proxy	Proxy Configuration	JSON	Proxy configuration of the Critical Stack source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
action		String	Must be *ti_criticalstacksettings*.	Mandatory
cs_enable_source	Enable Source	boolean	Parameter to enable or disable the Critical Stack source.	Mandatory
id		String	The ID of the existing Critical Stack source settings configuration.	Mandatory

Request Example:

```
{
   "data": {
      "cs_enable_source": true,
      "age_limit_unit": "Days",
      "fetch_interval_unit": "Days",
      "fetch_interval": 1,
      "cs_proxy": {
         "status": true,
         "ip": "192.168.1.1",
```

(continues on next page)

(continued from previous page)

```
        "protocol": "https",
        "port": 5555
      },
      "action": "ti_criticalstacksettings",
      "age_limit": 1,
   }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.3.5 Listing the Critical Stack Source Configurations

You can list the Critical Stack source configuration using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_criticalstack
```

Method:

```
GET
```

Success Response:

```
[
   {
      "status": "Pending",
      "api_name": "cs",
      "tid": "",
      "api_key": "1234xx234",
      "id": "5cc28966d8aaa442a5dd88f7"

   }
]
```

Note: The **PluginConfiguration - List** API is not updated immediately after deleting the threat intelligence sources. To update the list, you must execute the

PluginConfiguration - RefreshList API.

3.3.6 Listing the Critical Stack Configurations

You can list the Critical Stack configuration using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_criticalstacksettings
```

Method:

```
GET
```

Success Response:

```
[
  {
    "name": "CriticalstackSettings",
    "age_limit": 1,
    "cs_enable_source_confirmed": false,
    "age_limit_unit": "Days",
    "fetch_interval": 1,
    "cs_proxy": {
      "status": true,
      "ip": "1.1.1.1",
      "protocol": "http",
      "port": 22
    },
    "cs_enable_source": true,
    "tid": "",
    "id": "5c81f48c1095913d0e63a2c2",
    "fetch_interval_unit": "Hours"
  }
]
```

Note: The **PluginConfiguration - List** API is not updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.3.7 Getting a Critical Stack Source Configuration by ID

You can fetch the Critical Stack source configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_criticalstack/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing Critical Stack configuration to fetch.	Mandatory

Success Response:

```
{
    "status": "Pending",
    "api_name": "cs",
    "tid": "",
    "api_key": "1234xx234",
    "id": "5cc28966d8aaa442a5dd88f7"
}
```

3.3.8 Getting a Critical Stack Configuration by ID

You can fetch the Critical Stack configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_criticalstacksettings/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing Critical Stack settings configuration to fetch.	Mandatory

Success Response:


```
{
  "name": "CriticalstackSettings",
  "age_limit": 1,
  "cs_enable_source_confirmed": false,
  "age_limit_unit": "Days",
  "fetch_interval": 1,
  "cs_proxy": {
    "status": true,
    "ip": "1.1.1.1",
    "protocol": "http",
    "port": 22
  },
  "cs_enable_source": true,
  "tid": "",
  "id": "5c81f48c1095913d0e63a2c2",
  "fetch_interval_unit": "Hours"
}
```


3.3.9 Removing a Critical Stack Configuration by ID

You can delete the Critical Stack source configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *ti_criticalstack* or *ti_criticalstacksettings*.	Mandatory
id	String	The ID of the existing Critical Stack source/settings configuration to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

Note: For multiple critical stack sources, you must apply the same fetch interval and age limit.

3.4 CSIS

3.4.1 Adding a CSIS Configuration

You can configure CSIS using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
csis_proxy	Proxy Configuration	JSON	Proxy configuration of the CSIS source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
base_url	Base URL	String	Base URL of the CSIS source.	Mandatory
api_token	API Token	String	API Token of the CSIS source.	Mandatory
csis_enable_source	Enable Source	boolean	Parameter to enable or disable the CSIS source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_csis*.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory

Request Example:

```
{
   "data": {
      "csis_proxy": {
         "status": true,
```

(continues on next page)

(continued from previous page)

```
        "ip": "192.168.1.10",
        "protocol": "http",
        "port": 5555
    },
    "base_url": "https://cdn.csis.dk/categories.csv.gz",
    "api_key":"api_key",
    "csis_enable_source": true,
    "age_limit_unit": "Days",
    "fetch_interval_unit": "Days",
    "fetch_interval": 1,
    "action": "ti_csis",
    "age_limit": 1
  }
}
```

Success Response:

```
{
  "status": "Success",
  "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.4.2 Editing the CSIS Configuration

You can update the CSIS configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
csis_proxy	Proxy Configuration	JSON	Proxy configuration of the CSIS source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
base_url	Base URL	String	Base URL of the CSIS source.	Mandatory
api_token	API Token	String	API Token of the CSIS source.	Mandatory
csis_enable_source	Enable Source	boolean	Parameter to enable or disable the CSIS source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_csis*.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
id		String	The ID of the existing CSIS source configuration.	Mandatory

Request Example:

```
{
    "data": {
        "csis_proxy": {
            "status": true,
            "ip": "192.168.1.10",
            "protocol": "http",
            "port": 5555
        },
        "base_url": "https://cdn.csis.dk/categories.csv.gz",
        "api_key":"api_key",
        "csis_enable_source": true,
        "age_limit_unit": "Days",
        "fetch_interval": 1,
        "action": "ti_csis",
        "age_limit": 1,
        "fetch_interval_unit": "Days",
    }
}
```

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.4.3 Listing the CSIS Configurations

You can list the CSIS configurations using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_csis
```

Method:

```
GET
```

Success Response:

```
[
    {
        "csis_proxy": {
            "status": true,
            "ip": "1.1.1.1",
```

(continues on next page)

(continued from previous page)


```
        "protocol": "http",
        "port": 1
      },
      "api_key": "1234xx234",
      "base_url": "https://cdn.csis.dk/categories.csv.gz",
      "csis_enable_source_confirmed": null,
      "csis_enable_source": true,
      "age_limit_unit": "Hours",
      "fetch_interval": 2,
      "age_limit": 2,
      "tid": "",
      "id": "5c823b2e1095913d0e63a2c4",
      "fetch_interval_unit": "Days"
   }
]
```


Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.4.4 Getting a CSIS Configuration by ID

You can fetch the CSIS configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_csis/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing CSIS configuration you want to fetch.	Mandatory

Success Response:

```
{
    "csis_proxy": {
        "status": true,
        "ip": "1.1.1.1",
        "protocol": "http",
        "port": 1
    },
    "api_key": "1234xx234",
    "base_url": "https://cdn.csis.dk/categories.csv.gz",
    "csis_enable_source_confirmed": null,
    "csis_enable_source": true,
    "age_limit_unit": "Hours",
    "fetch_interval": 2,
    "age_limit": 2,
    "tid": "",
    "id": "5c823b2e1095913d0e63a2c4",
    "fetch_interval_unit": "Days"
}
```


3.4.5 Removing a CSIS Configuration by ID

You can delete the CSIS configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *ti_csis*.	Mandatory
id	String	The ID of the existing CSIS configuration you want to delete.	Mandatory

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.5 Custom CSV

Custom CSV enables you to upload a custom CSV file as a TI source. The CSV file must have the following headers:

```
domain, category, score, first_seen, last_seen, ports, ip, url, type, file_hash
```

Note:

- The field **ports** is optional. You can specify multiple ports by separating it with space.
- The **first_seen** and **last_seen** data fields must have the *yyyy-mm-dd* format.
- Threat Intelligence ignores fields and their values if the CSV is not in the above format.

3.5.1 Adding a Custom CSV Configuration

You can configure Custom CSV using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
csv_proxy	Proxy Configuration	JSON	Proxy configuration of the Custom CSV source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
csv_enable_source	Enable Source	boolean	Parameter to enable or disable the Custom CSV source.	Mandatory
base_url	Base URL	String	Base URL of the Custom CSV source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_customcsv*.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory

Request Example:

```
{
   "data": {
```

(continues on next page)

(continued from previous page)

```
    "csv_proxy": {
      "status": true,
      "ip": "192.168.1.10",
      "protocol": "http",
      "port": 5555
    },
    "base_url": "https://cdn.csis.dk/categories.csv.gz",
    "csv_enable_source": true,
    "age_limit_unit": "Days",
    "fetch_interval_unit": "Days",
    "fetch_interval": 1,
    "action": "ti_customcsv",
    "age_limit": 1
  }
}
```

Success Response:

```
{
  "status": "Success",
  "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.5.2 Editing the Custom CSV Configuration

You can update the Custom CSV configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
csv_proxy	Proxy Configuration	JSON	Proxy configuration of the Custom CSV source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
csv_enable_source	Enable Source	boolean	Parameter to enable or disable the Custom CSV source.	Mandatory
base_url	Base URL	String	Base URL of the Custom CSV source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_customcsv*.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
id		String	The ID of the existing Custom CSV source configuration.	Mandatory

Request Example:

```
{
    "data": {
        "csv_proxy": {
            "status": true,
            "ip": "192.168.1.1",
            "protocol": "http",
            "port": 5555
        },
        "base_url": "https://cdn.csis.dk/categories.csv.gz",
        "csv_enable_source": true,
        "age_limit_unit": "Days",
        "fetch_interval": 1,
        "action": "ti_customcsv",
        "age_limit": 1,
        "fetch_interval_unit": "Days",
    }
}
```

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.5.3 Listing the Custom CSV Configurations

You can list the Custom CSV configurations using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_customcsv
```

Method:

```
GET
```

Success Response:

```
[
    {
        "csv_enable_source": true,
        "base_url": "https://cdn.csis.dk/categories.csv.gz",
        "csv_proxy": {
```

(continues on next page)

(continued from previous page)


```
        "status": true,
        "ip": "192.168.1.1",
        "port": 1313,
        "protocol": "https"
      },
      "csv_enable_source_confirmed": null,
      "fetch_interval": 3,
      "fetch_interval_unit": "Days",
      "age_limit": 2,
      "age_limit_unit": "Hours",
      "tid": "",
      "id": "5c82360f1095913d168c80c4",
      "action": "ti_customcsv"
   }
]
```


Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.5.4 Getting a Custom CSV Configuration by ID

You can fetch the Custom CSV configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_customcsv/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing Custom CSV source configuration you want to fetch.	Mandatory

Success Response:

```
{
    "csv_enable_source": true,
    "base_url": "https://cdn.csis.dk/categories.csv.gz",
    "csv_proxy": {
        "status": true,
        "ip": "192.168.1.1",
        "port": 1313,
        "protocol": "https"
    },
    "csv_enable_source_confirmed": null,
    "fetch_interval": 3,
    "fetch_interval_unit": "Days",
    "age_limit": 2,
    "age_limit_unit": "Hours",
    "tid": "",
    "id": "5r82360f1095913d168c80c4",
    "action": "ti_customcsv"
}
```


3.5.5 Removing a Custom CSV Configuration by ID

You can delete the Custom CSV configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *ti_customcsv*.	Mandatory
id	String	The ID of the existing Custom CSV source configuration you want to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6 MISP

3.6.1 Adding a MISP Configuration

You can configure the MISP source using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmispsettings
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
cm_enable_source	Enable Source	boolean	Parameter to enable or disable the MISP source.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
cm_proxy	Proxy Configuration	JSON	Proxy configuration of the MISP source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Optional

Request Example:

```
{
    "data":{
        "age_limit":2,
        "age_limit_unit":"Days",
        "fetch_interval_unit":"Days",
        "fetch_interval":1,
        "cm_proxy":{
          "status":false,
          "ip":"192.168.1.1",
          "protocol":"http",
          "port":5555
        },
        "action":"ti_circlmispsettings",
        "cm_enable_source":true
        }
```

(continues on next page)

(continued from previous page)

```
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6.2 Adding a MISP Source Configuration using an API Key

You can configure the MISP source using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmisp
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
misp_feed	API	String	Parameter to enable or disable the MISP source.	Mandatory
base_url	Base URL	String	Base URL of the MISP source.	Mandatory
api_key	API Key	String	API key of the MISP source.	Mandatory
filter_date	Logs From	Date	Date from when the application fetches logs.	Mandatory
filter_parameter	Filter Parameter	JSON	Option to specify the MISP database parameters in a JSON format to filter incoming events from MISP.	Mandatory
verify	Verify	boolean	Parameter to ensure a secure connection.	Optional
upload_file	Upload Certificate File	Object	Option to upload a self-signed SSL certificate.	Optional
file_location	–	String	Location of the file.	Mandatory
files	Files	JSON	Self-signed SSL certificate: **cert_file**: Self-signed SSL certificate.	Mandatory

Request Example:

```
{
    "data":{
        "base_url":"https://misp.com",
        "api_key":"xxxxxxx",
        "filter_date":"2020/12/13",
        "misp_feed":"api",
        "filter_parameter":"{\"eventid\": \"100\", \"to_ids\": 1}",
        "action":"ti_circlmisp"
        "verify": true,
        "upload_file":true,
        "file_location":"private",
        "files":{
```

(continues on next page)

(continued from previous page)

```
        "cert_file": "mi_kedited.crt"
        }
        }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6.3 Adding a MISP Source Configuration for Free MISP Feeds

You can configure the MISP source using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmisp
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
misp_feed	Free Feed	String	Option to fetch free MISP feeds.	Mandatory
base_url	Base URL	String	Base URL of the MISP source.	Mandatory
filter_date	Logs From	Date	Date from when the application fetches logs.	Mandatory

Request Example:

```
{
      "data":{
        "base_url":"https://www.botvrij.eu/data/feed-osint",
        "filter_date":"2020/12/12",
        "misp_feed":"free_feed",
        "action":"ti_circlmisp"
```

(continues on next page)

(continued from previous page)

```
        }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6.4 Editing the MISP Configuration

You can update the MISP configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmispsettings/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
cm_enable_source	Enable Source	boolean	Parameter to enable or disable the MISP source.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
cm_proxy	Proxy Configuration	JSON	Proxy configuration of the MISP source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Optional
id		String	The ID of the existing MISP source configuration.	Mandatory

Request Example:

```
{
    "data":{
        "age_limit":2,
        "age_limit_unit":"Days",
        "fetch_interval_unit":"Days",
```

(continues on next page)

(continued from previous page)

```
        "fetch_interval":1,
        "cm_proxy":{
          "status":false,
          "ip":"192.168.1.1",
          "protocol":"http",
          "port":5555
        },
        "action":"ti_circlmispsettings",
        "cm_enable_source":true
        }

}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6.5 Editing the MISP Source Configuration using an API Key

You can update the MISP source configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmisp/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
misp_feed	API	String	Parameter to enable or disable the MISP source.	Mandatory
base_url	Base URL	String	Base URL of the MISP source.	Mandatory
api_key	API Key	String	API key of the MISP source.	Mandatory
filter_date	Logs From	Date	Date from when the application fetches logs.	Mandatory
filter_parameter	Filter Parameter	JSON	Option to specify the MISP database parameters in a JSON format to filter incoming events from MISP.	Mandatory
verify	Verify	boolean	Parameter to ensure a secure connection.	Optional
upload_file	Upload Certificate File	Object	Option to upload a self-signed SSL certificate.	Optional
file_location	–	String	Location of the file.	Mandatory
files	Files	JSON	Self-signed SSL certificate: **cert_file**: Self-signed SSL certificate.	Mandatory
id		String	The ID of the existing MISP source configuration.	Mandatory

Request Example:

```
{
    "data":{
        "base_url":"https://misp.com",
        "api_key":"xxxxxxx",
        "filter_date":"2020/12/13",
```

(continues on next page)

(continued from previous page)

```
        "misp_feed":"api",
        "filter_parameter":"{\"eventid\": \"100\", \"to_ids\": 1}",
        "action":"ti_circlmisp"
        "verify": true,
        "upload_file":true,
        "file_location":"public",
        "files":{
        "cert_file": "mi_kedited.crt"
        }
        }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6.6 Editing the MISP Source Configuration for Free MISP Feeds

You can update the MISP source configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmisp/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
misp_feed	Free Feed	String	Option to fetch free MISP feeds.	Mandatory
base_url	Base URL	String	Base URL of the MISP source.	Mandatory
filter_date	Logs From	Date	Date from when the application fetches logs.	Mandatory
id		String	The ID of the existing MISP source configuration.	Mandatory

Request Example:

```
{
      "data":{
        "base_url":"https://www.botvrij.eu/data/feed-osint",
        "filter_date":"2020/12/12",
        "misp_feed":"free_feed",
        "action":"ti_circlmisp"
          }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6.7 Listing the MISP Configurations

You can list all the MISP configurations using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmispsettings
```

Method:

```
GET
```

Success Response:

```
[
  {
    "name": "CirclmispSettings",
    "fetch_interval": 2,
    "fetch_interval_unit": "Days",
    "age_limit": 30,
    "age_limit_unit": "Days",
    "cm_enable_source": true,
    "cm_proxy": {
      "status": false
    },
    "cm_enable_source_confirmed": false,
    "id": "5ffc0dd57c5d493b1b145374"
  }
]
```


Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.6.8 Listing the MISP Source Configurations

You can list all the MISP source configurations using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmisp
```

Method:

```
GET
```

Success Response:


```
[
    {
      "base_url": "https://www.botvrij.eu/data/feed-osint",
      "api_key": "",
      "misp_feed": "free_feed",
      "tid": "",
      "filter_date": "",
      "filter_parameter": "",
      "status": "Valid",
```


(continues on next page)

(continued from previous page)

```
        "id": "5ffc0dd57c5d493b1b145375"
    },
]
```

Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.6.9 Getting a MISP Configuration by ID

You can fetch the MISP configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmispsettings/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing MISP source configuration you want to fetch.	Mandatory

Success Response:

```
[
  {
    "name": "CirclmispSettings",
    "fetch_interval": 2,
    "fetch_interval_unit": "Days",
    "age_limit": 30,
    "age_limit_unit": "Days",
    "cm_enable_source": true,
    "cm_proxy": {
      "status": false
```

(continues on next page)

(continued from previous page)

```
    },
    "cm_enable_source_confirmed": false,
    "id": "5ffc0dd57c5d493b1b145374"
  }
]
```

3.6.10 Getting a MISP Source Configuration by ID

You can fetch the MISP source configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_circlmisp/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing MISP source configuration you want to fetch.	Mandatory

Success Response:

```
[
    {
        "base_url": "https://www.botvrij.eu/data/feed-osint",
        "api_key": "",
        "misp_feed": "free_feed",
        "tid": "",
        "filter_date": "",
        "filter_parameter": "",
        "status": "Valid",
        "id": "5ffc0dd57c5d493b1b145375"
    },
]
```

3.6.11 Removing a MISP Configuration by ID

You can delete the MISP configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *Ti_circlmispsettings*.	Mandatory
id	String	The ID of the existing MISP source configuration you want to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.6.12 Removing a MISP Configuration by ID

You can delete the MISP source configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *Ti_circlmisp*.	Mandatory
id	String	The ID of the existing MISP source configuration you want to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.7 Blueliv

3.7.1 Adding a Blueliv Configuration

You can configure the Blueliv source using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
bl_proxy	Proxy Configuration	JSON	Proxy configuration of the Blueliv source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
bl_enable_source	Enable Source	boolean	Parameter to enable or disable the Blueliv source.	Mandatory
base_url	Base URL	String	Base URL of the Blueliv source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_blueliv*.	Mandatory
api_key	API Key	String	API key of the Blueliv source.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory

Request Example:

```
{
  "data": {
    "fetch_interval": 1,
    "bl_proxy": {
```

(continues on next page)

(continued from previous page)

```
        "status": true,
        "ip": "192.168.1.10",
        "protocol": "http",
        "port": 5555
      },
      "base_url": "https://example.com",
      "age_limit_unit": "Days",
      "bl_enable_source": true,
      "action": "ti_blueliv",
      "api_key": "1234xx234",
      "age_limit": 1,
      "fetch_interval_unit": "Days"
   }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.7.2 Editing the Blueliv Configuration

You can update the Blueliv configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
bl_proxy	Proxy Configuration	JSON	Proxy configuration of the Blueliv source: **status**: Parameter to enable or disable the proxy server. **IP**: IP of the proxy server. **port**: Port of the proxy server. **protocol**: *HTTP* or *HTTPS* protocol used by the proxy server.	Mandatory
bl_enable_source	Enable Source	boolean	Parameter to enable or disable the Blueliv source.	Mandatory
base_url	Base URL	String	Base URL of the Blueliv source.	Mandatory
age_limit_unit	Age Limit	String	Unit of the age limit.	Mandatory
fetch_interval	Fetch Interval	int	Interval between the adjacent fetches.	Mandatory
action		String	Must be *ti_blueliv*.	Mandatory
api_key	API Key	String	API key of the Blueliv source.	Mandatory
age_limit	Age Limit	int	Expiration period of the fetched data.	Mandatory
fetch_interval_unit	Fetch Interval	String	Unit of the fetch interval.	Mandatory
id		String	The ID of the existing Blueliv source configuration.	Mandatory

Request Example:

```
{
    "data": {
        "bl_enable_source": true,
        "bl_proxy": {
            "status": true,
            "ip": "192.168.1.1",
            "protocol": "http",
            "port": 5555
        },
        "base_url": "https://example.com",
        "age_limit_unit": "Days",
        "fetch_interval": 1,
        "action": "ti_blueliv",
        "api_key": "1234xx234",
        "age_limit": 1,
        "fetch_interval_unit": "Days",
    }
}
```

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.7.3 Listing the Blueliv Configurations

You can list all the Blueliv configurations using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_blueliv
```

Method:

```
GET
```

Success Response:

```
[
    {
        "api_key": "1234xx234",
        "bl_proxy": {
            "status": true,
```

(continues on next page)

(continued from previous page)


```
        "ip": "1.1.1.1",
        "protocol": "http",
        "port": 1
      },
      "bl_enable_source_confirmed": null,
      "bl_enable_source": true,
      "base_url": "https://cdn.csis.dk/categories.csv.gz",
      "age_limit_unit": "Hours",
      "fetch_interval": 3,
      "age_limit": 2,
      "tid": "",
      "id": "5c82360f1095913d168c80c4",
      "fetch_interval_unit": "Days"
   }
]
```


Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.7.4 Getting a Blueliv Configuration by ID

You can fetch the Blueliv configuration with the given ID using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_blueliv/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing Blueliv source configuration you want to fetch.	Mandatory

Success Response:

```
{
    "api_key": "1234xx234",
    "bl_proxy": {
        "status": true,
        "ip": "1.1.1.1",
        "protocol": "http",
        "port": 1
    },
    "bl_enable_source_confirmed": null,
    "bl_enable_source": true,
    "base_url": "https://cdn.csis.dk/categories.csv.gz",
    "age_limit_unit": "Hours",
    "fetch_interval": 3,
    "age_limit": 2,
    "tid": "",
    "id": "5c82360f1095913d168c80c4",
    "fetch_interval_unit": "Days"
}
```


3.7.5 Removing a Blueliv Configuration by ID

You can delete the Blueliv configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *ti_blueliv*.	Mandatory
id	String	The ID of the existing Blueliv source configuration you want to delete.	Mandatory

Success Response:


```
{
    "status": "Success",
```


(continues on next page)

(continued from previous page)

```
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.8 Mapping

Mapping enables you to standardize logs by assigning the fields of fetched logs to the fields of the *Logpoint Threat Intelligence Taxonomy*. Threat Intelligence initially validates if you have mapped the field of a search query. If you have not mapped the field, Threat Intelligence searches the column with the same field name and enriches the logs.

The following fields are mapped by default:

- source_address to ip_address
- destination_address to ip_address

3.8.1 Adding a Threat Intelligence Mapping

You can add a mapping configuration using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
column	Column	String	Mapping column.	Mandatory
key	Key	String	Mapping key.	Mandatory
action		String	Must be *ti_mapping*.	Mandatory

Request Example:

```
{
    "data": {
        "column": "threat_source",
        "key": "device_address",
        "action": "ti_mapping"
    }
}
```

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.8.2 Editing the Threat Intelligence Mapping

You can update the mapping configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

```
PUT
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
column	Column	String	Mapping column.	Mandatory
key	Key	String	Mapping key.	Mandatory
action		String	Must be *ti_mapping*.	Mandatory
id		String	The ID of the existing mapping configuration.	Mandatory

Request Example:

```
{
   "data": {
      "column": "threat_source",
      "key": "device_address",
      "action": "ti_mapping",
   }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.8.3 Listing the Threat Intelligence Mappings

You can list the Mapping configurations using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_mapping
```

Method:

```
GET
```

Success Response:

```
[
   {
      "column": "threat_source",
      "key": "device_address",
```

(continues on next page)

(continued from previous page)

```
    "id": "5c80faa31095912a4da42cbd",
    "vid": "",
    "tid": ""
  }
]
```

Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.8.4 Getting a Threat Intelligence Mapping by ID

You can fetch the mapping configuration using the **PluginConfiguration - Get** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_mapping/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing mapping configuration you want to fetch.	Mandatory

Success Response:

```
{
  "column": "threat_source",
  "key": "device_address",
  "id": "5c80faa31095912a4da42cbd",
  "vid": "",
  "tid": ""
}
```

3.8.5 Removing a Threat Intelligence Mapping by ID

You can delete the mapping configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *ti_mapping*.	Mandatory
id	String	The ID of the existing mapping configuration that you want to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

Note: You can only map one key value to one column of the LogPoint taxonomy.

3.9 Alias

Alias enables you to assign a pseudoname to one or multiple field names of the incoming log.

3.9.1 Adding a Threat Intelligence Alias

You can add an alias configuration using the **PluginConfiguration - Create** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence
```

Method:

```
POST
```

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
name	Alias	String	Name of the configuration.	Mandatory
keys	Fields	[String]	A list of keys.	Mandatory
mode	Mode	String	Mode of the configuration.	Mandatory
action		String	Must be *ti_alias*.	Mandatory

Request Example:

```
{
    "data": {
        "name": "Ram",
        "keys": [
            "source_address"
        ],
        "mode": "all",
        "action": "ti_alias"
    }
}
```

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.9.2 Editing the Threat Intelligence Alias

You can update the alias configuration using the **PluginConfiguration - Edit** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}
```

Method:

PUT

Parameters:

FIELD	LABEL IN UI	TYPE	DESCRIPTION	Required
name	Alias	String	Name of the configuration.	Mandatory
keys	Fields	[String]	A list of keys.	Mandatory
mode	Mode	String	Mode of the configuration.	Mandatory
action		String	Must be *ti_alias*.	Mandatory
id		String	The ID of the existing alias configuration.	Mandatory

Request Example:

```
{
   "data": {
      "name": "Ram",
      "keys": [
         "source_address"
      ],
      "mode": "all",
      "action": "ti_alias",
   }
}
```

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

3.9.3 Listing the Threat Intelligence Aliases

You can list the alias configurations using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_alias
```

Method:

```
GET
```

Success Response:

```
[
  {
    "name": "Ram",
    "keys": [
      "source_address"
    ],
    "mode": "all",
    "vid": "",
    "tid": "",
    "id": "5c82360f1095913d168c80c4"
  }
]
```

Note: The **PluginConfiguration - List** API is **not** updated immediately after deleting the threat intelligence sources. To update the list, you must execute the *PluginConfiguration - RefreshList API*.

3.9.4 Getting a Threat Intelligence Alias by ID

You can fetch the alias configuration using the **PluginConfiguration - List** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪Ti_alias/{id}
```

Method:

```
GET
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing alias configuration you want to fetch.	Mandatory

Success Response:

```
{
    "name": "Ram",
    "keys": [
        "source_address"
    ],
    "mode": "all",
    "vid": "",
    "id": "5c82360f1095913d168c80c4",
    "tid": ""
}
```

3.9.5 Removing a Threat Intelligence Alias by ID

You can delete the alias configuration using the **PluginConfiguration - Trash** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/{id}?action={action}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
action	String	Must be *ti_alias*.	Mandatory
id	String	The ID of the existing alias configuration you want to delete.	Mandatory

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

CHAPTER

FOUR

UNINSTALLING THREAT INTELLIGENCE

You can uninstall Threat Intelligence (TI) in a Fabric-enabled Logpoint using the **Plugins - Uninstall** API.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/Plugins/{id}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
id	String	The ID of the existing Threat Intelligence application (Can be obtained using **Plugins - List** API).	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

CHAPTER

FIVE

THREAT INTELLIGENCE - REFRESHLIST

You can refresh the updated collections of Threat Intelligence using the **PluginConfiguration - RefreshList** API. The API synchronizes the Threat Intelligence data between Logpoint and Director Fabric.

Endpoint URL:

```
https://api-server-host-name/configapi/{pool_UUID}/{logpoint_identifier}/PluginConfiguration/
↪ThreatIntelligence/refreshlist
```

Method:

```
POST
```

Request Example:

```
{
    "data": {}
}
```

Success Response:

```
{
    "status": "Success",
    "message": "/monitorapi/{pool_UUID}/{logpoint_identifier}/orders/{request_id}"
}
```

CHAPTER

SIX

THREAT INTELLIGENCE - TRASH

6.1 Deleting Threat Intelligence from the Fabric Storage

6.1.1 Private Location

You can delete Threat Intelligence (TI) from a private location in the Fabric Storage using the **Upload - TrashPrivate** API.

Endpoint URL:

```
https://api-server-host-name/configapi/Uploads/{pool_UUID}/{file_name}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
file_name	String	Name of the Threat Intelligence pak file to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "Threat Intelligence.pak successfully deleted."
}
```

6.1.2 Public Location

You can delete Threat Intelligence from a public location in the Fabric Storage using the **Upload - TrashPrivate** API.

Endpoint URL:

```
https://api-server-host-name/configapi/Uploads/{file_name}
```

Method:

```
DELETE
```

Parameters:

FIELD	TYPE	DESCRIPTION	Required
file_name	String	Name of the Threat Intelligence pak file to delete.	Mandatory

Success Response:

```
{
   "status": "Success",
   "message": "Threat Intelligence.pak successfully deleted."
}
```

CHAPTER

SEVEN

APPENDIX

7.1 LogPoint Threat Intelligence Taxonomy

The Logpoint threat intelligence taxonomy specifies the following fields:

accessed_ts, application, authentication, caller_user, computer, created_ts, destination_address, destination_port, directory, disabled, domain, email, end_ts, file, fqdn, gateway, group_name, hardware_address, hash, hash_type, host, ip_address, locked_out, login_ts, loggoff_ts, logon_type, modified_ts, port, priority, process, protocol, proxy_server, referer, request_method, rights, security_id, server_address, service, source_address, source_port, start_ts, status, status_code, url, user, user_agent

Among these field names, only *domain, url, category, type, threat_source, file_hash, ip_address, score, port, _eviction_timestamp, start_ts, and end_ts* are functional in Threat Intelligence.