

INFOWATCH

SDK. Access to IWTM events for external systems

17/05/2018

© JSC “InfoWatch”

Tel/Fax +7 (495) 229-00-22

http://www.infowatch.com

Table of Contents

1 Purpose of DataExport API application programming interface 3

2 Aauthorization to access DataExport API 4

2.1 Work with plugins 4

2.1.1 Adding plugin 4

2.1.2 Plugin removal 4

2.2 Work with tokens 4

2.2.1 Adding token 4

2.2.2 Token's data editing 5

2.2.3 Token's value update 5

2.2.4 Copying token 5

2.2.5 Deleting token 5

3 DataExport API functionality 6

4 DataExport API third-party component registration 7

4.1 DataExport API third-party component registry file format 7

5 Functional description of DataExport API application programming interface 8

5.1 Getting extracted text of event 8

5.2 Getting event metadata 8

5.3 Getting intercepted file of event 8

5.4 Getting list of security objects catalogues 9

5.5 Getting events list 11

5.6 Getting tags list 11

1 Purpose of DataExport API application programming interface

This application programming interface is intended to integrate InfoWatch Traffic Monitor with external data receiving systems. As a part of the implementation DataExport API external systems get the events content and metadata from InfoWatch Traffic Monitor. DataExport API supports only data reading operation from InfoWatch Traffic Monitor.

2 Aauthorization to access DataExport API

Authorization to access DataExport API is provided via tokens mechanism, similar to other InfoWatch Traffic Monitor API's. To get token and review DataExport API licensing status it is required adding new type of plugin into the System - Plugin for DataExport API. All the activities related to plugins are performed within the InfoWatch Traffic Monitor console.

2.1 Work with plugins

2.1.1 Adding plugin

To add plugin:

1. Go to **Administration - Plugins**;
2. Within the area **Plugins** click .
3. Click **Add**.
4. Review the plugin description and click **Install**.

> **Note:**
>
> To update a plugin, the updated version of this plugin should be added.

2.1.2 Plugin removal

To remove plugin:

1. Go to **Administration - Plugins**;
2. Within the area **Plugins** click 
3. Click **Yes** to confirm the deletion.

2.2 Work with tokens

Token is generated automatically after the plugin has been added. All the activities related to tokens are performed within the InfoWatch Traffic Monitor console.

2.2.1 Adding token

To add token:

1. Go to **Administration - Plugins**;
2. Choose the plugin within the **Plugins** area;
3. Go to **Tokens** bookmark;
4. Click .

Note:

Token's value is generated automatically when setting up a token and it cannot be set by user.

2.2.2 Token's data editing

To change token's data:

1. Go to **Administration - Plugins**;
2. Choose the plugin within the **Plugins** area;
3. Go to **Tokens** bookmark;
4. By double left button click, enter the required info within the fields *Name* and *Description*.

2.2.3 Token's value update

If the token was compromised or if there is a potential token leak to the third party, it is required generating a new token's value.

To update token's value:

1. Go to **Administration - Plugins**;
2. Choose the plugin within the **Plugins** area;
3. Go to **Tokens** bookmark;
4. Click  .

2.2.4 Copying token

To copy token's value into the clipboard:

1. Go to **Administration - Plugins**;
2. Choose the plugin within the **Plugins** area;
3. Go to **Tokens** bookmark;
4. Click  .

2.2.5 Deleting token

To delete token:

1. Go to **Administration - Plugins**;
2. Choose the plugin within the **Plugins** area;
3. Go to **Tokens** bookmark;
4. Click  .

3 DataExport API functionality

The functionality of DataExport API includes the following:

- Able to specify the number of attributes to be included within the result:
    - Event ID;
    - Senders: list of contacts and authentication result, taking into account the potential conflicts;
    - Recipients: list of contacts, and authentication result, taking into account the potential conflicts;
    - Sender's computer, taking into account the potential conflict;
    - Type of event;
    - Date and time of interception;
    - Protocol;
    - Level of violation;
    - Verdict;
    - User's decision;
    - Path to file;
    - Web resource address;
    - Security objects and catalogues;
    - User's decision;
    - Event's size;
    - Tags;
- Able to specify the number of attributes to filtrate:
    - Date and time of interception;
    - Type of event;
    - Protocol;
    - Security objects and catalogues;
    - User's decision;
    - Event's size;
    - Tags;
- Separate event metadata obtaining with the ability to specify the number of attributes to be included within the result. In this regard, if the event contains few enclosed objects, all the extracted text to be assembled into one file;
- Obtaining all the extracted text of an event;
- Obtaining binary data of an event.

4 DataExport API third-party component registration

To integrate third-party receiving system with InfoWatch Traffic Monitor the following actions are to be taken:

1. Register the authentication type string of company - the component user in Infowatch JSC. Thus letting to connect a user company to the license, to be installed within the InfoWatch Traffic Monitor System;
2. Obtain the developer's license related to company's ID. This would let unloading the events data from InfoWatch Traffic Monitor;
3. Create third-party component registry file in accordance with the format below. The license, obtained as per the point 2 is to be added to the file;
4. Download the third-party component registry file toInfoWatch Traffic Monitor System using the InfoWatch Traffic Monitor console interface. Section "Administration" - "Plugins". (see "Traffic Monitor. Administration guide");
5. Obtain an access token within the downloaded third-party component's properties;

The System is ready to send the company's events data.

4.1 DataExport API third-party component registry file format

Plugin is a .zip format archive. The archive includes:

- **licenses** folder, containing license files;
- **icon** folder, containing the files with pictograms used for registered events (optionally);
- **manifest.json** file, containing information on a plugin.

For external systems, receiving data from TM **manifest.json** should contain age following information:

Title	If it is mandatory	Description
Version	Yes	Plugin version
Developing company's ID	Yes	In accordance with company name, specified within the license
Name	Yes	Plugin name
Systems-data receivers	Yes	List of systems, receiving data from IWTM, registered within this Plugin.
Description	Yes	Plugin description.
"Preset" feature	No	If it is a part of the System.
Unique ID	Yes	Unique 128-bit plugin ID (UUID).

5 Functional description of DataExport API application programming interface

5.1 Getting extracted text of event

Method `GET`

Resource: `event/{id}/text`

Parameters:

- `{id}` - OBJECT_ID of event

Supporting HTTP-heading `Range`

Response: All the extracted text of the event intercepted (if there are few enclosed documents, the text is to be extracted from all the documents and to be assembled into one block).

```
Content-type: text/plain.
```

5.2 Getting event metadata

Method: `GET`

Resource: `event/{id}`

`Parameters:`

- `{id}` - `OBJECT_ID` of event
- `with[]` - list of additional entities, to be added to the object (for example, senders, recipients, etc.).

Response: Event

Example:

Get event OBJECT_ID "1" with senders and recipients:

```
/xapi/event/1?with[]=senders&with[]=recipients
```

5.3 Getting intercepted file of event

Method: `GET`

Resource: `event/{id}/raw?download=0`

Parameters:

- `{id}` - OBJECT_ID of event, mandatory
- `download` - adding HTTP-heading to download a file (`Content-Disposition`, `Content-Transfer-Encoding` and others), mandatory, the value is 1 or 0 (default value is 0)

Supporting HTTP-heading `Range`

Response: File of intercepted event.For example, eml (for emails) or the file itself for copying file event.

5.4 Getting list of security objects catalogues

Description: List of security objects catalogues, sorted out by name.

Method: `GET`

Resource: `event/protectedCatalog`

Parameters:

- `start` - from what position the catalogue should be started to be given (default value is 0)
- `limit` - how many catalogues are to be given (default value is 100, but it can't be more than 1000)

Response: List of security objects catalogues, described in the JSON-Schema below:

JSON scheme of catalogues list output

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "data": {
            "type": "array",
            "description": "Массив каталогов ОЗ",
            "items": {
                "type": "object",
                "description": "Каталог ОЗ",
                "properties": {
                    "CATALOG_ID": {
                        "type": "string",
                        "description": "Идентификатор"
                    },
                    "DISPLAY_NAME": {
                        "type": "string",
                        "description": "Имя"
                    },
                    "CREATE_DATE": {
                        "type": "string",
                        "description": "Дата создания"
                    },
                    "IS_DELETED": {
                        "type": "integer",
                        "description": "Удален ли из активной конфигурации",
                        "enum": [
                            0,
                            1
                        ]
                    }
                },
                "required": [
                    "CATALOG_ID",
                    "DISPLAY_NAME",
                    "CREATE_DATE",
                    "IS_DELETED"
                ]
            }
        },
```

```
        "meta": {
            "description": "Дополнительные данные",
            "type": "object",
            "properties": {
                "totalCount": {
                    "type": "integer",
                    "description": "Общее количество ОЗ"
                },
                "start": {
                    "type": "integer",
                    "description": "С какой позиции началось отдаваться"
                },
                "limit": {
                    "type": "integer",
                    "description": "Сколько всего отдалось"
                }
            },
            "required": [
                "totalCount",
                "start",
                "limit"
            ]
        }
    },
    "required": [
        "data",
        "meta"
    ]
}
```

Example:

Request

```
GET /xapi/event/protectedCatalog?start=0&limit=10
```

Response

```
{
    "data": [{
        "CATALOG_ID": "3D0F8CDAF9E74F82AC85EB32D12727A700000000",
        "DISPLAY_NAME": "Грифованная информация",
        "CREATE_DATE": "2016-05-12 10:24:41.342554",
        "IS_DELETED": 0
    }],
    "meta": {
        "totalCount": 1,
        "start": 0,
        "limit": 10
    }
}
```

5.5 Getting events list

Description: List of events, sorted out by the time of interception (from new to old ones) .

Method: `GET`

Resource: `event`

Parameters:

- `filter` - list of filtration:
  - `date[from]` — start date in UNIX-timestamp format (can be used separately from `date[to]`)
  - `date[to]` — finish date in UNIX-timestamp format (can be used separately from `date[from]`)
  - `object_type_code[]` — type of event (can be several values)
  - `protocol[]` — protocol (can be several values)
  - `user_decision[]` — user decision (can be several values)
  - `object_size[from]` — from what bytes the event should be started to search
  - (can be used separately from `object_size[to]`)
  - `object_size[to]` — to what bytes the event should be searched for(can be used separately from `object_size[from]`)
  - `protected_catalog_id[]` — security objects catalogue ID (can be several values)
  - `tag_id[]` — tag ID (can be several values)
- `with[]` - list of additional entities, to be added to the object (for example, senders, recipients, etc.).
- `start` - from what position the events should be given (default value is 0)
- `limit` - how many events to be given (default value is 100, but not more than 1000)

Response: List of events, sorted out by the time of interception (from new to old ones).

Example:

Get the first 100 events of January 2015

```
/xapi/event?filter[date][from]=1420059600&filter[date][to]=1422737999&start=0&limit=100
```

5.6 Getting tags list

Description: Tags list, sorted out by name.

Method: `GET`

Resource: `event/tag`

Parameters:

- `start` - from what position the catalogue should be started to be given (default value is 0)
- `limit` - how many catalogues are to be given (default value is 100, but it can't be more than 1000)

Response: List of tags, described in the JSON-Schema below:

JSON scheme of tags list output


```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "data": {
            "type": "array",
            "description": "Массив тегов",
            "items": {
                "type": "object",
                "description": "Тег",
                "properties": {
                    "TAG_ID": {
                        "type": "string",
                        "description": "Идентификатор"
                    },
                    "DISPLAY_NAME": {
                        "type": "string",
                        "description": "Имя"
                    },
                    "COLOR": {
                        "type": "string",
                        "description": "Цвет тега в HEX RGP, например #ffffff"
                    },
                    "CREATE_DATE": {
                        "type": "string",
                        "description": "Дата создания"
                    },
                    "IS_DELETED": {
                        "type": "integer",
                        "description": "Удален ли из активной конфигурации"
                    },
                    "IS_SYSTEM": {
                        "type": "integer",
                        "description": "Является ли тег системным"
                    }
                },
                "required": [
                    "TAG_ID",
                    "DISPLAY_NAME",
                    "COLOR",
                    "CREATE_DATE",
                    "IS_DELETED",
                    "IS_SYSTEM"
                ]
            }
        },
        "meta": {
            "description": "Дополнительные данные",
            "type": "object",
            "properties": {
                "totalCount": {
                    "type": "integer",
                    "description": "Общее количество ОЗ"
                },
                "start": {
```

```
                    "type": "integer",
                    "description": "С какой позиции началось отдаваться"
                },
                "limit": {
                    "type": "integer",
                    "description": "Сколько всего отдалось"
                }
            },
            "required": [
                "totalCount",
                "start",
                "limit"
            ]
        }
    },
    "required": [
        "data",
        "meta"
    ]
}
```

Example:

Request

```
GET /xapi/event/tag?start=0&limit=10
```

Response

```
{
    "data": [{
        "TAG_ID": "0AF69DD5742E6977E0530100007F2BB700000000",
        "DISPLAY_NAME": "VIP",
        "COLOR": "#4e954e",
        "CREATE_DATE": "2016-05-12 10:24:40.002673",
        "IS_DELETED": 0,
        "IS_SYSTEM": 1
    }],
    "meta": {
        "totalCount": 1,
        "start": 0,
        "limit": 10
    }
}
```