
py42

Release py42 v0.8.0

Code42 Software

Apr 15, 2020

CONTENTS

1	Features	3
2	Content	5
	Python Module Index	55
	Index	57

py42 is a Python wrapper around the Code42 REST APIs that also provides several utility methods. Use py42 to develop your own tools for working with Code42 data while avoiding the overhead of session / authentication management.

FEATURES

- Managing users, organizations, and devices.
- Searching file events and alerts.
- Adding/Removing employees from detection lists.

CONTENT

2.1 User Guides

2.1.1 Getting started with py42

- *Licensing*
- *Installation*
- *Authentication*
- *Troubleshooting and Support*

Licensing

This project uses the [MIT License](#).

Installation

You can install py42 from PyPI, from source, or from distribution.

From PyPI

The easiest and most common way is to use `pip`:

```
pip install py42
```

To install a previous version of py42 via `pip`, add the version number. For example, to install version 0.4.1, you would enter:

```
pip install py42==0.4.1
```

Visit the [project history](#) on PyPI to see all published versions.

From source

Alternatively, you can install py42 directly from [source code](#):

```
git clone https://github.com/code42/py42.git
```

When it finishes downloading, from the root project directory, run:

```
python setup.py install
```

From distribution

If you want create a `.tar` ball for installing elsewhere, run this command from the project's root directory:

```
python setup.py sdist
```

After it finishes building, the `.tar` ball will be located in the newly created `dist` directory. To install it, enter:

```
pip install py42-[VERSION].tar.gz
```

Authentication

Important: py42 currently only supports token-based authentication.

To initialize the `py42.sdk.SDKClient`, you must provide your credentials (basic authentication). If you are writing a script, we recommend using a secure password storage library, such as `keyring`, for retrieving passwords. However, subsequent requests use JWT authentication.

py42 currently does **not** support SSO login providers or any other identity providers such as Active Directory or Okta.

Troubleshooting and support

Debug mode

Debug mode may be useful if you are trying to determine if you are experiencing permissions issues. When debug mode is on, py42 logs HTTP request data to the console. Use the following as a guide for how to turn on debug mode in py42:

```
import py42.sdk
import py42.settings.debug as debug

py42.settings.debug.level = debug.DEBUG
```

File an issue on GitHub

If you are experiencing an issue with py42, you can create a *New issue* at the [project repository](#). See the [Github guide on creating an issue](#) for more information.

Contact Code42 Support

If you don't have a GitHub account and are experiencing issues, contact [Code42 support](#).

What's next?

Learn the basics by following the py42 [Basics guide](#).

2.1.2 py42 Basics

This guide explains the basic concepts of py42. Learning these basics can help you gain confidence in writing your own scripts. The following examples use `py42.sdk.clients.departing_employee.DepartingEmployeeClient` to demonstrate how to use py42:

- *py42 Basics*
 - *Initialization*
 - *Paging*
 - *Py42Response*
 - *Dates*
 - *Exceptions*

The examples from this guide are intended as blanket concepts that apply to other areas in py42. For example, paging over users and devices works the same way as over departing employees and alerts.

Initialization

To use py42, you must initialize the SDK:

```
import py42.sdk

sdk = py42.sdk.from_local_account("https://console.us.code42.com", "my_username", "my_
↳password")
```

Paging

py42 clients often have a method with the name (or name prefix) `get_all` which handles iterating over pages of response items. Here are some examples:

- `py42.sdk.devices.get_all()`
- `py42.sdk.users.get_all()`
- `py42.sdk.legalhold.get_all_matters()`
- `py42.sdk.orgs.get_all()`

These methods each return a [python generator](#). Looping over the pages returned by the generator gives you access to the actual list of items. Use the code snippet below as an example for working with generators and paging in py42:

```
# Prints the username and notes for all departing employees

pages = sdk.detectionlists.departing_employee.get_all() # pages has 'generator' type
for page in pages: # page has 'Py42Response' type
    employees = page["cases"]
    for employee in employees:
        username = employee["userName"]
        notes = employee["notes"]
        print("{0}: {1}".format(employee, notes))
```

Each page is a typical py42 response. The next section covers what you can do with Py42Response objects.

Py42Response

py42 clients return Py42Response objects which are intentionally similar to `requests.Response` objects. The Py42Response class hides unneeded metadata found on the raw `requests.Response.text` (which is available as `Py42Response.raw_text`), making it easier to get the most useful parts of the response. Also, the object is subscriptable, meaning you can access it with keys or indices (depending on the JSON type underneath data on Code42 API responses):

```
user = response["users"][0]
item = list_response[0]["itemProperty"]
```

To see all the keys on a response, observe its `.text` attribute. By printing the response, you essentially print its text property:

```
# Prints details about the response from getting a departing employee

response = sdk.detectionlists.departing_employee.get_by_username("test.user@example.
↳com")
print(response) # JSON as Dictionary - same as print(response.text)
print(response.raw_text) # Raw API response
print(response.status_code) # 200
cloud_usernames = response["cloudUsernames"]
print(cloud_usernames)
```

Dates

py42 supports [POSIX timestamps](#) for date parameters. As an example, see the `departing_on_or_after_epoch` parameter in the `py42.sdk.clients.departing_employee.DepartingEmployeeClient.get_all()` method.

```
import py42.sdk
import py42.util
from datetime import datetime, timedelta

sdk = py42.sdk.from_local_account("https://console.us.code42.com", "my_username", "my_
↳password")

# Prints all the departing employee cases on or after two weeks

# get a date in the future
```

(continues on next page)

(continued from previous page)

```

departing_date = datetime.utcnow() + timedelta(days=14)
epoch = (departing_date - datetime.utcfromtimestamp(0)).total_seconds()
# get an epoch time (float)
response = sdk.detectionlists.departing_employee.get_all(departing_on_or_after_
↳epoch=epoch)
for page in response:
    for case in page["cases"]:
        print(case)

```

Exceptions

py42 throws some of its own exceptions when failures occur. py42 exceptions are found in the `py42.sdk.exceptions` module. Some of the available exceptions are:

- `Py42ForbiddenError`: (403) With your currently signed-in account, you don't have the necessary permissions to perform the action you were trying to do.
- `Py42UnauthorizedError`: (401) The username or password is incorrect.
- `Py42InternalServerError`: (500) Likely an unhandled issue on our servers.

For example, you are making a `create_sdk()` function and want to print a more user-friendly message when the provided username or password are incorrect:

```

import keyring
import py42.sdk
from py42.exceptions import Py42UnauthorizedError

def create_sdk(username):
    """Tries to initialize SDK. If unauthorized, prints message and exits."""
    try:
        password = keyring.get_password("my_program", username)
        return py42.sdk.from_local_account("www.authority.example.com", username,
↳password)
    except Py42UnauthorizedError:
        print("Invalid username or password.")
        exit(1)

```

2.1.3 Executing Searches

py42 includes a powerful, flexible query system that allows you to quickly and easily search file events and alerts. This guide helps you understand the syntax for building queries and executing searches.

File Event Searches

First, import the required modules and classes and create the SDK:

```
import py42.sdk
from py42.sdk.queries.fileevents.filters import *
from py42.sdk.queries.fileevents.file_event_query import FileEventQuery

sdk = py42.sdk.from_local_account("https://console.us.code42.com", "my_username", "my_
↳password")
```

You will need to create `query_filter.FilterGroup` objects to conduct searches. Filter groups have a type (in the form of a class), such as `EmailSender`, and an operator (in the form of a function), such as `is_in()`. Some example filter groups look like this:

```
email_filter = EmailSender.is_in(["test.user@example.com", "test.sender@example.com"])
exposure_filter = ExposureType.exists()
ip_filter = PrivateIPAddress.eq("127.0.0.1")
```

There are two operators when building `file_event_query.FileEventQuery` objects: `any()` and `all()`.

`any()` gets results where at least one of the filters is true and `all()` gets results where all of the filters are true.

```
any_query = FileEventQuery.any(email_filter, exposure_filter)
all_query = FileEventQuery.all(exposure_filter, ip_filter)
```

For convenience, the `FileEventQuery` constructor does the same as `all()`:

```
all_query = FileEventQuery(exposure_filter, ip_filter)
```

You can put filters in an iterable and unpack them (using the `*` operator) in a `FileEventQuery`. This is a common use case for programs that need to conditionally build up filters:

```
# Conditionally appends filters to a list for crafting a query

filter_list = []
if need_shared:
    filter_list.append(Shared.is_true())
elif need_actors:
    actor_filter = Actor.is_in(["foo@example.com", "baz@example.com"])
    filter_list.append(actor_filter)
# Notice the use of the '*' operator to unpack filter_list
query = FileEventQuery(*filter_list)
```

To execute the search, use `securitydata.SecurityModule.search_file_events()`:

```
# Prints the MD5 hashes of all the files that caused exposure events where files were_
↳moved to an external drive.

query = FileEventQuery(ExposureType.eq(ExposureType.REMOVABLE_MEDIA))
response = sdk.securitydata.search_file_events(query)
```

(continues on next page)

(continued from previous page)

```
file_events = response["fileEvents"]
for event in file_events:
    print(event["md5Checksum"])
```

Alert Searches

Conducting alert searches is very similar to file event searches. Much of the content for file event searches applies to alert searches.

To start, import the filters and query object:

```
from py42.sdk.queries.alerts.filters import *
from py42.sdk.queries.alerts.alert_query import AlertQuery
```

The one difference between constructing alert queries and file event queries is that alert queries require a tenant ID. You can get the tenant ID from the `sdk.usercontext` object:

```
# Create a query for getting all open alerts with severity either 'High' or 'Medium'.

filters = [AlertState.eq(AlertState.OPEN), Severity.is_in([Severity.HIGH, Severity.
↳MEDIUM])]
tenant_id = sdk.usercontext.get_current_tenant_id()
# Notice the constructor takes the tenant ID first.
query = AlertQuery(tenant_id, *filters)
```

To execute the search, use the `alerts.AlertClient.search()` method:

```
# Prints the actor property from each search result
response = sdk.securitydata.alerts.search(query)
alerts = response["alerts"]
for alert in alerts:
    print(alert["actor"])
```

2.1.4 Search and Add user to Departing Employee List

Let's say you want to add a user to the departing employee list and all you know is their user UID. We can achieve this using the `py42.sdk.users.get_by_uid()` method to recover their username, and then use the `py42.sdk.detectionlists.departing_employee.create()` method which requires a username as a parameter.

This is how you can get the username from the user UID:

```
user = sdk.users.get_by_uid("895005723650937319")
username = user["username"]
```

Next, use the `py42.sdk.detectionlists.departing_employee.create()` method to add that user to the departing employee list. Take note of its case ID if you want to do anything else right away:

```
# Add the departing employee
response = sdk.detectionlists.departing_employee.create(username)
case_id = response["caseId"] # Make note of its case ID
```

Important: If the user is already in the departing employee list, you will get a response indicating that it is a bad request.

The case ID you get back when you create a departing employee is useful for doing other things, such as resolving the departing employee:

```
sdk.detectionlists.departing_employee.resolve(case_id)
```

Let's say you do not have the case ID, now what do you do? We can get a case by username using the `py42.sdk.detectionlists.departing_employee.get_by_username()` method and extract the case ID from the response:

```
# Resolves departing employee 'test.user@example.com'

employee_item = sdk.detectionlists.departing_employee.get_by_username("test.
↪user@example.com")
case_id = employee_item["caseId"]
sdk.detectionlists.departing_employee.resolve(case_id)
```

In conclusion, you need a username for creating a departing employee and its case ID for doing anything else with that departing employee. However, there is a way to get departing employee details from just the username, which contains the case ID you need.

2.1.5 Get Active Devices from an Organization

Sometimes you might want to do something useful that requires information about active devices in your organization. For example, you might want to create a simple report that illustrates how many devices are running each operating system in your Code42 environment. Your role in the organization determines which devices you have access to.

To begin, we need to initialize the SDK:

```
import py42.sdk
sdk = py42.sdk.from_local_account("https://console.us.code42.com", "my_username", "my_
↪password")
```

The `DeviceClient.get_all()` Function

Next, we will want to make use of `py42.sdk.clients.devices.DeviceClient` to search for active devices in your organization. We can achieve this using the `active` parameter on the `get_all()` method.

The `active` parameter has three different states:

- If `active` is set to `True`, you will only get active devices.
- If `active` is set to `False`, you will only get deactivated devices.
- If you don't use `active`, you will get all devices.

The `get_all()` function returns a generator of pages of devices. Depending on your account's role in the organization, you will get different lists of devices:

- If you are logged in as an ordinary end user, `get_all()` returns all *your* devices.
- If you are logged in as an organization administrator, `get_all()` returns all the devices in your organization.
- If you are a cross-organization administrator, `get_all()` returns all devices across all your organizations.

- Finally, if you are a customer cloud administrator, `get_all()` returns all devices in all organizations.

Examples

Here is an example using `get_all()` to get all active devices in your organization(s):

```
# For each active device in your organization, print its GUID and operating system

response = sdk.devices.get_all(active=True)
for page in response:
    devices = page["computers"]
    for device in devices:
        print("{0} - {1}".format(device["guid"], device["osName"]))
```

Another example might be that you are a cross-organization administrator and you wish to get all the active devices for just one of your organizations. To do this, we can make use of the `py42.sdk.clients.devices.OrgClient.get_by_name()` method. The `get_by_name()` method returns a list of organizations matching the name you give it.

```
# For each active device in the engineering organization, print its GUID and
↪ operating system.

# Assume there is only one org named "Engineering"
engineering_org = sdk.orgs.get_by_name("Engineering")[0]
engineering_org_uid = engineering_org["orgUid"]
response = sdk.devices.get_all(active=True, org_uid=engineering_org_uid)
for page in response:
    devices = page["computers"]
    for device in devices:
        print("{0} - {1}".format(device["guid"], device["osName"]))
```

We got the org UID from the engineering organization and then passed it as a parameter to the method to get all the devices, thus getting all the active devices in the engineering organization.

2.2 Method Documentation

The main SDK object by which all other methods are accessed is created by calling `py42.sdk.from_local_account`. For example:

```
import py42.sdk

sdk = py42.sdk.from_local_account("console.us.code42.com", "john.doe@example.com",
    ↪ "my_pw")
# access properties on 'sdk' to explore all the available methods
```

Important: `py42` cannot be used with SAML or Sigle Sign-On based accounts such as Okta or Active Directory. Only accounts that are added by having an administrator create them within the Code42 console are currently supported.

Explore the complete public documentation for `py42` below.

2.2.1 Orgs

class `py42.clients.orgs.OrgClient` (*session*)

Bases: `py42.clients.BaseClient`

A client for interacting with Code42 organization APIs.

Use the `OrgClient` to create and retrieve organizations. You can also use it to block and deactivate organizations.

block (*org_id*)

Blocks the organization with the given org ID as well as its child organizations. A blocked organization will not allow any of its users or devices to log in. New registrations will be rejected and all currently logged in clients will be logged out. Backups continue for any devices that are still active. [Rest Documentation](#)

Parameters `org_id` (*int*) – An ID for an organization.

Returns `py42.response.Py42Response`

create_org (*org_name*, *org_ext_ref=None*, *notes=None*, *parent_org_uid=None*)

Creates a new organization. [REST Documentation](#)

Parameters

- **org_name** (*str*) – The name of the new organization.
- **org_ext_ref** (*str*, *optional*) – External reference information, such as a serial number, asset tag, employee ID, or help desk issue ID. Defaults to None.
- **notes** (*str*, *optional*) – Descriptive information about the organization. Defaults to None.
- **parent_org_uid** (*int*, *optional*) – The org UID for the parent organization. Defaults to None.

Returns `py42.response.Py42Response`

deactivate (*org_id*)

Deactivates the organization with the given ID, including all users, plans, and devices. Backups stop and archives move to cold storage. [REST Documentation](#)

Parameters `org_id` (*int*) – An ID for an organization.

Returns `py42.response.Py42Response`

get_all (***kwargs*)

Gets all organizations. [REST Documentation](#)

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of organizations.

Return type generator

get_by_id (*org_id*, ***kwargs*)

Gets the organization with the given ID. [REST Documentation](#)

Parameters `org_id` (*int*) – An ID for an organization.

Returns A response containing the organization.

Return type `py42.response.Py42Response`

get_by_name (*org_name*, ***kwargs*)

Gets the organization with the given name. [REST Documentation](#)

Parameters `org_name` (*str*) – A name of an organization.

Returns A list of `py42.response.Py42Response` objects each containing an organization that has the given name.

Return type list

get_by_uid (*org_uid*, ***kwargs*)

Gets the organization with the given UID. [REST Documentation](#)

Parameters *org_uid* (*str*) – A UID for an organization.

Returns A response containing the organization.

Return type `py42.response.Py42Response`

get_current (***kwargs*)

Gets the organization for the currently signed-in user. [REST Documentation](#)

Returns A response containing the organization for the currently signed-in user.

Return type `py42.response.Py42Response`

reactivate (*org_id*)

Reactivates the organization with the given ID. Backups are *not* restarted automatically. [REST Documentation](#)

Parameters *org_id* (*int*) – An ID for an organization.

Returns `py42.response.Py42Response`

unblock (*org_id*)

Removes a block, if one exists, on an organization and its descendants with the given ID. All users in the organization remain blocked until they are unblocked individually. [REST Documentation](#)

Parameters *org_id* (*int*) – An ID for an organization.

Returns `py42.response.Py42Response`

2.2.2 Users

class `py42.clients.users.UserClient` (*session*)

Bases: `py42.clients.BaseClient`

A client for interacting with Code42 user APIs. Use the `UserClient` to create and retrieve users. You can also use it to block and deactivate users.

block (*user_id*)

Blocks the user with the given ID. A blocked user is not allowed to log in or restore files. Backups will continue if the user is still active. [REST Documentation](#)

Parameters *user_id* (*int*) – An ID for a user.

Returns `py42.response.Py42Response`

change_org_assignment (*user_id*, *org_id*)

Assigns a user to a different organization. [REST Documentation](#)

Parameters

- *user_id* (*int*) – An ID for a user.
- *org_id* (*int*) – An ID for the organization to move the user to.

Returns `py42.response.Py42Response`

create_user (*org_uid, username, email=None, password=None, first_name=None, last_name=None, notes=None*)

Creates a new user. WARNING: If the provided username already exists for a user, it will be updated in the database instead. [REST Documentation](#)

Parameters

- **org_uid** (*str*) – The org UID for the organization the new user belongs to.
- **username** (*str*) – The username for the new user.
- **email** (*str, optional*) – The email for the new user. Defaults to None.
- **password** (*str, optional*) – The password for the new user. Defaults to None.
- **first_name** (*str, optional*) – The first name for the new user. Defaults to None.
- **last_name** (*str, optional*) – The last name for the new user. Defaults to None.
- **notes** (*str, optional*) – Descriptive information about the user. Defaults to None.

Returns *py42.response.Py42Response*

deactivate (*user_id, block_user=None*)

Deactivates the user with the given user ID. Backups discontinue for a deactivated user, and their archives go to cold storage. [REST Documentation](#)

Parameters

- **user_id** (*int*) – An ID for a user.
- **block_user** (*bool, optional*) – Blocks the user upon deactivation. Defaults to None.

Returns *py42.response.Py42Response*

get_all (*active=None, email=None, org_uid=None, role_id=None, q=None, **kwargs*)

Gets all users. [REST Documentation](#)

Parameters

- **active** (*bool, optional*) – True gets active users only, and false gets deactivated users only. Defaults to None.
- **email** (*str, optional*) – Limits users to only those with this email. Defaults to None.
- **org_uid** (*str, optional*) – Limits users to only those in the organization with this org UID. Defaults to None.
- **role_id** (*int, optional*) – Limits users to only those with a given role ID. Defaults to None.
- **q** (*str, optional*) – A generic query filter that searches across name, username, and email. Defaults to None.

Returns An object that iterates over *py42.response.Py42Response* objects that each contain a page of users.

Return type generator

get_by_id (*user_id, **kwargs*)

Gets the user with the given ID. [REST Documentation](#)

Parameters **user_id** (*int*) – An ID for a user.

Returns A response containing the user.

Return type `py42.response.Py42Response`

get_by_uid (`user_uid`, `**kwargs`)

Gets the user with the given UID. [REST Documentation](#)

Parameters `user_uid` (`str`) – A UID for a user.

Returns A response containing the user.

Return type `py42.response.Py42Response`

get_by_username (`username`, `**kwargs`)

Gets the user with the given username. [REST Documentation](#)

Parameters `username` (`str`) – A username for a user.

Returns A response containing the user.

Return type `py42.response.Py42Response`

get_current (`**kwargs`)

Gets the currently signed in user. [REST Documentation](#)

Returns A response containing the user.

Return type `py42.response.Py42Response`

reactivate (`user_id`, `unblock_user=None`)

Reactivates the user with the given ID. [REST Documentation](#)

Parameters

- `user_id` (`int`) – An ID for a user.
- `unblock_user` (`bool`, *optional*) – Whether or not to unblock the user. Defaults to None.

Returns `py42.response.Py42Response`

unblock (`user_id`)

Removes a block, if one exists, on the user with the given user ID. Unblocked users are allowed to log in and restore. [REST Documentation](#)

Parameters `user_id` (`int`) – An ID for a user.

Returns `py42.response.Py42Response`

class `py42.usercontext.UserContext` (`administration_client`)

Bases: `object`

An object representing the currently logged in user.

get_current_tenant_id ()

Gets the currently signed in user's tenant ID.

2.2.3 Devices

class `py42.clients.devices.DeviceClient` (*session*)

Bases: `py42.clients.BaseClient`

A class to interact with Code42 device/computer APIs.

block (*device_id*)

Blocks a device causing the user not to be able to login or restore.

REST Documentation

Parameters `device_id` (*int*) – The ID of the device.

Returns `py42.response.Py42Response`

deactivate (*device_id*)

Deactivates a device, causing backups to stop and archives to go to cold storage.

REST Documentation

Parameters `device_id` (*int*) – The ID of the device.

Returns `py42.response.Py42Response`

deauthorize (*device_id*)

Deauthorizes the device with the given ID. If used on a cloud connector device, it will remove the authorization token for that account.

REST Documentation

Parameters `device_id` (*int*) – The ID of the device.

Returns `py42.response.Py42Response`

get_all (*active=None, blocked=None, org_uid=None, user_uid=None, destination_guid=None, include_backup_usage=None, include_counts=True, q=None, **kwargs*)

Gets all device information.

When no arguments are passed, all records are returned. To filter results, specify respective arguments. For example, to retrieve all active and blocked devices, pass `active=true` and `blocked=true`.

REST Documentation

Parameters

- **active** (*bool, optional*) – Filters results by device state. When set to `True`, gets all active devices. When set to `False`, gets all deactivated devices. When set to `None` or excluded, gets all devices regardless of state. Defaults to `None`.
- **blocked** (*bool, optional*) – Filters results by blocked status, `True` or `False`. Defaults to `None`.
- **org_uid** (*int, optional*) – An ID of an Organization. Defaults to `None`.
- **user_uid** (*int, optional*) – An ID of a User. Defaults to `None`.
- **destination_guid** (*str of int, optional*) – The GUID of the server that the desktop clients back up their files to. Defaults to `None`.
- **include_backup_usage** (*type, bool*) – A flag to denote whether to include destination and its backup summary. Defaults to `None`.
- **include_counts** (*bool, optional*) – A flag to denote whether to include total, warning, and critical counts. Defaults to `True`.

- **q** (*type, optional*) – Searches results flexibly by incomplete GUID, host-name, computer name, etc. Defaults to None.

Returns

An object that iterates over `py42.response.Py42Response` objects that each contain a page of devices.

- For a logged in ordinary end user, it returns all the user's devices.
- For an organization administrator, it returns all the devices in the organization.
- For a cross-organization administrator, it returns all the devices from all the user's organizations.
- Finally, for are a customer cloud administrator, it returns all devices in all organizations.

Return type generator

get_by_guid (*guid, include_backup_usage=None, **kwargs*)

Gets device information by GUID.

REST Documentation

Parameters

- **guid** (*str*) – The GUID of the device.
- **include_backup_usage** (*bool, optional*) – A flag to denote whether to include its backup summary. Defaults to None.

Returns A response containing device information.

Return type `py42.response.Py42Response`

get_by_id (*device_id, include_backup_usage=None, **kwargs*)

Gets device information by ID.

REST Documentation

Parameters

- **device_id** (*int*) – The ID of the device.
- **include_backup_usage** (*bool, optional*) – A flag to denote whether to include its backup summary. Defaults to None.

Returns A response containing device information.

Return type `py42.response.Py42Response`

get_settings (*guid, keys=None*)

Gets settings of the device.

REST Documentation

Parameters

- **guid** (*str*) – The GUID of the device.
- **keys** (*type, optional*) – A comma separated list of device keys. Defaults to None.

Returns A response containing settings information.

Return type `py42.response.Py42Response`

reactivate (*device_id*)

Activates a previously deactivated device.

[REST Documentation](#)

Parameters **device_id** (*int*) – The ID of the device.

Returns *py42.response.Py42Response*

unblock (*device_id*)

Unblocks a device, permitting a user to be able to login and restore again.

[REST Documentation](#)

Parameters **device_id** (*int*) – The ID of the device.

Returns *py42.response.Py42Response*

2.2.4 Security Data

```
class py42.modules.securitydata.SecurityModule (security_client, storage_client_factory,  
                                                microservices_client_factory)
```

Bases: object

property alerts

A collection of methods for retrieving and updating data about security alerts.

```
get_all_plan_security_events (plan_storage_info, cursor=None, include_files=True,  
                              event_types=None, min_timestamp=None,  
                              max_timestamp=None)
```

Gets legacy endpoint monitoring events. This does not retrieve cloud events. [Support Article](#)

Parameters

- **plan_storage_info** (*py42.sdk.modules.securitydata.PlanStorageInfo*) – Information about storage nodes for a plan to get security events for.
- **cursor** (*str, optional*) – A cursor position for only getting events you did not previously get. Defaults to None.
- **include_files** (*bool, optional*) – Whether to include the files related to the security events. Defaults to None.
- **event_types** – (*str, optional*): A comma-separated list of event types to filter by.

Available options are:

- `DEVICE_APPEARED`
- `DEVICE_DISAPPEARED`
- `DEVICE_FILE_ACTIVITY`
- `PERSONAL_CLOUD_FILE_ACTIVITY`
- `RESTORE_JOB`
- `RESTORE_FILE`
- `FILE_OPENED`
- `RULE_MATCH`
- `DEVICE_SCAN_RESULT`

– `PERSONAL_CLOUD_SCAN_RESULT`

Defaults to None.

- **`min_timestamp`** (*float, optional*) – A POSIX timestamp representing the lower bound of the date range of events to get. Defaults to None.
- **`max_timestamp`** (*float, optional*) – A POSIX timestamp representing the upper bound of the date range of events to get. Defaults to None.

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of events.

Return type generator

`get_all_user_security_events` (*user_uid, cursor=None, include_files=True, event_types=None, min_timestamp=None, max_timestamp=None*)

Gets legacy security events for the user with the given UID.

Parameters

- **`user_uid`** (*str*) – The UID for the user to get security events for.
- **`cursor`** (*str, optional*) – A cursor position for only getting events you did not previously get. Defaults to None.
- **`include_files`** (*bool, optional*) – Whether to include the files related to the security events. Defaults to None.
- **`event_types`** – (*str, optional*): A comma-separated list of event types to filter by.

Available options are:

– `DEVICE_APPEARED`
 – `DEVICE_DISAPPEARED`
 – `DEVICE_FILE_ACTIVITY`
 – `PERSONAL_CLOUD_FILE_ACTIVITY`
 – `RESTORE_JOB`
 – `RESTORE_FILE`
 – `FILE_OPENED`
 – `RULE_MATCH`
 – `DEVICE_SCAN_RESULT`
 – `PERSONAL_CLOUD_SCAN_RESULT`

Defaults to None.

- **`min_timestamp`** (*float, optional*) – A POSIX timestamp representing the lower bound of the date range of events to get. Defaults to None.
- **`max_timestamp`** (*float, optional*) – A POSIX timestamp representing the upper bound of the date range of events to get. Defaults to None.

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of events.

Return type generator

get_security_plan_storage_info_list (*user_uid*)

Gets IDs (plan UID, node GUID, and destination GUID) for the storage nodes containing the legacy security event data for the user with the given UID. [REST Documentation](#)

Parameters **user_uid** (*str*) – A UID for the user to get plan storage information for.

Returns list[*py42.modules.securitydata.PlanStorageInfo*]

search_file_events (*query*)

Searches for file events. [REST Documentation](#)

Parameters **query** (*py42.sdk.queries.fileevents.file_event_query.FileEventQuery*) – Also accepts a raw JSON str.

Returns A response containing the first 10,000 events.

Return type *py42.response.Py42Response*

class *py42.modules.securitydata.PlanStorageInfo* (*plan_uid*, *destination_guid*,
node_guid)

Bases: object

property **destination_guid**

A GUID for the destination containing the storage archive.

property **node_guid**

The GUID for the storage node containing the archive.

property **plan_uid**

A UID for a storage plan.

2.2.5 Legal Hold

class *py42.clients.legalhold.LegalHoldClient* (*session*)

Bases: *py42.clients.BaseClient*

A client for interacting with Code42 Legal Hold APIs.

The LegalHoldClient provides the ability to manage Code42 Legal Hold Policies and Matters. It can:

- Create, view, and list all existing Policies.
- Create, view, deactivate, reactivate, and list all existing Matters.
- Add/remove Custodians from a Matter.

add_to_matter (*user_uid*, *legal_hold_uid*)

Add a user (Custodian) to a Legal Hold Matter. [REST Documentation](#)

Parameters

- **user_uid** (*str*) – The identifier of the user.
- **legal_hold_uid** (*str*) – The identifier of the Legal Hold Matter.

Returns *py42.response.Py42Response*

create_matter (*name*, *hold_policy_uid*, *description=None*, *notes=None*, *hold_ext_ref=None*)

Creates a new, active Legal Hold Matter. [REST Documentation](#)

Parameters

- **name** (*str*) – The name of the new Legal Hold Matter.

- **hold_policy_uid** (*str*) – The identifier of the Preservation Policy that will apply to this Matter.
- **description** (*str*, *optional*) – An optional description of the Matter. Defaults to None.
- **notes** (*str*, *optional*) – Optional notes information. Defaults to None.
- **hold_ext_ref** (*str*, *optional*) – Optional external reference information. Defaults to None.

Returns *py42.response.Py42Response*

create_policy (*name*, *policy=None*)

Creates a new Legal Hold Preservation Policy. [V4 REST Documentation](#)

Parameters

- **name** (*str*) – The name of the new Policy.
- **policy** (*dict*, *optional*) – The desired Preservation Policy settings as a dict. Defaults to None (where the server-default backup set is used).

Returns *py42.response.Py42Response*

deactivate_matter (*legal_hold_uid*)

Deactivates and closes a Legal Hold Matter. [V4 REST Documentation](#)

Parameters **legal_hold_uid** (*str*) – The identifier of the Legal Hold Matter.

Returns *py42.response.Py42Response*

get_all_matter_custodians (*legal_hold_uid=None*, *user_uid=None*, *user=None*, *active=True*)

Gets all Legal Hold memberships.

Each user (Custodian) who has been added to a Legal Hold Matter is returned by the server as a LegalHoldMembership object in the response body. If the object's active state is "INACTIVE", they have been removed from the Matter and are no longer subject to the Legal Hold retention rules. Users can be Custodians of multiple Legal Holds at once (and thus would be part of multiple LegalHoldMembership objects).

[REST Documentation](#)

Parameters

- **legal_hold_uid** (*str*, *optional*) – Find LegalHoldMemberships for the Legal Hold Matter with this unique identifier. Defaults to None.
- **user_uid** (*str*, *optional*) – Find LegalHoldMemberships for the user with this identifier. Defaults to None.
- **user** (*str*, *optional*) – Find LegalHoldMemberships by flexibly searching on username, email, extUserRef, or last name. Will find partial matches. Defaults to None.
- **active** (*bool or None*, *optional*) – Find LegalHoldMemberships by their active state. True returns active LegalHoldMemberships, False returns inactive LegalHoldMemberships, None returns all LegalHoldMemberships regardless of state. Defaults to True.

Returns An object that iterates over *py42.response.Py42Response* objects that each contain a page of LegalHoldMembership objects.

Return type generator

get_all_matters (*creator_user_uid=None, active=True, name=None, hold_ext_ref=None*)

Gets all existing Legal Hold Matters. [REST Documentation](#)

Parameters

- **creator_user_uid** (*str, optional*) – Find Matters by the identifier of the user who created them. Defaults to None.
- **active** (*bool or None, optional*) – Find Matters by their active state. True returns active Matters, False returns inactive Matters, None returns all Matters regardless of state. Defaults to True.
- **name** (*str, optional*) – Find Matters with a ‘name’ that either equals or partially contains this value. Defaults to None.
- **hold_ext_ref** (*str, optional*) – Find Matters having a matching external reference field. Defaults to None.

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of Legal Hold Matters.

Return type generator

get_matter_by_uid (*legal_hold_uid*)

Gets a single Legal Hold Matter. [REST Documentation](#)

Parameters **legal_hold_uid** (*str*) – The identifier of the Legal Hold Matter.

Returns A response containing the Matter.

Return type `py42.response.Py42Response`

get_policy_by_uid (*legal_hold_policy_uid*)

Gets a single Preservation Policy. [V4 REST Documentation](#)

Parameters **legal_hold_policy_uid** (*str*) – The identifier of the Preservation Policy.

Returns A response containing the Policy.

Return type `py42.response.Py42Response`

get_policy_list ()

Gets a list of existing Preservation Policies. [V4 REST Documentation](#)

Returns A response containing the list of Policies.

Return type `py42.response.Py42Response`

reactivate_matter (*legal_hold_uid*)

Reactivates and re-opens a closed Matter. [REST Documentation](#)

Parameters **legal_hold_uid** (*str*) – The identifier of the Legal Hold Matter.

Returns `py42.response.Py42Response`

remove_from_matter (*legal_hold_membership_uid*)

Remove a user (Custodian) from a Legal Hold Matter. [REST Documentation](#)

Parameters **legal_hold_membership_uid** (*str*) – The identifier of the LegalHold-Membership representing the Custodian to Matter relationship.

Returns `py42.response.Py42Response`

2.2.6 Detection Lists

class `py42.modules.detectionlists.DetectionListsModule` (*microservice_client_factory*,
user_client)

Bases: `object`

add_cloud_alias (*user_id*, *aliases*)

Add one or more cloud alias.

Parameters

- **user_id** (*str* or *int*) – The user_id whose alias(es) need to be updated.
- **aliases** (*str* or *list of str*) – A single alias or multiple aliases in a list to be added. e.g u"x" or ["email@id", "y"], for python version 2.X, pass u"str" instead of "str"

Returns `py42.response.Py42Response`

add_risk_tag (*user_id*, *tags*)

Add one or more tags.

Parameters

- **user_id** (*str* or *int*) – The user_id whose tag(s) needs to be updated.
- **tags** (*str* or *list of str*) – A single tag or multiple tags in a list to be added. e.g u"tag1" or ["tag1", "tag2"], for python version 2.X, pass u"str" instead of "str"

Returns `py42.response.Py42Response`

create (*username*)

Create a detection list profile for a user.

Parameters **username** (*str*) – Username of the user.

Returns `py42.response.Py42Response`

get (*username*)

Get user details by username.

Parameters **username** (*str*) – Username of the user.

Returns `py42.response.Py42Response`

get_by_id (*user_id*)

Get user details by user_id.

Parameters **user_id** (*str* or *int*) – Id of the user.

Returns `py42.response.Py42Response`

remove_cloud_alias (*user_id*, *aliases*)

Remove one or more cloud alias.

Parameters

- **user_id** (*str* or *int*) – The user_id whose alias(es) need to be removed.
- **aliases** (*str* or *list of str*) – A single alias or multiple aliases in a list to be removed. e.g u"x" or ["email@id", "y"], for python version 2.X, pass u"str" instead of "str"

Returns `py42.response.Py42Response`

remove_risk_tag (*user_id*, *tags*)

Remove one or more tags.

Parameters

- **user_id** (*str* or *int*) – The user_id whose tag(s) needs to be removed.
- **tags** (*str* or *list of str*) – A single tag or multiple tags in a list to be removed. e.g u"tag1" or ["tag1", "tag2"], for python version 2.X, pass u"str" instead of "str"

Returns *py42.response.Py42Response*

update_notes (*user_id*, *notes*)

Add or update notes related to the user.

Parameters

- **user_id** (*str* or *int*) – The user_id whose notes need to be updated.
- **notes** (*str*) – User profile notes.

Returns *py42.response.Py42Response*

Departing Employee

class *py42.clients.detectionlists.departing_employee*.**DepartingEmployeeClient** (*session*,
user_context)

Bases: *py42.clients.BaseClient*

A client for interacting with Code42 Departing Employee APIs.

create (*username*, *tenant_id=None*, *notes=None*, *departure_epoch=None*, *cloud_usernames=None*)

Adds a user to Departing Employees. [REST Documentation](#)

Parameters

- **username** (*str*) – The username of the departing employee.
- **tenant_id** (*str*, *optional*) – The identifier of the Customer tenant the user is in. Defaults to None (the tenant_id of SDK authorization user will be used).
- **notes** (*str*, *optional*) – Optional descriptive information. Defaults to None.
- **departure_epoch** (*int*, *optional*) – Employee departure date as POSIX timestamp. Defaults to None.
- **cloud_usernames** (*list*, *optional*) – List of alternate usernames for this user from external data sources being searched by Forensic File Search (Google Drive, Box, OneDrive, Gmail, Office 365). Defaults to None.

Returns *py42.response.Py42Response*

get_all (*tenant_id=None*, *departing_on_or_after_epoch=None*, *sort_key='CREATED_AT'*,
sort_direction='DESC')

Gets all Departing Employees.

Parameters

- **tenant_id** (*str*, *optional*) – The identifier of the Customer tenant. Defaults to None (the tenant_id of SDK authorization user will be used).
- **departing_on_or_after_epoch** (*int*, *optional*) – Filter results by the departure date of employee, requires a POSIX timestamp. Defaults to None.

- **sort_key** (*str*, *optional*) – Key to sort results on. Options: (CREATED_AT, DEPARTURE_DATE, DISPLAY_NAME, NUM_EVENTS, TOTAL_BYTES). Defaults to CREATED_AT.
- **sort_direction** (*str*, *optional*) – Sort direction. Options: (ASC, DESC). Defaults to DESC.

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of departing employees.

Return type generator

get_by_id (*case_id*, *tenant_id=None*)

Gets details about a Departing Employee case. [REST Documentation](#)

Parameters

- **case_id** (*str*) – Identifier of the Departing Employee case.
- **tenant_id** (*str*, *optional*) – The identifier of the Customer tenant the case is in. Defaults to None (the tenant_id of SDK authorization user will be used).

Returns A response containing the Departing Employee case.

Return type `py42.response.Py42Response`

get_by_username (*username*, *tenant_id=None*)

Gets Departing Employee case detail for a given user.

Parameters

- **username** (*str*) – Username of the Departing Employee to retrieve case info on.
- **tenant_id** (*str*, *optional*) – The identifier of the Customer tenant the case is in. Defaults to None (the tenant_id of SDK authorization user will be used).

Returns A response containing the Departing Employee case.

Return type `py42.response.Py42Response`

resolve (*case_id*, *tenant_id=None*)

Removes a user from Departing Employees. [REST Documentation](#)

Parameters

- **case_id** (*str*) – The identifier of the Departing Employee.
- **tenant_id** (*str*, *optional*) – The identifier of the Customer tenant the case is in. Defaults to None (the tenant_id of SDK authorization user will be used).

Returns `py42.response.Py42Response`

toggle_alerts (*tenant_id=None*, *alerts_enabled=True*)

Enable or disable email alerting on Departing Employee exposure events. [REST Documentation](#)

Parameters

- **tenant_id** (*str*, *optional*) – The identifier of the Customer tenant. Defaults to None (the tenant_id of SDK authorization user will be used).
- **alerts_enabled** (*bool*) – Set alerting to on (True) or off (False). Defaults to True.

Returns `py42.response.Py42Response`

update (*case_id*, *tenant_id=None*, *display_name=None*, *notes=None*, *departure_epoch=None*, *cloud_usernames=None*)

Add or modify details of an existing Departing Employee case. [REST Documentation](#)

Parameters

- **case_id** (*str*) – Identifier of the Departing Employee case.
- **tenant_id** (*str*, *optional*) – The identifier of the Customer tenant the case is in. Defaults to None (the tenant_id of SDK authorization user will be used).
- **display_name** (*str*, *optional*) – The display name for the Departing Employee case. This defaults to username when adding a new Departing Employee, so it can be used to make the UI more user-friendly if your Organization has usernames that don't correspond to real names. Defaults to None.
- **notes** (*str*, *optional*) – Optional descriptive information. Defaults to None.
- **departure_epoch** (*int*, *optional*) – Employee departure date as POSIX timestamp. Defaults to None.
- **cloud_usernames** (*list*, *optional*) – List of alternate usernames for this user from external data sources being searched by Forensic File Search (Google Drive, Box, OneDrive, Gmail, Office 365). Defaults to None.

Returns `py42.response.Py42Response`

High Risk Employee

class `py42.clients.detectionlists.high_risk_employee.HighRiskEmployeeClient` (*session*, *user_context*, *de-tection_list_user_client*, *user_client*)

Bases: `py42.clients.BaseClient`

Administrator utility to manage High Risk employees.

Support Documentation

add (*user_id*)

Adds a user to high risk employee detection list.

A user profile in detection list will be created before adding to High Risk Employee list if it doesn't exist.

Returns failure when a user already exists in the high risk employee detection list.

Parameters **user_id** (*str or int*) – The Code42 `userId` of the user who needs to be added to HRE detection list.

Returns `py42.response.Py42Response`

get (*user_id*)

Get user information.

Parameters **user_id** (*str or int*) – The Code42 `userId` of the user has been added to the High Risk Employee detection list.

Returns `py42.response.Py42Response`

get_all (*filter_type='OPEN'*, *sort_key=None*, *sort_direction=None*)

Search High Risk employee list. Filter results by filter_type.

Parameters

- **filter_type** (*str*) – Valid filter types.
- **sort_key** (*str*) – Sort results based by field.
- **sort_direction** (*str*) – “ASC” or “DESC”

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of users.

Return type generator

remove (*user_id*)

Remove a user from high risk employee detection list.

Parameters **user_id** (*str or int*) – The Code42 `userId` of the user who needs to be added to HRE detection lists.

Returns `py42.response.Py42Response`

set_alerts_enabled (*enabled=True*)

Enable alerts.

Parameters **enabled** (*bool*) – Whether to enable alerts for all users

Returns `py42.response.Py42Response`

2.2.7 Alerts

class `py42.clients.alerts.AlertClient` (*session, user_context*)

Bases: `py42.clients.BaseClient`

A client for interacting with Code42 security alerts.

The `AlertClient` has the ability to search, resolve, and reopen alerts. Also, it can get the details for the file event query for the event that triggered the alert.

get_details (*alert_ids, tenant_id=None*)

Gets the details for the alerts with the given IDs, including the file event query that, when passed into a search, would result in events that could have triggered the alerts.

Parameters

- **alert_ids** (*iter[str]*) – The IDs of the alerts for which you want to get details for.
- **tenant_id** (*str, optional*) – The tenant ID for the tenant that the alerts belong to. When given `None`, it uses the currently logged in user’s tenant ID. Defaults to `None`.

Returns A response containing the alert details.

Return type `py42.response.Py42Response`

reopen (*alert_ids, tenant_id=None, reason=None*)

Reopens the resolved alerts with the given IDs.

Parameters

- **alert_ids** (*iter[str]*) – The IDs for the alerts to reopen.
- **tenant_id** (*str, optional*) – The ID for the tenant that the alerts belong to. When given `None`, it uses the currently logged in user’s tenant ID. Defaults to `None`.

- **reason** (*str*, *optional*) – The reason the alerts are reopened. Defaults to None.

Returns `py42.response.Py42Response`

```
resolve (alert_ids, tenant_id=None, reason=None)
```

Resolves the alerts with the given IDs.

Parameters

- **alert_ids** (*iter[*str*]*) – The IDs for the alerts to resolve.
- **tenant_id** (*str, optional*) – The ID for the tenant that the alerts belong to. When given None, it uses the currently logged in user’s tenant ID. Defaults to None.
- **reason** (*str, optional*) – The reason the alerts are now resolved. Defaults to None.

Returns `py42.response.Py42Response`

search (*query*)

Searches alerts using the given `py42.sdk.queries.alerts.alert_query.AlertQuery`.

Parameters `query` (py42.sdk.queries.alerts.alert_query.AlertQuery)
– An alert query. See *userguides.searches* to learn more about how to construct a query.

Returns A response containing the alerts that match the given query.

Return type `py42.response.Py42Response`

2.2.8 File Event Queries

```
class py42.clients.file_event.FileEventClient (session)
```

Bases: `py42.clients.BaseClient`

A client for querying Code42 Forensic Search events.

search (*query*)

Searches for file events matching query criteria. [REST Documentation](#)

Parameters `query` (`FileEventQuery` or `str`) – A composed `FileEventQuery` object or the raw query as a JSON formatted string.

Returns A response containing the query results.

Return type `py42.response.Py42Response`

[illegible]

Bases: `py42.sdk.queries.BaseQuery`

Helper class for building Code42 Forensic Search queries.

A `FileEventQuery` instance's `all()` and `any()` take one or more `FilterGroup` objects to construct a query that can be passed to the `FileEventClient.search()` method. `all()` returns results that match all of the provided filter criteria, `any()` will return results that match any of the filters.

For convenience, the `FileEventQuery` constructor does the same as `all()`.

Usage example:

```
email_filter = EmailSender.is_in(["test.user@example.com", "test.sender@example.
↪com"])
exposure_filter = ExposureType.exists()

query = FileEventQuery.all(email_filter, exposure_filter)
```

2.2.9 Filter Classes

The following classes construct filters for Forensic Search queries. Each filter class corresponds to a Forensic Search event detail. Call the appropriate classmethod on your desired filter class with the value you want to match and it will return a `FilterGroup` object that can be passed to `FileEventQuery`'s `all()` or `any()` methods to create complex queries that match multiple filter rules.

Example:

To search for events concerning a certain set of documents, you can use the `FileName` and `MD5` filter classes to construct `FilterGroups` that will search for matching filenames or (in case someone renamed the sensitive file) the known MD5 hashes of the files:

```
filename_filter = FileName.is_in(['confidential_plans.docx', 'confidential_plan_
↪projections.xlsx'])
md5_filter = MD5.is_in(['133765f4fff5e3038b9352a4d14e1532',
↪'eal6f0cbfc76f6eba292871f8a8c794b'])
```

See *Executing Searches* for more on crafting search queries.

Event Filters

class `py42.sdk.queries.fileevents.filters.event_filter.EventTimestamp`

Bases: `py42.sdk.queries.query_filter.QueryFilterTimestampField`

Class that filters events based on the timestamp of the event that occurred.

classmethod `in_range(start_value, end_value)`

Returns a `FilterGroup` to find events where the filter timestamp is in range between provided `start_value` and `end_value`.

classmethod `on_or_after(value)`

Returns a `FilterGroup` to find events where the filter timestamp is on or after provided `value`.

classmethod `on_or_before(value)`

Returns a `FilterGroup` to find events where the filter timestamp is on or before provided `value`.

classmethod `on_same_day(value)`

Returns a `FilterGroup` to find events where the filter timestamp is within the same calendar day as provided `value`.

class `py42.sdk.queries.fileevents.filters.event_filter.EventType`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters file events based on event type.

Available event types are provided as class attributes:

- `EventType.CREATED`
- `EventType.DELETED`
- `EventType EMAILED`

- `EventType.MODIFIED`
- `EventType.READ_BY_APP`

Example:

```
filter = EventType.isin([EventType.READ_BY_APP, EventType EMAILED])
```

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided value.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided value.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.event_filter.InsertionTimestamp`

Bases: `py42.sdk.queries.query_filter.QueryFilterTimestampField`

Class that filters events based on the timestamp of when the event was actually added to the event store (which can be after the event occurred on the device itself).

value must be a POSIX timestamp. (see the [Dates](#) section of the Basics userguide for details on timestamp arguments in py42)

classmethod `in_range(start_value, end_value)`

Returns a `FilterGroup` to find events where the filter timestamp is in range between provided *start_value* and *end_value*.

classmethod `on_or_after(value)`

Returns a `FilterGroup` to find events where the filter timestamp is on or after provided *value*.

classmethod `on_or_before(value)`

Returns a `FilterGroup` to find events where the filter timestamp is on or before provided *value*.

classmethod `on_same_day(value)`

Returns a `FilterGroup` to find events where the filter timestamp is within the same calendar day as provided *value*.

class `py42.sdk.queries.fileevents.filters.event_filter.Source`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided value.

Parameters `value` (*str*) – The value to match file events on.

```

classmethod exists()
    Returns a FilterGroup to find events where filter data exists.

classmethod is_in(value_list)
    Returns a FilterGroup to find events where the filter is in provided value_list.

    Parameters value_list (list) – The list of values to match file events on.

classmethod not_eq(value)
    Returns a FilterGroup to find events where the filter is not equal to provided value.

    Parameters value (str) – The value to exclude file events on.

classmethod not_exists()
    Returns a FilterGroup to find events where filter data does not exist.

classmethod not_in(value_list)
    Returns a FilterGroup to find events where the filter is not in provided value_list.

    Parameters value_list (list) – The list of values to exclude file events on.

```

File Filters

```

class py42.sdk.queries.fileevents.filters.file_filter.FileCategory
    Bases: py42.sdk.queries.query_filter.QueryFilterStringField
    Class that filters events by category of the file observed.

    classmethod eq(value)
        Returns a FilterGroup to find events where the filter equals provided value.

        Parameters value (str) – The value to match file events on.

    classmethod is_in(value_list)
        Returns a FilterGroup to find events where the filter is in provided value_list.

        Parameters value_list (list) – The list of values to match file events on.

    classmethod not_eq(value)
        Returns a FilterGroup to find events where the filter is not equal to provided value.

        Parameters value (str) – The value to exclude file events on.

    classmethod not_in(value_list)
        Returns a FilterGroup to find events where the filter is not in provided value_list.

        Parameters value_list (list) – The list of values to exclude file events on.

class py42.sdk.queries.fileevents.filters.file_filter.FileName
    Bases: py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField
    Class that filters events by the name of the file observed.

    classmethod eq(value)
        Returns a FilterGroup to find events where the filter equals provided value.

        Parameters value (str) – The value to match file events on.

    classmethod exists()
        Returns a FilterGroup to find events where filter data exists.

    classmethod is_in(value_list)
        Returns a FilterGroup to find events where the filter is in provided value_list.

        Parameters value_list (list) – The list of values to match file events on.

```

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.file_filter.FileOwner`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by the owner of the file observed.

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.file_filter.FilePath`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by path of the file observed.

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters *value* (*str*) – The value to exclude file events on.

```

classmethod not_exists()
    Returns a FilterGroup to find events where filter data does not exist.

classmethod not_in(value_list)
    Returns a FilterGroup to find events where the filter is not in provided value_list.

    Parameters value_list (list) – The list of values to exclude file events on.

class py42.sdk.queries.fileevents.filters.file_filter.FileSize
    Bases: py42.sdk.queries.fileevents.file_event_query.FileEventFilterComparableField
    Class that filters events by size of the file observed.

    Size value must be bytes.

classmethod greater_than(value)
    Returns a FilterGroup to find events where filter data is greater than the provided value.

classmethod less_than(value)
    Returns a FilterGroup to find events where filter data is less than than the provided value.

class py42.sdk.queries.fileevents.filters.file_filter.MD5
    Bases: py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField
    Class that filters events by the MD5 hash of the file observed.

classmethod eq(value)
    Returns a FilterGroup to find events where the filter equals provided value.

    Parameters value (str) – The value to match file events on.

classmethod exists()
    Returns a FilterGroup to find events where filter data exists.

classmethod is_in(value_list)
    Returns a FilterGroup to find events where the filter is in provided value_list.

    Parameters value_list (list) – The list of values to match file events on.

classmethod not_eq(value)
    Returns a FilterGroup to find events where the filter is not equal to provided value.

    Parameters value (str) – The value to exclude file events on.

classmethod not_exists()
    Returns a FilterGroup to find events where filter data does not exist.

classmethod not_in(value_list)
    Returns a FilterGroup to find events where the filter is not in provided value_list.

    Parameters value_list (list) – The list of values to exclude file events on.

class py42.sdk.queries.fileevents.filters.file_filter.SHA256
    Bases: py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField
    Class that filters events by SHA256 hash of the file observed.

classmethod eq(value)
    Returns a FilterGroup to find events where the filter equals provided value.

    Parameters value (str) – The value to match file events on.

classmethod exists()
    Returns a FilterGroup to find events where filter data exists.

```

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

Device Filters

class `py42.sdk.queries.fileevents.filters.device_filter.DeviceUsername`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by Code42 username of the device that observed the event.

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.device_filter.OSHostname`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by Hostname of the device that observed the event.

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.device_filter.PrivateIPAddress`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by private (LAN) IP address of the device that observed the event.

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.device_filter.PublicIPAddress`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by public (WAN) IP address of the device that observed the event.

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

Cloud Filters

class `py42.sdk.queries.fileevents.filters.cloud_filter.Actor`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by the cloud service username of the event originator (applies to cloud data source events only).

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.cloud_filter.DirectoryID`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by unique identifier of the cloud drive or folder where the event occurred (applies to cloud data source events only).

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq` (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided value.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists` ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in` (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.cloud_filter.Shared`

Bases: `py42.sdk.queries.query_filter.QueryFilterBooleanField`

Class that filters events by the shared status of the file at the time the event occurred (applies to cloud data source events only).

classmethod `is_false` ()

Returns a `FilterGroup` to find events where the filter state is False.

classmethod `is_true` ()

Returns a `FilterGroup` to find events where the filter state is True.

class `py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events by the list of users who had been granted access to the file at the time of the event (applies to cloud data source events only).

classmethod `eq` (*value*)

Returns a `FilterGroup` to find events where the filter equals provided value.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists` ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in` (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq` (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided value.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists` ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in` (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.cloud_filter.SharingTypeAdded`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters results to include events where a file's sharing permissions were changed to a value that increases exposure (applies to cloud data source events only).

Available options provided as class attributes:

- `SharingTypeAdded.SHARED_VIA_LINK`
- `SharingTypeAdded.IS_PUBLIC`
- `SharingTypeAdded.OUTSIDE_TRUSTED_DOMAIN`

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided value.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided value.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

Exposure Filters

class `py42.sdk.queries.fileevents.filters.exposure_filter.ExposureType`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on exposure type.

Available options are provided as class attributes:

- `ExposureType.SHARED_VIA_LINK`
- `ExposureType.SHARED_TO_DOMAIN`
- `ExposureType.APPLICATION_READ`
- `ExposureType.CLOUD_STORAGE`
- `ExposureType.REMOVABLE_MEDIA`
- `ExposureType.IS_PUBLIC`

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided value.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.ProcessName`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the process name involved in the exposure (applies to read by browser or other app events only).

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.ProcessOwner`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the process owner that was involved in the exposure (applies to read by browser or other app events only).

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters *value_list* (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters value (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters value_list (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaName`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the name of the removable media involved in the exposure (applies to removable media events only).

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters value (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters value_list (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters value (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters value_list (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaVendor`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the vendor of the removable media involved in the exposure (applies to removable media events only).

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters value (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters value_list (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters value (*str*) – The value to exclude file events on.

classmethod not_exists()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in(value_list)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters value_list (list) – The list of values to exclude file events on.

class py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaMediaName

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the name of the removable media (as reported by the vendor/device, usually very similar to `RemovableMediaName`) involved in the exposure (applies to `removable media` events only).

classmethod eq(value)

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters value (str) – The value to match file events on.

classmethod exists()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in(value_list)

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters value_list (list) – The list of values to match file events on.

classmethod not_eq(value)

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters value (str) – The value to exclude file events on.

classmethod not_exists()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in(value_list)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters value_list (list) – The list of values to exclude file events on.

class py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaVolumeName

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the name of the formatted volume (as reported by the operating system) of the removable media involved in the exposure (applies to `removable media` events only).

classmethod eq(value)

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters value (str) – The value to match file events on.

classmethod exists()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in(value_list)

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters value_list (list) – The list of values to match file events on.

classmethod not_eq(value)

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters value (str) – The value to exclude file events on.

classmethod not_exists()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in(*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaPartitionID`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the unique identifier assigned (by the operating system) to the removable media involved in the exposure (applies to removable media events only).

classmethod eq(*value*)

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in(*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters *value_list* (*list*) – The list of values to match file events on.

classmethod not_eq(*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in(*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters *value_list* (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaSerialNumber`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the serial number of the connected hardware as reported by the operating system (applies to removable media events only).

classmethod eq(*value*)

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters *value* (*str*) – The value to match file events on.

classmethod exists()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in(*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters *value_list* (*list*) – The list of values to match file events on.

classmethod not_eq(*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters *value* (*str*) – The value to exclude file events on.

classmethod not_exists()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.SyncDestination`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the name of the cloud service the file is synced with (applies to synced to cloud service events only).

Available options are provided as class attributes:

- `SyncDestination.ICLOUD`
- `SyncDestination.BOX`
- `SyncDestination.BOX_DRIVE`
- `SyncDestination.GOOGLE_DRIVE`
- `SyncDestination.GOOGLE_BACKUP_AND_SYNC`
- `SyncDestination.DROPBOX`
- `SyncDestination.ONEDRIVE`

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod is_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod not_eq (*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod not_exists ()

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod not_in (*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.TabURL`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the URL of the active browser tab at the time the file contents were read by the browser (applies to read by browser or other app events only).

classmethod eq (*value*)

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod exists ()

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.exposure_filter.WindowTitle`

Bases: `py42.sdk.queries.fileevents.file_event_query.FileEventFilterStringField`

Class that filters events based on the name of the browser tab or application window that was open when a browser or other app event occurred (applies to read by browser or other app events only).

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod `exists()`

Returns a `FilterGroup` to find events where filter data exists.

classmethod `is_in(value_list)`

Returns a `FilterGroup` to find events where the filter is in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to match file events on.

classmethod `not_eq(value)`

Returns a `FilterGroup` to find events where the filter is not equal to provided `value`.

Parameters `value` (*str*) – The value to exclude file events on.

classmethod `not_exists()`

Returns a `FilterGroup` to find events where filter data does not exist.

classmethod `not_in(value_list)`

Returns a `FilterGroup` to find events where the filter is not in provided `value_list`.

Parameters `value_list` (*list*) – The list of values to exclude file events on.

Email Filters

class `py42.sdk.queries.fileevents.filters.email_filter.EmailPolicyName`

Bases: `py42.sdk.queries.query_filter.QueryFilterStringField`

Class that filters events based on the email DLP policy that detected this file (applies to emails sent via Microsoft Office 365 only).

classmethod `eq(value)`

Returns a `FilterGroup` to find events where the filter equals provided `value`.

Parameters `value` (*str*) – The value to match file events on.

classmethod `is_in` (*value_list*)
Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters **`value_list`** (*list*) – The list of values to match file events on.

classmethod `not_eq` (*value*)
Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters **`value`** (*str*) – The value to exclude file events on.

classmethod `not_in` (*value_list*)
Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters **`value_list`** (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.email_filter.EmailSubject`
Bases: `py42.sdk.queries.query_filter.QueryFilterStringField`
Class that filters events based on the email's subject (applies to email events only).

classmethod `eq` (*value*)
Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters **`value`** (*str*) – The value to match file events on.

classmethod `is_in` (*value_list*)
Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters **`value_list`** (*list*) – The list of values to match file events on.

classmethod `not_eq` (*value*)
Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters **`value`** (*str*) – The value to exclude file events on.

classmethod `not_in` (*value_list*)
Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters **`value_list`** (*list*) – The list of values to exclude file events on.

class `py42.sdk.queries.fileevents.filters.email_filter.EmailRecipients`
Bases: `py42.sdk.queries.query_filter.QueryFilterStringField`
Class that filters events based on the email's recipient list (applies to email events only).

classmethod `eq` (*value*)
Returns a `FilterGroup` to find events where the filter equals provided *value*.

Parameters **`value`** (*str*) – The value to match file events on.

classmethod `is_in` (*value_list*)
Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters **`value_list`** (*list*) – The list of values to match file events on.

classmethod `not_eq` (*value*)
Returns a `FilterGroup` to find events where the filter is not equal to provided *value*.

Parameters **`value`** (*str*) – The value to exclude file events on.

classmethod `not_in` (*value_list*)
Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters **`value_list`** (*list*) – The list of values to exclude file events on.

class py42.sdk.queries.fileevents.filters.email_filter.**EmailSender**

Bases: py42.sdk.queries.query_filter.QueryFilterStringField

Class that filters events based on the email's sender (applies to email events only).

classmethod **eq**(*value*)

Returns a `FilterGroup` to find events where the filter equals provided value.

Parameters **value** (*str*) – The value to match file events on.

classmethod **is_in**(*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters **value_list** (*list*) – The list of values to match file events on.

classmethod **not_eq**(*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided value.

Parameters **value** (*str*) – The value to exclude file events on.

classmethod **not_in**(*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters **value_list** (*list*) – The list of values to exclude file events on.

class py42.sdk.queries.fileevents.filters.email_filter.**EmailFrom**

Bases: py42.sdk.queries.query_filter.QueryFilterStringField

Class that filters events based on the display name of the email's sender, as it appears in the "From:" field in the email (applies to email events only).

classmethod **eq**(*value*)

Returns a `FilterGroup` to find events where the filter equals provided value.

Parameters **value** (*str*) – The value to match file events on.

classmethod **is_in**(*value_list*)

Returns a `FilterGroup` to find events where the filter is in provided *value_list*.

Parameters **value_list** (*list*) – The list of values to match file events on.

classmethod **not_eq**(*value*)

Returns a `FilterGroup` to find events where the filter is not equal to provided value.

Parameters **value** (*str*) – The value to exclude file events on.

classmethod **not_in**(*value_list*)

Returns a `FilterGroup` to find events where the filter is not in provided *value_list*.

Parameters **value_list** (*list*) – The list of values to exclude file events on.

2.2.10 Archive

class py42.modules.archive.**ArchiveModule**(*archive_accessor_manager*, *archive_client*)

Bases: object

A module for getting information about archives on storage nodes along with functionality for streaming a file from backup.

get_all_device_restore_history(*days*, *device_id*)

Gets all restore jobs from the past given days for the device with the given ID. [REST Documentation](#)

Parameters

- **days** (*int*) – Number of days of restore history to retrieve.

- **device_id** (*int*) – The ID for the device to get restore history for.

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of restore history.

Return type generator

get_all_org_restore_history (*days, org_id*)

Gets all restore jobs from the past given days for the organization with the given ID. [REST Documentation](#)

Parameters

- **days** (*int*) – Number of days of restore history to retrieve.
- **org_id** (*int*) – The ID for the organization to get restore history for.

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of restore history.

Return type generator

get_all_user_restore_history (*days, user_id*)

Gets all restore jobs from the past given days for the user with the given ID. [REST Documentation](#)

Parameters

- **days** (*int*) – Number of days of restore history to retrieve.
- **user_id** (*int*) – The ID for the user to get restore history for.

Returns An object that iterates over `py42.response.Py42Response` objects that each contain a page of restore history.

Return type generator

get_backup_sets (*device_guid, destination_guid*)

Gets all backup set names/identifiers referring to single destination for a specific device. [Support Page](#)

Parameters

- **device_guid** (*str*) – The GUID for the device to get backup sets for.
- **destination_guid** (*str*) – The GUID for the destination containing the archive to get backup sets for.

Returns A response containing the backup sets.

Return type `py42.response.Py42Response`

stream_from_backup (*file_path, device_guid, destination_guid=None, archive_password=None, encryption_key=None*)

Streams a file from an archive to memory. [REST Documentation](#)

Parameters

- **file_path** (*str*) – The path to the file in your archive.
- **device_guid** (*str*) – The GUID for the device the file belongs to.
- **destination_guid** (*str, optional*) – The GUID for the destination that stores the backup of the file. If None, it will use the first destination GUID it finds for your device. 'destination_guid' may be useful if the file is missing from one of your destinations or if you want to optimize performance. Defaults to None.
- **archive_password** (*str, None*) – The password for archives that are protected with an additional password. This is only relevant to users with this level of archive security. Defaults to None.

- **encryption_key** (*str*, *None*) – A custom encryption key for decryption an archive's file contents, necessary for restoring files. This is only relevant to users with this level of archive security. Defaults to *None*.

Returns A response containing the streamed content.

Return type `py42.response.Py42Response`

Usage example:

```
stream_response = sdk.archive.stream_from_backup("/full/path/to/file.txt",
↪ "1234567890")
with open("/path/to/my/file", 'wb') as f:
    for chunk in stream_response.iter_content(chunk_size=128):
        if chunk:
            f.write(chunk)
```

2.2.11 Response

class `py42.response.Py42Response` (*requests_response*)

Bases: `object`

property `encoding`

The encoding used to decode the response text.

property `headers`

A case-insensitive dictionary of response headers.

iter_content (*chunk_size=1*, *decode_unicode=False*)

Iterates over the response data. When `stream=True` is set on the request, this avoids reading the content at once into memory for large responses.

Parameters

- **chunk_size** (*int*, *optional*) – The number of bytes it should read into memory. A value of *None* will function differently depending on the value of *stream*. `stream=True` will read data as it arrives in whatever size the chunks are received. If `stream=False`, data is returned as a single chunk. This is not necessarily the length of each item. Defaults to 1.
- **decode_unicode** (*bool*, *optional*) – If *True*, content will be decoded using the best available encoding based on the response. Defaults to *False*.

property `raw_text`

The `response.Response.text` property. It contains raw metadata that is not included in the `Py42Response.text` property.

property `status_code`

An integer code of the response HTTP Status, e.g. 404 or 200.

property `text`

The more useful parts of the HTTP response dumped into a dictionary.

property `url`

The final URL location of response.

2.2.12 Exceptions

exception `py42.exceptions.Py42ArchiveFileNotFoundError` (*device_guid, file_path*)
 Bases: `py42.exceptions.Py42Error`

An exception raised when a resource file is not found or the path is invalid.

exception `py42.exceptions.Py42BadRequestError` (*exception*)
 Bases: `py42.exceptions.Py42HTTPError`

A wrapper to represent an HTTP 400 error.

exception `py42.exceptions.Py42Error`
 Bases: `Exception`

A generic, Py42 custom base exception.

exception `py42.exceptions.Py42FeatureUnavailableError`
 Bases: `py42.exceptions.Py42Error`

An exception raised when a requested feature is not supported in your Code42 instance.

exception `py42.exceptions.Py42ForbiddenError` (*exception*)
 Bases: `py42.exceptions.Py42HTTPError`

A wrapper to represent an HTTP 403 error.

exception `py42.exceptions.Py42HTTPError` (*http_response*)
 Bases: `py42.exceptions.Py42Error`

A base custom class to manage all HTTP errors raised by an API endpoint.

property response

The response containing the HTTP error.

exception `py42.exceptions.Py42InternalServerError` (*exception*)
 Bases: `py42.exceptions.Py42HTTPError`

A wrapper to represent an HTTP 500 error.

exception `py42.exceptions.Py42NotFoundError` (*exception*)
 Bases: `py42.exceptions.Py42HTTPError`

A wrapper to represent an HTTP 404 error.

exception `py42.exceptions.Py42SecurityPlanConnectionError` (*error_message*)
 Bases: `py42.exceptions.Py42Error`

An exception raised when the user is not authorized to access the requested resource.

exception `py42.exceptions.Py42SessionInitializationError` (*exception*)
 Bases: `py42.exceptions.Py42Error`

An exception raised when a user session is invalid. A session might be invalid due to session timeout, invalid token, etc.

exception `py42.exceptions.Py42StorageSessionInitializationError` (*error_message*)
 Bases: `py42.exceptions.Py42Error`

An exception raised when the user is not authorized to initialize a storage session. This may occur when trying to restore a file or trying to get legacy security events.

exception `py42.exceptions.Py42UnauthorizedError` (*exception*)
 Bases: `py42.exceptions.Py42HTTPError`

A wrapper to represent an HTTP 401 error.

`py42.exceptions.raise_py42_error(raised_error)`

Raises the appropriate `py42.exceptions.Py42HttpError` based on the given error's response status code.

2.2.13 Util

`py42.util.convert_datetime_to_timestamp_str(date)`

Converts the given datetime to a formatted date str. The format matches strftime directives `%Y-%m-%dT%H:%M:%S.%f`.

Parameters `date(datetime)` –

Returns A str representing the given date. Example output looks like `'2020-03-25T15:29:04.465Z'`.

Return type (str)

`py42.util.convert_timestamp_to_str(timestamp)`

Converts the given POSIX timestamp to a date str. The format matches strftime directives `%Y-%m-%dT%H:%M:%S.%f`.

Parameters `timestamp(float)` – A POSIX timestamp.

Returns A str representing the given timestamp. Example output looks like `'2020-03-25T15:29:04.465Z'`.

Return type (str)

`py42.util.format_json(json_string)`

Converts a minified JSON str to a prettified JSON str.

Parameters `json_string(str)` – A str representing minified JSON.

Returns A str representing prettified JSON.

Return type (str)

`py42.util.print_response(response, label=None)`

Prints a `py42.response.Py42Response` as prettified JSON. If unable to load, it prints the given response.

Parameters

- **response** (`py42.response.Py42Response`) –
- **label** (`str, optional`) – A label at the beginning of the printed text. Defaults to None.

`py42.sdk.from_local_account(host_address, username, password)`

Creates a `SDKClient` object for accessing the Code42 REST APIs using the supplied credentials. Currently, only accounts created within the Code42 console or using the APIs (including py42) are supported. Username/passwords that are based on Active Directory, Okta, or other Identity providers cannot be used with this method.

Parameters

- **host_address** (`str`) – The domain name of the Code42 instance being authenticated to, e.g. `console.us.code42.com`
- **username** (`str`) – The username of the authenticating account.
- **password** (`str`) – The password of the authenticating account.

Returns `py42.sdk.SDKClient`

class `py42.sdk.SDKClient` (*sdk_dependencies*)

Bases: `object`

property archive

A collection of methods for accessing Code42 storage archives. Useful for doing web-restores or finding a file on an archive.

Returns `py42.modules.archive.ArchiveModule`

property detectionlists

A collection of properties each containing methods for managing specific detection lists, such as departing employees.

Returns `py42.modules.detectionlists.DetectionListsModule`

property devices

A collection of methods for retrieving or updating data about devices in the Code42 environment.

Returns `py42.clients.devices.DeviceClient`

classmethod `from_local_account` (*host_address, username, password*)

Creates a `SDKClient` object for accessing the Code42 REST APIs using the supplied credentials. Currently, only accounts created within the Code42 console or using the APIs (including py42) are supported. Username/passwords that are based on Active Directory, Okta, or other Identity providers cannot be used with this method.

Parameters

- **host_address** (*str*) – The domain name of the Code42 instance being authenticated to, e.g. `console.us.code42.com`
- **username** (*str*) – The username of the authenticating account.
- **password** (*str*) – The password of the authenticating account.

Returns `py42.sdk.SDKClient`

property legalhold

A collection of methods for retrieving and updating legal-hold matters, policies, and custodians.

Returns `py42.clients.legalhold.LegalHoldClient`

property orgs

A collection of methods for retrieving or updating data about organizations in the Code42 environment.

Returns `py42.clients.orgs.OrgClient`

property securitydata

A collection of methods and properties for getting security data such as:

- File events
- Alerts
- Security plan information

Returns `py42.modules.securitydata.SecurityModule`

property serveradmin

A collection of methods for getting server information for on-premise environments and tenant information for cloud environments.

Returns `py42.clients.administration.AdministrationClient`

property `usercontext`

A collection of methods related to getting information about the currently logged in user, such as the tenant ID.

Returns `py42.usercontext.UserContext`

property `users`

A collection of methods for retrieving or updating data about users in the Code42 environment.

Returns `py42.clients.users.UserClient`

PYTHON MODULE INDEX

p

`py42.exceptions`, [51](#)
`py42.sdk`, [52](#)
`py42.util`, [52](#)

INDEX

A

Actor (class in `py42.sdk.queries.fileevents.filters.cloud_filter`), 38
 add() (`py42.clients.detectionlists.high_risk_employee.HighRiskEmployeeClient` method), 28
 add_cloud_alias() (`py42.modules.detectionlists.DetectionListsModule` method), 25
 add_risk_tag() (`py42.modules.detectionlists.DetectionListsModule` method), 25
 add_to_matter() (`py42.clients.legalhold.LegalHoldClient` method), 22
 AlertClient (class in `py42.clients.alerts`), 29
 alerts() (`py42.modules.securitydata.SecurityModule` property), 20
 archive() (`py42.sdk.SDKClient` property), 53
 ArchiveModule (class in `py42.modules.archive`), 48

B

block() (`py42.clients.devices.DeviceClient` method), 18
 block() (`py42.clients.orgs.OrgClient` method), 14
 block() (`py42.clients.users.UserClient` method), 15

C

change_org_assignment() (`py42.clients.users.UserClient` method), 15
 convert_datetime_to_timestamp_str() (in module `py42.util`), 52
 convert_timestamp_to_str() (in module `py42.util`), 52
 create() (`py42.clients.detectionlists.departing_employee.DepartingEmployeeClient` method), 26
 create() (`py42.modules.detectionlists.DetectionListsModule` method), 25
 create_matter() (`py42.clients.legalhold.LegalHoldClient` method), 22
 create_org() (`py42.clients.orgs.OrgClient` method), 14
 create_policy() (`py42.clients.legalhold.LegalHoldClient` method), 23

create_user() (`py42.clients.users.UserClient` method), 15
 deactivate() (`py42.clients.devices.DeviceClient` method), 18
 deactivate() (`py42.clients.orgs.OrgClient` method), 14
 deactivate() (`py42.clients.users.UserClient` method), 16
 deactivate_matter() (`py42.clients.legalhold.LegalHoldClient` method), 23
 deauthorize() (`py42.clients.devices.DeviceClient` method), 18
 DepartingEmployeeClient (class in `py42.clients.detectionlists.departing_employee`), 26
 destination_guid() (`py42.modules.securitydata.PlanStorageInfo` property), 22
 detectionlists() (`py42.sdk.SDKClient` property), 53
 DetectionListsModule (class in `py42.modules.detectionlists`), 25
 DeviceClient (class in `py42.clients.devices`), 18
 devices() (`py42.sdk.SDKClient` property), 53
 DeviceUsername (class in `py42.sdk.queries.fileevents.filters.device_filter`), 36
 DirectoryID (class in `py42.sdk.queries.fileevents.filters.cloud_filter`), 38
 EmailFrom (class in `py42.sdk.queries.fileevents.filters.email_filter`), 48
 EmailPolicyName (class in `py42.sdk.queries.fileevents.filters.email_filter`), 46
 EmailRecipients (class in

```

py42.sdk.queries.fileevents.filters.email_filter), eq () (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaV
47 class method), 42
EmailSender (class in eq () (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaV
py42.sdk.queries.fileevents.filters.email_filter), class method), 43
47 eq () (py42.sdk.queries.fileevents.filters.exposure_filter.SyncDestination
EmailSubject (class in class method), 45
py42.sdk.queries.fileevents.filters.email_filter), eq () (py42.sdk.queries.fileevents.filters.exposure_filter.TabURL
47 class method), 45
encoding () (py42.response.Py42Response property), eq () (py42.sdk.queries.fileevents.filters.exposure_filter.WindowTitle
50 class method), 46
eq () (py42.sdk.queries.fileevents.filters.cloud_filter.Actor eq () (py42.sdk.queries.fileevents.filters.file_filter.FileCategory
class method), 38 class method), 33
eq () (py42.sdk.queries.fileevents.filters.cloud_filter.DirectoryID eq () (py42.sdk.queries.fileevents.filters.file_filter.FileName
class method), 38 class method), 33
eq () (py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith eq () (py42.sdk.queries.fileevents.filters.file_filter.FileOwner
class method), 39 class method), 34
eq () (py42.sdk.queries.fileevents.filters.cloud_filter.SharingTypeAdd eq () (py42.sdk.queries.fileevents.filters.file_filter.FilePath
class method), 40 class method), 34
eq () (py42.sdk.queries.fileevents.filters.device_filter.DeviceUsernam eq () (py42.sdk.queries.fileevents.filters.file_filter.MD5
class method), 36 class method), 35
eq () (py42.sdk.queries.fileevents.filters.device_filter.OSThostname eq () (py42.sdk.queries.fileevents.filters.file_filter.SHA256
class method), 36 class method), 35
eq () (py42.sdk.queries.fileevents.filters.device_filter.PrivateIPAdres timestamp (class in
class method), 37 py42.sdk.queries.fileevents.filters.event_filter),
eq () (py42.sdk.queries.fileevents.filters.device_filter.PublicIPAdres31
class method), 37 EventType (class in
eq () (py42.sdk.queries.fileevents.filters.email_filter.EmailFrom py42.sdk.queries.fileevents.filters.event_filter),
class method), 48 31
eq () (py42.sdk.queries.fileevents.filters.email_filter.EmailPolicyName) (py42.sdk.queries.fileevents.filters.cloud_filter.Actor
class method), 46 class method), 38
eq () (py42.sdk.queries.fileevents.filters.email_filter.EmailRecipients () (py42.sdk.queries.fileevents.filters.cloud_filter.DirectoryID
class method), 47 class method), 38
eq () (py42.sdk.queries.fileevents.filters.email_filter.EmailSender exists () (py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith
class method), 48 class method), 39
eq () (py42.sdk.queries.fileevents.filters.email_filter.EmailSubject exists () (py42.sdk.queries.fileevents.filters.cloud_filter.SharingTypeAdd
class method), 47 class method), 40
eq () (py42.sdk.queries.fileevents.filters.event_filter.EventType exists () (py42.sdk.queries.fileevents.filters.device_filter.DeviceUsernam
class method), 32 class method), 36
eq () (py42.sdk.queries.fileevents.filters.event_filter.Source exists () (py42.sdk.queries.fileevents.filters.device_filter.OSThostname
class method), 32 class method), 36
eq () (py42.sdk.queries.fileevents.filters.exposure_filter.ExposureType) (py42.sdk.queries.fileevents.filters.device_filter.PrivateIPAdres
class method), 40 class method), 37
eq () (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessName) (py42.sdk.queries.fileevents.filters.device_filter.PublicIPAdres
class method), 41 class method), 37
eq () (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessOwner) (py42.sdk.queries.fileevents.filters.event_filter.EventType
class method), 41 class method), 32
eq () (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaID) (py42.sdk.queries.fileevents.filters.event_filter.Source
class method), 43 class method), 32
eq () (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaName) (py42.sdk.queries.fileevents.filters.exposure_filter.ExposureType
class method), 42 class method), 40
eq () (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaPath) (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessName
class method), 44 class method), 41
eq () (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaSize) (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessOwner
class method), 44 class method), 41

```

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaMediaName
class method), 43

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaMediaName
class method), 42

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaPartitionID
class method), 44

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaSerialNumber
class method), 44

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaVendor
class method), 42

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaVolumeName
class method), 43

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.SyncDestination
class method), 45

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.TabURL
class method), 45

exists() (py42.sdk.queries.fileevents.filters.exposure_filter.WindowTitle
class method), 46

exists() (py42.sdk.queries.fileevents.filters.file_filter.FileName
class method), 33

exists() (py42.sdk.queries.fileevents.filters.file_filter.FileOwner
class method), 34

exists() (py42.sdk.queries.fileevents.filters.file_filter.FilePath
class method), 34

exists() (py42.sdk.queries.fileevents.filters.file_filter.MD5
class method), 35

exists() (py42.sdk.queries.fileevents.filters.file_filter.SHA256
class method), 35

ExposureType (class in py42.sdk.queries.fileevents.filters.exposure_filter),
40

F

FileCategory (class in py42.sdk.queries.fileevents.filters.file_filter), 33

FileEventClient (class in py42.clients.file_event), 30

FileEventQuery (class in py42.sdk.queries.fileevents.file_event_query),
30

FileName (class in py42.sdk.queries.fileevents.filters.file_filter), 33

FileOwner (class in py42.sdk.queries.fileevents.filters.file_filter), 34

FilePath (class in py42.sdk.queries.fileevents.filters.file_filter), 34

FileSize (class in py42.sdk.queries.fileevents.filters.file_filter), 35

format_json() (in module py42.util), 52

from_local_account() (in module py42.sdk), 52

from_local_account() (py42.sdk.SDKClient class
method), 53

get() (py42.clients.detectionlists.high_risk_employee.HighRiskEmployee
method), 28

get() (py42.modules.detectionlists.DetectionListsModule
method), 25

get_all() (py42.clients.detectionlists.departing_employee.DepartingEm
method), 28

get_all() (py42.clients.detectionlists.high_risk_employee.HighRiskEmp
method), 28

get_all() (py42.clients.devices.DeviceClient
method), 18

get_all() (py42.clients.orgs.OrgClient method), 14

get_all() (py42.clients.users.UserClient method), 16

get_all_device_restore_history()
(py42.modules.archive.ArchiveModule
method), 48

get_all_matter_custodians()
(py42.clients.legalhold.LegalHoldClient
method), 23

get_all_matters()
(py42.clients.legalhold.LegalHoldClient
method), 23

get_all_org_restore_history()
(py42.modules.archive.ArchiveModule
method), 49

get_all_plan_security_events()
(py42.modules.securitydata.SecurityModule
method), 20

get_all_user_restore_history()
(py42.modules.archive.ArchiveModule
method), 49

get_all_user_security_events()
(py42.modules.securitydata.SecurityModule
method), 21

get_backup_sets()
(py42.modules.archive.ArchiveModule
method), 49

get_by_guid() (py42.clients.devices.DeviceClient
method), 19

get_by_id() (py42.clients.detectionlists.departing_employee.Departing
method), 27

get_by_id() (py42.clients.devices.DeviceClient
method), 19

get_by_id() (py42.clients.orgs.OrgClient method),
14

get_by_id() (py42.clients.users.UserClient method),
16

get_by_id() (py42.modules.detectionlists.DetectionListsModule
method), 25

get_by_name() (py42.clients.orgs.OrgClient
method), 14

get_by_uid() (py42.clients.orgs.OrgClient method),
15

get_by_uid() (py42.clients.users.UserClient

method), 17
 get_by_username() (py42.clients.detectionlists.departing_employee.DepartingEmployeeClient method), 27
 get_by_username() (py42.clients.users.UserClient method), 17
 get_current() (py42.clients.orgs.OrgClient method), 15
 get_current() (py42.clients.users.UserClient method), 17
 get_current_tenant_id() (py42.usercontext.UserContext method), 17
 get_details() (py42.clients.alerts.AlertClient method), 29
 get_matter_by_uid() (py42.clients.legalhold.LegalHoldClient method), 24
 get_policy_by_uid() (py42.clients.legalhold.LegalHoldClient method), 24
 get_policy_list() (py42.clients.legalhold.LegalHoldClient method), 24
 get_security_plan_storage_info_list() (py42.modules.securitydata.SecurityModule method), 21
 get_settings() (py42.clients.devices.DeviceClient method), 19
 greater_than() (py42.sdk.queries.fileevents.filters.file_filter.FileFilter class method), 35

H

headers() (py42.response.Py42Response property), 50
 HighRiskEmployeeClient (class in py42.clients.detectionlists.high_risk_employee), 28

I

in_range() (py42.sdk.queries.fileevents.filters.event_filter.EventFilter class method), 31
 in_range() (py42.sdk.queries.fileevents.filters.event_filter.EventFilter class method), 32
 InsertionTimestamp (class in py42.sdk.queries.fileevents.filters.event_filter), 32
 is_false() (py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith class method), 39
 is_in() (py42.sdk.queries.fileevents.filters.cloud_filter.Actor class method), 38
 is_in() (py42.sdk.queries.fileevents.filters.cloud_filter.DirectoryID class method), 38
 is_in() (py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith class method), 39
 is_in() (py42.sdk.queries.fileevents.filters.cloud_filter.SharingTypeAdded class method), 40
 is_in() (py42.sdk.queries.fileevents.filters.device_filter.DeviceUsername class method), 36
 is_in() (py42.sdk.queries.fileevents.filters.device_filter.OSHostname class method), 36
 is_in() (py42.sdk.queries.fileevents.filters.device_filter.PrivateIPAddress class method), 37
 is_in() (py42.sdk.queries.fileevents.filters.device_filter.PublicIPAddress class method), 37
 is_in() (py42.sdk.queries.fileevents.filters.email_filter.EmailFrom class method), 48
 is_in() (py42.sdk.queries.fileevents.filters.email_filter.EmailPolicyName class method), 46
 is_in() (py42.sdk.queries.fileevents.filters.email_filter.EmailRecipients class method), 47
 is_in() (py42.sdk.queries.fileevents.filters.email_filter.EmailSender class method), 48
 is_in() (py42.sdk.queries.fileevents.filters.email_filter.EmailSubject class method), 47
 is_in() (py42.sdk.queries.fileevents.filters.event_filter.EventType class method), 32
 is_in() (py42.sdk.queries.fileevents.filters.event_filter.Source class method), 33
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.ExposureType class method), 40
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessName class method), 41
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessOwner class method), 41
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia class method), 43
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia class method), 42
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia class method), 44
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia class method), 44
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia class method), 42
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia class method), 43
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.SyncDestination class method), 45
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.TabURL class method), 45
 is_in() (py42.sdk.queries.fileevents.filters.exposure_filter.WindowTitle class method), 46
 is_in() (py42.sdk.queries.fileevents.filters.file_filter.FileCategory class method), 33
 is_in() (py42.sdk.queries.fileevents.filters.file_filter.FileName class method), 33

`is_in()` (`py42.sdk.queries.fileevents.filters.file_filter.FileOwner` class method), 34
`is_in()` (`py42.sdk.queries.fileevents.filters.file_filter.FilePath` class method), 34
`is_in()` (`py42.sdk.queries.fileevents.filters.file_filter.MD5` class method), 35
`is_in()` (`py42.sdk.queries.fileevents.filters.file_filter.SHA256` class method), 35
`is_true()` (`py42.sdk.queries.fileevents.filters.cloud_filter.Shared` class method), 39
`iter_content()` (`py42.response.Py42Response` class method), 50

L

`legalhold()` (`py42.sdk.SDKClient` property), 53
`LegalHoldClient` (class in `py42.clients.legalhold`), 22
`less_than()` (`py42.sdk.queries.fileevents.filters.file_filter.FileSize` class method), 35

M

`MD5` (class in `py42.sdk.queries.fileevents.filters.file_filter`), 35
`module`
 `py42.exceptions`, 51
 `py42.sdk`, 52
 `py42.util`, 52

N

`node_guid()` (`py42.modules.securitydata.PlanStorageInfo` property), 22
`not_eq()` (`py42.sdk.queries.fileevents.filters.cloud_filter.Actor` class method), 38
`not_eq()` (`py42.sdk.queries.fileevents.filters.cloud_filter.DirectoryID` class method), 39
`not_eq()` (`py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith` class method), 39
`not_eq()` (`py42.sdk.queries.fileevents.filters.cloud_filter.SharingTypeAddress` class method), 40
`not_eq()` (`py42.sdk.queries.fileevents.filters.device_filter.DeviceUsername` class method), 36
`not_eq()` (`py42.sdk.queries.fileevents.filters.device_filter.OSHostname` class method), 37
`not_eq()` (`py42.sdk.queries.fileevents.filters.device_filter.PrivateIPAddress` class method), 37
`not_eq()` (`py42.sdk.queries.fileevents.filters.device_filter.PublicIPAddress` class method), 37
`not_eq()` (`py42.sdk.queries.fileevents.filters.email_filter.EmailFrom` class method), 48
`not_eq()` (`py42.sdk.queries.fileevents.filters.email_filter.EmailPolicyName` class method), 47
`not_eq()` (`py42.sdk.queries.fileevents.filters.email_filter.EmailRecipients` class method), 47
`not_eq()` (`py42.sdk.queries.fileevents.filters.email_filter.EmailSender` class method), 48
`not_eq()` (`py42.sdk.queries.fileevents.filters.email_filter.EmailSubject` class method), 47
`not_eq()` (`py42.sdk.queries.fileevents.filters.event_filter.EventType` class method), 32
`not_eq()` (`py42.sdk.queries.fileevents.filters.event_filter.Source` class method), 33
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.ExposureType` class method), 40
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.ProcessName` class method), 41
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.ProcessOwner` class method), 41
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia` class method), 43
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia` class method), 42
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia` class method), 44
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia` class method), 44
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia` class method), 42
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia` class method), 43
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.SyncDestination` class method), 45
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.TabURL` class method), 46
`not_eq()` (`py42.sdk.queries.fileevents.filters.exposure_filter.WindowTitle` class method), 46
`not_eq()` (`py42.sdk.queries.fileevents.filters.file_filter.FileCategory` class method), 33
`not_eq()` (`py42.sdk.queries.fileevents.filters.file_filter.FileName` class method), 33
`not_eq()` (`py42.sdk.queries.fileevents.filters.file_filter.FileOwner` class method), 34
`not_eq()` (`py42.sdk.queries.fileevents.filters.file_filter.FilePath` class method), 34
`not_eq()` (`py42.sdk.queries.fileevents.filters.file_filter.MD5` class method), 35
`not_eq()` (`py42.sdk.queries.fileevents.filters.file_filter.SHA256` class method), 36
`not_exists()` (`py42.sdk.queries.fileevents.filters.cloud_filter.Actor` class method), 38
`not_exists()` (`py42.sdk.queries.fileevents.filters.cloud_filter.DirectoryID` class method), 39
`not_exists()` (`py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith` class method), 39
`not_exists()` (`py42.sdk.queries.fileevents.filters.cloud_filter.SharingType` class method), 40
`not_exists()` (`py42.sdk.queries.fileevents.filters.device_filter.DeviceUser` class method), 36

```

not_exists() (py42.sdk.queries.fileevents.filters.device_filter.OSHostname class method), 37
not_exists() (py42.sdk.queries.fileevents.filters.device_filter.PrivateIPAddresses class method), 37
not_exists() (py42.sdk.queries.fileevents.filters.device_filter.PublicIPAddresses class method), 38
not_exists() (py42.sdk.queries.fileevents.filters.event_filter.EventType class method), 32
not_exists() (py42.sdk.queries.fileevents.filters.event_filter.Source class method), 33
not_exists() (py42.sdk.queries.fileevents.filters.exposure_filter.ExposedType class method), 41
not_exists() (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessName class method), 41
not_exists() (py42.sdk.queries.fileevents.filters.exposure_filter.ProcessOwner class method), 42
not_exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMedia class method), 43
not_exists() (py42.sdk.queries.fileevents.filters.exposure_filter.RemovableMediaSyncDestination class method), 45
not_exists() (py42.sdk.queries.fileevents.filters.exposure_filter.TabURL class method), 46
not_exists() (py42.sdk.queries.fileevents.filters.exposure_filter.WindowTitle class method), 46
not_exists() (py42.sdk.queries.fileevents.filters.file_filter.FileName class method), 34
not_exists() (py42.sdk.queries.fileevents.filters.file_filter.FileOwner class method), 34
not_exists() (py42.sdk.queries.fileevents.filters.file_filter.FilePath class method), 34
not_exists() (py42.sdk.queries.fileevents.filters.file_filter.MD5 class method), 35
not_exists() (py42.sdk.queries.fileevents.filters.file_filter.SHA256 class method), 36
not_in() (py42.sdk.queries.fileevents.filters.cloud_filter.Anchor class method), 38
not_in() (py42.sdk.queries.fileevents.filters.cloud_filter.DirectoryID class method), 39
not_in() (py42.sdk.queries.fileevents.filters.cloud_filter.SharedWith class method), 39
not_in() (py42.sdk.queries.fileevents.filters.cloud_filter.SharingType class method), 40
not_in() (py42.sdk.queries.fileevents.filters.device_filter.DeviceID class method), 36

```

not_in() (py42.sdk.queries.fileevents.filters.file_filter.SHA256Response (class in py42.response), 50
 class method), 36

O

on_or_after() (py42.sdk.queries.fileevents.filters.event_filter.EventTimestamp
 class method), 31

on_or_after() (py42.sdk.queries.fileevents.filters.event_filter.InsertionTimestamp
 class method), 32

on_or_before() (py42.sdk.queries.fileevents.filters.event_filter.EventTimestamp
 class method), 31

on_or_before() (py42.sdk.queries.fileevents.filters.event_filter.InsertionTimestamp
 class method), 32

on_same_day() (py42.sdk.queries.fileevents.filters.event_filter.EventTimestamp (py42.clients.devices.DeviceClient
 class method), 31

on_same_day() (py42.sdk.queries.fileevents.filters.event_filter.InsertionTimestamp (py42.clients.orgs.OrgClient method),
 class method), 32

OrgClient (class in py42.clients.orgs), 14

orgs() (py42.sdk.SDKClient property), 53

OSHostname (class in py42.sdk.queries.fileevents.filters.device_filter),
 36

P

plan_uid() (py42.modules.securitydata.PlanStorageInfo
 property), 22

PlanStorageInfo (class in py42.modules.securitydata), 22

print_response() (in module py42.util), 52

PrivateIPAddress (class in py42.sdk.queries.fileevents.filters.device_filter),
 37

ProcessName (class in py42.sdk.queries.fileevents.filters.exposure_filter),
 41

ProcessOwner (class in py42.sdk.queries.fileevents.filters.exposure_filter),
 41

PublicIPAddress (class in py42.sdk.queries.fileevents.filters.device_filter),
 37

py42.exceptions
 module, 51

py42.sdk
 module, 52

py42.util
 module, 52

Py42ArchiveFileNotFoundError, 51

Py42BadRequestError, 51

Py42Error, 51

Py42FeatureUnavailableError, 51

Py42ForbiddenError, 51

Py42HTTPError, 51

Py42InternalServerError, 51

Py42NotFoundError, 51

Py42Response (class in py42.response), 50

Py42SecurityPlanConnectionError, 51

Py42SessionInitializationError, 51

Py42StorageSessionInitializationError,

Py42UnauthorizedError, 51

R

reactivate() (py42.clients.users.UserClient
 method), 17

reactivate_matter() (py42.clients.legalhold.LegalHoldClient
 method), 24

RemovableMediaMediaName (class in
 py42.sdk.queries.fileevents.filters.exposure_filter),
 43

RemovableMediaName (class in
 py42.sdk.queries.fileevents.filters.exposure_filter),
 42

RemovableMediaPartitionID (class in
 py42.sdk.queries.fileevents.filters.exposure_filter),
 44

RemovableMediaSerialNumber (class in
 py42.sdk.queries.fileevents.filters.exposure_filter),
 44

RemovableMediaVendor (class in
 py42.sdk.queries.fileevents.filters.exposure_filter),
 42

RemovableMediaVolumeName (class in
 py42.sdk.queries.fileevents.filters.exposure_filter),
 43

remove() (py42.clients.detectionlists.high_risk_employee.HighRiskEmployee
 method), 29

remove_cloud_alias() (py42.modules.detectionlists.DetectionListsModule
 method), 25

remove_from_matter() (py42.clients.legalhold.LegalHoldClient
 method), 24

remove_risk_tag() (py42.modules.detectionlists.DetectionListsModule
 method), 25

reopen() (py42.clients.alerts.AlertClient method), 29

resolve() (py42.clients.alerts.AlertClient method), 30

resolve() (py42.clients.detectionlists.departing_employee.DepartingEmployee
 method), 27

`response()` (*py42.exceptions.Py42HTTPError* property), 51

S

`SDKClient` (class in *py42.sdk*), 52

`search()` (*py42.clients.alerts.AlertClient* method), 30

`search()` (*py42.clients.file_event.FileEventClient* method), 30

`search_file_events()` (*py42.modules.securitydata.SecurityModule* method), 22

`securitydata()` (*py42.sdk.SDKClient* property), 53

`SecurityModule` (class in *py42.modules.securitydata*), 20

`serveradmin()` (*py42.sdk.SDKClient* property), 53

`set_alerts_enabled()` (*py42.clients.detectionlists.high_risk_employee.HighRiskEmployeeClient* method), 29

`SHA256` (class in *py42.sdk.queries.fileevents.filters.file_filter*), 35

`Shared` (class in *py42.sdk.queries.fileevents.filters.cloud_filter*), 39

`SharedWith` (class in *py42.sdk.queries.fileevents.filters.cloud_filter*), 39

`SharingTypeAdded` (class in *py42.sdk.queries.fileevents.filters.cloud_filter*), 39

`Source` (class in *py42.sdk.queries.fileevents.filters.event_filter*), 32

`status_code()` (*py42.response.Py42Response* property), 50

`stream_from_backup()` (*py42.modules.archive.ArchiveModule* method), 49

`SyncDestination` (class in *py42.sdk.queries.fileevents.filters.exposure_filter*), 45

T

`TabURL` (class in *py42.sdk.queries.fileevents.filters.exposure_filter*), 45

`text()` (*py42.response.Py42Response* property), 50

`toggle_alerts()` (*py42.clients.detectionlists.departing_employee.DepartingEmployeeClient* method), 27

U

`unlock()` (*py42.clients.devices.DeviceClient* method), 20

`unlock()` (*py42.clients.orgs.OrgClient* method), 15

`unlock()` (*py42.clients.users.UserClient* method), 17

`update()` (*py42.clients.detectionlists.departing_employee.DepartingEmployeeClient* method), 27

`update_notes()` (*py42.modules.detectionlists.DetectionListsModule* method), 26

`url()` (*py42.response.Py42Response* property), 50

`UserClient` (class in *py42.clients.users*), 15

`UserContext` (class in *py42.usercontext*), 17

`usercontext()` (*py42.sdk.SDKClient* property), 53

`users()` (*py42.sdk.SDKClient* property), 54

W

`WindowTitle` (class in *py42.sdk.queries.fileevents.filters.exposure_filter*), 46