DFRWS2023 HYBRID USA

JULY 9–12

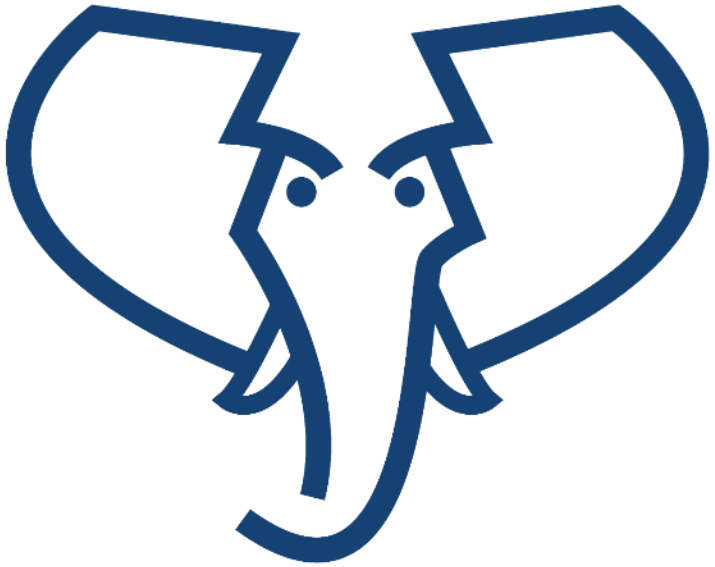
Baltimore, MD

in cooperation with

JOHNS HOPKINS UNIVERSITY

"HUMAN INGENUITY CANNOT CONCOCT A CIPHER WHICH HUMAN INGENUITY CANNOT RESOLVE."

— EDGAR ALLEN POE



Hansken

The open digital forensic platform
investigate - innovate - share

Providing DFaaS with code notebooks

DFRWS USA 2023 Workshop

Harm van Beek

Netherlands Forensic Institute
Ministry of Justice and Security

Quintin Walters

MITRE

Mattijs Ugen

Netherlands Forensic Institute
Ministry of Justice and Security

Agenda

Time	Title
09:00	**Part I**
	Introduction
	The Hansken SDK
	Live demonstration of Hansken and the Hansken SDK
10:45	Preparing for the exercises
11:00	Break
11:15	**Part II**
	Excercises
13:00	End

Agenda

Introduction

- Digital Forensics as a Service
- Hansken's main characteristics
- Hansken interfaces
- Hansken Community

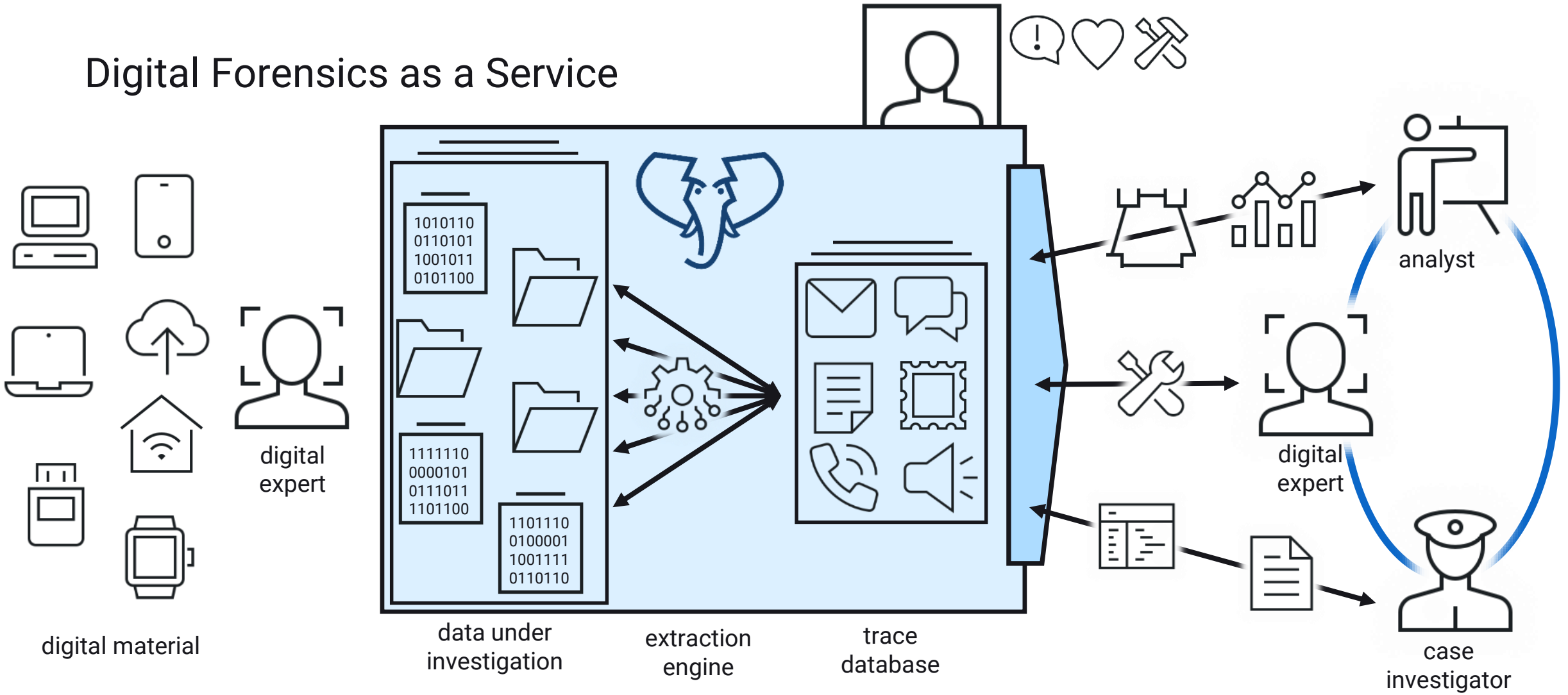
The Hansken SDK

- Trace Model
- Query Language
- REST API
- Python API
- Code notebooks in the Expert UI

Introduction

Hansken in a nutshell

Digital Forensics as a Service



Hansken's main characteristics

Governmental

- developed and maintained by Netherlands Forensic Institute √
- Steering committee by Dutch governmental partners √

Transparent

- all evidence is traceable (chain of evidence) √
- advanced logging of embedded data processors √

Scalable

- > 2000 cases, > 100 concurrent cases √
- cases with > 1,000 seized devices √
- cases with > 700 terabytes of data √
- cases with > 1,000,000,000 traces √

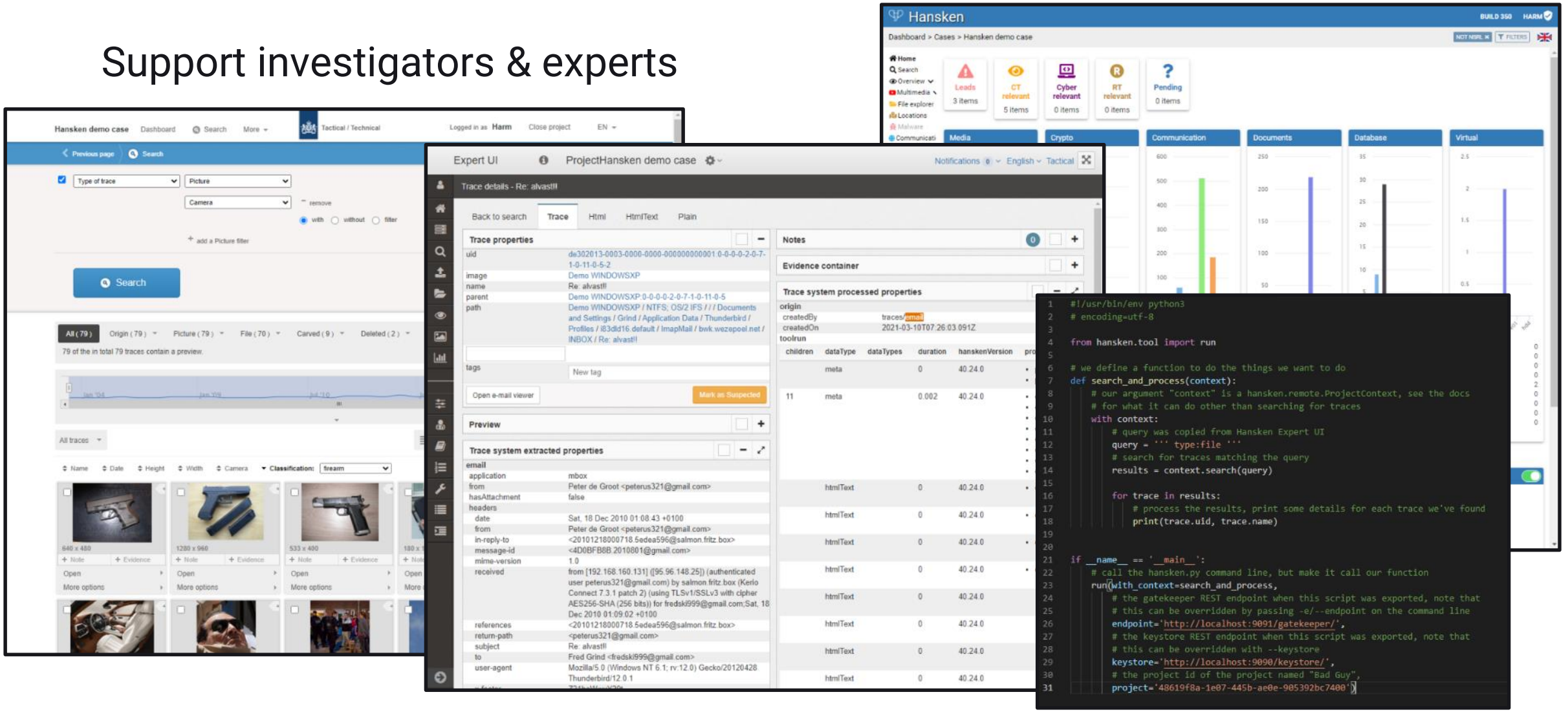
Legal

- all handling of traces is auditable (chain of custody) √
- support for handling privileged communication √
- profound judicial review in Dutch courts √

Hansken

The open digital forensic platform
investigate - innovate - share

Support investigators & experts



Building the Hansken Community in the Netherlands….

Netherlands Forensic Institute
Ministry of Justice and Security

POLITIE

NETHERLANDS
PUBLIC PROSECUTION SERVICE

Ministry of the Interior and
Kingdom Relations

Netherlands Food and Consumer
Product Safety Authority
*Ministry of Agriculture,
Nature and Food Quality*

FIOD
Belastingdienst

Royal Netherlands Marechaussee

Netherlands Labour Authority
Ministry of Social Affairs and Employment

Human Environment and Transport
Inspectorate
*Ministry of Infrastructure
and Water Management*

NEDERLANDSE ORDE VAN ADVOCATEN

POLITIEACADEMIE

Justitiële ICT Organisatie
Ministerie van Justitie en Veiligheid

de Rechtspraak

Building the international Hansken Community….

The Hansken Academic Network

- Educating digital forensic students
- Experimentation platform for testing innovative DF techniques
- Develop data for testing and for learning
- User interface innovation
- Hansken R&D License agreement for academia
- Option to experiment with Hanksen in a research project with non-academic partners

NTNU
Norwegian University of Science and Technology

UNIL | Université de Lausanne

hogeschool Leiden

HÖGSKOLAN I HALMSTAD

AGH

Agenda

Introduction

- Digital Forensics as a Service
- Hansken's main characteristics
- Hansken interfaces
- Hansken Community

The Hansken SDK

- Trace Model
- Query Language
- REST API
- Python API
- Code notebooks in the Expert UI

The Hansken SDK

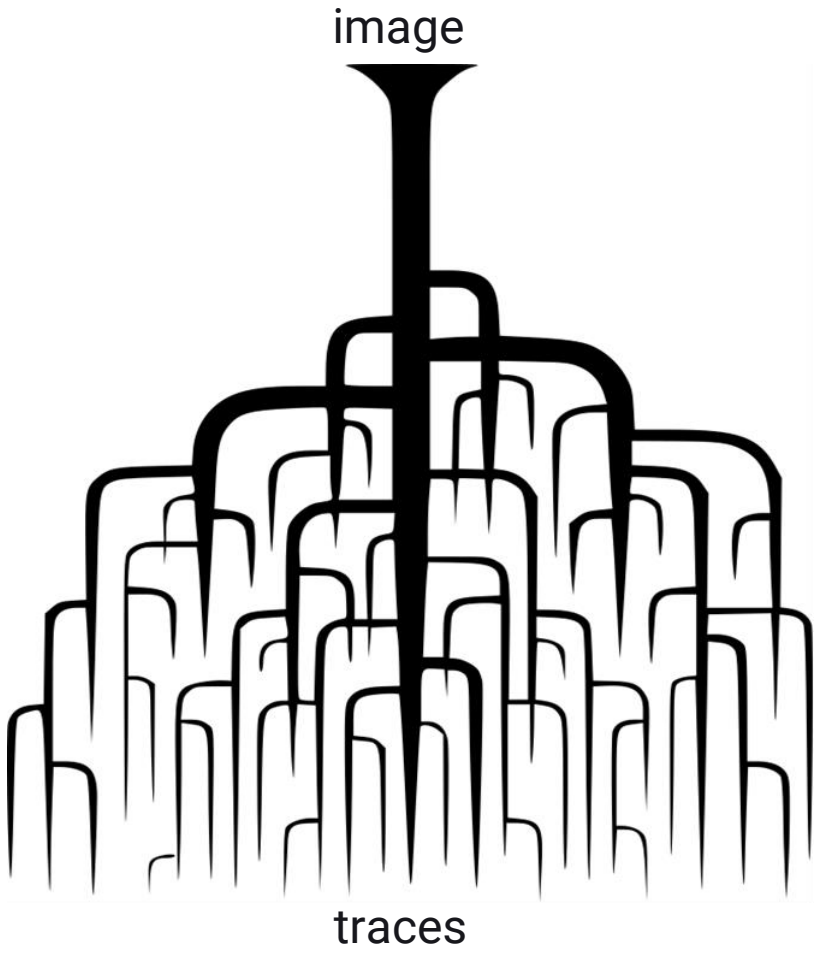
What is in the Hansken Software Development Kit?

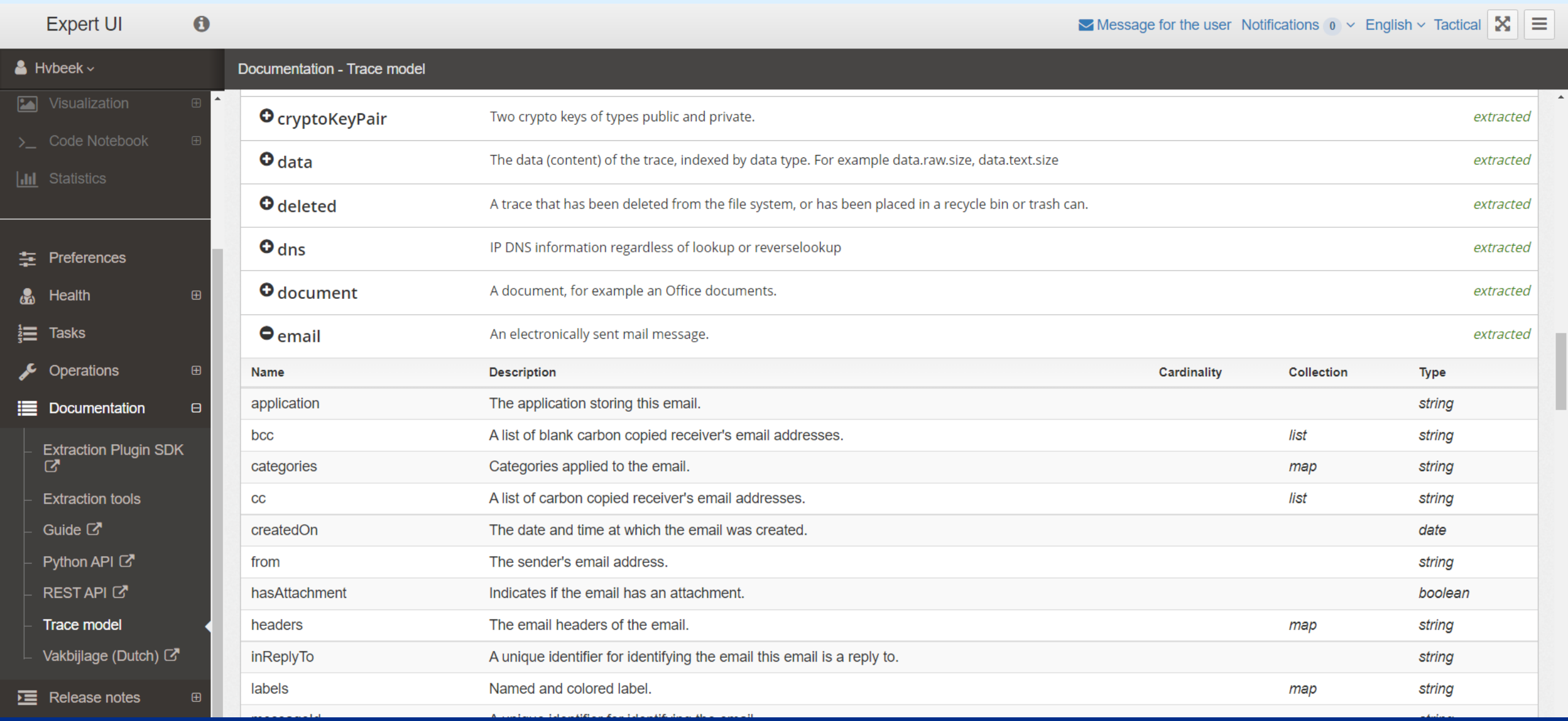
Trace Model

Uniform Trace Representation

Nested traces

- 🗁 /
  - 🗁 Program Files
  - 🗁 Users
    - 🗁 Jack
    - 🗁 John
    - 🗁 Peter
      - 🗁 Desktop
        - 🗋 manual.pdf
        - 🗐 private email.pst
          - 📪 outbox
          - 📪 inbox
            - ✉ invoice hotel
            - ✉ invitation for workshop
              - 🗋 route-to-nfi.pdf





cryptoKeyPair	Two crypto keys of types public and private.	*extracted*
data	The data (content) of the trace, indexed by data type. For example data.raw.size, data.text.size	*extracted*
deleted	A trace that has been deleted from the file system, or has been placed in a recycle bin or trash can.	*extracted*
dns	IP DNS information regardless of lookup or reverselookup	*extracted*
document	A document, for example an Office documents.	*extracted*
email	An electronically sent mail message.	*extracted*

Name	Description	Cardinality	Collection	Type
application	The application storing this email.			*string*
bcc	A list of blank carbon copied receiver's email addresses.		*list*	*string*
categories	Categories applied to the email.		*map*	*string*
cc	A list of carbon copied receiver's email addresses.		*list*	*string*
createdOn	The date and time at which the email was created.			*date*
from	The sender's email address.			*string*
hasAttachment	Indicates if the email has an attachment.			*boolean*
headers	The email headers of the email.		*map*	*string*
inReplyTo	A unique identifier for identifying the email this email is a reply to.			*string*
labels	Named and colored label.		*map*	*string*

Trace system extracted properties

data	
raw	
entropy	7.871242
fileType	Jpeg
hash	
md5	5bd060e10d6802fb3b856b4d67a0efe8
sha1	efff0971d1ed772e61811cf9de7829e260ea8450
sha256	d23bb2e09e6c703741cc2585a0c0ff5444998b6a2b595fb9f0351cc37ee20465
hashMisses	nsrl
mimeClass	picture
mimeType	image/jpeg
size	84744B (82.76 KiB)
file	
accessedOn	2022-02-21T10:15:49.475Z
changedOn	2022-02-21T10:15:49.237Z
createdOn	2022-02-21T10:15:23.201Z
modifiedOn	2022-02-21T10:15:23.201Z
name	IMG_0021.JPG
timestamps	
addedOn	2022-02-21T10:15:47.882Z
gps	
createdOn	2022-02-21T10:12:35.439Z
latlong	+52.16617+004.50893
picture	
aspectRatio	1.333
camera	iPhone 6s Plus
digitizedOn	2022-02-21T10:12:35.439Z
exif	
Aperture Value	f/2.2
Apple Makernote/Acceleration Vector	0.03g left, 0.11g up, 0.98g backward
Apple Makernote/Live Photo ID	0
Apple Makernote/Run Time	[104 values]
Apple Run Time/Epoch	0
Apple Run Time/Flags	Valid
Apple Run Time/Scale	1000000000
Apple Run Time/Value	1180 seconds

Hansken Query Language

Search and Find Relevant Traces

Hansken Query Language

Data model (terminology)

image
Hansken input data, can be any binary object like a copy of a disk, a phone extraction or a file archive; identified by a universally unique identifier (uuid)

trace
one artifact extracted from an image

type
type of a trace, like email, picture, document. One trace can have multiple types, like file and picture or document and attachment

tracelet
the properties and values of a single type within a trace.
Some types support multiple tracelets per trace, for example: toolrun, entity

property
property of a trace for a type, like email.subject, picture.camera, document.author. Types can have overlapping properties, like file.name and folder.name

value
the value of a property, supported types: string, long, double, date, latitude/longitude, set, map (with string keys)

data
special "type" for the contents of a trace, containing maps of properties with metadata of the contents

data.text | data.html | data.rtf | data.raw
special "type" containing metadata of the text | html | rtf | raw contained in the trace, like the textual contents of a PDF document or the html or rtf contents of an email or data.raw.size and data.raw.mimeType of the raw bytes

preview
base64-encoded previews per mime type, like preview.image/jpg for picture thumbnails

toolrun
special tracelet "type" containing details about the tools that were applied to the trace

Values <value>

hansken | bomb | 100px
single word; a series of at least 3 characters, including support for emailadresses, ipv4 addresses, etc.

"quick brown fox" | "lorem ipsum" | "patiënt zero"
phrase; a series of (tokenized) words between double quotes, can contain spaces and other breaking characters

'$bad*' | 're: sell me a bomb!' | '*😀' | '\u{1f600}*'
metadata; a series of characters representing a value in trace metadata (file name, email subject, ...)

1234 | -4321 | 12.34 | -43.21

```
<value> | data ( : | = ) <value> | meta ( : | = ) <value>
type ( : | = ) <type>
type:{ <expr> }
<property> ( : | = | != | >= | <= | < ) <value>
<property> : <value>..<value>
<expr> <expr> | <expr> (&& | AND) <expr>
<expr> (|| | OR) <expr>
-<expr> | NOT <expr>
( <expr> ) | <property>:$(<expr>)
```

Text queries

bomb | data:bomb | meta:bomb
traces containing bomb anywhere | in their data | in their metadata

b?mb | data:b?mb | meta:b?mb
traces with a word matching the wildcard expression, such as 'bomb', 'bumb' (? matches exactly one character) anywhere | in their data | in their metadata

bom* | data:bom* | meta:bom*
traces with a word starting with the characters "bom" like bomb, bomber, bombing, bom (* matches zero or more characters) anywhere | in their data | in their metadata

/regex/ | data:/regex/ | meta:/regex/ | /'regex'/
traces matching a regular expression anywhere | in their data | in their metadata | in their untokenized metadata.
Supported regex operators: |()[^].?+{}*

're: sell me a bomb!'
traces matching 're: sell me a bomb!' in their untokenized metadata, case insensitive and normalized (e hits ë, é, ê, ...). Supports wildcards **?** And *.

"sell bomb" | data:"sell bomb" | meta:"sell bomb"
traces that contain the sequence of words "sell bomb", tokenized (words of at least 3 characters), case insensitive and normalized (e hits ë, é, ê, ...), anywhere | in their data | in their metadata

"sell bomb"~3 | data:"sell bomb"~3 | meta:"sell bomb"~3
traces containing "sell bomb" or similar with the sequence of tokens up to three positions displaced, like "sell him this bomb" anywhere | in their data | in their metadata

Metadata queries

file: {} | type:file | type=file
all traces that are files

entity:{} | prediction:{}
all traces that have one or more entity | prediction tracelets

parent:'ade699ae-5bed-11e7-b730-6faa89b89afb:0-1' | parent:'*:0-4'
all traces having a parent with a matching identifier

file.name='$bad*'
files with their name starting with $bad, single quotes match exact metadata

email.sentOn:2020-06..2020-08
emails sent from June 2020 to August 2020 (inclusive)

document.lastPrintedOn:2020
documents with a last printed date in 2020

email.sentOn:2020-11
emails with a sent date in November 2020

file.deletedOn:2020-11-30
files with a delete date on 30 November 2015

gps.latlong:(38.5,110)..(41,-117)
traces with a GPS location in geo box with south-west corner (38 degrees 30 minutes North ,110 degrees East) and north-east corner (41 degrees North , 117 degrees West)

User added data queries

tags:*
all traces that are tagged

tags:important
all traces with a tag containing the word *important*

tags:'not interesting'
all traces with a tag containing *not interesting*, untokenized, case insensitive and ascii normalized (e hits ë, é, ê, ...)

#note:* | #note:{}
all traces having a note

#note:evidence
all traces having a note containing the word *evidence*

privileged:suspected | privileged:confirmed | privileged:rejected
all traces that are suspected | confirmed to be privileged communication (available to reviewers) | all traces that were suspected but rejected

privileged:*
all traces that have something to do with privileged communication, depending on your role you get all (reviewer), or only those that were suspected but rejected (investigator)

Combined queries

file.extension:txt data.size>1000 | file.extension:txt AND data.size>1000 | file.extension:txt && data.size>1000
files with extension txt and a size over 1000 bytes

type:picture type:attachment
pictures that are attachments

email.from:john@doe.com OR email.from:jane@doe.com | email.from:john@doe.com || email.from:jane@doe.com
emails with a sender of either john@doe.com or jane@doe.com

special "type" containing metadata of the text | html | rtf | raw contained in the trace, like the textual contents of a PDF document or the html or rtf contents of an email or data.raw.size and data.raw.mimeType of the raw bytes

preview
base64-encoded previews per mime type, like preview.image/jpg for picture thumbnails

toolrun
special tracelet "type" containing details about the tools that were applied to the trace

Values <value>

hansken | bomb | 100px
single word; a series of at least 3 characters, including support for emailadresses, ipv4 addresses, etc.

"quick brown fox" | "lorem ipsum" | "patiënt zero"
phrase; a series of (tokenized) words between double quotes, can contain spaces and other breaking characters

'$bad*' | 're: sell me a bomb!' | '*😀' | '\u{1f600}*'
metadata; a series of characters representing a value in trace metadata (file name, email subject, ...)

1234 | -4321 | 12.34 | -43.21
number; positive and negative integers (signed 64 bits) or floating point numbers (64 bits double)

2015 | 2015-11 | 2015-11-30T14 | 2015-11-13T14:30:12
date; a date with format YYYY-MM-DD**T**HH:MM:SS, prefixes indicate a range (inclusivity/exclusivity depends on the operator)

(+39.92117,+116.38300) | (40, -115) | (-1,2)
location; location represented by latitude (-90 to 90 degrees) and longitude (-180 to 180 degrees)

Properties <property>

data.raw.size | email.subject | document.author
a single property

data.size
short for data.raw.size or data.text.size or data.html.size or data.rtf.size

[email.from,email.to,email.cc] | [file.name,folder.name]
a list of properties

picture.exif.version | email.headers.delivered-to
property inside a map like picture.exif or email.headers

id
special properties containing the id of the trace, excluding image uuid

parent
special properties containing the id of the trace's parent, including image uuid

Expressions <expr>

Query (<expr>):
traces matching 're: sell me a bomb!' in their untokenized metadata, case insensitive and normalized (e hits ë, é, ê, ...). Supports wildcards **?** And *.

"sell bomb" | data:"sell bomb" | meta:"sell bomb"
traces that contain the sequence of words "sell bomb", tokenized (words of at least 3 characters), case insensitive and normalized (e hits ë, é, ê, ...), anywhere | in their data | in their metadata

"sell bomb"~3 | data:"sell bomb"~3 | meta:"sell bomb"~3
traces containing "sell bomb" or similar with the sequence of tokens up to three positions displaced, like "sell him this bomb" anywhere | in their data | in their metadata

Metadata queries

file: {} | type:file | type=file
all traces that are files

entity:{} | prediction:{}
all traces that have one or more entity | prediction tracelets

parent:'ade699ae-5bed-11e7-b730-6faa89b89afb:0-1' | parent:'*:0-4'
all traces having a parent with a matching identifier

file.name='$bad*'
files with their name starting with *$bad*. single quotes match exact metadata values, including special characters, with wildcard support

email.subject:'spam**'**
emails with **spam* in the subject, use \ to escape wildcard characters (*, ?, \)

picture.exif:* | picture.exif=*
pictures with exif information

email.bcc:* | email.bcc=*
emails that have a blind carbon copy (bcc) field

file.name:"example.txt" | file.name="example.txt"
files with file name equal to example.txt

[file.name,folder.name]:*a*
files and folders with the character *a* in its name

[email.from,email.to,email.cc]:john@doe.com
emails with either the *from*, *to* or *cc* set to john@doe.com

data.size:1000 | data.size=1000
traces with a size of 1000 bytes

-data.size:1000 | -data.size=1000 | data.size!=1000
traces with a size not equal to 1000

data.size>1024 | data.size>=1024
traces larger than 1024 bytes | traces of 1024 bytes or larger
similarely, **<** / **<=** can be used to query for smaller / smaller or equal values

data.size:10..20
traces between 10 and 20 bytes in size (inclusive)

#note:evidence
all traces having a note containing the word *evidence*

privileged:suspected | privileged:confirmed | privileged:rejected
all traces that are suspected | confirmed to be privileged communication (available to reviewers) | all traces that were suspected but rejected

privileged:*
all traces that have something to do with privileged communication, depending on your role you get all (reviewer), or only those that were suspected but rejected (investigator)

Combined queries

file.extension:txt data.size>1000 | file.extension:txt AND data.size>1000 | file.extension:txt && data.size>1000
files with extension txt and a size over 1000 bytes

type:picture type:attachment
pictures that are attachments

email.from:john@doe.com OR email.from:jane@doe.com | email.from:john@doe.com || email.from:jane@doe.com
emails with a sender of either john@doe.com or jane@doe.com

-email.to:john@doe.com | NOT email.to:john@doe.com
emails not having receiver john@doe.com

(email.from:john@doe.com OR email.from:jane@doe.com) email.sentOn>=2014-03-25 -email.subject:lawyer
emails sent by john@doe.com or jane@doe.com on or after March 25, 2014 that do not have the word *lawyer* in the subject

data.raw.hash.sha256:$(tag:suspect)
traces having the same SHA-256 hash value as those tagged with 'suspect'

type:file AND data.raw.hash.sha256:$(type:attachment)
files that also occur as attachment with the same SHA-256 hash value

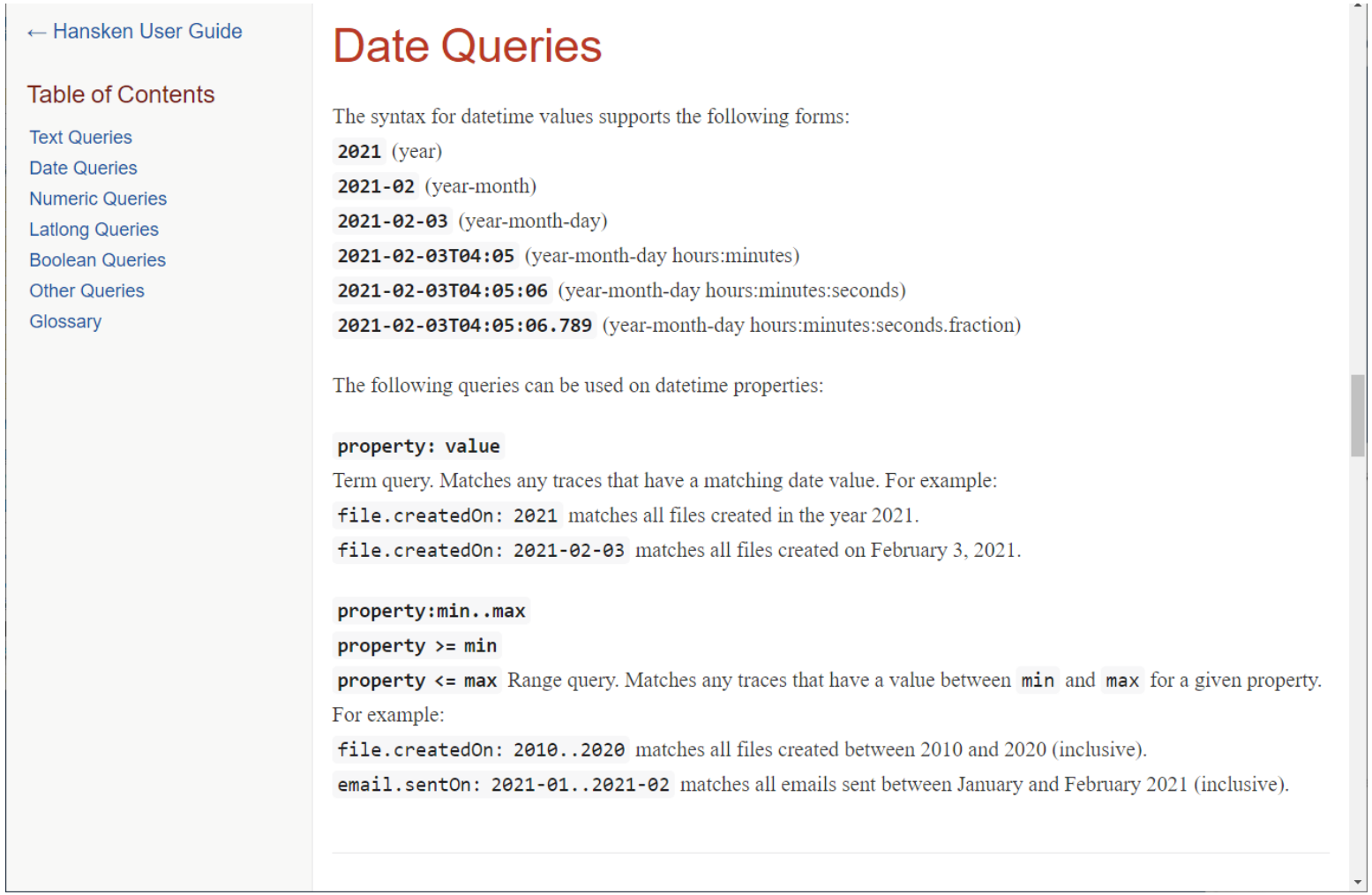
#note.user:user #note.createdOn:2019
traces with at least one note created by user and at least one note created in they year 2019, it does not have to be the same note

#note: { user:user createdOn:2019 }
traces with at least one note created by user in they year 2019

entity.type:iban trace: { type:email }
iban entities extracted from email traces (use in tracelet search of type entity)

```
file.createdOn:2021

dates:2021-10
```



← Hansken User Guide

Table of Contents

Date Queries

The syntax for datetime values supports the following forms:

`2021` (year)

`2021-02` (year-month)

`2021-02-03` (year-month-day)

`2021-02-03T04:05` (year-month-day hours:minutes)

`2021-02-03T04:05:06` (year-month-day hours:minutes:seconds)

`2021-02-03T04:05:06.789` (year-month-day hours:minutes:seconds.fraction)

The following queries can be used on datetime properties:

`property: value`

Term query. Matches any traces that have a matching date value. For example:

`file.createdOn: 2021` matches all files created in the year 2021.

`file.createdOn: 2021-02-03` matches all files created on February 3, 2021.

`property:min..max`

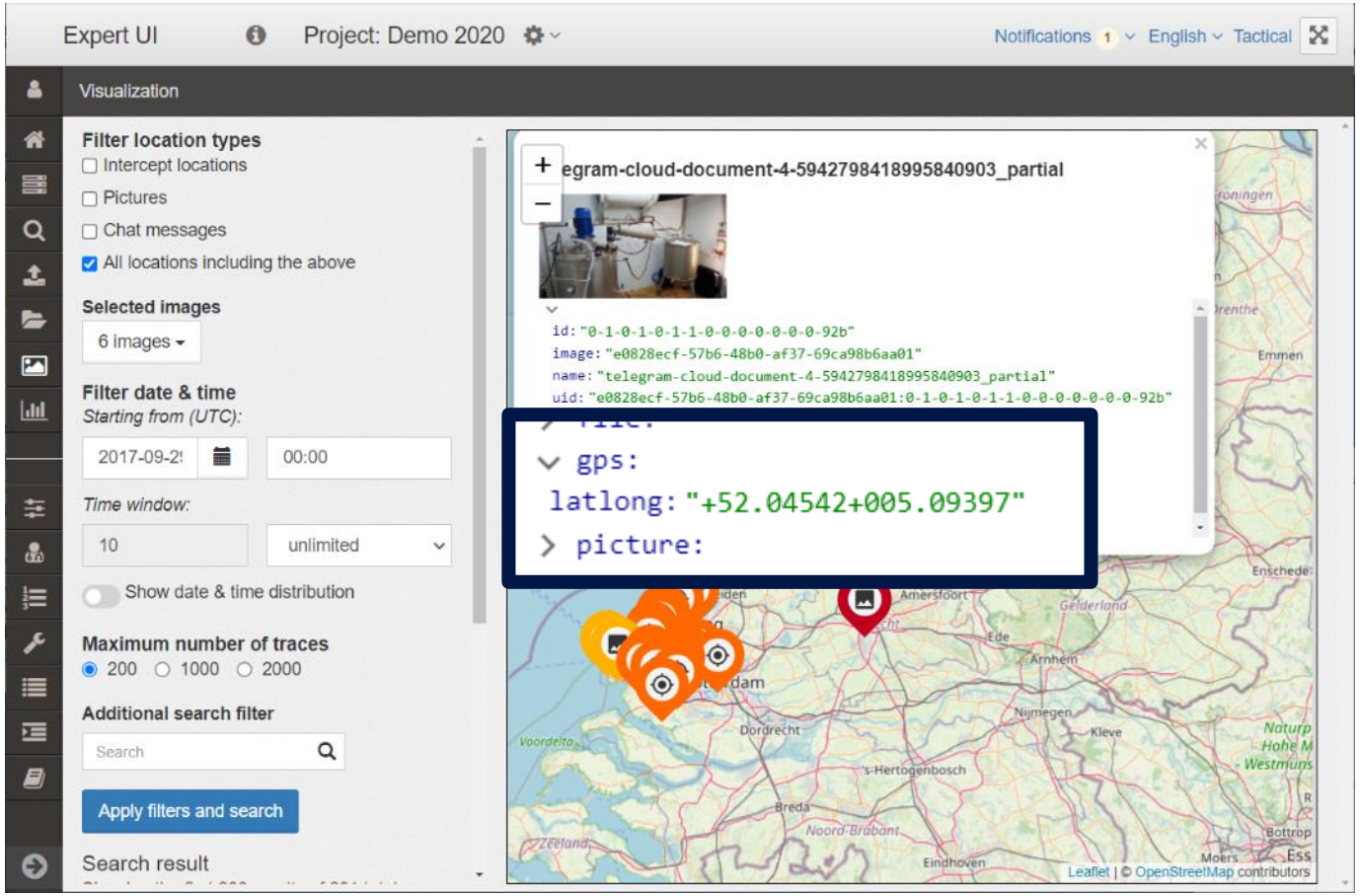
`property >= min`

`property <= max` Range query. Matches any traces that have a value between `min` and `max` for a given property. For example:

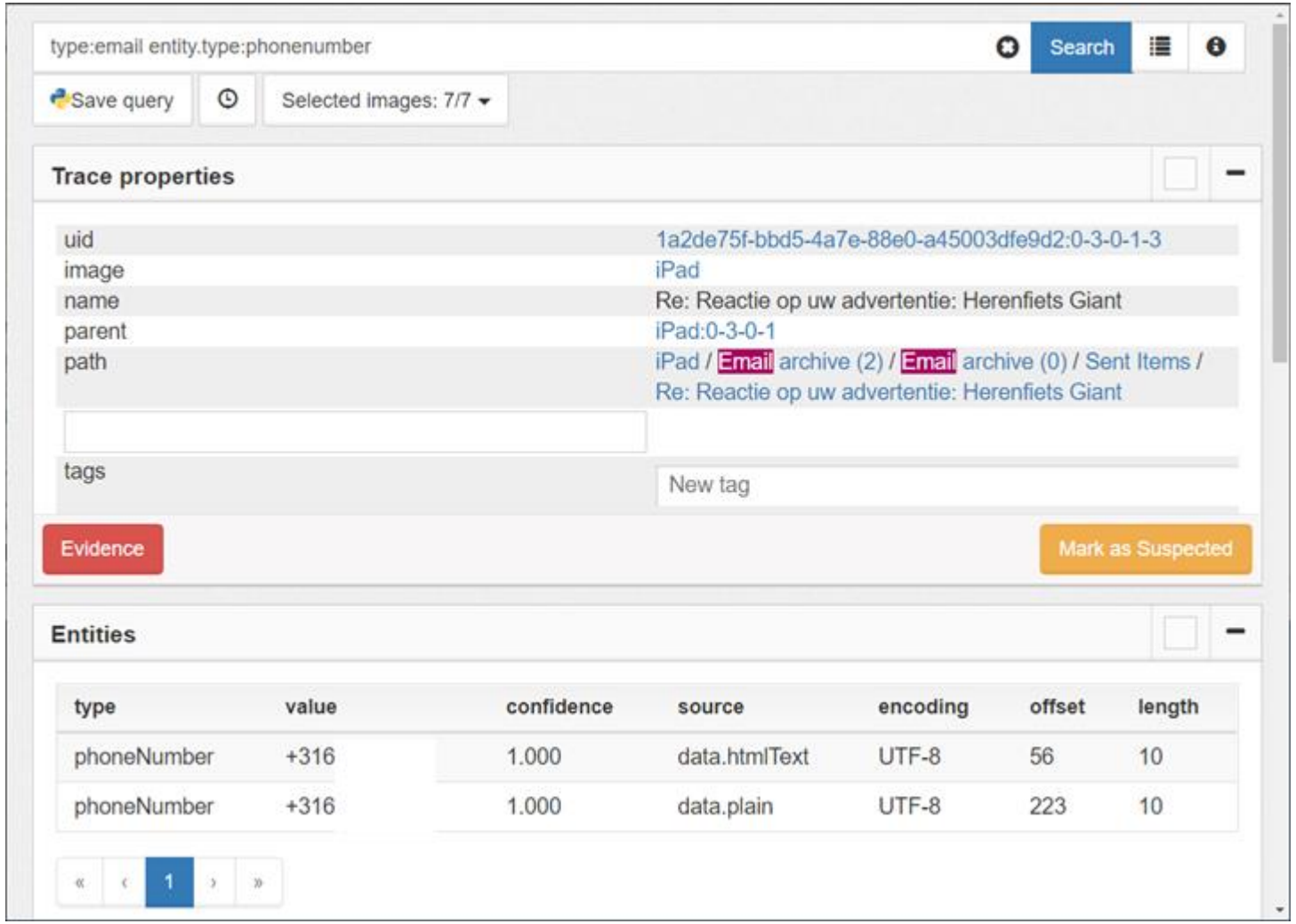
`file.createdOn: 2010..2020` matches all files created between 2010 and 2020 (inclusive).

`email.sentOn: 2021-01..2021-02` matches all emails sent between January and February 2021 (inclusive).

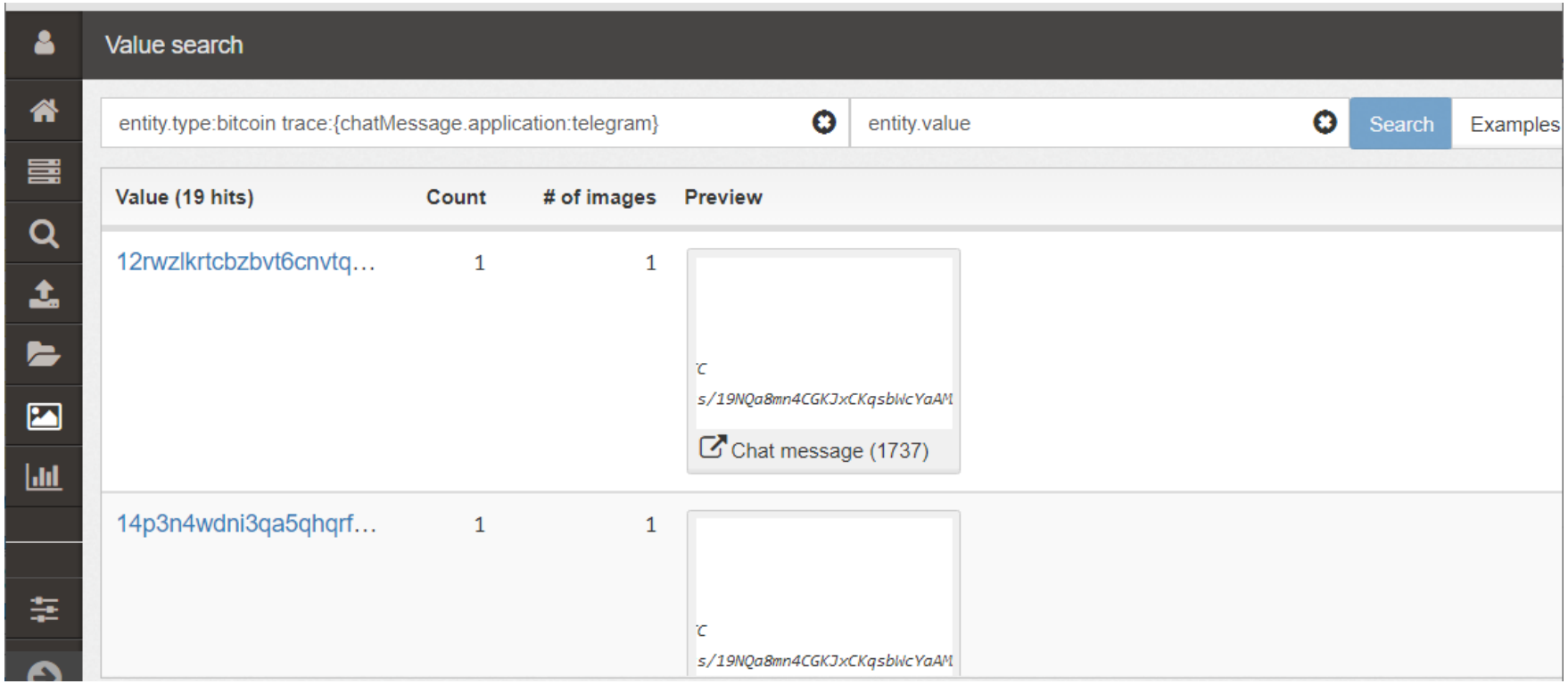
```
picture.exif.Make:Samsung
gps.latlong:(49,3)..(53,7)
```



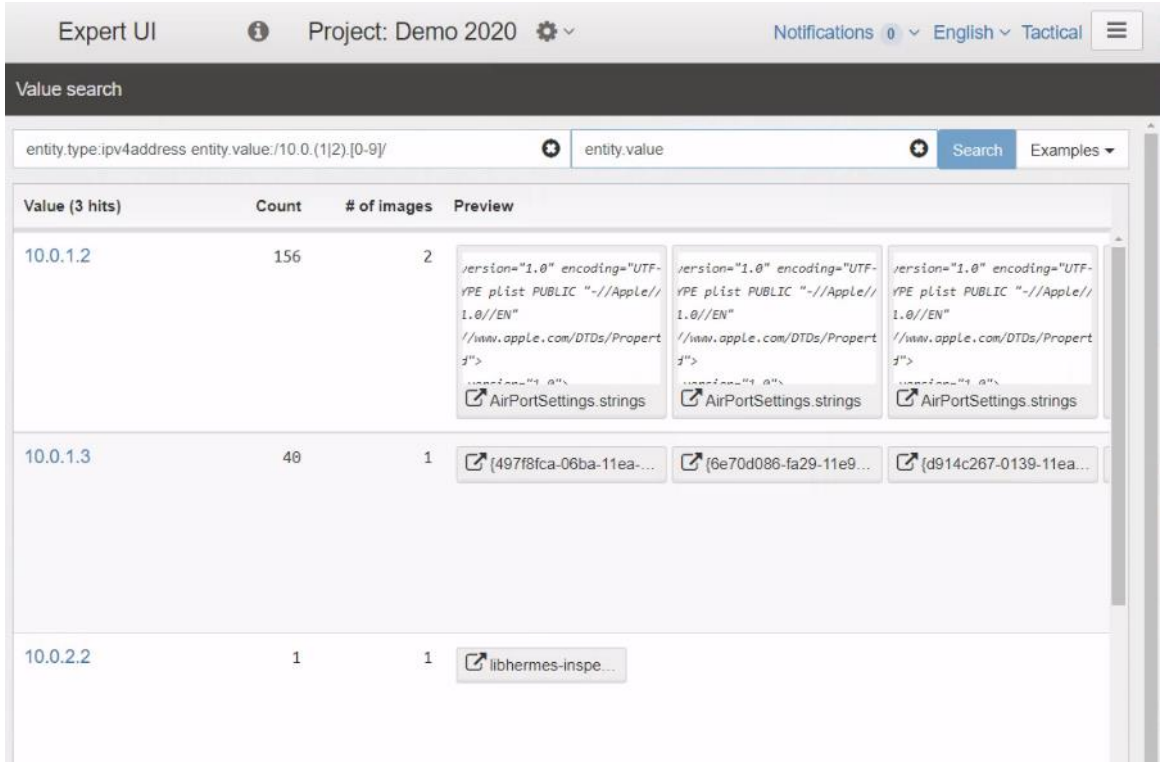
```
type:email
entity.type:phonenumber
```



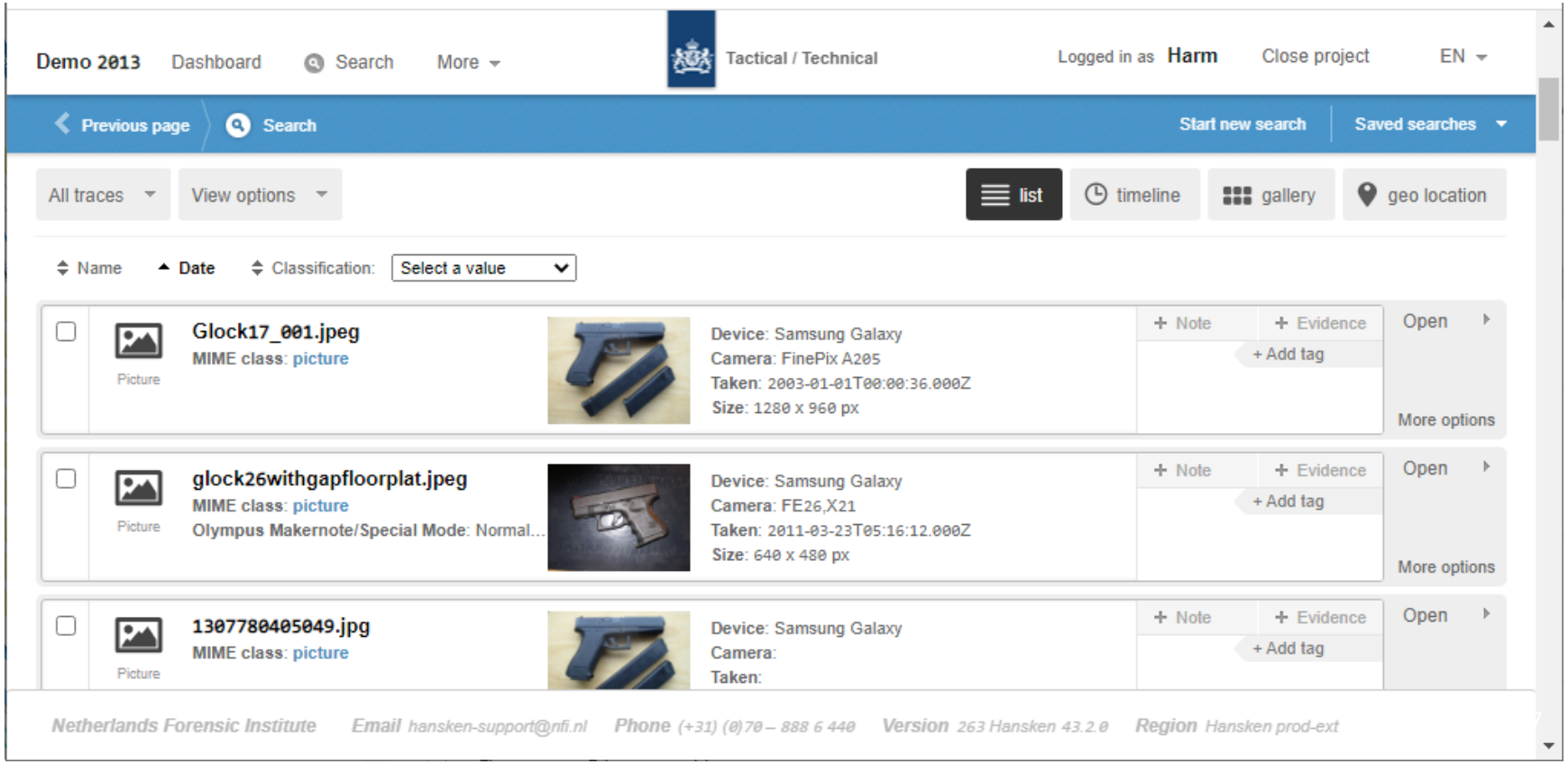
```
entity.type:bitcoin   trace:{chatMessage.application:telegram}
```



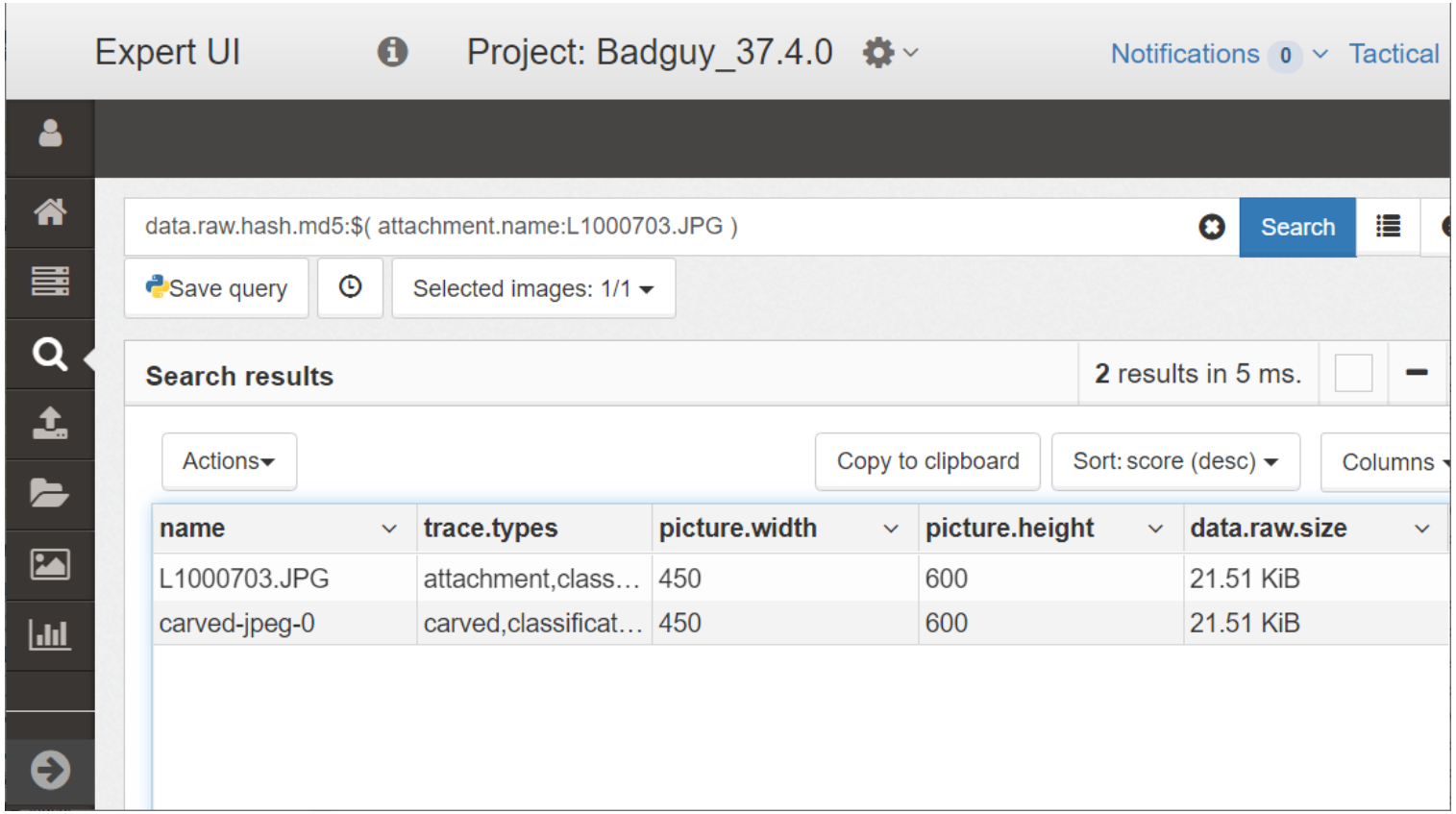
```
entity.type:ipv4address  entity.value:/10.0.(1|2).[0-9]/
```



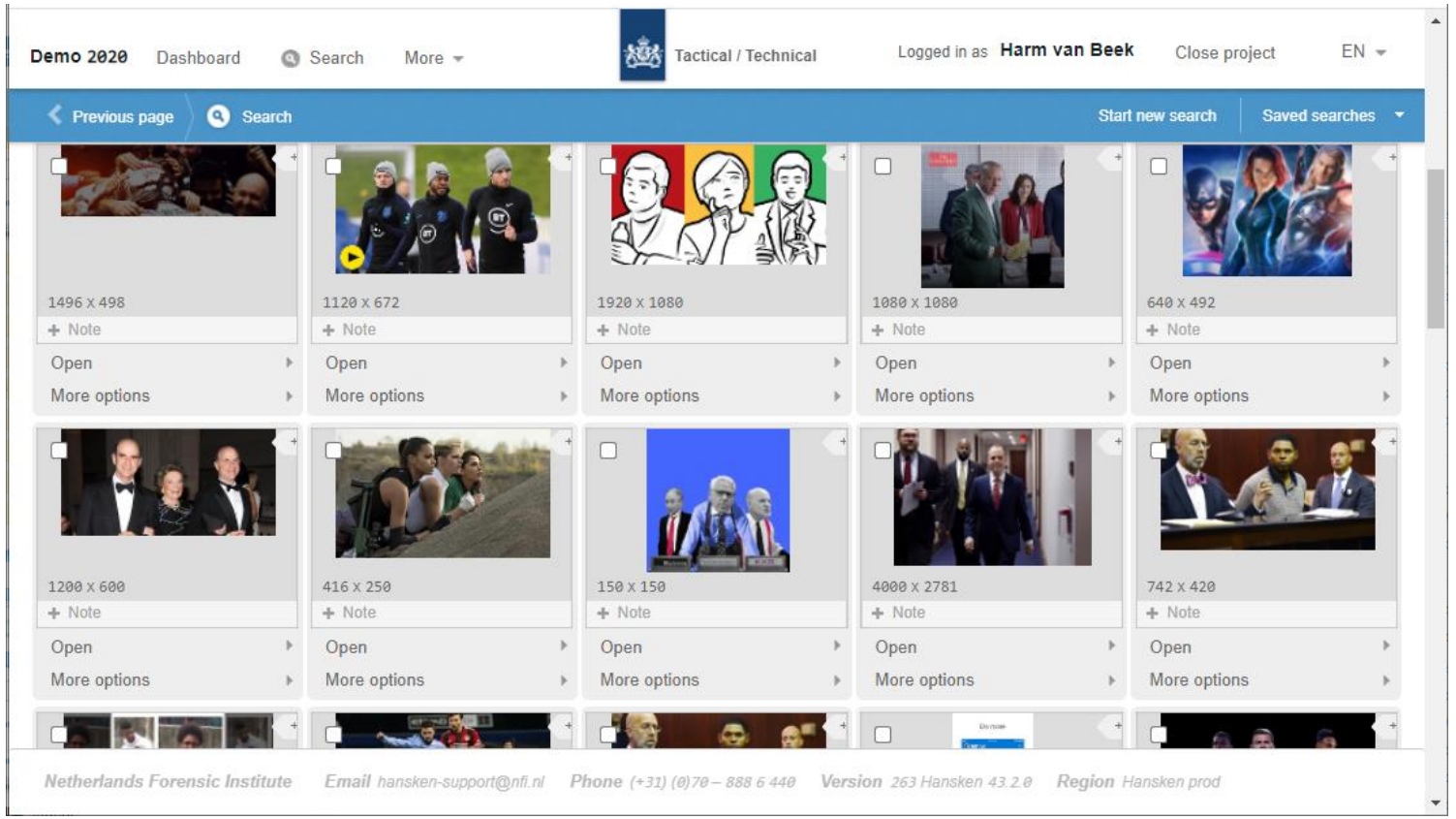
```
type:picture  prediction:{class:firearm confidence>0.8}
```



```
data.raw.hash.md5:$( attachment.name:L1000703.JPG )
```



```
picture.misc.faceCount:3
```



Hansken REST API

Operate and Investigate

Hansken's Operational API

- Case
  - Create/delete
  - Backup/restore
- Images
  - Upload/delete
  - Digest validation
  - Link to case
- Extractions
  - Active/passive scheduling
  - Status
  - Logs
  - Statistics
  - Available tools
  - Resource handling
- Trace indexes
  - Open/close
  - (Partial) Clone
  - List related images
  - Status
  - Statistics
  - Update shards
- Platform health overview
- Documentation
  - Version
  - Guide
  - Licenses
  - Release notes

Hansken's Investigative API

- Search/filter traces
  - Aggregate results
  - Including chains of evidence & custody
  - Follow tree structure
  - Download trace data
- Search/filter values
- Search/filter tracelets
- Annotate traces
  - Notes
  - Tags
  - Privileged communication
- Keyword suggestions
- Get text snippets
- Import traces
- Single file extractions
- Documentation
  - Trace model
  - Guide
- Security
  - Login, open session
  - Who am I
  - Authorizations

DELETE /resources/{resourceId} Delete a resource (metadata + actual data)

GET /resources/{resourceId} Get meta data about single resource.

POST /resources/{resourceId}/private Update resource visibility to private

POST /resources/{resourceId}/public Update resource visibility to public

POST /resources/{resourceId}/upload Upload resource data

Search Search for traces in a project

POST /projects/{projectIds}/close Close projects

POST /projects/{projectIds}/open Open projects

POST ~~/projects/{projectIds}/search~~

POST /projects/{projectIds}/suggest Suggest search terms

POST /projects/{projectIds}/tracelets/search Search tracelets

POST /projects/{projectIds}/traces/search Search traces

POST /projects/{projectIds}/values/search Search values

POST /projects/{projectId}/snippets Get trace

POST /projects/{projectId}/traces/{traceUid}/children/search Search in trace children

Security Security related endpoints

Hansken Python API

Document and Automate Investigations

» Getting started

Getting started

To get started with `hansken.py`, first make sure you have a working version of Python. Try `python3 --version` or `python --version` in a terminal or console window. Python 3.5 or higher is required for use with `hansken.py`

Most installations of Python come with a tool called `pip`, the Python package manager. As `hansken.py` is a Python package, make sure you have `pip3` or `pip` available as well. Pip can then be used to install `hansken.py`. Assuming you have a local file named something `hansken-2019.6.1-py3-none-any.whl` (see installation), install `hansken.py` with the following command:

```
$ pip3 install hansken-2019.6.1-py3-none-any.whl
... (lots of output from pip)
```

Make sure your new installation works by querying it for its version:

```
$ hansken --version
2019.06.01
```

Should the command `hansken` not be available after installation, it might be that it has not been added to your system's 'path'. `hansken.py` can also be invoked through the Python binary as such:

```
$ python3 -m hansken --version
2019.06.01
```

Either of these commands should print out a version number matching the version number in the file name (the astute reader will have noticed this is a date).

When all of this works, your installation is good to go. There is however an additional step called the 'quickstart', which can save a lot of time in the future. The quickstart can be invoked as a sub-command of the `hansken` command:

hansken.query — Constructing Hansken queries

Calls like `ProjectContext.search` accept query arguments. Aside from passing `None` for those arguments, a search query can either be a `str` formatted as Hansken Query Language that is used for user interfaces as well, or a `Query` object.

Query values

Most queries will search for traces where a particular property of a trace, like `picture.width`, matches a particular value, like `1024`. `picture.width` is clearly a numeric property. Hansken will treat any value provided with a query for a numeric property like a number, attempting to convert it if needed. The `Term` query `Term('picture.width', 1024)` is equivalent to the queries `Term('picture.width', '1024')` or `Term('picture.width', 1024.0)`. `hansken.py` will only format query values in the following two cases:

- When the value is an instance of `datetime.date` or `datetime.datetime`, the value will be converted to ISO 8601 string representation. `hansken.py` will *require* that a timezone is set on the value, so-called naive instances will cause a `ValueError` to be raised.
- When the value is a `list` or `tuple` of length 2 containing only numbers, the value is assumed to be a location and will be converted to ISO 6709 string representation.

Note

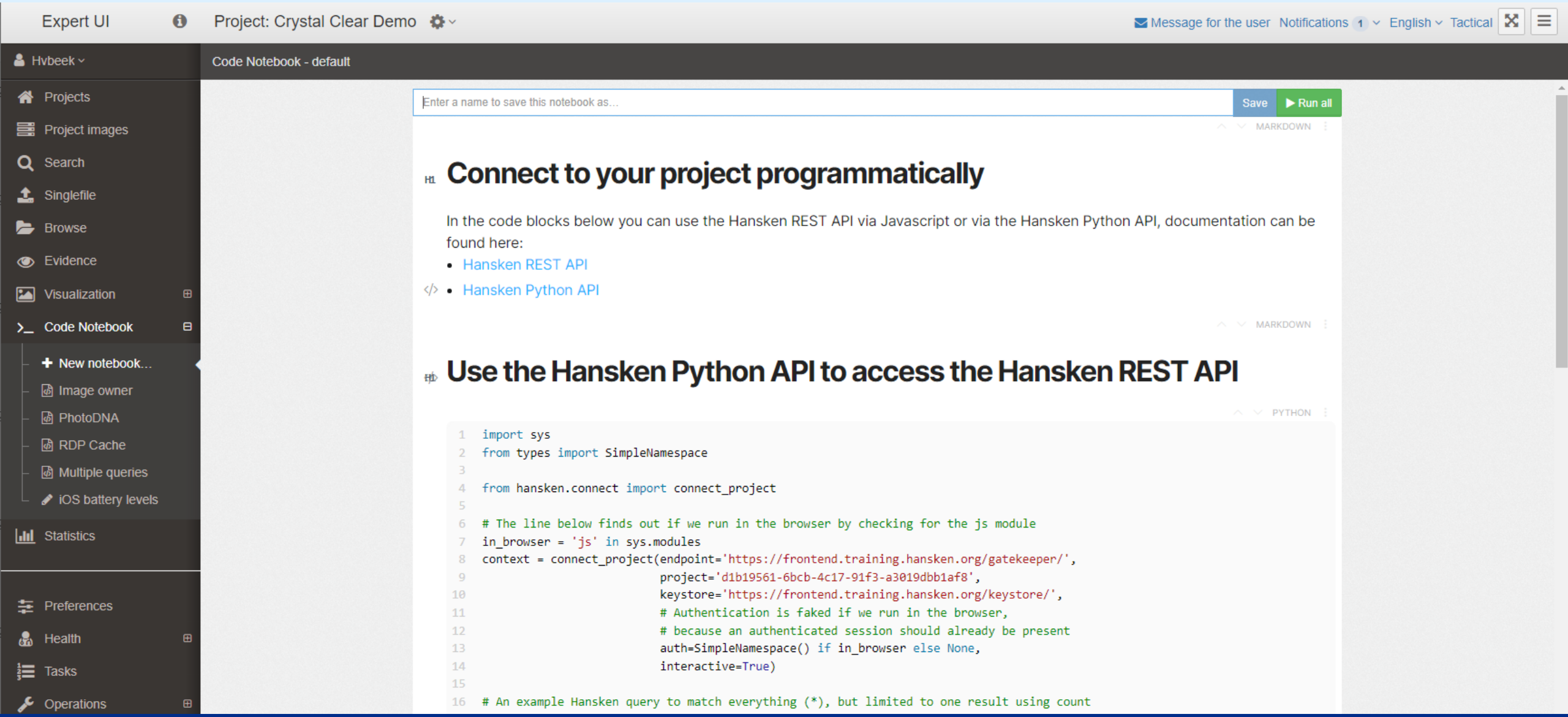
Query types that accept text values (like `Term` and `Phrase`) can be made to match the 'raw', 'original' text, before tokenization. Hansken calls this a 'full matching' query. Full matching queries are only supported on metadata, so `hansken.py` will switch a full matching query like `Term('Bomb/explos!ve', full=True)` to match only metadata if no property to match is provided.

Combining queries

Combinations of queries typically concern the boolean operations *and*, *or*, *not*. To make these combinations, use either the corresponding query types `And`,

Code Notebook in Expert UI

Document and Automate Investigations


Expert UI
Project: Crystal Clear Demo
Message for the user
Notifications 1
English
Tactical
Hvbeek
Code Notebook - default
Projects
Project images
Search
Singlefile
Browse
Evidence
Visualization
Code Notebook
New notebook...
Image owner
PhotoDNA
RDP Cache
Multiple queries
iOS battery levels
Statistics
Preferences
Health
Tasks
Operations
Enter a name to save this notebook as...
Save
Run all
MARKDOWN
Connect to your project programmatically
In the code blocks below you can use the Hansken REST API via Javascript or via the Hansken Python API, documentation can be found here:
Hansken REST API
Hansken Python API
MARKDOWN
Use the Hansken Python API to access the Hansken REST API
PYTHON
import sys
from types import SimpleNamespace
from hansken.connect import connect_project
# The line below finds out if we run in the browser by checking for the js module
in_browser = 'js' in sys.modules
context = connect_project(endpoint='https://frontend.training.hansken.org/gatekeeper/',
project='d1b19561-6bcb-4c17-91f3-a3019dbb1af8',
keystore='https://frontend.training.hansken.org/keystore/',
# Authentication is faked if we run in the browser,
# because an authenticated session should already be present
auth=SimpleNamespace() if in_browser else None,
interactive=True)
# An example Hansken query to match everything (*), but limited to one result using count

Expert UI | Project: Crystal Clear Demo | Message for the user | Notifications 1 | English | Tactical

Hvbeek

- Projects
- Project images
- Search
- Singlefile
- Browse
- Evidence
- Visualization
- Code Notebook
  - New notebook...
  - Image owner
  - PhotoDNA
  - RDP Cache
  - Multiple queries
  - iOS battery levels
- Statistics
- Preferences
- Health
- Tasks
- Operations

Code Notebook - iOS battery levels

iOS battery levels | Save | Run all

MARKDOWN

iOS battery levels from knowledgeC.db

In this notebook, we extract the battery levels from a file that is present on iOS devices, called `knowledgeC.db`

PYTHON

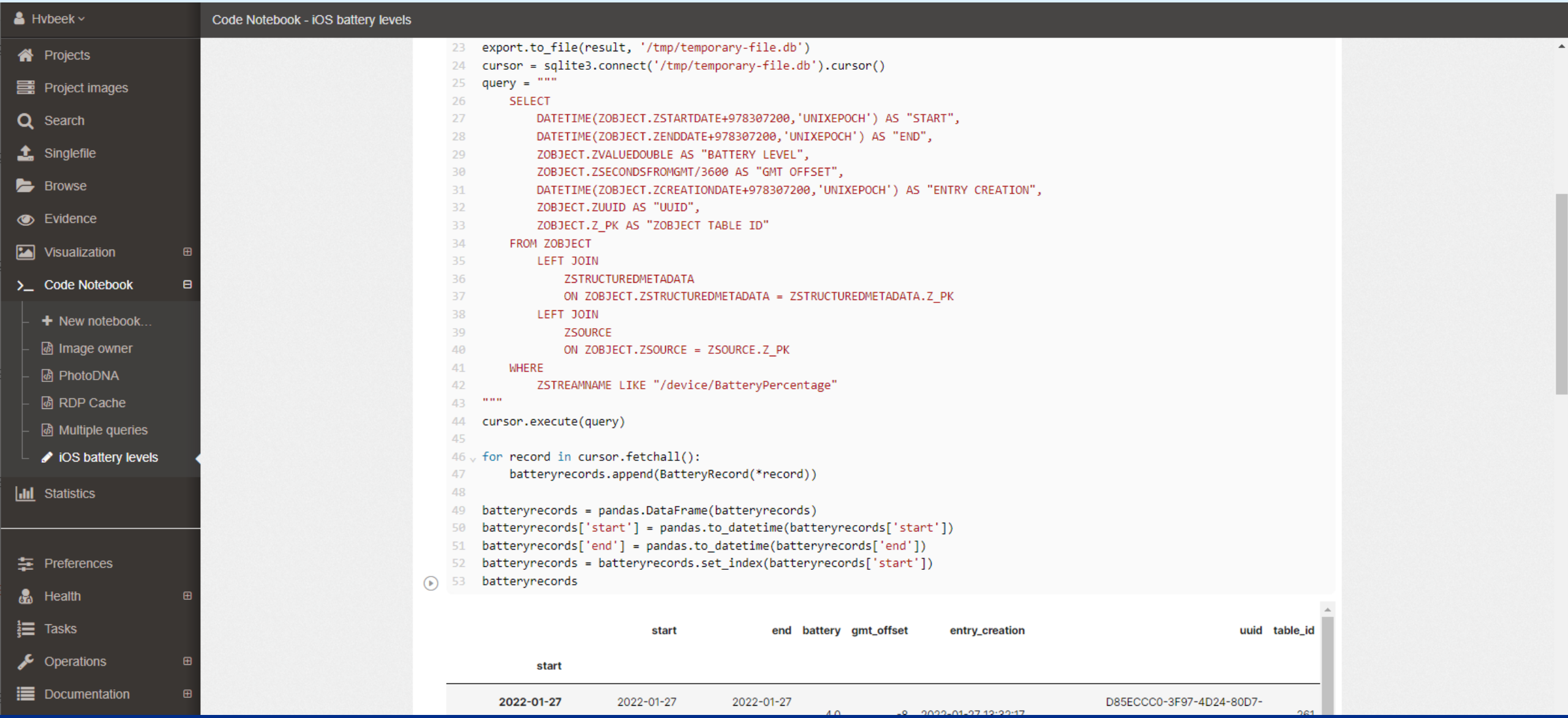
```python
import sys
from types import SimpleNamespace
from collections import namedtuple
from base64 import b64encode

import sqlite3
import pandas

from hansken.recipes import export
from hansken.connect import connect_project

in_browser = 'js' in sys.modules
BatteryRecord = namedtuple('BatteryRecord', ['start', 'end', 'battery', 'gmt_offset',
                                             'entry_creation', 'uuid', 'table_id'])
context = connect_project(endpoint='https://frontend.training.hansken.org/gatekeeper/',
                          project='d1b19561-6bcb-4c17-91f3-a3019dbb1af8',
                          keystore='frontend.training.hansken.org/keystore/',
                          auth=SimpleNamespace() if in_browser else None,
                          interactive=True)

result = context.search("file.name:knowledgeC.db", count=1).takeone()  # currently limited to one result!
batteryrecords = []
export.to_file(result, '/tmp/temporary-file.db')
cursor = sqlite3.connect('/tmp/temporary-file.db').cursor()
query = """
```


Hvbeek
Code Notebook - iOS battery levels
Projects
Project images
Search
Singlefile
Browse
Evidence
Visualization
Code Notebook
New notebook...
Image owner
PhotoDNA
RDP Cache
Multiple queries
iOS battery levels
Statistics
Preferences
Health
Tasks
Operations
Documentation
export.to_file(result, '/tmp/temporary-file.db')
cursor = sqlite3.connect('/tmp/temporary-file.db').cursor()
query = """
SELECT
DATETIME(ZOBJECT.ZSTARTDATE+978307200,'UNIXEPOCH') AS "START",
DATETIME(ZOBJECT.ZENDDATE+978307200,'UNIXEPOCH') AS "END",
ZOBJECT.ZVALUEDOUBLE AS "BATTERY LEVEL",
ZOBJECT.ZSECONDSFROMGMT/3600 AS "GMT OFFSET",
DATETIME(ZOBJECT.ZCREATIONDATE+978307200,'UNIXEPOCH') AS "ENTRY CREATION",
ZOBJECT.ZUUID AS "UUID",
ZOBJECT.Z_PK AS "ZOBJECT TABLE ID"
FROM ZOBJECT
LEFT JOIN
ZSTRUCTUREDMETADATA
ON ZOBJECT.ZSTRUCTUREDMETADATA = ZSTRUCTUREDMETADATA.Z_PK
LEFT JOIN
ZSOURCE
ON ZOBJECT.ZSOURCE = ZSOURCE.Z_PK
WHERE
ZSTREAMNAME LIKE "/device/BatteryPercentage"
"""
cursor.execute(query)
for record in cursor.fetchall():
batteryrecords.append(BatteryRecord(*record))
batteryrecords = pandas.DataFrame(batteryrecords)
batteryrecords['start'] = pandas.to_datetime(batteryrecords['start'])
batteryrecords['end'] = pandas.to_datetime(batteryrecords['end'])
batteryrecords = batteryrecords.set_index(batteryrecords['start'])
batteryrecords
start
end
battery
gmt_offset
entry_creation
uuid
table_id
start
2022-01-27
2022-01-27
2022-01-27
D85ECCC0-3F97-4D24-80D7-

Hvbeek

Code Notebook - iOS battery levels

- Projects
- Project images
- Search
- Singlefile
- Browse
- Evidence
- Visualization
- Code Notebook
  - New notebook...
  - Image owner
  - PhotoDNA
  - RDP Cache
  - Multiple queries
  - iOS battery levels
- Statistics
- Preferences
- Health
- Tasks
- Operations
- Documentation

2022-01-27 13:37:44	2022-01-27 13:37:44	2022-01-27 13:41:48	9.0	-8	2022-01-27 13:41:48	91FB1E9D-3D5C-48DD-B4C8-FE7F2ADF247C	265
2022-01-27 13:41:48	2022-01-27 13:41:48	2022-01-27 13:44:32	13.0	-8	2022-01-27 13:44:33	308B2E20-57A0-4690-9E3A-DB5C8CD8FCA0	266
...	...	...	...	...	...	...	...

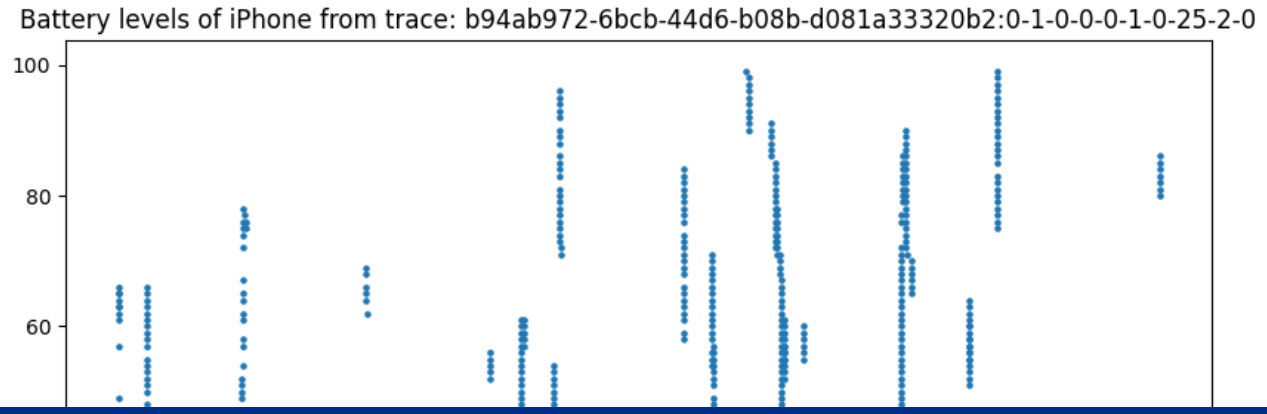
MARKDOWN

Using matplotlib, we can plot the battery draining and charging over time, by plotting each of these points.

PYTHON

```
from js import document
from matplotlib import pyplot

pyplot.figure(figsize=(10, 6))
pyplot.scatter(batteryrecords['start'], batteryrecords['battery'], s=5)
pyplot.title(f'Battery levels of iPhone from trace: {result.uid}')
pyplot.show()
```



Hansken SDK overview

- Trace Model
- Query Language
- REST API
- Python API
- Expert UI
- Code notebooks

All documented, with examples and guides, available via the Expert UI

Demonstration

Hansken SDK

Agenda

Time	Title
09:00	Part I
	Introduction
	The Hansken SDK
	Live demonstration of Hansken and the Hansken SDK
10:45	Prepping for the exercises
11:00	Break
11:15	**Part II**
	Excercises
13:00	End

Preparing for the exercises

- We recommend that you work in pairs
- Browse to one of the IP addresses of the EC2 instance of the Hansken SDK using **http**!
- Open the Expert UI, search and see if you get results
- Setup your laptop with Visual Code Studio
- The Exercises can be found on Github:

https://github.com/NetherlandsForensicInstitute/hansken-python-workshop

Agenda

Time	Title
09:00	Part I
	Introduction
	The Hansken SDK
	Live demonstration of Hansken and the Hansken SDK
10:45	Prepping for the exercises
11:00	Break
11:15	**Part II**
	Excercises
13:00	End

Carrying out the exercises

- Open exercise 1 and modify the IP address in the code to match your EC2 instance
- Run through the cells in the codebook and try to understand what is happening
- Try to extend the script or customize it to a different type of data
- Do this also for examples 2, 3, and 5
- Advanced: Ingest your own data file(s) in Hansken and visualise it