

# DFRWS2023 HYBRID USA

JULY 9–12

Baltimore, MD

in cooperation with

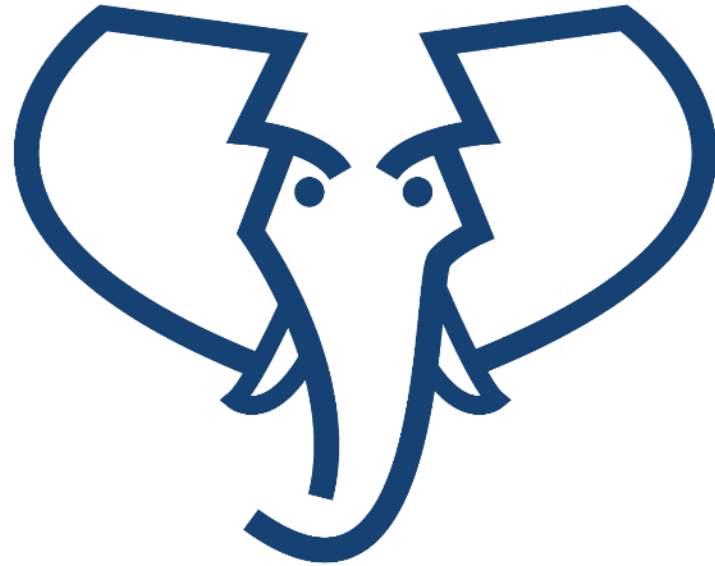


JOHNS HOPKINS  
UNIVERSITY



**“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
<b>09:00</b>	<b>Part I</b>
	Introduction
	The Hansken SDK
	Live demonstration of Hansken and the Hansken SDK
10:45	Preparing for the exercises
11:00	Break
<b>11:15</b>	<b>Part II</b>
	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

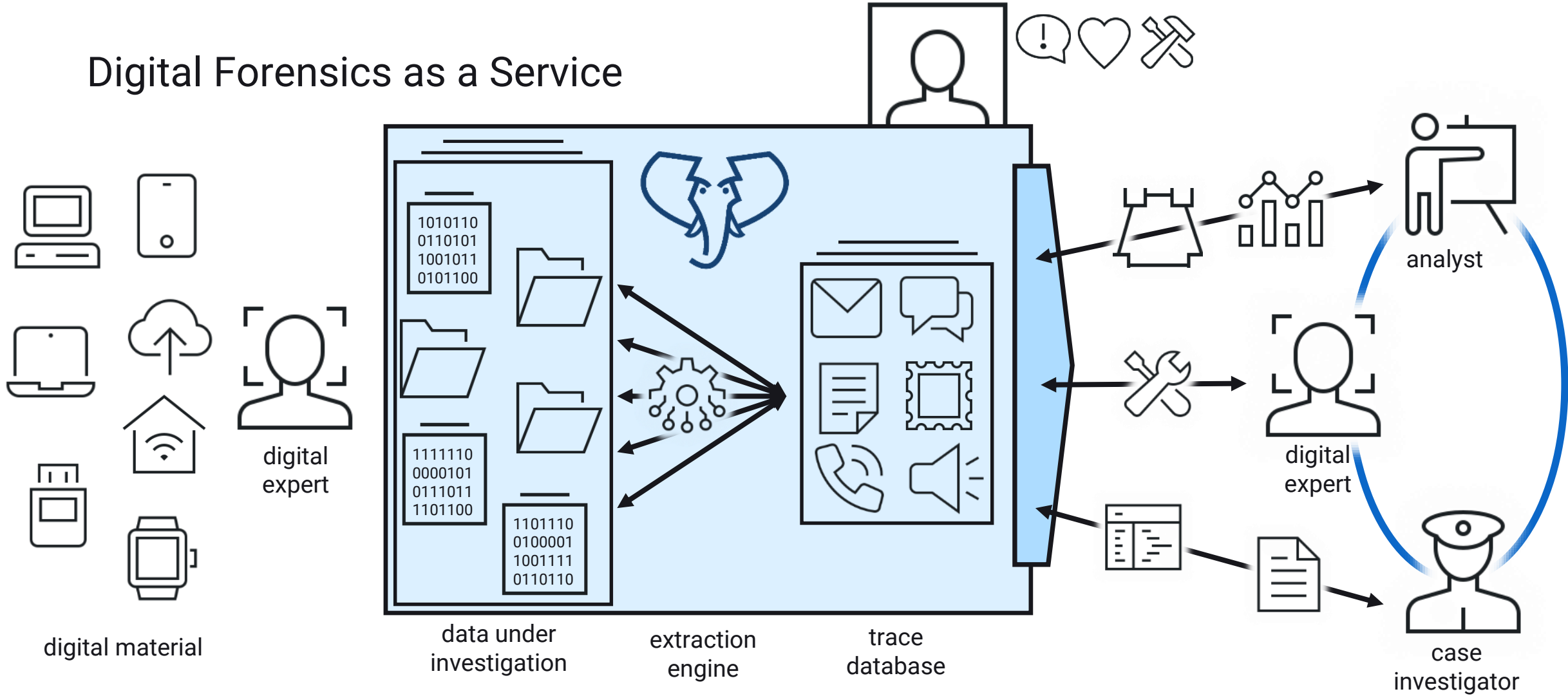


# Introduction

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

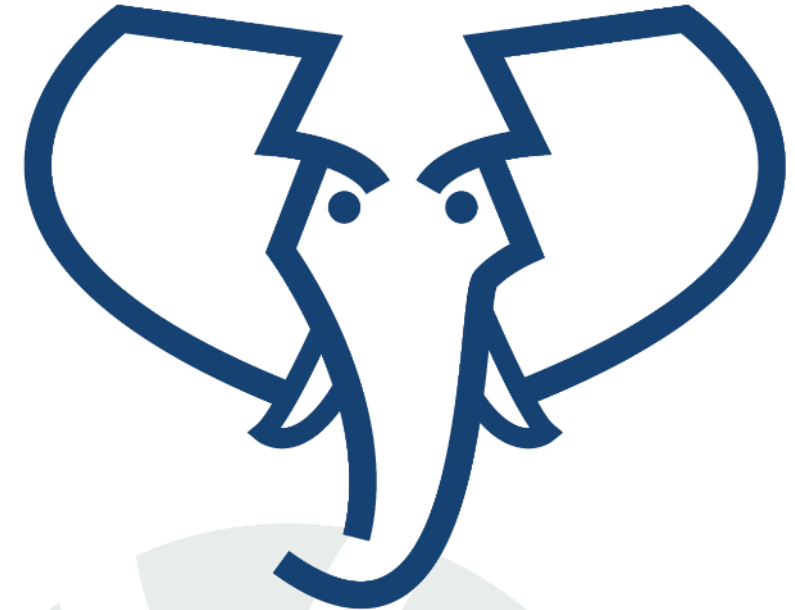
- all evidence is traceable (chain of evidence) ✓
- advanced logging of embedded data processors ✓

### 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

The image displays a composite of three main interface components from the Hansen platform:

- Search Interface (Left):** Shows a search bar with filters for 'Type of trace' (set to 'Picture') and 'Camera'. A 'Search' button is visible. Below, a list of traces is shown with a grid of image thumbnails.
- Expert UI (Center):** Displays detailed information for a trace with uid 'de302013-0003-0000-0000-000000000001:0-0-0-2-0-7-1-0-11-0-5-2'. It includes sections for 'Trace properties', 'Trace system processed properties', and 'Trace system extracted properties' (showing email headers and references).
- Dashboard & Code Notebook (Right):** Shows a dashboard with various charts and filters. Overlaid on this is a code notebook window with the following Python code:

```
1 #!/usr/bin/env python3
2 # encoding=utf-8
3
4 from hansen.tool import run
5
6 # we define a function to do the things we want to do
7 def search_and_process(context):
8     # our argument "context" is a hansen.remote.ProjectContext, see the docs
9     # for what it can do other than searching for traces
10    with context:
11        # query was copied from Hansen Expert UI
12        query = 'type:file'
13        # search for traces matching the query
14        results = context.search(query)
15
16    for trace in results:
17        # process the results, print some details for each trace we've found
18        print(trace.uid, trace.name)
19
20
21 if __name__ == '__main__':
22     # call the hansen.py command line, but make it call our function
23     run(with_context=search_and_process,
24         # the gatekeeper REST endpoint when this script was exported, note that
25         # this can be overridden by passing -e/--endpoint on the command line
26         endpoint='http://localhost:9091/gatekeeper/',
27         # the keystore REST endpoint when this script was exported, note that
28         # this can be overridden with --keystore
29         keystore='http://localhost:9090/keystore/',
30         # the project id of the project named "Bad Guy",
31         project='48619f8a-1e07-445b-ae0e-905392bc7400')
```



# Building the Hansken Community in the Netherlands....



Netherlands Forensic Institute  
*Ministry of Justice and Security*



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*



POLITIEACADEMIE

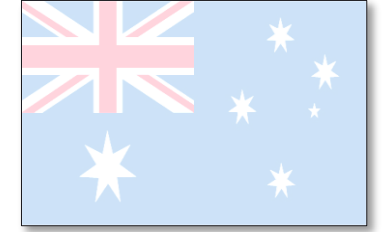
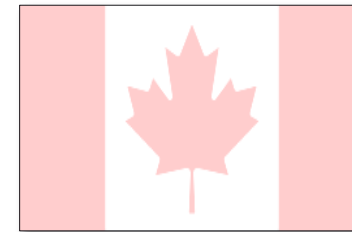
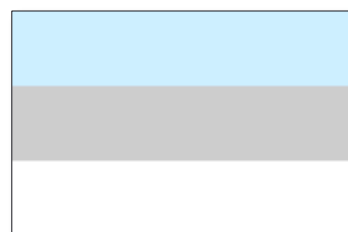
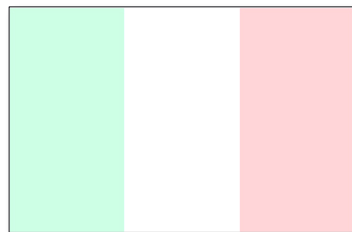
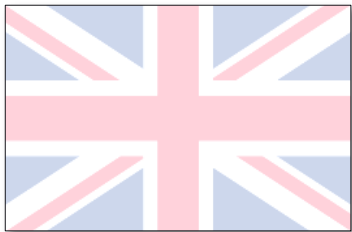
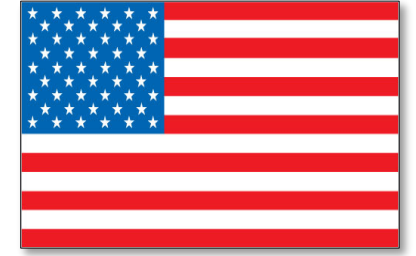
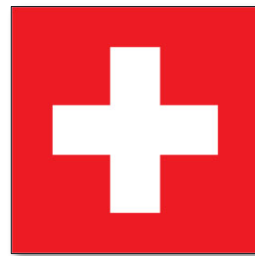
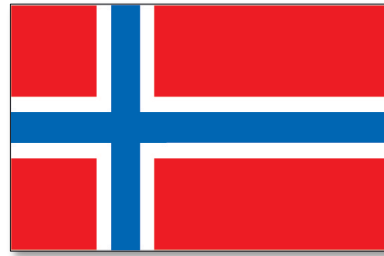
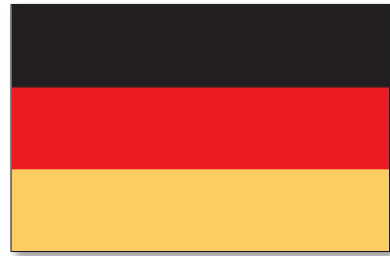


Justitiële ICT Organisatie  
*Ministerie van Justitie en Veiligheid*





## 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 Hansken in a research project with non-academic partners





# 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?



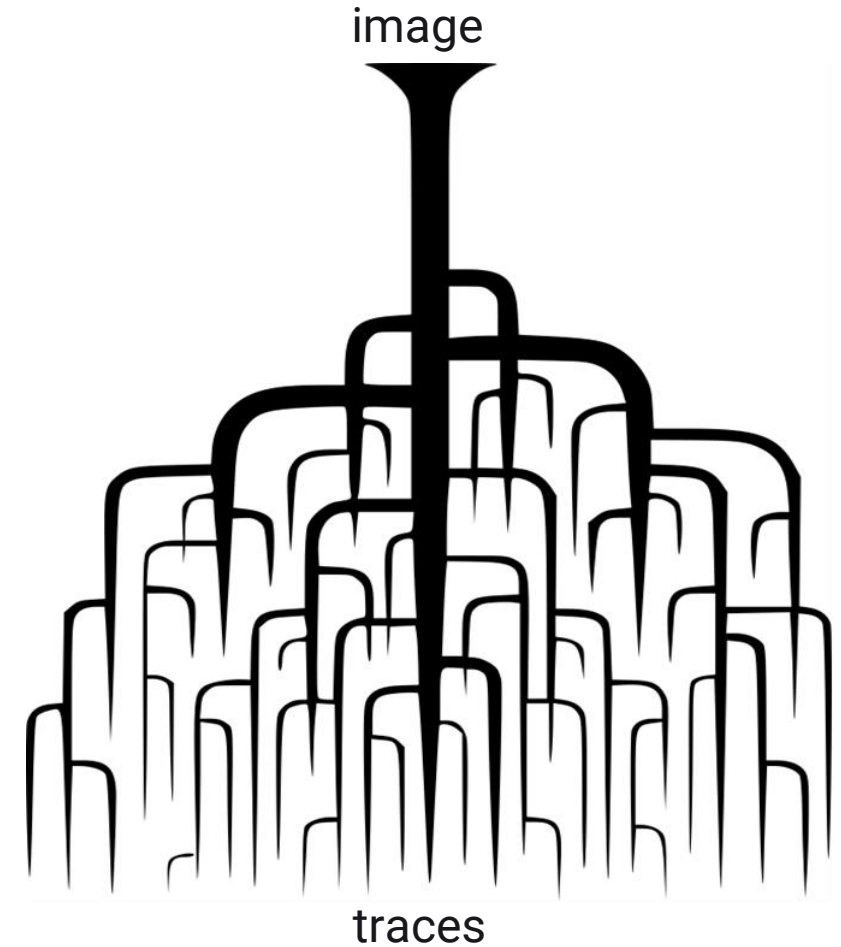
# Trace Model

## 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







Hvbeek v

- Visualization
- Code Notebook
- Statistics
- Preferences
- Health
- Tasks
- Operations
- Documentation
  - Extraction Plugin SDK
  - Extraction tools
  - Guide
  - Python API
  - REST API
  - Trace model
  - Vakbijlage (Dutch)
- Release notes

Documentation - Trace model

<b>+</b> cryptoKeyPair	Two crypto keys of types public and private.			<i>extracted</i>
<b>+</b> data	The data (content) of the trace, indexed by data type. For example data.raw.size, data.text.size			<i>extracted</i>
<b>+</b> deleted	A trace that has been deleted from the file system, or has been placed in a recycle bin or trash can.			<i>extracted</i>
<b>+</b> dns	IP DNS information regardless of lookup or reverselookup			<i>extracted</i>
<b>+</b> document	A document, for example an Office documents.			<i>extracted</i>
<b>-</b> email	An electronically sent mail message.			<i>extracted</i>
Name	Description	Cardinality	Collection	Type
application	The application storing this email.			<i>string</i>
bcc	A list of blank carbon copied receiver's email addresses.		<i>list</i>	<i>string</i>
categories	Categories applied to the email.		<i>map</i>	<i>string</i>
cc	A list of carbon copied receiver's email addresses.		<i>list</i>	<i>string</i>
createdOn	The date and time at which the email was created.			<i>date</i>
from	The sender's email address.			<i>string</i>
hasAttachment	Indicates if the email has an attachment.			<i>boolean</i>
headers	The email headers of the email.		<i>map</i>	<i>string</i>
inReplyTo	A unique identifier for identifying the email this email is a reply to.			<i>string</i>
labels	Named and colored label.		<i>map</i>	<i>string</i>



Trace system extracted properties	
<b>data</b>	
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)
<b>file</b>	
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
<b>gps</b>	
createdOn	2022-02-21T10:12:35.439Z
latlong	+52.16617+004.50893
<b>picture</b>	
aspectRatio	1.333
camera	iPhone 6s Plus
digitizedOn	2022-02-21T10:12:35.439Z
<b>exif</b>	
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

```

<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

```

### 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 emailaddresses, ipv4 addresses, etc.

**"quick brown fox"** | **"lorem ipsum"** | **"patient 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**



# DFRWS USA 2023 workshop: Providing DFaaS with code notebooks

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/jpeg 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 emailaddresses, 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-DDTHH: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  
similarly, < / <= 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

- [Text Queries](#)
- [Date Queries](#)
- [Numeric Queries](#)
- [Latlong Queries](#)
- [Boolean Queries](#)
- [Other Queries](#)
- [Glossary](#)

## 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)
```

The screenshot displays the Expert UI interface for a project named "Demo 2020". The interface is divided into several sections:

- Visualization:** A sidebar on the left contains various icons for navigation and analysis.
- Filter location types:** Includes checkboxes for "Intercept locations", "Pictures", "Chat messages", and "All locations including the above" (which is checked).
- Selected images:** Shows "6 images" selected.
- Filter date & time:** Includes a date range selector (set to "2017-09-2!" and "00:00") and a "Time window" dropdown (set to "10" and "unlimited").
- Maximum number of traces:** Radio buttons for "200", "1000", and "2000" (with "200" selected).
- Additional search filter:** A search input field with a magnifying glass icon.
- Apply filters and search:** A blue button to execute the search.
- Search result:** A section at the bottom for displaying search results.

The main area shows a map of the Netherlands with several orange location markers. A detailed view of a Telegram document is overlaid on the map, showing the following metadata:

```
telegram-cloud-document-4-5942798418995840903_partial  
id: "0-1-0-1-0-1-1-0-0-0-0-0-0-92b"  
image: "e0828ecf-57b6-48b0-af37-69ca98b6aa01"  
name: "telegram-cloud-document-4-5942798418995840903_partial"  
uid: "e0828ecf-57b6-48b0-af37-69ca98b6aa01:0-1-0-1-0-1-1-0-0-0-0-0-0-92b"  
file:  
  > gps:  
    latlong: "+52.04542+005.09397"  
  > picture:
```



type:email

entity.type:phonenummer

The screenshot shows a search interface with the following components:

- Query:** type:email entity.type:phonenummer
- Actions:** Save query, Selected images: 7/7, Search, and a menu icon.
- Trace properties:**
  - uid: 1a2de75f-bbd5-4a7e-88e0-a45003dfe9d2:0-3-0-1-3
  - image: iPad
  - name: Re: Reactie op uw advertentie: Herenfiets Giant
  - parent: iPad:0-3-0-1
  - path: iPad / Email archive (2) / Email archive (0) / Sent Items / Re: Reactie op uw advertentie: Herenfiets Giant
  - tags: New tag
- Evidence:** A red button labeled "Evidence" and an orange button labeled "Mark as Suspected".
- Entities:** A table with the following data:

type	value	confidence	source	encoding	offset	length
phoneNumber	+316	1.000	data.htmlText	UTF-8	56	10
phoneNumber	+316	1.000	data.plain	UTF-8	223	10
- Navigation:** A pagination bar showing page 1 of 1.





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

The screenshot shows a search interface with a sidebar on the left containing navigation icons. The main area is titled 'Value search' and contains a search bar with the query 'entity.type:bitcoin trace:{chatMessage.application:telegram}' and a search button. Below the search bar, there are two search results displayed in a table format.

Value (19 hits)	Count	# of images	Preview
<a href="#">12rwzlkrtcbzbvt6cnvtq...</a>	1	1	 Chat message (1737)
<a href="#">14p3n4wdni3qa5qhqr...</a>	1	1	



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

The screenshot shows the Expert UI interface for a search query. The search bar contains the query: `entity.type:ipv4address entity.value:/10.0.(1|2).[0-9]/`. The results are displayed in a table with columns: Value (3 hits), Count, # of images, and Preview.

Value (3 hits)	Count	# of images	Preview
10.0.1.2	156	2	<pre>&lt;string&gt;\"/&gt;</pre>
10.0.1.3	40	1	
10.0.2.2	1	1	









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

Demo 2013 Dashboard Search More Tactical / Technical Logged in as Harm Close project EN

Previous page Search Start new search Saved searches

All traces View options list timeline gallery geo location

Name	Date	Classification	Select a value
<input type="checkbox"/>  <b>Glock17_001.jpeg</b> MIME class: picture			
		Device: Samsung Galaxy Camera: FinePix A205 Taken: 2003-01-01T00:00:36.000Z Size: 1280 x 960 px	+ Note + Evidence + Add tag
<input type="checkbox"/>  <b>glock26withgapfloorplat.jpeg</b> MIME class: picture Olympus Makernote/Special Mode: Normal...			
		Device: Samsung Galaxy Camera: FE26,X21 Taken: 2011-03-23T05:16:12.000Z Size: 640 x 480 px	+ Note + Evidence + Add tag
<input type="checkbox"/>  <b>1307780405049.jpg</b> MIME class: picture			
		Device: Samsung Galaxy Camera: Taken:	+ Note + Evidence + Add tag

Netherlands Forensic Institute Email hansken-support@nfi.nl Phone (+31) (0)70 - 888 6 440 Version 263 Hansken 43.2.0 Region Hansken prod-ext



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

The screenshot shows the Expert UI interface. At the top, it displays 'Expert UI', 'Project: Badguy\_37.4.0', and 'Notifications 0'. The search bar contains the query 'data.raw.hash.md5:\$( attachment.name:L1000703.JPG )'. Below the search bar, there are buttons for 'Save query', a refresh icon, and 'Selected images: 1/1'. The search results section shows '2 results in 5 ms.' and a table with the following data:

name	trace.types	picture.width	picture.height	data.raw.size
L1000703.JPG	attachment,class...	450	600	21.51 KiB
carved-jpeg-0	carved,classificat...	450	600	21.51 KiB



picture.misc.faceCount:3

The screenshot displays a web application interface for image search. At the top, there is a navigation bar with 'Demo 2020', 'Dashboard', 'Search', and 'More'. The user is logged in as 'Harm van Beek'. Below the navigation bar is a search bar with 'Previous page' and 'Search' buttons. The main content area shows a grid of image search results. Each result includes a thumbnail image, its dimensions, a '+ Note' button, and 'Open' and 'More options' buttons. The footer contains contact information for the Netherlands Forensic Institute, including email, phone, version, and region.

Image Dimensions	Image Description
1496 x 498	Group of people in a crowd
1120 x 672	Three people in winter gear
1920 x 1080	Cartoon illustration of three people
1080 x 1080	Two men in suits talking
640 x 492	Two women in superhero costumes
1200 x 600	Three men in tuxedos
416 x 250	Group of people outdoors
150 x 150	Two men in suits at a podium
4000 x 2781	Group of men in suits
742 x 420	Two men in suits at a table

Netherlands Forensic Institute | Email [hansken-support@nfi.nl](mailto:hansken-support@nfi.nl) | Phone (+31) (0)70 - 888 6 440 | Version 263 Hansken 43.2.0 | Region Hansken prod



# 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





# DFRWS USA 2023 workshop: Providing DFaaS with code notebooks

<b>DELETE</b>	<code>/resources/{resourceId}</code>	Delete a resource (metadata + actual data)
<b>GET</b>	<code>/resources/{resourceId}</code>	Get meta data about single resource.
<b>POST</b>	<code>/resources/{resourceId}/private</code>	Update resource visibility to private
<b>POST</b>	<code>/resources/{resourceId}/public</code>	Update resource visibility to public
<b>POST</b>	<code>/resources/{resourceId}/upload</code>	Upload resource data
<b>Search</b> Search for traces in a project <span>∨</span>		
<b>POST</b>	<code>/projects/{projectIds}/close</code>	Close projects
<b>POST</b>	<code>/projects/{projectIds}/open</code>	Open projects
<b>POST</b>	<code>/projects/{projectIds}/search</code>	
<b>POST</b>	<code>/projects/{projectIds}/suggest</code>	Suggest search terms
<b>POST</b>	<code>/projects/{projectIds}/tracelets/search</code>	Search tracelets
<b>POST</b>	<code>/projects/{projectIds}/traces/search</code>	Search traces
<b>POST</b>	<code>/projects/{projectIds}/values/search</code>	Search values
<b>POST</b>	<code>/projects/{projectId}/snippets</code>	Get trace
<b>POST</b>	<code>/projects/{projectId}/traces/{traceUid}/children/search</code>	Search in trace children

## Security ∨

Security related endpoints



# Hansken Python API

## Document and Automate Investigations



## 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.py  
2022.3.1.2

Search docs

Getting started

Examples

API Documentation

- hansken.tool — Command line interaction
- hansken.connect — Connecting to Hansken
- hansken — Global values
- hansken.trace — Interact with traces / search results
- hansken.query — Constructing Hansken queries**
  - Query values
  - Combining queries
  - Sorting results
  - Facets
  - Classes in hansken.query
- hansken.auth — Authentication
- hansken.remote — Communication with Hansken
- hansken.util — Utilities for internal and external use

Recipes

Command line interface

» API Documentation » hansken.query — Constructing Hansken queries

## 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/explosive', 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

```
1 import sys
2 from types import SimpleNamespace
3
4 from hansken.connect import connect_project
5
6 # The line below finds out if we run in the browser by checking for the js module
7 in_browser = 'js' in sys.modules
8 context = connect_project(endpoint='https://frontend.training.hansken.org/gatekeeper/',
9                           project='d1b19561-6bcb-4c17-91f3-a3019dbb1af8',
10                          keystore='https://frontend.training.hansken.org/keystore/',
11                          # Authentication is faked if we run in the browser,
12                          # because an authenticated session should already be present
13                          auth=SimpleNamespace() if in_browser else None,
14                          interactive=True)
15
16 # 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    Code Notebook - iOS battery levels

Save    Run all

## 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`

```
1 import sys
2 from types import SimpleNamespace
3 from collections import namedtuple
4 from base64 import b64encode
5
6 import sqlite3
7 import pandas
8
9 from hansken.recipes import export
10 from hansken.connect import connect_project
11
12 in_browser = 'js' in sys.modules
13 BatteryRecord = namedtuple('BatteryRecord', ['start', 'end', 'battery', 'gmt_offset',
14                                             'entry_creation', 'uuid', 'table_id'])
15 context = connect_project(endpoint='https://frontend.training.hansken.org/gatekeeper/',
16                          project='d1b19561-6bcb-4c17-91f3-a3019dbb1af8',
17                          keystore='frontend.training.hansken.org/keystore/',
18                          auth=SimpleNamespace() if in_browser else None,
19                          interactive=True)
20
21 result = context.search("file.name:knowledgeC.db", count=1).takeone() # currently limited to one result!
22 batteryrecords = []
23 export.to_file(result, '/tmp/temporary-file.db')
24 cursor = sqlite3.connect('/tmp/temporary-file.db').cursor()
25 query = ""
```



# DFRWS USA 2023 workshop: Providing DFaaS with code notebooks

Code Notebook - iOS battery levels

```
23 export.to_file(result, '/tmp/temporary-file.db')
24 cursor = sqlite3.connect('/tmp/temporary-file.db').cursor()
25 query = """
26     SELECT
27         DATETIME(ZOBJECT.ZSTARTDATE+978307200,'UNIXEPOCH') AS "START",
28         DATETIME(ZOBJECT.ZENDDATE+978307200,'UNIXEPOCH') AS "END",
29         ZOBJECT.ZVALUEDOUBLE AS "BATTERY LEVEL",
30         ZOBJECT.ZSECONDSFROMGMT/3600 AS "GMT OFFSET",
31         DATETIME(ZOBJECT.ZCREATIONDATE+978307200,'UNIXEPOCH') AS "ENTRY CREATION",
32         ZOBJECT.ZUUID AS "UUID",
33         ZOBJECT.Z_PK AS "ZOBJECT TABLE ID"
34     FROM ZOBJECT
35     LEFT JOIN
36         ZSTRUCTUREDMETADATA
37         ON ZOBJECT.ZSTRUCTUREDMETADATA = ZSTRUCTUREDMETADATA.Z_PK
38     LEFT JOIN
39         ZSOURCE
40         ON ZOBJECT.ZSOURCE = ZSOURCE.Z_PK
41     WHERE
42         ZSTREAMNAME LIKE "/device/BatteryPercentage"
43     """
44     cursor.execute(query)
45
46     for record in cursor.fetchall():
47         batteryrecords.append(BatteryRecord(*record))
48
49     batteryrecords = pandas.DataFrame(batteryrecords)
50     batteryrecords['start'] = pandas.to_datetime(batteryrecords['start'])
51     batteryrecords['end'] = pandas.to_datetime(batteryrecords['end'])
52     batteryrecords = batteryrecords.set_index(batteryrecords['start'])
53     batteryrecords
```

	start	end	battery	gmt_offset	entry_creation	uuid	table_id
	2022-01-27	2022-01-27	2022-01-27	4.0	-8	2022-01-27 12:22:17	D85ECCC0-3F97-4D24-80D7-261





# DFRWS USA 2023 workshop: Providing DFaaS with code notebooks

Hvbeek ▾

Code Notebook - iOS battery levels

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
...	...	...	...	...	...	...	...
2022-02-01	2022-02-01	2022-02-01			2022-02-01	40220408-D41E-4026-8B9E-	

MARKDOWN

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

```
1 from js import document
2 from matplotlib import pyplot
3
4 pyplot.figure(figsize=(10, 6))
5 pyplot.scatter(batteryrecords['start'], batteryrecords['battery'], s=5)
6 pyplot.title(f'Battery levels of iPhone from trace: {result.uid}')
7 pyplot.show()
```

PYTHON

Battery levels of iPhone from trace: b94ab972-6bcb-44d6-b08b-d081a33320b2:0-1-0-0-1-0-25-2-0



## 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	<b>Part I</b>
	Introduction
	The Hansken SDK
	Live demonstration of Hansken and the Hansken SDK
10:45	Prepping for the exercises
11:00	Break
<b>11:15</b>	<b>Part II</b>
	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	<b>Part I</b>
	Introduction
	The Hansken SDK
	Live demonstration of Hansken and the Hansken SDK
10:45	Prepping for the exercises
11:00	Break
<b>11:15</b>	<b>Part II</b>
	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