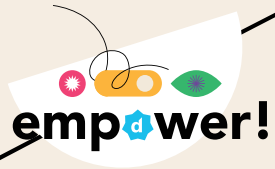


Welc^dme



[YOUR TRAINER THIS SESSION]

Product Owner
Data
Acquisition



Jan-Klaas
Kesteloot

janklaas.kesteloot@skyline.be

Data API &  

-----> Scripted

Connectors

Agenda



Fundamentals track

1. Introduction
2. Data API
3. Scripted Connectors
4. Hands-on!
5. Limitations

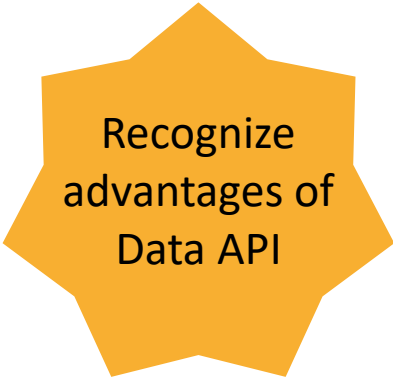


Goals


- Understand the concept of Data API
- Recognize the advantages of Data API
- Create a Scripted Connector



Understand
concept of Data
API



Recognize
advantages of
Data API



Create a
scripted
connector

Introduction

Because we need to start somewhere.. 

Preview Availability



Currently in preview with the **<DataAPI>** soft-launch option enabled.

Introduction

What is it?

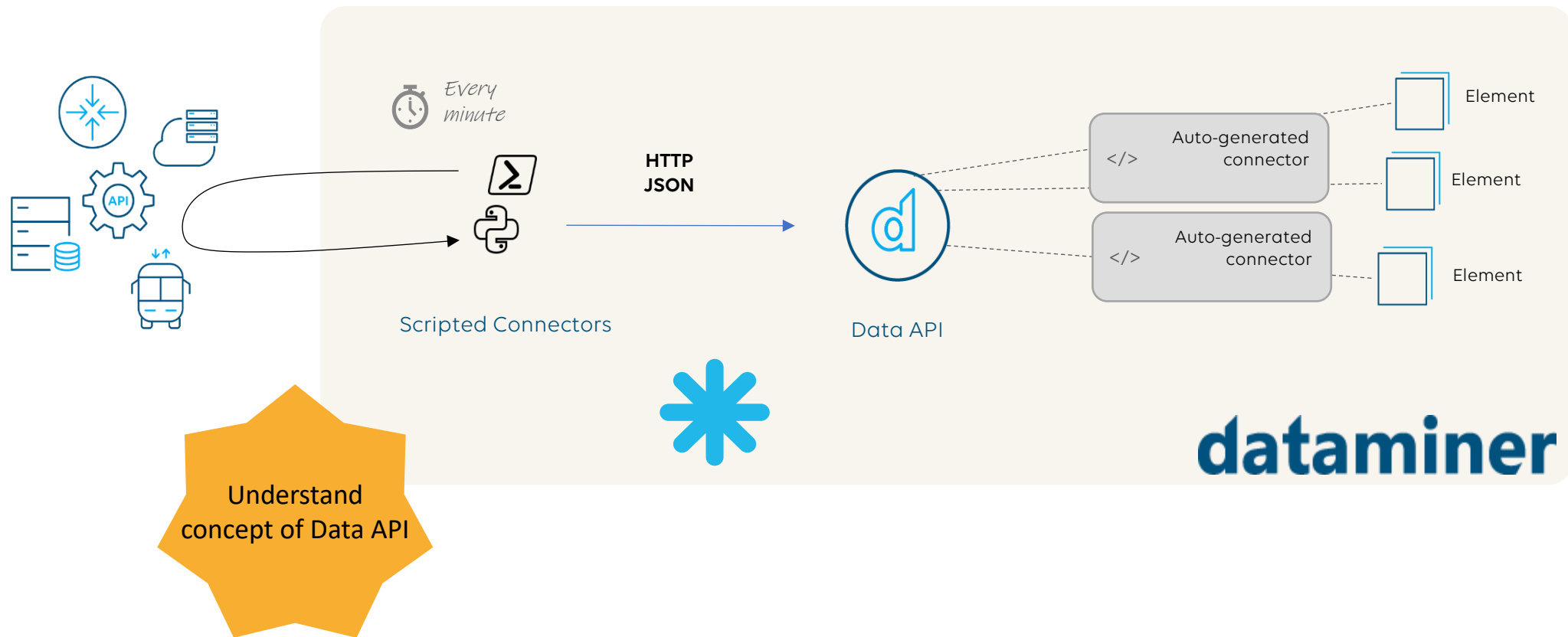
An easy way to script and deploy ad-hoc integrations with products and data-sources in your operation on the fly, without the need for a DataMiner connector

(or the knowledge to implement a connector)

- **Data API:** Ingest the requests, pushing values into parameters of dynamically created elements that rely on automatically generated connector.
- **Scripted Connectors:** Python or PowerShell scripts to integrate new data sources, transmitting JSON data to the Data API.

Introduction

What is it?



Introduction

JSON data handling

- Auto-generate elements with parameters and values
- Automatic updates to parameter values
- Trended by default
- No JSON schema

```
{  
  "Server Name": "WebServer001",  
  "CPU Utilization": 78.5  
}
```

| | | | | |
|---|-----------------|------|-------------|--------------|
| > | Parameters | | | |
| ☐ | CPU Utilization | 78.5 | Server Name | WebServer001 |
| ☐ | | | | |
| ☐ | | | | |

Introduction

Why should you use it

DATA API & SCRIPTED CONNECTORS

- No specific scheme or syntax
- Python & PowerShell - but Data API is basically open for any client language
- Easy to use

STANDARD CONNECTORS

- Adhere to DataMiner Connector Markup Language
- Additional logic in C#
- Support for DCF, SRM Functions

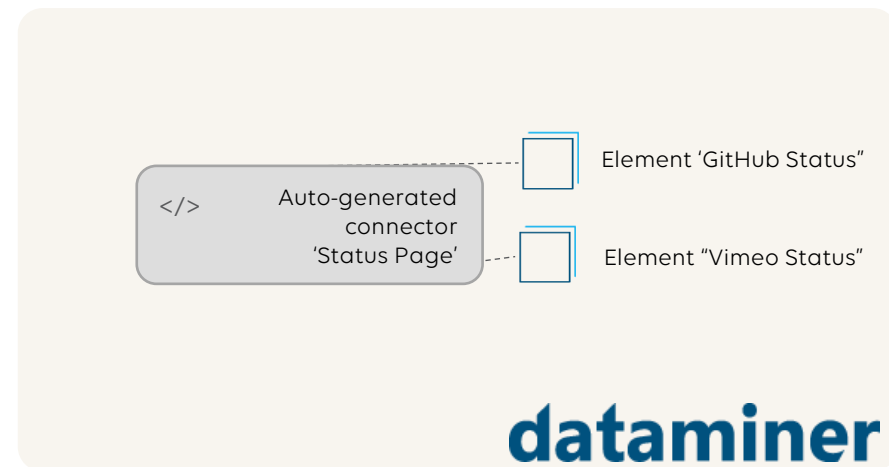
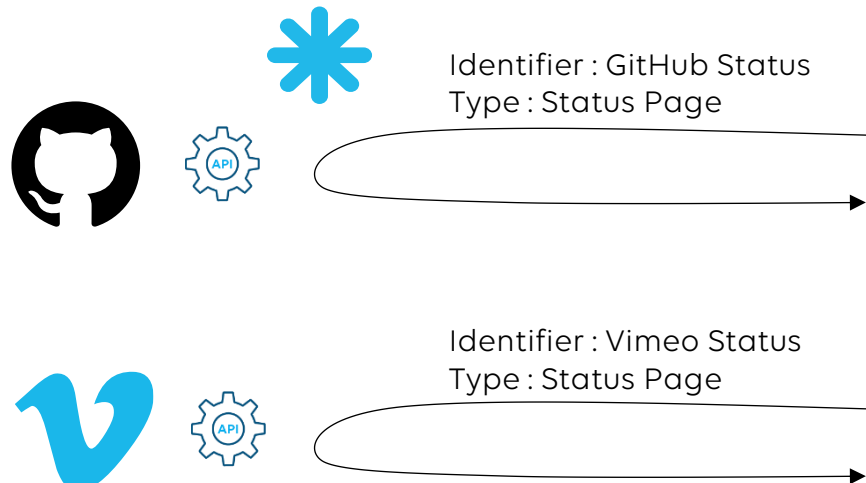
Recognize
advantages of
Data API

Data API

Data API

HTTP API

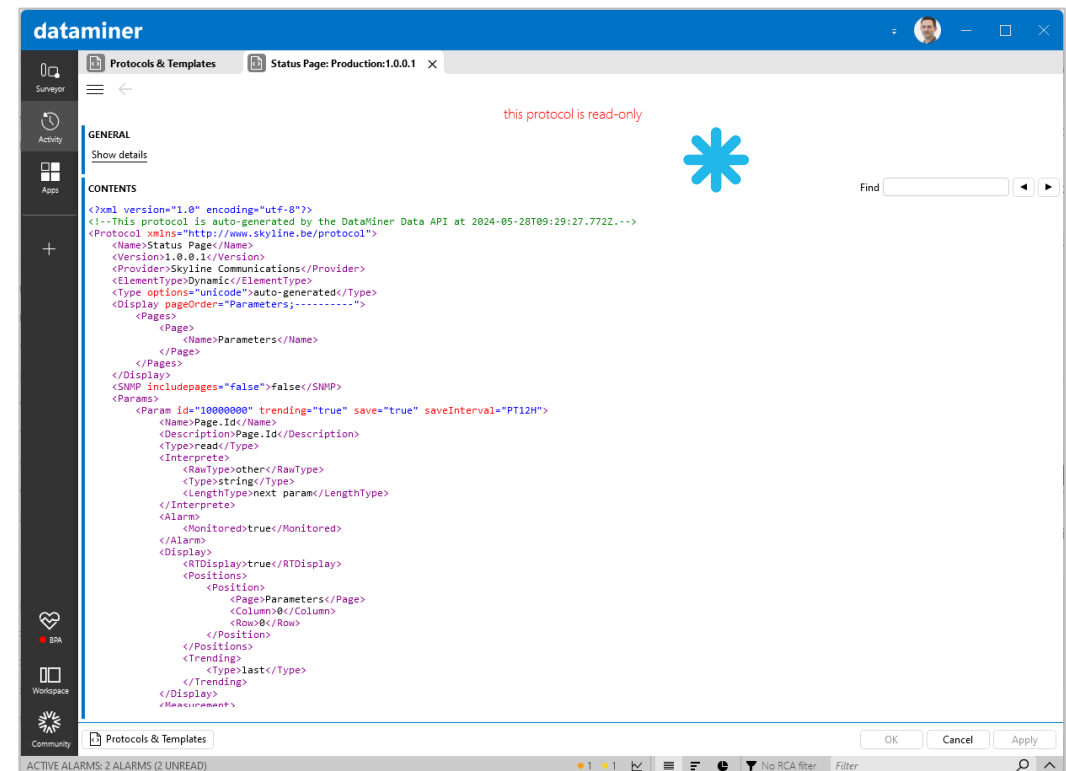
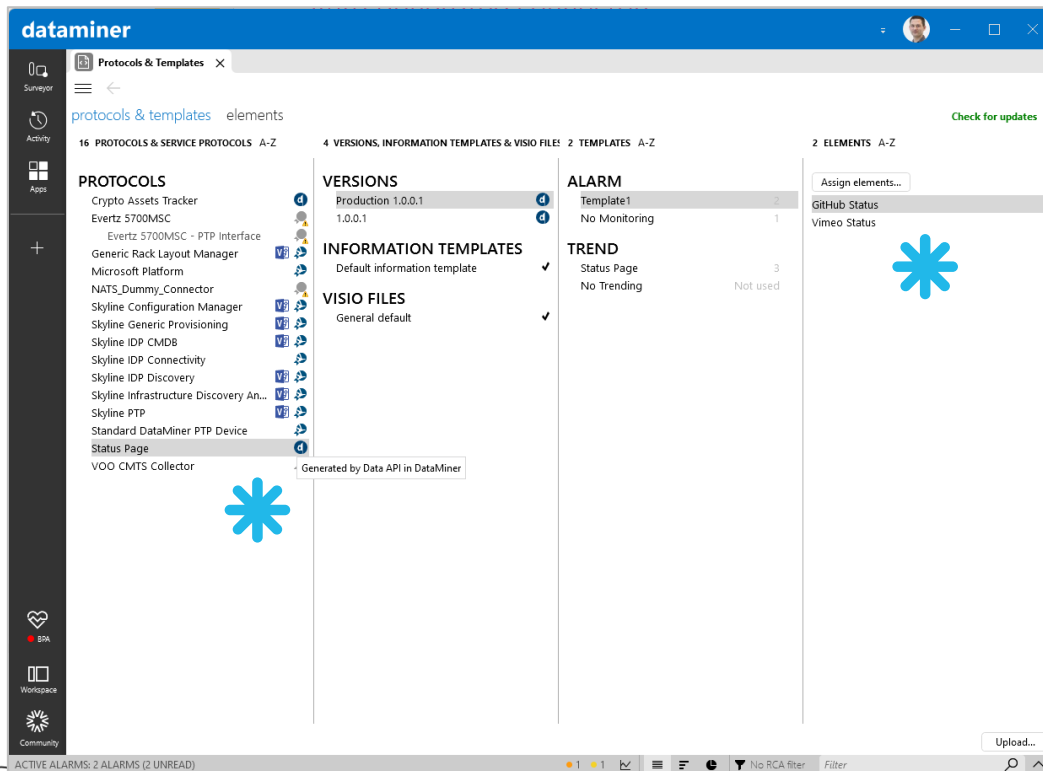
- HTTP endpoint `<host address>/api/data/parameters`
- HTTP Request should include the following HTTP header fields:
 - The **identifier**, is used to identify the source of the request. It must be unique within the DMS cluster. The identifier serves as the initial name of the element, which can be renamed later at any time.
 - The **type** is used to group identifiers of the same type together, making it easier to manage these elements. The type is used as the name of the auto-generated connector.



Data API

Auto-generated connectors

- Data API creates & maintains auto-generated connectors.
- It provisions elements based on the identifiers.



Data API

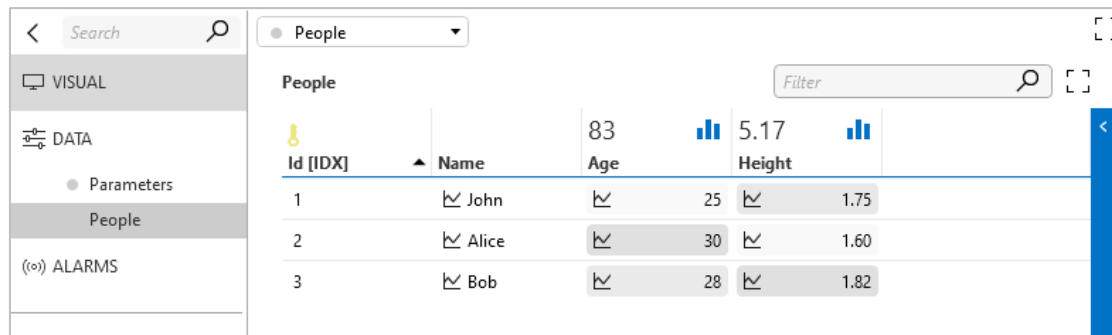
Support for tables

- Convert JSON arrays from HTTP body to table in the element.
- Transform nested arrays into multiple tables with foreign keys.

Data API

Support for tables

- This JSON array is transformed into a table called **People** with columns *ID*, *Name*, *Age*, and *Height*.
- The field *Id* always serves as the primary key for the table.



The screenshot shows a web interface with a sidebar on the left containing 'VISUAL', 'DATA', 'Parameters', 'People', and 'ALARMS'. The main area displays a table titled 'People' with a search bar and a filter icon. The table has columns: Id [IDX], Name, Age, and Height. There are three rows of data.

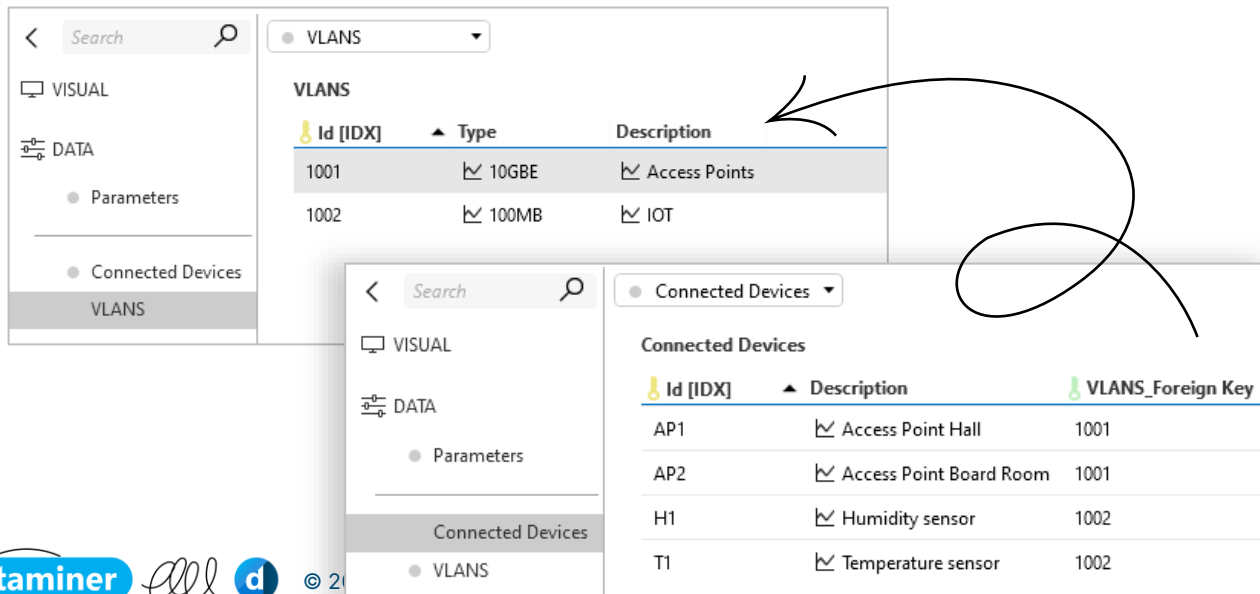
| Id [IDX] | Name | Age | Height |
|----------|-------|-----|--------|
| 1 | John | 25 | 1.75 |
| 2 | Alice | 30 | 1.60 |
| 3 | Bob | 28 | 1.82 |

```
{
  "People": [
    {
      "Id": 1,
      "Name": "John",
      "Age": 25,
      "Height": 1.75
    },
    {
      "Id": 2,
      "Name": "Alice",
      "Age": 30,
      "Height": 1.60
    },
    {
      "Id": 3,
      "Name": "Bob",
      "Age": 28,
      "Height": 1.82
    }
  ]
}
```

Data API

Support for tables

- The information from the VLANS array is distributed into two tables:
 - The **VLANS** table
 - The **Connected Devices** table



The screenshot shows the Data API interface with two tables. The top table is 'VLANS' and the bottom table is 'Connected Devices'. A curved arrow points from the 'Connected devices' field in the VLANS table to the Connected Devices table.

| Id [IDX] | Type | Description |
|----------|-------|---------------|
| 1001 | 10GBE | Access Points |
| 1002 | 100MB | IOT |

| Id [IDX] | Description | VLANS_Foreign Key |
|----------|-------------------------|-------------------|
| AP1 | Access Point Hall | 1001 |
| AP2 | Access Point Board Room | 1001 |
| H1 | Humidity sensor | 1002 |
| T1 | Temperature sensor | 1002 |

```
{
  "Device": "Backbone Switch",
  "Ping": 3,
  "IP": "10.10.10.10",
  "MAC Address": "08-58-F2-F9-36-94",
  "VLANS": [
    {
      "Id": 1001,
      "Type": "10GBE",
      "Description": "Access Points",
      "Connected devices": [
        {
          "Id": "AP1",
          "Description": "Access Point Hall"
        },
        {
          "Id": "AP2",
          "Description": "Access Point Board Room"
        }
      ]
    },
    {
      "Id": 1002,
      "Type": "100MB",
      "Description": "IOT",
      "Connected devices": [
        {
          "Id": "T1",
          "Description": "Temperature sensor"
        },
        {
          "Id": "H1",
          "Description": "Humidity sensor"
        }
      ]
    }
  ]
}
```

Data API

Units & decimal precision

- Default behavior:
 - No units 🤪
 - Decimal precision defined by first data
- Fear not!
 - Reconfigure it by using config endpoint `<host address>/api/config`
 - Use it after a first data request already has been made

| | | | |
|------------------|------------|---------------|-----------------------|
| Parameters ▾ | | [] | |
| Cpuutilization ✓ | 31 | Date ✓ | May 29, 2024 09:55:56 |
| Location ✓ | Izegem | Memoryusage ✓ | 27 |
| Name ✓ | Lab router | Summary ✓ | Lab router device |
| Temperature ✓ | 30.81 | | |



Data API

Units & decimal precision

| | | | |
|----------------|------------|-------------|-----------------------|
| Parameters | | | |
| Cpuutilization | 31 | Date | May 29, 2024 09:55:56 |
| Location | Izegem | Memoryusage | 27 |
| Name | Lab router | Summary | Lab router device |
| Temperature | 30.81 | | |

| | | | |
|----------------|------------|-------------|-----------------------|
| Parameters | | | |
| Cpuutilization | 31.00 % | Date | May 29, 2024 09:55:56 |
| Location | Izegem | Memoryusage | 27 % |
| Name | Lab router | Summary | Lab router device |
| Temperature | 30.8 deg C | | |

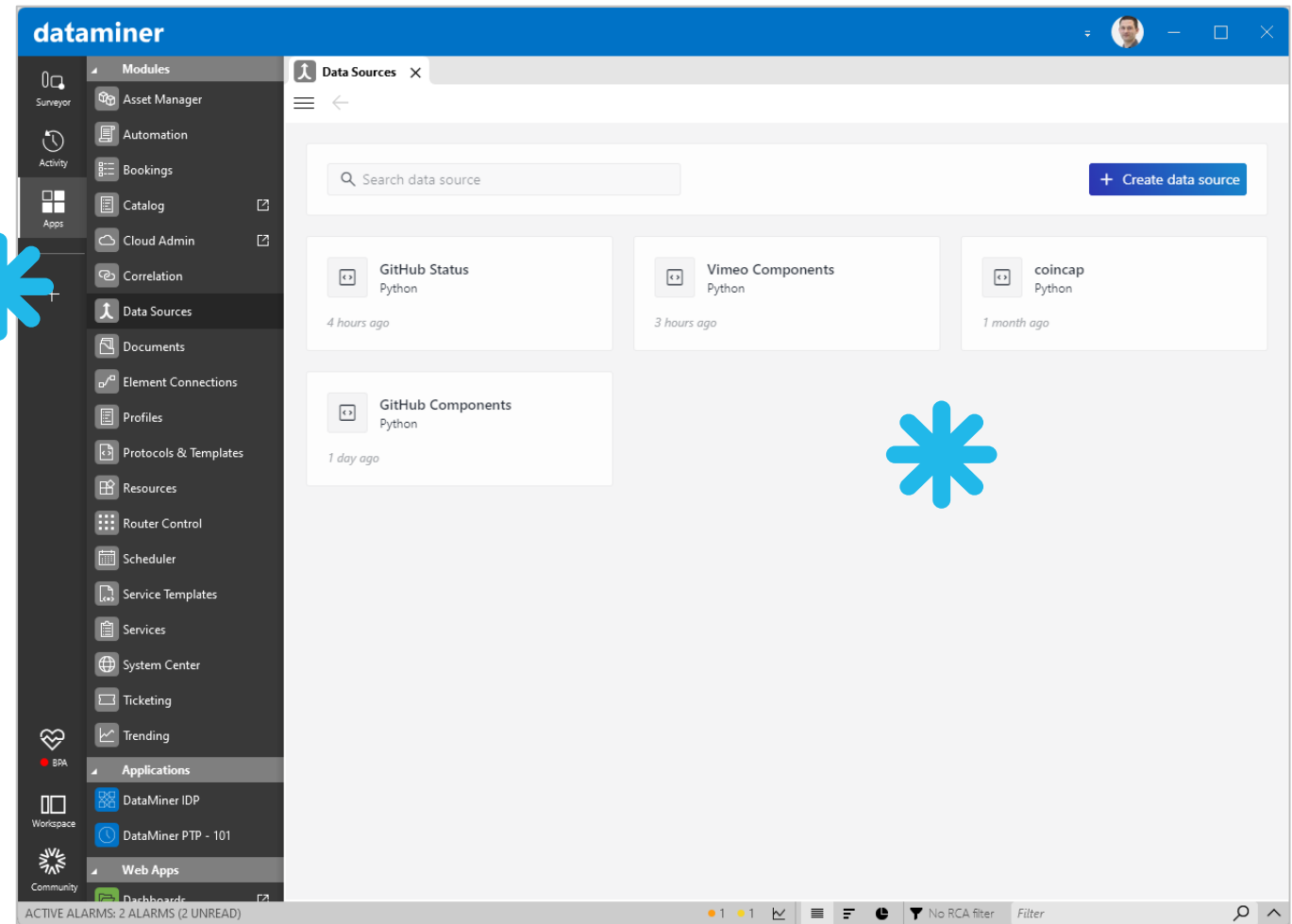
```
1  {
2      "decimals": {
3          "Cpuutilization": 2,
4          "Temperature": 1
5      },
6      "units": {
7          "Cpuutilization": "%",
8          "Temperature": "deg C",
9          "Memoryusage": "%"
10     }
11 }
```

Scripted Connectors

Wait! What!? What's that? A connector? A script?

Scripted Connectors

- Python or PowerShell scripts
- Run on DataMiner system
- Available via module **Data Sources**
- Run every minute



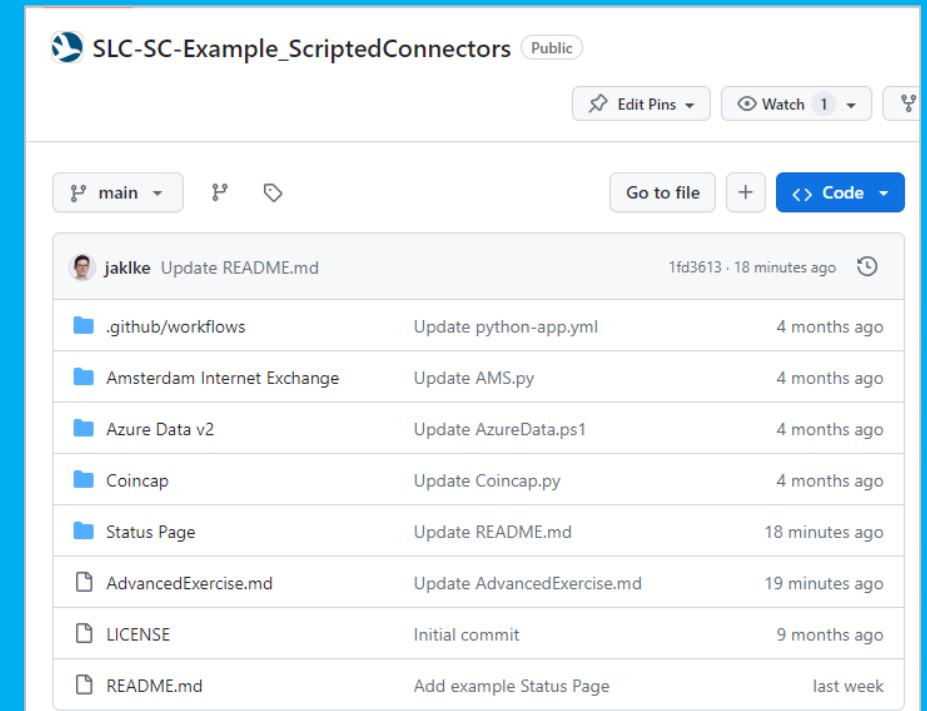
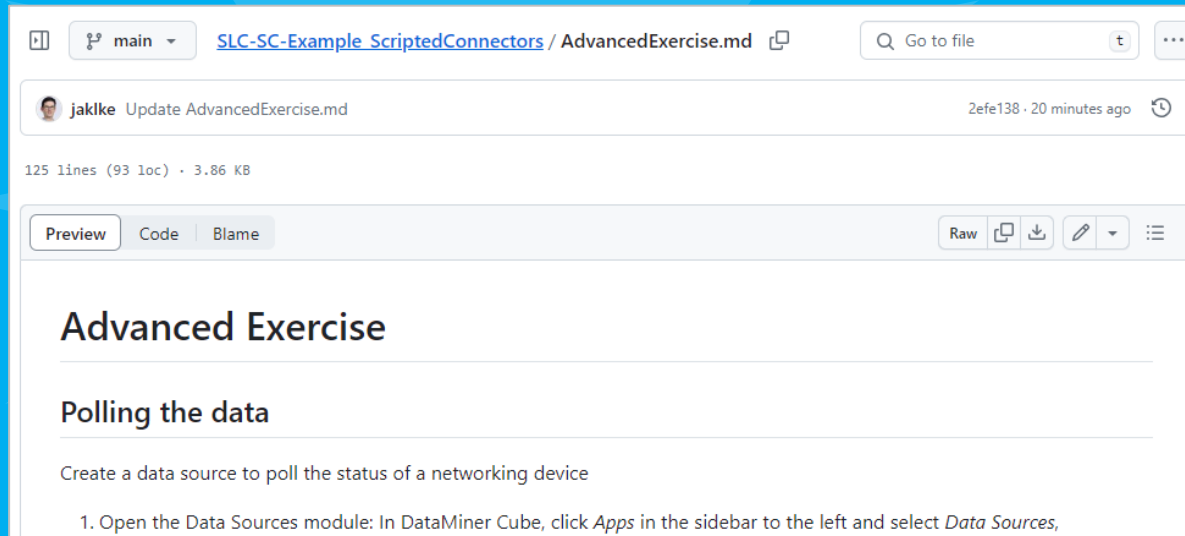
Hands-on!

Steps of the classroom exercise

1. Find & understand the example Python script
2. Understanding expected outcome
3. Change the Python Script with GitHub .dev
4. Create a scripted connector
5. (tweak the trending, configure the alarm thresholds)

Advanced Exercise

1. Go to 'aka.dataminer.services/2'
2. Navigate to **AdvancedExercise.md**



25

Earn 25 DevOps points by emailing screenshots of the **Parameters & Interfaces** page to support.data-acquisition@skyline.be.
Be sure to send them before Monday March 24th , 7 PM CET..



dataminer

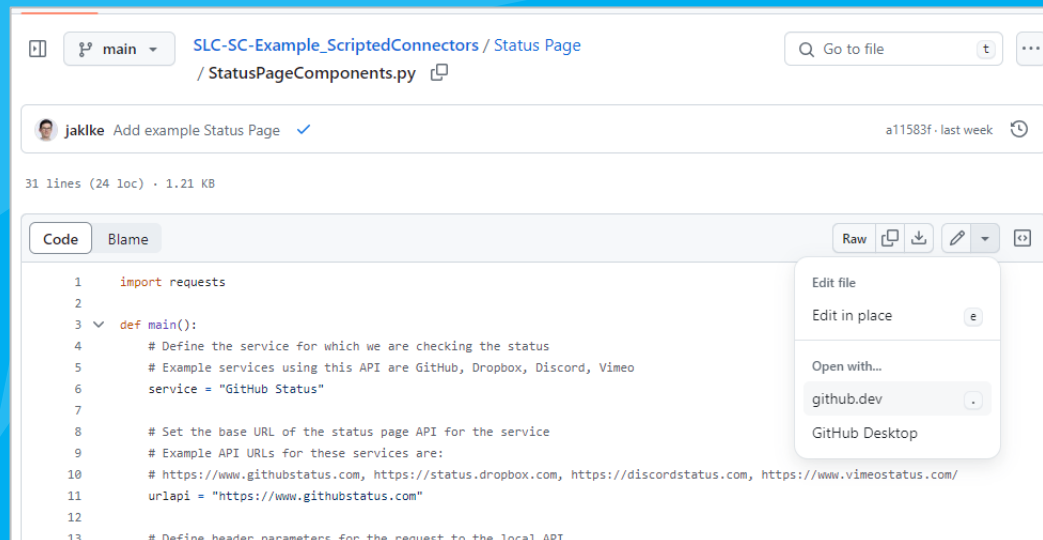


© 2025 Skyline Communications

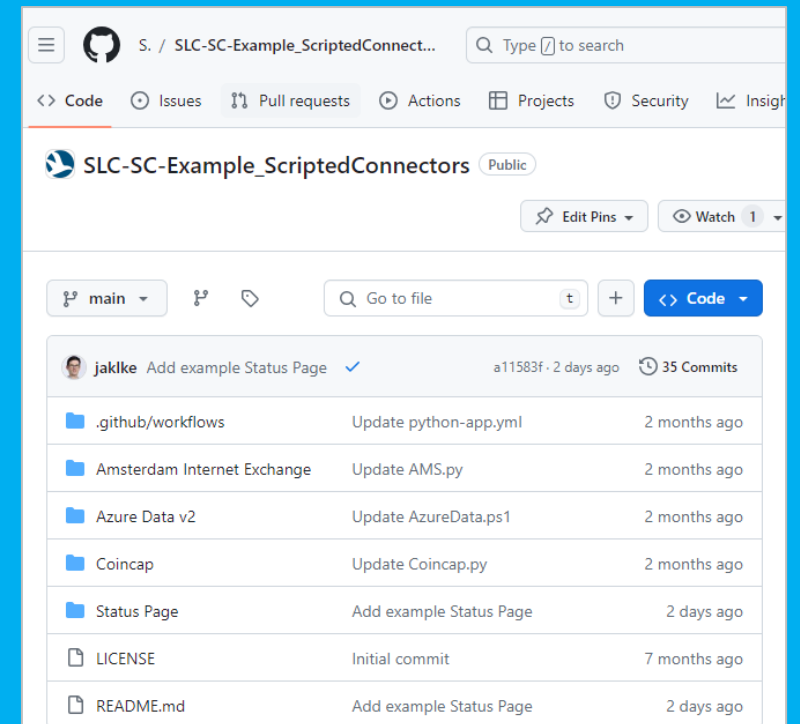
Data API and Scripted Connectors • REV002

Classroom exercise

- Find & Understand
 1. Go to 'aka.dataminer.services/2'
 2. Go to **Status Page** -> StatusPageComponents.py
 3. Open in GitHub .dev

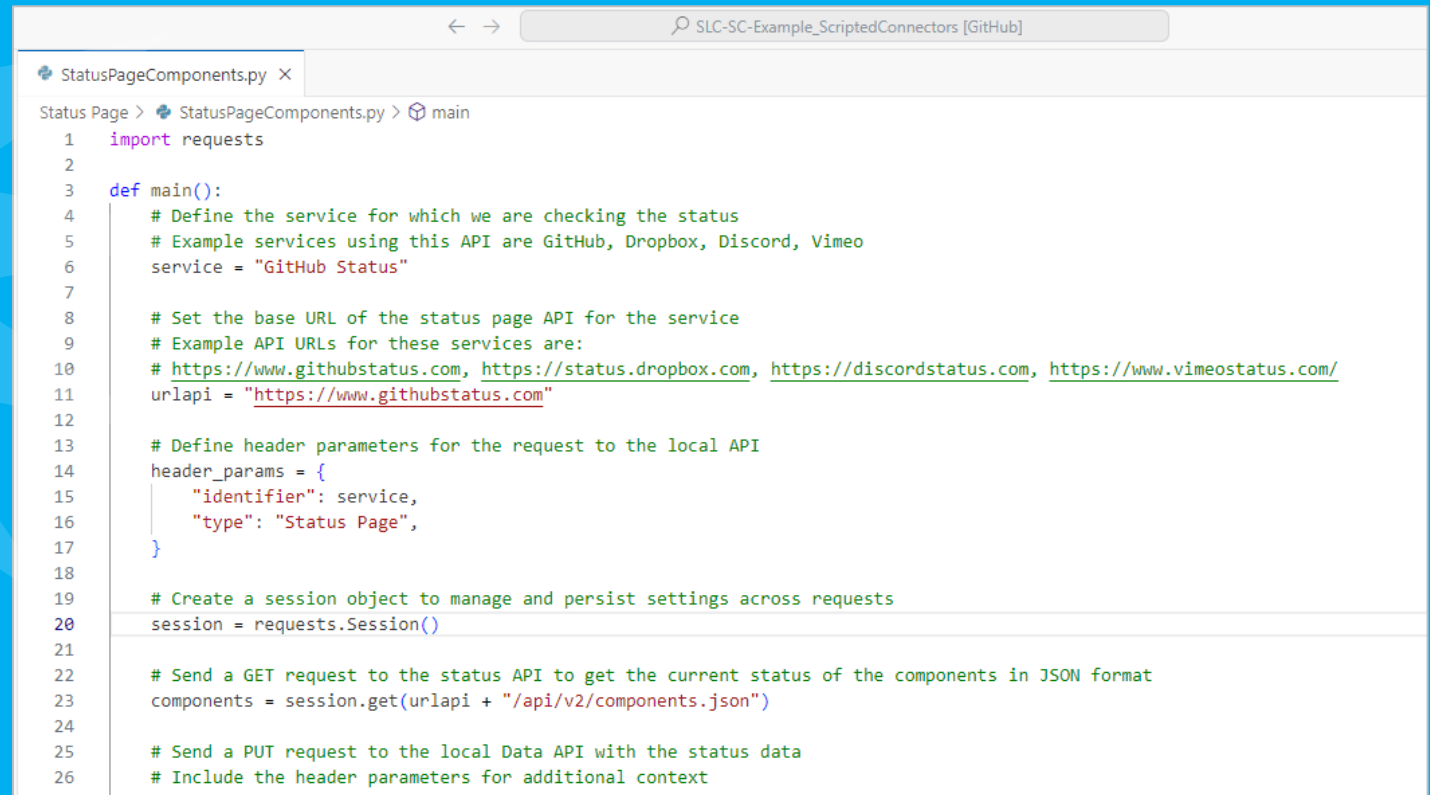


```
1 import requests
2
3 def main():
4     # Define the service for which we are checking the status
5     # Example services using this API are GitHub, Dropbox, Discord, Vimeo
6     service = "GitHub Status"
7
8     # Set the base URL of the status page API for the service
9     # Example API URLs for these services are:
10    # https://www.githubstatus.com, https://status.dropbox.com, https://discordstatus.com, https://www.vimeostatus.com/
11    urlapi = "https://www.githubstatus.com"
12
13    # Define header parameters for the request to the local API
```



Classroom exercise

- Prepare Scripted Connector
 1. Change service name
 2. Copy script to clipboard



```
SLC-SC-Example_ScriptedConnectors [GitHub]
StatusPageComponents.py x
Status Page > StatusPageComponents.py > main
1 import requests
2
3 def main():
4     # Define the service for which we are checking the status
5     # Example services using this API are GitHub, Dropbox, Discord, Vimeo
6     service = "GitHub Status"
7
8     # Set the base URL of the status page API for the service
9     # Example API URLs for these services are:
10    # https://www.githubstatus.com, https://status.dropbox.com, https://discordstatus.com, https://www.vimeostatus.com/
11    urlapi = "https://www.githubstatus.com"
12
13    # Define header parameters for the request to the local API
14    header_params = {
15        "identifier": service,
16        "type": "Status Page",
17    }
18
19    # Create a session object to manage and persist settings across requests
20    session = requests.Session()
21
22    # Send a GET request to the status API to get the current status of the components in JSON format
23    components = session.get(urlapi + "/api/v2/components.json")
24
25    # Send a PUT request to the local Data API with the status data
26    # Include the header parameters for additional context
```

Create scripted connector

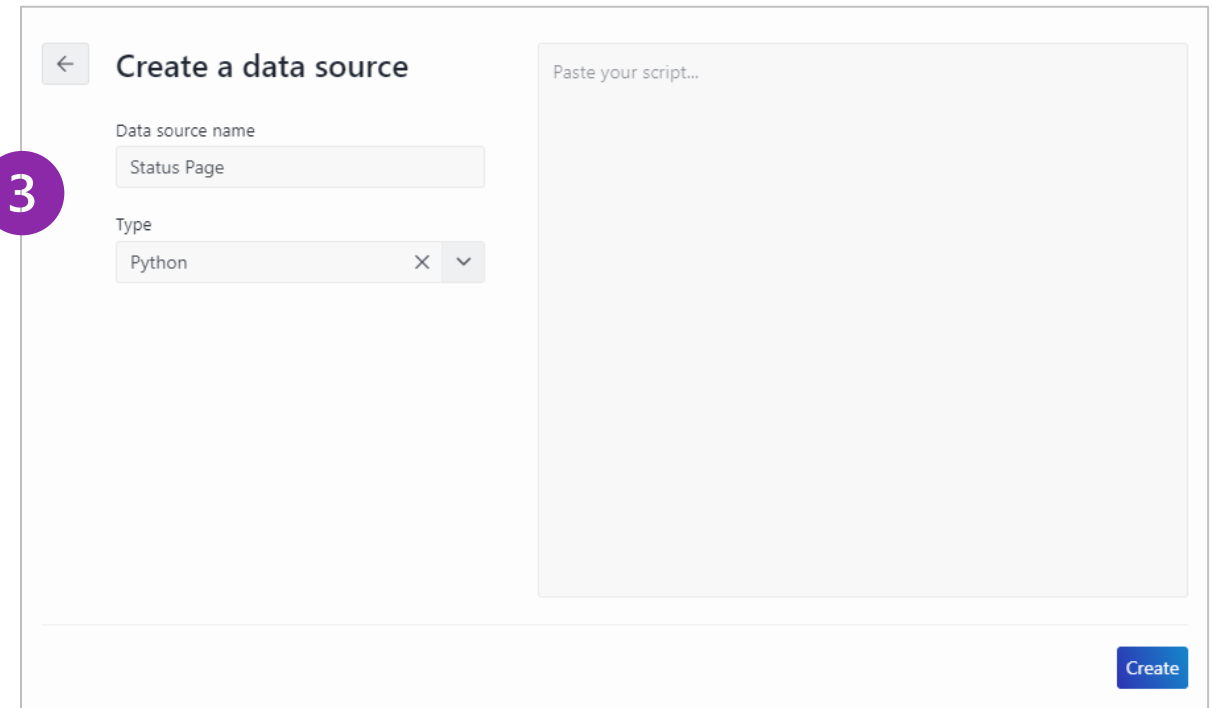
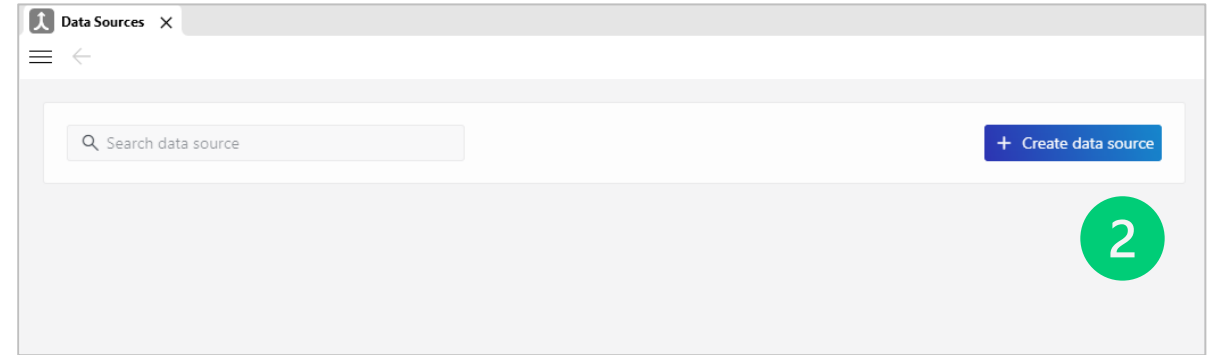
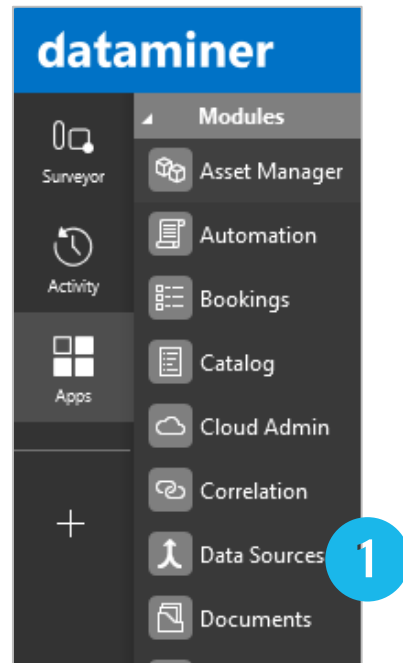
1, 2, 3

1 Open Data Sources

2 Create data source

3 Give in the details

- Name : Status
- Type : Python



Create scripted connector

4, 5, 6

4

Paste Python script

5

Click 'Create'

←

Create a data source

Data source name

Status Page

Type

Python

×

▼

4

```
import requests

def main():
    # Define the service for which we are checking the status
    # Example services using this API are GitHub, Dropbox, Discord, Vimeo
    service = "Dropbox"

    # Set the base URL of the status page API for the service
    # Example API URLs for these services are:
    # https://www.githubstatus.com, https://status.dropbox.com,
    # https://discordstatus.com, https://www.vimeostatus.com/
    urlapi = "https://status.dropbox.com"

    # Define header parameters for the request to the local API
    header_params = {
        "identifier": service,
        "type": "Status Page",
    }

    # Create a session object to manage and persist settings across requests
```

5

Create



Data Sources Dropbox

Root View Dropbox

Search Components

Visual

DATA

Parameters

Components

ALARMS

TICKETS

REPORTS

DOCUMENTS

NOTES

GENERAL PARAMETERS

Components

Filter

67

| Id [IDX] | Name | Status | Created_At | Updated_At | Position | Description | Showcase | Start_Date | Group_Id | Pa |
|--------------|---------------------|----------------------|----------------|----------------|----------|----------------|----------|------------|----------|----|
| 6hfxwc3bylr | Paper | operational | 2017-05-02T... | 2024-05-15T... | 5 | | False | | | |
| 28p7kxfc2400 | Replay | operational | 2022-09-21T... | 2024-04-25T... | 8 | | False | 2022-09-21 | | |
| 0888k9dj2x6x | Support Services | operational | 2020-10-27T... | 2023-11-03T... | 10 | Support ser... | False | 2020-10-27 | | |
| 9249rggclbt | Mobile Application | degraded_performance | 2016-08-12T... | 2024-05-24T... | 3 | | False | | | |
| 6806922ybdI4 | Desktop Application | operational | 2015-09-15T... | 2024-05-28T... | 2 | | False | | | |
| g0f37btjhcq | API | operational | 2015-09-14T... | 2024-05-15T... | 4 | | False | | | |
| j2bmcbh0qddh | Passwords | operational | 2021-01-15T... | 2024-05-15T... | 6 | | False | 2021-01-15 | | |
| jd1xbk17qkv | DocSend | operational | 2023-11-10T... | 2023-12-07T... | 12 | | True | 2023-11-09 | | |
| vtv54lb5d5xc | Capture | operational | 2022-08-18T... | 2024-02-26T... | 7 | | False | 2022-08-18 | | |
| xxdqx0ykqsdv | Website | operational | 2015-09-14T... | 2024-05-30T... | 1 | | False | | | |
| zr2pd7n95xx1 | Dash | operational | 2023-07-06T... | 2024-04-24T... | 9 | | False | 2023-07-06 | | |

25

Earn 25 DevOps points by emailing screenshots of the **Parameters** page to support.data-acquisition@skyline.be.

Be sure to send them before Monday, March 24th , 7 PM CET.

Create a
scripted
connector



dataminer



© 2025 Skyline Communications

Data API and Scripted Connectors • REV002

Limitations

Mind the fine print

Limitations

Data API

- Rejects payloads over 1 MB
- Rate limited to 5 requests per second.
- Local requests only (*)
 - POC remote access via User-Defined API [DataAPI-Proxy | Catalog](#)
- Requires "Id" field in JSON arrays as primary key
- Nested arrays: child table links to one parent table

Parameters in auto-generated connectors

- Automatically assigned to pages in element layout, unmodifiable

Scripted connectors

- Monitoring (no control)
- Fixed frequency: one minute
- No support for arguments
- Stored locally on the server
- No synchronization in DMS cluster

[ASK AWAY]



Questions?

Resources

Getting Started : <https://aka.dataminer.services/scripted-connectors>

Data Sources UI :

- <https://<dataminer>/data-sources>

APIs

- Data API : <http://localhost:34567/swagger/index.html>
- Aggregator : <http://localhost:12345/swagger/index.html>

[SEE YOU AT 01:40 PM]

LUNCH

