Proc HTTP - Das Internet in der Hand

David Weik, SAS

Wie viele Sprachen/Syntaxen gibt es?

- Data Step
- Standard Proc‘s
- SQL
- TPL/PCL
- DS2
- Lua
- Groovy
- X-Kommando
- Java Object
- IML
- IML-R Interface
- REST
- SCL
- CASL
- Macro
- Optmodel
- LITI
- Proc FCMP
- Proc Python
- …

Wie viele Sprachen/Syntaxen gibt es?

- Data Step
- Standard Proc‘s
- SQL
- TPL/PCL
- DS2
- Lua
- Groovy
- X-Kommando
- Java Object
- IML
- IML-R Interface
- REST
- SCL
- CASL
- Macro
- Optmodel
- LITI
- Proc FCMP
- Proc Python
- …

Proc HTTP

Warum eigentlich?

APIs abrufen, zusätzliche Datenquellen nutzen

Daten/Ergebnisse weitergeben

Prozessautomatisierung

…

Anatomie eines REST Aufrufes

HTTP **verbs** for REST APIs

GET – fetches a record or set of resources from the server

POST – creates a new set of resources or a resource

PUT – updates or replaces the given record

DELETE – deletes the given resource

Sample REST API **call** (aka, **endpoint**)

Send metadata, such as request & response data types, as **headers**

Authorization: bearer=<access token>

Content-Type: application/json

Accept: application/json

Send pertinent data in the REST call **body**

```
{data: {
  id: 24,
  title: 'Behavior-Driven Development Report',
  author: 'Viktor Ali'}
 }
```

Der aktuelle Bitcoin Preis

```
filename res temp;

proc http
        url='https://api.coindesk.com/v1/bpi/currentprice.json'
        out=res;
run;
```

Wie gehe ich mit JSON um?

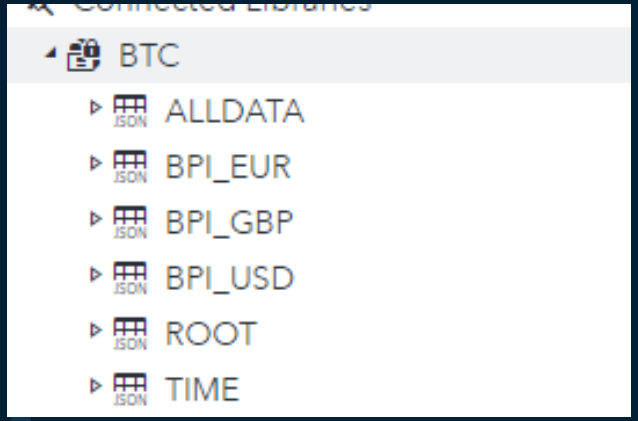
{"time": {"updated": "Mar 29, 2023 08:54:00 UTC","updatedISO": "2023-03-29T08:54:00+00:00","updateduk": "Mar 29, 2023 at 09:54 BST"},"disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion rate from openexchangerates.org","chartName": "Bitcoin","bpi": {"USD": {"code": "USD","symbol": "$","rate": "28,508.5465","description": "United States Dollar","rate_float": 28508.5465},"GBP": {"code": "GBP","symbol": "£","rate": "23,821.5134","description": "British Pound Sterling","rate_float": 23821.5134},"EUR": {"code": "EUR","symbol": "€","rate": "27,771.4865","description": "Euro","rate_float": 27771.4865}}}

Libname Engine JSON

Einlesen von JSON Dateien

libname libref JSON <Pfadangabe, Optionen> ;

libname btc json fileref=res;



```
{
  "time": {
    "updated": "Mar 29, 2023 08:54:00 UTC",
    "updatedISO": "2023-03-29T08:54:00+00:00",
    "updateduk": "Mar 29, 2023 at 09:54 BST"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "28,508.5465",
      "description": "United States Dollar",
      "rate_float": 28508.5465
    },
    "GBP": {
      "code": "GBP",
      "symbol": "&pound;",
      "rate": "23,821.5134",
      "description": "British Pound Sterling",
      "rate_float": 23821.5134
    },
    "EUR": {
      "code": "EUR",
      "symbol": "&euro;",
      "rate": "27,771.4865",
      "description": "Euro",
      "rate_float": 27771.4865
    }
  }
}
```

ALLDATA

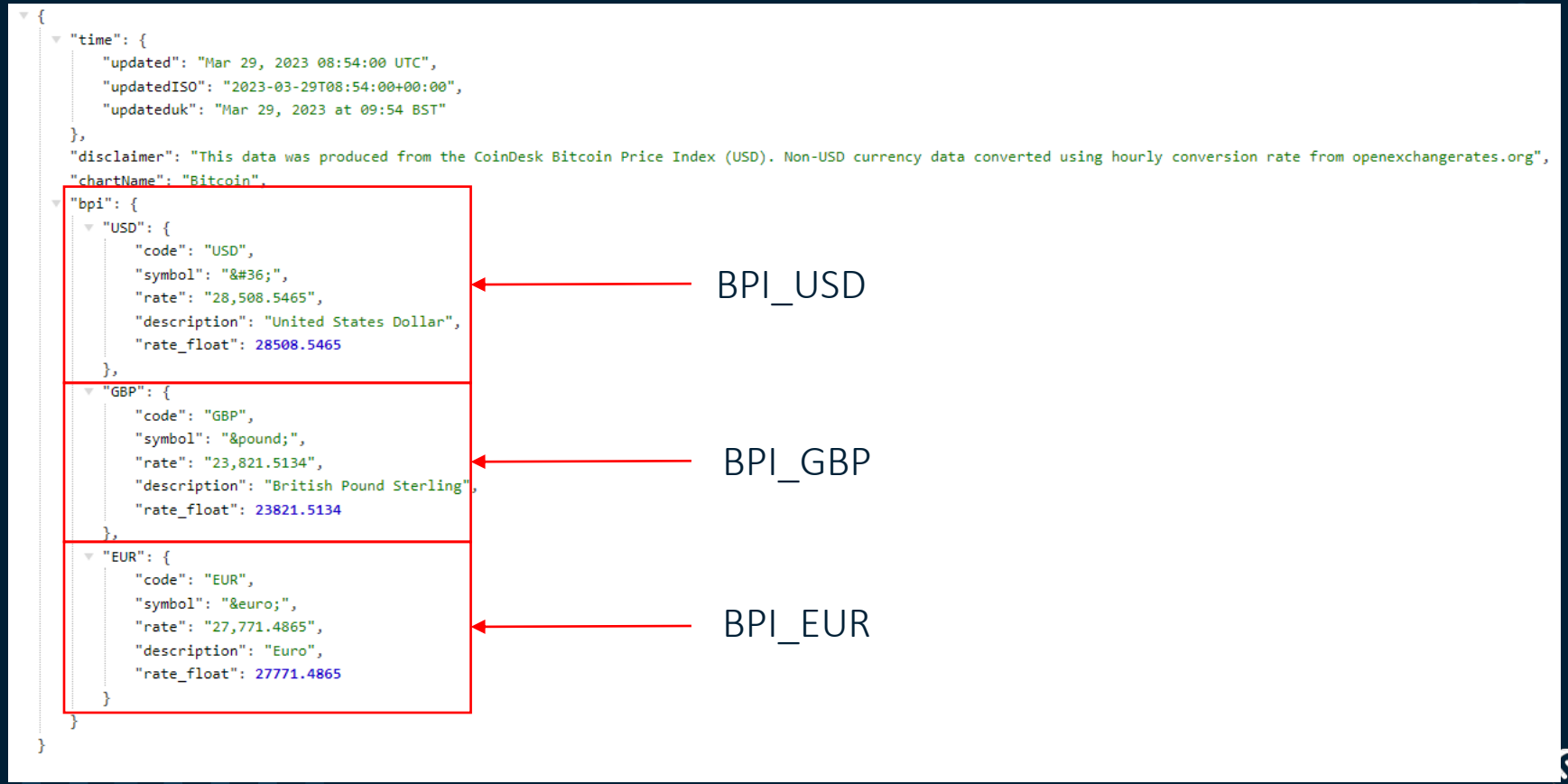

```
{
  "time": {
    "updated": "Mar 29, 2023 08:54:00 UTC",
    "updatedISO": "2023-03-29T08:54:00+00:00",
    "updateduk": "Mar 29, 2023 at 09:54 BST"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "28,508.5465",
      "description": "United States Dollar",
      "rate_float": 28508.5465
    },
    "GBP": {
      "code": "GBP",
      "symbol": "&pound;",
      "rate": "23,821.5134",
      "description": "British Pound Sterling",
      "rate_float": 23821.5134
    },
    "EUR": {
      "code": "EUR",
      "symbol": "&euro;",
      "rate": "27,771.4865",
      "description": "Euro",
      "rate_float": 27771.4865
    }
  }
}
```

ROOT


```
{
  "time": {
    "updated": "Mar 29, 2023 08:54:00 UTC",
    "updatedISO": "2023-03-29T08:54:00+00:00",
    "updateduk": "Mar 29, 2023 at 09:54 BST"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "28,508.5465",
      "description": "United States Dollar",
      "rate_float": 28508.5465
    },
    "GBP": {
      "code": "GBP",
      "symbol": "&pound;",
      "rate": "23,821.5134",
      "description": "British Pound Sterling",
      "rate_float": 23821.5134
    },
    "EUR": {
      "code": "EUR",
      "symbol": "&euro;",
      "rate": "27,771.4865",
      "description": "Euro",
      "rate_float": 27771.4865
    }
  }
}
```

TIME


```
{
  "time": {
    "updated": "Mar 29, 2023 08:54:00 UTC",
    "updatedISO": "2023-03-29T08:54:00+00:00",
    "updateduk": "Mar 29, 2023 at 09:54 BST"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "28,508.5465",
      "description": "United States Dollar",
      "rate_float": 28508.5465
    },
    "GBP": {
      "code": "GBP",
      "symbol": "&pound;",
      "rate": "23,821.5134",
      "description": "British Pound Sterling",
      "rate_float": 23821.5134
    },
    "EUR": {
      "code": "EUR",
      "symbol": "&euro;",
      "rate": "27,771.4865",
      "description": "Euro",
      "rate_float": 27771.4865
    }
  }
}
```



Start Page | * SAS Program.sas | BTC.BPI_EUR ×

BPI_EUR

Table rows: 1 | Columns: 7 of 7 | Rows 1 to 1

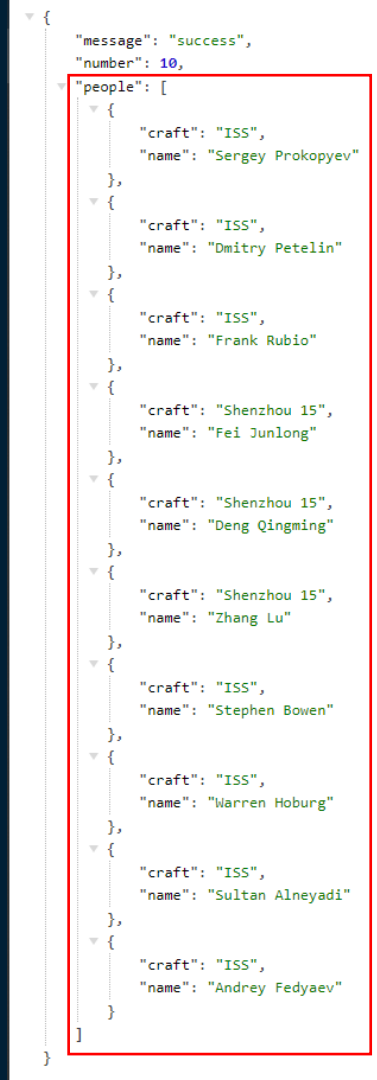
Enter expression

	ordinal_bpi	ordinal_EUR	code	symbol	rate	description	rate_float
1	1	1	EUR	€	27,615.2689	Euro	27615.2689

```
filename astro temp;

proc http
        url='http://api.open-notify.org/astros.json'
        out=astro;
run;

libname astro json;
```



Start Page | * SAS Program.sas | ASTRO.PEOPLE ×

PEOPLE

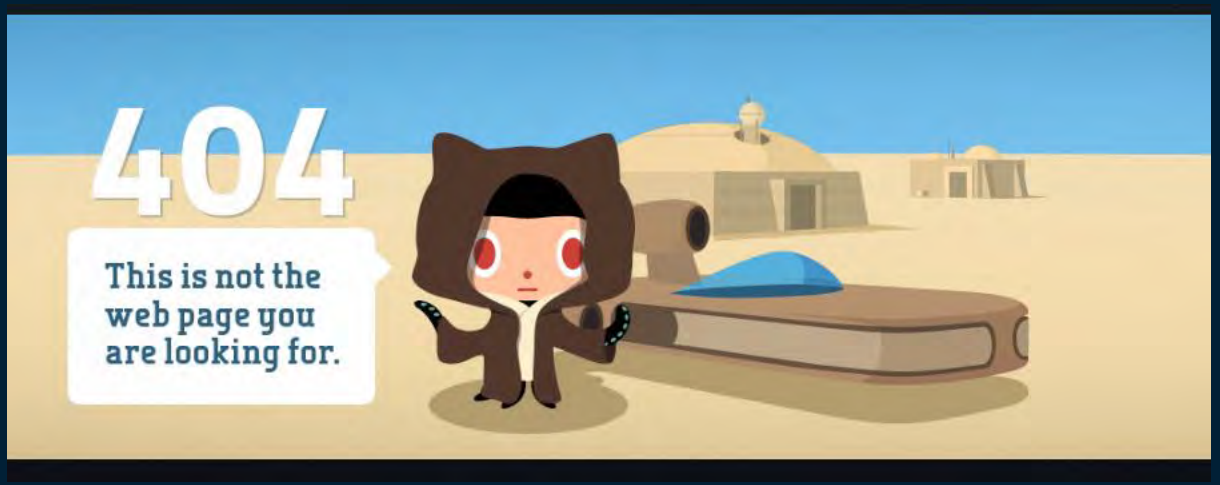
Table rows: 10 | Columns: 4 of 4 | Rows 1 to 10

Enter expression

	ordinal_root	ordinal_people	craft	name
1	1	1	ISS	Sergey Prokopyev
2	1	2	ISS	Dmitry Petelin
3	1	3	ISS	Frank Rubio
4	1	4	Shenzhou 15	Fei Junlong
5	1	5	Shenzhou 15	Deng Qingming
6	1	6	Shenzhou 15	Zhang Lu
7	1	7	ISS	Stephen Bowen
8	1	8	ISS	Warren Hoburg
9	1	9	ISS	Sultan Alneyadi
10	1	10	ISS	Andrey Fedyaev

Status Code Handling

Nicht jede Anfrage ist erfolgreich



Macro Variablen Nutzen

- sys_prochttp_status_code – in unserem vorherigen Fall: 404
- sys_prochttp_status_phrase – in unserem vorherigen Fall: Not Found

```
proc http
        url='https://github.com/404';
run;

%put &=sys_prochttp_status_code.;
%put &=sys_prochttp_status_phrase.;
```

Macro Variablen Nutzen

- sys_prochttp_status_code – in unserem vorherigen Fall: 404
- sys_prochttp_status_phrase – in unserem vorherigen Fall: Not Found

Macro Variablen Nutzen

- sys_prochttp_status_code – in unserem vorherigen Fall: 404
- sys_prochttp_status_phrase – in unserem vorherigen Fall: Not Found

```
proc http
        url='https://github.com/404';
run;

%put &=sys_prochttp_status_code.;
%put &=sys_prochttp_status_phrase.;
```

```
80   proc http
81   url='https://github.com/404';
82   run;
NOTE: 404 Not Found
NOTE: PROCEDURE HTTP used (Total process time):
      real time           0.20 seconds
      cpu time            0.01 seconds

83
84   %put &=sys_prochttp_status_code.;
SYS_PROCHTTP_STATUS_CODE=404
85   %put &=sys_prochttp_status_phrase.;
SYS_PROCHTTP_STATUS_PHRASE=Not Found
```

Jetzt mal schönere

```
* Return Code Handling;
data _null_;
    if &sys_prochttp_status_code. < 400 then do;
        putlog "NOTE: The request was succesfull, it returned with &sys_prochttp_status_code.: &sys_prochttp_status_phrase..";
    end;
    else if &sys_prochttp_status_code. > 500 then do;
        putlog 'ERROR: The request was unsuccsfull, because of a server error';
        putlog "ERROR: The request returned with &sys_prochttp_status_code.: &sys_prochttp_status_phrase..";
    end;
    else if &sys_prochttp_status_code. > 400 then do;
        putlog 'ERROR: The request was unsuccsfull, because of an error in the request';
        putlog "ERROR: The request returned with &sys_prochttp_status_code.: &sys_prochttp_status_phrase..";
    end;
    else do;
        putlog "ERROR: Unkown Response - &sys_prochttp_status_code.: &sys_prochttp_status_phrase..";
    end;
run;
```

Jetzt mal schönere

```
* Return Code Handling;
data _null_;
    if &sys_
        putl
    end;
    else if
        putl
        putl
    end;
    else if
        putl
        putl
    end;
    else do;
        putlog "ERROR: Unkown Response - &sys_prochttp_status_code.: &sys_prochttp_status_phrase..";
    end;
run;
```

```
95   end;
96   run;
ERROR: The request was unsuccsfull, because of an error in the request
ERROR: The request returned with 404: Not Found.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

cURL zu Proc HTTP

Dokumentationsstandard cURL

curl --location 'https://api.coindesk.com/v1/bpi/currentprice.json‘

Wie komme ich davon nach proc http?

Dokumentationsstandard cURL

curl --location 'https://api.coindesk.com/v1/bpi/currentprice.json‘

Wie komme ich davon nach proc http?

1. Zerlegen der Request in die Einzelteile

Dokumentationsstandard cURL

curl --location 'https://api.coindesk.com/v1/bpi/currentprice.json‘

Wie komme ich davon nach proc http?

1. Zerlegen der Request in die Einzelteile
2. Übersetzen der Einzelteile in proc http statements

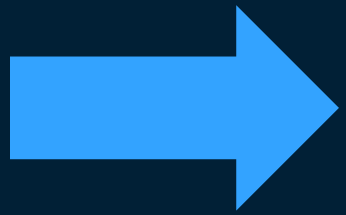
Dokumentationsstandard cURL

curl --location 'https://api.coindesk.com/v1/bpi/currentprice.json‘

Wie komme ich davon nach proc http?

1. Zerlegen der Request in die Einzelteile
2. Übersetzen der Einzelteile in proc http statements
3. Zusammensetzen der Einzelteile

curl --location 'https://api.coindesk.com/v1/bpi/currentprice.json‘



```
* Create a temporary file;
filename outResp temp;

* Actual Proc HTTP Request;
proc http
url='https://api.coindesk.com/v1/bpi/currentprice.json'
out=outResp;
run;
```

cURL to Proc HTTP

Den Code gibt es auf GitHub: https://github.com/Criptic/sas_snippets/blob/master/Convert-cURL-to-Proc-HTTP.sas

Mein Blog dazu: https://www.davidweik.org/blog/from-curl-to-proc-http

Proc HTTP und Python requests

```
filename res temp;
proc http
        url =
'https://api.coindesk.com/v1/bpi/
currentprice.json'
        out = res;
run;


libname res json;
```

```
import requests

res = requests.request('GET',
'https://api.coindesk.com/v1/bpi/
currentprice.json')

res.json()
```

Proc HTTP und Python requests

TL;DR sehr ähnlich

Proc HTTP

- Automatische Variablen für Status Codes
- Unterstützung für alle gängigen Methoden
- Explizite Definition der Ausgabe
- Keine Interaktivität
- Zusätzliche Konstrukte für Pagination, Output-Handling

requests

- Automatische Variable für Status Codes
- Unterstützung für alle gängigen Methoden
- Implizite Definition der Ausgabe
- Keine Interaktivität
- Zusätzliche Konstrukte für Pagination, Output-Handling
- Zusätzliche Tools wie bs4, selenium ermöglichen viel Komfort und eröffnen mehr Möglichkeiten

Zusammenfassung

- Proc HTTP bietet alles um erfolgreich APIs mit SAS zu integrieren
- Durch die Kombination mit der libname engine JSON können Rückgabewerte leicht verarbeitet werden
- Das Filename statement eröffnet aber für die speichern von Rückgaben noch mehr Möglichkeiten
- Fehlerbehandlung ist durch das Abprüfen von Macrovariablen einfach
- Pagination kann durch Macros abgebildet werden

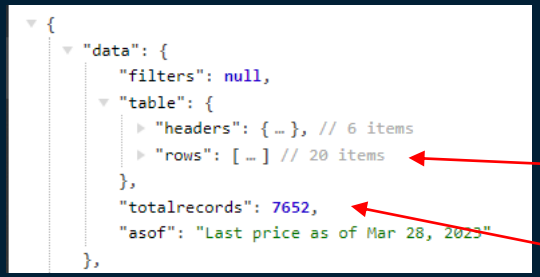
Weiterführend

Mehr Codebeispiele gibt es auf GitHub:
https://github.com/Criptic/sas_snippets/tree/master/AtE-Proc-HTTP

Und ein Webinar zu dem Thema gibt es hier:
https://www.sas.com/de_de/webinars/ask-the-expert-serie-wie-kann-ich-mit-sas-eine-api-aufrufen.html

Zusatz Pagination

API liefert nur ein Teil



20 Sätze in der Antwort, aber eigentlich über 7.000

Wir müssen blättern

Dazu müssen wir die gesamt Anzahl extrahieren (7652)

Durch die Anzahl pro Anfrage teilen (20)

Dann die Anfrage n mal wiederholen (7652 / 20 =~ 383)

Bei jeder Anfrage mitgeben wo wir zuletzt waren (i * 20)

Alles Zusammen

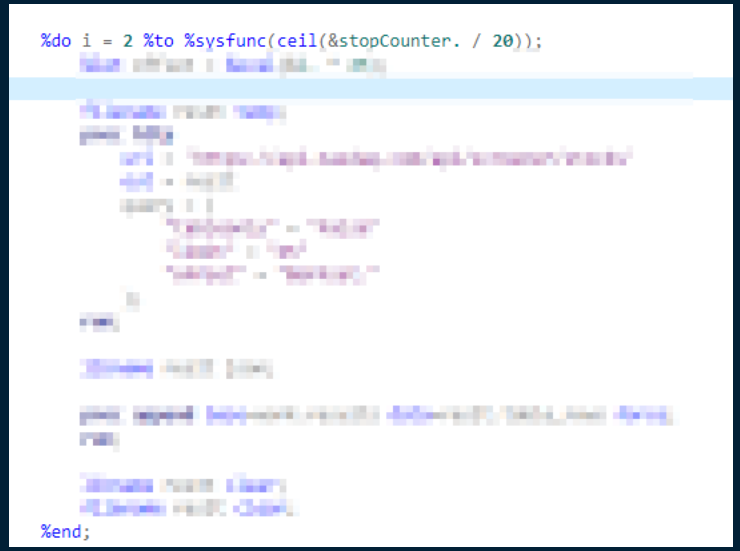
Schritt 1

```
data _null_;
    set res.alldata(where=(p2='totalrecords'));
    call symputx ('stopCounter', value);
run;
```

Schritt 2

```
ceil(&stopCounter. / 20));
```

Schritt 3



Schritt 4

```
%let offset = %eval(&i. * 20);
```

Alles Zusammen

```
%do i = 2 %to %sysfunc(ceil(&stopCounter. / 20));
    %let offset = %eval(&i. * 20);

    filename resIt temp;
    proc http
        url = 'https://api.nasdaq.com/api/screener/stocks'
        out = resIt
        query = (
            "tableonly" = "false"
            "limit" = "20"
            "offset" = "&offset."
        );
    run;

    libname resIt json;

    proc append base=work.results data=resIt.table_rows force;
    run;

    libname resIt clear;
    filename resIt clear;
%end;
```