The Boston Area SAS® Users Group

Gathering SAS Users Since 1983

How Do You Use SAS® to Access Data and APIs From the Web?

Chris Hemedinger is the Director of SAS User Engagement. His talented team looks after SAS online communities, SAS user groups, developer experience and GitHub, tech newsletters, expert webinars and tutorials. Chris is a recovering software developer who helped build popular SAS products such as SAS Enterprise Guide. Inexplicably, Chris is still coasting on the limited fame he earned as an author of SAS For Dummies. You can follow Chris on Twitter as @cjdinger.

How Do You Use SAS to Access Data and APIs From the Web?

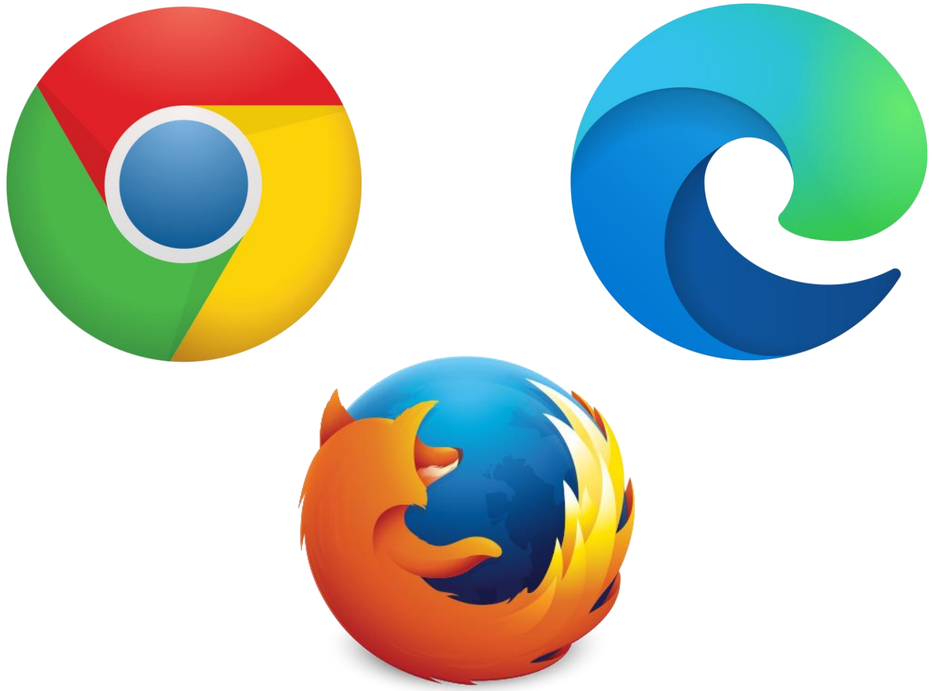
For BASUG

Chris Hemedinger, Director of SAS User Engagement

The Internet: Not just for web pages

And content is not just for web browsers

For end users



For automated processes

curl://

POSTMAN

```
FILENAME URL
PROC HTTP
```

Any application that connects to a web site/service with HTTP is a web client

Today's topics

- Intro to FILENAME URL and PROC HTTP
- Use case: Download a file and read it with SAS
- Use case: Download a file with user/password
- Use case: Download and read a ZIP file in SAS
- Use case: Post data to a site via web form
- Use case: "Scrape" data from a web page
- Use case: Call REST API with JSON response

FILENAME URL

Simple. But limited.

```
filename data URL "https://sitewithdata.com/path-to-data";
```

Example:

```
filename data URL
"https://www.federalreserve.gov/paymentsystems/files/coin_currcircvolume.txt";
```

- One line!
- Supports HTTP GET method only
- Supports basic options: user/password, proxy*, and a few others
- No control over where SAS stores the data that's fetched

PROC HTTP

More lines of code. More power.

```
proc http
 method="method"
 url="http://sitewithdata.com/path"
 out=fileref;
```

Example:

```
filename data "/local/path/save-the-data-file";

proc http
 method="GET"
 url =
  "https://www.federalreserve.gov/paymentsystems/files/coin_currcircvolume.txt"
 out=data;

run;
```

```
proc http
 method="method"
 url="http://sitewithdata.com/"
 out=fileref;
```

DEMO

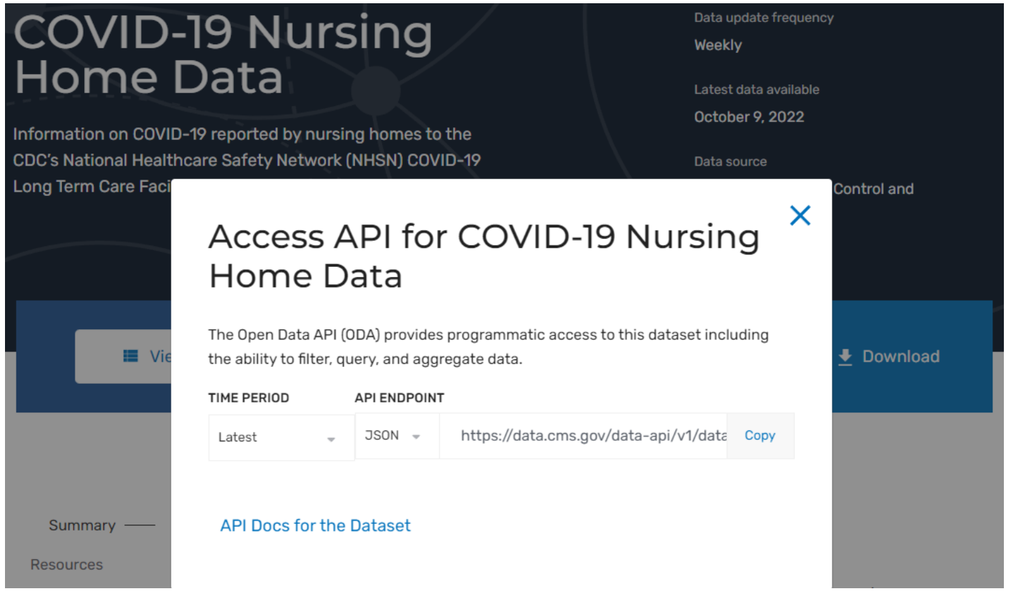
PROC HTTP

More lines. More power.

- Supports HTTP GET, POST, PUT, and any other method
- Provides (requires) control for location of output
- Many options for authentication, caching, cookie behavior – aspects of a full HTTP client
- Rich DEBUG support
- SAS language analogy to cURL

Use case: Download a data file

Data from CMS.gov




```
filename nh temp;
proc http
 url="https://data.cms.gov/data-api/v1/dataset/137f90cb-
ac53-4b3d-8358-e65cf64e03d3/data"
 method="GET"
 out=nh;
run;

options validvarname=v7;
libname cms json fileref=nh;

data records;
 set cms.root;
run;

title; footnote;
proc freq data=records order=freq;
 table provider_name;
run;
```

Download and import: Result

Log | Output Data (1)

Where | Query Builder | Tasks

RECORDS

	ordinal_root	week_ending	federal_provider_number	provider_name	provider_address	provider_city	provider_state
1	1	2020-05-24	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
2	2	2020-05-31	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
3	3	2020-06-07	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
4	4	2020-06-14	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
5	5	2020-06-21	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
6	6	2020-06-28	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
7	7	2020-07-05	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
8	8	2020-07-12	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
9	9	2020-07-19	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL
10	10	2020-07-26	015009	BURNS NURSING HOME, INC.	701 MONROE STREET NW	RUSSELLVILLE	AL

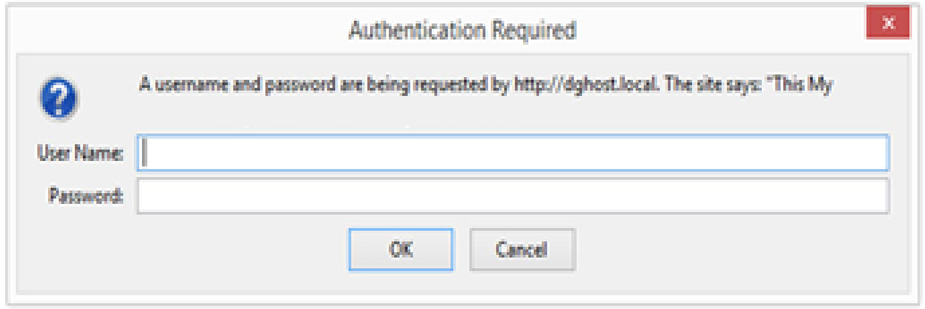
Results (1)

The FREQ Procedure

provider_name	Frequency	Percent	Cumulative Frequency	Cumulative Percent
ATHENS HEALTH AND REHABILITATION LLC	125	12.50	125	12.50
BURNS NURSING HOME, INC.	125	12.50	250	25.00
COOSA VALLEY HEALTHCARE CENTER	125	12.50	375	37.50
EASTVIEW REHABILITATION & HEALTHCARE CENTER	125	12.50	500	50.00
HATLEY HEALTH CARE INC	125	12.50	625	62.50
HIGHLANDS HEALTH AND REHAB	125	12.50	750	75.00
MERRY WOOD LODGE	125	12.50	875	87.50
PLANTATION MANOR NURSING HOME	125	12.50	1000	100.00

Use case: Download a file with user/password

Access a site with Basic authentication




```
proc http
   url="www.secured-site.com"
   out=resp
   method="GET"
   WEBUSERNAME="user"
   WEBPASSWORD="pass"
   AUTH_BASIC
   AUTH_NEGOTIATE;
run;
```

Example of AUTH_BASIC

```
filename resp temp;
proc http
 url="http://httpbin.org/basic-auth/chris/pass123"
 method="GET"
 AUTH_BASIC
 out=resp
 webusername="chris"
 webpassword="pass123"
 ;
run;

data _null_;
 rc = jsonpp('resp','log');
run;
```

```
{
   "authenticated": true,
   "user": "chris"
}
```

Test methods using http://httpbin.org/

Tip: Using httpbin.org to test logic

```
filename resp "%sysfunc(getoption(WORK))/stream.json";
proc http
 url="https://httpbin.org/stream/1"
 method="GET"
 out=resp;
run;

/* Supported with SAS 9.4 Maint 5 */
%put HTTP Status code = &SYS_PROCHTTP_STATUS_CODE. : &SYS_PROCHTTP_STATUS_PHRASE.;

data _null_;
 rc = jsonpp('resp','log');
run;
```

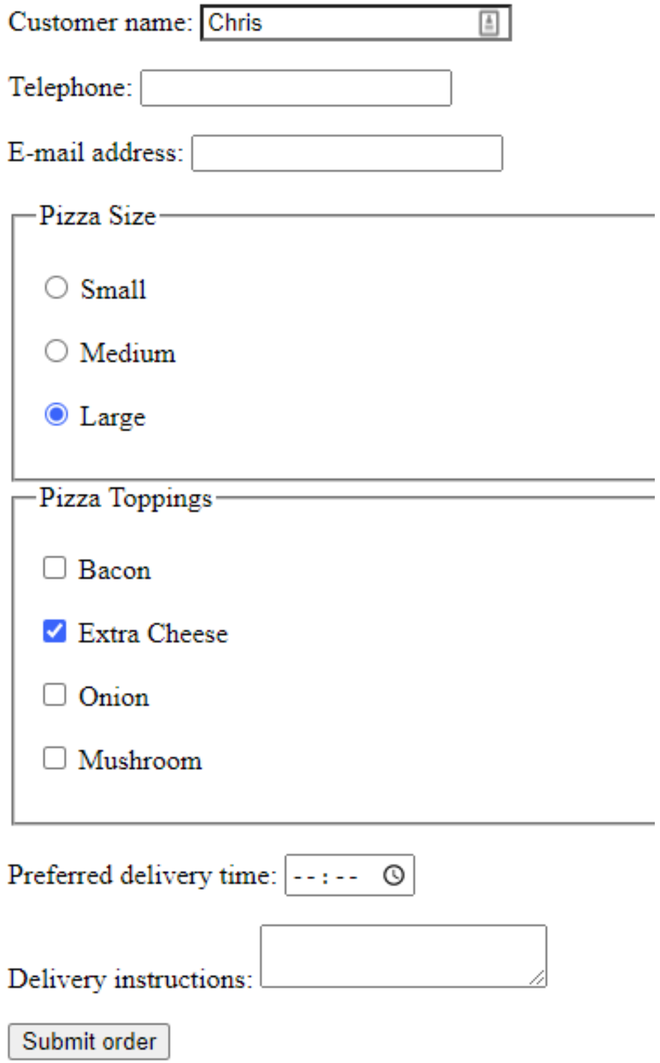
```
HTTP Status code = 200 : OK
{
  "url": "https://httpbin.org/stream/1",
  "args": {

  },
  "headers": {
    "Host": "httpbin.org",
    "X-Amzn-Trace-Id": "Root=1-5f3bdd40-542722e0066832f012490708",
    "User-Agent": "SAS/9",
    "Accept": "*/*"
  },
  "origin": "199.199.188.99",
  "id": 0
}
```

Use case: Download and read data in ZIP archive

Combine PROC HTTP and FILENAME ZIP

```
/* Download the ZIP file */
filename dl "%sysfunc(getoption(WORK))/ri130701_13dn01.zip";

proc http
  url="http://www.freddiemac.com/mbs/data/stacr/ri130701_13dn01.zip"
  method='GET'
  out=dl
  ct="application/zip";
run;

filename dl clear;

/* Crack it open and read with FILENAME ZIP */
filename inzip ZIP "%sysfunc(getoption(WORK))/ri130701_13dn01.zip";

data orig;
  infile inzip(ri130701_13dn01.txt);
…
```

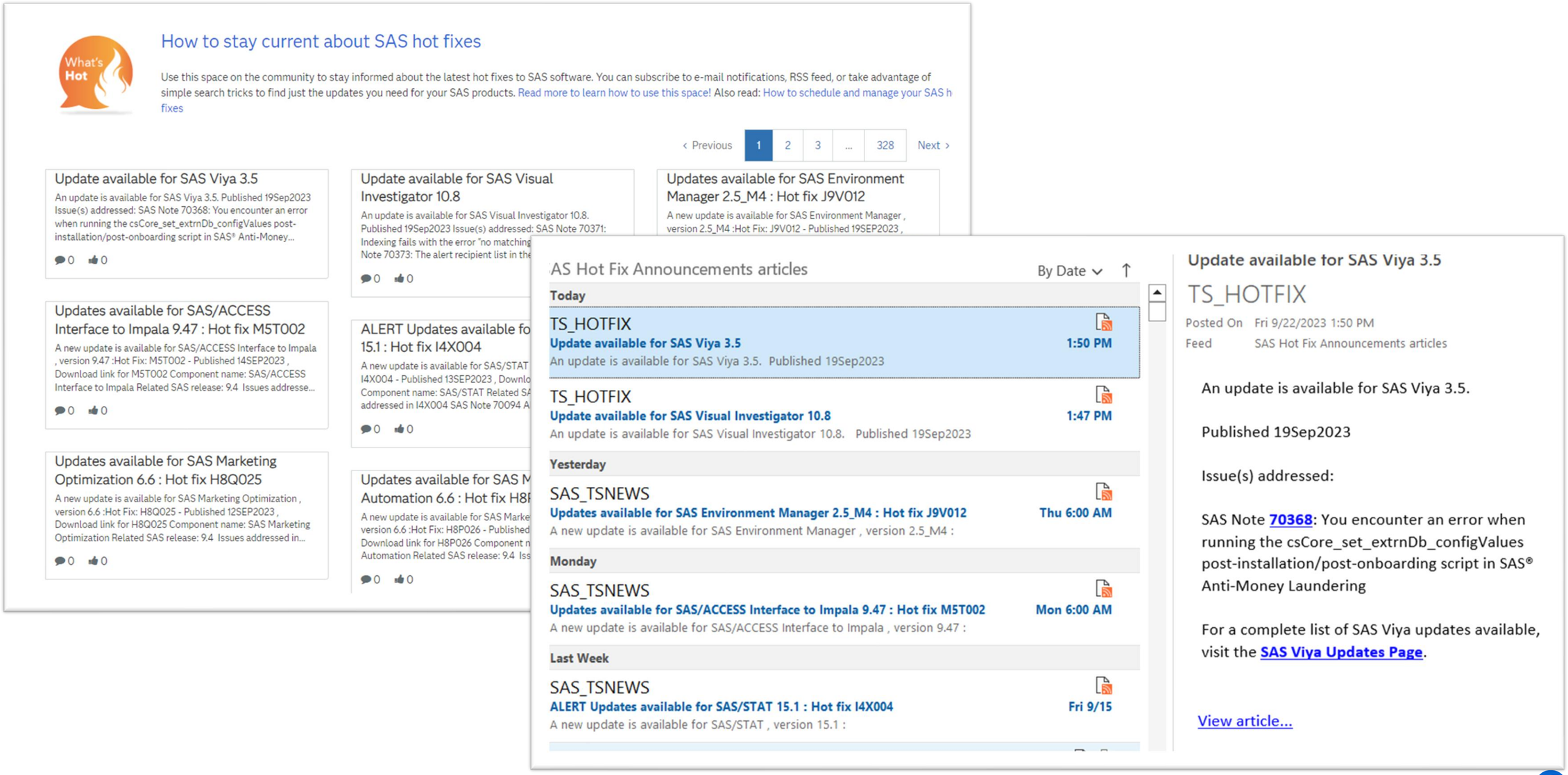
Use case: Post data to the web via form




```
filename resp temp;
proc http
 url="http://httpbin.org/post"
 method="POST"
 in="custname=Chris&size=large&topping=cheese"
 out=resp;
run;

data _null_;
 rc = jsonpp('resp','log');
run;
```

```
"form": {
   "custname": "Chris",
   "size": "large",
   "topping": "cheese"
 },
```

Example of POST in action: SAS Hot fix notices



Use case: Scrape data from the web

But first, an editorial

Web scraping is lossy, fragile process.

The information on the web page does not include data types, lengths, or constraints metadata.

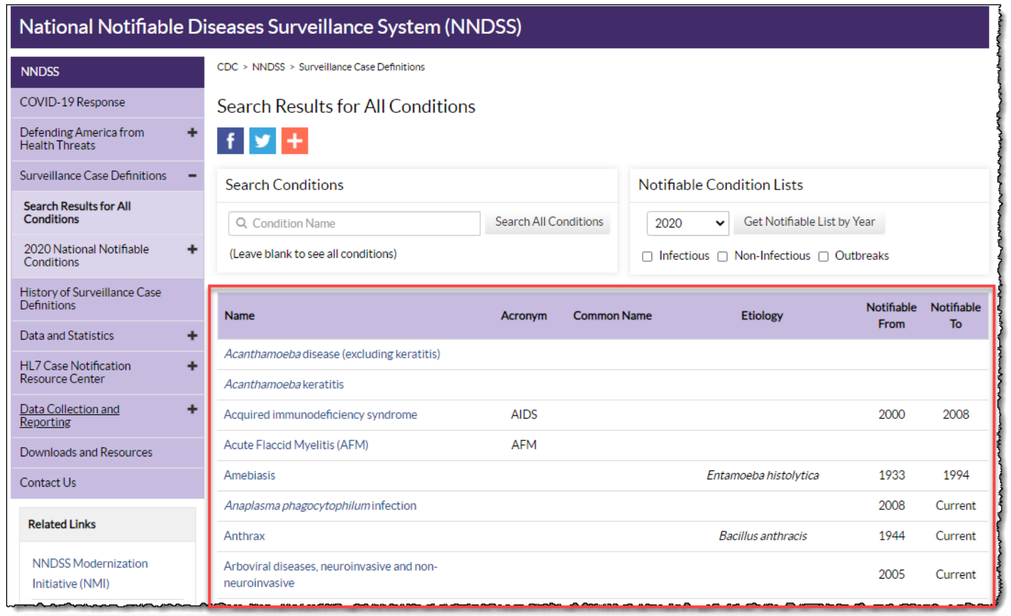
And one tweak to the presentation of the web page can break any automated scraping process.

Use case: Scrape data from the web

But if you must…

For this step	Use these features
Get the contents of the web page	PROC HTTP or FILENAME URL
Process/parse the web page contents	DATA step, with parsing functions such as FIND, SCAN, and regular expressions via PRXMATCH. Use SAS informats to convert text to native data types.
Repeat across subsequent pages	SAS macro language (%DO %UNTIL processing) or DATA step with CALL EXECUTE to generate multiple iterations of the fetch/parse steps.

Example of "web scraping"



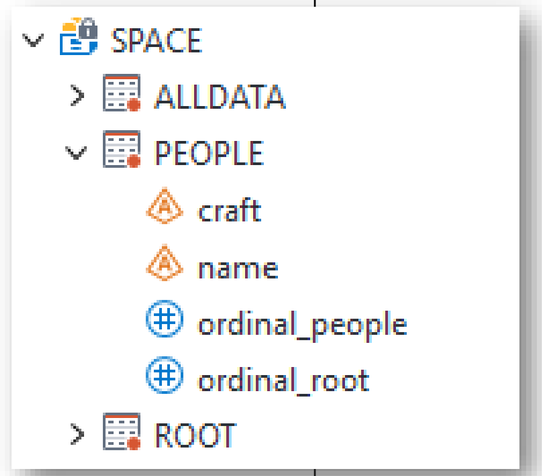

```
<table class="text-centered table" style="margin-bottom:0 !important; width:100%; ">
 <thead>…</thead>
 <tbody>
  <tr>
   <td style="text-align:left;vertical-align:middle;">
    <a href="/nndss/conditions/acanthamoeba-disease-excluding-keratitis/" class="tp-link-po
     <i>Acanthamoeba</i>
     " disease (excluding keratitis)
                                  "
     <span class="print-only">…</span>
     </a>
   </td>
   <td class="tablet-hidden"></td>
   <td class="tablet-hidden"></td>
   <td class="tablet-hidden"></td>
   <td> </td>
   <td> </td>
```

```
/* Get all of the nonblank lines */
filename CDC url "https://wwwn.cdc.gov/nndss/conditions/search/";
data rep;
infile CDC length=len lrecl=32767;
input line $varying32767. len;
 line = strip(line);
 if len>0;
run;
filename CDC clear;

/* Parse the lines and keep just condition names */
/* When a condition code is found, grab the line following (full name of
condition) */
/* and the 8th line following (Notification To date)
*/
/* Relies on this page's exact layout and line break scheme */
data parsed (keep=condition_code condition_full note_to);
 length condition_code $ 40 condition_full $ 60;
 set rep;
 if find(line,"/nndss/conditions/") then do;
   condition_code=scan(line,4,'/');
   pickup= _n_+1 ;
   pickup2 = _n_+8;
   set rep (rename=(line=condition_full)) point=pickup;
   set rep (rename=(line=note_to)) point=pickup2;
   output;
  end;
run;
```

Use case: REST API with JSON response

```
/* Neat service from Open Notify project */
filename resp temp;
proc http
 url="http://api.open-notify.org/astros.json"
 method= "GET"
 out=resp;
run;

data _null_;
 rc = jsonpp('resp','log');
run;

/* Assign a JSON library to the HTTP response */
libname space JSON fileref=resp;
```


```
{
  "people": [
    {
      "name": "Sergey Prokopyev",
      "craft": "ISS"
    },
    {
      "name": "Dmitry Petelin",
      "craft": "ISS"
    },
    {
      "name": "Frank Rubio",
      "craft": "ISS"
    },
    {
      "name": "Jing Haiping",
      "craft": "Tiangong"
    },
    {
      "name": "Gui Haichow",
      "craft": "Tiangong"
    },
    {
      "name": "Zhu Yangzhu",
      "craft": "Tiangong"
    },
    {
      "name": "Jasmin Moghbeli",
      "craft": "ISS"
    },
  . . .
```

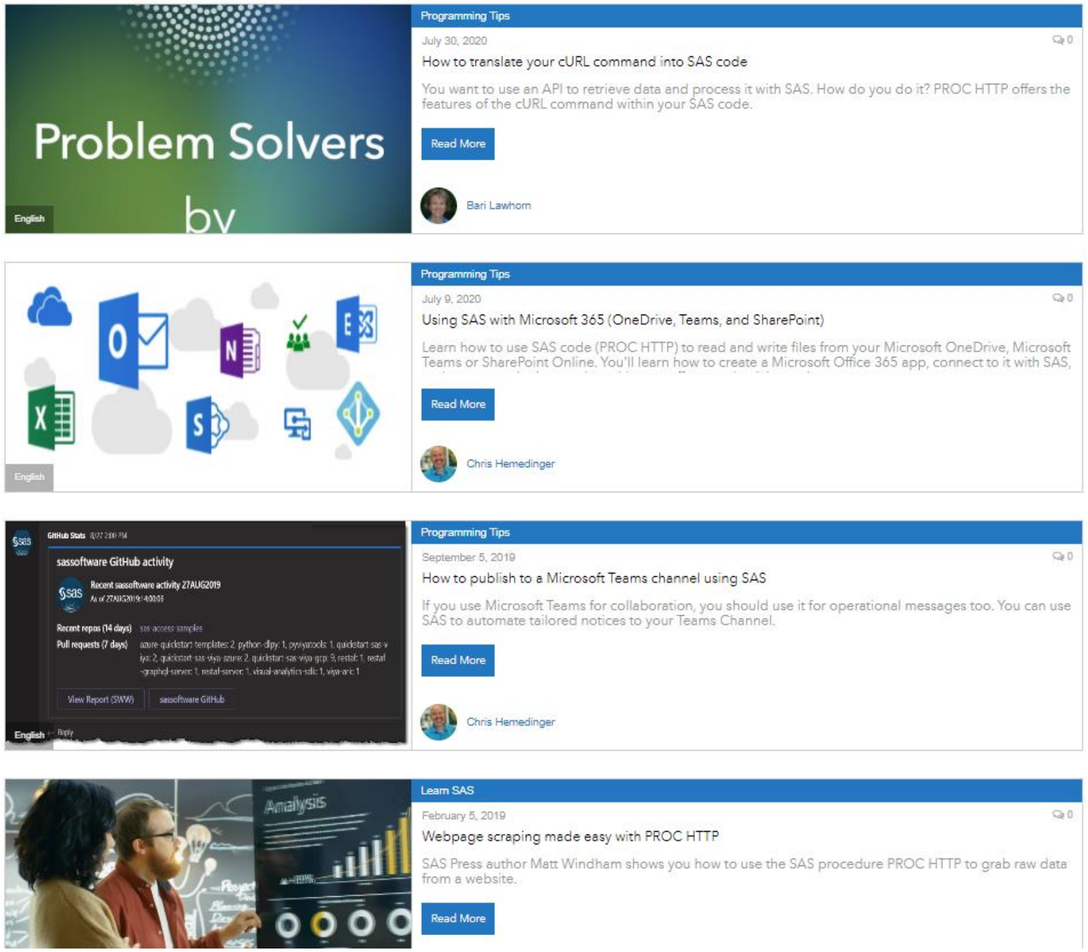
Other functions to help

HTMLDECODE Function	Decodes a string that contains HTML numeric character references or HTML character entity references and returns the decoded string.
HTMLENCODE Function	Encodes characters using HTML character entity references and returns the encoded string.
URLDECODE Function	Returns a string that was decoded using the URL escape syntax.
URLENCODE Function	Returns a string that was encoded using the URL escape syntax. From: `urlencode(%str(ga:sessions,ga:pageviews,ga:users));` To: `ga%3Asessions,ga%3Apageviews,ga%3Ausers`
%str(&)	Prevent SAS from interpreting “&” in a URL as a macro variable. `in="custname=Joe%str(&)size=large%str(&)topping=cheese"` Avoids "`WARNING: Apparent symbolic reference TOPPING not resolved.`"
JSONPP function	"Pretty print" a JSON response to a file or log

Learn more

https://blogs.sas.com/content/tag/proc-http/

Tag: PROC HTTP



SAS Conference papers

ABCs of PROC HTTP

REST Just Got Easy with SAS and PROC HTTP

Ask the Expert webinar
How Do You Use SAS® to Access Data and APIs From the Web?

BASUG proud moments

2023 User Group Leader award

2023 Distinguished Educator of the Year

Happy 40th Birthday BASUG!