

# Screen Scraping Using SAS® for Healthcare Fraud Prevention and Detection

Hitesh Kharbanda, Optum Global Solution

UnitedHealth Group

## ABSTRACT

Fraud in U.S. Healthcare System is rampant. It is estimated that 3-10% of U.S. Healthcare annual expenditure is lost to fraud and abuse. One of the most common types of fraud is phantom provider billing. Phantom provider billing occurs when providers either do not exist or do not hold a valid license and bill false claims for services that were not rendered. Phantom providers frequently change billing patterns on submitted claims to avoid detection from payer edits. One method used to avoid detection is to submit a new Tax Identification Number (TIN) into the payment system whenever their current TIN has been identified as fraudulent. In addition, phantom providers will often use a TIN belonging to an existing legitimate provider in order to disguise fraudulent claims, in a similar manner to identity theft with personal credit cards. Accurate detection of phantom providers is difficult due to this type of evolving claim submission behavior, resulting in payment leakage when claims from a phantom provider avoid detection by existing edits.

One method to detect evolving behavior of phantom providers is to scrape and update the TIN/address information from actual owners or authorized agents for the documented fraud providers. This can be done in a time consuming manual process. It can be automated by writing a program to scan the internet via screen/web scraping using SAS®.

## INTRODUCTION

The Internet is rich with data, and much of that data that is available on web pages can be scraped for facilitating analysis, and be further used in a desired manner for analytic development.

This paper describes how SAS can be used to scrape data from the web as an application for detecting phantom provider healthcare fraud.

## OVERVIEW OF WEB SCRAPING

Most of the data available over the web is not readily available in a database. It is present in an unstructured format (HTML) and is not downloadable. Therefore, it requires knowledge & expertise to use this data effectively.

Web scraping is a technique for converting the data present in unstructured format (HTML tags) over the web to the structured format which can easily be accessed and used for an automated process.

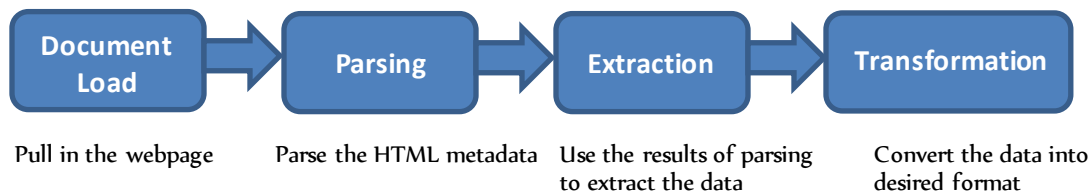
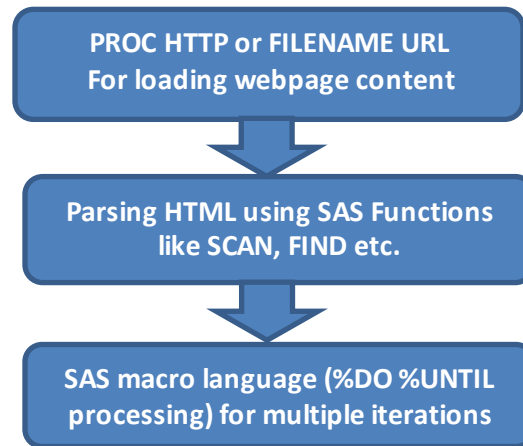


Figure 1. Anatomy of Web Scraping

## USING SAS FOR WEB SCRAPING

Base SAS software is a product that can be used for web scraping, and one does not require any special extension of SAS products. The following features of Base SAS software can be used for web scraping purpose.



**Figure 2. SAS Features for Web Scraping**

### PHANTOM PROVIDER ADDRESS “ANALYTIC”

Most of the healthcare payers in United States maintain a list of providers who have been identified as phantom, known for false billing of claims causing millions of dollars loss to payers. In order to bypass payment edits designed to detect false claims, these phantom providers frequently change TINs.

Optum’s Phantom Provider Address Analytic aims to detect new addresses, TINS, and associated providers for these known phantom providers using web scraping technology leveraging SAS. One may use various authenticated resources over the internet that stores details of registered businesses in any state. These details include name of the owner or authorized agent of the business, other associated business persons, TINS, associated addresses, etc. For purposes of this paper, we used sunbiz.org, an official State of Florida website; however, the results have been modified to be fictitious.

Authenticated web resources can be scraped using SAS in a dynamic way to frequently update new business details of phantom providers. Claims billed by the newly identified providers and addresses can be flagged to prevent any loss from false claims, thus making the analytic dynamic.

Below are the required steps with a sample program to learn how to use SAS for scraping business details from the web for known entity/business/person (Provider). :

#### **1. Import the provider/agent details from the input source.**

This file would typically include provider name for which we are seeking the latest business details/address/TIN. The suspect name in this paper is referred to as *Agent*.

```

/* Reading agents from the input source */

PROC IMPORT DATAfile="Agents.xlsx"
  OUT= work_agents
  DBMS= xlsx replace;
  GETNAMES= no;
  RUN;

/* Finding agents volume */

PROC SQL;
  SELECT count(distinct agent) into :agent_vol
  FROM work_agents;
  QUIT;

/* Holding agent names into macro variables */

PROC SQL noprint;
  SELECT agent
  into :agent1 - :agent%eval(&agent_vol)
  FROM work_agents
  QUIT;

%put &agent1;

```

## 2. Load the content of the webpage, and search agent details on URL (search.sunbiz.org) using SAS features – FILENAME URL and PROC HTTP

To search for the business details of any provider/agent on sunbiz.org, we can search for any agent/entity over sunbiz web address via various options available on their search home page. Select the option to search "by office/registered agent name".

Please check out URL:

<http://search.sunbiz.org/Inquiry/CorporationSearch/ByOfficerOrRegisteredAgent>

It will show something like below:


**Search Corporations, Limited Liability Companies, Limited Partnerships, and Trademarks by Officer or Registered Agent**

Name:

If an individual, enter as: Last Name, First Name, Middle Initial. Partial names are acceptable.

**Other Search Options**

Search by:

- [Entity Name](#)
- [Officer/Registered Agent](#) 
- [Registered Agent Name](#)
- [Trademark Name](#)
- [Trademark Owner Name](#)
- [FEI/EIN](#)
- [Detail By Document Number](#)
- [Zip Code](#)
- [Street Address](#)

Display 1. Screen capture of search page

If we will search with name of any agent/provider, it will take us to next page showing results of all the agents/provider with that name. Something like below:

### Officer/Registered Agent Name List

Officer/RA Name	Entity Name	Entity Number
Agent_name1	Business Name 1	123455678
Agent_name2	Business Name 2	838383838
Agent_name3	Business Name 3	293930030
Agent_name4	Business Name 4	920202020
Agent_name5	Business Name 5	202020020

Details of different providers/agents having same name as the searched provider/agent

### Display 2. Search results

For the above page, we get the following URL on our web browser:

<http://search.sunbiz.org/Inquiry/CorporationSearch/SearchResults/OfficerRegisteredAgentName/AGENTNAME/Page1>

You may see in the above web address, at the end of the URL there is a unique tag for agent we searched for. This is our target URL, and by referencing each agent name in URL we can run a SAS code in loop for each agent/provider name to get HTML structure behind each agent's search page result similar to one we have shown above (Display 3).

Below the screen shot is an example of HTML content that we are interested in from the search result page of "agent name". In this HTML content, we get each search result specific URL that further when clicked upon takes us into business details of each search result/Agent, i.e. for all agents/providers listed in search output we will have unique URL associated with them for their respective business detail.

```
<td class="medium-width"><a href="/Inquiry/CorporationSearch,
inquirytype=OfficerRegisteredAgentName&directionType=Initial&searchNameOrder:
&aggregateId=flal-119000025225-99a21f30-d684-4147-aa7a-a847baf8a94f&searchTer

<td class="medium-width"><a href="/Inquiry/CorporationSearch,
inquirytype=OfficerRegisteredAgentName&directionType=Initial&searchNameOrder:
&aggregateId=flal-116000183712-8ac17a67-8a9c-4783-9b08-420457b5e066&searchTer
```

These are Agent specific URL we are interested in, and these URL will give us all business details of the Agent/Provider

### Display 3. HTML content for search results

```

/* Searching agent details on search.sunbiz.org */

options DLCREATEDIR;
%let htmldir = %sysfunc(getoption(WORK))/html;
Libname html "&htmldir.";
options symbolgen mprint mlogic;

%macro search (num=);

    %do i = 1 %to &num.;
        %let url =
http://search.sunbiz.org/Inquiry/CorporationSearch/SearchResults/OfficerRe
gisteredAgentName/&&agent&i./Page1;

        FILENAME src "&htmldir./&i..html";
        PROC HTTP
            method="GET"
            url= "&url."
            out=src;
            RUN;
        %end;

    %mend search;
%search (num=&agent_vol.);

/* Reading the search results */

DATA results;
    infile "&htmldir./*.html" Length=len lrecl=32767;
    input line $varying32767. len;
    line = strip(line);
    if len>0;
    RUN;

    Libname html clear;

```

Below is the screen shot to show how HTML content can be loaded into a SAS file. When we run the above code we get the unique URL associated with each provider in SAS output which can be loaded in a SAS file.

39120	<link href="/Content/cms?v=773TS7nsPXeuYYpOn6ZYEIuFnheZzIxfMLXgzNPFVNg1" rel="stylesheet"/>
39121	<link href="/Content/css?v=YjTsaROk3HQKOSU6Q5ygJPBgdI0Vr6Iz4ZXs9m6tLaY1" rel="stylesheet"/>
39122	<script src="/bundles/jquery?v=KVwTEjS-NVeUd4hOvP7INIsGD7rWshJQoQzaFplgLQ1"></script>
39123	<script src="/bundles/modernizr?v=rGcoDow97GYrNMSwHq7xCCjlcB3UIY4_0hPRc6BBSQA1"></script>
39124	<script type="text/javascript">
39125	var _gaq = _gaq    [];
39126	_gaq.push(['_setAccount', 'UA-3263504-14']);
39127	_gaq.push(['_setDomainName', 'sunbiz.org']);

Loaded the HTML content in SAS using PROC HTTP

#### Display 4. HTML data loaded in SAS file

In the next section, we are using the unique URL for each agent to go their specific business detail page, and repeating the use of PROC HTTP to get HTML content of their business detail page. This HTML content will have all the information we require such as address/TIN associated with provider/agent.

When we click on our first result for search "agent name", we are routed to following window with all business details for that particular search result. Below screenshot shows how business details for any agent look like when we click on each result in search output.

### Detail by Officer/Registered Agent Name

<b>Agent Name</b>	
<b>Business Name</b>	
<b><u>Filing Information</u></b>	
<b>Document Number</b>	xxxxxxx
<b>FEI/EIN Number</b>	xxxxxxx
<b>Date Filed</b>	xxxxxxx
<b>State</b>	xxxxxxx
<b>Status</b>	xxxxxxx
<b>Last Event</b>	xxxxxxx
<b>Event Date Filed</b>	xxxxxxx
<b>Event Effective Date</b>	xxxxxxx
<b><u>Principal Address</u></b>	
xxxxxxxxxxxxxxxxxxxxxxxx	

### Display 5. Agent business details

### 3. Parse the webpage/HTML content of unique URL associated with each Agent

In the code below, we have mapped the agent name in URL from previous output with first 11 characters of our Agent/provider name that we had in our input file. We have also utilized SPEDIS function, which returns a measure of how a word is close to another word in spelling.

```

/* Getting the URL and agentname details */

DATA parsed0 (keep=URL line agentname);
  Length URL $3000;
  SET results;
  if find(line,"Go to Detail Screen") then
    do
      URL= tranwrd(substr(LINE,35,index(LINE, 'title') -
37), '&', ', '&');
      agentname= compress(UPCASE(compress(substr(line,index(LINE, 'Go
to Detail Screen')+21,50), '</a></td' '')), "~!@#$$%^&* () -
_ =+ \ | [ ] { } ; : ' , . < > ? / %", "ps");
      output;
    end;
  RUN;

PROC SQL;
  CREATE table parsed as
    SELECT distinct *, substr(agentname,1,11) as Agentname_n
  FROM parsed0;

```

```

        QUIT;

        DATA work_AGENTS1;
        SET work_AGENTS;
        Agent=UPCASE(compress(agent,"`~!@#%&*()_-+=\|[]{};:'.<>?/%", "ps"));
        RUN;

        PROC SQL;
        CREATE table agent as
        SELECT distinct agent,substr(agent,1,11) as agent_n
        FROM work_AGENTS1;
        QUIT;

/* Mapping the results DATA with the agent input on first 11 chars of
agent name */

        PROC SQL;
        CREATE table url_final_n as
        SELECT distinct a.URL, a.agentname_N, b.agent_N,
        a.agentname, b.agent
        FROM work.parsed a LEFT join AGENT b
        on a.Agentname_n = b.agent_n
        where agent_n <> "";
        QUIT;

/* Cartesian Mapping one results DATA with the agent */

        PROC SQL;
        CREATE table url_final as
        SELECT distinct a.URL, a.agentname, b.agent
        FROM work.parsed a cross join AGENT b;
        QUIT;

/* Applying the SPEDIS function on result DATA */

        DATA url_final1;
        SET url_final;
        if spedis(agent,agentname)<=18;
        RUN;

/* Appending both results -> 11chars mapping and result DATA on SPEDIS
function */

        DATA url_final_DATA;
        SET url_final_n(keep=url agentname agent) url_final1;
        RUN;

        PROC SQL;
        CREATE table url_final_DATA1
        as SELECT distinct * from url_final_DATA;
        QUIT;

```

#### 4. Extract web content of each agent, and further parse content to get business details

Here, we are running each specific URL in a loop to get the HTML content for all of them in single file, and then we use SAS functions like Scan, Find etc. to get the business details by each agent.

```
/* Creating the URL search for each agent to get the complete agent details
*/
```

```
    %LET S = %NRSTR(%nrstr(&));
    %LET ABC = %SUPERQ(S);

    DATA parsed (keep=url1 agentname);
    Length url1 $ 3500 new $ 100;
    SET url_final DATA1;
    new = "http://search.sunbiz.org";
    url1=tranwrd(cats(new,url),"&","%SUPERQ(S)");
    RUN;
```

```
/* Getting the URL volume */
```

```
    PROC SQL;
    SELECT count(distinct url1) into :urlvol
    FROM parsed;
    QUIT;
```

```
/* Holding each URL into macro variable */
```

```
    PROC SQL noprint;
    SELECT url1
    into :url1 - :url%left(&urlvol.)
    FROM work.parsed;
    QUIT;
```

```
/* Passing each URL details to the below search loop to get details of
agents */
```

```
    Options DLCREATEDIR;
    %let htmdir1 = %sysfunc(getoption(WORK))/html;
    libname html1 "&htmdir1.";

    %macro search2 (num=);
    %do i = 1 %to &num.;
    %let urlA = &&url&i.;
    filename src1 "&htmdir1./&i..html1";

    PROC HTTP
        method="GET"
        url= "&urlA."
        out=src1;
    RUN;

    %end;
    %mend search2;
```



```

                %search2 (num=&urlvol.);

/* Reading the results */

DATA results1;
  infile "&htmldir1./*.html1" Length=len lrecl=32767;
  input line $varying32767. len;
  line = strip(line);
        if len>0;
          RUN;

          Libname html1 clear;

DATA test;
  SET results1;
  Lag_line = lag(line);
  RUN;

/* Reading agent name */

DATA Agent (keep = agent );
  Length agent $40;
  SET test;
        if find(lag_line,"Registered Agent Name & Address") then
          Agent = tranwrd(line,"<span>","");
          RUN;

/* Reading agent Status */

DATA status (keep=status);
  Length status $40;
  SET test;
        IF FIND (line,"Detail_Status") THEN
          status = tranwrd(substr(line,48,8),"</","");
  RUN;

/* Reading agent EIN */

DATA EIN (keep=EIN);
  Length EIN $40;
  SET test;
        IF FIND (line,"EIN Number") then
          EIN=tranwrd(substr(line,62,10),"</span","");
          RUN;

/* Reading agent Addresses */

DATA Address (keep=Add1 Add2 Add3);
  Length Add1 $ 60 Add2 $ 60 Add3 $ 60;
  SET test;

  if find (line,"Principal Address") then
    do;
      pickup = _n_+3;
      pickup2 = _n_+4;
      pickup3 = _n_+5;
    end;

```

```

        SET test point= pickup;
        Add1= substr(line,1,index(line, '<')-1);
        SET test point= pickup2;
        Add2= substr(line,1,index(line, '<')-1);
        SET test point= pickup3;
        Add3= substr(line,1,index(line, '<')-1);
    end;
RUN;

DATA address;
SET address;
where add1<> "" and add2<> "" and add3<> "";
RUN;

DATA address;
SET address;
    if add3 ne "</div>" then
add1 = catx("",add1,add2);
    RUN;

        DATA address(keep=add1 add2);
SET address;
    if add3 ne "</div>" then
add2= add3;
    RUN;

/* Removing blank cases */

DATA EIN;
SET EIN;
where EIN <> "";
    RUN;

DATA Agent;
SET Agent;
where agent <> "";
    RUN;

DATA status;
SET status;
where status <> "";
    RUN;

/* mapping all the DATA */

DATA final;
    MERGE Agent ein Address status;
    RUN;

```

## Sample output of program

	Agent	EIN	Add1	Add2	Status
1	Geoffrey Cohen	NONE	55666555 NE T...	MIAMI, FL 33172	INACTIVE
2	John Doe	12-345789	9999999 NW 73...	MIAMI, FL 33178	INACTIVE
3	Monty Python	NONE	1100001 STREE...	MIAMI, FL 33186	INACTIVE
4	Luther Blissett	NONE	900000 SAN LO...	CORAL GABLE...	INACTIVE
5	Monty Cantsin	98-765432	12191919-Lakesi...	Davie, FL 33328	ACTIVE
6	Poor Konrad	NONE	848484 SW 150...	MIAMI, FL 33196	INACTIVE
7	Dale Nixon	NONE	848484 SW 150...	MIAMI, FL 33196	INACTIVE
8	Netochka Nezva...	00-000000	54646 PORTOFI...	WESTON, FL 33...	ACTIVE

## Display 6. Final output

### CONCLUSION

This paper describes how to leverage features of SAS like PROC HTTP and FILENAME to load the entire webpage into one single document which can be further parsed to scrape the required information. This paper discussed the potential application of screen scraping using SAS for prevention of fraud in health insurance business.

### REFERENCES

Hemedinger, Chris. "How to scrape data from a web page using SAS" The SAS Dummy. December 4, 2017. Available <https://blogs.sas.com/content/sasdummy/2017/12/04/scrape-web-page-data/>

Duggins, Jonathan. SESUG Paper 236-2018, "Web Scraping in SAS: A Macro-Based Approach": Available at <https://www4.stat.ncsu.edu/~duggins/SESUG2018/SESUG%20236-2018.pdf>

### ACKNOWLEDGMENTS

I would like to thank Gina Hedstrom, Amy Neftzger, Naveen Pruthi, Neelabh Mishra, and Ashish Sharma for their constant support and encouragement. I will also like to thank my team members for their valuable input and support.

### CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Hitesh Kharbanda  
Optum Global Solutions  
Oxygen SEZ, Sec 144, Noida  
UP, India - 201306  
Hitesh\_kharbanda@optum.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.