SPIRION

Spirion Extensions

Algorithmic Validation of NPI Numbers with Spirion's Search API

Table of Contents

Algorithmic Validation of NPI Numbers with Spirion's Search API

Introduction

This cookbook shows how Spirion's Search API can be used to incorporate Luhn validation when searching for National Provider Identifier (NPI) numbers that uniquely identify healthcare service providers as part of HIPAA. Using this feature, sample computational logic in a C++ DLL validates potential matches initially captured by pattern-based definitions.

Requirements

Before working on the steps outlined in this document, please confirm the following:

- The latest Sensitive Data Manager (SDM) console is accessible.
- An integrated development environment (IDE) capable of editing and compiling Microsoft Visual C++ files is available.
    - Microsoft Visual Studio Enterprise 2019 is used in the procedure explained below.
- The Search API sample project has been downloaded from the Spirion Knowledge Base.
- NPI sample data is obtainable and is included in file npi_data.txt.

Important Notes

Spirion Software Versions

SDM Console version 11.8.2 and Spirion Agent version 11.8.7 were used to perform the steps described in this document.

National Provider Identifier

According to the Centers for Medicare & Medicaid Services (CMS), "*when an NPI is used as a card issuer identifier on a standard health identification card, it is preceded by the prefix 80840, in which 80 indicates health applications and 840 indicates the United States. The prefix is required only when the NPI is used as a*

card issuer identifier. However, in order that any NPI could be used as a card issuer identifier on a standard health identification card, the check digit will always be calculated as if the prefix is present."

To accommodate the logic described above, the logic contained in this Search API data type is designed to optionally match values with or without the prefix, however when the prefix is absent the U.S. region is assumed.

Luhn Algorithm

Also known as the "modulus 10" or "mod 10" algorithm, the Luhn algorithm establishes a check digit through the following logic:

1. Starting with the rightmost digit, double the value of every second digit.
2. A new value for the doubled digits is determined conditionally:
    a. If the value of the doubled digit is greater than 9 (i.e. two digits), add them together.
        i. For example, `6x2=12` which would lead to a ***new value of 3*** since `1+2=3`.
    b. Otherwise, the doubled value is the new value.
        i. For example, `4x2=8` which would lead to a ***new value of 8***.
3. Generate a `sum` from the new values (calculated from the doubled digits) as well as the non-doubled digits.
4. Determine the check digit by calculating `(10-(sum%10))%10`.
    a. **NOTE:** The "`%`" symbol above is used to indicate the `mod10` operation.

Example

The number "`8675309`" would calculate its check digit using the logic per the table below.

Original	8	6	7	5	3	0	9
Multipliers	2	1	2	1	2	1	2
	=	=	=	=	=	=	=
	16	6	14	5	6	0	18
Luhn Digits	7	6	5	5	6	0	9

The sum of the transformed digits is `38`.

The check digit is equal to `(10-(38%10))%10=2`.

The full value for "`8675309`" would be "`86753092`" when including a Luhn-validated check digit.

NOTE: *If working with a value that already includes a check digit, it should not be included in the procedure above – omit it from the computation (but use it to validate the result).*

Process

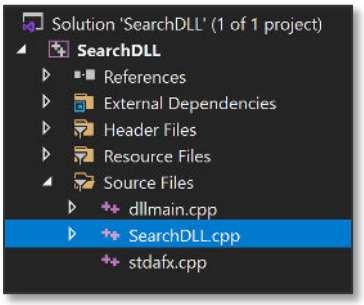
Custom data types created using Spirion's Search API require the creation of a DLL file that contains the user-defined logic articulated in C++ code. Once compiled, the DLL file is added to the SDM Console and hosted locally on any Spirion Agent engaged in a search for the custom data type.

Creating a DLL

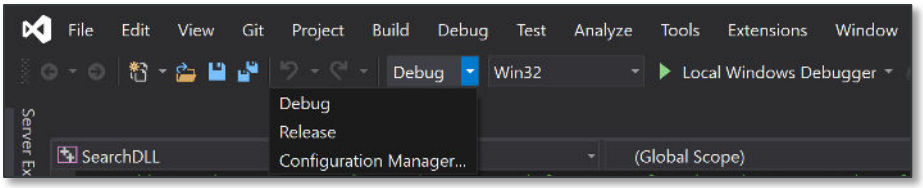
Spirion has created a sample project that should be used as the template for any custom data types created using the Search API. It is available for download within the Search API Knowledge Base article. Unpack it and open ***SearchDLL.vcxproj*** in an IDE, such as Visual Studio 2019 documented in this walkthrough.

NOTE: *The line(s) specified throughout this section references the SearchDLL file in its original state (i.e. prior to adding any new code specified below).*

1. In the project tree, navigate to ***SearchDLL.cpp***.



2. From the ***Solution Configurations*** pulldown menu, toggle from "Debug" to "Release."



3. On ***line 19***, specify a ***CUSTOM_SEARCH_NAME***.
4. On ***line 20***, specify a ***RESULT_TYPE***.
    a. The default value of "12001" may be used once, but all subsequent DLLs must feature a unique value.
5. On ***line 63***, specify a regular expression per the guidance in the comments.
    a. **NOTE:** *As emphasized in the project's comments, backslashes must be escaped.*
    b. For NPIs, enter the following baseline regex:

```
\\b(80840|80\\d{3})?\\d{10}\\b
```

6. On ***line 92***, specify any keywords that should be used to validate a match.
    a. If keyword validation is not necessary, comment out ***line 104***.
7. Replace ***lines 160 through 166*** with the following code:

```
//NPI LOGIC

int answerSize = answer.size();

int matchLength = answerSize - 1;

int sumScope = answerSize - 2;


int chksum = answer[matchLength] - _T('0');

int sumTotal = 0;


for (int i = sumScope; i >= 0; i--) {

        int val = answer[i] - _T('0');

        if ((i % 2 != 0 && answerSize % 2 != 0) || (i % 2 == 0 && answerSize % 2 == 0)) {

                int dubs = val * 2;

                if (dubs > 9) {

                        sumTotal += dubs - 9;

                }

                else {
```

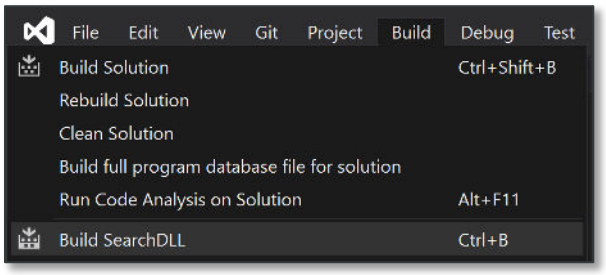
```
                        sumTotal += dubs;
                }
        }
        else {
                sumTotal += val;
        }
}

if (answerSize != 15) {
        sumTotal += 24;
}

int luhnVal = (10 - (sumTotal % 10)) % 10;

//Validating the check digit
if (luhnVal != chksum) {
        return(false);
}
```

8. Repeat the previous step on ***lines 212 through 218***.
9. ***Save*** changes to "SearchDLL.cpp."
10. Navigate to ***Build*** in the main menu and select ***Build SearchDLL***.



11. Note the export path indicated in the IDE's compilation logs to find where the newly built DLL file resides.
12. Rename this file to "SearchNPI.dll".

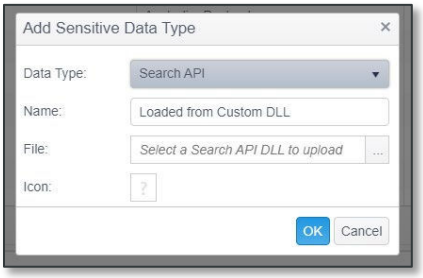
a. For ongoing maintenance of this custom data type, consider renaming the "SearchDLL" project to "SearchNPI" for clarity.

Configuring the Spirion Console

Adding the DLL

Once built, the DLL now needs to be added to the SDM Console..

1. Log into SDM.
2. Navigate to the ***Admin*** tab and select ***Sensitive Data Types*** from the menu on the left.
3. Click ***Add*** from the ribbon menu.
4. From the ***Data Type*** pulldown, select "Search API."
5. Click the **...** button next to ***File***.



6. Browse to the DLL(s) generated for this exercise in the previous section and click ***Open***.
7. Select ***Ok*** to finalize the entry.
8. The custom data type associated with the DLL file will now appear as a "Search API" entry with its associated "Type Number" and "Name" as specified in the previous section.

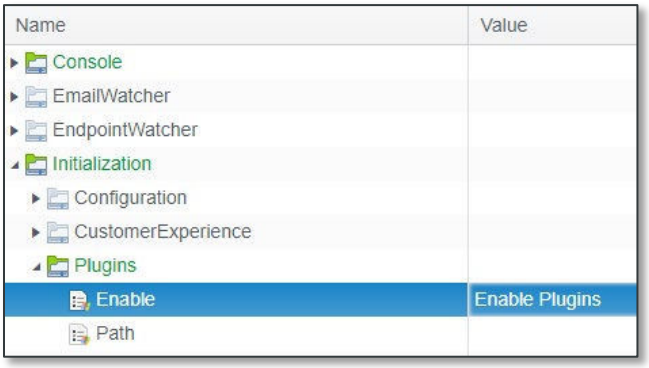
Sensitive Data Type	Type Number	Name	Value	Icon	ID
Search API	12861	NPI	*Click View/Edit to view or ...*		NPI

Setting the SDM Scan Policy

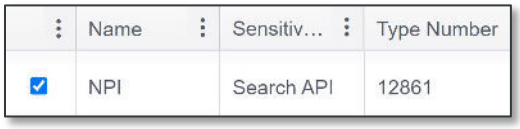
The Search API should be enabled via "Scheduled Task" policy (as opposed to a "System" policy).

1. Navigate to the ***Policies*** tab.

2. Modify or create a ***Scheduled Task*** policy that will be sent to the Spirion Agent(s) for discovery scans using the Search API custom data type(s).
3. Under ***Settings*** expand ***Initialization > Plugins*** and set ***Enable*** to "Enable Plugins."
4. ***Initialization > Plugins > Path*** can be left blank (its default setting) unless the Spirion Agent is installed to a nonstandard location.



5. Under ***Sensitive Data Types*** within the same policy, select NPI.



6. The Agent(s) assigned to this policy ***must*** be configured per the guidance in the following section before being used in a search.

Staging the Spirion Agent

The DLL file needs to be hosted locally within the Spirion Agent's install directory for Search API data types to be returned in scans initiated by either the Console or Agent UI.

1. Using Windows File Explorer, navigate to the installation path of the Spirion Agent.
    a. This is ***C:\Program Files (x86)\Spirion*** by default.
2. Create a folder called "Plugins" and place the DLL file(s) in that new directory.

Outcomes

With the custom DLL added to both the SDM Console and the Spirion Agent, scans configured to use the Search API will include matches for NPI numbers that are validated programmatically.

SDM Console Searches

Search API entries in the SDM Console are included in scan results so long as "Initialization > Plugins > Enable" is set to "Enable Plugins" in an actively engaged policy – selection via the policy's "Sensitive Data Types" menu is used for tracking purposes only.

Restricting Results

Individual Search API data types can be selectively included in a discovery scan by editing the Scheduled Task policy to include the following adjustments:

1. Navigate to the ***Settings*** of a Scheduled Task configured to enable the Search API.
2. Under ***Console***, edit ***matchTypesCustom*** to include the "Type Number" of the custom data type that should be included.
3. All other Search API entries will be omitted from scan results.