SPIRION

Spirion Extensions

Algorithmic Validation of Mobile Device Identifiers with Spirion's SearchAPI

Algorithmic Validation of Mobile Device Identifiers with Spirion's SearchAPI

Introduction

Mobile devices connect to their intended network (LTE, 5G, etc.) using a cellular modem/radio, and they are authorized to send or receive mobile data via an associated Subscriber Identity Module (SIM). These two components – cellular radio and SIM – have unique identifiers: IMEI and ICCID numbers, respectively. Much like how MAC addresses are used in traditional IT infrastructure, these values are utilized by network operators and enterprise organizations to manage security and services for customers, employees, and unattended equipment communicating over wireless WAN connections.

Both IMEIs and ICCIDs are linked to individual identities, meaning they are considered to be personal data as specified by regulations such as the California Consumer Privacy Act (CCPA) and the General Data Protection Regulation (GDPR). They also both conform to a numeric syntax that includes a checksum formula, the Luhn algorithm. Values constructed with procedural logic are best searched for by incorporating their underlying rules in the discovery process – this enables accuracy beyond the capabilities of pattern matching as articulated by keywords, dictionary lists, or regular expressions alone.

Spirion's Search API creates customized algorithmic search parameters for user-defined data types. Using this feature, custom SearchAPI DLL validates potential matches initially captured by pattern-based definitions. This document covers the steps necessary to incorporate Luhn validation when configuring IMEI and ICCID searches with Spirion.

Requirements

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

- The latest 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.
- IMEI and/or ICCID sample data is obtainable.

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.

Test Data

Sample IMEIs and ICCIDs can be obtained from a cellular phone. The steps below were performed on devices running Android v12 and iOS v16.

Android:

1. Navigate to ***Settings > About Phone > Status information***.
2. Click on ***SIM card status*** to view the ICCID number.
3. Navigate back to ***Status information***.
4. Select ***IMEI information*** to view the IMEI number.

iOS:

1. Navigate to ***Settings > General > About***.
2. Scroll down to ***Physical SIM***.
3. Reference the values in the "IMEI" and "ICCID" fields.

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
Sum 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

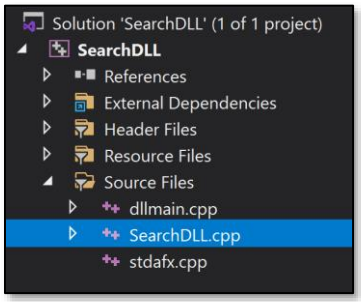
Custom data types created using Spirion's Search API require the creation of a DLL file that contains the user-defined logic articulated in C++ code. Once compiled, the DLL file is added to the SDM Console and hosted locally on any Spirion Agent engaged in a search for the custom data type.

Creating a DLL

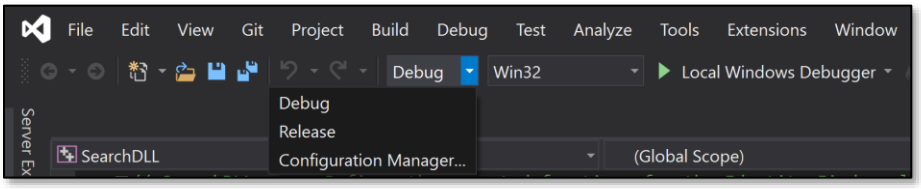
Spirion has created a sample project that should be used as the template for any custom data types created using the Search API. It is available for download within the Search API Knowledge Base article. Unpack it and open ***SearchDLL.vcxproj*** in an IDE, such as Visual Studio 2019 documented in this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

NOTE: *Two versions, one for IMEI and one for ICCID, are attached to this document's Spirion Marketplace article for additional reference.*

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



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

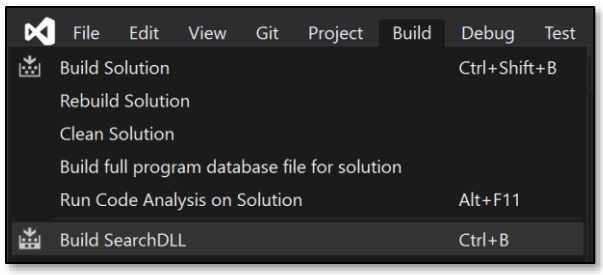


3. On ***line 19***, specify a ***CUSTOM_SEARCH_NAME***.
4. On ***line 20***, specify a ***RESULT_TYPE***.
    a. The default value of "12001" may be used once, but all subsequent DLLs must feature a unique value.
5. On ***line 63***, specify a regular expression per the guidance in the comments.
    a. **NOTE:** *In this document, the regex has been simplified to emphasize the algorithmic validation outlined in subsequent steps. Additional consideration for refining the pattern-matching portion of this example may be warranted if looking to isolate specific IMEI Type Allocation Codes (TACs) or ICCID subparts.*
    b. For IMEIs, enter the following baseline regex: `(^|\\s)\\d{15}($|\\s)`
    c. For ICCIDs, use: `(^|\\s)89(\\d{17}|\\d{18})($|\\s)`
6. Comment out ***line 104*** – this function offers additional keyword validation that is not relevant to the process covered in this document.

7. Replace ***lines 158 through 166*** with the following code:

```
int answerSize = answer.size();
int lastDigit = answerSize - 1;
int payloadLength = answerSize - 2;

int checkDigit = answer[lastDigit] - _T('0');
int sumTotal = 0;

for (int i = payloadLength; i >= 0; i--) {
        int currentDigit = answer[i] - _T('0');
        if (i % 2 != 0) {
                int doubleDigit = currentDigit * 2;
                if (doubleDigit > 9) {
                        sumTotal += doubleDigit - 9;
                }
                else {
                        sumTotal += doubleDigit;
                }
        }
        else {
                sumTotal += currentDigit;
        }
}

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

//Test if the Luhn value equals check digit
if (luhnVal != checkDigit) { return(false); }
```

8. Repeat the previous step on ***lines 227 through 218***.
    a. **NOTE:** *The line range specified above references the file in its original state (i.e. prior to adding the new code specified in the previous step).*
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 "SearchIMEI.dll".
    a. For ongoing maintenance of this custom data type, consider renaming the "SearchDLL" project to "SearchIMEI" for clarity.

13. Repeat this process using a separate project to build a DLL for the ICCID custom data type, replacing “IMEI” with “ICCID” for any naming conventions and referencing the following code instead of what’s listed previously for step #7:

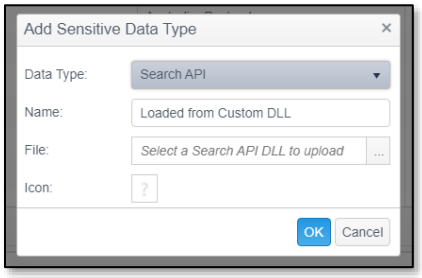
```
int answerSize = answer.size();
int lastDigit = answerSize - 1;
int payloadLength = answerSize - 2;

int checkDigit = answer[lastDigit] - _T('0');
int sumTotal = 0;

for (int i = payloadLength; i >= 0; i--) {
        int currentDigit = answer[i] - _T('0');
        if ((i % 2 != 0 && answerSize % 2 != 0) || (i % 2 == 0 && answerSize % 2 == 0)) {
                int doubleDigit = currentDigit * 2;
                if (doubleDigit > 9) {
                        sumTotal += doubleDigit - 9;
                }
                else {
                        sumTotal += doubleDigit;
                }
        }
        else {
                sumTotal += currentDigit;
        }
}

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

//Test if the Luhn value equals check digit
if (luhnVal != checkDigit) { return(false); }
```

Configuring the Spirion Console

Adding the DLL(s)

Once built, the DLL now needs to be added to the SDM Console. This process must be repeated for each DLL generated – presumably once for IMEI and once for ICCID.

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***.



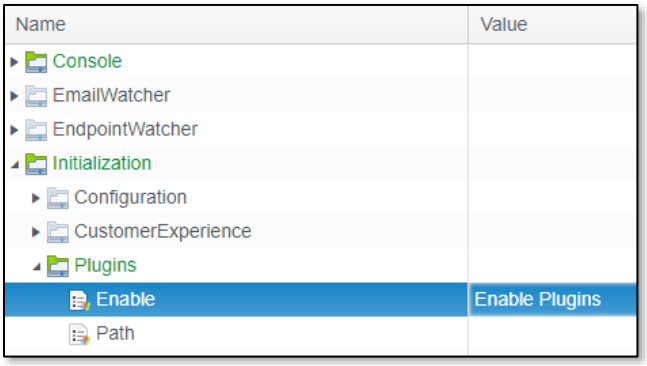
6. Browse to the DLL(s) generated for this exercise in the previous section and click ***Open***.
7. Select ***Ok*** to finalize the entry.
8. The custom data type associated with the DLL file will now appear as a "Search API" entry with its associated "Type Number" and "Name" as specified in the previous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

Sensitive Data Type	Type Number ↑	Name
Search API	12001	IMEI
Search API	12002	ICCID

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 IMEI and ICCID.

	Name	Sensitive Data Type	Type Number
☑	ICCID	Search API	12002
☑	IMEI	Search API	12001

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(s) need 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 DLLs added to both the SDM Console and the Spirion Agent, searches will include matches for both IMEIs and ICCIDs that are validated programmatically with a checksum.

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.
    a. For example, assuming the SearchDLL for IMEIs was created using the instructions in this document, entering "12001" under ***Console > matchTypesCustom*** will configure a scan to only include IMEIs; ICCIDs will not be returned in scan results even if it is present on the Agent and selected in the Scheduled Task policy.

Spirion Agent Searches

Custom data types defined by the Search API are automatically included in scans for Agent-initiated searches so long as the DLLs are located in the Plugins folder within the Spirion installation directory, which is "Program Files (x86)" by default.

Restricting Results

The only way to prevent their inclusion is to remove the DLL file prior to starting an Agent-initiated search.