Last modified: 10/28/2019

Hashing FAQs & Examples

What are hashing algorithms?

Hashing algorithms are functions that take a text input and return a hash code or message fingerprint of that input. The resulting hash code is a fixed length and will vary widely with small variations in input.

Why are hashing algorithms used?

One of the primary features of hashing algorithms is that they are one-way functions which means that it is nearly impossible to determine the original input based only on the hash code. Hashing algorithms power the security behind internet banking and allow Jornaya to receive data in a way that protects consumer privacy.

What algorithms are compatible with Jornaya Activate?

MD5 and SHA-256 are two popular algorithms that Jornaya Activate supports. Many databases and programming languages have built-in functions that will hash database content for you.

- MD5 algorithm always produces 32-character strings
- SHA-256 algorithm always produces 64-character strings

If the hashed fields being sent to Jornaya are not 32 or 64 characters, please check that you are using one of the supported hashing algorithms.

How are hashing and encryption different?

Encryption turns data into a series of unreadable characters that aren't of a fixed length. The key difference between encryption and hashing is that encrypted strings can be reversed back into their original decrypted form if you have the right key.

Hash Tips and Examples

When hashing values, keep in mind:

- Upper and lower case values produce different hash values (use lower case for Jornaya Activate)
- Leading/trailing white spaces will modify the hash values (please strip these out for Jornaya Activate)
- All characters impact the hash (please be sure to strip out parentheses, hyphens, and/or periods from phone numbers submitted to Jornaya Activate)

Hashing FAQs & Examples

Example 1: Using MySQL to hash a single email address

```
SELECT SHA2('test@example.com', 256) as email01;
```

RESULT:

email01
973dfe463ec85785f5f95af5ba3906eedb2d931c24e69824a89ea65dba4e813b

Example 2: Using MySQL to hash a single email address with trailing whitespace

```
SELECT SHA2('test@example.com ', 256) as email02;
```

RESULT:

email02
1f0a3bbdda1b7eb2c6688ac0ff5213b8e1fdc6cb9b0e419314262e0c98e8b517

***Note**: The resulting hash completely changes by adding a single space before applying the hashing algorithm.*

Example 3: Using MySQL to hash a 10-digit phone number

```
SELECT SHA2('6105551212', 256) as phone01;
```

RESULT:

phone01
230d628373ef5c69d80b0efe374cf9b5676fadba1946f78dd5d895c70290e8ee

Example 4: Using MySQL to hash a 10-digit phone number with hyphens

```
SELECT SHA2('610-555-1212', 256) as phone02;
```

RESULT:

phone02
c6a14a90387c06a0963360aafd01f7b401447cbe47cf4ae5c00d2a5c07891892

***Note**: The resulting hash completely changes by including hyphens before applying the hashing algorithm.*

Example 5: Using Microsoft SQL Server to hash a signal email address

```
SELECT ISNULL(
    CONVERT(varchar(255), HASHBYTES('SHA2_256', 'test@example.com'), 2), ''
) AS email01;
```

RESULT:

email01
973dfe463ec85785f5f95af5ba3906eedb2d931c24e69824a89ea65dba4e813b

Explanation:

1. HASHBYTES('SHA2_256', 'test@example.com') produces the hash value in binary.
2. CONVERT(varchar(255), HASHBYTES('SHA2_256', 'test@example.com'), 2) converts the binary output to a varchar.
3. ISNULL(CONVERT(varchar(255), HASHBYTES('SHA2_256', 'test@example.com'), 2), '') the hash value if available or an empty string.

Example 6: Using SAS to hash a single email address

```
proc sql;
    CREATE table temp_table AS
    SELECT
        sha256 (strip('test@example.com')) as email01 format $hex64. length=64
    FROM &source_table;
quit;
```

RESULT:

email01
973dfe463ec85785f5f95af5ba3906eedb2d931c24e69824a89ea65dba4e813b

Explanation using proc sql statement:

1. Create a temporary table 'temp_table' to store the result of the SQL query.
2. The source of the consumer's PII is source_table
3. strip('test@example.com') strips all whitespace from the input
4. sha256 (strip('test@example.com')) as email01 format $hex64. length=64 hashes the input, converting the default binary as a 64-character long hexadecimal string.