Tokenization User Guide

Last update: June 27, 2023

> **Content applies to flat data**
>
> The content in this article applies to tokenization of flat data (i.e., delimiter-separated columnar data with one record per row). Starting in v4.2, the Datavant CLI supports JSON and NDJSON input formats. For more information, refer to the Nested Data Tokenization User Guide.

Overview

This guide walks you through how to use Datavant to tokenize identified data, transform tokens to share with a partner, or transform tokens from a partner. Datavant's tokenization engine creates tokens, or de-identified patient keys, from different permutations of patient demographics, such as name, date of birth, gender, and social security number. The tokenization engine is required if you:

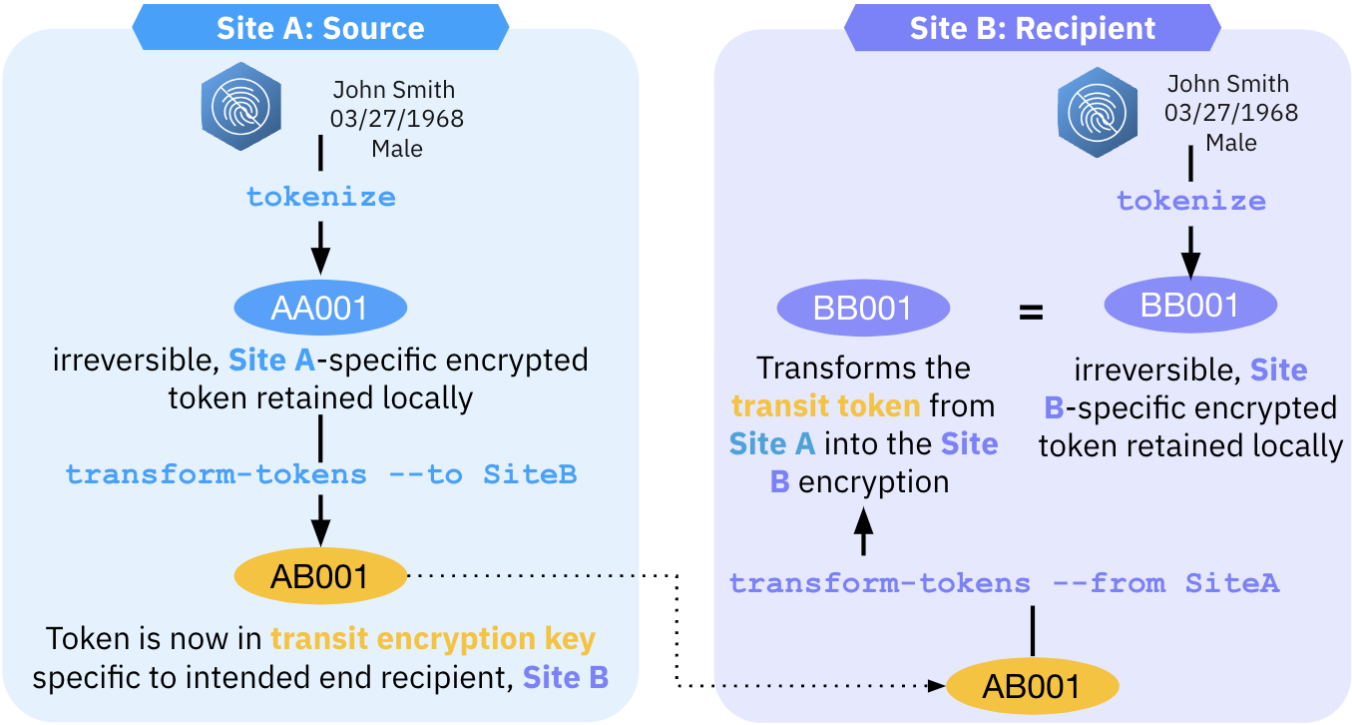
- are tokenizing first-party data
- transforming tokens to send to a partner for downstream linking or matching with other datasets
- transforming tokens you've received from a partner so they can be linked or matched with other datasets
- are transforming tokens to onboard to the Datavant portal for use with Assess, Match, or Distribution. For more information, refer to the Data Onboarding User Guide.

The tokenization engine facilitates two key functions via three operations:

1. Generating tokens
    - *tokenize*: creates *site-specific tokens* in a site-specific encryption key from patient demographics and applies de-identification operations to identified data; run by sites on first party identified data. In the case of nested data, if a token is created using nested data elements, the number of elements in the dictionary or list will be the number of tokens generated. (e.g: If there are two given names in the list, then two Token 1's will be generated).

2. Sharing tokens
    - *transform-tokens to:* creates transit tokens by transforming tokens from a site-specific encryption key to a *transit* encryption key that is specific to the sender and intended recipient; run by sites prior to sending any tokens outside their environment to partners.
    - *transform-tokens from:* creates site-specific tokens by transforming tokens from a transit encryption key to a site-specific encryption key; run by sites after receiving tokens from partners.

The diagram below illustrates how all three operations are used in a simple matching use case with two parties.



Determine How to Use the Datavant Tokenization Engine

Datavant's tokenization engine comes in several modes that, depending on your requirements and use case, can be deployed on-premise or in the Datavant cloud. The majority of customers

use the Datavant Command Line Interface (CLI) in batch mode, but you can choose from any of the following options:

- **Datavant CLI in batch mode:** in this mode, you install the CLI on-premise and run records through the engine batches (one file at a time, or a group of files at a time). For more details, refer to Use Datavant CLI in Batch Mode.
- **Datavant CLI in server mode:** in this mode, you install the CLI on-premise and send JSON-formatted records (up to 1000 records at a time) to a local API, allowing for record-by-record processing. For more details, refer to Use Datavant CLI in Server Mode.
- **Datavant CLI in streaming mode:** in this mode, you install the CLI on-premise and stream records in via stdin and/or out via stdout, allowing for continuous record processing. For more details, refer to Use Datavant CLI in Streaming Mode.
- **Datavant desktop:** in this mode, you install a graphical user interface (GUI) on-premise and run flat files through the engine in batches (one file at a time, or a group of files at a time). For more details, refer to Use Datavant Desktop.
- **Datavant cloud tokenization:** in this mode, you send PII (under a BAA) in flat files to the Datavant Cloud Tokenization environment, where it is tokenized and then uploaded to the Datavant portal. For more details, refer to Use Datavant Cloud Tokenization.

The following table illustrates the features and requirements of each mode:

Category	Feature	CLI Batch Mode	CLI Server Mode	CLI Streaming Mode	Datavant desktop	Cloud tokenization
Installation and Automation	Installation	On-premise	On-premise	On-premise	On-premise	Datavant cloud
	Integration effort	Minimal coding	More coding	More coding	No coding	Minimal to no coding
	Automation support	Yes	Yes	Yes	No	Yes
OS support	Mac support	Yes	Yes	Yes	Yes	N/A
	Windows support	Yes	Yes	Yes	Yes	N/A
	Linux support	Yes	Yes	Yes	No	N/A
File types and sizes, request types, and encoding options	CSV support	Optional	No	Optional	Required	Required
	JSON support	Optional	Required	Optional	No	No
	NDJSON support	Optional	Optional	No	No	No
	API support	No	Yes	No	No	No
	Character encoding other than UTF-8	Yes	Yes	Yes	No	No
	Large file support	Yes	Max 1000 records/request	Yes	Limited	Yes
Processing options	Multiple configs/partners allowed in same process	No	Yes	No	No	No
	Continuous processing support	No	Yes	Yes	No	No
	Run tokenization and token transformation in same process	No	Yes	No	Yes	Yes
	Customize file/log names	Full customization	N/A	Full customization	No customization	No customization

Use Datavant CLI in Batch Mode

Step 1. Complete technical pre-requisites for on-premise software

- Ensure your machine or environment is appropriately sized. Our recommended minimum specifications are 3GHz 64-bit quad-core processor with 8GB RAM.
- Ensure your OS is compatible with the mode you've chosen.
  - Datavant CLI batch/server/streaming modes:
    - Linux Ubuntu 18.04 or later; Red Hat v6.9 or later; CentOS v6.10 or later
    - Windows 8.1 or later; Windows 2016 Server or later
    - Mac OS 10.15 or later
  - Datavant desktop:
    - Windows 8.1 or later; Windows 2016 Server or later

    - Mac OS 10.15 or later

- Ensure the following endpoints are accessible outbound over port 443:
    - sec.datavant.com
    - auth.datavant.com
    - api.datavant.com
      These connections enable the application to access cryptographic secrets (the Datavant master salt and encryption keys), as well as software configuration information when tokenizing. We recommend allowing *.datavant.com.

- Ensure any proxy server is set to pass-through mode. If you have a proxy server, ensure it is configured to pass through the incoming SSL certificate from Datavant's endpoints, as opposed to passing back its own self-signed certificate. A direct SSL connection is required with the endpoints listed above to ensure that the Datavant application and sensitive data are protected.
- Verify network connectivity by running the below command. If network is configured correctly, all tests will pass. If tests fail, refer to the Troubleshooting section.

Mac

```
./Datavant_Mac diagnose
```

Windows

```
Datavant_Win diagnose
```

Linux

```
./Datavant_Linux diagnose
```

Step 2. Create a configuration (only if running tokenize)

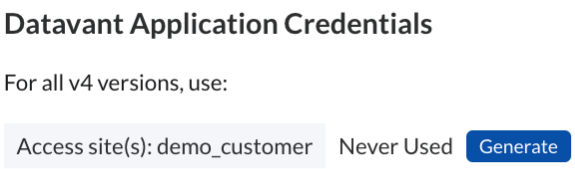
Refer to the Configurations user guide for instructions on creating a configuration to tokenize data. If you are only tokenizing data, create a **Tokenization** configuration. If you will later onboard the data to the Datavant portal, create a **Tokenization and Data Onboarding** configuration.

Step 3. Download the application

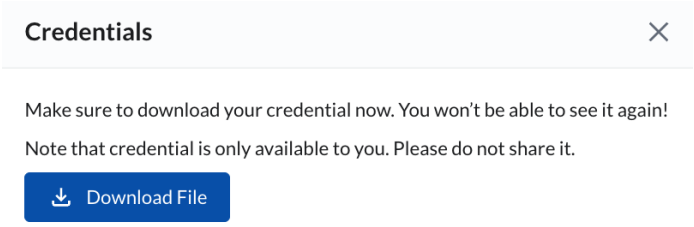
- In the Datavant portal, go to the **Download** page.
- Adjust the drop-down for your operating system. The CLI is available on Mac, Windows, and Linux. Datavant desktop is available on Mac and Windows.
- Select the download link for either the CLI or Datavant desktop.

Step 4. Generate user credentials

- In the Datavant portal, go to the **Download** page.
- Scroll to the **Datavant Application Credential** section and click **Generate**.



- Click **Download File** to download the credentials.txt file to your computer. Do not skip this step! Once you generate the credential, you can't view it again.



> ### Important
>
> The lifetime of your credential is 10 years. You and any admins on your account will receive email notifications when your credential is close to expiring and has expired. To reset it, follow the same steps as above. **Do not share your credential** with anyone else, including Datavant employees.

Step 5. Ensure your input data conforms to validation rules (tokenization only)

Refer to Data Hygiene Best Practices for validation rules.

Step 6. Ensure your input data is properly formatted

The format of your data depends on whether you are running the **tokenize**, **transform-tokens to**, or **transform-tokens from** operation.

- **tokenize**: Your input data should be a flat file containing rows of identified data with columns in the order specified in your configuration. Each row should represent one record. Refer to the Format tab of your configuration for the specified delimiter, header, and quoting level information required for your file. By default, UTF-8 character encoding is expected. To use input data files encoded in iso-8859-1 through iso-8859-16 or cp-1252, use the --input-encoding argument to specify the character encoding. You may tokenize multiple files at once provided they all have the same layout.
  - You can download a test input file by clicking on the details menu next to your configuration on the Configurations page > **Download Test Input File**.
- **transform-tokens to**: Your input data should be the output from the **tokenize** operation. No additional modification is needed prior to running this operation.
- **transform-tokens from**: Your input data should be the files sent to you by your partner. No additional modification is needed prior to running this operation.

Step 7. Build your command

Mac, tokenize

```
cat [CREDENTIALS_FILENAME] | ./Datavant_Mac tokenize \
-s [MY_SITE_NAME] \
-c [CONFIG_NAME] \
-i [INPUT_FILE] \
-o [OUTPUT_FILE] \
[optional_arguments]
```

Example:

```
cat credentials.txt | ./Datavant_Mac tokenize \
-s demo_customer \
-c demo_config \
-i demo_input.csv \
-o demo_tokenized_output.csv
```

Mac, transform-tokens to

```
cat [CREDENTIALS_FILENAME] | ./Datavant_Mac transform-tokens \
--to [PARTNER_SITE_NAME] \
-s [MY_SITE_NAME] \
-i [INPUT_FILE] \
-o [OUTPUT_FILE] \
[optional_arguments]
```

Example:

```
cat credentials.txt | ./Datavant_Mac transform-tokens \
--to demo_customer_partner \
-s demo_customer \
-i demo_tokenized_output.csv \
-o demo_output_transformed.csv
```

Mac, transform-tokens from

```
cat [CREDENTIALS_FILENAME] | ./Datavant_Mac transform-tokens \
--from [PARTNER_SITE_NAME] \
-s [MY_SITE_NAME] \
-i [INPUT_FILE] \
-o [OUTPUT_FILE] \
[optional_arguments]
```

Example:

```
cat credentials.txt | ./Datavant_Mac transform-tokens \
--from demo_customer_partner \
-s demo_customer \
-i demo_output_transformed.csv \
-o demo_output_tokenized.csv
```

Windows, tokenize

via PowerShell:

```
type [CREDENTIALS_FILENAME] | .\Datavant_Win.exe tokenize `
-s [MY_SITE_NAME] `
-c [CONFIG_NAME] `
-i [INPUT_FILE] `
-o [OUTPUT_FILE] `
[optional_arguments]
```

Example:

```
type credentials.txt | .\Datavant_Win.exe tokenize `
-s demo_customer `
-c demo_config `
-i demo_input.csv `
-o demo_output_tokenized.csv
```

via cmd:

```
type [CREDENTIALS_FILENAME] | Datavant_Win.exe tokenize ^
-s [MY_SITE_NAME] ^
-c [CONFIG_NAME] ^
-i [INPUT_FILE] ^
-o [OUTPUT_FILE] ^
[optional_arguments]
```

Example:

```
type credentials.txt | Datavant_Win.exe tokenize ^
-s demo_customer ^
-c demo_config ^
-i demo_input.csv ^
-o demo_output_tokenized.csv
```

Windows, transform-tokens to

via PowerShell:

```
type [CREDENTIALS_FILENAME] | .\Datavant_Win.exe transform-tokens `
--to [PARTNER_SITE_NAME] `
-s [MY_SITE_NAME] `
-i [INPUT_FILE] `
-o [OUTPUT_FILE] `
[optional_arguments]
```

Example:

```
type credentials.txt | .\Datavant_Win.exe transform-tokens `
--to demo_customer_partner `
-s demo_customer `
-i demo_output_tokenized.csv `
-o demo_output_transformed.csv
```

via cmd:

```
type [CREDENTIALS_FILENAME] | Datavant_Win.exe transform-tokens ^
--to [PARTNER_SITE_NAME] ^
-s [MY_SITE_NAME] ^
-i [INPUT_FILE] ^
-o [OUTPUT_FILE] ^
[optional_arguments]
```

Example:

```
type credentials.txt | Datavant_Win.exe transform-tokens ^
--to demo_customer_partner ^
-s demo_customer ^
-i demo_output_tokenized.csv ^
-o demo_output_transformed.csv
```

Windows, transform-tokens from

via Powershell:

```
type [CREDENTIALS_FILENAME] | .\Datavant_Win.exe transform-tokens `
--from [PARTNER_SITE_NAME] `
-s [MY_SITE_NAME] `
-i [INPUT_FILE] `
-o [OUTPUT_FILE] `
[optional_arguments]
```

Example:

```
type credentials.txt | .\Datavant_Win.exe transform-tokens `
--from demo_customer_partner `
-s demo_customer `
-i demo_tokenize_transformed.csv `
-o demo_output_tokenized.csv
```

via cmd:

```
type [CREDENTIALS_FILENAME] | Datavant_Win.exe transform-tokens ^
--from [PARTNER_SITE_NAME] ^
-s [MY_SITE_NAME] ^
-i [INPUT_FILE] ^
-o [OUTPUT_FILE] ^
[optional_arguments]
```

Example:

```
type credentials.txt | Datavant_Win.exe transform-tokens ^
--from demo_customer_partner ^
-s demo_customer ^
-i demo_tokenize_transformed.csv ^
-o demo_output_tokenized.csv
```

Linux, tokenize

```
cat [CREDENTIALS_FILENAME] | ./Datavant_Linux tokenize \
-s [MY_SITE_NAME] \
```

```
-c [CONFIG_NAME] \
-i [INPUT_FILE] \
-o [OUTPUT_FILE] \
[optional_arguments]
```

Example:

```
cat credentials.txt | ./Datavant_Linux tokenize \
-s demo_customer \
-c demo_config \
-i demo_input.csv \
-o demo_tokenized_output.csv
```

Linux, transform-tokens to

```
cat [CREDENTIALS_FILENAME] | ./Datavant_Linux transform-tokens \
--to [PARTNER_SITE_NAME] \
-s [MY_SITE_NAME] \
-i [INPUT_FILE] \
-o [OUTPUT_FILE] \
[optional_arguments]
```

Example:

```
cat credentials.txt | ./Datavant_Linux transform-tokens \
--to demo_customer_partner \
-s demo_customer \
-i demo_tokenized_output.csv \
-o demo_output_transformed.csv
```

Linux, transform-tokens from

```
cat [CREDENTIALS_FILENAME] | ./Datavant_Linux transform-tokens \
--from [PARTNER_SITE_NAME] \
-s [MY_SITE_NAME] \
-i [INPUT_FILE] \
```

```
-o [OUTPUT_FILE] \
[optional_arguments]
```

Example:

```
cat credentials.txt | ./Datavant_Linux transform-tokens \
--from demo_customer_partner \
-s demo_customer \
-i demo_output_transformed.csv \
-o demo_output_tokenized.csv
```

Optional Arguments

Argument	Tokenize or Transform Tokens	Description	Behavior
`-h, -help`	Both	Print a help message	Show a help message for how to use the application and exit
`-v, --version`	Both	Print version number	The version number will be printed to the console. Use this parameter when troubleshooting with Datavant.
`--delimiter`	Tokenize	Specify file delimiter	Specify the delimiter used to separate fields in your input data. If this is not present, DeID will check your configuration.
`--num-threads`	Both	Specify the number of sub-	By default, 1 thread is used. If performance improvements are

		files to process in parallel	needed, we recommend setting this to the number of cores in your environment. This will create temporary files that are processed in parallel, which will require extra storage to be available (between 1-2x the size of the input file) in both the input and output directories. Set the argument to `auto` to optimize the number of threads used for processing based on system processor capabilities.
`--error-codes-suppressed-in-output`	Tokenize	Suppress error codes in output files	Output files will not contain error tokens. Instead, tokens that fail to generate will generate no output (i.e., an empty string).
`--dev-log-path`	Both	Specify the path to write the log file	Override the default name and location for dev-log files.
`--token-log-path`	Both	Specify the path to write the token error log file	Override the default name and location for token error log files.
`--no-logs`	Both	Suppress all logging	No log files will be

			generated.
`--console-log`	Both	Output exceptions to the console instead of to a file.	All runtime exception log messages will be output to the console instead of generating an exception log file.
`--disable-output-reordering`	Tokenize	Ensures output row order exactly matches the input	Instead of randomizing the order of output rows (the default for privacy reasons), the output row order will match the input row order.
`--token-columns`	Transform tokens	Specify the columns that should be processed	Explicitly specify the token columns that should be transformed. This overrides token auto-detection. Columns should be 0-indexed and space separated. For example, `--token-columns 0 1 3` will transform tokens in the first, second, and fourth columns.
`--credits`	Both	Display software packages	Display a URL to the Datavant portal, with list of packages and licenses used to build the application.
`--environment-credentials`	Both	Allows passing in the password as	Use the password set in your environment as

		an environment variable	`DV_USER_CREDENTIALS` to pass in the password as an environment variable.
`--input-encoding`	Both	Specify the character encoding of the file	Use an encoding other than UTF-8 for input files. Supported encodings are `UTF-8`, `iso-8859-1` through `iso-8859-16`, and `cp-1252`.
`--input-format`	Both	Specify the input file format	Use an input file format other than csv. Supported formats are `csv` and `json`.
`--output-format`	Both	Specify the output file format	Use an output file format other than csv. Supported formats are `csv` and `json`.
`--local-ip-address`	Both	Specify the local IP address to run on (v3.8+)	Sets the IP address on which to run the application as a server. If the argument is not used, localhost is used. This IP address must be a private address. This argument is only allowed when using `serve`.

`--port`	Both	Specify the network port to run on (v3.8+)	Sets the network port on which to run the application as a server. If the argument is not used, port 8250 is used. This argument is only allowed when using `serve`.
`--silent`	Both	Specify if console output should be suppressed	Suppresses output to the console
`onboard`	Neither	Specify if requesting to upload data to the Datavant portal (v4.1+)	Onboards data to the Datavant portal to be used in Profiles, Overlaps, and more.
`--dataset`	Neither	Specify location for data upload to the Datavant portal (v4.1+)	Sets the location for data upload.
`--data-quality-log-path`	Tokenize	Specify the path to write the data quality log file (v4.1+)	Override the default name and location for the data quality log file.
`--skip-name-preprocessing`	Tokenize	Skip pre-processing steps of first and last name PII elements introduced in version 4.1. (v4.3+)	Overrides the default pre-processing steps of first and last name PII elements.

Step 8. Run the CLI in Batch Mode

- Open a command line window
- If you are using Mac or Linux, run the following command to enable executable permissions on the application:

```
chmod 755 Datavant_Mac
```

```
chmod 755 Datavant_Linux
```

- Paste in the command you built and press Enter.

Step 9. Review the output and log files

The output from the on-premise application depends on which operation you've run. Any errors will be sent to the console (if console output is enabled) and also visible in the dev log.

Outputs

- **tokenize:** the output from **tokenize** is a file (or files) that has had remediations applied to it, per your configuration, and with tokens in your site-specific encryption key. Before you send it to a partner, you must run **transform-tokens to.** The following log files are available:
  - dev log
  - token error log
  - data quality log (starting in v4.1)
- **transform-tokens to**: the output from **transform-tokens to** is a file (or files) in which the tokens have been transformed, or re-encrypted, from your site-specific encryption key to a transit encryption key that is shared between your site and your partner site. This file is ready to be sent to a partner. The following log files are available:
  - dev log
  - token error log
- **transform-tokens from**: the output from **transform-tokens from** is a file (or files) in which the tokens have been transformed, or re-encrypted, from a transit encryption key that is shared between your site and your partner site, to your site-specific encryption key. This file is ready to be linked with other tokens in your site-specific encryption key. The following log files are available:

- dev log
- token error log

Dev log file

The dev log file contains all messages sent to the console throughout the run, as well as error messages for fields or records that failed processing. For unexpected errors, stack traces may also be included in the logs.

- If the optional **--dev-log-path** command line argument is not present, the dev log will be written to the same directory the executable is in and given a name of **datavant_{YYYYMMDDHHMMSS}.log**, where YYYYMMDD is the date and HHMMSS is the local time.

Token error log file

The token error log is created whenever error tokens are generated within the software. When running tokenize, an error token is created if an input field used to create a token was missing or did not pass validation rules. *In general*, error tokens are expected unless you have 100% fill rates on all input fields and all inputs are valid for tokenization. You can use the data quality log produced from the **tokenize** operation to quickly assess the quality and completeness of your input data.

If the optional **--token-log-path** command line argument is not present, the token error log will be written to "token_errors_{YYYYMMDDTHHMMSS}.log, where YYYYMMDD is the date and the HHMMSS is the local time.

Each row in the token error log file contains 3 elements: Line #, [ERROR_CODE], [ERROR_NAME]. For example, if row 5 of the input file has an invalid social security number, the following row would be added to the log file:

```
Line 5, XXX - S00000, The given value represents an invalid SSN.
```

Error code format

Error tokens follow a standard format: they will be 12-character strings that begin with "XXX -" followed by 6 characters describing the error(s). An error token may be generated due to multiple errors, and each individual error is represented by a single character (see the Error Code List). If fewer than 6 errors occur, the error code will be padded with zeros (0) at the end so that the token is a consistent 12-character length.

Examples:

- XXX - L00000: last name was missing or invalid
- XXX - LFGD00: last name, first name, gender, and date of birth were all missing or invalid.
- XXX - T00000: token transformation failed

Error code list

- L: last name was missing or invalid
- F: first name was missing or invalid
- G: gender was missing or invalid
- D: date of birth was missing or invalid
- S: SSN was missing or invalid
- Z: ZIP code was missing or invalid
- E: email was missing or invalid
- P: cell phone was missing or invalid
- I: patient ID was missing or invalid
- A: address was missing or invalid
- U: state was missing or invalid
- M: state was missing or invalid
- C: in a custom token, a custom field was missing or invalid

Data quality log file

The data quality log is created when running the **tokenize** operation.

If the optional **--data-quality-log-path** command line argument is not present, the data quality log will be written to "dataquality_{YYYYMMDDTHHMMSS}.log, where YYYYMMDD is the date and the HHMMSS is the local time.

The data quality log is split into 3 sections:

- Possible Issues Detected:
  - Lists all tokens with success rates < 90%, all PII fields with fill rates < 90%, and any possible filler values or instances of date rounding if the input data was at least fifty entries

long.

- Filler data for dates is flagged if there is a statistically significant clustering of dates on the first or fifteenth of the month.
- Filler data for names is flagged if there is a statistically significant clustering names after adjusting for age (based on SSA data)

```
============================ POSSIBLE ISSUES DETECTED ==================
We detected the following possible issues with your data. For more deta
information, see the sections below.
36.36% (4/11) of token_107 generated successfully.
36.36% (4/11) of token_108 generated successfully.
36.36% (4/11) of token_109 generated successfully.
```

- Token Error Summary:
  - For each token type in the file, details how many generated error tokens
  - For each PII field that contributed to an error token, details the errors present and the count of error tokens that resulted from that type of error

```
============================== TOKEN ERROR SUMMARY =====================
Errors occurred while creating tokens from your data. The token table li
number of error tokens of each token type. The PII Field table lists the
fields that caused these error tokens along with the reason for the erro
Token                 Error Count
-----------------  -------------
token_5                         1
token_106                       1
token_107                       7
token_108                       7
token_109                       7

PII Field             Error Count  Detailed Error
----------------  -------------  ------------------------------------
ssn                             1  The given value represents an invalid
phone                           6  The given value uses an invalid area c
phone                           1  The given value uses an invalid prefi
zip_code                        1  Zip code is formatted incorrectly. Mus
```

- Comprehensive Data Quality Information:
  - Lists the fill rate for each type of token
  - Lists the fill for each PII field

```
==================== COMPREHENSIVE DATA QUALITY INFORMATION ==========
Tokens were successfully created at the following rates and PII was dete
with the following fill rates.

Token Name           Tokens Created    Total    Success Rate
-----------------  ----------------  -------  --------------
token_1                         11       11         100.00%
token_2                         11       11         100.00%
token_5                         10       11          90.91%
token_101                       11       11         100.00%
token_102                       11       11         100.00%
token_103                       11       11         100.00%
token_106                       10       11          90.91%
token_107                        4       11          36.36%
token_108                        4       11          36.36%
token_109                        4       11          36.36%
token_110                       11       11         100.00%
token_111                       11       11         100.00%

PII Field            Records Filled    Total        Fill Rate
-----------------  ----------------  -------  ---------------
first_name                      11       11          100.00%
last_name                       11       11          100.00%
date_of_birth                   11       11          100.00%
gender                          11       11          100.00%
ssn                             11       11          100.00%
zip_code                        11       11          100.00%
email                           11       11          100.00%
phone                           11       11          100.00%
```

Troubleshooting

If you run into errors or issues, refer to the Troubleshooting section.

Have more questions? Submit a request