The Journal of Open Source Software

X12-837-Fake-Data-Generator: An Open-Source Python Package for Generating and Parsing Synthetic Healthcare Claims

Hants Williams [1]

1 Stony Brook University, Departments of Applied Health Informatics and Biomedical Informatics

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Healthcare claims data in X12 837 format represents the standard for electronic healthcare claim submission in the United States, and is mandated by the federal government for processing billions of dollars in healthcare transactions annually. However, access to real claims data is severely restricted due to patient privacy regulations, creating significant barriers for healthcare informatics education, software development, and research. The `X12-837-Fake-Data-Generator` is an open-source Python package that generates realistic but synthetic X12 837 claim files, in addition to parsing them into structured CSV datasets for analysis.

This package combines authentic healthcare reference data from CMS databases (provider NPIs, insurance payers, ICD-10 diagnoses, CPT procedures) with synthetic patient demographics to create privacy-compliant test data that maintains the complexity and structure of real healthcare claims.Lastly, providing the capabilities of both 837 generation and parsing through multiple interfaces (command-line, web application, and REST API), this package democratizes access to healthcare claims data for education, testing, and research purposes while eliminating privacy concerns.

Statement of Need

The X12 formatted 837 file is the standard for healthcare claim submission, used by providers, payers, and clearinghouses to process healthcare services. Understanding its structure is essential for healthcare informaticists learning data standards and interoperability, software developers building claims processing and revenue cycle management systems, data analysts performing utilization analysis and cost management, healthcare administrators testing system integrations and workflows, and researchers studying claims-based outcomes and population health.

Despite the ubiquity of claims data, accessing the raw versions of X12-formatted 837 files for learning or testing is difficult due to HIPAA restrictions on Protected Health Information (PHI) and Personally Identifiable Information (PII) contained within these files. There are existing solutions and solutions that can be explore but they all have significant limitations. First are the commercial X12 test generators which can cost thousands of dollars annually, and provide limited customization (Accredited Standards Committee X12, 2023). A second approach could be going with manual claim creation, which is time-intensive, error-prone, and not scalable. While a third approach could be de-identifying real data, which requires access to data in the first place, and carries re-identification risks and requires IRB approvals and data use agreements. Lastly, the few clinical data generators that we were able to find like Synthea (Walonoski et al., 2018) focus on EHR data rather than billing and claims formats, while the CMS synthetic data (SynPUF) (Centers for Medicare & Medicaid Services, 2013) is

Medicare-only, limited to specific years, and does not provide the data in the raw X12 format.

We are not aware of any existing open-source tool that provides both generation and parsing of realistic X12 837 claims, with validation against current X12 standards. `X12-837-Fake-Data-Generator` fills this critical gap by enabling privacy-compliant (fake) claims data generation.

The package provides realistic medical content using real ICD-10 diagnoses, CPT procedures, provider NPIs, and insurance payers from authoritative CMS databases. It generates synthetic patient data using validated algorithms through the Faker library, ensuring zero re-identification risk. The system produces standards-compliant X12 transactions verifiable with industry validators such as Stedi EDI Inspector and DataInsight Health. The package also combines both generation and parsing functionality in a single unified system, supporting diverse user needs through multiple interfaces including CLI for automation, web UI for accessibility, and REST API for integration. Comprehensive documentation and examples make the package particularly valuable for teaching healthcare data standards.

This package is particularly valuable for academic institutions teaching health informatics, healthcare organizations testing claims systems, and researchers needing reproducible synthetic data for methods development.

Architecture and Implementation

System Design

`X12-837-Fake-Data-Generator` consists of two primary modules, with three shared interface implementations:

1. Generator Module (`generator_837/`)

The generator creates synthetic X12 837 claims through a four-stage pipeline:

Stage 1 (reference data loading): the system loads medical coding systems and provider/payer databases including ICD-10-CM diagnosis codes (approximately 72,000 codes, 14 MB), CPT procedure codes (a subset of approximately 10,000 codes, 249 KB), the healthcare payer database from Healthcare.gov (approximately 4,000 insurers, 4 MB), the NPPES National Provider Identifier registry (approximately 6 million providers, 29 MB), and hospital and facility data from CMS (approximately 6,000 facilities, 1.4 MB).

Stage 2 (entity selection): the system randomly selects realistic healthcare organizations and payers from actual CMS databases, ensuring authenticity without privacy concerns since organizations are public entities.

Stage 3 (patient synthesis): the system generates synthetic patient demographics using the Faker library (Faker Contributors, 2024), producing statistically realistic but entirely fictional individuals including names, addresses, dates of birth, and identifiers.

Stage 4 (transaction assembly): the system constructs complete X12 837 transactions. This includes envelope segments (ISA, GS, ST) with control numbers, header loops for submitter, receiver, and billing provider information, subscriber loops containing patient demographics, claim information with diagnosis codes in HI segments, service lines with procedures including CPT codes, charges, and diagnosis pointers, and trailer segments (SE, GE, IEA) with validated segment counts. Generated claims include 3-8 diagnoses and 1-5 service lines per claim, with diagnosis pointers linking services to relevant diagnoses following Medicare billing requirements.

2. Parser Module (`parser_837/`)

The parser extracts structured data from X12 837 files into three normalized CSV tables: a header file, a diagnoses file, and a service line file. The header table contains transaction metadata, provider information, and subscriber demographics. The diagnoses table includes ICD-10 codes with qualifiers and sequence numbers. The service lines table captures CPT codes, charges, units, service dates, and diagnosis pointers. The parser preserves critical relationships, particularly diagnosis-to-service linkages, which are essential for claims analytics and enable downstream analysis of utilization patterns, costs, and quality metrics.

Technical Implementation

The package has only been tested in Python 3.11 or higher. Core dependencies include Faker (Faker Contributors, 2024) for synthetic demographic data generation, Pandas (McKinney, 2010) for reference data loading and CSV operations, Flask (Pallets Projects, 2023) as the web application framework, Flask-RESTx (Flask-RESTX Contributors, 2023) for REST API development with automatic OpenAPI/Swagger documentation, and the regex library for advanced pattern matching in X12 segment parsing.

Code Organization:

```
x12-837-fake-data-generator/
├── generator_837/
│   ├── api/              # Core generation logic
│   ├── cli/              # Command-line interface
│   └── web/              # Flask web application
├── parser_837/
│   ├── api/              # Core parsing logic
│   ├── cli/              # Command-line interface
│   └── web/              # Flask web application
└── web/                  # Combined web application
    ├── app.py            # Unified Flask app
    └── api.py            # REST API endpoints
```

Validation

Generated X12 837 transactions are validated against industry-standard EDI validators including Stedi EDI Inspector (https://www.stedi.com/edi/inspector) and DataInsight Health EDI Viewer (https://datainsight.health/edi/viewer/). These third-party tools verify syntax compliance with X12 specifications, including proper segment structure, element positioning, data types, and control number accuracy.

Installation and Usage

Installation

The package is available on GitHub and can be installed using pip:

```
git clone https://github.com/hantswilliams/x12-837-fake-data-generator.git
cd x12-837-fake-data-generator
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install -r requirements.txt
```

Command-Line Interface

Generate 837 Claims

```
# Generate 10 synthetic 837 claim files
python -m generator_837.cli.main -n 10 -o output_directory/
```

Parameters: - -n, --number: Number of claim files to generate (default: 1) - -o, --output: Output directory path (default: generator_837_output/)

Parse 837 Claims

```
# Parse directory of 837 files
python -m parser_837.cli.main -i input_directory/ -o parsed_output/

# Parse single file
python -m parser_837.cli.main -i claim_file.txt -o parsed_output/
```

Parameters: - -i, --input: Input file or directory containing 837 files - -o, --output: Output directory for parsed CSV files

Web Application

A unified web interface provides both generation and parsing capabilities:

```
python web/app.py
# Navigate to http://localhost:5007
```

The web application features form-based generation with customizable claim count (1-25 files), file upload for parsing with ZIP download of parsed CSVs, and interactive Swagger API documentation at the /api endpoint.

REST API

The Flask-RESTx API provides programmatic access:

Generate Claims

```
# Generate single claim (returns text file)
curl -X GET "http://localhost:5007/api/generate/" \
     --output single_claim.txt

# Generate multiple claims (returns ZIP archive)
curl -X POST "http://localhost:5007/api/generate/" \
     -H "Content-Type: application/json" \
     -d '{"number": 5}' \
     --output claims.zip
```

Parse Claims

```
# Upload and parse 837 file (returns ZIP of 3 CSVs)
curl -X POST "http://localhost:5007/api/parse/" \
     -F "file=@claim_file.txt" \
     --output parsed_data.zip
```

Example X12 Generated Output

Generated 837 files follow X12 format:

```
ISA*00*          *00*          *ZZ*SUBMITTER123   *ZZ*RECEIVER456    *241106*1430*^*00501*
GS*HC*SUBMITTER123*RECEIVER456*20241106*1430*1*X*005010X223A2~
ST*837*0001*005010X223A2~
BHT*0019*00*1234*20241106*1430*CH~
NM1*41*2*HEALTHCARE CLINIC*****46*1234567890~
NM1*40*2*BLUE SHIELD OF CA*****46*9876543210~
```

```
HL*1**20*1~
NM1*85*2*GENERAL HOSPITAL*****XX*1234567890~
HL*2*1*22*0~
NM1*IL*1*SMITH*JOHN****MI*123456789~
CLM*CLAIM001*1000.00***11:A:1*Y*A*Y*Y~
HI*ABK:E119*ABF:I10*ABF:E785~
LX*1~
SV1*HC:99213*250.00*UN*1***1~
DTP*472*D8*20241105~
SE*15*0001~
GE*1*1~
IEA*1*000000001~
```

Use Cases

This package enables diverse applications across healthcare informatics, in particular within the domains of education, software development, research, and quality improvement.

1. Education and training: supports teaching X12 EDI standards in health informatics curricula, facilitating hands-on laboratories for claims processing workflows, enabling understanding of revenue cycle management without privacy concerns, and providing practical training for medical billing and coding specialists who need experience with real-world claim structures.
2. Software development and testing contexts: enables unit and integration testing for claims processing systems, supports ETL pipeline development for claims data warehouses, validates adjudication logic and payment calculations, and facilitates load testing of clearinghouse and payer submission systems without requiring access to protected health information.
3. Research and analytics applications: enables development of claims analysis algorithms without data access barriers, supports testing of risk adjustment and predictive models, facilitates prototyping of population health dashboards and visualizations, and provides reproducible synthetic data for methods papers and algorithm validation studies.
4. Quality improvement initiatives: benefit from simulation of quality measure calculations such as HEDIS and CMS Stars metrics, testing of prior authorization systems, validation of denial management workflows, and auditing of billing compliance rules in a risk-free environment.

Comparison to Similar Tools

Tool	X12 Claims	Synthetic	Open Source	Parser	Cost
This package	Yes	Yes	Yes	Yes	Free
Synthea	No (EHR)	Yes	Yes	N/A	Free
CMS SynPUF	Yes	Yes	No	No	Free
Commercial EDI tools	Yes	Limited	No	Limited	$5K-$50K
Stedi/DataInsight	Validate only	No	No	Yes	Free/Paid

Synthea (Walonoski et al., 2018) excels at generating longitudinal patient clinical data but does not produce X12 billing transactions, focusing instead on electronic health record formats. CMS SynPUF (Centers for Medicare & Medicaid Services, 2013) provides Medicare synthetic claims but delivers data in research file format rather than native X12 format, is limited to the elderly Medicare population, and uses data that could be up to a decade old. Commercial EDI generators produce valid X12 transactions but are expensive proprietary black boxes with annual costs ranging from $5,000 to $50,000, making them unsuitable for education or research reproducibility where open access to methods is essential. While this package uniquely combines X12 format compliance, synthetic data generation, comprehensive parsing capabilities, and open-source accessibility in a single unified system.

Future Enhancements

Planned developments will expand the package's capabilities across multiple dimensions. (1) Integration with Synthea to enable generation of claims linked to patient clinical trajectories. This would allow correlation between clinical events and billing data. (2) Support for additional claim types including 837P (professional claims) and 837D (dental claims) will broaden the package's applicability beyond institutional claims. (3) Development of longitudinal data capabilities will enable generation of multi-claim patient histories suitable for outcomes research and care coordination studies. (4) Enhanced geographic realism will correlate patient locations with nearby providers using the National Address Database, improving authenticity of provider-patient relationships. (5) Payer-specific companion guide rule checking will strengthen validation capabilities beyond basic X12 syntax compliance. (6) Denial simulation features will generate claims with common billing errors to support training in claims correction and resubmission workflows. Finally, (7) bidirectional transformation capabilities between X12 and HL7 FHIR Claims resources (Health Level Seven International, 2023) will facilitate interoperability between traditional EDI and modern FHIR-based healthcare data exchange standards.

Acknowledgments

This work utilizes publicly available healthcare databases from the Centers for Medicare & Medicaid Services (CMS), including the National Provider Identifier (NPI) registry and Healthcare.gov insurance plan data. The author thanks the open-source community for foundational libraries (Faker, Flask, Pandas) that made this package possible.

Code Generation/LLM Use

Claude Code (v2, Opus 4.5 – Claude Max) assisted in software development and manuscript preparation. For development, it was used to iterate on and refine the functional logic in `generator_837/api/segments.py` and `generator_837/api/generator.py`, and to draft and revise unit tests in `generator_837/tests/test_generator.py`. For the manuscript, Claude Code was primarily used to: (a) check for grammatical and syntactic errors, (b) reformat and perform edits to the markdown code blocks and some inline code examples, and (c) suggest minor wording changes for clarity and internal consistency of concepts. All output from Claude Code's assistant was manually reviewed, and revised where needed.

References

Accredited Standards Committee X12. (2023). *HIPAA 5010 Implementation Guides*. https://x12.org/products/hipaa-transaction-sets.

Centers for Medicare & Medicaid Services. (2013). *CMS 2008-2010 Data Entrepreneurs' Synthetic Public Use File (DE-SynPUF)*. U.S. Department of Health; Human Services; https://www.cms.gov/Research-Statistics-Data-and-Systems/Downloadable-Public-Use-Files/SynPUFs.

Faker Contributors. (2024). *Faker: Python package that generates fake data for you*. https://github.com/joke2k/faker.

Flask-RESTX Contributors. (2023). *Flask-RESTX: Fully featured framework for fast, easy and documented API development with flask*. https://flask-restx.readthedocs.io/.

Health Level Seven International. (2023). *HL7 FHIR Claim Resource*. https://www.hl7.org/fhir/claim.html.

McKinney, W. (2010). *Data structures for statistical computing in python* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Pallets Projects. (2023). *Flask: A python microframework*. https://flask.palletsprojects.com/.

Walonoski, J., Kramer, M., Nichols, J., Quina, A., Moesel, C., Hall, D., Duffett, C., Dube, K., Gallagher, T., & McLachlan, S. (2018). Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, *25*(3), 230–238. https://doi.org/10.1093/jamia/ocx079