Python-JXES: Python Implementation of JSON Serialization for XES Standard

Maxim Vidgof[1]

[1]Vienna University of Economics and Business (WU Wien), Welthandelsplatz 1, 1020 Vienna, Austria

Abstract

Process mining requires event logs. XES is a widely accepted format for storing and exchanging event log data, however, it only has XML serialization, leading to sub-optimal storage and time requirements in certain scenarios. JXES is a recently proposed serialization format for XES based on JSON, a more lightweight data interchange format. This paper presents Python-JXES: a Python implementation of JXES. Its read and write performance is evaluated against a certified state-of-the-art tool, and file sizes of JXES and XES serializations of open-source event logs are compared. JXES achieves up to 33% storage savings and up to 73% faster read speeds. Python-JXES can be used to facilitate efficient event log storage, streaming process mining and process mining on IoT devices.

Keywords

JXES, XES, Event log format, Event data, Process mining

Metadata description	Value
Tool name	Python-JXES
Current version	0.1
Legal code license	Apache 2.0
Languages, tools and services used	Python
Supported operating environment	Microsoft Windows, GNU/Linux
Download/Demo URL	https://pypi.org/project/jxes/
Documentation URL	https://github.com/MaxVidgof/python-jxes
Source code repository	https://github.com/MaxVidgof/python-jxes
Screencast video	https://youtu.be/8adiYqeczAs

1. Introduction

Exchanging event logs is crucial for research and practice of Process Mining (PM) [1]. XES is a widely accepted XML-based standard for event log serialization, facilitating exchange of event logs. Despite the flexibility and tool support of XML, however, JSON is gaining popularity as a more lightweight data interchange format. This can be seen by the fact that newer standards for event log data start offering JSON serialization: OCEL [2] offers JSON serialization in addition to XML and SQLite, and some OCED reference implementations [3] also rely on OCEL-JSON for serializing OCED data.

ICPM 2024 Tool Demonstration Track, October 14-18, 2024, Kongens Lyngby, Denmark
maxim.vidgof@wu.ac.at (M. Vidgof)
https://complex.wu.ac.at/nm/vidgof (M. Vidgof)
0000-0003-2394-2247 (M. Vidgof)

JSON serialization for XES has already been proposed by Narayana, Khalifa & van der Aalst [4] but has not received its well-deserved attention and was only implemented as ProM plugin. This paper presents Python implementation of JXES and showcases its benefits by comparing it to a certified state-of-the-art XES implementation. It shows how JXES can be beneficial in scenarios like event log storage, Streaming Process Mining and PM on IoT devices.

2. Background

2.1. XES Standard

XES standard 1849-2023[5] defines a format for storing event logs and event streams. A *log* contains records of executions of a business process. A log consists of *traces*, each corresponding to one case. Traces consist of atomic *events*. A log can also consist only of events without traces, such log is called a *stream*. Information about a log, a trace or an event is stored in *attributes*. XES standard allows attributes to be nested, however, leaves this feature optional for implementations. Attributes can be *elementary* (integer, string, date and time, etc.) or *composite* (lists). *Global attributes* describe attributes that are available for every event or trace in the log. *Classifiers* make events comparable to each other and are represented by an ordered list of attribute keys. XES supports *extensions* that allow to attach semantics to the described elements and defines a set of standard extensions.

2.2. JXES

JXES is a JSON format of event log adhering to the XES principles [4]. JXES uses XES meta-model and defines JSON serialization for each of its components, allowing to store all information an XES log can contain inside a JSON file. Java implementation exists in a form of ProM Plugin.

3. Implementation

Python-JXES defines a set of tools to convert PM4Py [6] event logs and event streams to JSON objects, store them as JXES (`.jxes`) files, as well as load them from the JXES files. It uses PM4Py's *legacy log object* for internal representation as it follows the XES structure more closely in comparison to the new tabular format. It must also be noted that tools for bidirectional conversion between the two formats (legacy and tabular) exist in PM4Py, thus it can be safely said that all PM4Py log representations are supported.

JSON offers a smaller set of possible data formats than XML, thus `time` and `ID` attribute types are represented as `strings` in JXES. Time is serialized and de-serialized according to ISO 8601[1] using a regular expression.

While this implementation strives to be strictly conforming to the XES standard as defined in Clause 8.1. and supports all features described in Section 2.1, some discrepancies exist. First, as the standard only defines XML serialization, conforming log instances are also defined only in terms of XML, making all other serialization formats technically non-conforming. Second,

[1]https://www.iso.org/standard/40874.html

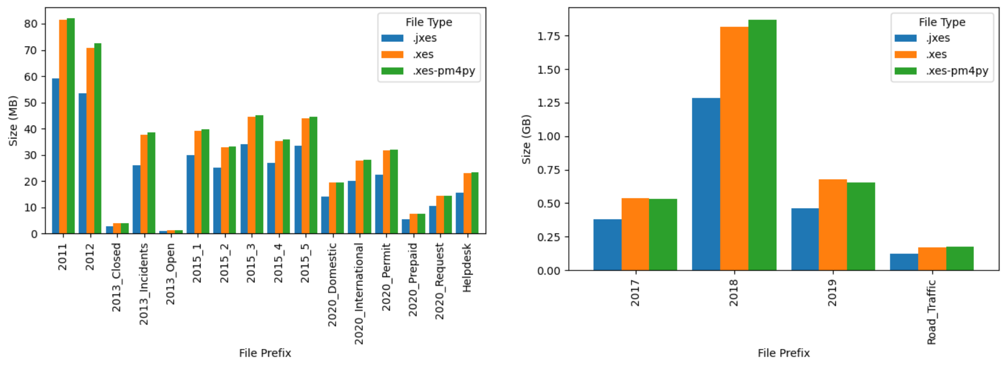
log attribute `xes.version` is assigned string value `"1849-2023"`, although it had to be of type `xs:decimal` (i.e., `float` in JSON) as per standard Clause 5.1.2. Finally, as the original JXES proposal included *containers* described in XES 2.0 standard [7], despite them being removed from the 1849-2023 version of the standard, Python-JXES still supports them. However, this feature can be made optional in future versions of the tool to ensure scrict conformance.

The implementation and basic examples are publicly available on PyPi[2] and GitHub[3].

4. Evaluation

In this section, the implementation will be evaluated. The file size in JXES will be compared to standard XML serialization. Read and write speeds of python-jxes will be compared to a state-of-the-art tool PM4Py [6], which is certified by XES Working Group[4].

The evaluation was performed the following way: first, an XES event log was read by PM4Py, 5 times for each importer variant[5] except `rustxes`, then, this log was serialized back to XES using PM4Py, also 5 times for each of the two exporter variants. Following that, the in-memory log was serialized as JXES for 5 times. Finally, the JXES log was imported for 5 times. In order to prevent caching, cache was flushed before each of the 5 iterations of every test. For every combination of file, tool and variant, the median of the 5 runs was used. All evaluation results can be found in a separate GitHub repository[6].



(a) XES logs smaller than 150 MB

(b) XES logs larger than 150 MB

Figure 1: File sizes of original XES logs, XES serialized by PM4Py and JXES logs.

4.1. Setup

All tests ran on a laptop with Intel®Core ™ i7-1260P CPU, 32 GB DDR4 RAM and Corsair MP600 CORE XT NVMe SSD, running Ubuntu 22.04, Python 3.11 and PM4Py version 2.7.11.13.

[2]https://pypi.org/project/jxes/
[3]https://github.com/MaxVidgof/python-jxes
[4]https://www.tf-pm.org/resources/xes-standard/for-vendors/tool-support
[5]https://github.com/pm4py/pm4py-core/tree/release/pm4py/objects/log/importer/xes/variants
[6]https://github.com/MaxVidgof/python-jxes-evaluation

Open-source event logs such as BPI Challenge 2011-2013, 2015, 2017-2019, as well as Italian helpdesk log and Road traffic fines log were user for evaluation.

4.2. Size

Figure 1 shows the file size of the original XES file, the JXES file and XES file exported by PM4Py (even if PM4Py was used to read an XES file and then to export it, it was slightly different from the original XES file). The figure had to be split into two subfigures in order to account for different file sizes of event logs.

Figure 2 shows the saved space per log if using JXES compared to PM4Py's XES serialization. On average, JXES file is 28% smaller than XES file. Minimum difference is 25.8% , and maximum difference reaches 33%.

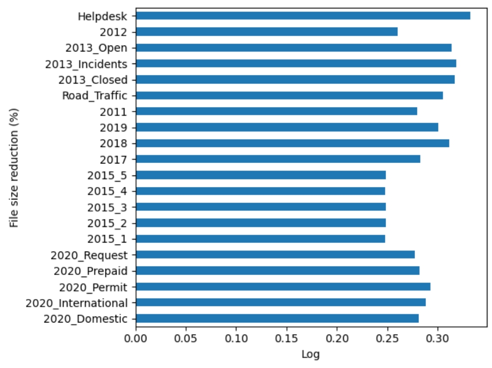


Figure 2: File size reduction of JXES versus XES.

4.3. Read speed

Figure 3 shows time required to parse BPI Challenge 2017 log. It shows time taken by various PM4Py XES importer variants to import the XES file as well as time taken by Python-JXES to read the corresponding JXES file. This log was chosen for demonstrative purposes because it has the most significant difference: JXES reduces the read time by more than 73%, however, the median performance gain against the best PM4Py variant is 52% and the minimum gain is 32%.

It is important to notice that some configurations of pm4py render empty logs without errors, so for BPIC 2017 and 2019 `line_by_line` parser have been removed; also log 2011 and 2012 could not be read with `chunk_regex` and were also excluded.

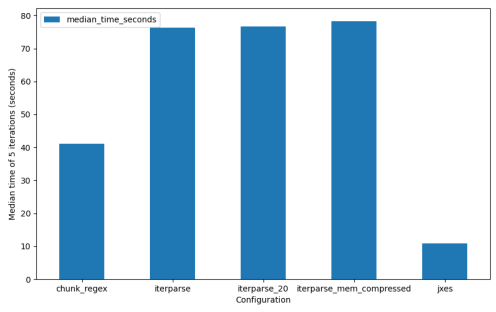


Figure 3: Time taken to read BPIC 2017 log.

4.4. Write speed

Write benchmarks show similar results. Serializing event logs in JXES is faster than the best PM4Py XES serialization by at least 28%. Median write performance improvement is 40%, and maximum improvement almost reaches 47%.

5. Discussion

Space savings by JXES are achieved because a lot less characters are needed. First, JSON removes the necessity to specify data types as opposed to XML. Second, there is no need for opening and closing tags in JSON. Because of this, an event `<event><string key="concept:name" value="example"/></event>` can be written as `{"concept:name":"example"}`, leading to significant space reduction. This is important for at least the following scenarios:

1. **Storing large amounts of historical data.** This might not seem critical for smaller event logs ranging between tens of kilobytes and tens of megabytes in size. However, for real-life logs reaching tens of gigabytes in size, this becomes relevant.
2. **Data transfer.** When logs have to be transferred, e.g., between the system collecting them and the system analyzing them, data size is also crucial. Also important is the case of Streaming Process Mining, where smaller sizes may also reduce latency because it will take less time to transmit an event completely.
3. **Streaming Process Mining.** In previous work [8], JXES was suggested as the event serialization format. This implementation now enables such approach.
4. **Constrained devices and IoT.** There are some approaches to perform Process Mining directly on IoT devices. The small devices will obviously profit from a more lightweight data format, which both takes less space to store and less time to import and export.

It must be noted at this stage that binary serialization formats can offer even more compression. For instance, pickle[7] requires even less space, e.g., 1.1 GB for pickle versus 1.3 GB for JXES versus 1.9 GB XES for BPIC 2018. However, in contrast to both XES and JXES, pickle is a binary format that is not readable for humans and only allows to share data between Python programs, leaving out a significant fraction of the PM ecosystem. Similarly, file compression can reduce the file sizes significantly, however, it might still be impractical for streaming or IoT scenarios as (de-)compressing requires additional time and computational resources. In addition, the compressed files also cannot be read without decompression. It is worth pointing out though, that compressed JXES files are still about 10% smaller than compressed XES files containing the same event log.

While JXES shows significant improvements in read and write speeds, it must be noted that, in comparison to XML, Python's default implementation of JSON does not support line-by-line reading, which could improve reading of large event logs or streams even further. However, non-standard implementations for incremental JSON parsing do exist, and exploring them is left for future work.

6. Conclusion

XES is a standard XML-based serialization format for event logs. As JSON-based formats are becoming increasingly adopted, JXES – a JSON serialization format for XES was proposed. This paper presents Python implementation of JXES and evaluates it against state-of-the-art XES

[7]https://docs.python.org/3/library/pickle.html

tools for open-source event logs, showcasing significant improvements in storage requirements and read and write performance.

6.1. Future work

Future work is aimed at evaluating JXES in scenarios where it can be most beneficial, such as Streaming Process Mining and PM of IoT devices. Also evaluation in regular PM scenarios would be beneficial. Finally, the implementation might receive new features such as incremental JSON parsing.

References

[1] W. M. van der Aalst, Process Mining: Data Science in Action, Second Edition, Springer, 2016.

[2] A. Berti, I. Koren, J. N. Adams, G. Park, B. Knopp, N. Graves, M. Rafiei, L. Liß, L. T. genannt Unterberg, Y. Zhang, C. T. Schwanen, M. Pegoraro, W. M. P. van der Aalst, OCEL (object-centric event log) 2.0 specification, CoRR abs/2403.01975 (2024). URL: https://doi.org/10.48550/arXiv.2403.01975. doi:10.48550/ARXIV.2403.01975. arXiv:2403.01975.

[3] D. Calegari, A. Delgado, A model-driven engineering perspective for the object-centric event data (OCED) metamodel, in: J. D. Weerdt, L. Pufahl (Eds.), Business Process Management Workshops - BPM 2023 International Workshops, Utrecht, The Netherlands, September 11-15, 2023, Revised Selected Papers, volume 492 of *Lecture Notes in Business Information Processing*, Springer, 2023, pp. 508–520. URL: https://doi.org/10.1007/978-3-031-50974-2_38. doi:10.1007/978-3-031-50974-2_38.

[4] M. B. S. Narayana, H. Khalifa, W. M. P. van der Aalst, JXES: JSON support for the XES event log standard, CoRR abs/2009.06363 (2020). URL: https://arxiv.org/abs/2009.06363. arXiv:2009.06363.

[5] Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams, IEEE Std 1849-2023 (Revision of IEEE Std 1849-2016) (2023) 1–55. doi:10.1109/IEEESTD.2023.10267858.

[6] A. Berti, S. J. van Zelst, D. Schuster, Pm4py: A process mining library for python, Softw. Impacts 17 (2023) 100556. URL: https://doi.org/10.1016/j.simpa.2023.100556. doi:10.1016/J.SIMPA.2023.100556.

[7] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, W. M. P. van der Aalst, Xes, xesame, and prom 6, in: P. Soffer, E. Proper (Eds.), Information Systems Evolution - CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers, volume 72 of *Lecture Notes in Business Information Processing*, Springer, 2010, pp. 60–75. URL: https://doi.org/10.1007/978-3-642-17722-4_5. doi:10.1007/978-3-642-17722-4_5.

[8] M. Vidgof, Towards process mining on kafka event streams (short paper), in: S. Böhm, D. Lübke (Eds.), Proceedings of the 16th ZEUS Workshop, Ulm, Germany, February 29-March 1, 2024, volume 3673 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 1–8. URL: https://ceur-ws.org/Vol-3673/paper1.pdf.