
*Production Systems and Information Engineering*
Volume 10 (2), pp. 1–10
doi: 10.32968/psaie.2022.2.1.



# CONVERSION OF CUSTOMER SERVICE EVENT LOGS TO STANDARD FORMATS

Erika Baksáné Varga
University of Miskolc, Hungary
Department of Information Engineering
vargaerika@iit.uni-miskolc.hu

**Abstract.** The act of logging the events (transactions, errors, intrusions, etc.) happening within an information system is about the same age as the system itself. Mining these historical records, however, is a recent demand to support robotic process automation initiatives. Our goal is to create an RPA solution for heavily overloaded customer services and we now face the problem of getting logs with different syntax and structure. This paper presents the standard event log formats and reviews the steps of transforming the most frequent non-standard log formats into a uniform formalism.

*Keywords*: event log formats, XES standard, OCEL standard

## 1. Introduction

During the execution of processes in today's information systems, tremendous amount of historical traces are recorded. These records may have many forms, such as database entries, audit trails, or system logs. These historical data are utilized by process mining to uncover useful information and gain insights into business process performance [1]. A recent research shows that process mining has numerous applications, exploiting event logs as the fundamental data source [2]. In order to guarantee reliable results, process mining should start with high-quality event logs [3]. In practice, however, event logs are derived from heterogeneous sources and because of data recording or transformation errors, they are often far from the desired quality [4], [5]. For this reason, a lot of manual effort is spent on identifying and correcting data quality problems. [1] lists the most important aspects that should be taken into consideration related to event log quality.

- Ensuring correlation and consistency of events derived from different information systems.

- Chronology of events. When loading data from distinct systems, events are ordered according to the timestamp attribute. However, the actual time of occurrence of a particular event and the timestamp recorded in the event log may differ significantly. As a result, the sequence of events will be unreliable.
- Completeness of logged traces.
- Validity of data.
- The level of granularity should be identical for all traces.

There are proposals for qualitative models which aim to indicate the quality of event logs before applying process mining algorithms [6], [7].

Beside data quality problems, ambiguities in formats also hamper the preprocessing of event logs. Logs can take plenty of different forms and instantiations. Every system architecture that includes some sort of logging mechanism has so far developed their own solution for this task. The first approach to standardizing event logs was MXML (Mining eXtensible Markup Language), which stored timestamps, resources, and transactions in a unified format, while other ad-hoc extensions could be added to it as needed. Following that, the first XES (Extensible Event Stream) standard was introduced in 2010 [8]. This standard was revised and officially published in 2016 as the 1849-2016 IEEE Standard for eXtensible Event Stream for Achieving Interoperability in Event Logs and Event Streams [9]. This formalism, however, has problems when dealing with object-centric data (e.g. database tables) due to the existence of one-to-many and many-to-many relations. [10] proposes a new approach to extract, transform and store object-centric data, resulting in eXtensible Object-Centric (XOC) event logs. Based on this concept, the Object-Centric Event Log (OCEL) Standard was released in 2021 [11], which opens the way for object-centric process mining. It complements the XES standard while providing a more comprehesive view of the process, i.e., different types of objects can be stored without enforcing a single viewpoint. Based on logs written in this format, process mining techniques can create object-centric process models and reveal the evolutionary states of a database.

## 2. Event log standards

### 2.1. XES

XES is the standard format for storing event logs. Its purpose is to provide a generally-acknowledged format for the interchange of event log data between tools and application domains. The XES standard defines an XML-based language for storing, transmitting, and processing event data. The standard contains two XML Schema descriptions: one defines the structure of XES event

logs, and the other defines the structure of extensions to event logs [9]. The general structure of an event log is:

```
<log  xes.version="1.0"  xes.features="..."
          xmlns="http://www.xes-standard.org/">
 <trace>
     <event>
         ...
     </event>
 </trace>
</log>
```

The event log is used to store completed process instances, called traces, within which any number of events can occur. The log, trace, and event tags do not carry information, they only define the structure of the event log. Information is stored in key-value pairs by their attributes. There are six elementary attribute types defined by the data type they represent (String, Date, Int, Float, Boolean, and ID); and two composite attribute types: ordered lists (list) and sets (container). However, complex attribute types can only be used if the "nested attributes" feature is set at the beginning of the event log (<log xes.version = "1.0" xes.features = "nested-attributes">) and the applied XES parser supports this feature. The identifiers and semantics of the attributes of the structure tags are not predefined. As a result, their meaning can be ambiguous. This shortcoming can be overcome by using extensions. The first XES standard defines seven extensions that are specified in XESEXT XML format: concept, cost, id, lifecycle, organizational, semantic, and time. The newer IEEE 1849-2016 standard defines five additional extensions: artifactlifecycle, micro, swcomm, swevent, and swtelemetry.

### 2.2. OCEL

XES represents a process-centric approach to describing event logs, i.e. the central element is the trace, and all the activities (events) recorded sequentially in the event log are connected to a process instance. In information systems, however, processes are running in parallel and may refer to identical objects. Following the process-centric model, events are logged together with their related objects which may lead to duplicated data storage. On the other hand, in object-centric modeling, a single copy of each object is stored, and events save only references to these objects.

OCEL is the standard language of object-centric event logs [11]. It is very similar in syntax to the XES format. The main differences between the two languages are the following.

- OCEL is not process-centric, so it does not use the trace tag.
- The central tag in OCEL is object. Every artifact of a process is defined as an object and events contain only references to the related objects.

An OCEL log starts with the enumeration of the attributes stored in connection with the log itself, with events and with objects in the corresponding <global

scope="..."> tag. Then comes the description of each object within the <objects> <object> ... </object> </objects> tags. Similarly, each event is specified within the <events> <event> ... </event> </events> tags. Each event has three types of attributes:

- mandatory attributes, like activity and timestamp,
- a set of object references (zero or more per object types), and
- additional attributes.

## 3. Conversion of text logs to XES

### 3.1. Transformation of TXT and CSV logs

XES is a process-centric description, where the central tag is the trace. In a text or CSV log, each row represents a separate event, and the events are recorded in order of occurrence.

When transforming a text log into the standard XES format, the first and most important task is to connect each recorded event to an XES trace. Basically, two cases may happen. Either, each row in a text log contains a process ID indicating that the event belongs to the given process instance. In this case the assignment of the event to a trace is straightforward. On the other hand, if this information is missing from the text log, one should check the logging strategy to determine whether each row of the log is for a separate process instance, or all rows of the log belong to the same trace.

The next step is to group the recorded attribute values under XES traces and events. Those attributes, that have the same value for all events of a given process instance are mapped to trace attributes. While the rest correspond to event attributes.

In an XES log, attributes are given by key=value pairs and the data types of the attributes are also stored. The list of keys can be derived from the header of the text log. If there is no header, generated keys are applied. Data type information is hard to extract from the text log, so the most general string type can be used. Fig.1 shows the first four rows in a CSV event log and the corresponding XES description of the first event.

### 3.2. Conversion of JSON logs

JSON (JavaScript Object Notation) is a semi-structured data format language, which has become the main data exchange format over the World Wide Web in recent years, and gained popularity in database research. It is actually a string whose format very much resembles the JavaScript object literal format [12]. Its structure is simple and contains a comma separated list of key : value pairs within curly brackets. A value can be atomic, or set of values within square brackets, or nested list of key : value pairs within curly brackets. These value types can be arbitrarily combined and embedded, thereby yielding quite complex data structures.

| Case-id | Request-id | Description | Priority | Source | Activity-id | Timestamp | State | Type | Category | Input | Agent-id | Group | Role | Host |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 111 | 222 | invoice complaint | urgent | 333 | 441 | 2021-11-07T02:26:35.000+01:00 | completed | register request | invoicing | | 555 | customer service | clerk | 193.4.5.6 |
| 111 | 222 | invoice complaint | urgent | 333 | 442 | 2021-11-08T02:26:35.000+01:00 | completed | examine request | invoicing | invoice data | 666 | accounting | clerk | 194.6.5.8 |
| 111 | 222 | invoice complaint | urgent | 333 | 443 | 2021-11-10T02:26:35.000+01:00 | completed | decide | invoicing | decision | 777 | accounting | administrator | 195.4.5.6 |
| 111 | 222 | invoice complaint | urgent | 333 | 444 | 2021-11-11T02:26:35.000+01:00 | completed | reject request | invoicing | | 555 | customer service | clerk | 193.4.5.6 |

**(a)** Sample CSV log with four events

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <extension name="Organizational" prefix="org" uri="http://code.fluxicon.com/xes/org.xesext" />
    <string key="concept:name" value="XES Event Log"/>
    <trace>
        <string key="concept:name" value="111"/>
        <string key="request" value="222">
            <string key="description" value="invoice complaint" />
            <string key="priority" value="urgent" />
            <string key="source" value="333" />
        </string>
        <event>
            <string key="concept:name" value="441"/>
            <date key="time:timestamp" value="2009-11-25T12:45:32.345+02:00" />
            <string key="lifecycle:transition" value="completed"/>
            <string key="type" value="register request" />
            <string key="category" value="invoicing" />
            <string key="input" value="" />
            <string key="agent" value="555">
                <string key="org:group" value="customer service" />
                <string key="org:role" value="clerk" />
                <string key="host" value="193.4.5.6" />
            </string>
        </event>
    </trace>
</log>
```

**(b)** XES description of the first event

**Figure 1.** Conversion of plain text event log to XES

When JSON is applied as log format, its recommended to use the JSON Lines variant (https://jsonlines.org/). It essentially consists of several lines where each individual line is a valid JSON object, separated by newline character. The JSONL correspondent of the first sample event is given in Fig.2.

```
{"Activity-category":"invoicing","Activity-id":441,"Activity-input":"","Activity-
state":"completed","Activity-time":"2021-11-07T02:26:35.000+01:00","Activity-type":"register
request","Agent-group":"customer service","Agent-host":"193.4.5.6","Agent-id":555,"Agent-
role":"clerk","Case-id":111,"Request-desc":"invoice complaint","Request-id":222,"Request-
priority":"urgent","Request-source":333}
```

**Figure 2.** JSONL description of the first sample event

In the conversion of JSON Lines logs to XES, the first problem is the binding of events to traces. The solution is based on the same method as in the case of CSV

files. That is, the decision can be made either based on the recorded data or the logging strategy. The mapping of key : value pairs is straightforward in the case of plain structures. If a value is actually a set of values, XES nested attribute types must be applied. For nested structures, new abstract attributes must be introduced as XES event attributes.

### 3.3. Transformation of EVTX and XML logs

The Windows operating system generates event logs for five different categories, including Application, Security, Setup, System, and Forwarded Events. These are saved in a proprietary binary format (EVT, EVTX) that can only be viewed within the Event Viewer program. An EVTX log [13] can be converted to XES format in two steps. In the first step, it is transformed into XML using EvtxParser. The structure of the resulting XML log looks as follows.

```
<Events>
   <Event>
     <System> ... </System>
     <EventData> ... </EventData>
   </Event>
   <Event>
     ...
   </Event>
</Events>
```

The description of each event starts with the System tag. Basic information such as the location of the event (the computer's IP address), the event timestamp, or the event ID is automatically stored here. Additionally, one of the following tags is included within the event tag: BinaryEventData, DebugData, EventData, ProcessingErrorData, RenderingInfo, or UserData. The most common descriptor is EventData, which contains the parameters passed by the application that triggered the event.

The conversion of this XML format to XES is quite straightforward. The only question is how to match events with traces. This assignment can either be determined based on the stored data, or the logging strategy like in the case of CSV logs.

## 4. Conversion of text logs to OCEL

In customer service systems several case notions such as request and agent are involved and interact with each other. Therefore, creating an event log where each event is assigned to a single case (i.e. process instance) leads to

- convergence, i.e. the duplication of an event related to different cases, and
- divergence, i.e. the inability to separate events within the same case

problems at the same time [11]. To avoid such problems, the use of object-centric event logs are recommended.

Let us take the example CSV log shown in Fig.1a. The data stored in connection with customer service processes are: case id; customer request id, description, priority and source; activity id, timestamp, state, type, category and input value; and agent id, group, role and host. Following the object-centric paradigm, the first step is to specify the list of event related attributes:

```
<global scope="event">
   <string key="case-id" value="__INVALID__"/>
   <string key="activity-id" value="__INVALID__"/>
   <string key="timestamp" value="__INVALID__">
   ...
   <string key="omap" value="__INVALID__"/>
</global>
```

Note here, that OCEL eliminates traces, so process instance information should be included among event attributes. The last attribute is the list of the referred objects. Each object should have at least two attributes:

```
<global scope="object">
   <string key="id" value="__INVALID__"/>
   <string key="type" value="__INVALID__"/>
</global>
```

All other attributes can be given as log attributes, together with the list of object types:

```
<global scope="log">
   <string key="version" value="0.1" />
   <string key="ordering" value="timestamp" />
   <list key="attribute-names">
      <string key="name" value="request-priority"/>
      ...
   </list>
   <list key="object-types">
      <string key="type" value="request"/>
      <string key="type" value="agent"/>
   </list>
</global>
```

After this preamble, the OCEL log enumerates the objects and the events. In our example, the first event can be given as:

```
<events>
  <event>
   <string key="case-id" value="111"/>
   <string key="activity-id" value="441"/>
   <string key="timestamp" value="2020-07-09T08:20:01.527+01:00"/>
   <string key="state" value="completed"/>
```

```
    <string key="type" value="register request"/>
    <string key="category" value="invoicing"/>
    <list key="omap">
      <string key="object-id" value="222"/>
      <string key="object-id" value="555"/>
    </list>
   </event>
 </events>
```

Here, object 222 is a request and object 555 is an agent, defined as:

```
<objects>
  <object>
    <string key="id" value="222"/>
    <string key="type" value="invoice complaint"/>
    <list key="ovmap">
      <string key="request-source" value="333" />
      <string key="request-priority" value="urgent" />
    </list>
  </object>
  <object>
    <string key="id" value="555"/>
    <string key="type" value="clerk"/>
    <list key="ovmap">
      <string key="agent-group" value="customer service" />
      <string key="agent-host" value="193.4.5.6" />
    </list>
  </object>
</objects>
```

## 5. Summary

Our goal is to develop a neural network based RPA solution for heavily overloaded customer services. As a first step, we need to provide reliable input for training the neural network that will realize various process mining tasks. The input for process mining is the data stored in event logs which are automatically recorded by information systems. An event log can be seen as a collection of cases and a case can be seen as a sequence of events (called trace). Since event data may come from a wide variety of sources, the format and structure of event logs are quite diverse. To handle this problem, the XES process-centric and the OCEL object-centric standard formats have been developed to be utilized in process mining applications. The native format of event logs, however, is seldom standard. Often CSV files are used as an intermediate format, and process mining tools convert these files into event logs by assigning columns to process mining concepts.

Related to our goal, we are working on a conversion modul that is able to transform event logs originating from customer service systems to standard XES and OCEL descriptions [14]. This paper has presented the standard formats and the steps of the conversion, and showed that the transformation is technically feasible.

**Acknowledgement.** The described article was carried out as part of the 2020-1.1.2-PIACI-KFI-2020-00165 "ERPA - Development of Robotic Process Automation solution for heavily overloaded customer services" project implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2020–1.1.2-PIACI KFI funding scheme.

## References

[1] van der Aalst, W. M. P.: *Process Mining: Data Science in Action*. Springer, 2016, URL https://doi.org/10.1007/978-3-662-49851-4.

[2] Emamjome, F., Andrews, R., and ter Hofstede, A. H. M.: A case study lens on process mining in practice. In H. Panetto, C. Debruyne, M. Hepp, D. Lewis, C. A. Ardagna, and R. Meersman (eds.), *On the Move to Meaningful Internet Systems: OTM 2019 Conferences*, Springer International Publishing, Cham, ISBN 978-3-030-33246-4, 2019, pp. 127–145, URL https://doi.org/10.1007/978-3-030-33246-4_8.

[3] Andrews, R., van Dun, C., Wynn, M., Kratsch, W., Röglinger, M., and ter Hofstede, A.: Quality-informed semi-automated event log generation for process mining. *Decision Support Systems*, **132**, (2020), 113265, URL https://doi.org/10.1016/j.dss.2020.113265.

[4] Suriadi, S., Andrews, R., ter Hofstede, A., and Wynn, M.: Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information Systems*, **64**, (2017), 132–150, URL https://doi.org/10.1016/j.is.2016.07.011.

[5] Fischer, D. A., Goel, K., Andrews, R., van Dun, C. G. J., Wynn, M. T., and Röglinger, M.: Enhancing event log quality: Detecting and quantifying timestamp imperfections. In D. Fahland, C. Ghidini, J. Becker, and M. Dumas (eds.), *Business Process Management*, Springer International Publishing, Cham, ISBN 978-3-030-58666-9, 2020, pp. 309–326, URL https://doi.org/10.1007/978-3-030-58666-9_18.

[6] Kherbouche, M. O., Laga, N., and Masse, P.-A.: Towards a better assessment of event logs quality. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8, URL https://doi.org/10.1109/SSCI.2016.7849946.

[7] Wynn, M. T. and Sadiq, S.: Responsible process mining - a data quality perspective. In T. Hildebrandt, B. F. van Dongen, M. Röglinger, and J. Mendling (eds.), *Business Process Management*, Springer International Publishing, Cham,

ISBN 978-3-030-26619-6, 2019, pp. 10–15, URL https://doi.org/10.1007/978-3-030-26619-6_2.

[8] Günther, C. W. and Verbeek, E.: XES standard definition. URL https://xes-standard.org/_media/xes/xesstandarddefinition-2.0.pdf.

[9] IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pp. 1–50.

[10] Li, G., de Murillas, E. G. L., de Carvalho, R. M., and van der Aalst, W. M.: Extracting object-centric event logs to support process mining on databases. In *CAiSE Forum*, 2018.

[11] Ghahfarokhi, A., Park, G., Berti, A., and Aalst, W.: OCEL: A standard for object-centric event logs. ISBN 978-3-030-85081-4, 2021, pp. 169–175.

[12] Lv, T., Yan, P., and He, W.: Survey on JSON data modelling. *Journal of Physics: Conference Series*, **1069**, (2018), 012101, URL https://doi.org/10.1088/1742-6596/1069/1/012101.

[13] Schuster, A.: Introducing the Microsoft Vista event log file format. *Digital Investigation*, **4**, (2007), 65–72, URL https://www.sciencedirect.com/science/article/pii/S1742287607000424.

[14] Mileff, P. and Dudra, J.: Activity logs in practice. *Production Systems and Information Engineering*, **10**, (2022), 11–26.