IEEE Task Force on
Process Mining

XES CERTIFICATION FOR
THE ETHEREUM
LOGGING FRAMEWORK
# TABLE OF CONTENTS

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>1</td>
</tr>
<tr>
<td>Meta</td>
<td>2</td>
</tr>
<tr>
<td>Export</td>
<td>3</td>
</tr>
<tr>
<td>Appendix A: Proof for CryptoKitties Data</td>
<td>37</td>
</tr>
<tr>
<td>Contact Information</td>
<td>62</td>
</tr>
</tbody>
</table>
Tool

NAME
Ethereum Logging Framework

VENDOR
CSIRO Data61

VERSION
0.2.1

REQUESTED CERTIFICATION LEVELS

1 https://github.com/ChrisKlinkmueller/Ethereum-Logging-Framework
Meta

AUTHORS
Christopher Klinkmüller, CSIRO Data61, christopher.klinkmueller@data61.csiro.au

DATE
7/20/2021

HISTORY

CHANGES

<table>
<thead>
<tr>
<th>AUTHOR(S)</th>
<th>DATE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Christopher Klinkmüller</td>
<td>2/7/2021</td>
<td>Creation of the report</td>
</tr>
<tr>
<td>Christopher Klinkmüller</td>
<td>20/7/2021</td>
<td>The ELF validator now enforces that all defined global event attributes were added to all XES emission statements</td>
</tr>
</tbody>
</table>
Export

The **Ethereum Logging Framework** (ELF) enables users to export data from Ethereum\(^2\), a blockchain technology for executing decentralized applications. Due to Ethereum's smart contract capabilities, developers can deploy and execute arbitrary applications with custom data schemas and logic on Ethereum networks. To obtain data from those applications in a specific analysis context, users can flexibly configure the data export process via ELF's **Ethereum Querying Language** (EthQL). Besides configurations of connections and output folders, an EthQL script specifies (i) which data must be extracted, (ii) how it must be transformed, and (iii) how it must be formatted. While ELF offers a variety of operators for all three steps, this report solely focuses on capabilities related to the export of XES files that comply with the XES certification levels A-X. More details about ELF's capabilities are presented in various publications\(^3,4,5\).

As the users have full control over the data export process, the XES certification levels that an exported log adheres to are not predetermined but depend on the user's specific information needs. In this regard, two ELF components ensure that the exported logs comply with the intended XES certification levels. First, given an EthQL script the *validator* determines the applicable certification levels, identifies issues in the script, and provides users with a list of errors and warnings regarding violations of the XES certification levels. Second, the *extractor* only executes valid scripts and, where applicable, automatically inserts elements to ensure that the exported logs conform to the identified XES certification levels.

This report outlines how the two ELF components implement the different certification levels and to this end uses the **CryptoKitties** application for illustration purposes. CryptoKitties is a popular game in which virtual cats can be bred and traded. Since its deployment on the Ethereum mainnet on 23 November 2017\(^6\) it has been used extensively, resulting in more than 18,000,000 log entries or events, respectively. The examples in this report are based on a small subset of these log entries. This subset stems from the block range [6000000,6000024] and as shown in Table 1 and Table 2 comprises four log entries related to the birth and eight log entries related to the transfer of CryptoKitties. Evidence for the existence of the log entries is provided in Appendix A in the form of screenshots from Etherscan which is "[...] a Block Explorer and Analytics Platform for Ethereum [...]"\(^7\).

---

\(^{2}\)https://ethereum.org/en/


\(^{6}\)https://etherscan.io/tx/0x691f348ef11e9ef95d540a2da2c5f38e36072619aa44db0827e1b8a276f120f

\(^{7}\)https://etherscan.io
Table 1: The birth log entries created by CryptoKitties in the block range [6000000,6000024]

<table>
<thead>
<tr>
<th>block number</th>
<th>owner</th>
<th>kittyId</th>
<th>matronId</th>
<th>sireId</th>
<th>genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000000</td>
<td>0x7891f796a5d434666c291</td>
<td>851836</td>
<td>73402</td>
<td>843147</td>
<td>6.8377203809982671806013803274605534827931635</td>
</tr>
<tr>
<td></td>
<td>0x0669092ae497a290</td>
<td></td>
<td></td>
<td></td>
<td>46714644889349673255907399e+71</td>
</tr>
<tr>
<td>6000001</td>
<td>0x9d2ac7c3e17163f104e6a</td>
<td>851837</td>
<td>851455</td>
<td>848263</td>
<td>3.45324947766212555662460961028125531409057</td>
</tr>
<tr>
<td></td>
<td>b5f374502bf9d1b02</td>
<td></td>
<td></td>
<td></td>
<td>5531153457525594575740035e+71</td>
</tr>
<tr>
<td>6000021</td>
<td>0xdbed6357ea19cad45a316</td>
<td>851838</td>
<td>564479</td>
<td>733495</td>
<td>4.5332925374533338277971748213499515169551482484</td>
</tr>
<tr>
<td></td>
<td>33562436c6132f102</td>
<td></td>
<td></td>
<td></td>
<td>4168391838535397842994497619e+71</td>
</tr>
<tr>
<td>6000021</td>
<td>0x837ed29d5c4ab6666c550b</td>
<td>851839</td>
<td>851652</td>
<td>851664</td>
<td>4.4911491691208691725515255415540923111554874</td>
</tr>
<tr>
<td></td>
<td>7218d25dc028ef6689</td>
<td></td>
<td></td>
<td></td>
<td>71980646449142346398314529e+71</td>
</tr>
</tbody>
</table>

The Birth log entries contain five attributes:

- **owner** – the account address of the initial owner;
- **kittyId** – the identifier of the CryptoKitty;
- **matronId** – the identifier of the CryptoKitty's mother;
- **sireId** – the identifier of the CryptoKitty's father; and
- **genes** – the integer representation of the CryptoKitty’s DNA.

Similarly, the Transfer log entries comprise three attributes:

- **from** – the account address of the original owner;
- **to** – the account address of the new owner; and
- **tokenId** – the identifier of the CryptoKitty.
Figure 1: EthQL template script for exporting CryptoKitties log entries

All CryptoKitties examples in this report extract these log entries from the Ethereum mainnet, but they vary with respect to the exported attributes. To this end, all examples use scripts that follow the EthQL script template, which is shown in Figure 1. The preamble (lines 1 to 4) defines the connection to the Ethereum mainnet node from which the data is extracted, and the output folder into which the data is exported. After that, the export process is specified (lines 5 to 24). First, the BLOCKS filter (line 6) narrows down the range of blocks from which data is extracted. Within the scope of this filter, there are two filters for LOG ENTRIES (lines 7 to 14 and lines 16 to 23). These filters specify from which smart contract the log entries should be extracted. A smart contract is a component of an application deployed on Ethereum. The script uses the address of the main smart contract of the CryptoKitties application. Each filter also defines the signature of the log entry for which data must be extracted. Here, the signatures of the birth and transfer log entries are used. Within the scope of each LOG ENTRIES filter there is one EMIT XES EVENT statement. The first three configuration options of this statement (in round brackets) define how events are sorted into the log hierarchy. Here, all events are added to the “CryptoKitties” log, i.e., one XES file ‘CryptoKitties.xes’ containing all event information will be written into the output folder. Moreover, the identifier of the CryptoKitties is used as the trace identifier, i.e., the process notion corresponds to the lifecycle of a single CryptoKitty. We do not specify an event identifier, meaning that each time an emission statement is executed, a new XES event is created and added to the respective trace. The last parameter of the EMIT XES EVENT statements is a list of attributes that must be exported. This list can be configured by the user and ultimately determines the compliance to the XES certification levels. It is important to note that ELF iterates through block ranges, transactions, and log entries in the order in which they were included in Ethereum’s blockchain structure. Thus, the data that ELF exports preserves the order in which it was created. Moreover, in addition to application-specific data, log entries, transactions, and blocks have standard attributes. For example, the block number, transaction index, and log index are identifiers for these elements. Following, the support for the different certification levels is outlined.
Level A1

Figure 2 shows a script that exports the CryptoKitties log entries in compliance with certification level A1. The script only defines the concept:name attribute for events and its value ("Birth" or "Transfer") depends on the type of the log entry (lines 14 and 23). The definition of XES attributes follows the pattern "<value> as <xml type> <attribute name>". While in Figure 2 the value is defined statically, attribute values can also be determined dynamically at runtime. This feature will be shown later, e.g., when exporting a variable or a computation result. ELF supports primitive xml types including strings, boolean values, dates, and integer and floating numbers. Attribute names can be freely chosen. Lastly, the script defines concept:name as a global event attribute with the default value “default activity” (line 5).

```cpp
// Preamble
connectIPC("/data2/geth-archive/chaindata/geth.ipc");
setOutputFolder("./output");
addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");

// Export definition

LOG ENTRIES

{ (x32108ac5c9f7bea050d6c2ae2378079f587f8e7a256d) (
  birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
  { EMIT XES EVENT ("A1_i") (kittyId)();
    "Birth" as xs:string concept:name
  };
}

LOG ENTRIES

{ (x32108ac5c9f7bea050d6c2ae2378079f587f8e7a256d) (transfer(address from, address to, uint256 tokenId))
  { EMIT XES EVENT ("A1_i") (tokenId)();
    "Transfer" as xs:string concept:name
  };
}
```

Figure 2: EthQL script for exporting A1 compliant XES logs
Executing the script results in the XES file in Figure 3. ELF automatically recognizes the use of standard extension attributes and adds the respective extension. Here, it added the Concept extension to the file (line 9). Following the specification from the script, concept:name is defined as a global event attribute (lines 10 to 13). Lastly, for each CryptoKitties log entry from the Ethereum mainnet the file contains an event with the concept:name attribute being set to “Birth” or “Transfer”, respectively (e.g., lines 14 to 16 and 17 to 19).

Figure 3: XES log extracted with the script from Figure 2, only top part shown, entire file available online

https://www.dropbox.com/s/jekcjrrxpkt85xx/A1_1.xes?dl=0
By default, ELF treats the `concept:name` attribute as a global event attribute which, if not explicitly declared, is automatically defined as such in the exported log. For example, removing the definition of `concept:name` as a global event attribute, see Figure 4, is recognized by the validator which issues a warning as shown in Figure 5. However, when using the modified script to extract data, the global value for `concept:name` is now set to “No global value for concept:name defined” in the resulting XES file (line 11 in Figure 6). The file still includes the Concept extension (line 9) and the events only contain the `concept:name` attribute (e.g., lines 14 to 16 and 17 to 19).

![Figure 4: The modified version of the script from Figure 2 without the global attribute definition](https://www.dropbox.com/s/2dcpvkl6axdehy9/A1_2.xes?dl=0)

![Figure 5: Validation result of the script from Figure 4](https://www.dropbox.com/s/2dcpvkl6axdehy9/A1_2.xes?dl=0)

![Figure 6: XES log extracted with the script from Figure 4, only top part shown, entire file available online](https://www.dropbox.com/s/2dcpvkl6axdehy9/A1_2.xes?dl=0)

---

The concept:name attribute is the only mandatory XES attribute whose use is enforced by ELF. Users must add it to all XES event emission statements, otherwise the EthQL script is invalid and ELF will not execute it. For instance, in the script in Figure 7 the concept prefix was not added to the attribute name in the context of the Birth event emission (line 14). The validation of this script results in an error message which indicates that the concept:name attribute was not set, see Figure 8.

![Figure 7: The EthQL script is invalid, as it does not specify concept:name for XES events](image)

![Figure 8: Validation result for the script from Figure 7](image)
ELF also validates that the type that is specified in the script for the `concept:name` attribute is `xs:string`. This kind of type checking is supported for all attributes that are defined in known XES extensions. Currently, ELF only supports the XES standard extensions. While the explicit definition of the XES type is superfluous, it is currently required. To reduce the manual specification effort, inference of types of exported XES attributes from standard extension definitions and EthQL variable types will be added in the future.

To illustrate the type checking, consider the modifications to the script from Figure 2 that are presented in Figure 9 and Figure 10. Here, the user tries to export integer values for the `concept:name` attribute in the context of an event emission statement and in the context of a global event attribute definition, respectively. As this violates the type definition for `concept:name` from the Concept standard extension, ELF’s validator in both cases issues error messages that point the user to the problem, see Figure 11 and Figure 12.

Figure 9: The modification to the script from Figure 2 tries to emit an integer value for `concept:name`

Figure 10: The modification to the script from Figure 2 tries to define a global integer value for `concept:name`

Figure 11: Validation result for the script from Figure 9

Figure 12: Validation result for the script from Figure 10
Level A2

Users can add event classifiers that rely on the `concept:name` attribute. To this end, users need to specify the global classifier in the preamble as shown in Figure 13 (line 7). Exporting CryptoKitties data based on the modified script results in the XES file in Figure 14. In addition to the traces, the events, the global attribute definition and the Concept extension, the file now also includes a classifier according to the user's specification (line 13).

```
1 // Preamble
2 connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3 setOutputFolder("./output");
4 addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
5 addXesEventClassifier("Event Name", "{concept:name}"");
6 // Export definition
7 BLOCKS (6000000) (6000024) {1
```

Figure 13: Modifying the script from Figure 2 to include an event classifier

```
1 <log version="1.0" encoding="UTF-8">
2 <!-- This file has been generated with the OpenXES library. It conforms -->
3 <!-- to the XML serialization of the XES standard for log storage and -->
4 <!-- management. -->
5 <!-- XES standard version: 1.0 -->
6 <!-- OpenXES library version: 1.0RC7 -->
7 <!-- OpenXES is available from http://www.openxes.org/ -->
8 </log>
9 <extension name="Concept" prefix="concept" url="http://www.xes-standard.org/concept.xesext"/>
10 <global scope="event">
11 <string key="concept:name" value="default activity"/>
12 </global>
13 <classifier name="Event Name" keys="concept:name"/>
14 <trace>
15 <event>
16 <string key="concept:name" value="Birth"/>
17 </event>
18 <event>
19 <string key="concept:name" value="Transfer"/>
20 </event>
21 </trace>
```

Figure 14: XES log extracted with the script from Figure 13, only top part shown, entire file available online

10 [https://www.dropbox.com/s/73nkdo6nbudafe/A2_1.xes?dl=0](https://www.dropbox.com/s/73nkdo6nbudafe/A2_1.xes?dl=0)
Level B1

Users can optionally specify lifecycle:transition and time:timestamp attributes for events. For example, the script in Figure 15 adds both attributes to the EMIT XES EVENT statements (lines 15 to 19 and 26 to 30). In more detail, as log entries created by the CryptoKitties application are created when a transaction was successfully executed, the lifecycle:transition attribute is set to “Completed”. Moreover, the time:timestamp attribute is mapped to the timestamp of the block that included the log entry. Additionally, both attributes are defined as global event attributes (lines 6 to 7). Note that Ethereum timestamps are represented as the number of seconds that have passed since the beginning of the epoch. ELF adopts this convention and does not provide a data type for timestamps. However, when emitting XES events users can cast integer values into xs:date, resulting in the emission of ISO conform dates.

Figure 15: Adding lifecycle:transition and time:timestamp attributes to the script from Figure 2
The export result is shown in Figure 16. In addition to the Concept extension, the XES file now includes the Time and Lifecycle extension (lines 9 to 10). Moreover, the file contains default values for the lifecycle:transition and time:timestamp attributes (line 14 and 15) and each event also comprises the two attributes with the respective values (lines 19 to 23 and 24 to 28). Note that ELF assumes that users comply with the BPAF lifecycle transactional model, when using the lifecycle:transition attribute. Hence, the lifecycle:model attribute for the log is set to “bpafl” (line 17), whenever the lifecycle:transition attribute is used. Note that attribute values might be created dynamically during the execution of EthQL scripts. Hence, the adherence to the BPAF model cannot be validated at compile time and it is the user’s responsibility to ensure adherence to this model.

![XES log example]

Figure 16: XES log extracted with the script from Figure 13, only top part shown, entire file available online

---

11 [https://www.dropbox.com/s/v36j1yj0q19b8bz/B1_1.xes?dl=0](https://www.dropbox.com/s/v36j1yj0q19b8bz/B1_1.xes?dl=0)
Similar to the concept:name attribute, ELF treats the lifecycle:transition and time:timestamp attributes as global event attributes. Hence, users do not need to explicitly specify the global values for the attributes as shown in Figure 17.

Figure 17: Removing the global values for lifecycle:transition and time:timestamp from the script in Figure 15
In this case, ELF automatically defines these attributes as global event attributes. This is demonstrated by the XES file in Figure 18 that was exported with the modified script from Figure 17 and that contains global event attribute definitions for the two attributes (lines 14 to 15). Moreover, Figure 19 shows that the validator issues warnings, if the global values for the two attributes were not set in the script.

```
xml version="1.0" encoding="UTF-8">
<log xmlns="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
  <extension name="Time" prefix="time" url="http://www.xes-standard.org/time.xesext"/>
  <extension name="Lifecycle" prefix="lifecycle" url="http://www.xes-standard.org/lifecycle.xesext"/>
  <global scope="event">
    <string key="concept:name" value="default activity"/>
    <date key="time:timestamp" value="1970-01-01T00:00:00:00:00"/>
    <string key="lifecycle:transition" value="completed"/>
  </global>
  <string key="lifecycle:model" value="mpaf"/>
  <trace>
    <event>
      <string key="concept:name" value="Birth"/>
      <string key="lifecycle:transition" value="completed"/>
      <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
    </event>
    <event>
      <string key="concept:name" value="Transfer"/>
      <string key="lifecycle:transition" value="Completed"/>
      <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
    </event>
  </trace>
</log>
```

Figure 18: XES log extracted with the script from Figure 17, only top part shown, entire file available online\(^{12}\)

```
The script is valid.
- Warning on Ln 2, Col 0: XES compliance problem: The XES standard extension attribute 'time:timestamp' must be specified as a global event attribute, in order to comply with XES certification level B1. As this was not done explicitly in the script, a global event attribute definition with a default value will be added automatically during extraction.
- Warning on Ln 2, Col 0: XES compliance problem: The XES standard extension attribute 'lifecycle:transition' must be specified as a global event attribute, in order to comply with XES certification level B1. As this was not done explicitly in the script, a global event attribute definition with a default value will be added automatically during extraction.
```

Figure 19: Validation result for the script from Figure 17

---

\(^{12}\) [https://www.dropbox.com/s/nobxgedxidij190v/B1_2.xes?dl=0](https://www.dropbox.com/s/nobxgedxidij190v/B1_2.xes?dl=0)
Contrary to `concept:name`, the `lifecycle:transition` and `time:timestamp` attributes are optional and users are not required to define them in a script. However, if users add one of these two attributes anywhere in the script, i.e., to an XES event emission or to a definition of a global event attribute, the validator enforces that the two attributes are defined for all XES event emissions.

Consider for example the script in Figure 20 where the user modified the script from Figure 2 and only added the `time:timestamp` attribute once in the context of the birth event emission (line 15). The validation output for this script is shown in Figure 21. In compliance with the XES certification level B1, there are error messages related to the missing `time:timestamp` attribute for the transfer event and the missing `lifecycle:transition` attributes for both events. Moreover, the validator issues warnings that there are no explicit global event attribute definitions for these two attributes.

Similarly, the script in Figure 22 is a modified version of the script from Figure 2 where the `lifecycle:transition` attribute is specified as a global event attribute (line 6), but not used anywhere else in the script. The validation output for this script is shown in Figure 23. Again, there are error messages related to the missing `time:timestamp` and `lifecycle:transition` attributes for both events. Moreover, the validator issues a warning that there is no explicit global event attribute definition for `time:timestamp`.

![Figure 20: Adding the time:timestamp attribute only for the birth event to the script from Figure 2](image)
Figure 21: Validation result for the script from Figure 20

```
// Preamble
connect_ipc("/data2/eth-archive/chaindata/geth.ipc");
setOutputFolder("./output");
addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");

// Export definition
BLOCKS (6000000) (6000024) {
  LOG ENTRIES
  (0x8b012c8c97f7ba5d6e2f37070f9587fb8e7a06d)
  (Birch(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
  {
    EMIT XES EVENT ("B_1") (kittyId)()
    "Birth" as xs:string concept:name
  }

  LOG ENTRIES
  (0x8b012c8c97f7ba5d6e2f37070f9587fb8e7a26d)
  (Transfer(address from, address to, uint256 tokenId))
  {
    EMIT XES EVENT ("B_1") (tokenId)()
    "Transfer" as xs:string concept:name
  }
}
```

Figure 22: Adding the lifecycle:transition attribute as a global event attribute to the script from Figure 2
The script is invalid.
- Error on ln 14, Col 8: XES compliance problem: The XES standard extension attribute 'lifecycle:transition' is used in the script. Thus, all XES event emissions must also contain the standard extension attribute 'time:timestamp' to comply with XES certification level B1.
- Error on ln 14, Col 8: XES compliance problem: The XES standard extension attribute 'lifecycle:transition' is used in the script. Thus, all XES event emissions must also contain this attribute to comply with XES certification level B1.
- Error on ln 23, Col 8: XES compliance problem: The XES standard extension attribute 'lifecycle:transition' is used in the script. Thus, all XES event emissions must also contain the standard extension attribute 'time:timestamp' to comply with XES certification level B1.
- Error on ln 23, Col 8: XES compliance problem: The XES standard extension attribute 'lifecycle:transition' is used in the script. Thus, all XES event emissions must contain this attribute to comply with XES certification level B1.
- Warning on ln 2, Col 8: XES compliance problem: The XES standard extension attribute 'time:timestamp' must be specified as a global event attribute, in order to comply with XES certification level B1. As this was not done explicitly in the script, a global event attribute definition with a default value will be added automatically during extraction.

Figure 23: Validation result for the script from Figure 22
**Level B2**

The `lifecycle:transition` and `time:timestamp` attributes can be used for event classifiers, if the script satisfies the conditions of certification level B1. This is shown in Figure 24 where in addition to the classifier “Event Name” (line 9), a classifier “Event Name and Transition” based on the attributes `concept:name` and `time:timestamp` is added (line 10) to the script from Figure 15. Executing the script results in the XES file shown in Figure 25. The file now contains the two classifiers specified by the user (lines 17 and 18).

![Figure 24: Adding classifiers to the script from Figure 15](image)

![Figure 25: XES log extracted with the script from Figure 24, only top part shown, entire file available online](image)

13 [https://www.dropbox.com/s/b4wa5g9tebxdudw/B2_1.xes?dl=0](https://www.dropbox.com/s/b4wa5g9tebxdudw/B2_1.xes?dl=0)
Level C1

Users can also define the org:resource attribute for events as shown in Figure 26. Here, org:resource is defined as a global event attribute (line 6). The identifier of the CryptoKitty (kittyId and tokenId) is also used as the value for this attribute (lines 16 and 26). Note that the identifiers from the Ethereum log entries are integer values, but org:resource is defined as a string attribute in the Organizational extension. Here, the emission statements use explicit type conversion to cast the integer values into string values. In general, ELF supports type conversions known from conventional programming languages. Unsupported conversions, for example, when casting strings into integers, are flagged by the validator.

![Figure 26: Adding the org:resource attribute to all events in the script from Figure 2](image-url)
The execution of the script from Figure 26 yields the XES file in Figure 27. Due to the use of the `org:resource` attribute, the Organizational extension was automatically added (line 9). Moreover, the attribute is defined as a global event attribute according to the specification from the script (line 13) and the CryptoKitty identifiers are added as `org:resource` attributes to all events (lines 18 and 22).

![XML code]

Figure 27: XES log extracted with the script from Figure 26, only top part shown, entire file available online\(^\text{14}\)

\(^{14}\) [https://www.dropbox.com/s/56gorj24z3t1nih/C1_1.xes?dl=0](https://www.dropbox.com/s/56gorj24z3t1nih/C1_1.xes?dl=0)
Similar to the lifecycle:transition and time:timestamp attributes, the org:resource attribute is optional, but if it is used anywhere in the script, it must be added to all XES event.

For example, the script in Figure 28 adds the org:resource attribute only to the birth event (line 15) and does not specify it as a global event attribute. As shown in Figure 29, for this script the validator issues an error message that the org:resource attribute must be added to the transfer event as well. Moreover, the validator also emits a warning that the org:resource attribute was not explicitly defined as a global event attribute and that such a definition will hence automatically be added during extraction.

Similarly, the script in Figure 30 specifies the org:resource attribute as a global event attribute (line 6) without adding it to any of the XES event emissions. The validator recognizes this and returns error messages that ask the user to add the attribute to both XES event emissions, see Figure 31.

```
1 // Preamble
2 connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3 setOutputFolder("./output");
4
5 addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6
7 // Export definition
8 BLOCKS (6000008) (6000024) {
9   LOG EVENTS
10     (8x06012c8cf978e6d5deae237870f9587f8e7a266d)
11       (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
12       {
13         EMIT XES EVENT ("C1_2")(kittyId)(
14           "Birth" as xs:string concept:name,
15           kittyId as xs:string org:resource
16         );
17       };
18
19   LOG ENTRIES
20     (8x06012c8cf978e6d5deae237870f9587f8e7a266d)
21       (Transfer(address from, address to, uint256 tokenId))
22       {
23         EMIT XES EVENT ("C1_2")(tokenId)(
24           "Transfer" as xs:string concept:name
25         );
26       }
27 }
```

Figure 28: Adding the org:resource attribute only for the birth event to the script from Figure 2

Figure 29: Validation result for the script from Figure 28
Figure 30: Adding the org:resource attribute as a global event attribute to the script from Figure 2

```plaintext
// Preamble
connectIpc("/data2/geth-archive/chaindata/geth.ipc");
setOutputFolder("./output");
addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");

// Export definition
BLOCKS (6000000) (6000024) {
  LOG ENTRIES
  (0x08012c8f9be4d050e4e570f597f7e7a266d)
  (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
  
  EMT XES EVENT ("C1_3") (kittyId) {
    "Birth" as xs:string concept:name
  
  };

  LOG ENTRIES
  (0x08012c8f9be4d050e4e570f597f7e7a266d)
  (Transfer(address from, address to, uint256 tokenId))
  
  EMT XES EVENT ("C1_2") (tokenId) {
    "Transfer" as xs:string concept:name
  
  };
}
```

Figure 31: Validation result for the script from Figure 30

The script is invalid.
- Error on ln 14, col 8: XES compliance problem: The XES standard extension attribute 'org:resource' is used in the script. Thus, all XES event emissions must contain this attribute to comply with XES certification level C1.
- Error on ln 23, col 8: XES compliance problem: The XES standard extension attribute 'org:resource' is used in the script. Thus, all XES event emissions must contain this attribute to comply with XES certification level C1.
Level C2

The `org:resource` attribute can also be used in event classifiers as shown in Figure 32 (line 9). As expected, the execution of this script results in an XES file with two event classifiers, see in Figure 33 (lines 15 to 16).

```java
// Preamble
connectIpc("/data2/ethereum-archive/chaindata/ethereum.ipc");
setOutputFolder("./output");
addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
addXesEventClassifier("Event Name", {"concept:name"});
addXesEventClassifier("Event Name and Resource", {"concept:name", "org:resource"});
// Export definition
BLOCKS (6000000) (6000024) {
```

Figure 32: Adding global event classifiers to the script from Figure 26

```xml
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
  <extension name="Organizational" prefix="org" url="http://www.xes-standard.org/organizational.xesext"/>
  <extension name="Concept" prefix="concept" url="http://www.xes-standard.org/concept.xesext"/>
  <global scope="event">
    <string key="concept:name" value="default activity"/>
    <string key="org:resource" value="default resource"/>
  </global>
  <classifier name="Event Name" keys="concept:name"/>
  <classifier name="Event Name and Resource" keys="concept:name org:resource"/>
  <trace>
    <event>
      <string key="concept:name" value="Birth"/>
      <string key="org:resource" value="851838"/>
    </event>
    <event>
      <string key="concept:name" value="Transfer"/>
      <string key="org:resource" value="851838"/>
    </event>
  </trace>
</log>
```

Figure 33: XES log extracted with the script from Figure 32, only top part shown, entire file available online

15 [https://www.dropbox.com/s/unffbqqqi6x9arv/C2_1.xes?dl=0](https://www.dropbox.com/s/unffbqqqi6x9arv/C2_1.xes?dl=0)
**Level D1**

In addition to the standard extension attributes covered by the certification levels A-C, ELF also supports the remaining standard extension attributes. While users can add these attributes to any XES event emission, ELF does not explicitly add global event attribute definitions for standard attributes that are not covered by certification levels A to C. If required, such definitions must be added manually by the user.

For example, the script in Figure 34 extends the base script from Figure 2. That is, the `org:role` attribute is now added to the birth event (line 17). Additionally, both events contain the `cost:total` attribute that provides information regarding the specific cost associated with the execution of the transfer transaction (line 28 and 30). The cost is calculated from information about the transaction that included the log entry (line 14 and 26). Here, `tx.gasPrice` is the price that the sender or requester of the transaction was willing to pay per unit of gas and `tx.gasUsed` is the amount of gas that was actually consumed by this transaction. Note that gas is a unit used to measure the computational effort of operation execution on Ethereum networks. Lastly, the `cost:total` attribute is also defined as a global event attribute (line 6).

![Figure 34: Adding org:role and cost:total attributes to the script from Figure 2](image-url)
The output generated by this script is shown in Figure 35. ELF recognized the use of attributes from the Cost and Organizational extension and automatically adds the two standard extensions (lines 9 to 10). Moreover, the birth events now contain the org:role attribute (line 19), whereas all events also contain the cost:total attribute (line 20 and 24). Moreover, the cost:total attribute was defined as a global event attribute following the specification from the script (line 14).

![XES log extracted with the script from Figure 34, only top part shown, entire file available online](https://www.dropbox.com/s/qbpr0tatm71o2wn/D1_1.xes?dl=0)

16 [https://www.dropbox.com/s/qbpr0tatm71o2wn/D1_1.xes?dl=0](https://www.dropbox.com/s/qbpr0tatm71o2wn/D1_1.xes?dl=0)
When a standard extension attribute is defined as a global event attribute, ELF ensures that it is added to all XES event emissions. For example, consider that the user forgot to add the `cost:total` attribute to the transfer event (lines 26 to 28) in Figure 36. The validator realizes the missing `cost:total` attribute for the transfer event and issues the error message from Figure 37.

**Figure 36:** Removing `cost:total` from the transfer event emission in the script from Figure 34

**Figure 37:** Validation result for the script from Figure 36
In general, ELF checks that the standard extension attributes exist and that their types are correctly used. For example, the script in Figure 38 contains a few errors.

1. line 6: an attribute with the name cost:tota is not defined in the Cost extension;
2. line 7: the org:role attribute must be of type xs:string and not of type xs:boolean;
3. line 18: the org:role attribute must be of type xs:string and not of type xs:int; and
4. line 31: there is no standard extension with the cst-prefix, hence an cst:total does not exist.

All these errors are identified by ELF’s validator which issues the error messages in Figure 39.
Level D2

The standard extension attributes can be used to define event classifiers. For example, in Figure 40 an event classifier based on the concept:name and cost:total attributes (line 10) is added to the script from Figure 34. Based on this modification, the exported log now also contains this classifier, see Figure 41 (line 17).

```java
// Preamble
connectTcp("/data2/geth-archive/chaindata/geth.ipc");
setOutputFolder("./output");
addGlobalXesEventAttribute("concept:name", "string", "default activity");
addGlobalXesEventAttribute("cost:total", "float", 0);
addXesEventClassifier("Event Name", ["concept:name"]);
addXesEventClassifier("Event Name and Cost", ["concept:name", "cost:total"]);

// Export definition
BLOCKS (6000000) (6000024) {
```

Figure 40: Adding an event classifier to the script from Figure 34

```xml
<log xes:version="1.0" encoding="UTF-8"/>
<extension name="Organizational" prefix="org" url="http://www.xes-standard.org/organization.xesext"/>
<extension name="Cost" prefix="cost" url="http://www.xes-standard.org/cost.xesext"/>
<extension name="Concept" prefix="concept" url="http://www.xes-standard.org/concept.xesext"/>
<global xref="event">
  <string key="concept:name" value="default activity"/>
  <float key="cost:total" value="0.0"/>
</global>
<classifier xref="Event Name" keys="concept:name"/>
<classifier xref="Event Name and Cost" keys="concept:name cost:total"/>
<trace>
  <event>
    <string key="concept:name" value="Birth"/>
    <string key="org:role" value="CryptoKitty"/>
    <float key="cost:total" value="8.26438576E15"/>
  </event>
  <event>
    <string key="concept:name" value="Transfer"/>
    <float key="cost:total" value="8.26438576E15"/>
  </event>
</trace>
```

Figure 41: XES log extracted with the script from Figure 40, only top part shown, entire file available online

17 https://www.dropbox.com/s/gg9ocj1oftuzwj2/D2_1.xes?dl=0
ELF verifies that standard extension attributes which are part of classifiers are defined as global event attributes. If this is not the case, the validator returns a respective error message. For example, the script in Figure 42 adds the `org:role` attribute instead of the `cost:total` attribute to the classifier (line 9). As `org:role` is not defined as a global event attribute, this modification results in the error in Figure 43.

![Figure 42: Removing global event attributes from the script in Figure 40](image)

```plaintext
// Preamble
connectipc("/data2/chaindata/chaindata/geth.ipc");
setOutputFolder("./output");
addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
addGlobalXesEventAttribute("cost:total", "xs:float", 0);
addXesEventClassifier("Event Name",{"concept:name");
addXesEventClassifier("Event Name and Cost",{"concept:name", "org:role");
// Export definition
BLOCKS (600000) (6000024) {
```

Figure 43: Validation result for the script from Figure 42

The script is invalid.
- Error on Ln 9, Col 0: XES compliance problem: The attribute 'org:role' is part of an XES event classifier, but is not defined as a global event attribute.
Flag X1

Depending on the use case, analysts might require data that cannot be modeled by the XES standard extension attributes. In such situations, users can add arbitrary attributes. Note that currently ELF does not support the use of custom extensions, but it supports all standard extensions.

For example, the script in Figure 44 uses the standard extension attributes covered by certification levels A-C and includes additional information via several non-standard attributes. The attributes blockNumber, txHash, txSender and txRecipient are emitted for both log entry types (lines 22 to 25 and 38 to 41). These attributes provide standard information about the block and the transaction that included the respective log entry. Moreover, the attribute txRecipient is added as a global event attribute (line 9).

```
// Preamble
connectTcp("/data2/geth-archive/chaindata/geth.ipc");
setOutputFolder("./output");

addGlobalXesEventAttribute(\"concept:name\", \"x:string\", \"default activity\", [\"\"]);
addGlobalXesEventAttribute(\"lifecycle:transition\", \"x:string\", \"Open\", [\"\"]);
addGlobalXesEventAttribute(\"timestamp\", \"x:date\", \"162346712\", [\"\"]);
addGlobalXesEventAttribute(\"org:resource\", \"x:string\", \"default resource\", [\"\"]);
addGlobalXesEventAttribute(\"txRecipient\", \"x:string\", \"0x80e128cf978e0d3d3e23378f987f887e7a266d\", [\"\"]);

// Export definition

BLOCKS (0000000000000000) { 

LOG ENTRIES
(bx80e128cf978e0d3d3e23378f987f887e7a266d)
(Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))

EMIT XES EVENT (\"X1-1\")((kittyId,))
\"Birth\" as x:string concept:name,
\"completed\" as x:string lifecycle:transition,
block.timestamp as x:date time:timestamp,
kittyId as x:string org:resource,
block.number as x:int blockNumber,
tx.hash as x:string txHash,
tx.from as x:string txSender,
tx.to as x:string txRecipient

LOG ENTRIES
(bx80e128cf978e0d3d3e23378f987f887e7a266d)
(Transfer(address from, address to, uint256 tokenId))

EMIT XES EVENT (\"X1-1\")((tokenId,))
\"Transfer\" as x:string concept:name,
\"completed\" as x:string lifecycle:transition,
block.timestamp as x:date time:timestamp,
tokemId as x:string org:resource,
block.number as x:int blockNumber,
tx.hash as x:string txHash,
tx.from as x:string txSender,
tx.to as x:string txRecipient

}

```

Figure 44: Using attributes that are not defined in standard extensions
The execution of the script results in the log from Figure 45. Here, the standard extension attributes and the txRecipient attribute are defined as global event attributes (lines 13 to 19). All events comprise the specified standard extension attributes and additionally the blockNumber, txHash, txSender and txRecipient attributes (lines 23, 26, 27 and 29 and lines 33, 36, 37 and 39).

Figure 45: XES log extracted with the script from Figure 44, only top part shown, entire file available online

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the XML serialization of the XES standard for log storage and -->
<!-- management. -->
  <extension name="Organizational" prefix="org" url="http://www.xes-standard.org/org.xesext"/>
  <extension name="Time" prefix="time" url="http://www.xes-standard.org/time.xesext"/>
  <extension name="Lifecycle" prefix="lifecycle" url="http://www.xes-standard.org/lifecycle.xesext"/>
  <extension name="Concept" prefix="concept" url="http://www.xes-standard.org/concept.xesext"/>
  <global scope="event">
    <string key="concept:name" value="default activity"/>
    <string key="lifecycle:transition" value="Open"/>
    <date key="time:timestamp" value="2021-07-08T22:51:52+00:00"/>
    <string key="org:resources" value="default resource"/>
    <string key="tx:recipient" value="5d569286c4d7f35d06ce5243787af877887a2d9d"/>
  </global>
  <trace>
    <event>
      <string key="txSender" value="8a2a98476934651436e8dec96b5e51686968291"/>
      <string key="org:resource" value="8518c6"/>
      <string key="concept:name" value="Birthday"/>
      <string key="tx:recipient" value="8e68df61e9020d5a6b1000ba82986ed77c376be75edeffec22ac68572cc9f9876"/>
      <date key="time:timestamp" value="2018-07-21T0:0:29+04:10:00"/>
    </event>
    <event>
      <string key="txSender" value="8a2a98476934651436e8dec96b5e51686968291"/>
      <string key="org:resource" value="8518c6"/>
      <string key="concept:name" value="Transfer"/>
      <string key="tx:recipient" value="8e68df61e9020d5a6b1000ba82986ed77c376be75edeffec22ac68572cc9f9876"/>
      <date key="time:timestamp" value="2018-07-21T0:0:29+04:10:00"/>
    </event>
    <event>
      <string key="txSender" value="8a2a98476934651436e8dec96b5e51686968291"/>
      <string key="org:resource" value="8518c6"/>
      <string key="concept:name" value="Transfer"/>
      <string key="tx:recipient" value="8e68df61e9020d5a6b1000ba82986ed77c376be75edeffec22ac68572cc9f9876"/>
      <date key="time:timestamp" value="2018-07-21T0:0:29+04:10:00"/>
    </event>
    <event>
      <string key="txSender" value="8a2a98476934651436e8dec96b5e51686968291"/>
      <string key="org:resource" value="8518c6"/>
      <string key="concept:name" value="Transfer"/>
      <string key="tx:recipient" value="8e68df61e9020d5a6b1000ba82986ed77c376be75edeffec22ac68572cc9f9876"/>
      <date key="time:timestamp" value="2018-07-21T0:0:29+04:10:00"/>
    </event>
  </trace>
</log>
```

18 [https://www.dropbox.com/s/h7y2y43e3bm3jbt/X1_1.xes?dl=0](https://www.dropbox.com/s/h7y2y43e3bm3jbt/X1_1.xes?dl=0)
The validator performs various checks to support the use of non-standard attributes. For example, the script in Figure 46 comprises four different types of errors.

1. Lines 17 to 25: The txRecipient attribute was defined as a global event attribute (line 9), but the birth event emission does not include this attribute.
2. Line 24: tx.from encodes the identifier of the account that requested the transaction. On Ethereum it has the type address (a hexadecimal string of length 20), but the script tries to emit it as an integer. This is an unsupported type conversion.
3. Line 37: The blockNumber attribute is not used consistently. It is emitted as an integer value for the birth event (line 22) and as a string value for the transfer event (line 37).
4. Line 40: The txRecipient attribute is also not used consistently. In the global event definition, its type was set to xs:string (line 9), but in the transfer event emission the type was changed to xs:int (line 40).

The validator recognizes these errors and when processing the script returns the error messages in Figure 47.
Figure 46: Invalid changes to the script from Figure 44

The script is invalid.
- Error on ln 17, Col 8: XES compliance problem: The XES event attribute 'txRecipient' was defined globally. It must thus be added to all XES event emissions to comply with XES certification level X1.
- Error on Ln 24, Col 12: XES compliance problem: The solidity type 'address' cannot be exported as XES type 'xs:Int'.
- Error on Ln 37, Col 38: XES compliance problem: The XES event attribute 'blockNumber' was already defined with a different type at Ln 22, Col 35.
- Error on Ln 40, Col 42: XES compliance problem: The XES event attribute 'txRecipient' was already defined with a different type at Ln 9, Col 0.

Figure 47: Validation result for the script from Figure 46
Flag X2

Non-standard attributes can also be used to define classifiers. For example, the script in Figure 48 adds an event classifier "Transaction recipient" to the script from Figure 44 (line 11). This classifier uses the txRecipient attribute which is also defined as a global event attribute (line 9). As shown in Figure 49, when exporting the data using this script, the log now contains the specified classifier (line 20).

Figure 48: Defining an event classifier based on a non-standard attribute, based on the script from Figure 44

```
// Preamble
connectIpc("/data2/geth-archive/chaindata/../../ipc");
setOutputFolder("/output");
addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
addGlobalXesEventAttribute("time:timestamp", "xs:dateTime", "2021-07-06T22:51:52+10:00");
addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
addGlobalXesEventAttribute("txRecipient", "xs:string", "0x060126c4f978e005d6e237070f9597fe8e265d");
addEventClassifier("Transaction Recipient", { "txRecipient" });
```

Figure 49: XES log extracted with the script from Figure 48, only top part shown, entire file available online

https://www.dropbox.com/s/4rs23rmnojazt8s/X2_1.xes?dl=0
However, users cannot add attributes to classifiers without specifying them as global event attributes. For example, the script in Figure 50 defined a second classifier “Transaction Sender” (line 12). It contains the txSender attribute which was not defined as a global event attribute. This is recognized by the validator which informs the user about the problem by issuing the error message in Figure 51.

```
// Preamble
connectIp("/data/geth-archive/chaindata/-geth.ipc");
setOutputFolder("./output");
addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
addGlobalXesEventAttribute("life_cycle:transition", "xs:string", "Open");
addGlobalXesEventAttribute("time:timestamp", "xs:date", 1625748712);
addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
addGlobalXesEventAttribute("tx:Recipient", "xs:string", "bd85612c0cf97eada5dea6eac23807f695b8f8e74266d");
addXesEventClassifier("Transaction Recipient", {"tx:Recipient"});
addXesEventClassifier("Transaction Sender", {"txSender"});
```

**Figure 50:** Adding a second classifier to the script from Figure 48

```
The script is invalid.
- Error on line 12, col 8: XES compliance problem: The attribute 'txSender' is part of an XES event classifier, but is not defined as a global event attribute.
```

**Figure 51:** Validation result for the script from Figure 50
Appendix A: Proof for CryptoKitties Data

Figure 52: One birth and one transfer log entry were created by the CryptoKitties smart contract in block 6,000,000
Figure 53: Details for transaction 0xa8f2cf69e302da6c8100b80298ed77c37b6e75eed1177ca22ac5772c9fb9876 that included one birth and one transfer log entry from the CryptoKitties smart contract in block 6,000,001
Figure 54: One birth and one transfer log entry were created by the CryptoKitties smart contract in block 6,000,001
Figure 55: Details for transaction 0x7fa569ac010ceac5ac405e4fb5d8d7e050e8362c0d39daaf9609b965bd847c7b8 that included one birth and one transfer log entry from the CryptoKitties smart contract in block 6,000,001
Figure 56: No log entries were created by the CryptoKitties smart contract in block 6,000,002
Figure 57: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,003
Figure 58: Details for transaction 0xb81146ccfa12bf24bac2709e925597841ae9843418a7afaee39421be140d7c1c that included one transfer log entry from the CryptoKitties smart contract in block 6,000,003
Figure 59: No log entries were created by the CryptoKitties smart contract in block 6,000,044.

Figure 60: No log entries were created by the CryptoKitties smart contract in block 6,000,005.
Figure 61: No log entries were created by the CryptoKitties smart contract in block 6,000,006

Figure 62: No log entries were created by the CryptoKitties smart contract in block 6,000,007
Figure 63: No log entries were created by the CryptoKitties smart contract in block 6,000,008

Figure 64: No log entries were created by the CryptoKitties smart contract in block 6,000,009
Figure 65: No log entries were created by the CryptoKitties smart contract in block 6,000,010

Figure 66: No log entries were created by the CryptoKitties smart contract in block 6,000,011
Figure 67: No log entries were created by the CryptoKitties smart contract in block 6,000,012

Figure 68: No log entries were created by the CryptoKitties smart contract in block 6,000,013
Figure 69: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,014
Figure 70: Details for transaction 0x29c09f80fc1c5141faea0795d2398e55a92218184db4be283129ce72c7b2c0f that included one transfer log entry from the CryptoKitties smart contract in block 6,000,014
Figure 71: No log entries were created by the CryptoKitties smart contract in block 6,000,015
Figure 72: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,016
**Figure 73:** Details for transaction 0x050a11c46f9f29c21883c9df55e37bf7170ba9c45f4d673ba21f2ea7dce7260 that included one transfer log entry from the CryptoKitties smart contract in block 6,000,016

---

**EXPORT**

---
Figure 74: No log entries were created by the CryptoKitties smart contract in block 6,000,017

Figure 75: No log entries were created by the CryptoKitties smart contract in block 6,000,018
Figure 76: No log entries were created by the CryptoKitties smart contract in block 6,000,019

Figure 77: No log entries were created by the CryptoKitties smart contract in block 6,000,020
Figure 78: Two birth and two transfer log entries were created by the CryptoKitties smart contract in block 6,000,021
Figure 79: Details for transaction 0x3d2a60292a8e713ac489756919f416972a6034460fe6f5f5424bf263357120e that included two birth and two transfer log entries from the CryptoKitties smart contract in block 6,000,021
Figure 80: No log entries were created by the CryptoKitties smart contract in block 6,000,022

Figure 81: No log entries were created by the CryptoKitties smart contract in block 6,000,023
Figure 82: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,024
Figure 83: Details for transaction 0xc4fddacabcb09a5ab96f4e0b20e0c2a2a5a3ecb7420ed2adfb41425b25ca that included a transfer log entry from the CryptoKitties smart contract in block 6,000,024
# Contact Information

<table>
<thead>
<tr>
<th>WIL VAN DER AALST</th>
<th>CHRISTIAN GÜNTER</th>
<th>ERIC VERBEEK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tel +31 40 247 4295</td>
<td>Tel +31 64 1780680</td>
<td>Tel +31 40 247 3755</td>
</tr>
<tr>
<td><a href="mailto:w.m.p.v.d.aalst@tue.nl">w.m.p.v.d.aalst@tue.nl</a></td>
<td><a href="mailto:christian@fluxicon.com">christian@fluxicon.com</a></td>
<td><a href="mailto:h.m.w.verbeek@tue.nl">h.m.w.verbeek@tue.nl</a></td>
</tr>
</tbody>
</table>

IEEE XES Working Group  
IEEE Task Force on Process Mining  
http://www.win.tue.nl/ieeetfpm