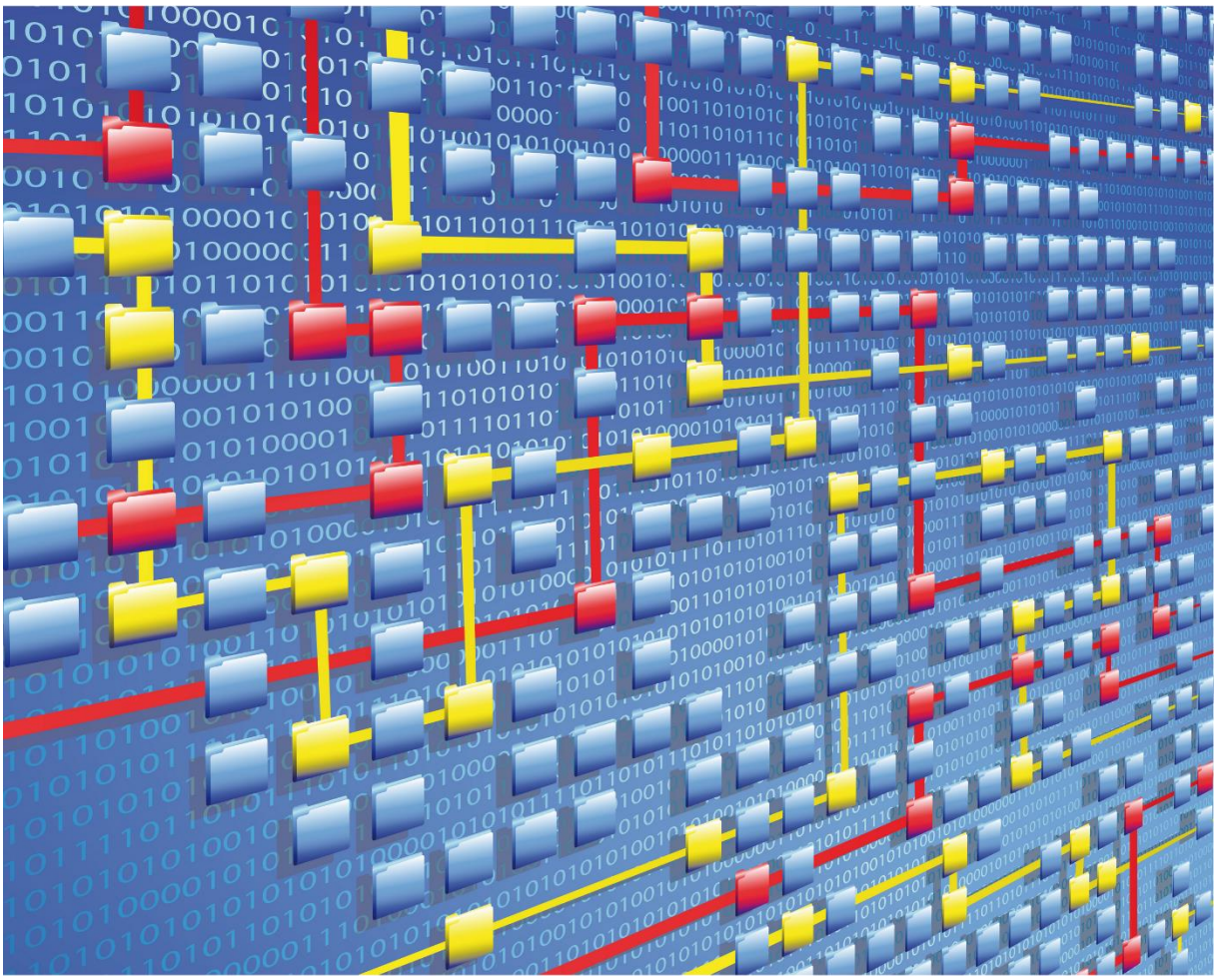
IEEE Task Force on



Process Mining

XES CERTIFICATION FOR THE ETHEREUM LOGGING FRAMEWORK

TABLE OF CONTENTS

Contents

EXPORT

Tool

NAME

Ethereum Logging Framework[1]

VENDOR

CSIRO Data61

VERSION

0.2.1

REQUESTED CERTIFICATION LEVELS

Export

A-X

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

EXPORT

Meta

AUTHORS

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

DATE

7/20/2021

HISTORY

CHANGES

AUTHOR(S)	DATE	DESCRIPTION
Christopher Klinkmüller	2/7/2021	Creation of the report
Christopher Klinkmüller	20/7/2021	The ELF validator now enforces that all defined global event attributes were added to all XES emission statements

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 focusses 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/

[3] C. Klinkmüller, A. Ponomarev, A.B. Tran, I. Weber, W. van der Aalst (2019): "Mining Blockchain Processes: Extracting Process Mining Data from Blockchain Applications". In: 17th International Conference on Business Process Management (Blockchain Forum).

[4] C. Klinkmüller, I. Weber, A. Ponomarev, A.B. Tran, W. van der Aalst (2020): Efficient Logging for Blockchain Applications. arXiv:2001.10281.

[5] R. Hobeck, C. Klinkmüller, H.M.N.D. Bandara, I. Weber, W. van der Aalst (2021): Process Mining on Blockchain Data: A Case Study of Augur. In: 19th International Conference on Business Process Management.

[6] https://etherscan.io/tx/0x691f348ef11e9ef95d540a2da2c5f38e36072619aa44db0827e1b8a276f120f4

[7] https://etherscan.io

Table 1: The birth log entries created by CryptoKitties in the block range [6000000,6000024]

block number	owner	kittyId	matronId	sireId	genes
6000000	0x7891f796a5d43466fc29f102069092aef497a290	851836	733402	843147	6.837720380099826718906138032746055348279316135467146480889348173259407 39e+71
6000001	0x9d2ac7c3e17163f104e6abf5374f502b9f1db102	851837	851455	848263	3.453234984776826125556236409611028125531409057553115343572559345757400 35e+71
6000021	0xdfad6357ae19cad45a316335f428f3c61c32ffb0	851838	564479	733495	4.583329537453338277917448213499516159551482484416391838533978429944957 19e+71
6000021	0x837ed29de4cab664c550b721bf26dfc028ef6689	851839	851652	851664	4.491149161912086916725015255410540923111554874719806464491423463983145 29e+71

Table 2: The transfer log entries created by CryptoKitties in the block range [6000000,6000024]

block number	from	to	tokenId
6000000	0x0000000000000000000000000000000000000000	0x7891f796a5d43466fc29f102069092aef497a290	851836
6000001	0x0000000000000000000000000000000000000000	0x9d2ac7c3e17163f104e6abf5374f502b9f1db102	851837
6000002	0xb1690c08e213a35ed9bab7b318de14420fb57d8c	0xc9a3a9a083a54cf124d8778df29e75b0b6dea159	807523
6000014	0xefe090106ca863145f4a0d50a46021d0643efd6a	0x7ec915b8d3ffee3deaae5aa90def8ad826d2e110	816161
6000016	0xb1690c08e213a35ed9bab7b318de14420fb57d8c	0x36ed2d75a82e180e0871456b15c239b73b4ee9f4	572791
6000021	0x0000000000000000000000000000000000000000	0xdfad6357ae19cad45a316335f428f3c61c32ffb0	851838
6000021	0x0000000000000000000000000000000000000000	0x837ed29de4cab664c550b721bf26dfc028ef6689	851839
6000024	0xb1690c08e213a35ed9bab7b318de14420fb57d8c	0x9d2ac7c3e17163f104e6abf5374f502b9f1db102	699686

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.

```
1   // Preamble
2   connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3   setOutputFolder("./test_output");
4
5   // Export definition
6   BLOCKS (6000000) (6000024) {
7       LOG ENTRIES
8           (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
9           (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
10      {
11          EMIT XES EVENT ("CryptoKitties")(kittyId)()(
12              // specification of the exported attributes
13          );
14      }
15
16      LOG ENTRIES
17          (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
18          (Transfer(address from, address to, uint256 tokenId))
19      {
20          EMIT XES EVENT ("CryptoKitties")(tokenId)()(
21              // specification of the exported attributes
22          );
23      }
24  }
```

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.

EXPORT

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).

```
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 (6000000) (6000024) {
9      LOG ENTRIES
10         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
11         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
12     {
13         EMIT XES EVENT ("A1_1")(kittyId)()(
14             "Birth" as xs:string concept:name
15         );
16     }
17
18     LOG ENTRIES
19         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
20         (Transfer(address from, address to, uint256 tokenId))
21     {
22         EMIT XES EVENT ("A1_1")(tokenId)()(
23             "Transfer" as xs:string concept:name
24         );
25     }
26 }
```

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).

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
10     <global scope="event">
11         <string key="concept:name" value="default activity"/>
12     </global>
13     <trace>
14         <event>
15             <string key="concept:name" value="Birth"/>
16         </event>
17         <event>
18             <string key="concept:name" value="Transfer"/>
19         </event>
20     </trace>
21     <trace>
22         <event>
23             <string key="concept:name" value="Birth"/>
24         </event>
25         <event>
26             <string key="concept:name" value="Transfer"/>
27         </event>
28     </trace>
```

Figure 3: XES log extracted with the script from Figure 2, only top part shown, entire file available online[8]

[8] 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).

```
1   // Preamble
2   connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3   setOutputFolder("./output");
4
5   // Export definition
6   BLOCKS (6000000) (6000024) {
```

Figure 4: The modified version of the script from Figure 2 without the global attribute definition

```
The script is valid.
- Warning on Ln 2, Col 0: XES compliance problem: The XES standard extension
attribute 'concept:name' must be specified as a global event attribute, in order
to comply with XES certification level A1. 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 5: Validation result of the script from Figure 4

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
10     <global scope="event">
11         <string key="concept:name" value="No global value for concept:name defined"/>
12     </global>
13     <trace>
14         <event>
15             <string key="concept:name" value="Birth"/>
16         </event>
17         <event>
18             <string key="concept:name" value="Transfer"/>
19         </event>
20     </trace>
```

Figure 6: XES log extracted with the script from Figure 4, only top part shown, entire file available online[9]

[9] 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.

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default value");
6
7  // Export definition
8  BLOCKS (6000000) (6000024) {
9      LOG ENTRIES
10         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
11         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
12     {
13         EMIT XES EVENT ("A1")(kittyId)()(
14             "Birth" as xs:string name
15         );
16     }
17
18     LOG ENTRIES
19         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
20         (Transfer(address from, address to, uint256 tokenId))
21     {
22         EMIT XES EVENT ("A1")(tokenId)()(
23             "Transfer" as xs:string concept:name
24         );
25     }
26 }
```

Figure 7: The EthQL script is invalid, as it does not specify concept:name for XES events

```
The script is invalid.
- Error on Ln 13, Col 8: XES compliance problem: XES event emissions must contain
the 'concept:name' attribute to comply with XES certification level A1.
```

Figure 8: Validation result for the script from Figure 7

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.

```
14                  kittyId as xs:int concept:name
```

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

```
5   addGlobalXesEventAttribute("concept:name", "xs:int", 0);
```

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

```
The script is invalid.
- Error on Ln 14, Col 30: XES compliance problem: The type 'xs:int' was specified
for attribute 'concept:name'. This violates the definition from the standard
extension, where the type 'xs:string' was specified.
```

Figure 11: Validation result for the script from Figure 9

```
The script is invalid.
- Error on Ln 5, Col 0: XES compliance problem: The type 'xs:int' was specified
for attribute 'concept:name'. This violates the definition from the standard
extension, where the type 'xs:string' was specified.
```

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
5   addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6
7   addXesEventClassifier("Event Name", {"concept:name"});
8
9   // Export definition
10  BLOCKS (6000000) (6000024) {
```

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

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Concept" prefix="concept" uri="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]

[10] https://www.dropbox.com/s/73nkdho6nbudafe/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.

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
7  addGlobalXesEventAttribute("time:timestamp", "xs:date", 1625748712);
8
9  // Export definition
10 BLOCKS (6000000) (6000024) {
11     LOG ENTRIES
12         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
13         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
14     {
15         EMIT XES EVENT ("B1_1")(kittyId)()(
16             "Birth" as xs:string concept:name,
17             "Completed" as xs:string lifecycle:transition,
18             block.timestamp as xs:date time:timestamp
19         );
20     }
21
22     LOG ENTRIES
23         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
24         (Transfer(address from, address to, uint256 tokenId))
25     {
26         EMIT XES EVENT ("B1_1")(tokenId)()(
27             "Transfer" as xs:string concept:name,
28             "Completed" as xs:string lifecycle:transition,
29             block.timestamp as xs:date time:timestamp
30         );
31     }
32 }
```

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 Lifecyle 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 "bpaf" (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.

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
10     <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
11     <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
12     <global scope="event">
13         <string key="concept:name" value="default activity"/>
14         <string key="lifecycle:transition" value="Open"/>
15         <date key="time:timestamp" value="2021-07-08T22:51:52+10:00"/>
16     </global>
17     <string key="lifecycle:model" value="bpaf"/>
18     <trace>
19         <event>
20             <string key="concept:name" value="Birth"/>
21             <string key="lifecycle:transition" value="Completed"/>
22             <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
23         </event>
24         <event>
25             <string key="concept:name" value="Transfer"/>
26             <string key="lifecycle:transition" value="Completed"/>
27             <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
28         </event>
29     </trace>
```

Figure 16: XES log extracted with the script from Figure 13, only top part shown, entire file available online[11]

[11] 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.

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

addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");

// Export definition
BLOCKS (6000000) (6000024) {
    LOG ENTRIES
        (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
        (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
    {
        EMIT XES EVENT ("B1_2")(kittyId)()(
            "Birth" as xs:string concept:name,
            "Completed" as xs:string lifecycle:transition,
            block.timestamp as xs:date time:timestamp
        );
    }

    LOG ENTRIES
        (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
        (Transfer(address from, address to, uint256 tokenId))
    {
        EMIT XES EVENT ("B1_2")(tokenId)()(
            "Transfer" as xs:string concept:name,
            "Completed" as xs:string lifecycle:transition,
            block.timestamp as xs:date time:timestamp
        );
    }
}
```

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 attribtutes. 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.

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
10     <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
11     <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
12     <global scope="event">
13         <string key="concept:name" value="default activity"/>
14         <date key="time:timestamp" value="1970-01-01T10:00:00+10:00"/>
15         <string key="lifecycle:transition" value="Completed"/>
16     </global>
17     <string key="lifecycle:model" value="bpaf"/>
18     <trace>
19         <event>
20             <string key="concept:name" value="Birth"/>
21             <string key="lifecycle:transition" value="Completed"/>
22             <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
23         </event>
24         <event>
25             <string key="concept:name" value="Transfer"/>
26             <string key="lifecycle:transition" value="Completed"/>
27             <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
28         </event>
29     </trace>
```

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/nobxgedxidj190v/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`.

```
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 (6000000) (6000024) {
9      LOG ENTRIES
10         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
11         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
12     {
13         EMIT XES EVENT ("B1_3")(kittyId)()(
14             "Birth" as xs:string concept:name,
15             block.timestamp as xs:date time:timestamp
16         );
17     }
18
19     LOG ENTRIES
20         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
21         (Transfer(address from, address to, uint256 tokenId))
22     {
23         EMIT XES EVENT ("B1_3")(tokenId)()(
24             "Transfer" as xs:string concept:name
25         );
26     }
27 }
```

Figure 20: Adding the time:timestamp attribute only for the birth event to the script from Figure 2

```
The script is invalid.
- Error on Ln 13, Col 8: XES compliance problem: The XES standard extension
attribute 'time:timestamp' is used in the script. Thus, all XES event emissions
must also contain the standard extension attribute 'lifecycle:transition' to
comply with XES certification level B1.
- Error on Ln 23, Col 8: XES compliance problem: The XES standard extension
attribute 'time:timestamp' is used in the script. Thus, all XES event emissions
must contain this attribute to comply with XES certification level B1.
- Error on Ln 23, Col 8: XES compliance problem: The XES standard extension
attribute 'time:timestamp' is used in the script. Thus, all XES event emissions
must also contain the standard extension attribute 'lifecycle:transition' to
comply with XES certification level B1.
- 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 21: Validation result for the script from Figure 20

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
7
8  // Export definition
9  BLOCKS (6000000) (6000024) {
10     LOG ENTRIES
11         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
12         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
13     {
14         EMIT XES EVENT ("B1_3")(kittyId)()(
15             "Birth" as xs:string concept:name
16         );
17     }
18
19     LOG ENTRIES
20         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
21         (Transfer(address from, address to, uint256 tokenId))
22     {
23         EMIT XES EVENT ("B1_3")(tokenId)()(
24             "Transfer" as xs:string concept:name
25         );
26     }
27 }
```

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 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 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.
```

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).

```
1   // Preamble
2   connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3   setOutputFolder("./output");
4
5   addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6   addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
7   addGlobalXesEventAttribute("time:timestamp", "xs:date", 1625748712);
8
9   addXesEventClassifier("Event Name", {"concept:name"});
10  addXesEventClassifier("Event Name and Transition", {"concept:name", "lifecycle:transition"});
11
12  // Export definition
13  BLOCKS (6000000) (6000024) {
```

Figure 24: Adding classifiers to the script from Figure 15

```
1   <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9       <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
10      <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
11      <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
12      <global scope="event">
13          <string key="concept:name" value="default activity"/>
14          <string key="lifecycle:transition" value="Open"/>
15          <date key="time:timestamp" value="2021-07-08T22:51:52+10:00"/>
16      </global>
17      <classifier name="Event Name" keys="concept:name"/>
18      <classifier name="Event Name and Transition" keys="concept:name lifecycle:transition"/>
19      <string key="lifecycle:model" value="bpaf"/>
20      <trace>
21          <event>
22              <string key="concept:name" value="Birth"/>
23              <string key="lifecycle:transition" value="Completed"/>
24              <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
25          </event>
26          <event>
27              <string key="concept:name" value="Transfer"/>
28              <string key="lifecycle:transition" value="Completed"/>
29              <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
30          </event>
31      </trace>
```

Figure 25: XES log extracted with the script from Figure 24, only top part shown, entire file available online[13].

[13] 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.

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
7
8  // Export definition
9  BLOCKS (6000000) (6000024) {
10     LOG ENTRIES
11         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
12         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
13     {
14         EMIT XES EVENT ("C1_1")(kittyId)()(
15             "Birth" as xs:string concept:name,
16             kittyId as xs:string org:resource
17         );
18     }
19
20     LOG ENTRIES
21         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
22         (Transfer(address from, address to, uint256 tokenId))
23     {
24         EMIT XES EVENT ("C1_1")(tokenId)()(
25             "Transfer" as xs:string concept:name,
26             tokenId as xs:string org:resource
27         );
28     }
29 }
```

Figure 26: Adding the org:resource attribute to all events in the script from Figure 2

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).

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
10     <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
11     <global scope="event">
12         <string key="concept:name" value="default activity"/>
13         <string key="org:resource" value="default resource"/>
14     </global>
15     <trace>
16         <event>
17             <string key="concept:name" value="Birth"/>
18             <string key="org:resource" value="851836"/>
19         </event>
20         <event>
21             <string key="concept:name" value="Transfer"/>
22             <string key="org:resource" value="851836"/>
23         </event>
24     </trace>
```

Figure 27: XES log extracted with the script from Figure 26, only top part shown, entire file available online[14]

[14] 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 (6000000) (6000024) {
9      LOG ENTRIES
10         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
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         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
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

```
The script is invalid.
- 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.
- Warning on Ln 2, Col 0: XES compliance problem: The XES standard extension
attribute 'org:resource' must be specified as a global event attribute, in order
to comply with XES certification level C1. 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 29: Validation result for the script from Figure 28

```
// 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
        (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
        (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
    {
        EMIT XES EVENT ("C1_2")(kittyId)()(
            "Birth" as xs:string concept:name
        );
    }

    LOG ENTRIES
        (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
        (Transfer(address from, address to, uint256 tokenId))
    {
        EMIT XES EVENT ("C1_2")(tokenId)()(
            "Transfer" as xs:string concept:name
        );
    }
}
```

Figure 30: Adding the org:resource 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 '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.
```

Figure 31: Validation result for the script from Figure 30

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).

```
1   // Preamble
2   connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3   setOutputFolder("./output");
4
5   addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6   addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
7
8   addXesEventClassifier("Event Name", {"concept:name"});
9   addXesEventClassifier("Event Name and Resource", {"concept:name", "org:resource"});
10
11  // Export definition
12  BLOCKS (6000000) (6000024) {
```

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

```
1   <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9       <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
10      <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
11      <global scope="event">
12          <string key="concept:name" value="default activity"/>
13          <string key="org:resource" value="default resource"/>
14      </global>
15      <classifier name="Event Name" keys="concept:name"/>
16      <classifier name="Event Name and Resource" keys="concept:name org:resource"/>
17      <trace>
18          <event>
19              <string key="concept:name" value="Birth"/>
20              <string key="org:resource" value="851836"/>
21          </event>
22          <event>
23              <string key="concept:name" value="Transfer"/>
24              <string key="org:resource" value="851836"/>
25          </event>
26      </trace>
```

Figure 33: XES log extracted with the script from Figure 32, only top part shown, entire file available online[15]

[15] https://www.dropbox.com/s/unffbqqqj6x9arv/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 18 and 29). 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 executionss on Ethereum networks. Lastly, the `cost:total` attribute is also defined as a global event attribute (line 6).

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("cost:total", "xs:float", 0);
7
8  // Export definition
9  BLOCKS (6000000) (6000024) {
10     LOG ENTRIES
11         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
12         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
13     {
14         int gasCost = multiply(tx.gasUsed, tx.gasPrice);
15         EMIT XES EVENT ("D1_1")(kittyId)()(
16             "Birth" as xs:string concept:name,
17             "CryptoKitty" as xs:string org:role,
18             gasCost as xs:float cost:total
19         );
20     }
21
22     LOG ENTRIES
23         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
24         (Transfer(address from, address to, uint256 tokenId))
25     {
26         int gasCost = multiply(tx.gasUsed, tx.gasPrice);
27         EMIT XES EVENT ("D1_1")(tokenId)()(
28             "Transfer" as xs:string concept:name,
29             gasCost as xs:float cost:total
30         );
31     }
32 }
```

Figure 34: Adding org:role and cost:total attributes to the script from Figure 2

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).

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
10     <extension name="Cost" prefix="cost" uri="http://www.xes-standard.org/cost.xesext"/>
11     <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
12     <global scope="event">
13         <string key="concept:name" value="default activity"/>
14         <float key="cost:total" value="0.0"/>
15     </global>
16     <trace>
17         <event>
18             <string key="concept:name" value="Birth"/>
19             <string key="org:role" value="CryptoKitty"/>
20             <float key="cost:total" value="8.26430576E15"/>
21         </event>
22         <event>
23             <string key="concept:name" value="Transfer"/>
24             <float key="cost:total" value="8.26430576E15"/>
25         </event>
26     </trace>
```

Figure 35: XES log extracted with the script from Figure 34, only top part shown, entire file available online[16]

[16] 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.

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("cost:total", "xs:float", 0);
7
8  // Export definition
9  BLOCKS (6000000) (6000024) {
10     LOG ENTRIES
11         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
12         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
13     {
14         int gasCost = multiply(tx.gasUsed, tx.gasPrice);
15         EMIT XES EVENT ("D1_2")(kittyId)()(
16             "Birth" as xs:string concept:name,
17             "CryptoKitty" as xs:string org:role,
18             gasCost as xs:float cost:total
19         );
20     }
21
22     LOG ENTRIES
23         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
24         (Transfer(address from, address to, uint256 tokenId))
25     {
26         EMIT XES EVENT ("D1_2")(tokenId)()(
27             "Transfer" as xs:string concept:name
28         );
29     }
30 }
```

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

```
The script is invalid.
- Error on Ln 26, Col 8: XES compliance problem: The XES event attribute
'cost:total' was defined globally. It must thus be added to all XES event
emissions to comply with XES certification level D1.
```

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.

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("cost:tota", "xs:float", 0);
7  addGlobalXesEventAttribute("org:role", "xs:boolean", false);
8
9  // Export definition
10 BLOCKS (6000000) (6000024) {
11     LOG ENTRIES
12         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
13         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
14     {
15         int gasCost = multiply(tx.gasUsed, tx.gasPrice);
16         EMIT XES EVENT ("D1_1")(kittyId)()(
17             "Birth" as xs:string concept:name,
18             0 as xs:int org:role,
19             gasCost as xs:float cost:total
20         );
21     }
22
23     LOG ENTRIES
24         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
25         (Transfer(address from, address to, uint256 tokenId))
26     {
27         int gasCost = multiply(tx.gasUsed, tx.gasPrice);
28         EMIT XES EVENT ("D1_1")(tokenId)()(
29             "Transfer" as xs:string concept:name,
30             "CryptoKitty" as xs:string org:role,
31             gasCost as xs:float cst:total
32         );
33     }
34 }
```

Figure 38: Incorrect usage of standard extension attributes

```
The script is invalid.
- Error on Ln 6, Col 0: XES compliance problem: The XES standard extension
attribute 'cost:tota' does not exist.
- Error on Ln 7, Col 0: XES compliance problem: The type 'xs:boolean' was
specified for attribute 'org:role'. This violates the definition from the
standard extension, where the type 'xs:string' was specified.
- Error on Ln 18, Col 24: XES compliance problem: The type 'xs:int' was
specified for attribute 'org:role'. This violates the definition from the
standard extension, where the type 'xs:string' was specified.
- Error on Ln 31, Col 32: XES compliance problem: The XES standard extension
attribute 'cst:total' does not exist.
```

Figure 39: Validation result for the script from Figure 38

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).

```
1   // Preamble
2   connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3   setOutputFolder("./output");
4
5   addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6   addGlobalXesEventAttribute("cost:total", "xs:float", 0);
7
8   addXesEventClassifier("Event Name", {"concept:name"});
9   addXesEventClassifier("Event Name and Cost", {"concept:name", "cost:total"});
10
11  // Export definition
12  BLOCKS (6000000) (6000024) {
```

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

```
1   <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9       <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
10      <extension name="Cost" prefix="cost" uri="http://www.xes-standard.org/cost.xesext"/>
11      <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
12      <global scope="event">
13          <string key="concept:name" value="default activity"/>
14          <float key="cost:total" value="0.0"/>
15      </global>
16      <classifier name="Event Name" keys="concept:name"/>
17      <classifier name="Event Name and Cost" keys="concept:name cost:total"/>
18      <trace>
19          <event>
20              <string key="concept:name" value="Birth"/>
21              <string key="org:role" value="CryptoKitty"/>
22              <float key="cost:total" value="8.26430576E15"/>
23          </event>
24          <event>
25              <string key="concept:name" value="Transfer"/>
26              <float key="cost:total" value="8.26430576E15"/>
27          </event>
28      </trace>
```

Figure 41: XES log extracted with the script from Figure 40, only top part shown, entire file available online[17]

[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.

```
1   // Preamble
2   connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3   setOutputFolder("./output");
4
5   addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6   addGlobalXesEventAttribute("cost:total", "xs:float", 0);
7
8   addXesEventClassifier("Event Name", {"concept:name"});
9   addXesEventClassifier("Event Name and Cost", {"concept:name", "org:role"});
10
11  // Export definition
12  BLOCKS (6000000) (6000024) {
```

Figure 42: Removing global event attributes from the script in Figure 40

```
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.
```

Figure 43: Validation result for the script from Figure 42

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).

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
7  addGlobalXesEventAttribute("time:timestamp", "xs:date", 1625748712);
8  addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
9  addGlobalXesEventAttribute("txRecipient", "xs:string", "0x06012c8cf97BEaD5deAe237070F9587f8E7A266d");
10
11 // Export definition
12 BLOCKS (6000000) (6000024) {
13     LOG ENTRIES
14         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
15         (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
16     {
17         EMIT XES EVENT ("X1_1")(kittyId)()(
18             "Birth" as xs:string concept:name,
19             "Completed" as xs:string lifecycle:transition,
20             block.timestamp as xs:date time:timestamp,
21             kittyId as xs:string org:resource,
22             block.number as xs:int blockNumber,
23             tx.hash as xs:string txHash,
24             tx.from as xs:string txSender,
25             tx.to as xs:string txRecipient
26         );
27     }
28
29     LOG ENTRIES
30         (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
31         (Transfer(address from, address to, uint256 tokenId))
32     {
33         EMIT XES EVENT ("X1_1")(tokenId)()(
34             "Transfer" as xs:string concept:name,
35             "Completed" as xs:string lifecycle:transition,
36             block.timestamp as xs:date time:timestamp,
37             tokenId as xs:string org:resource,
38             block.number as xs:int blockNumber,
39             tx.hash as xs:string txHash,
40             tx.from as xs:string txSender,
41             tx.to as xs:string txRecipient
42         );
43     }
44 }
```

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).

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
10     <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
11     <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
12     <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
13     <global scope="event">
14         <string key="concept:name" value="default activity"/>
15         <string key="lifecycle:transition" value="Open"/>
16         <date key="time:timestamp" value="2021-07-08T22:51:52+10:00"/>
17         <string key="org:resource" value="default resource"/>
18         <string key="txRecipient" value="0x06012c8cf97BEaD5deAe237070F9587f8E7A266d"/>
19     </global>
20     <string key="lifecycle:model" value="bpaf"/>
21     <trace>
22         <event>
23             <string key="txSender" value="0x2a9847093ad514639e8cdec960b5e51686960291"/>
24             <string key="org:resource" value="851836"/>
25             <string key="concept:name" value="Birth"/>
26             <string key="txRecipient" value="0xc7ed8919c70dd8ccf1a57c0ed75b25ceb2dd22d1"/>
27             <int key="blockNumber" value="6000000"/>
28             <string key="lifecycle:transition" value="Completed"/>
29             <string key="txHash" value="0xa8f2cf69e302da6c8100b80298ed77c37b6e75eed1177ca22acd5772c9fb9876"/>
30             <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
31         </event>
32         <event>
33             <string key="txSender" value="0x2a9847093ad514639e8cdec960b5e51686960291"/>
34             <string key="org:resource" value="851836"/>
35             <string key="concept:name" value="Transfer"/>
36             <string key="txRecipient" value="0xc7ed8919c70dd8ccf1a57c0ed75b25ceb2dd22d1"/>
37             <int key="blockNumber" value="6000000"/>
38             <string key="lifecycle:transition" value="Completed"/>
39             <string key="txHash" value="0xa8f2cf69e302da6c8100b80298ed77c37b6e75eed1177ca22acd5772c9fb9876"/>
40             <date key="time:timestamp" value="2018-07-21T06:29:24+10:00"/>
41         </event>
42     </trace>
```

Figure 45: XES log extracted with the script from Figure 44, only top part shown, entire file available online[18]

[18] 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.

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

addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
addGlobalXesEventAttribute("time:timestamp", "xs:date", 1625748712);
addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
addGlobalXesEventAttribute("txRecipient", "xs:string", "0x06012c8cf97BEaD5deAe237070F9587f8E7A266d");

// Export definition
BLOCKS (6000000) (6000024) {
    LOG ENTRIES
        (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
        (Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes))
    {
        EMIT XES EVENT ("X1_1")(kittyId)()(
            "Birth" as xs:string concept:name,
            "Completed" as xs:string lifecycle:transition,
            block.timestamp as xs:date time:timestamp,
            kittyId as xs:string org:resource,
            block.number as xs:int blockNumber,
            tx.hash as xs:string txHash,
            tx.from as xs:int txSender
        );
    }

    LOG ENTRIES
        (0x06012c8cf97BEaD5deAe237070F9587f8E7A266d)
        (Transfer(address from, address to, uint256 tokenId))
    {
        EMIT XES EVENT ("X1_1")(tokenId)()(
            "Transfer" as xs:string concept:name,
            "Completed" as xs:string lifecycle:transition,
            block.timestamp as xs:date time:timestamp,
            tokenId as xs:string org:resource,
            block.number as xs:string blockNumber,
            tx.hash as xs:string txHash,
            tx.gas as xs:int txSender,
            tx.transactionIndex as xs:int txRecipient
        );
    }
}
```

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).

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
7  addGlobalXesEventAttribute("time:timestamp", "xs:date", 1625748712);
8  addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
9  addGlobalXesEventAttribute("txRecipient", "xs:string", "0x06012c8cf97BEaD5deAe237070F9587f8E7A266d");
10
11 addXesEventClassifier("Transaction Recipient", { "txRecipient" });
12
13 // Export definition
14 BLOCKS (6000000) (6000024) {
```

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

```
1  <?xml 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 xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
9      <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
10     <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
11     <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
12     <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
13     <global scope="event">
14         <string key="concept:name" value="default activity"/>
15         <string key="lifecycle:transition" value="Open"/>
16         <date key="time:timestamp" value="2021-07-08T22:51:52+10:00"/>
17         <string key="org:resource" value="default resource"/>
18         <string key="txRecipient" value="0x06012c8cf97BEaD5deAe237070F9587f8E7A266d"/>
19     </global>
20     <classifier name="Transaction Recipient" keys="txRecipient"/>
21     <string key="lifecycle:model" value="bpaf"/>
22     <trace>
23         <event>
24             <string key="txSender" value="0x2a9847093ad514639e8cdec960b5e51686960291"/>
25             <string key="org:resource" value="851836"/>
```

Figure 49: XES log extracted with the script from Figure 48, only top part shown, entire file available online[19]

[19] 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.

```
1  // Preamble
2  connectIpc("/data2/geth-archive/chaindata/geth.ipc");
3  setOutputFolder("./output");
4
5  addGlobalXesEventAttribute("concept:name", "xs:string", "default activity");
6  addGlobalXesEventAttribute("lifecycle:transition", "xs:string", "Open");
7  addGlobalXesEventAttribute("time:timestamp", "xs:date", 1625748712);
8  addGlobalXesEventAttribute("org:resource", "xs:string", "default resource");
9  addGlobalXesEventAttribute("txRecipient", "xs:string", "0x06012c8cf97BEaD5deAe237070F9587f8E7A266d");
10
11 addXesEventClassifier("Transaction Recipient", { "txRecipient" });
12 addXesEventClassifier("Transaction Sender", {"txSender"});
13
14 // Export definition
15 BLOCKS (6000000) (6000024) {
```

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

```
The script is invalid.
- Error on Ln 12, Col 0: 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

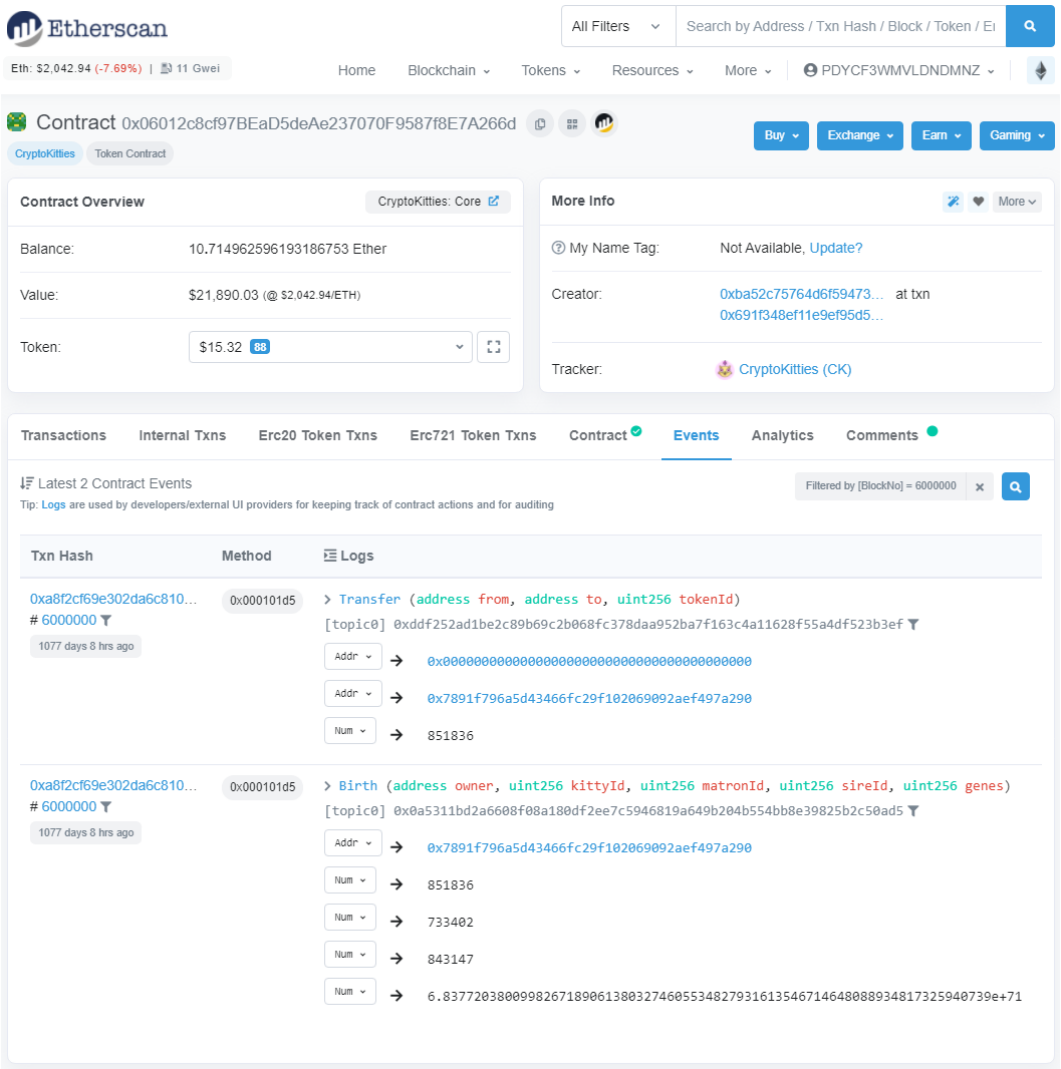


Figure 52: One birth and one transfer log entry were created by the CryptoKitties smart contract in block 6,000,000

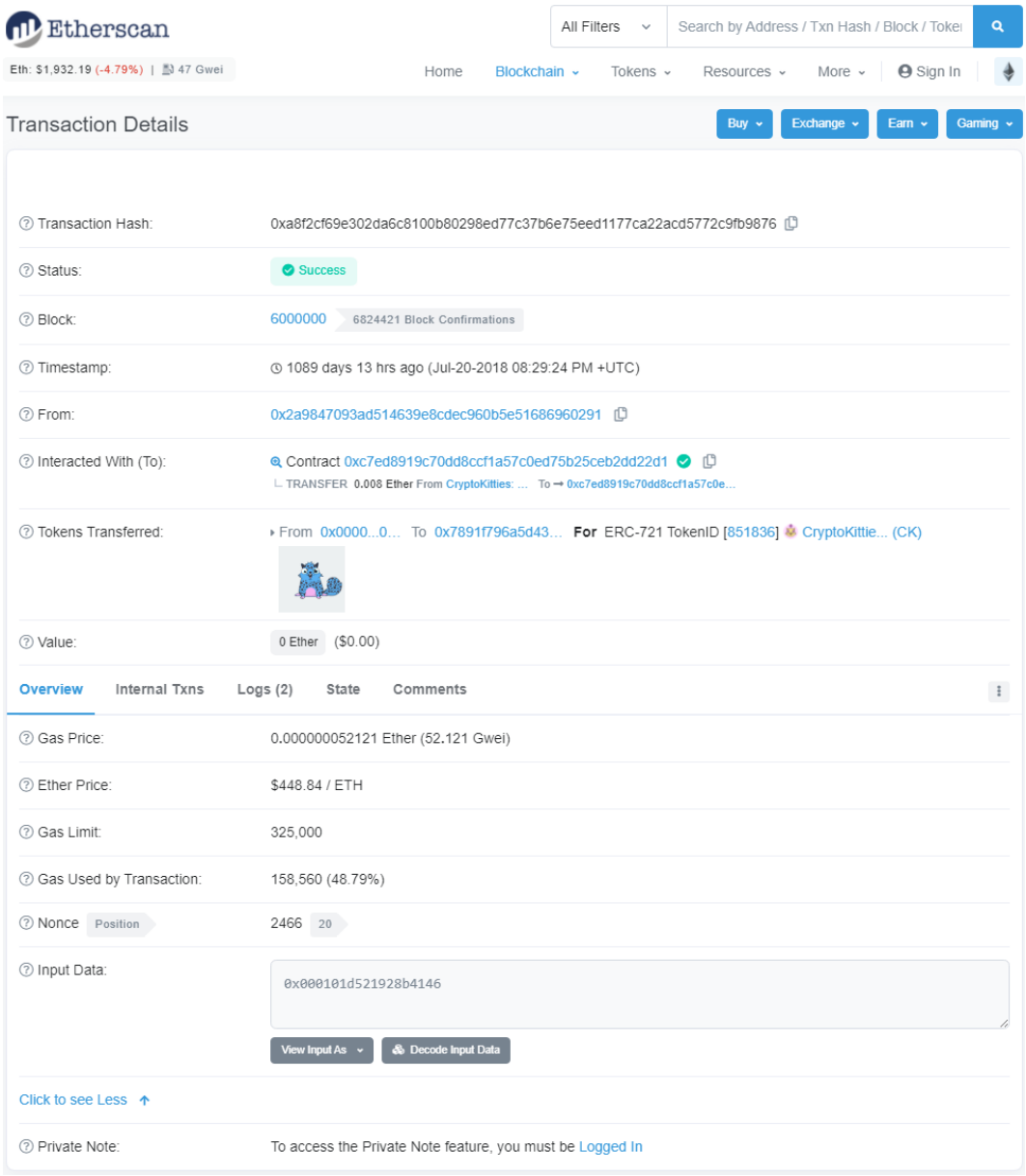


Figure 53: Details for transaction 0xa8f2cf69e302da6c8100b80298ed77c37b6e75eed1177ca22acd5772c9fb9876 that included one birth and one transfer log entry from the CryptoKitties smart contract in block 6,000,001

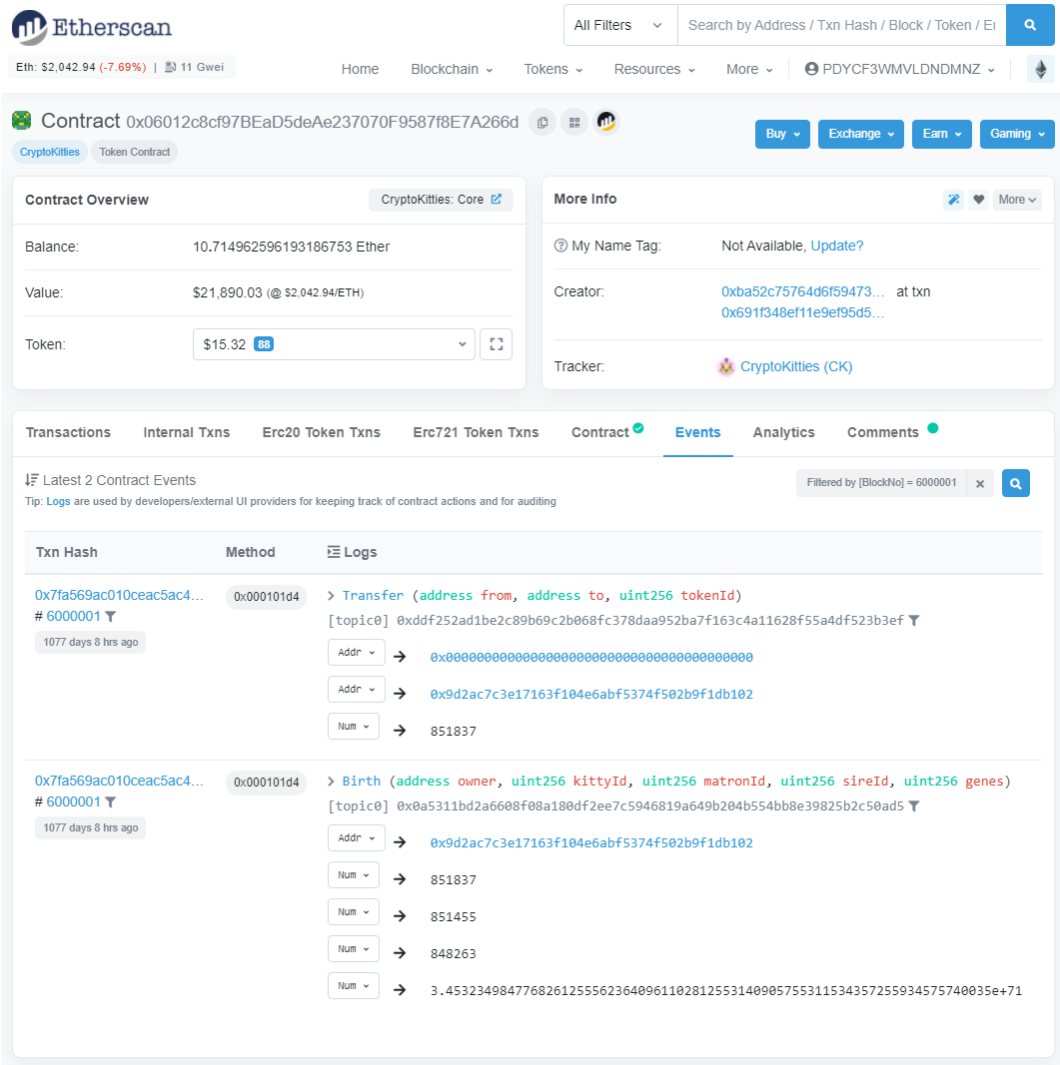


Figure 54: One birth and one transfer log entry were created by the CryptoKitties smart contract in block 6,000,001

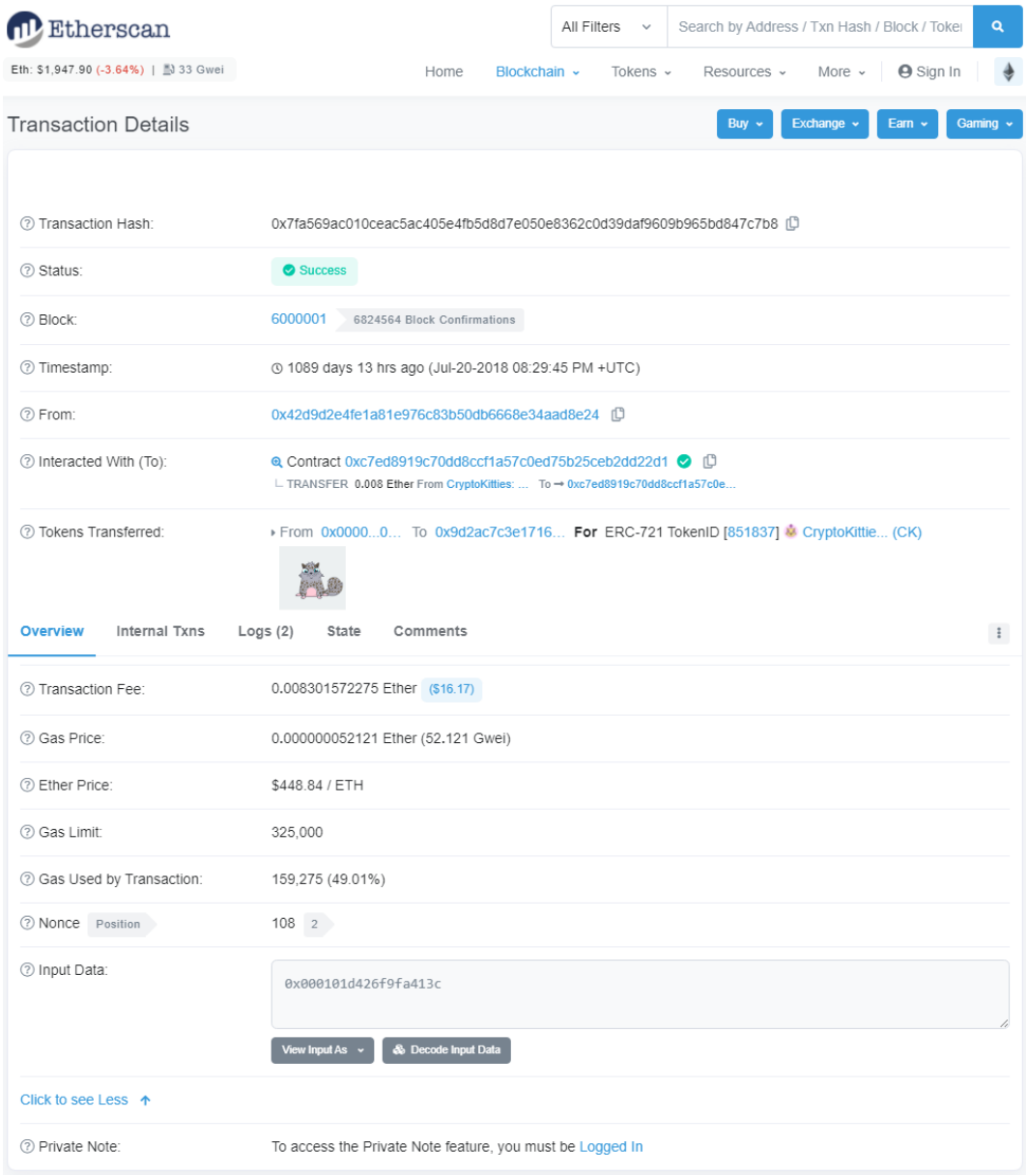


Figure 55: Details for transaction 0x7fa569ac010ceac5ac405e4fb5d8d7e050e8362c0d39daf9609b965bd847c7b8 that included one birth and one transfer log entry from the CryptoKitties smart contract in block 6,000,001

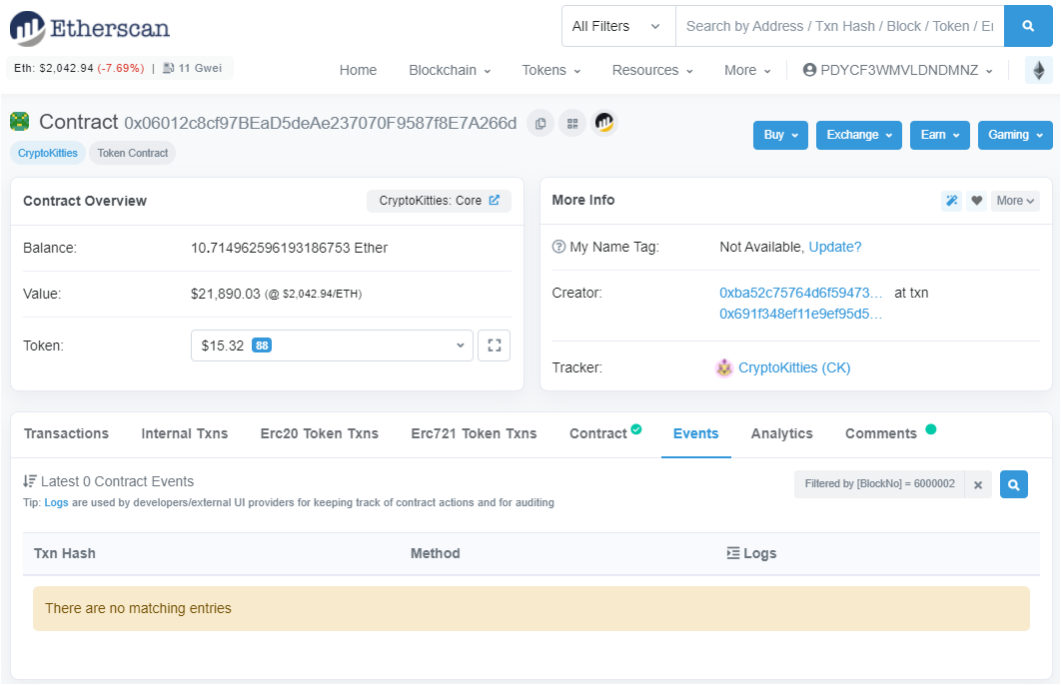


Figure 56: No log entries were created by the CryptoKitties smart contract in block 6,000,002

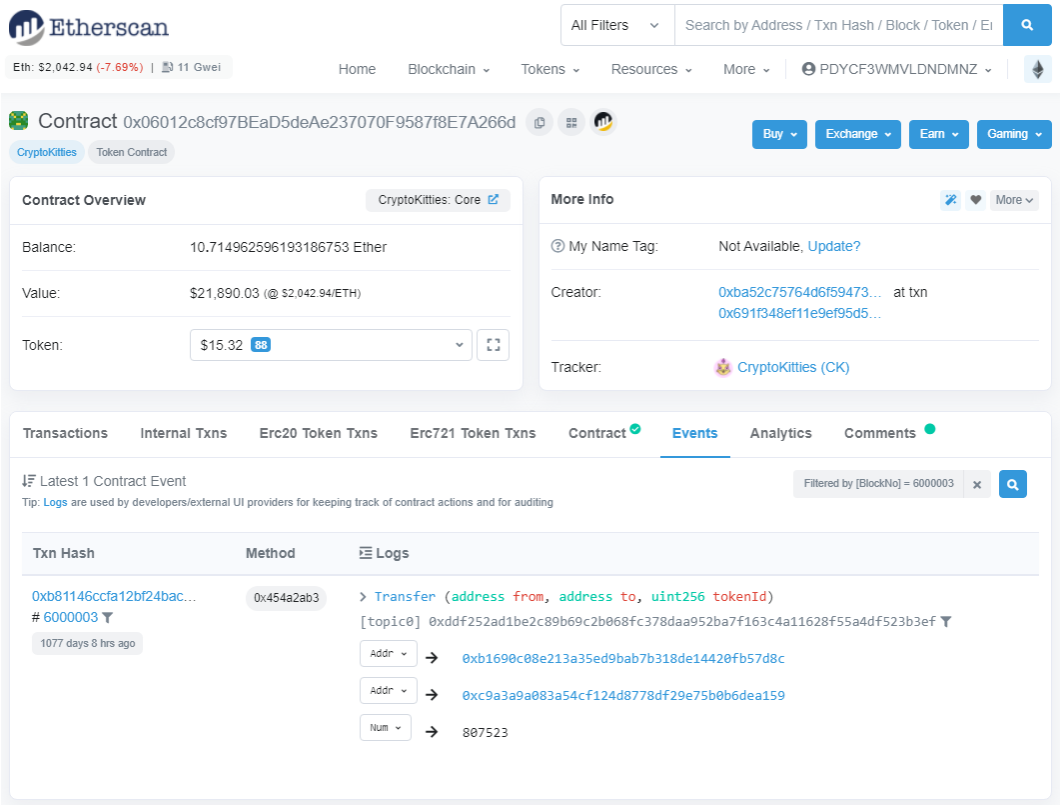


Figure 57: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,003

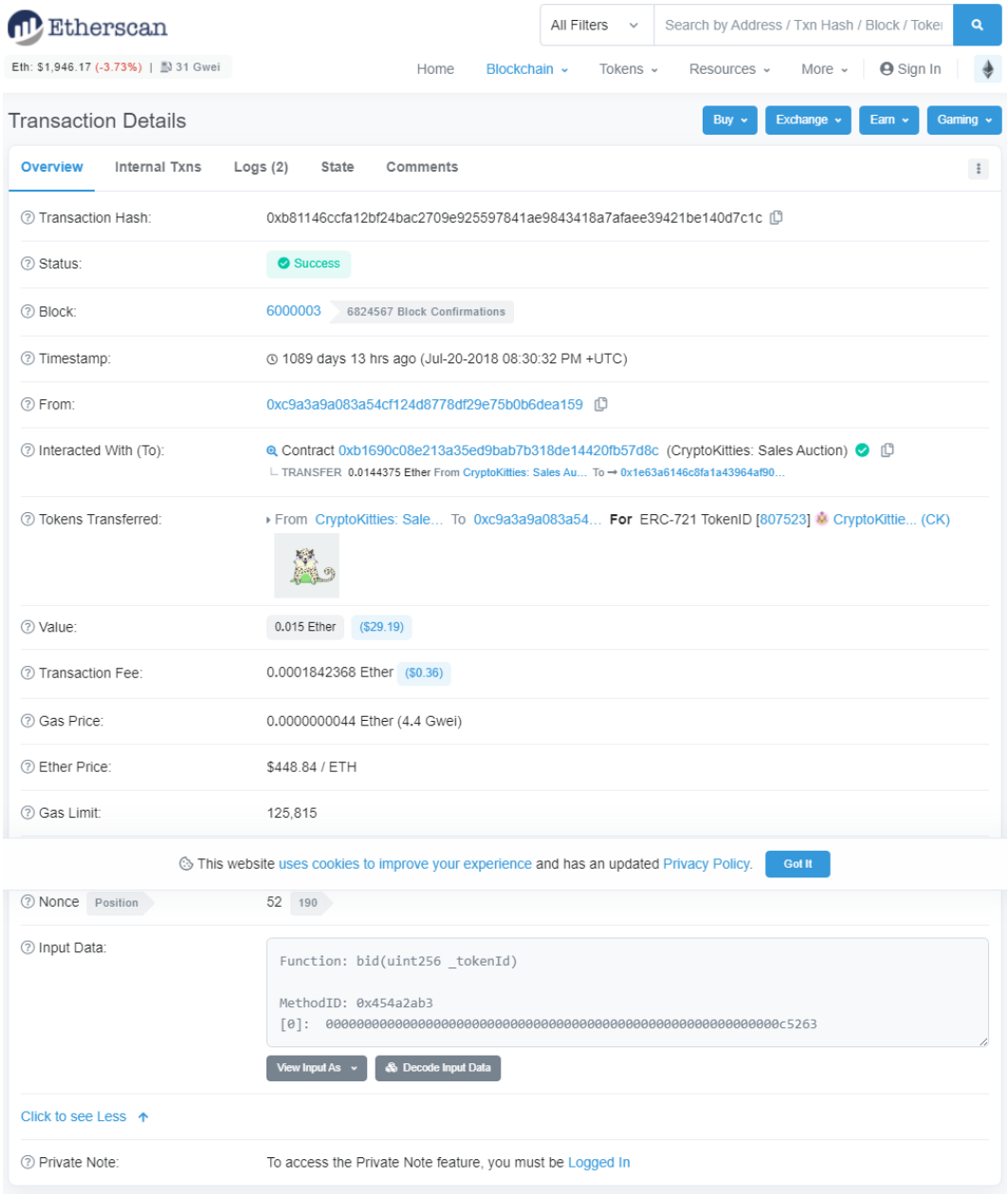


Figure 58: Details for transaction 0xb81146ccfa12bf24bac2709e925597841ae9843418a7afaee39421be140d7c1c that included one transfer log entry from the CryptoKitties smart contract in block 6,000,003

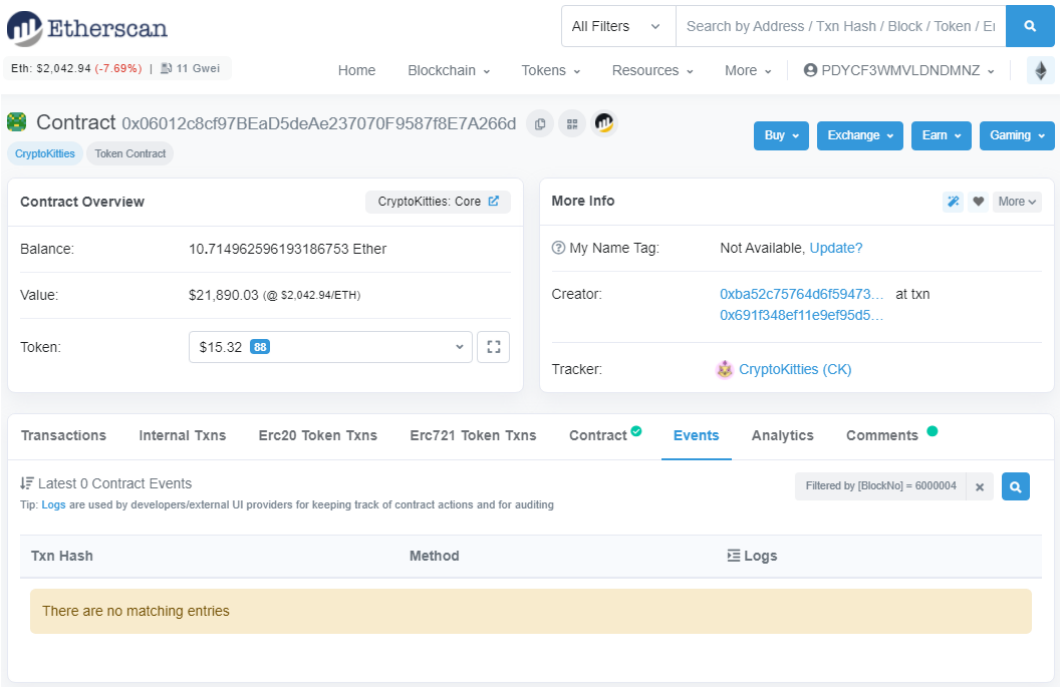


Figure 59: No log entries were created by the CryptoKitties smart contract in block 6,000,004

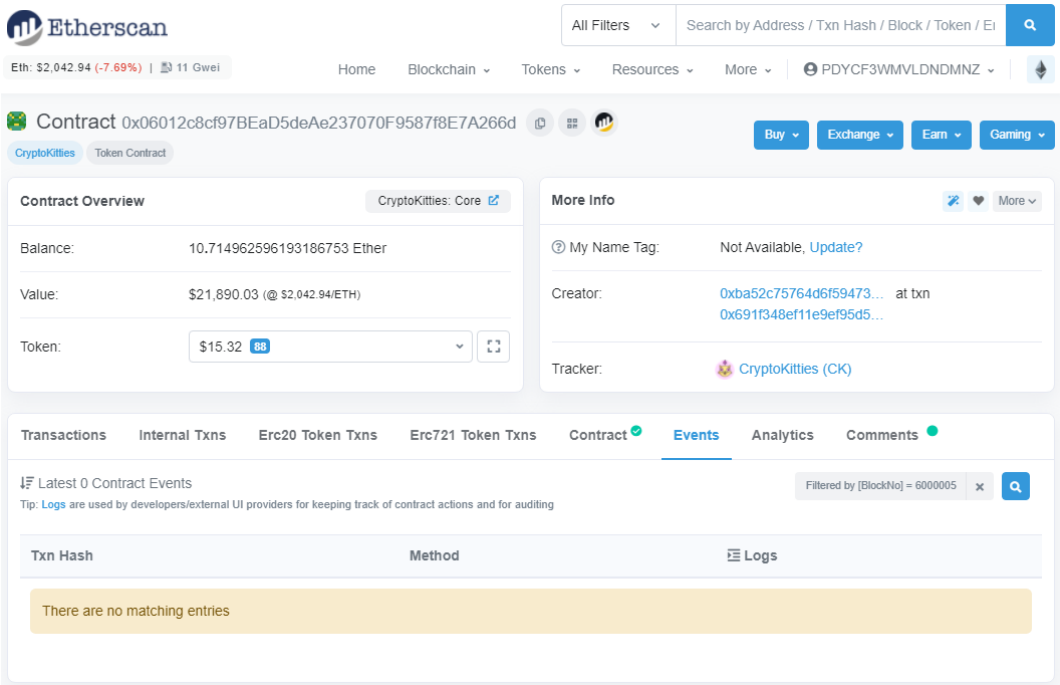


Figure 60: No log entries were created by the CryptoKitties smart contract in block 6,000,005

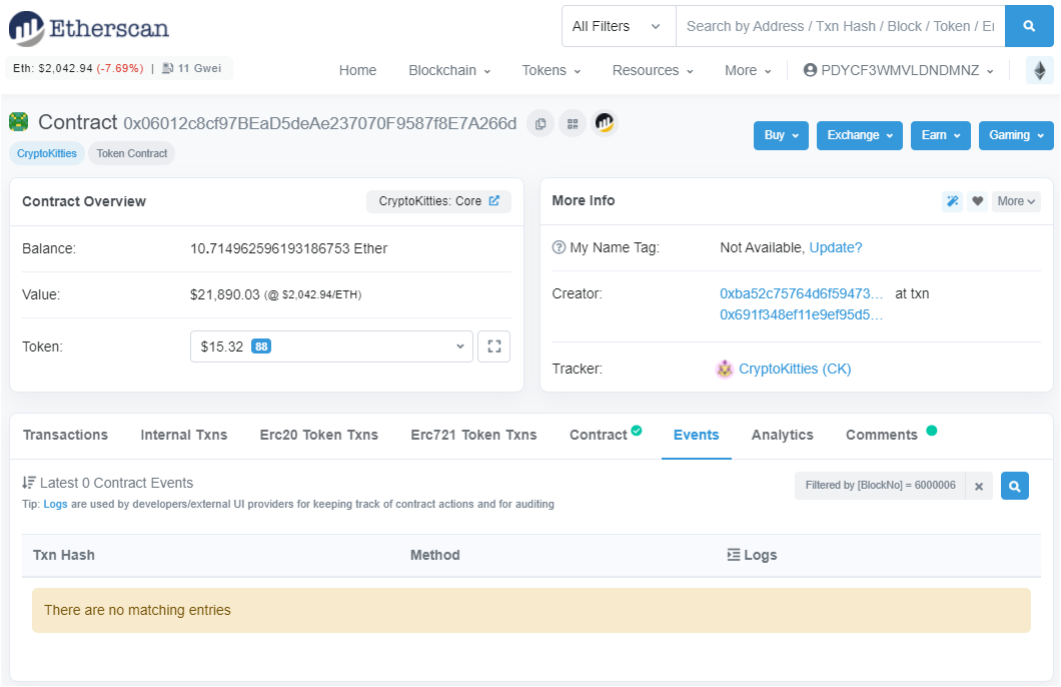


Figure 61: No log entries were created by the CryptoKitties smart contract in block 6,000,006

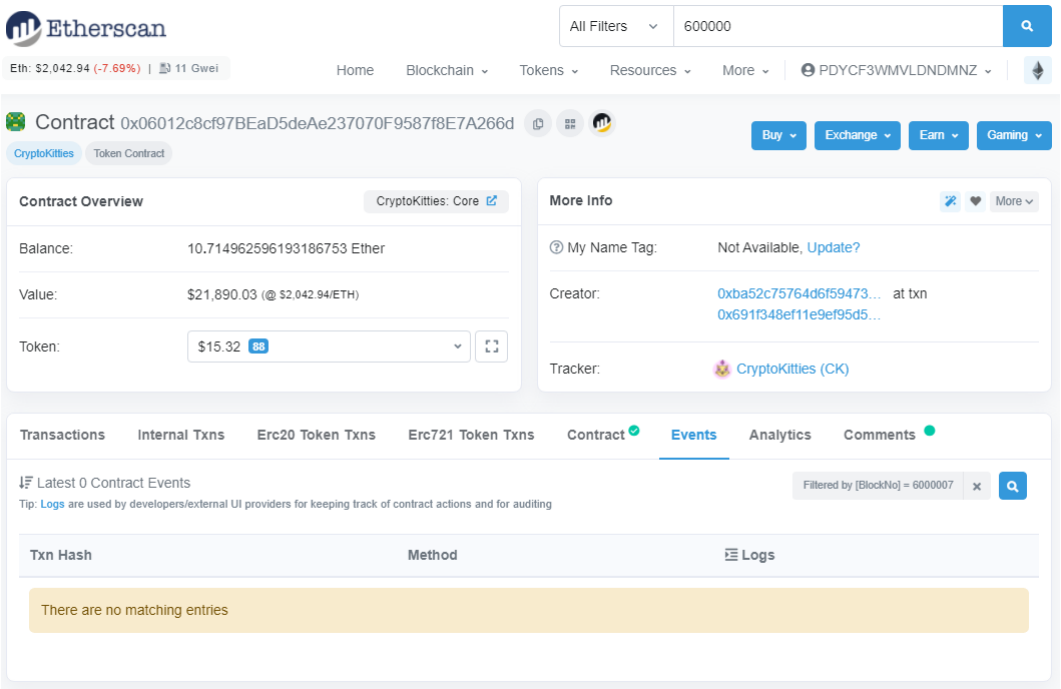


Figure 62: No log entries were created by the CryptoKitties smart contract in block 6,000,007

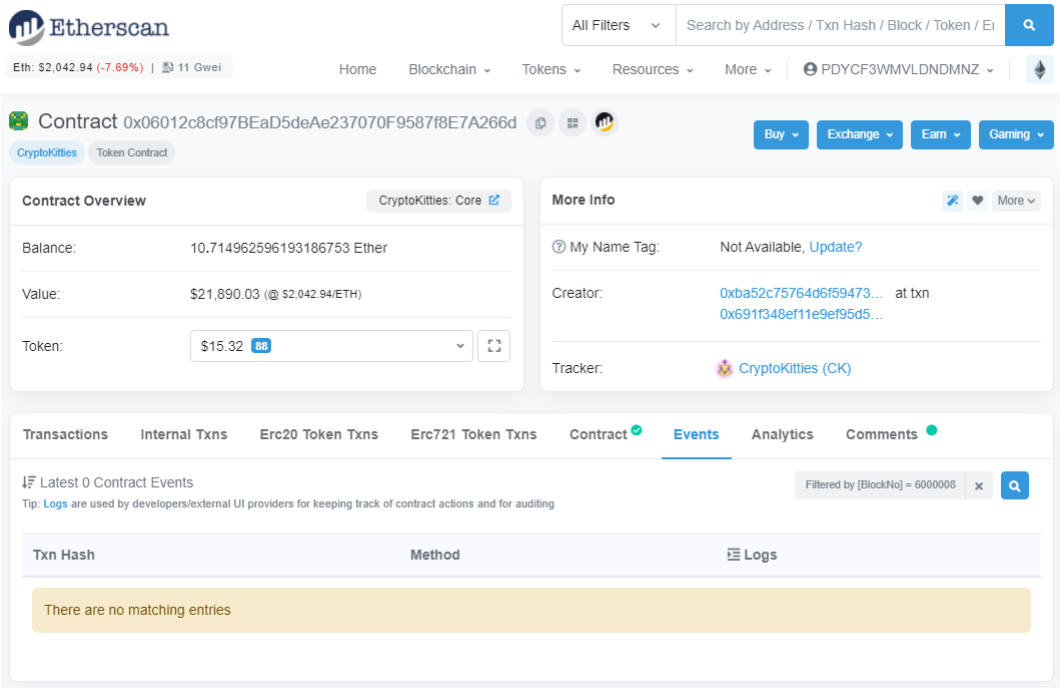


Figure 63: No log entries were created by the CryptoKitties smart contract in block 6,000,008

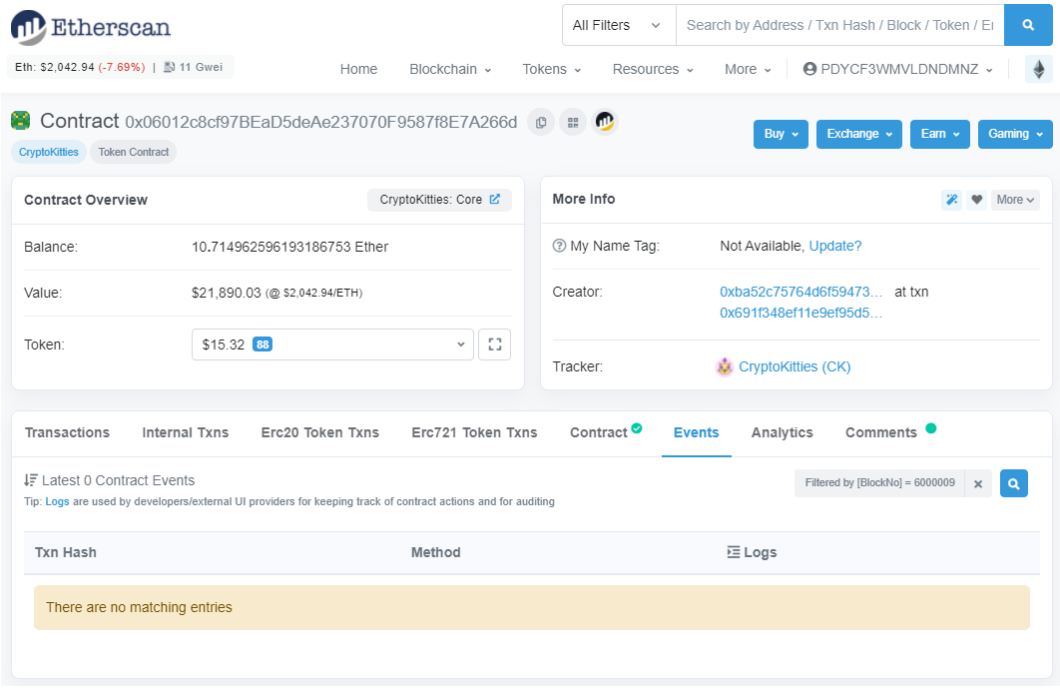


Figure 64: No log entries were created by the CryptoKitties smart contract in block 6,000,009

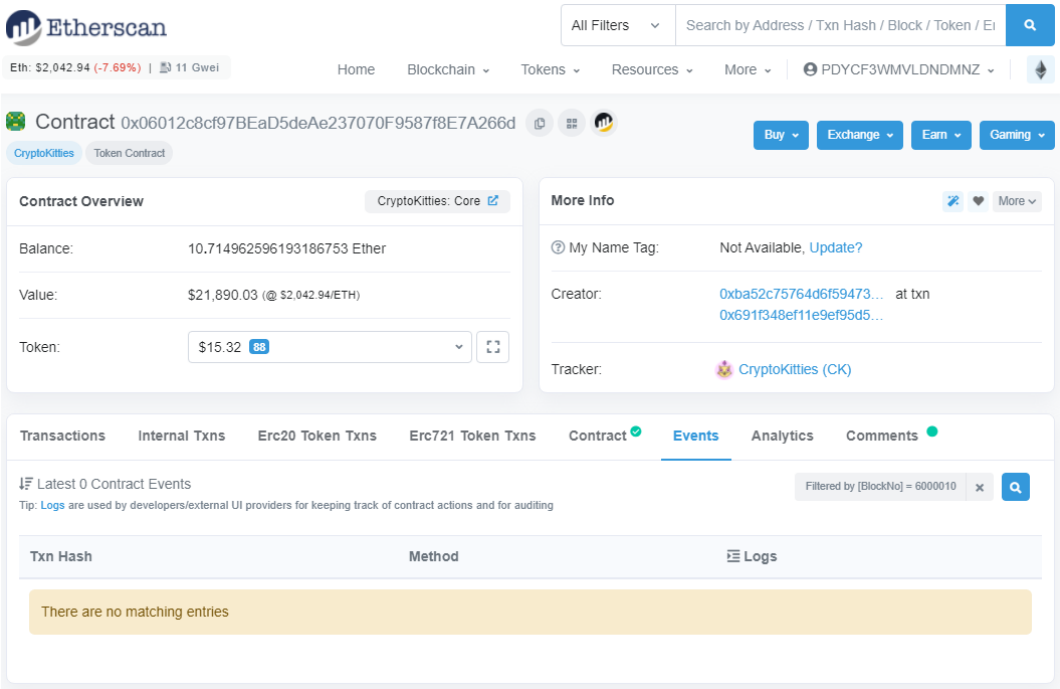


Figure 65: No log entries were created by the CryptoKitties smart contract in block 6,000,010

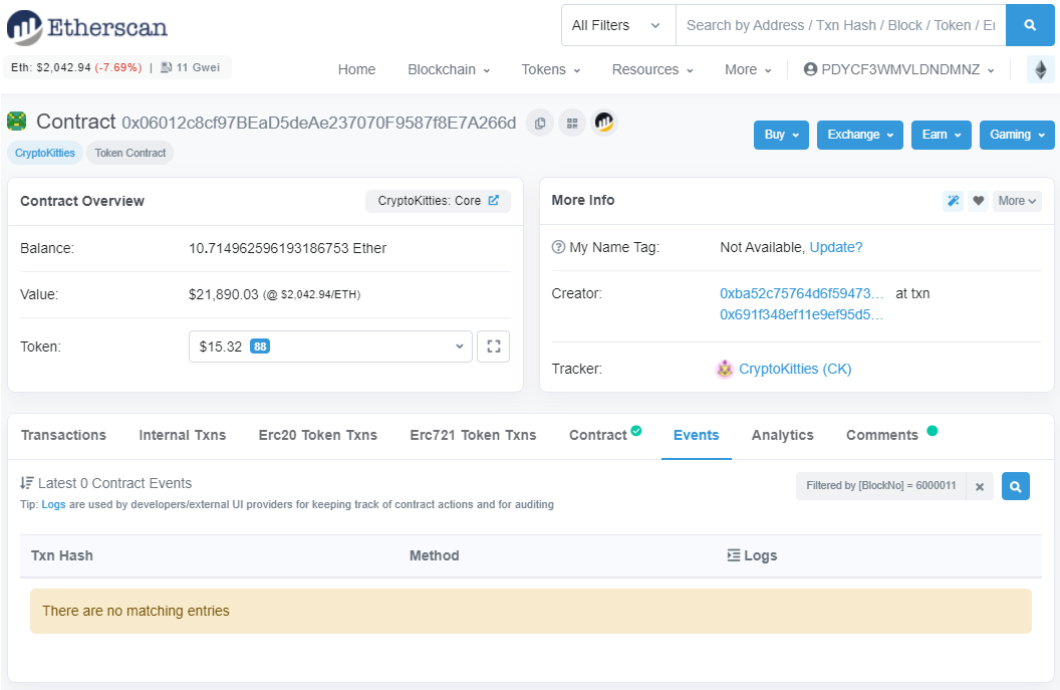


Figure 66: No log entries were created by the CryptoKitties smart contract in block 6,000,011

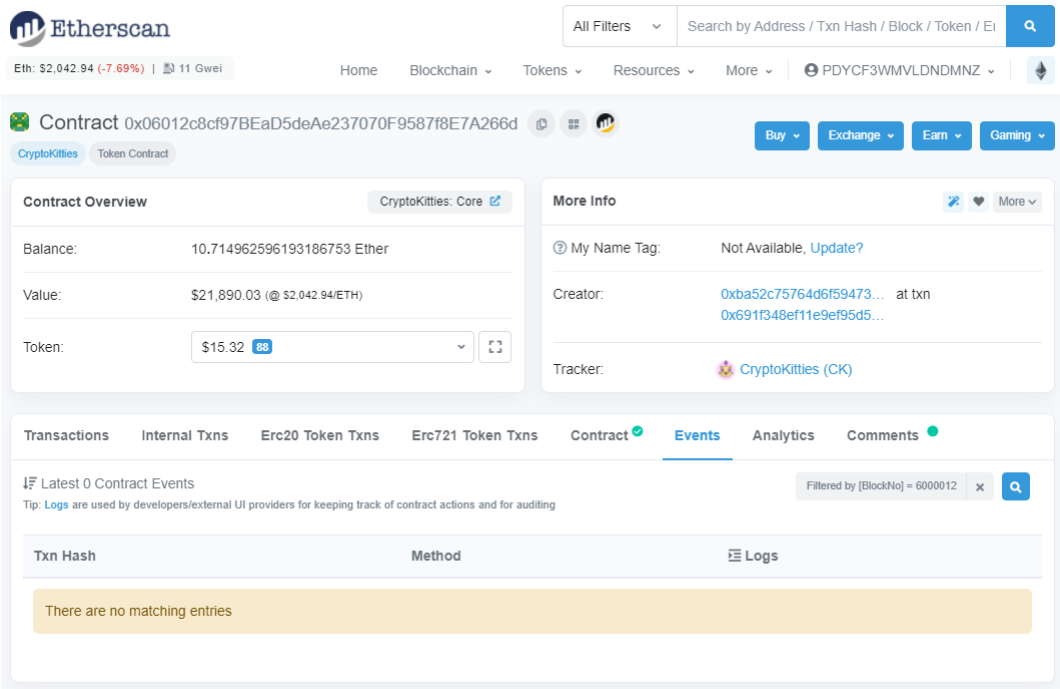


Figure 67: No log entries were created by the CryptoKitties smart contract in block 6,000,012

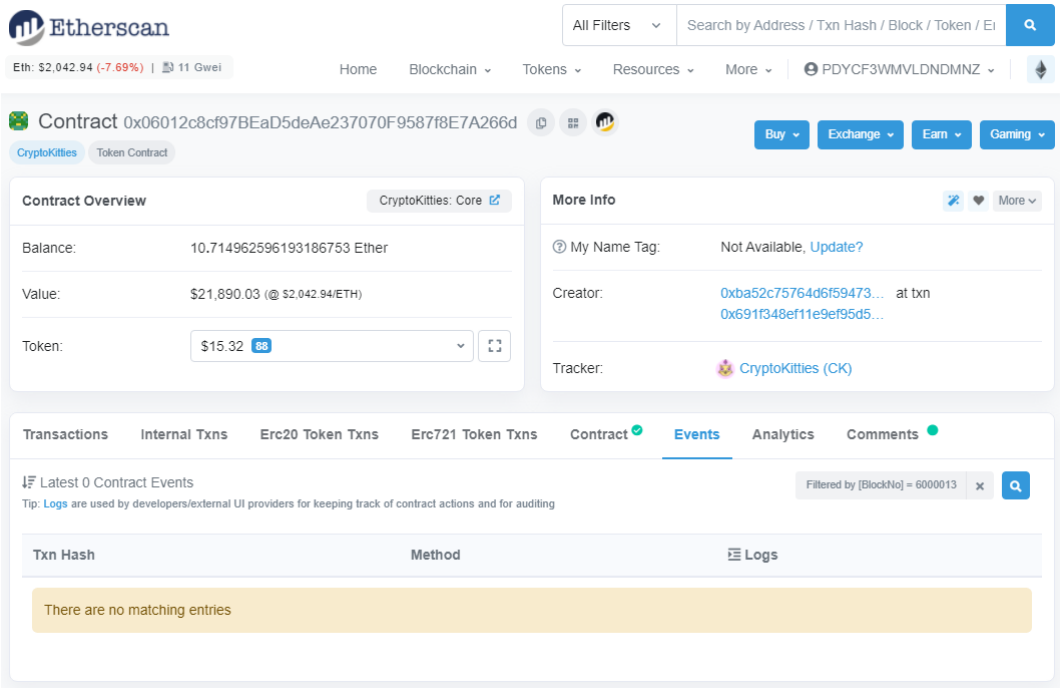


Figure 68: No log entries were created by the CryptoKitties smart contract in block 6,000,013

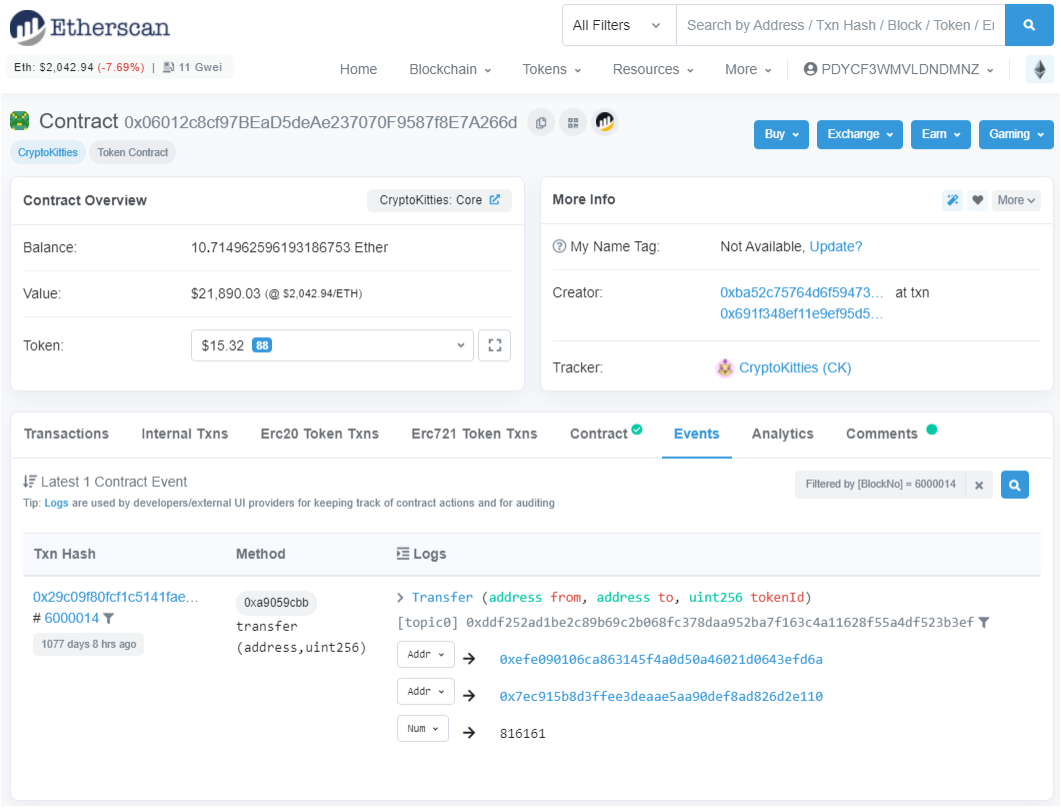


Figure 69: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,014

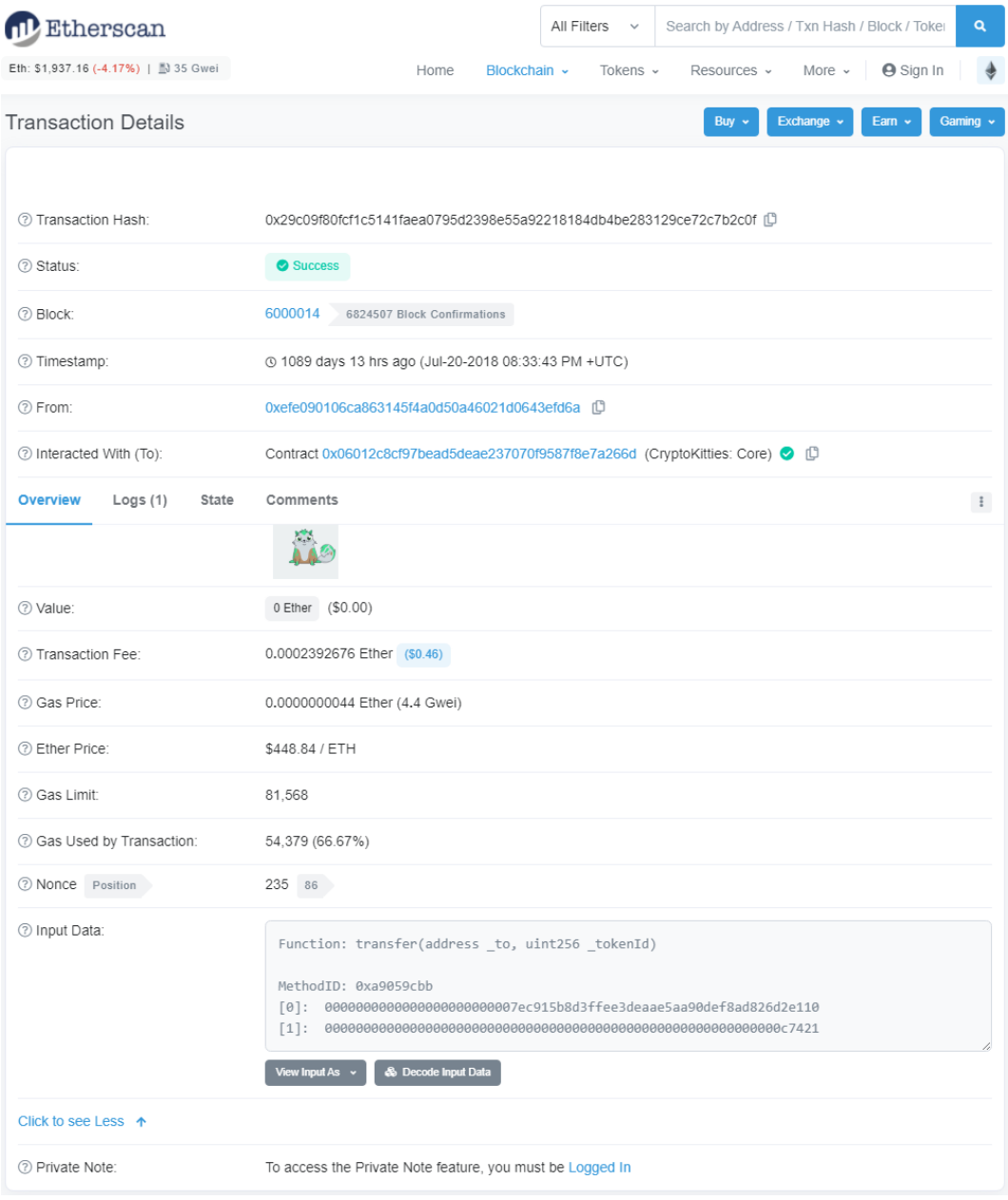


Figure 70: Details for transaction 0x29c09f80fcf1c5141faea0795d2398e55a92218184db4be283129ce72c7b2c0f that included one transfer log entry from the CryptoKitties smart contract in block 6,000,014

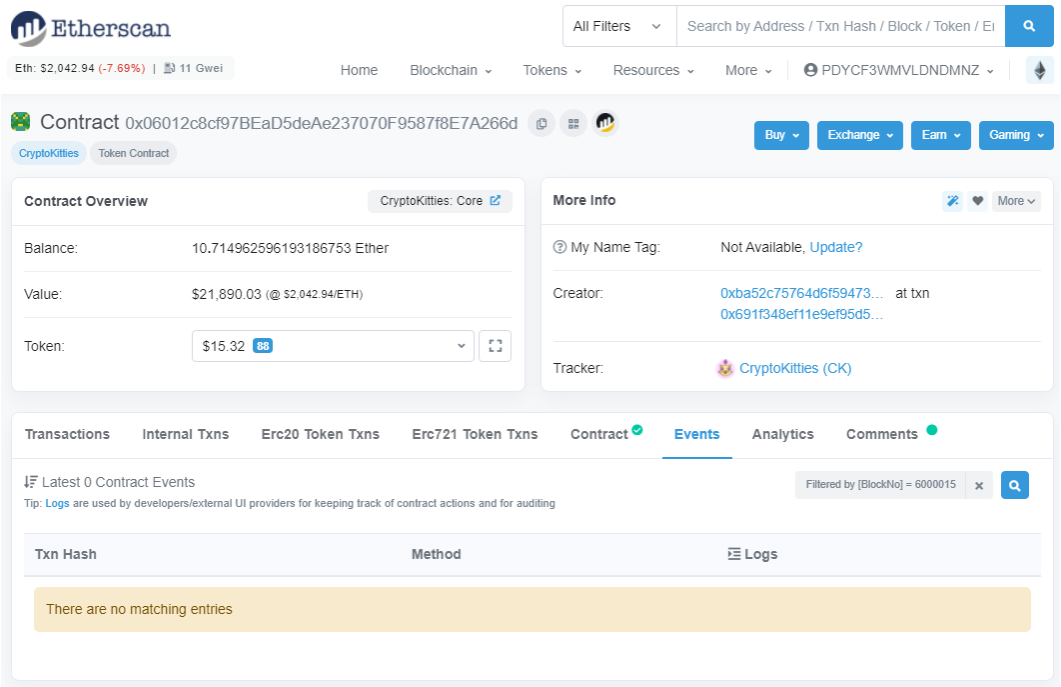


Figure 71: No log entries were created by the CryptoKitties smart contract in block 6,000,015

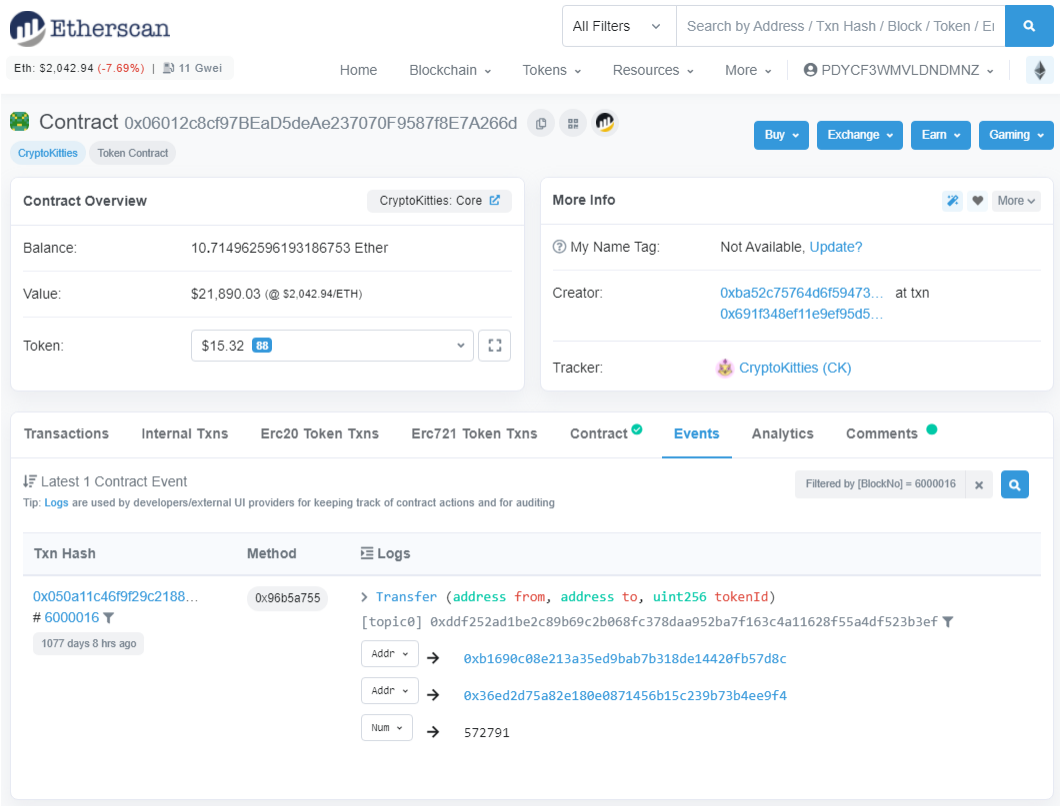


Figure 72: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,016

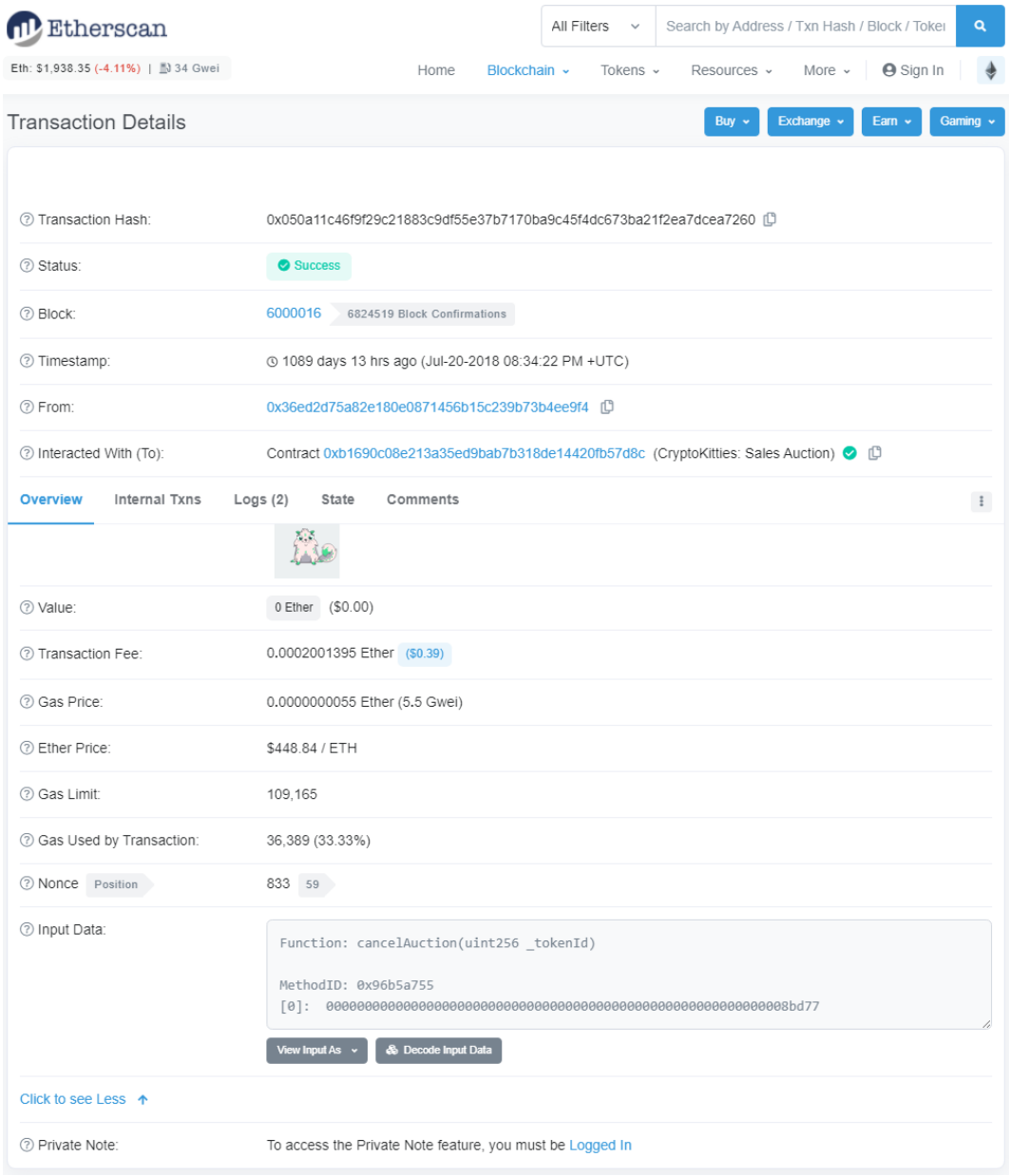


Figure 73: Details for transaction 0x050a11c46f9f29c21883c9df55e37b7170ba9c45f4dc673ba21f2ea7dcea7260 that included one transfer log entry from the CryptoKitties smart contract in block 6,000,016

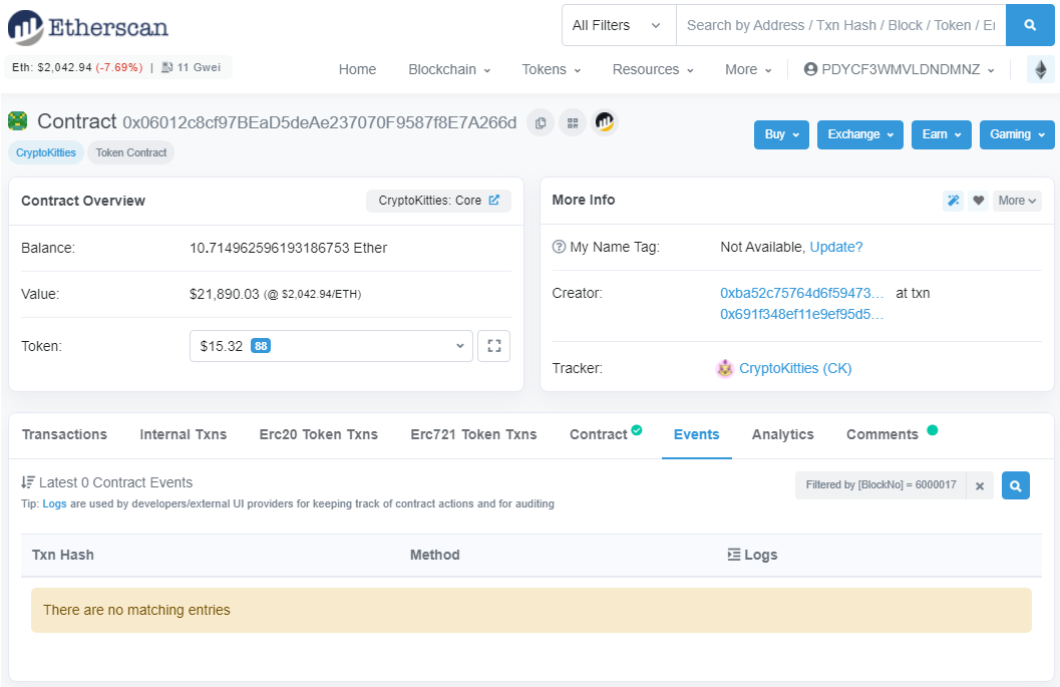


Figure 74: No log entries were created by the CryptoKitties smart contract in block 6,000,017

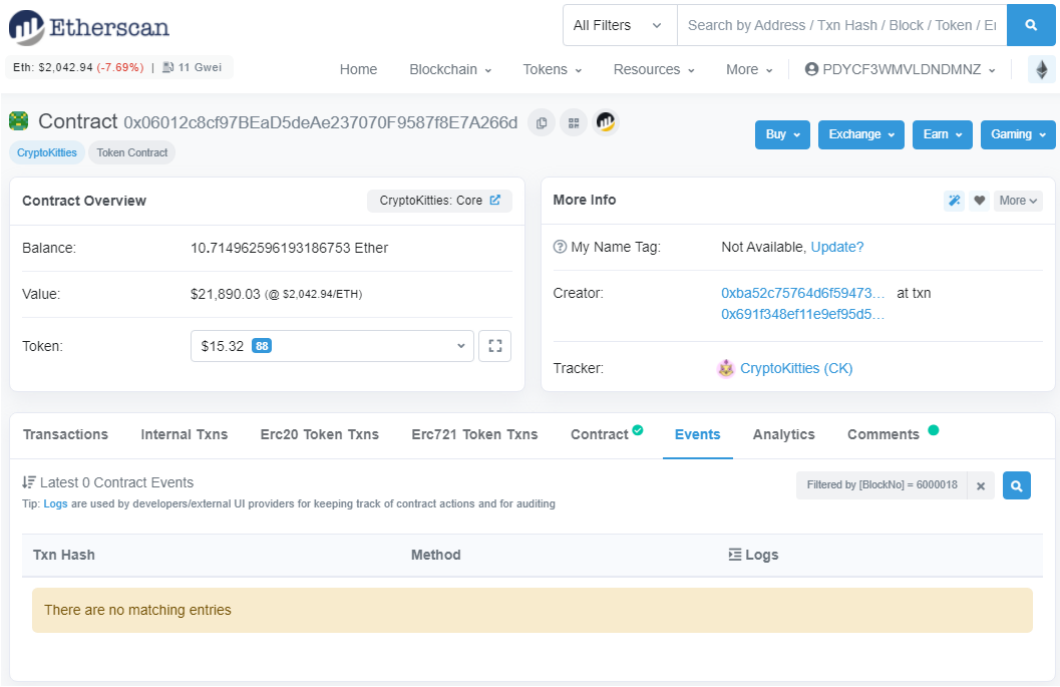


Figure 75: No log entries were created by the CryptoKitties smart contract in block 6,000,018

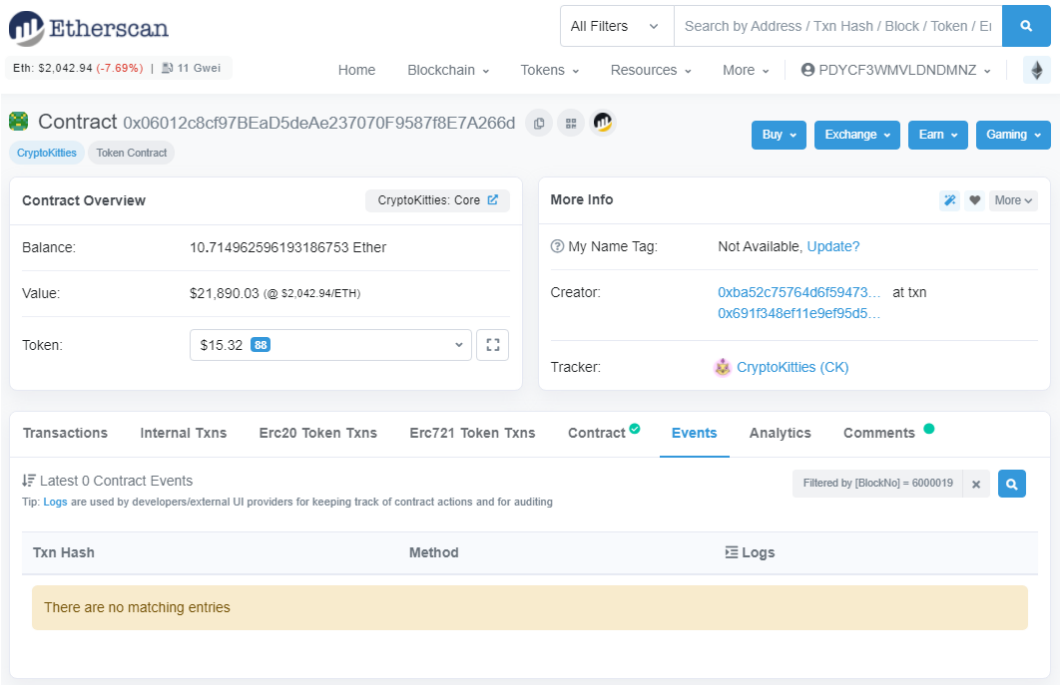


Figure 76: No log entries were created by the CryptoKitties smart contract in block 6,000,019

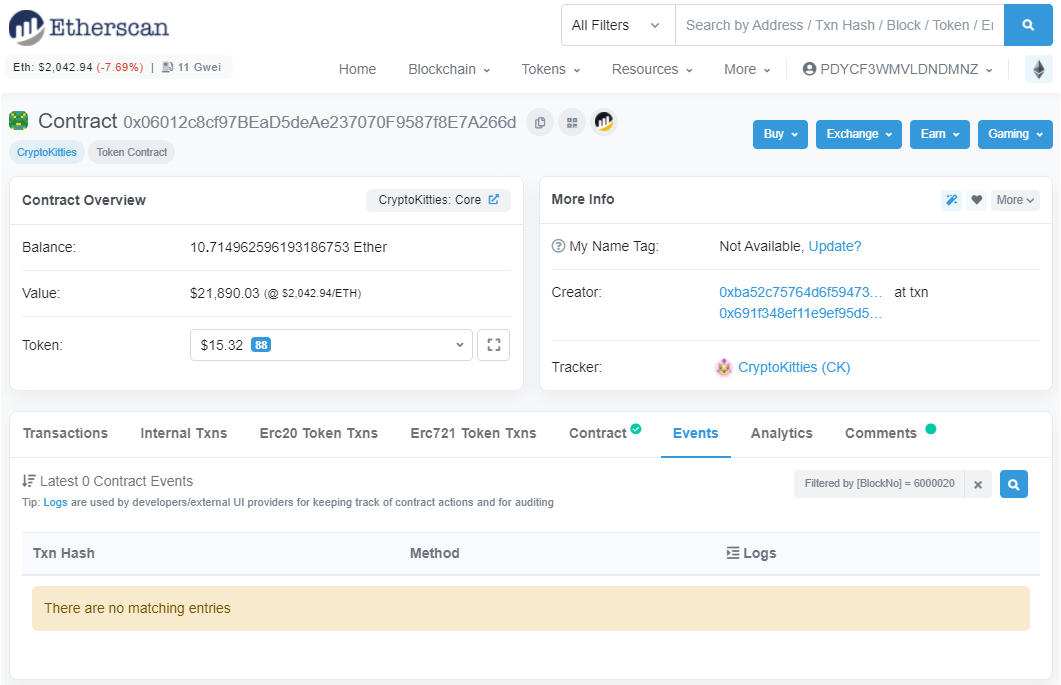


Figure 77: No log entries were created by the CryptoKitties smart contract in block 6,000,020

EXPORT

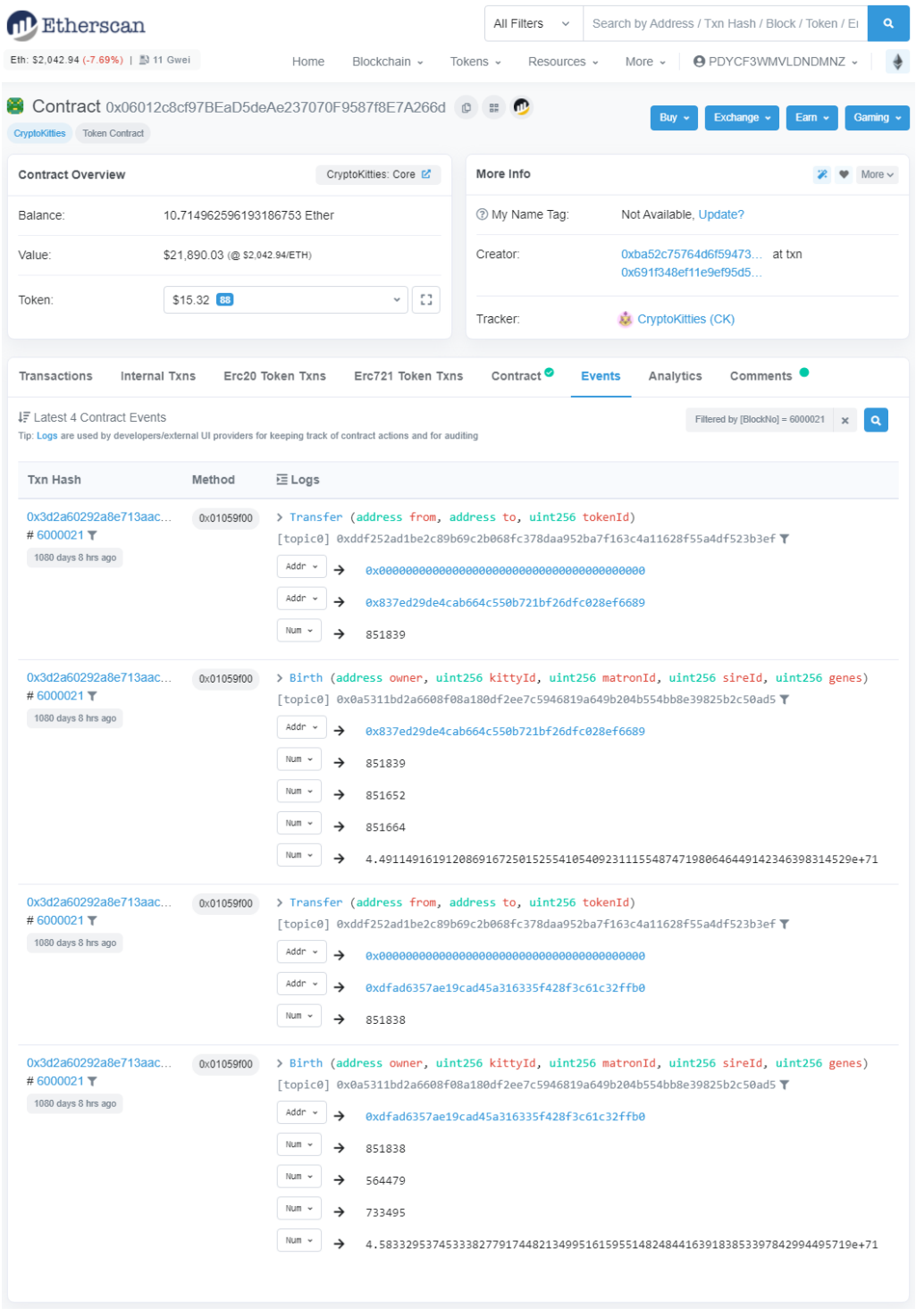

Etherscan
All Filters
Search by Address / Txn Hash / Block / Token / Et
Eth: $2,042.94 (-7.69%) | 11 Gwei
Home
Blockchain
Tokens
Resources
More
PDYCF3WMVLDNDMNZ
Contract 0x06012c8cf97BEaD5deAe237070F9587f8E7A266d
CryptoKitties
Token Contract
Buy
Exchange
Earn
Gaming
Contract Overview
CryptoKitties: Core
Balance:
10.714962596193186753 Ether
Value:
$21,890.03 (@ $2,042.94/ETH)
Token:
$15.32
More Info
My Name Tag:
Not Available, Update?
Creator:
0xba52c75764d6f59473... at txn
0x691f348ef11e9ef95d5...
Tracker:
CryptoKitties (CK)
Transactions
Internal Txns
Erc20 Token Txns
Erc721 Token Txns
Contract
Events
Analytics
Comments
Latest 4 Contract Events
Tip: Logs are used by developers/external UI providers for keeping track of contract actions and for auditing
Filtered by [BlockNo] = 6000021
Txn Hash
Method
Logs
0x3d2a60292a8e713aac...
6000021
1080 days 8 hrs ago
0x01059f00
Transfer (address from, address to, uint256 tokenId)
[topic0] 0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef
Addr
0x0000000000000000000000000000000000000000
Addr
0x837ed29de4cab664c550b721bf26dfc028ef6689
Num
851839
0x3d2a60292a8e713aac...
6000021
1080 days 8 hrs ago
0x01059f00
Birth (address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes)
[topic0] 0x0a5311bd2a6608f08a180df2ee7c5946819a649b204b554bb8e39825b2c50ad5
Addr
0x837ed29de4cab664c550b721bf26dfc028ef6689
Num
851839
Num
851652
Num
851664
Num
4.491149161912086916725015255410540923111554874719806464491423463983145 29e+71
0x3d2a60292a8e713aac...
6000021
1080 days 8 hrs ago
0x01059f00
Transfer (address from, address to, uint256 tokenId)
[topic0] 0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef
Addr
0x0000000000000000000000000000000000000000
Addr
0xdfad6357ae19cad45a316335f428f3c61c32ffb0
Num
851838
0x3d2a60292a8e713aac...
6000021
1080 days 8 hrs ago
0x01059f00
Birth (address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes)
[topic0] 0x0a5311bd2a6608f08a180df2ee7c5946819a649b204b554bb8e39825b2c50ad5
Addr
0xdfad6357ae19cad45a316335f428f3c61c32ffb0
Num
851838
Num
564479
Num
733495
Num
4.583329537453338277917448213499516159551482484416391838533978429944957 19e+71

Figure 78: Two birth and two transfer log entries were created by the CryptoKitties smart contract in block 6,000,021

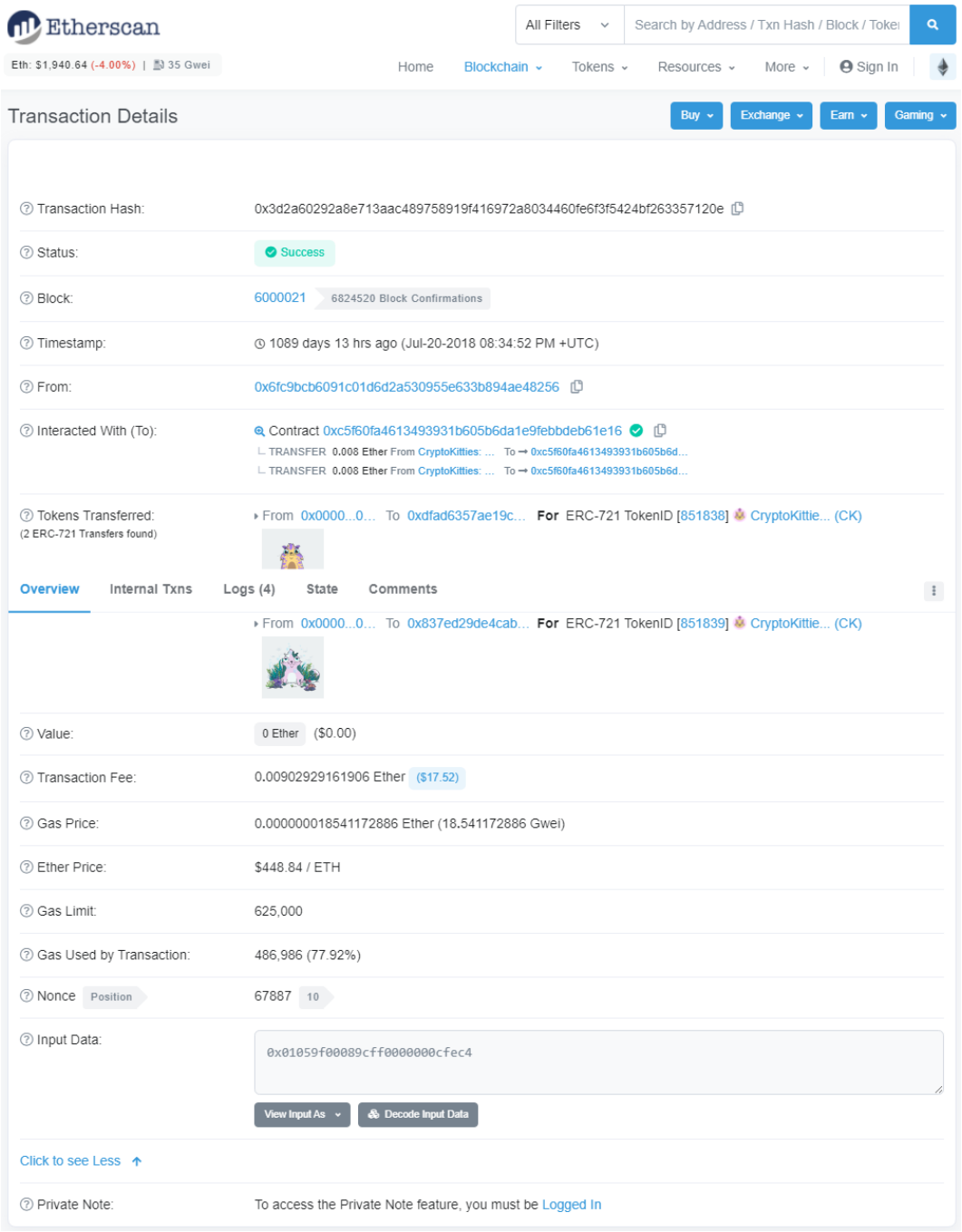


Figure 79: Details for transaction 0x3d2a60292a8e713aac489758919f416972a8034460fe6f3f5424bf263357120e that included two birth and two transfer log entries from the CryptoKitties smart contract in block 6,000,021

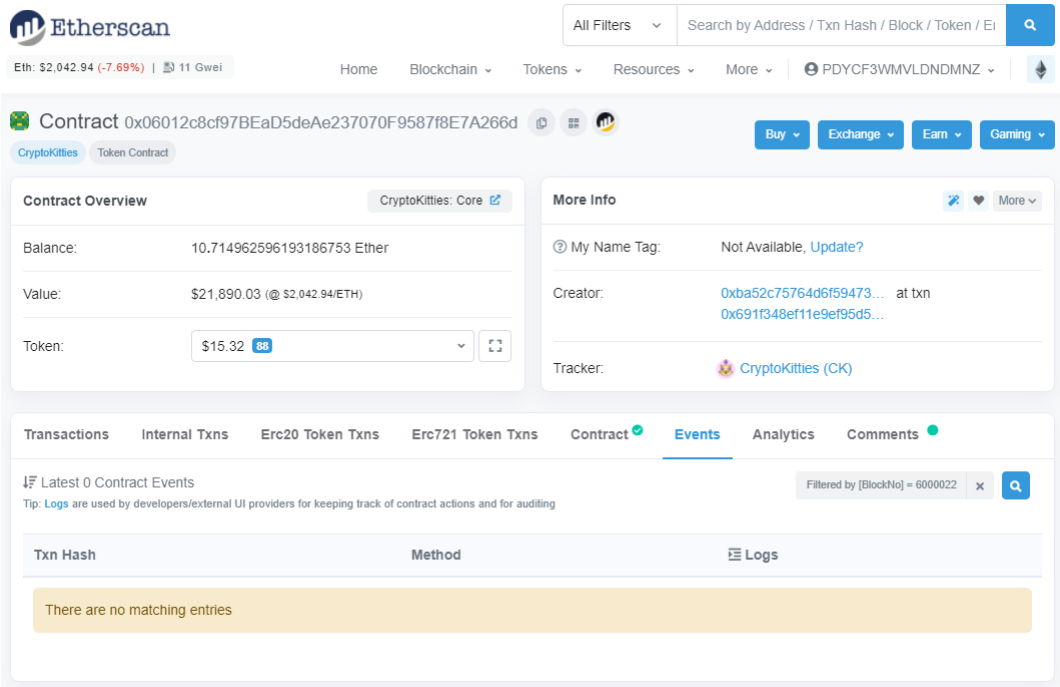


Figure 80: No log entries were created by the CryptoKitties smart contract in block 6,000,022

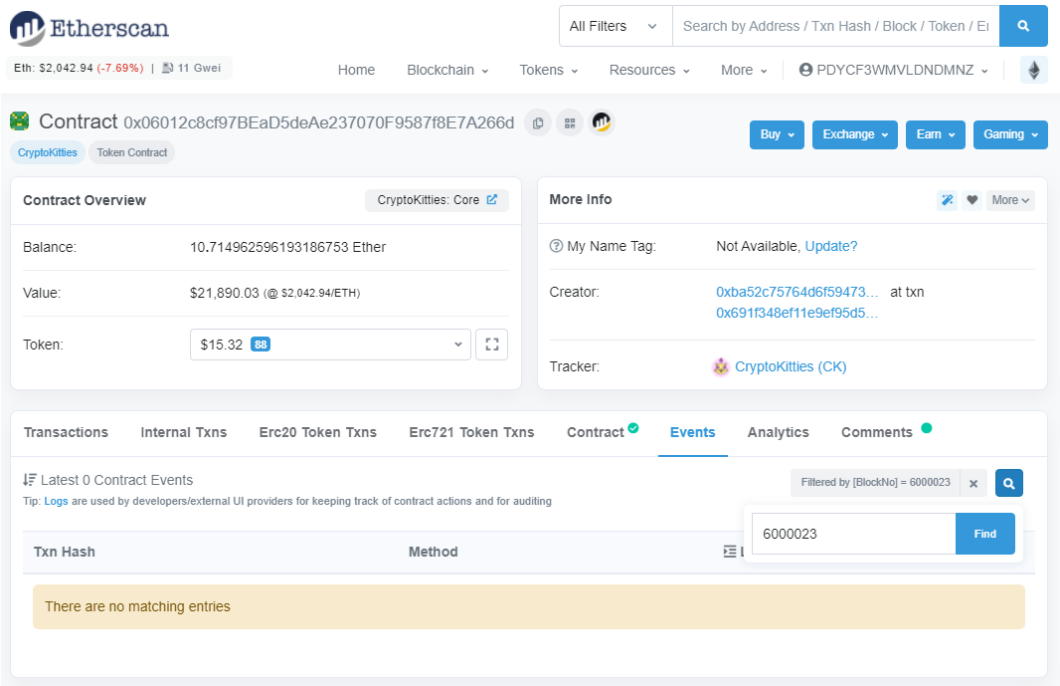


Figure 81: No log entries were created by the CryptoKitties smart contract in block 6,000,023

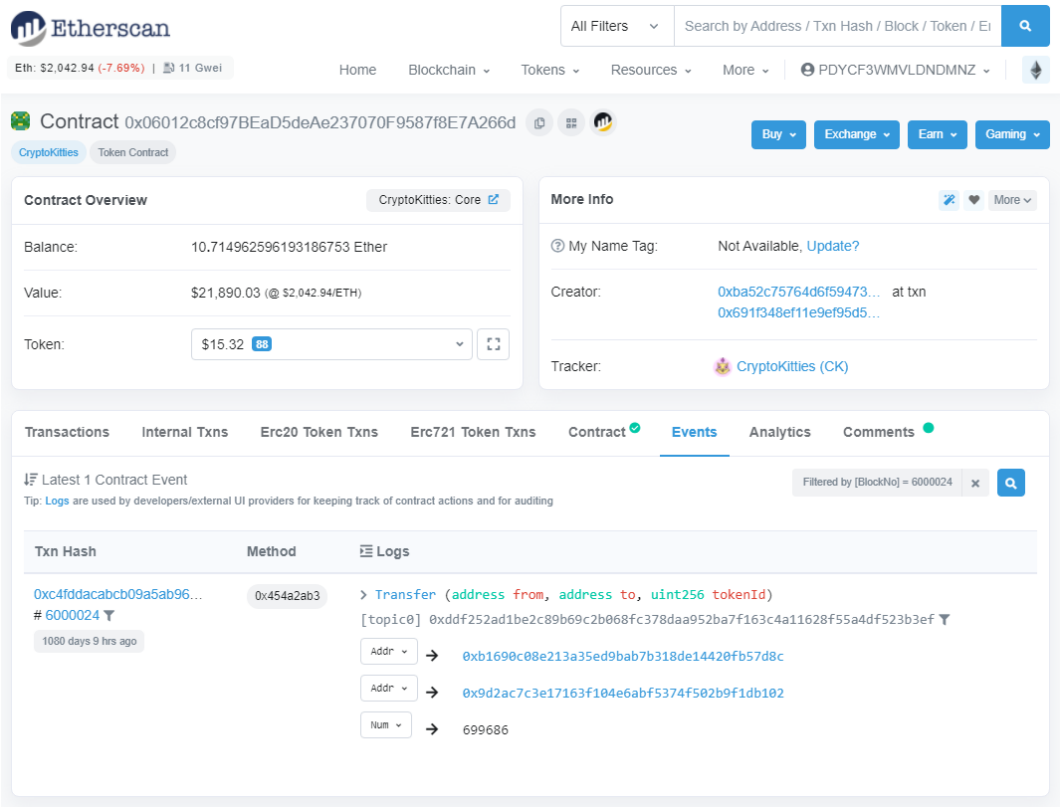


Figure 82: One transfer log entry was created by the CryptoKitties smart contract in block 6,000,024

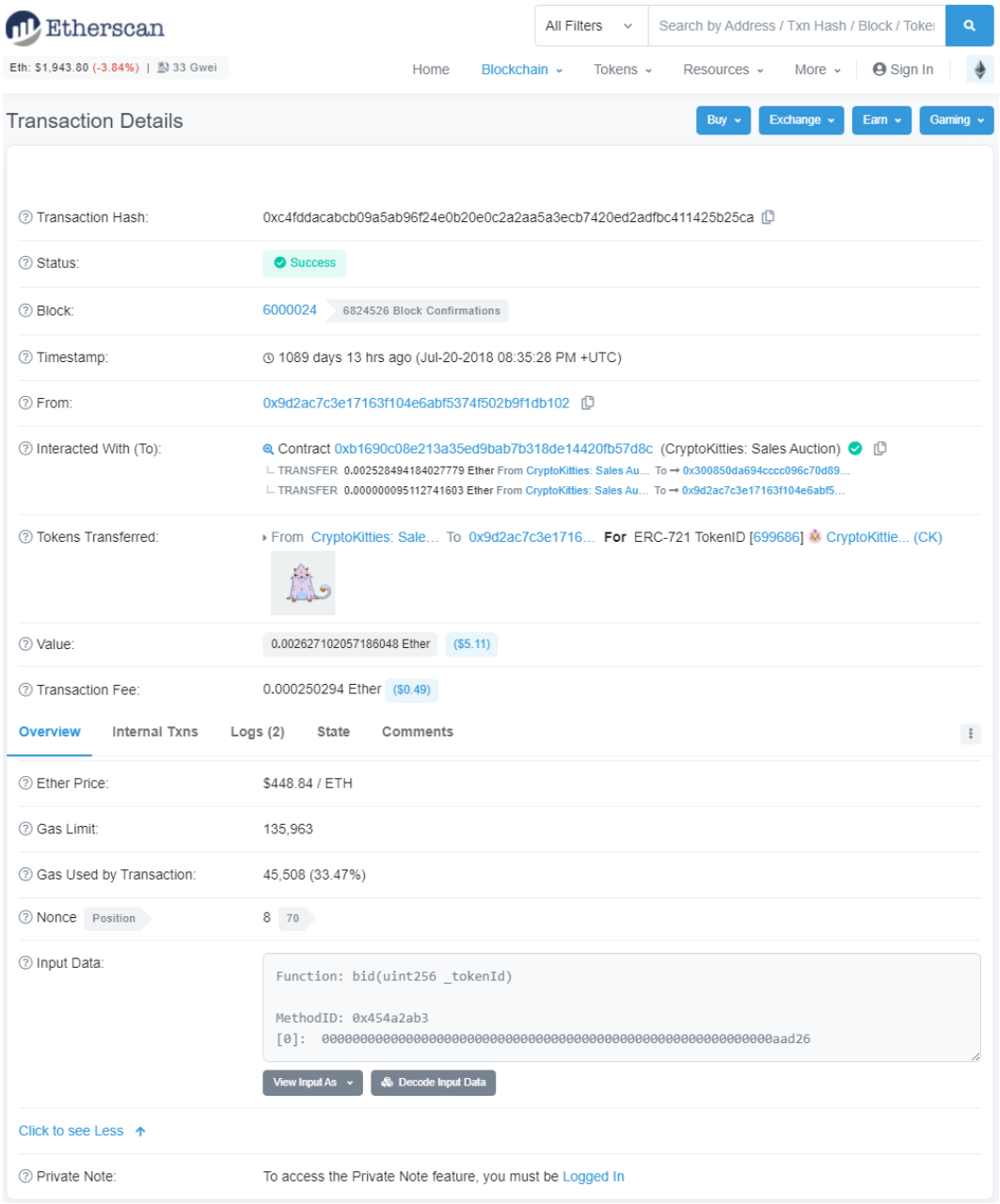
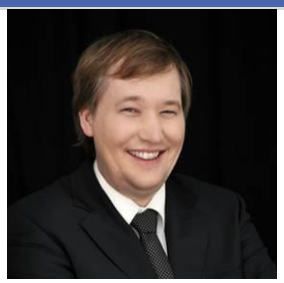


Figure 83: Details for transaction 0xc4fddacabcb09a5ab96f24e0b20e0c2a2aa5a3ecb7420ed2adfbc411425b25ca that included a transfer log entry from the CryptoKitties smart contract in block 6,000,024

Contact Information

WIL VAN DER AALST CHAIR	CHRISTIAN GÜNTHER VICE-CHAIR	ERIC VERBEEK SECRETARY
		
Tel +31 40 247 4295 w.m.p.v.d.aalst@tue.nl	**Tel** +31 64 1780680 christian@fluxicon.com	**Tel** +31 40 247 3755 h.m.w.verbeek@tue.nl

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