XES Software Communication Extension

M. Leemans C. Liu

June 22, 2017

Documents:	XES Software Communication Extension
Authors:	M. Leemans, C. Liu
Email:	m.leemans@tue.nl
Date:	June 22, 2017
Version:	5.2

1 Introduction and Background

During the execution of software, execution data can be recorded. With the development of process mining techniques on the one hand, and the growing availability of software execution data on the other hand, a new form of software analytics comes into reach. That is, applying process mining techniques to analyze software execution data. To enable process mining for software, event logs should be capable of capturing software-specific data.

In the context of multi-process and distributed software, there are multiple software applications interacting and communicating with each other. The *Software Communication* extension supports recording IP-based communication for relating events across software applications.

Extension definition

Name:	Software Communication
Prefix:	swcomm
URI:	http://www.xes-standard.org/swcomm.xesext
XML:	`<extension name="Software Communication" prefix="swcomm" uri="http://www.xes-standard.org/swcomm.xesext"/>`

The remainder of this extension is organized as follows. In Section 2 we explain some basic terminology. In Section 3 we detail how communication is recorded. Section 4 provides an example, and the XES Extension definition is given in Section 4.1. Finally, in Section 5 provides a reference glossary.

2 Terminology

In this extension, we will use some software-specific terminology. We provided a reference glossary in Section 5. For this extension, we will assume the use of the IP protocol.

In a multi-process or distributed software application, multiple programs are collaborating. Whenever a program A is collaborating with program B, for example, to request a service, they use a communication channel. Such a communication channel is identified by two endpoints: a local and a remote endpoint. Each endpoint is described by a combination of a host and a port.

For example, suppose program A runs on host 127.0.0.1 at port 9000, and program B runs on host 127.0.0.1 at port 9021. Then, when A sends some data to B, we can record two events: 1) the sending of data with local endpoint 127.0.0.1:9000 and remote endpoint 127.0.0.1:9021, and 2) the receiving of data with local endpoint 127.0.0.1:9021 and remote endpoint 127.0.0.1:9000. Observe that events can be related across applications through matching endpoints.

3 Communication Information

At runtime, whenever an application communicates with another program, it does so over a communication channel. Such a communication channel is identified by two endpoints: a local and a remote endpoint. Each endpoint is described by a combination of a host and a port.

Typical candidate locations for evens with communication information are sockets and other network or communication layer software code.

Level	Key	Type	Description
event	localHost	string	The *local host* name for the local endpoint.
event	localPort	int	The *local port* for the local endpoint.
event	remoteHost	string	The *remote host* name for the remote endpoint.
event	remotePort	int	The *remote port* for the remote endpoint.

4 Example

In Listing 1 an example XES trace is given for a simple client-server application. In Figure 1, a message sequence diagram is given as a visual aid. We assume a setup with three applications: ClientA at 127.0.0.1:9000, ClientB at 127.0.0.1:9001 and Server at 127.0.0.1:9021. In this example, first ClientA, and then ClientB, performs a requestService. At the Server, first the request from ClientB arrives, and after that the request from ClientA. Using the communication information, we can track these requests and event correlations. This example log consists of 1 trace with 8 events.

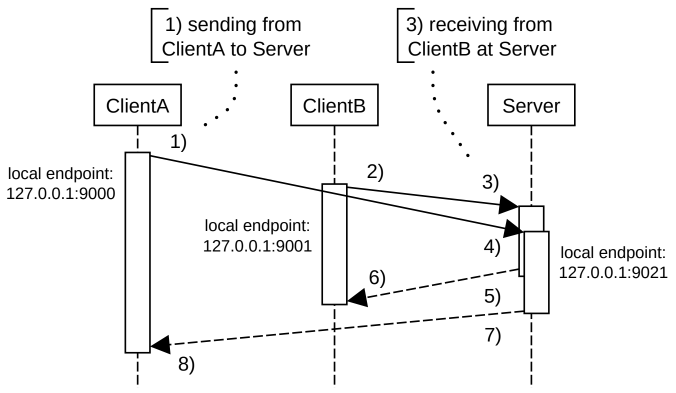
1) sending from ClientA to Server
3) receiving from ClientB at Server
ClientA
ClientB
Server
1)
2)
3)
4)
5)
6)
7)
8)
local endpoint: 127.0.0.1:9000
local endpoint: 127.0.0.1:9001
local endpoint: 127.0.0.1:9021

Figure 1: Message sequence diagram for the events in Listing 1. The numbers 1) to 8) refer to the actual events in Listing 1.

Listing 1: Example XES log for a simple client-server application.

```
<log>
  <extension name="Concept" prefix="concept"
     uri="http://www.xes-standard.org/concept.xesext"/>
  <extension name="Software Communication" prefix="swcomm"
     uri="http://www.xes-standard.org/swcomm.xesext"/>
  <trace>
    <event>----------------------- 1) sending data from ClientA to Server
      <string key="concept:name" value="requestService" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9000" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9021" />
    </event>
    <event> ---------------------- 2) sending data from ClientB to Server
      <string key="concept:name" value="requestService" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9001" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9021" />
    </event>
    <event>---------------------- 3) receiving data from ClientB at Server
      <string key="concept:name" value="receiveRequest" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9021" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9001" />
    </event>
    <event>---------------------- 4) receiving data from ClientA at Server
      <string key="concept:name" value="receiveRequest" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9021" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9000" />
    </event>
    <event> ---------------------- 5) sending data from Server to ClientB
      <string key="concept:name" value="handleRequest" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9021" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9001" />
    </event>
    <event>---------------------- 6) receiving data from Server at ClientB
      <string key="concept:name" value="receiveResponse" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9001" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9021" />
    </event>
    <event>---------------------- 7) sending data from Server to ClientA
      <string key="concept:name" value="handleRequest" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9021" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9000" />
    </event>
    <event>---------------------- 8) receiving data from Server at ClientA
      <string key="concept:name" value="receiveResponse" />
      <string key="swcomm:localHost" value="127.0.0.1" />
      <int key="swcomm:localPort" value="9000" />
      <string key="swcomm:remoteHost" value="127.0.0.1" />
      <int key="swcomm:remotePort" value="9021" />
    </event>
  </trace>
</log>
```

4.1 XES Extension

Listing 2: XES Extension - Software Communication.

```
<extension name="Software Communication" prefix="swcomm"
  uri="http://www.xes-standard.org/swcomm.xesext"/>
  <event>
    <string key="localHost">
      <alias mapping="EN" name="Local endpoint - host name"/>
    </string>
    <int key="localPort">
      <alias mapping="EN" name="Local endpoint - port"/>
    </int>
    <string key="remoteHost">
      <alias mapping="EN" name="Remote endpoint - host name"/>
    </string>
    <int key="remotePort">
      <alias mapping="EN" name="Remote endpoint - port"/>
    </int>
  </event>
</extension>
```

5 Glossary

local The *local* endpoint is the current application.

remote The *remote* endpoint is the destination application to which is communicated.

host The *host* address identifies a machine or node.

port The *port* is a communication endpoint, identifying a service on a host machine.