PAJÉ – trace file format
Version **1.3.1** - September 1st, 2015

This report describes the PAJÉ trace file format, a self-defined, textual, and generic format to describe the behavior of computer programs that are executed in parallel or distributed systems. A file in this format has three sections: event definition, type hierarchy declaration and timestamped events, complying with the event definitions, to be used to build visualizations according to the type hierarchy. See Section 6 for file revisions.

Current maintainer
- Lucas M. Schnorr (UFRGS, Brazil)

Contributors
- Benhur de Oliveira Stein (UFSM, Brazil)
- Mathieu Faverge (INRIA Bordeaux, France)
- François Trahay (Telecom SudParis, France)

Previous contributors
- Jacques Chassin de Kergommeaux (INPG, France)

Contents

More information about PAJÉ at `http://paje.sf.net`.

1. Introduction and overview

This report describes the PAJÉ trace file format, a self-defined, textual, and generic format to describe the behavior of computer programs that are executed in parallel or distributed systems. A file in this format includes three different kind of information: event definition, type hierarchy definition and a set of recorded events, complying with the format definition, to be used to build visualizations according to the type hierarchy.

Before going into the details of the file format itself, it is necessary to understand the five concepts of objects that exist in the PAJÉ format. They are:

Container. A container object can be any monitored entity, be it hardware or software. It represents anything which behavior can be described along time. A container can be a processor, an network link, a process, or a thread. **Note:** Container is the only PAJÉ object that can hold other objects, including other containers. It is the main component to define the **type hierarchy** necessary to PAJÉ trace files. See Section 3.1.1 to define specific container types.

State. A state is anything that has a beginning and ending timestamp. It is always associated to a container, and can be used to describe periods of time where a given container stays at the same state. See Section 3.1.2 to define specific state types.

Event. An event is anything remarkable enough to be uniquely identified in a trace file, having only one timestamp (when it happened). It is always associated to a container. See Section 3.1.3 to define specific event types.

Variable. A variable object represents the evolution of the value of a variable along time. It is always associated to a container and can be used to represent the numeric evolution of a variable along time. See Section 3.1.4 to define specific variable types.

Link. A link is used to represent a directed relation between two containers, having a beginning and ending timestamps (which represent the start and end of the relation). It is always associated to a container. It can be used to represent point-to-point or collective communications between processes. See Section 3.1.5 to define specific link types.

These five concepts of objects in the PAJÉ data format must be organized as a **type hierarchy** in a PAJÉ trace file. The *containers* types are the nodes of the tree, while the other four types (state, event, variable, and link) are the leaves of the hierarchy. This hierarchy is specific to each trace file, although one can use always the same type hierarchy for traces that are of the same scenario with the same monitored objects. All MPI applications, for example, might share the same type hierarchy. The type hierarchy definition is often written in the beginning of a PAJÉ trace file, by defining the specific types and the child/parent relation among them. The Section 4.6 presents the details on how to define the type hierarchy in the presence of multiple trace files that belong to the same program execution. The Section 3 details all the PAJÉ events that can be used to define types.

After the type hierarchy definition, there is the list of timestamped events that together describe the behavior of the monitored entities. Section 3 details all the PAJÉ events that can be used in this point to describe behavior.

The text is organized as follows. Section 2 presents the PAJÉ trace file format specification. Section 3 presents all the PAJÉ events: those used to declare new types and those to describe the behavior of monitored entities along time. Section 4 presents technical details about the PAJÉ file format, such as the notion of time, and use of aliases to reduce trace size. Section 5 lists the reference PAJÉ header that can be copied into your trace file. Section 6 presents the change log of this file.

2. PAJÉ trace file format specification

A trace file is composed of events. An event can be seen as a table composed of named fields, as shown in figure 1. The first event in the figure can represent the sending of a message containing 320 bytes by process 5 to process 3, 3.233222 seconds after the process started executing. The second event shows that process 5 unblocked at time 5.123002, and that this happened while executing line 98 of file sync.c.

Field Name	Field Type	Field Value
EventName	string	SendMessage
Time	timestamp	3.233222
ProcessId	integer	5
Receiver	integer	3
Size	integer	320

Field Name	Field Type	Field Value
EventName	string	UnblockProcess
Time	timestamp	5.123002
ProcessId	integer	5
FileName	string	sync.c
LineNumber	integer	98

Figure 1: Examples of events

Each event has some fields, each of them composed of a name, a type and a value. Generally, there are lots of similar events in a trace file (lots of "SendMessage" events, all with the same fields); a typical trace file contains thousands of events of tens of different types. Usually, events of the same type have the same fields. It is therefore wise, in order to reduce the trace file size, not to put the information that is common to many events in each of those events. The most common solution is to put only the type of each event and the values of its fields in the trace file. Information on what event types exist and the fields that constitute each of these event types being kept elsewhere. In some trace file formats, this information is hardcoded in the trace generator and trace reader, making the trace structure hard to change in order to incorporate new types of events, new data in existing events or to remove unused or unknown data from those events.

A PAJÉ trace file is self defined, meaning that the event definition information is put inside the trace file itself, much like the SDDF file format used by the Pablo visualization tool[1]. The file is constituted of two parts: the definition of the events at the beginning of the file followed by the events themselves. The definition of events contains the name of each event type and the names and types of each field. The second part of the trace file contains the events, with the values associated to each field, in the same order as in the definition. The correspondence of an event with its definition is made by means of a number, that must be unique for each event description; this number appears in an event definition and at the beginning of each event contained in the trace file.

[1] See "The Pablo Self-Defining Data Format", by Ruth A. Aydt (May 1994)

➲ First Part: Event Definition

The event definition part of a PAJÉ trace file follows the following format:

- All the lines start with a '%' character.
- Each event definition starts with a %EventDef line and terminates with a %EndEventDef line.
- The %EventDef line contains the name and the unique number of an event type. The number (an integer) will be used to identify the event in the second part of the trace file. The choice of this number is left to the user. The numbers given in event definitions are thus **arbitrary**. The name of the event will be put in a field called *PajeEventName*. There cannot be another field called so. The name is used to identify the type of an event.
- the fields of an event are defined between the %EventDef and the %EndEventDef lines, one field per line, with the name of the field followed by its type (see below for details on its types).

The type of a field can be one of the following:

date: for fields that represent dates. It's a double precision floating-point number, usually meaning seconds since program start (See Section 4.2 for a discussion about the Notion of Time in PAJÉ)
int: for fields containing integer numeric values;
double: for fields containing floating-point values;
hex: for fields that represent addresses, in hexadecimal;
string: for strings of characters.
color: for fields that represent colors. A color is a sequence of three floating-point numbers between 0 and 1, inside double quotes ("). The three numbers are the values of red, green and blue components.

The structure of the two events of figure 1 are shown below:

```
%EventDef SendMessage 21
%   Time       date
%   ProcessId  int
%   Receiver   int
%   Size       int
%EndEventDef

%EventDef UnblockProcess 17
%   Time       date
%   ProcessId  int
%   LineNumber int
%   FileName   string
%EndEventDef
```

➲ Second Part: Events

The second part of the trace file contains one event per line, whose fields are separated by spaces or tabs, the first field being the number that identifies the event type, followed by the other fields, in the same order that they appear in the definition of the event. Fields of type string must be inside double quotes (") if they contain space or tab characters, or if they are empty.

For example, the two events of figure 1 are shown below:

```
21 3.233222 5 3 320
17 5.123002 5 98 sync.c
```

In PAJÉ, event numbers are used only as a means to find the definition of an event; they are discarded as soon as an event is read. After being read, events are identified by their names. Two different definitions can have the same name (and different numbers), making it possible to have, in the same trace file, two events of the same type containing different fields. We may use this feature to optionally include the source file identification in some events. The "UnblockProcess" event in the examples above could also be defined without the fields FileName and LineNumber, for use in places where this information is not known or not necessary.

3. PAJÉ events

PAJÉ includes a simulator module which builds the hierarchical data structure from the elementary event records of the trace files. PAJÉ has no predefined containers or entities. Before an entity can be created and visualized, a hierarchy of container and entity types must be defined, and containers must be instantiated. In this section all the PAJÉ events to define and create entities are detailed.

3.1. Types

New types can be defined at any moment of a PAJÉ trace file. The events used to define these types share the fact that they have no timestamp, and they are taken into account for the events that follow the definition. In other words, a PAJÉ trace file is invalid if a type is used before its definition.

This Section describes how new types are defined in a PAJÉ trace file based on the five objects described in Section 1 (containers, states, events, links and variables). For each

Note: a figure is presented for each type definition, and they show the obligatory and optional fields. Obligatory fields are presented first. Optional fields can be safely omitted from a PAJÉ trace file, without prejudice to the type definition.

3.1.1 ➲ PajeDefineContainerType

The PajeDefineContainerType event is used to define container types. It must contain the fields *Name* and *Type*. It defines a new **container** type called *Name*, contained in a previously defined container type of *Type* (or the number zero – **0** – indicating the root container type).

Optionally this event can contain a field *Alias* with an alias to identify this type. See Section 4.3 to correctly use alias in a PAJÉ trace file.

PajeDefineContainerType		
Field Name	**Field Type**	**Description**
Name	string or integer	Name of new container type
Type	string or integer	Parent container type
Alias	string or integer	Alternative name of new container type

An invalid use of this event happens on the following cases:

- *Name* was already used in a previous definition
- *Type* is not a container type, or was not previously defined
- *Alias* was already used in a previous definition

Related timestamped PAJÉ events:

- PajeCreateContainer (Section 3.2.1)
- PajeDestroyContainer (Section 3.2.2)

3.1.2 ⊃ PajeDefineStateType

The PajeDefineStateType event is used to define new state types. It must contain the fields *Name* and *Type*. It defines a new **state** entity type called *Name*, contained in a previously defined container type of *Type*.

Optionally this event can contain a field *Alias* with an alias to identify this type. See Section 4.3 to correctly use alias in a PAJÉ trace file.

PajeDefineStateType		
Field Name	**Field Type**	**Description**
Name	string or integer	Name of the new state type
Type	string or integer	Parent container type
Alias	string or integer	Alternative name of new state type

An invalid use of this event happens on the following cases:

- *Name* was already used in a previous definition
- *Type* is not a container type, or was not previously defined
- *Alias* was already used in a previous definition

Related timestamped PAJÉ events:

- PajeSetState (Section 3.3.1)
- PajePushState (Section 3.3.2)
- PajePopState (Section 3.3.3)
- PajeResetState (Section 3.3.4)

Related PAJÉ events:

- PajeDefineContainerType (Section 3.1.1)
- PajeDefineEntityValue (Section 3.1.6)

3.1.3 ⊃ PajeDefineEventType

The PajeDefineEventType event is used to define new event types. This event must contain the fields *Name* and *Type*. It defines a new **event** entity type called *Name*, contained in a previously defined container type of *Type*.

Optionally this event can contain a field *Alias* with an alias to identify this type. See Section 4.3 to correctly use alias in a PAJÉ trace file.

PajeDefineEventType		
Field Name	**Field Type**	**Description**
Name	string or integer	Name of new event type
Type	string or integer	Parent container type
Alias	string or integer	Alternative name of new event type

An invalid use of this event happens on the following cases:
- *Name* was already used in a previous definition
- *Type* is not a container type, or was not previously defined
- *Alias* was already used in a previous definition

Related timestamped PAJÉ events:
- PajeNewEvent (Section 3.4.1)

Related PAJÉ events:
- PajeDefineContainerType (Section 3.1.1)
- PajeDefineEntityValue (Section 3.1.6)

3.1.4 ⊃ PajeDefineVariableType

The PajeDefineVariableType event is used to define new variable types. This event must contain the fields *Name*, *Type*, and *Color*. It defines a new **variable** entity type called *Name*, contained in a previously defined container type of *Type*, with the color *Color*, if defined.

Optionally this event can contain a field *Alias* with an alias to identify this type. See Section 4.3 to correctly use alias in a PAJÉ trace file.

PajeDefineVariableType		
Field Name	**Field Type**	**Description**
Name	string or integer	Name of new variable type
Type	string or integer	Parent container type
Color	color	The color of the new variable
Alias	string or integer	Alternative name of new variable type

An invalid use of this event happens on the following cases:
- *Name* was already used in a previous definition
- *Type* is not a container type, or was not previously defined
- *Alias* was already used in a previous definition

Related timestamped PAJÉ events:
- PajeSetVariable (Section 3.5.1)
- PajeAddVariable (Section 3.5.2)
- PajeSubVariable (Section 3.5.3)

3.1.5 ➲ PajeDefineLinkType

The PajeDefineLinkType event is used to define new link types. This event must contain the fields *Name* and *Type*, *StartContainerType*, and *EndContainerType*. It defines a new **link** entity type called *Name*, contained in a previously defined container type of *Type*, that connects the previously defined container type *StartContainerType* to the previously defined container type *EndContainerType*. Additionally, the container type provided in *Type* must be a common ancestral of containers types provided in *StartContainerType* and *EndContainerType*.

Optionally this event can contain a field *Alias* with an alias to identify this type. See Section 4.3 to correctly use alias in a PAJÉ trace file.

PajeDefineLinkType		
Field Name	**Field Type**	**Description**
Name	string or integer	Name of new link type
Type	string or integer	Common ancestral container type
StartContainerType	string or integer	Container type of the starting container
EndContainerType	string or integer	Container type of the ending container
Alias	string or integer	Alternative name of new link type

An invalid use of this event happens on the following cases:

- *Name* was already used in a previous definition
- *Type* is not a container type, or was not previously defined
- *Type* is not a common ancestral of *StartContainerType* and *EndContainerType*
- *StartContainerType* is not a container type, or was not previously defined
- *EndContainerType* is not a container type, or was not previously defined
- *Alias* was already used in a previous definition

Related timestamped PAJÉ events:

- PajeStartLink (Section 3.6.1)
- PajeEndLink (Section 3.6.2)

Related PAJÉ events:

- PajeDefineContainerType (Section 3.1.1)
- PajeDefineEntityValue (Section 3.1.6)

3.1.6 ➲ PajeDefineEntityValue

The PajeDefineEntityValue event is used to define the possible values of an entity type. The entity types whose values can be defined are the states, links, and events. The advantage of using this event is that colors can be hard-coded in the trace, but this event is completely optional. It defines a new value called *Name* for the previously defined type specified in *Type*. The color of this new value is defined in the *Color* field, if provided.

Optionally this event can contain a field *Alias* with an alias to identify this type. See Section 4.3 to correctly use alias in a PAJÉ trace file.

PajeDefineEntityValue		
Field Name	**Field Type**	**Description**
Type	string or integer	A state, event, or link type
Name	string or integer	Name of the new value
Color	color	Color of the value
Alias	string or integer	Alternative name of the new value

An invalid use of this event happens on the following cases:

- *Name* was already used in a previous definition
- *Type* is not a state, event, or link type, or was not previously defined
- *Color* (if provided) is not a valid color
- *Alias* was already used in a previous definition

Related PAJÉ events:

- PajeDefineStateType (Section 3.1.2)
- PajeDefineEventType (Section 3.1.3)
- PajeDefineLinkType (Section 3.1.5)

3.2. Containers

Instances of containers are created using the PajeCreateContainer event, and destroyed using the PajeDestroyContainer event. If instances of containers are not destroyed, they must exist until the end of the trace file. These two events are detailed below.

3.2.1 ⊃ PajeCreateContainer

The PajeCreateContainer event creates a container instance in a given timestamp. This event must contain the fields *Time*, *Name*, *Type*, and *Container*. It creates, at timestamp *Time*, a container instance uniquely identified by *Name* of the container type *Type*. This new container instance is a new child of the previously created container instance identified by *Container* (the parent container instance).

Optionally this event can contain a field *Alias* with an alias to identify this type. See Section 4.3 to correctly use alias in a PAJÉ trace file.

Optionally this event can contain a field *Filename* with a the name of a file that contains the events related to this *Container*.

PajeCreateContainer		
Field Name	**Field Type**	**Description**
Time	date	Time of creation of container
Name	string or integer	Name of new container
Type	string or integer	Container type of new container
Container	string or integer	Parent of new container
Alias	string or integer	Alternative name of new container
Filename	string	Filename with events of this container

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Name* instance identifier was already used in a previous container creation
- *Type* is not a container type, or was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- *Alias* instance identifier was already used in a previous container creation
- *Filename* is provided but file does not exist

Related type definition PAJÉ events:

- PajeDefineContainerType (Section 3.1.1)

3.2.2 ⊃ PajeDestroyContainer

The PajeDestroyContainer event destroys a container instance in a given timestamp. This event must contain the fields *Time*, *Name*, *Type*. It destroys, at timestamp *Time*, a container instance uniquely identified by *Name* of the container type *Type*.

PajeDestroyContainer		
Field Name	**Field Type**	**Description**
Time	date	Time of destruction of container
Name	string or integer	Name of container
Type	string or integer	Type of container

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Name* instance identifier does not exist (see Section 4.3 if alias are used in the trace file)
- *Type* is not a container type, or was not previously defined
- *Type* is not the container type of the container instance identified by *Name*

Related type definition PAJÉ events:

- PajeDefineContainerType (Section 3.1.1)

3.3. States

The PajeSetState updates a state of a container instance to a new value (and erase any previously saved values). The PajePushState pushes a new value of a state of a container instance, saving the old state. The PajePopState pops the previously saved value of a state of a container instance. The PajeResetState resets the stacked values of a state of a container instance, without pushing a new value to it. These three events are detailed below.

3.3.1 ⊃ PajeSetState

The PajeSetState event is used to update the value of a state of a container instance. This event must contain the fields *Time*, *Type*, *Container* and *Value*. It changes the state type *Type* to the value *Value* of the container identified by *Container* at time *Time*.

PajeSetState		
Field Name	**Field Type**	**Description**
Time	date	Time the state changed
Type	string or integer	State type
Container	string or integer	Container whose state changed
Value	string or integer	Value of new state of container

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a state type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created

Related type definition PAJÉ events:

- PajeDefineStateType (Section 3.1.2)

3.3.2 ⊃ PajePushState

The PajePushState event is used to push the value of a state of a container instance, saving the existing value of the same state. This event must contain the fields *Time*, *Type*, *Container* and *Value*. It pushes the value *Value* of state type *Type* in the container identified by *Container* at time *Time*.

PajePushState		
Field Name	**Field Type**	**Description**
Time	date	Time the state changed
Type	string or integer	State type
Container	string or integer	Container whose state changed
Value	string or integer	Value of new state of container

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a state type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- There is no existing value to be saved (see note below)

Related type definition PAJÉ events:

- PajeDefineStateType (Section 3.1.2)

Note: Considering a state *Type* in a *Container*, when a PajePushState is read from the trace file, it is expected that a previous PajeSetState was already read for the same state type and container.

3.3.3 ⊃ PajePopState

The PajePopState event is used to pop the previously saved value (with PajePushState) of a state of a container instance. This event must contain the fields *Time*, *Type*, and *Container*. It pops the state type *Type* in the container identified by *Container* at time *Time*.

PajePopState		
Field Name	**Field Type**	**Description**
Time	date	Time the state changed
Type	string or integer	State type
Container	string or integer	Container whose state changed

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a state type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- There is no saved value to pop (see note below)

Related type definition PAJÉ events:

- PajeDefineStateType (Section 3.1.2)

Note: Considering a state *Type* in a *Container*, when a PajePopState is read from the trace file, it is expected that a previous PajePushState was already read for the same state type and container.

3.3.4 ➲ PajeResetState

The PajeResetState event is used to clear all previously saved values (with PajePushState or with PajeSetState) of a state of a container instance. This event must contain the fields *Time*, *Type*, and *Container*. It clears the state type *Type* in the container identified by *Container* at time *Time*. If there was no stacked values for the state, this event does nothing.

PajeResetState		
Field Name	**Field Type**	**Description**
Time	date	Time the state changed
Type	string or integer	State type
Container	string or integer	Container to be considered

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a state type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created

Related type definition PAJÉ events:

- PajeDefineStateType (Section 3.1.2)

3.4. Events

3.4.1 ➲ PajeNewEvent

The PajeNewEvent event instantiates a remarkable event with a unique timestamp. This event must contain the fields *Time*, *Type*, *Container*, and *Value*.

PajeNewEvent		
Field Name	**Field Type**	**Description**
Time	date	Time the event happened
Type	string or integer	Type of event
Container	string or integer	Container that produced event
Value	string or integer	Value of new event

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a event type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created

Related type definition PAJÉ events:

- PajeDefineEventType (Section 3.1.3)

3.5. Variables

The PajeSetVariable changes the value of a variable of a container instance to a new value. The PajeAddVariable adds the value provided to the current value, while PajeSubVariable subtracts the value provided from the current value. These three events are detailed below.

3.5.1 ➲ PajeSetVariable

The PajeSetVariable event sets the value of a variable type of a container instance to a new value. This event must contain the fields *Time*, *Type*, *Container*, and *Value*.

PajeSetVariable		
Field Name	**Field Type**	**Description**
Time	date	Time the variable changed value
Type	string or integer	Type of variable
Container	string or integer	Container whose value changed
Value	double	New value of variable

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a variable type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- *Value* is not a double

Related type definition PAJÉ events:

- PajeDefineVariableType (Section 3.1.4)

3.5.2 ⊃ PajeAddVariable

The PajeAddVariable event adds a value to the existing value of a variable type of a container instance. This event must contain the fields *Time*, *Type*, *Container*, and *Value*.

PajeAddVariable		
Field Name	**Field Type**	**Description**
Time	date	Time the variable changed value
Type	string or integer	Type of variable
Container	string or integer	Container whose value changed
Value	double	Value to be added to variable

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a variable type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- *Value* is not a double
- There is no previous defined value for the variable (see note below)

Related type definition PAJÉ events:

- PajeDefineVariableType (Section 3.1.4)

Note: Considering a variable *Type* in a *Container*, when a PajeAddVariable is read from the trace file, it is expected that a previous PajeSetVariable was already read for the same variable type and container.

3.5.3 ⊃ PajeSubVariable

The PajeSubVariable event subtracts a value from the existing value of a variable type of a container instance. This event must contain the fields *Time*, *Type*, *Container*, and *Value*.

PajeSubVariable		
Field Name	**Field Type**	**Description**
Time	date	Time the variable changed value
Type	string or integer	Type of variable
Container	string or integer	Container whose value changed
Value	double	Value to be subtracted from variable

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a variable type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- *Value* is not a double
- There is no previous defined value for the variable (see note below)

Related type definition PAJÉ events:

- PajeDefineVariableType (Section 3.1.4)

Note: Considering a variable *Type* in a *Container*, when a PajeSubVariable is read from the trace file, it is expected that a previous PajeSetVariable was already read for the same variable type and container.

3.6. Links

A PAJÉ link is defined by two events: a PajeStartLink and a PajeEndLink. These two events are matched and considered to form a link when their respective *Container*, *Value* and *Key* fields, which should always be the same.

3.6.1 ➲ PajeStartLink

The PajeStartLink event indicates the beginning of a link between two containers. This event must contain the fields *Time*, *Type*, *Container*, *StartContainer*, *Value* and *Key*.

PajeStartLink		
Field Name	**Field Type**	**Description**
Time	date	Time the link started
Type	string or integer	Type of link
Container	string or integer	Container that has the link
StartContainer	string or integer	Container where link started
Value	string or integer	Value of link
Key	string or integer	Used to match to link end

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a link type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- *StartContainer* instance was not previously created
- *Key* was already used by another PajeStartLink

Related type definition PAJÉ events:

- PajeDefineLinkType (Section 3.1.5)

3.6.2 ➲ PajeEndLink

The PajeEndLink event indicates the ending of a link between two containers. This event must contain the fields *Time*, *Type*, *Container*, *EndContainer*, *Value* and *Key*.

PajeEndLink		
Field Name	**Field Type**	**Description**
Time	date	Time the link started
Type	string or integer	Type of link
Container	string or integer	Container that has the link
EndContainer	string or integer	Container where link ended
Value	string or integer	Value of link
Key	string or integer	Used to match to link start

An invalid use of this event happens on the following cases:

- *Time* is not present, or is not a timestamp (see Section 4.2 for time considerations)
- *Type* is not a link type, or it was not previously defined
- *Type* is not a child type of the container type of *Container*
- *Container* instance was not previously created
- *EndContainer* instance was not previously created
- *Key* was already used by another PajeEndLink

Related type definition PAJÉ events:

- PajeDefineLinkType (Section 3.1.5)

3.7. Multiple traces

3.7.1 ⊃ PajeTraceFile

The PajeTraceFile event indicates that another trace file contains events that describe the behavior of the given container. This event must contain the fields *Container*, *Type* and *Filename*. See Section 4.6 for details on how to organize the events in multiple paje trace files.

PajeTraceFile		
Field Name	**Field Type**	**Description**
Container	string	Container
Type	string	Type of the given container
Filename	string	Filename with events of this container

An invalid use of this event happens on the following cases:

- *Container* instance was not previously created
- *Type* is not a child type of the container type of *Container*
- *Filename* does not exist

Related type definition PAJÉ events:

- PajeDefineContainerType (Section 3.1.1)
- PajeCreateContainer (Section 3.2.1)

4. Technical details

4.1. Comments and blank lines

All lines starting by a **#** are ignored the same way blank lines are. They can be anywhere in the trace file. Within the same line, everything after a **#** is also ignored.

4.2. Notion of time

The notion of time accepted by PAJÉ is assumed to be in **seconds** since program start. So, the first timestamped event of a trace file should have a time that equals to zero (or is close to zero). The resolution of timestamps, in micro or nanoseconds, is left to the user and handled by PAJÉ. The resolution is separated from the number of seconds by a point. Here are some examples of timestamps accepted by PAJÉ:

```
0.0
10.45
12.345676
23.542831209
```

Although the PAJÉ file format assumes timestamps in seconds, the user is free to use another notion of time, such as logical clocks. Analysis tools that read PAJÉ trace files must be configured to accordingly.

4.3. Using aliases

The field *Alias* is an optional argument in PAJÉ events that can be used to decrease the size of trace files. When they are used, all references to types (through *Type*, *StartContainerType*, and *EndContainerType* fields) and containers (through *Container*, *StartContainer*, and *EndContainer* fields) have to be done through the value of their *Alias* (the value provided when the type was defined or the container created), and not through the *Name* of types and containers.

The use of aliases also enables for the definition of more than one container with the same name, by using one different alias for each container.

4.4. String delimitation

Fields of type string must be inside double quotes (") if they contain space or tab characters, or if they are empty. Single quotes (') are not used.

4.5. Reserved words

The only reserved word in the PAJÉ file format is the 0 (zero). It is the root representation of types and entities. It should not be used as an *Alias* or a *Name* of a type or entity.

4.6. Parallel reading of PAJÉ trace files

A trace of a single parallel application execution can be organized in multiple files. The choice of how many files are used to register a single application execution is up to the tracer that is registering the events. This section gives the guidelines on how this arbitrary number of trace files should be internally organized. The general idea is that each trace file can be self-sufficient, containing the header information it needs and all the type hierarchy definitions it requires to the events be parsed and simulated correctly by the reader responsible for a trace file.

Multiple trace files support. There are two ways to do it: first, using the PajeTraceFile event and second, using the *Filename* field of the PajeCreateContainer. In the former, the main trace file has multiple PajeTraceFile events with information on the *Filename* stating on which file the events of a given container are registered. In this case, the container should be created prior to the appearance of the respective PajeTraceFile. In the later, the *Filename* field of the PajeCreateContainer event contains the file name where the behavior of the given container is registered.

Header appearance. The event definitions (all lines starting with percentage) can appear in all the multiple trace files belonging to the same execution. The rule is that if they are present in a given trace file, the events in the second section should obey the event definitions of the header. If they are missing, the events of a given trace file should obey the header definitions of the trace file that imported it (either with a PajeTraceFile or a PajeCreateContainer event).

Type definitions. Each trace file belonging to the same trace set should have all the type definitions it requires to be correctly parsed and simulated. For example, if a given type appears only in a container, it could be defined only in the corresponding trace file that has the behavior of such container. If it is a "global" type, it should be defined only once in the main trace file that import all other files of the trace set.

5. Reference PAJÉ header

This is the reference of PAJÉ header, with the *Alias* field. Since the *Alias* field is optional, you can obtain a PAJÉ header without it just by removing all lines containing the word "Alias".

```
%EventDef PajeDefineContainerType 0
%  Alias string
%  Type string
%  Name string
%EndEventDef
%EventDef PajeDefineVariableType 1
%  Alias string
%  Type string
%  Name string
```

```
%  Color color
%EndEventDef
%EventDef PajeDefineStateType 2
%  Alias string
%  Type string
%  Name string
%EndEventDef
%EventDef PajeDefineEventType 3
%  Alias string
%  Type string
%  Name string
%  Color color
%EndEventDef
%EventDef PajeDefineLinkType 4
%  Alias string
%  Type string
%  StartContainerType string
%  EndContainerType string
%  Name string
%EndEventDef
%EventDef PajeDefineEntityValue 5
%  Alias string
%  Type string
%  Name string
%  Color color
%EndEventDef
%EventDef PajeCreateContainer 6
%  Time date
%  Alias string
%  Container string
%  Type string
%  Name string
%  Filename string
%EndEventDef
%EventDef PajeDestroyContainer 7
%  Time date
%  Type string
%  Name string
%EndEventDef
%EventDef PajeSetVariable 8
%  Time date
%  Container string
%  Type string
%  Value double
%EndEventDef
```

```
%EventDef PajeAddVariable 9
%  Time date
%  Container string
%  Type string
%  Value double
%EndEventDef
%EventDef PajeSubVariable 10
%  Time date
%  Container string
%  Type string
%  Value double
%EndEventDef
%EventDef PajeSetState 11
%  Time date
%  Container string
%  Type string
%  Value string
%EndEventDef
%EventDef PajePushState 12
%  Time date
%  Container string
%  Type string
%  Value string
%EndEventDef
%EventDef PajePopState 13
%  Time date
%  Container string
%  Type string
%EndEventDef
%EventDef PajeResetState 14
%  Time date
%  Container string
%  Type string
%EndEventDef
%EventDef PajeStartLink 15
%  Time date
%  Container string
%  Type string
%  StartContainer string
%  Value string
%  Key string
%EndEventDef
%EventDef PajeEndLink 16
%  Time date
%  Container string
```

```
%  Type string
%  EndContainer string
%  Value string
%  Key string
%EndEventDef
%EventDef PajeNewEvent 17
%  Time date
%  Container string
%  Type string
%  Value string
%EndEventDef
%EventDef PajeTraceFile 18
%  Container string
%  Type string
%  Filename string
%EndEventDef
```

6. ChangeLog

Version 1.3.1 (September 1st, 2015)

- Color is optional for PajeDefineVariableType
- Color is optional for PajeDefineEntityValue

Version 1.3 (October 15th, 2014)

- New event PajeTraceFile to support parallel reading (see Section 3.7.1)
- New optional field *Filename* added to the PajeCreateContainer event definition (Section 3.2.1)
- Section 4.6 added with information on how the traces should be organized for parallel reading

Version 1.2.5 (February 14th, 2013)

- Fix text in Section 2
- Update e-mail and affiliation of current maintainer

Version 1.2.4 (December 7th, 2012)

- Adding a section to describe the reserved words in PAJÉ
- Change field order in recommended headers (for PajeSetVariable, PajeAddVariable, PajeSubVariable and PajeCreateContainer)
- Removing PAJÉ header recommendation without *Alias*

Version 1.2.3 (September 12th, 2012)

- Fix typographical error in the reference header file with *Alias*

Version 1.2.2 (June 7th, 2012)
- New event PajeResetState added

Version 1.2.1 (April 30th, 2012)
- Description for PajeDefineEntityValue event added
- The PajeDefineVariableType event has a *Color* field, fixed.
- Rename deprecated *EntityType* field to *Type*

Version 1.2 (March 14th, 2012)
- Update to the latest format (as accepted by the PAJÉ visualization tool)
  - the field *SourceContainer* is deprecated, replaced by *StartContainer*
  - the field *DestContainer* is deprecated, replaced by *EndContainer*
  - the field *ContainerType* is deprecated, replaced by *Type*
  - the field *SourceContainerType* is deprecated, replaced by *StartContainerType*
  - the field *DestContainerType* is deprecated, replaced by *EndContainerType*
  - the fields *Shape*, *Height* and *Width* (previously deprecated) are removed

Version 1.1 (February 24th, 2010)
- No longer accept interchangeable *Name* and *Alias* reference for types and containers

Version 1.0 (March 22nd, 2003)
- Original proposal of the PAJÉ file format