

Log Event Context

Jim Christopher
@beefarino

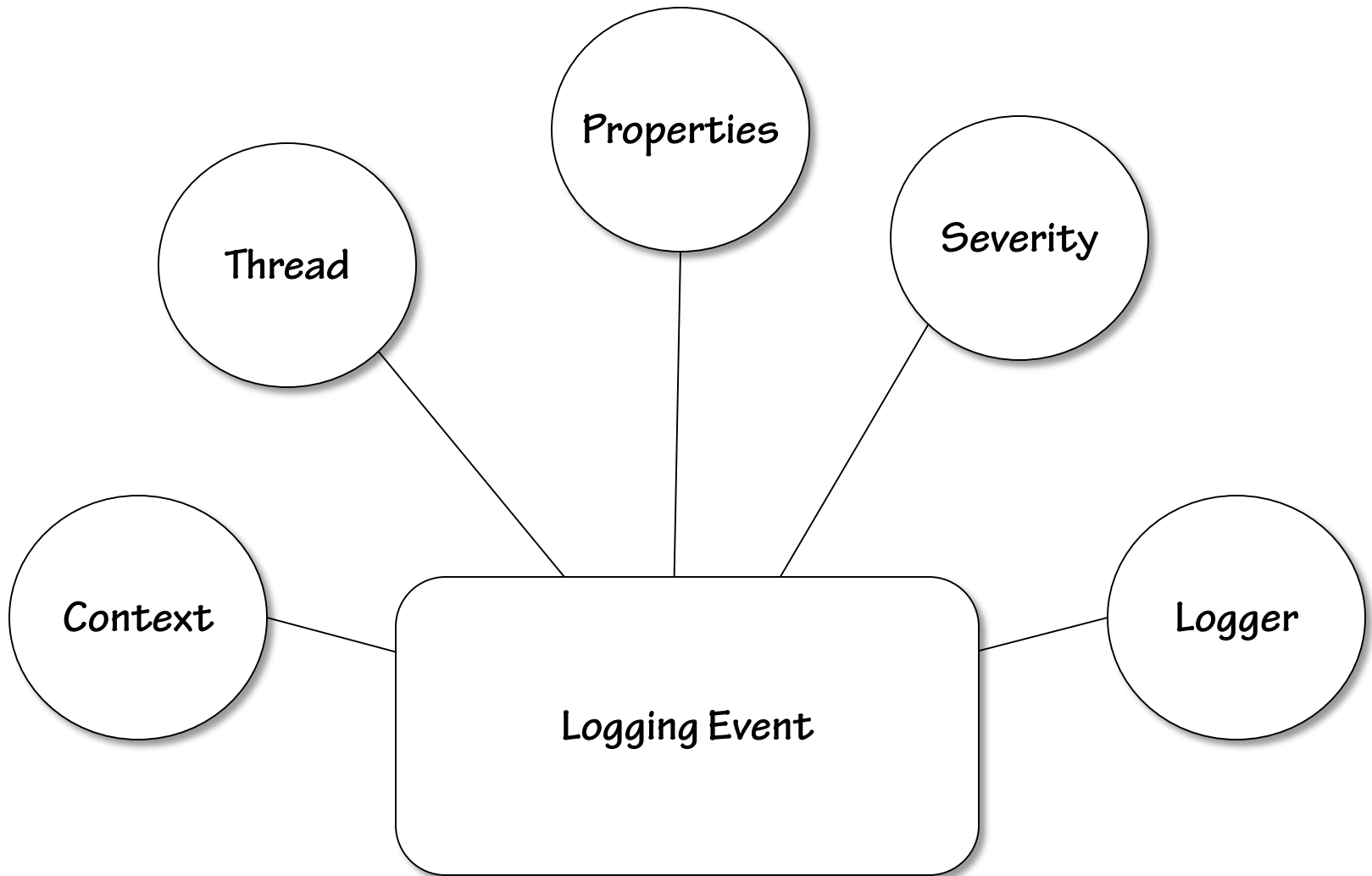


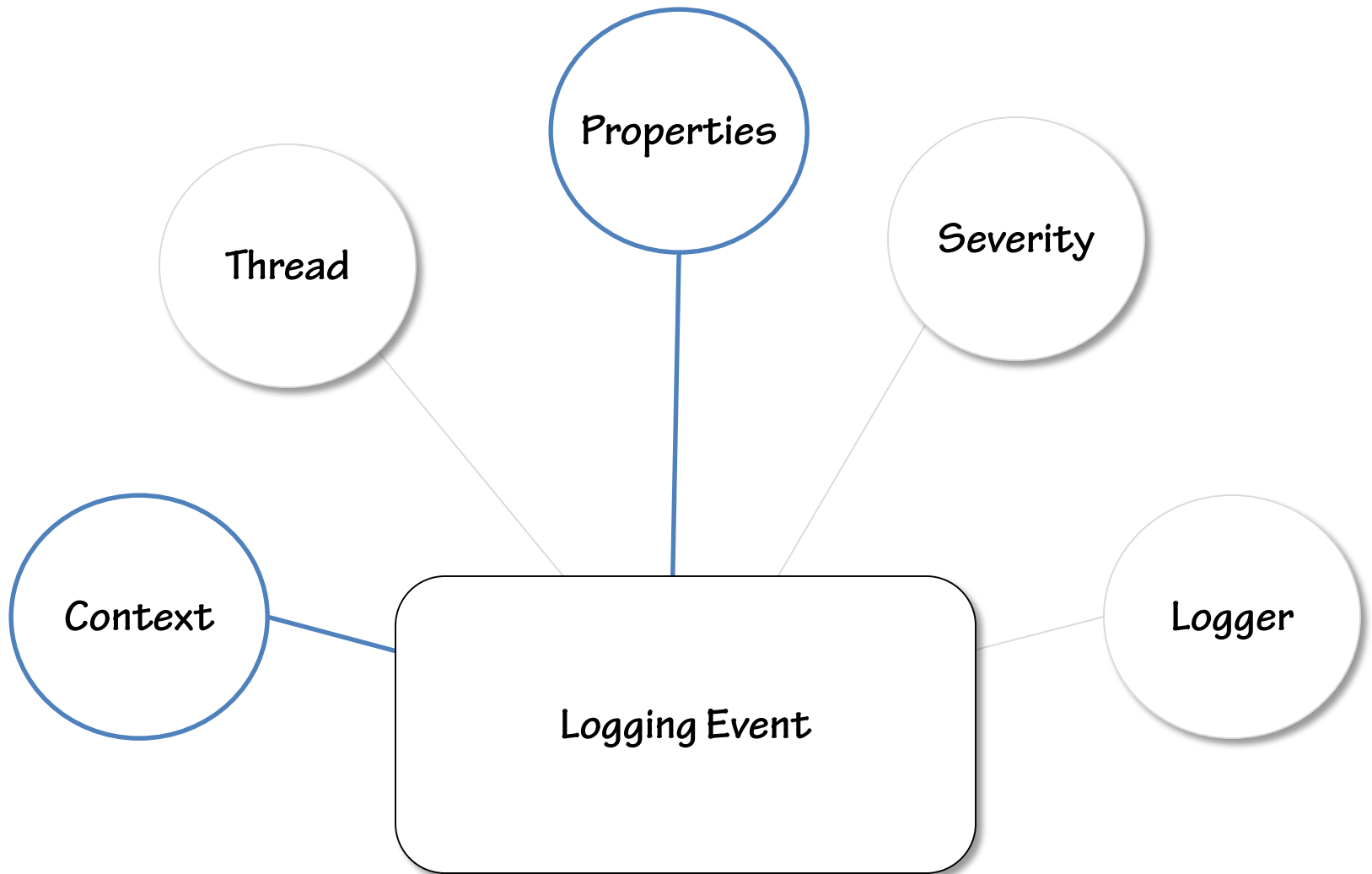


Logger

The diagram consists of two main components. On the left is a solid red rounded rectangle labeled 'Logger'. To its right is a white rounded rectangle with a black border, labeled 'Logging Event'. A small white triangular pointer extends from the right side of the red rectangle towards the left side of the white rectangle, indicating a relationship or flow between the two.

Logging Event





```
graph TD; A((Current CPU Utilization)) --> D(Logging Event); B((Number of Active Threads)) --> D; C((Ambient Temperature)) --> D;
```

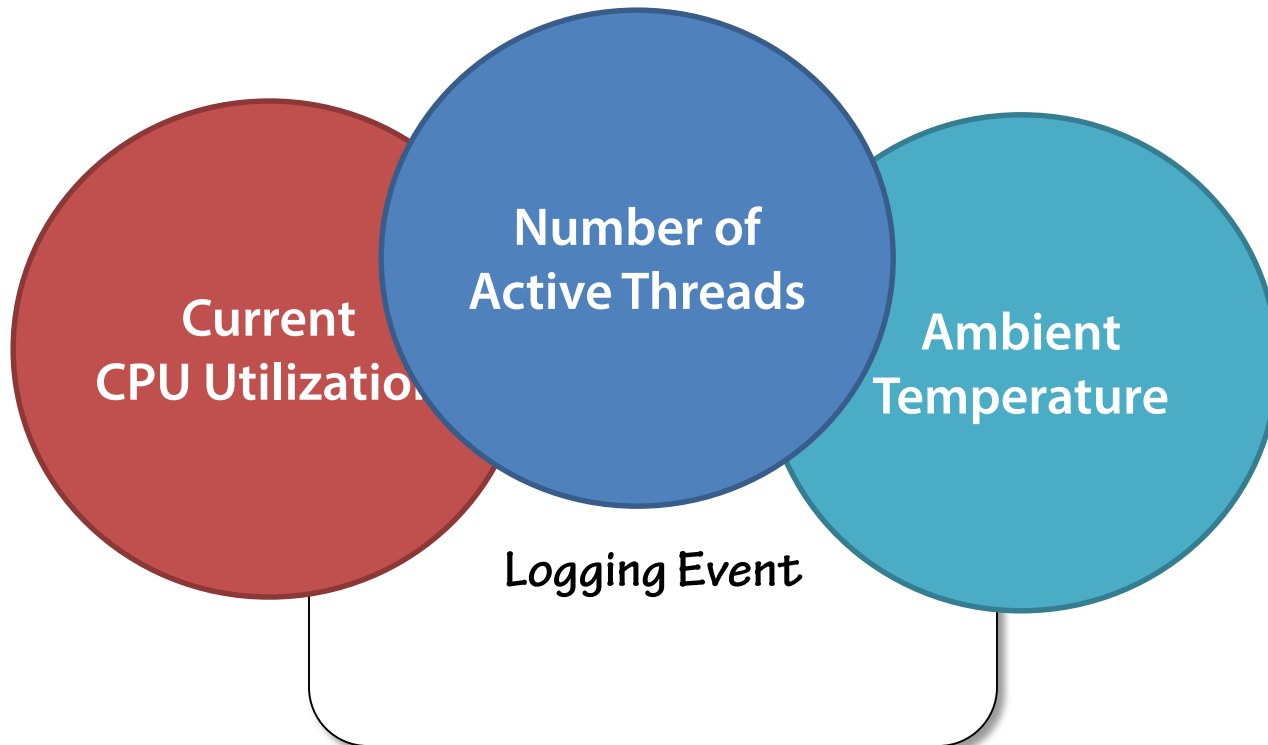
**Current
CPU Utilization**

**Number of
Active Threads**

**Ambient
Temperature**

Logging Event

Appender





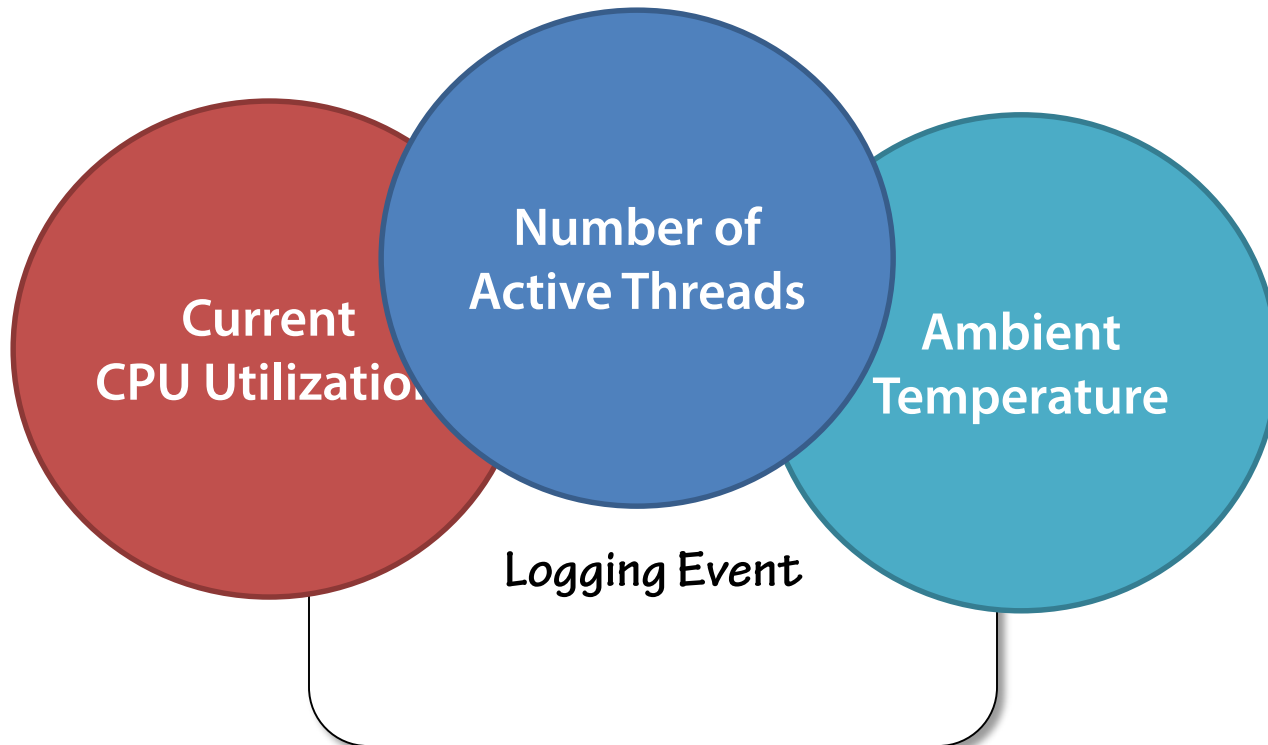
CURRENT CPU LOAD: 88%

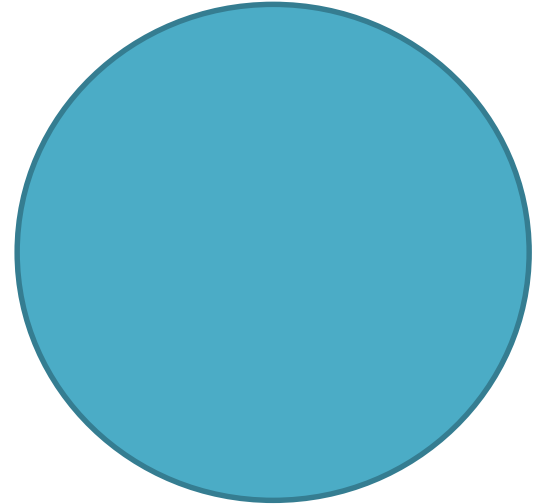
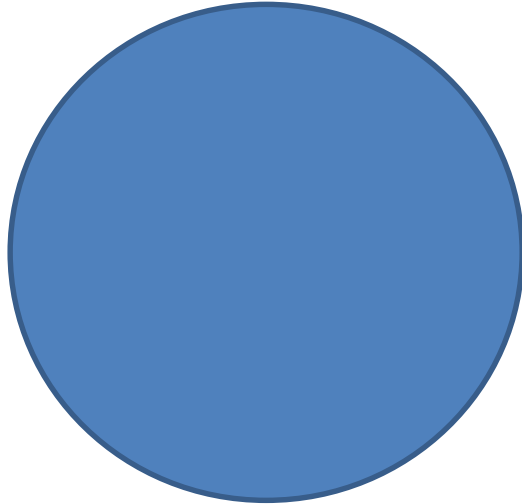
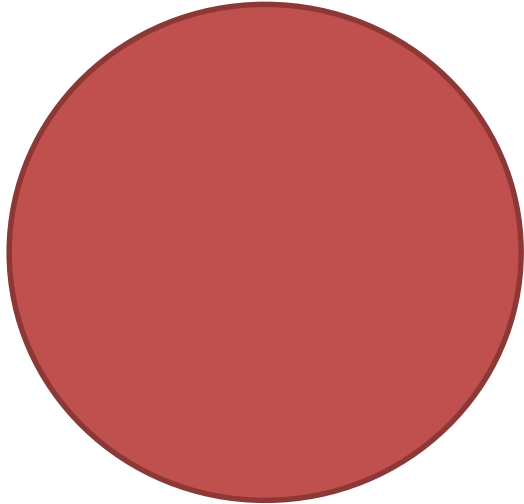
ACTIVE THREAD COUNT: 51

AMBIENT ROOM TEMPERATURE: 29C

Property Format Specifier

`%property{name}`





Logging Event



**Global
Context**

**Thread
Context**

**Logical
Thread
Context**

Global
Context

Thread
Context

Logical
Thread
Context

<http://bit.ly/13Dwu66>



Global
Context

Thread
Context

Logical
Thread
Context



**Global
Context**

**Thread
Context**

**Logical
Thread
Context**



**Global
Context**

**Thread
Context**

**Logical
Thread
Context**

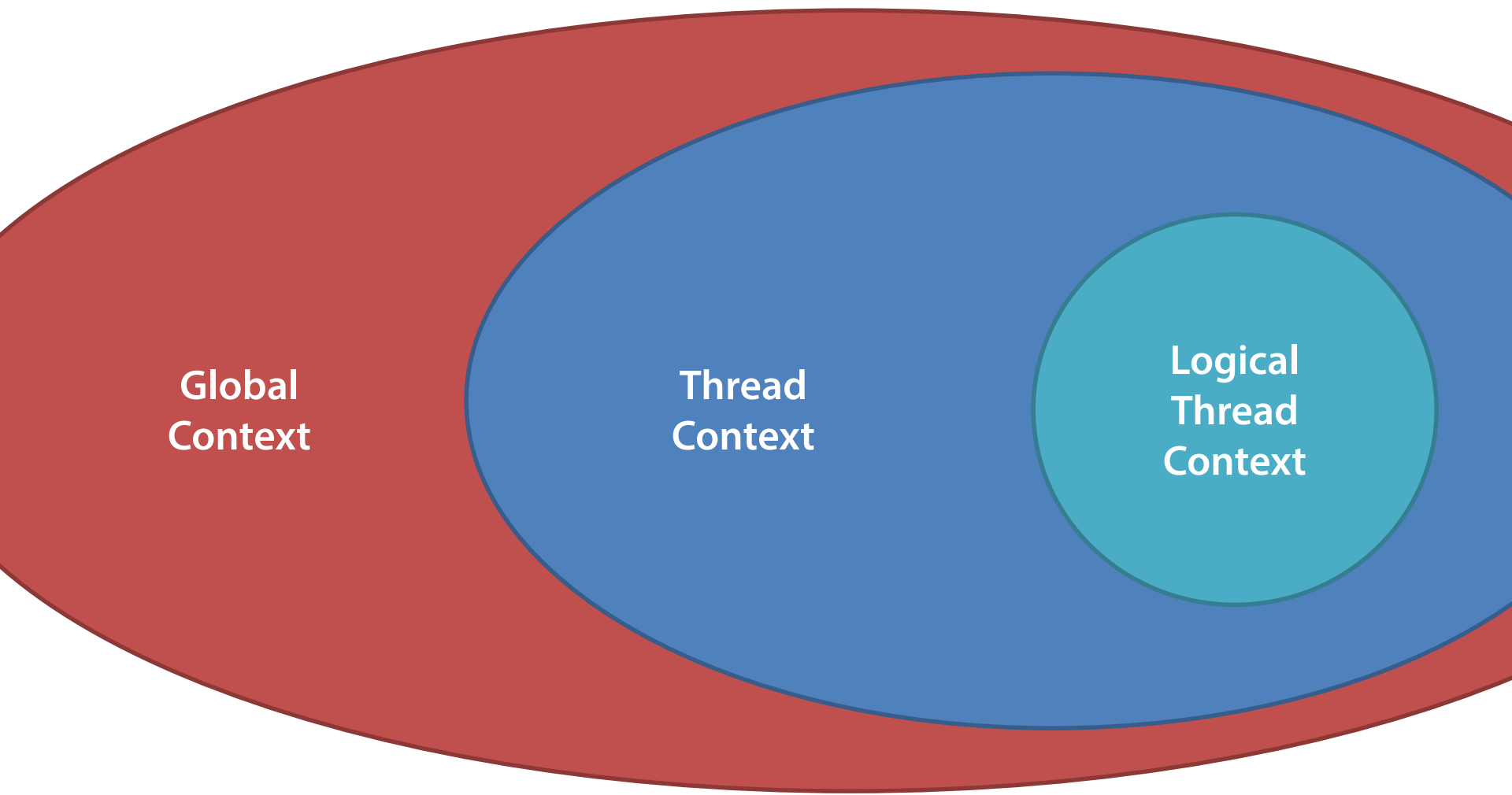


The diagram illustrates the relationship between three levels of context. On the left is a red circle labeled 'Global Context'. To its right is a large blue oval labeled 'Thread Context'. Inside the blue oval is a smaller teal circle labeled 'Logical Thread Context'. This visualizes that the Logical Thread Context is contained within the Thread Context, which in turn exists within the Global Context.

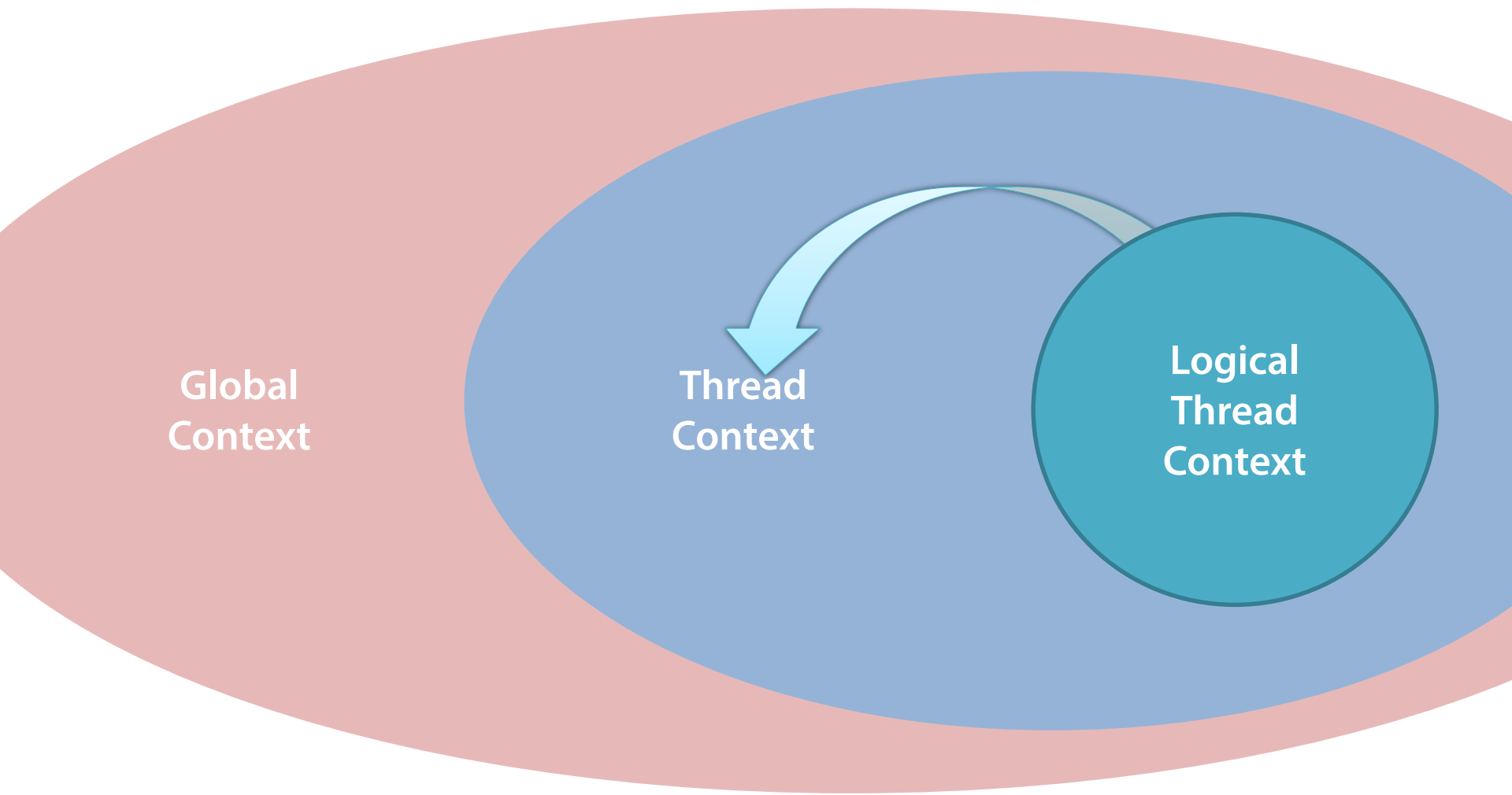
Global
Context

Thread
Context

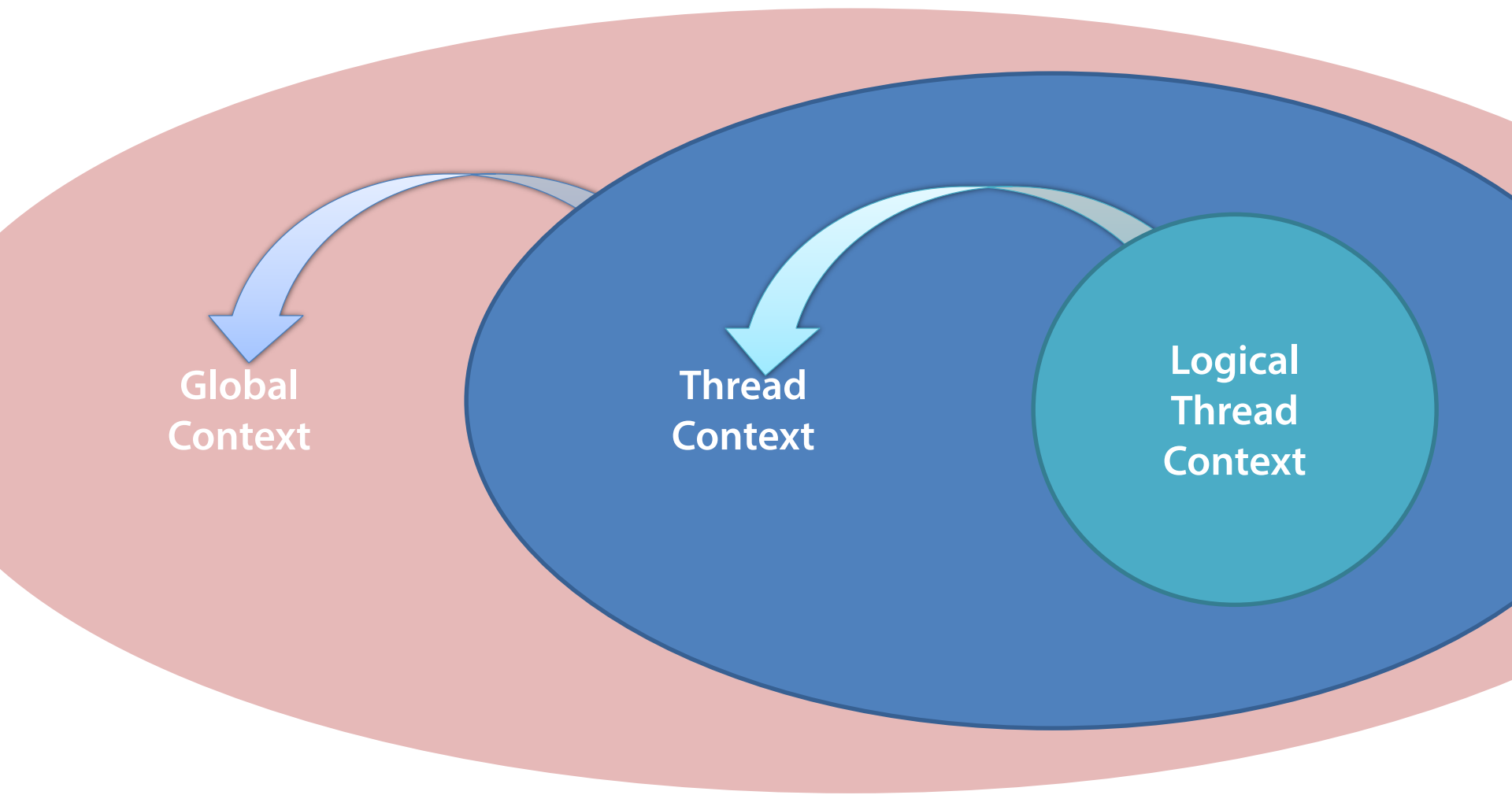
Logical
Thread
Context



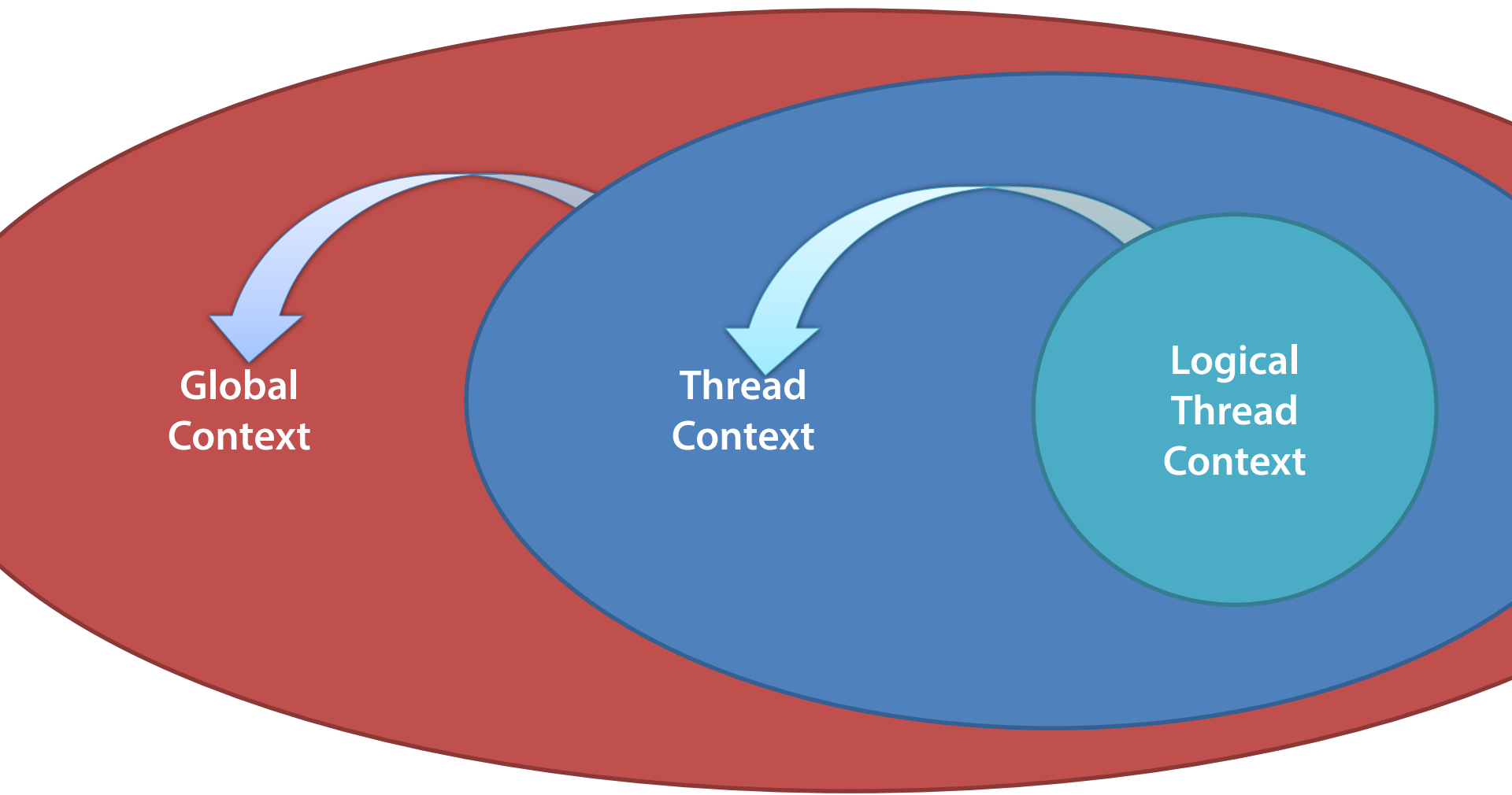
`%property{foobar}`



`%property{foobar}`



`%property{foobar}`



`%property{foobar}`

ThreadContext.Properties["myProperty"] = "A"

↳ ThreadContext.Properties["myProperty"] = "B"

↳ ThreadContext.Properties["myProperty"] = "C"

%property{myProperty}

C

B

A

```
ThreadContext.Stacks["myProperty"].Push("A")
```

```
└─ ThreadContext.Stacks["myProperty"].Push("B")
```

```
└─ ThreadContext.Stacks["myProperty"].Push("C")
```

%property{myProperty}

C

B

A

```
ThreadContext.Stacks["myProperty"].Push("A")
```

```
└─ ThreadContext.Stacks["myProperty"].Push("B")
```

```
└─ ThreadContext.Stacks["myProperty"].Push("C")
```

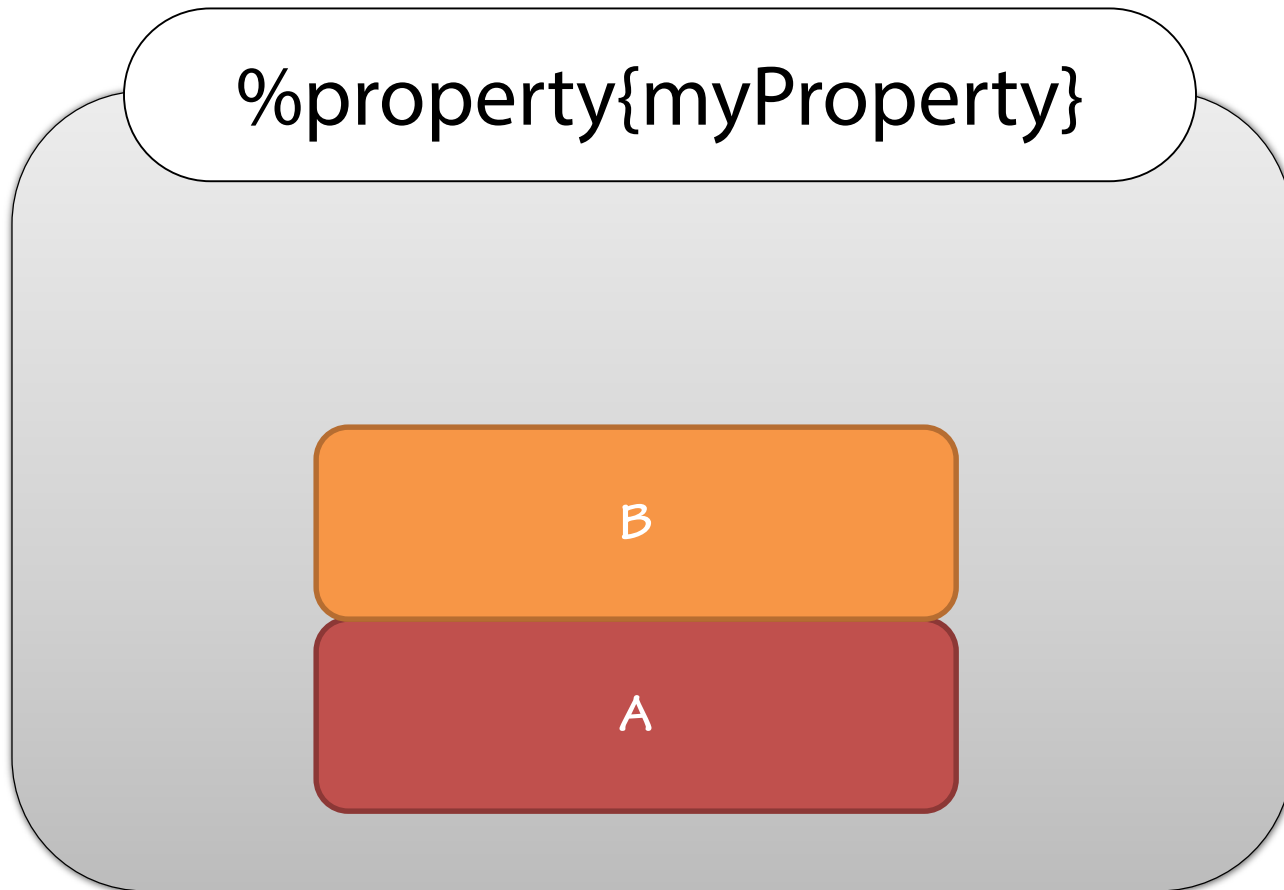
%property{myProperty}

C

B

A


```
ThreadContext.Stacks["myProperty"].Push("A")  
↳ ThreadContext.Stacks["myProperty"].Push("B")
```




```
ThreadContext.Stacks["myProperty"].Push("A")
```



GlobalContext["Value"] = "Value"



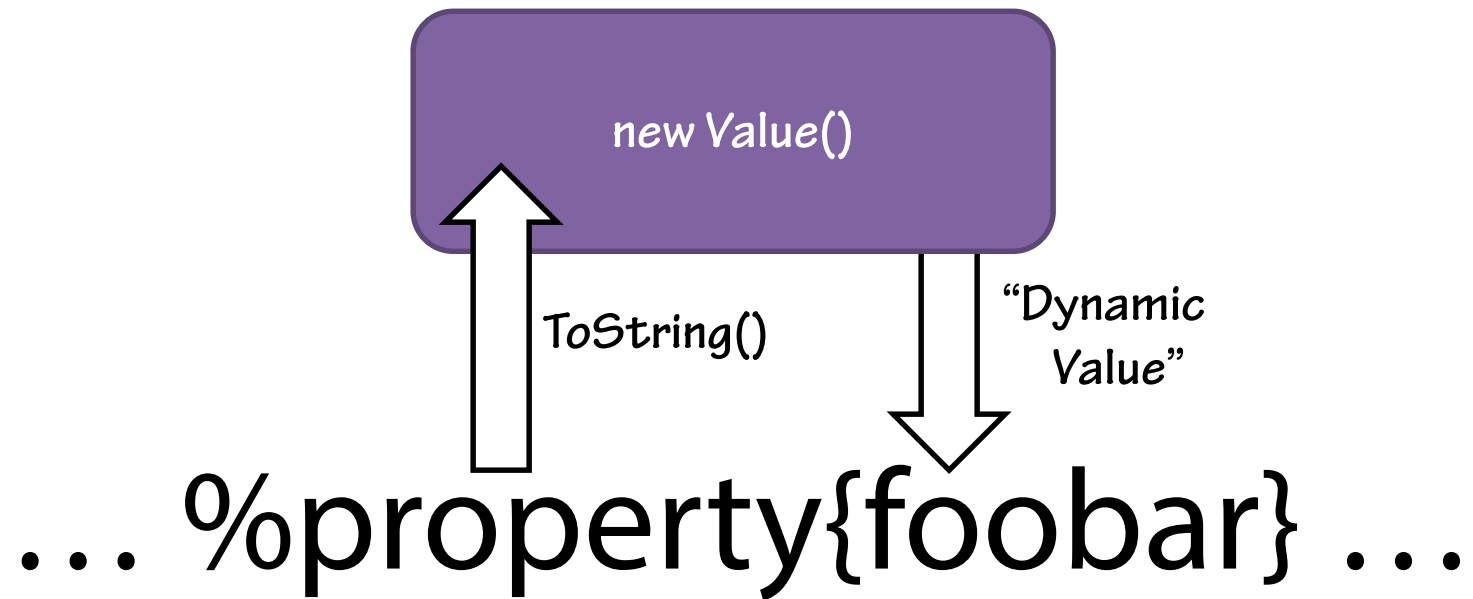


"Value"

... %property{foobar} ...

... Value ...

```
GlobalContext.new Value()"] = new Value()
```



Log Event Context

- Custom Log Event Properties
 - %property{name}

Log Event Context

- Custom Log Event Properties
 - %property{name}
- Log Event Contexts
 - Global
 - Thread
 - Logical Thread

Log Event Context

- **Custom Log Event Properties**
 - %property{name}
- **Log Event Contexts**
 - Global
 - Thread
 - Logical Thread
- **Context Stacks**
 - Thread Context
 - Logical Thread Context

Log Event Context

- **Custom Log Event Properties**
 - `%property{name}`
- **Log Event Contexts**
 - Global
 - Thread
 - Logical Thread
- **Context Stacks**
 - Thread Context
 - Logical Thread Context
- **Calculated Context Properties**
 - `Override ToString()`