Troubleshooting

Rules

4/7/2025

Rules

Working with simultaneous events

Sequence patterns do not always work in cases where multiple events for the same visit are sent simultaneously.

A sequence pattern is described by the following construction:

```
$event2: Event(eval($event2.getName().equals('...')) && this after $event1)
```

The Web Engagement rules engine can only process patterns like this one if **event2** arrives at least 1 millisecond later than **event1**. However, simultaneously generated events may arrive in the same millisecond, which means that the pattern will not work.

Under a rare set of circumstances, multiple events might be sent simultaneously for the same visit. If this happens in your environment, you need to modify the default GRDT templates, changing your sequence pattern to the following form:

```
$event2: Event(eval($event2.getName().equals('...')) && this after[0ms] $event1)
```

Prevent OutOfMemory exceptions when using multiple conditions on the same event

GRAT only allows you to use 5 parameters in a single conditional statement when creating a linear rule. You may want to work around this limitation and apply more conditional statements to the same event, so that the total number of applied parameters will be greater than 5. In this case, you can use additional **eval** statements, as explained below.

The default Web Engagement GRDT templates contain conditional statements that look something like this:

```
$event2: Event(eval($event2.getName().equals('MyEvent')) && this after $event1)
```

This statement uses the **Event** keyword when evaluating the nested expressions. However, if you use more than 1 conditional statement that uses the **Event** keyword, the Web Engagement rules engine might produce a large number of fake "activation" objects that could lead to OutOfMemory exceptions.

Here is an example of a set of statements that can cause this issue, because each of them uses this keyword:

```
$event1: Event(eval($event1.getName().equals('MyEvent')))
Event(eval($event1.getDataPropertyValue('XXX') != null ))
...
Event(eval($event1.getDataPropertyValue('XXX') != null ))
```

Fortunately, you can use **eval** conditions without needing the **Event** keyword, as shown here:

```
$event1: Event(eval($event1.getName().equals('MyEvent')))
eval($event1.getDataPropertyValue('XXX') != null )
...
eval($event1.getDataPropertyValue('XXX') != null )
```

Since only the first statement uses the **Event** keyword, the rules engine will not create fake "activation" objects, meaning that you can avoid this type of OutOfMemory problem.