LUX Event Query Syntax Guide

Last Update: 11/20/17

LUX Event Queries

LUX Event Query strings are very similar to XPaths, applied to json instead of xml.

Differences Between Alert JSON and Engine Representation

The internal JSON representation of events is slightly different from what appears in the alert json:

- In the alert json, the 'attributes' object contents is represented like this:

```
"attributes": [
      {
        "name": "myAttribute",
           "value": "someValue"
      },
      {
        "name": "myAttribute",
           "value": "someValue2"
      },
      {
           "name": "myOtherAttribute",
           "value": "someOtherValue"
      }
]
```

  In the JSON used internally by the engine, it's represented like this:

```
"attributes": {
      "myAttribute": ["someValue", "someValue2"],
      "myOtherAttribute": "someOtherValue"
}
```

- In the alert json, properties appear inside the events in a "properties" field. This might appear to be the query for accessing the “myProperty” property if you look at the alert json:

```
/properties/myProperty
```

  That won’t work, however, because the properties aren’t actually part of the event JSON document in the engine. Inside the engine, properties are an entirely separate structure that isn't part of the event JSON, and they have their own special syntax for accessing them:

```
${myProperty}
```

Note: Event queries don't distinguish between event-level properties (added by enrichments) and event-list-level properties (added by analytics). `${myProperty}` would give all the values of any property by that name associated with the event or the event-list containing it.

- In the alert json, geometry properties are added to the "geometries" field of the event list; these properties don't appear in the "properties" field like other properties. Inside the matching-engine, `/geometries` only selects geometries that were part of the original event produced by the ingest. Geometries added as properties would be selected via `${myGeometryProperty}/`

Combining Queries

| and || can be used to combine values from multiple queries. The | operator selects the union of multiple queries, the || operator selects from the first query that returns a non-empty result.

This query would return all values of the text attribute and all values of the common_text property:

```
/attributes/text | ${common_text}
```

This query would return all values of the text attribute if there are any. If there aren't, it would return all values of the common_text property:

```
/attributes/text || ${common_text}
```

This query would search, in order, for a text attribute, a common_text property, a content attribute, or a translated_text property, and return the values of the first one of those queries that had any values to return:

```
/attributes/text || ${common_text} || /attributes/content ||
${translated_text}
```

Handling JSON field names that wouldn’t work in XPath

When attributes have names containing certain punctuation or whitespace, they need to be wrapped in single or double quotes:

```
/attributes/'my.attribute.name'
```

A string enclosed in quotes by itself is interpreted as a string literal, so if an element needing quotes is to be referenced in the first part of a predicate, it needs a ./ for disambiguation. This query won’t ever return anything because the literal string "my.attribute.name" never equals "value1":

```
/attributes["my.attribute.name" = "value1"]/otherAttribute
```

To correct it, prefix “my.attribute.name” with ./ like this:

```
/attributes[./"my.attribute.name" = "value1"]/otherAttribute
```

Examples

LUXEvent attributes

- `/attributes/myAttribute`

LUXEvent hits and hitAttributes

- `/hits[name = "myHitName"]/hitAttributes/myHitAttribute`
- `/hits[type = "myHitType"]/hitAttributes/myHitAttribute`
- `/hits[name = "myHitName"]/type`
- `/hits/hitAttributes/myHitAttribute`

Other LUXEvent fields

- `/eventId`
- `/eventDate`
- `/pubDate`
- `/title`
- `/publisher`
- `/source`
- `/type`
- `/link`
- `/description`
- `/rights`
- `/dataGroups`

LUXEvent Geometry

- `/geometries`

LUXEvent Properties

- `${myProperty}`

Geometries added as properties

- `${myGeoProperty}/`
- `${myGeoProperty}/properties/styleName`
- `${myGeoProperty}/[properties/confidence >= 50.0]`
- `${myGeoProperty}/[properties/confidence > 20.0 and properties/confidence < 40.0]`

Geometries added as properties (legacy xml style)

- `${myGeoProperty}/georss:where`

Default Values

- `${language} || “unknown”`