Data Field Naming for Events Service 4.5.3 and Above

On this page:

This document explains how to update data field names.

To use this procedure appropriately for your deployment, follow the guidelines below.

Deployment	Version	Required Action
SaaS	4.5.3 and later	Update data field names as described below.
On-premises	4.5.2 and older	No action is required, but AppDynamics strongly recommends that you update data field names to prepare for future Events Service releases.

Update Overview

Version 4.5.3 of the Events Service runs Elasticsearch 5.6, whereas previous Events Services run earlier versions of Elasticsearch. To upgrade to Events Service 4.5.3, you may need to rename some fields used in collecting transaction analytics, log analytics, or other types of data.

Review the requirements below and follow instructions where applicable. To understand the rationale for the changes, see More About Field Names.

Requirements

Do Not Use Empty Field Names

In pre-5.6 versions of Elasticsearch it was possible to create fields with empty names. This is no longer allowed. Empty field names now cause indexing errors.

Required action: Give alphanumeric names to any fields whose names are empty.

Do Not Use Dots in Field Names

In the past, some customers have used dots to separate name components in a semantically meaningful way. This is no longer recommended and may cause the upgrade to Events Service 4.5.3 to fail.

Strongly recommended action: Replace dots in field names with hyphens or underscores.

More About Field Names

This section discusses dotted field names, meaning field names with embedded periods ('`.`'), such as `a.b.c` or `transit.signals.yellow`.

Elasticsearch stores data in JSON documents whose structure is hierarchical. Dotted field names can be used to query into those JSON documents: the field name is treated as a path whose components are separated by dots. See https://www.elastic.co/guide/en/elasticsearch/reference/2.4/dots-in-names.html.

It can be impossible to know whether a dotted field name is intended as a path to a JSON element, or just a plain field name. To treat this problem in a consistent way, Elasticsearch, beginning with version 5.6, always automatically expands dotted field names into hierarchical JSON structures. Each dot creates another level nested lower in the hierarchy.

Events Service 4.5.3 attempts to gracefully handle dots in field names and allow the default behavior of Elasticsearch. However, in a few corner cases, Elasticsearch still fails to index events to which field names correspond. In these situations, the only recourse is to change the field names.

These corner cases can be avoided by following three rules:

1. Field names cannot contain multiple consecutive dots
2. Field names cannot start or end with dots
3. Field names cannot share prefixes

The rest of this section explains these rules.

Field Names Cannot Contain Multiple Consecutive Dots

Field names that contain multiple consecutive dots expand into structures where the name of some elements is the empty string. These are invalid as JSON objects. Note the empty names of the most deeply nested nodes in the following example:

```
"a.very.long.field.name.truncated.with.dots..." ->
"a": {
  "very": {
    "long": {
      "field": {
        "name": {
          "truncated": {
            "with": {
              "dots": {
                "": {
                 "": {
                  "": {
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Field Names Cannot Start or End with Dots

Field names that start or end with dots expand into structures where the name of some elements is the empty string. These are invalid as JSON objects. For example:

```
".a.b.c" ->
"": {
  "a": {
    "b": {
      "c": "value of field"
    }
  }
}


"a.b.c." ->
"a": {
  "b" : {
    "c": {
      "" : {
      }
    }
  }
}
```

Field Names Cannot Share Prefixes

When two or more field names have the same prefix, it becomes impossible to create valid JSON objects for them all.

Consider the following field names and values:

```
a.b = "alphabaker"

a.b.c = "alphabakercharlie"
```

Trying to share a prefix runs into trouble because:

- all the nodes that need to be created are text nodes, and
- some nodes need to be nested, but
- in JSON, text nodes are not allowed to contain other objects.

We'll demonstrate the problem by examining what happens when Elasticsearch tries to create the JSON objects for our example.

- The first field name results in "b" being mapped to a text node with the value "`alphabaker`."

```
"a": {
  "b": = "alphabaker"
}
```

- To expand the second name, Elasticsearch tries to map "c" to a text node with the value "`alphabakercharlie.`" This fails because "`c` " needs to be nested within "`b,`" which is a text node and cannot contain nested objects.