Приемы парсинга событий

Парсинг нестандартной даты

Дата Время

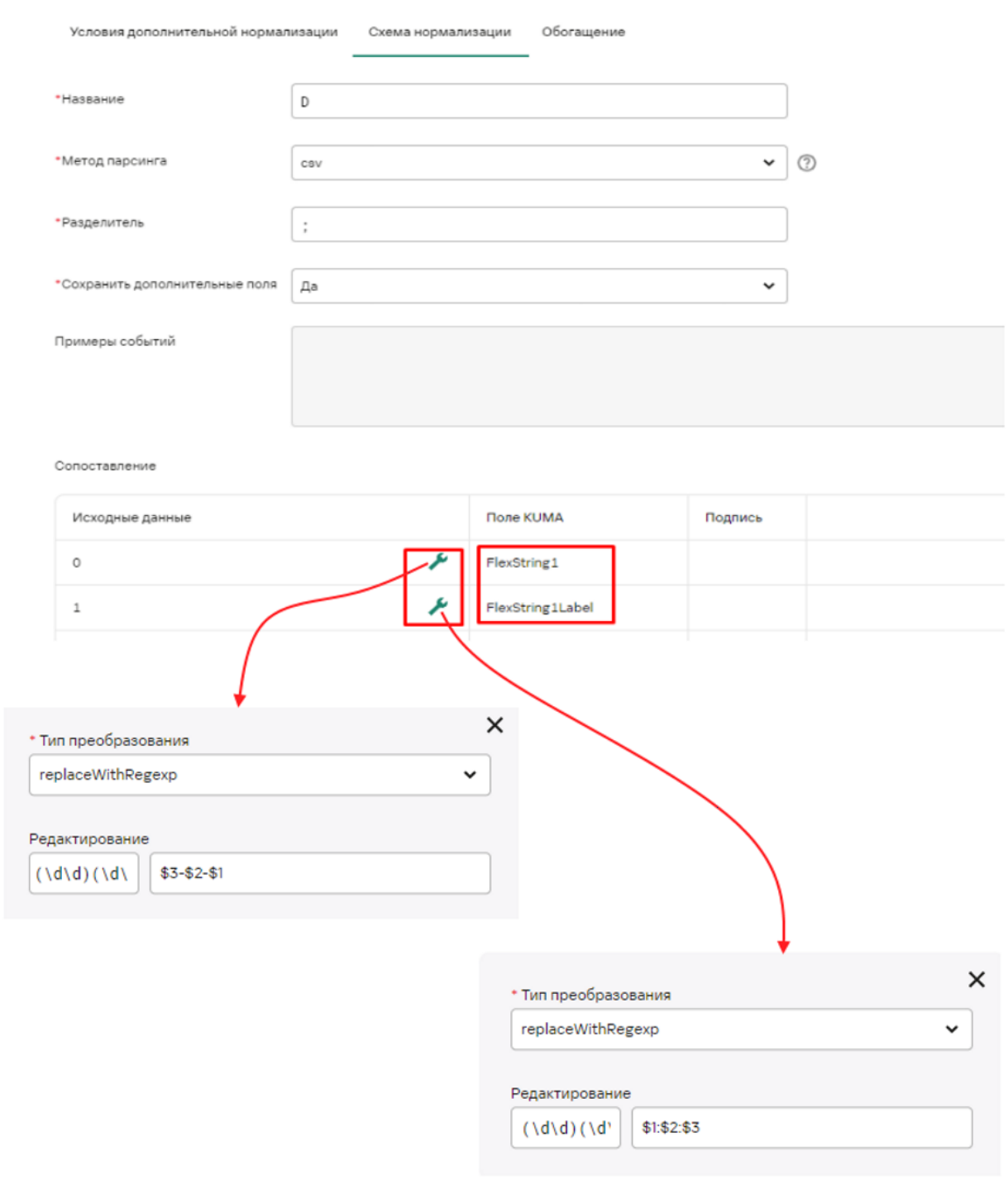
#D#
220523;080050;5522.5957;N;08605.0047;E;0;65;142.400000;22;NA;NA;NA;NA;NA;flg:1:7,mileage:2:1097.000000,timp:1:0,pwr_ext:2:13.700000,battery:1:41,sum_acc:2:1.140000,amtr_x:1:2,wln_accel_max:2:0.020000,wln_brk_max:2:0.000000,amtr_y:1:0,wln_crn_max:2:0.000000,amtr_z:1:0,hours_koef:1:16777216,tlls1:1:0,clls1:1:0,flls1:1:3,mid:1:1,cmd:1:149

Нужно привести к такому виду:

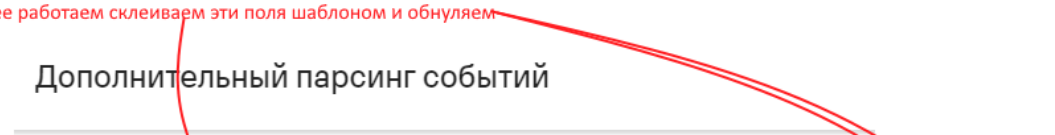
ISO8601 (Syslog timestamp)
[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}(\.[0-9]+)?([zZ]|([\+-])([01]\d|2[0-3]):?([0-5]\d)?)

2020-03-12T13:34:56.123Z INFO [org.example.Class]: This is a #simple #logline containing a 'value'.

Как выглядит парсер:



Далее работаем склеиваем эти поля шаблоном и обнуляем



Ветвление событий от beats в зависимости от input типа

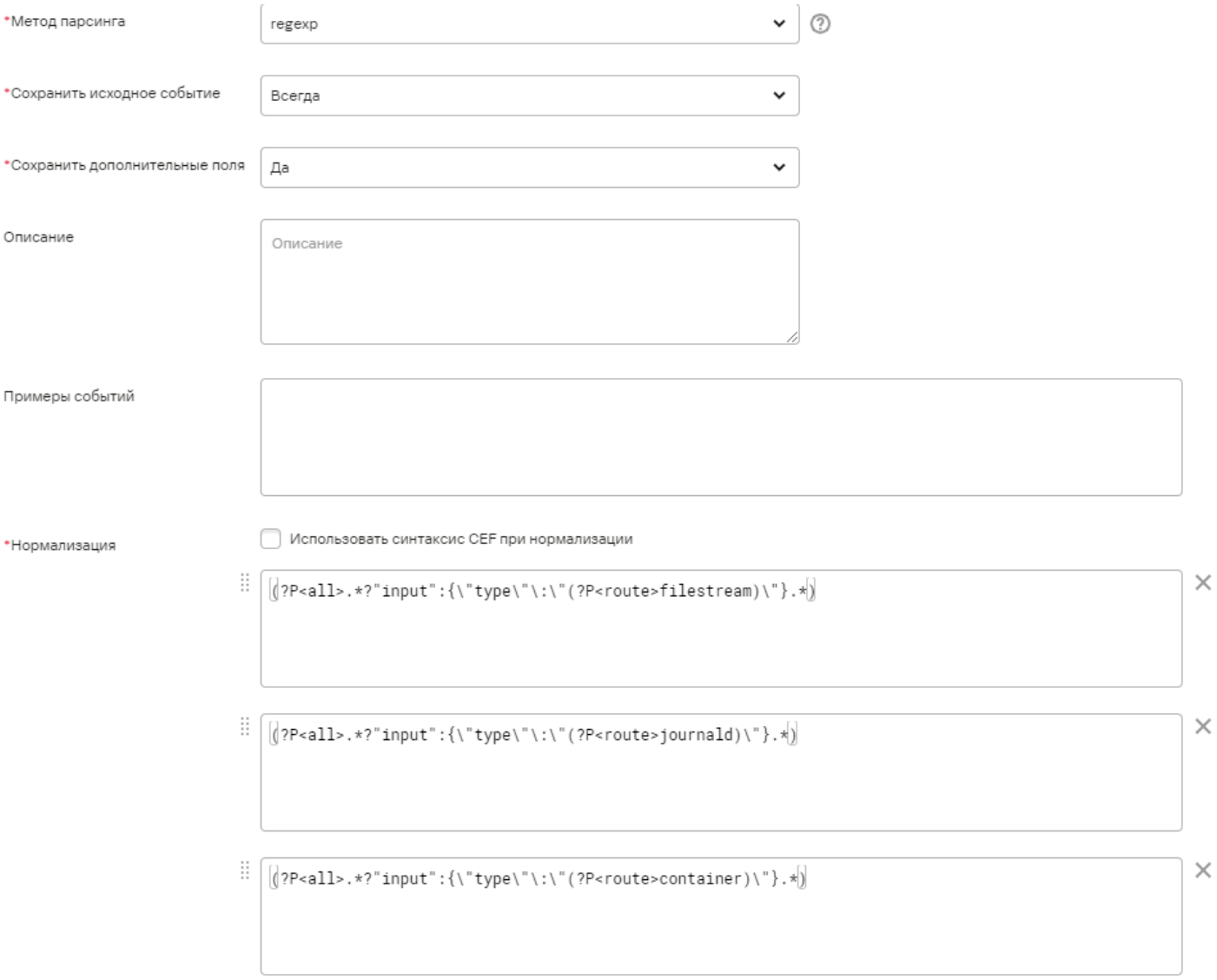
Даны следующие типы событий (содержимое тестового сообщения сокращено для лучшего понимания):

```
{"tags":["beats_input_raw_event"],"input":{"type":"filestream"}}
{"message":"I0130 14:38:47.090079 1837403 utils.go:187] ID: 544472 GRPC response:
{}","input":{"type":"container"}}
{"journal":{"system":"true"},"tags":["beats_input_codec_plain_applied"],"input":{"type":"journald"}}
{"input":{"type":"journald"},"journal":{"system":"true"},"tags":["beats_input_codec_plain_applied"]}
{"journal":{"system":"true"},"input":{"type":"journald"},"tags":["beats_input_codec_plain_applied"]}
```

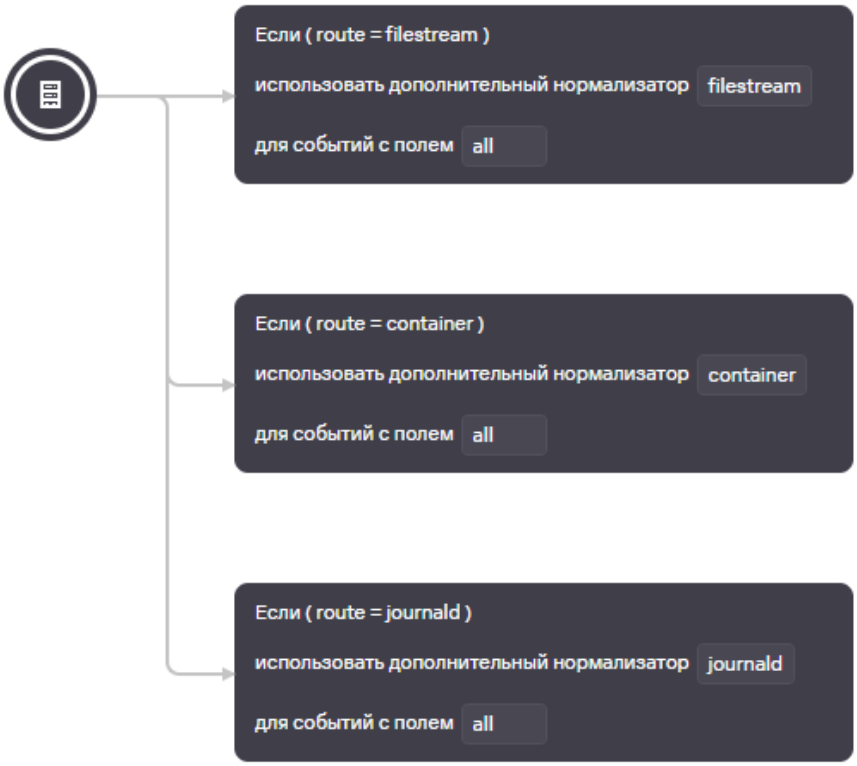
Необходимо в парсинге разветлять (тк у каждого типа свой набор полей) парсинг в зависимости от типа input поля, мы имеем три типа в данном примере:

- "input":{"type":"container"}
- "input":{"type":"journald"}
- "input":{"type":"filestream"}

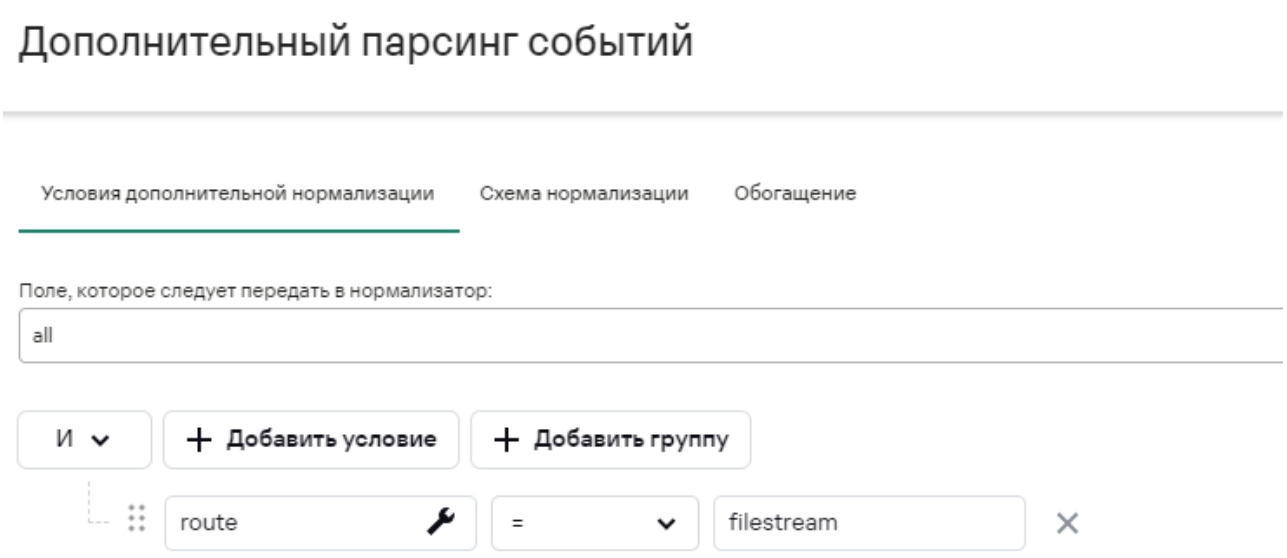
Причем, поле input может находиться как в начале, так и в середине, и в конце сообщения. Поэтому для ветвления в первом шаге парсинга будут использоваться регулярные выражения:



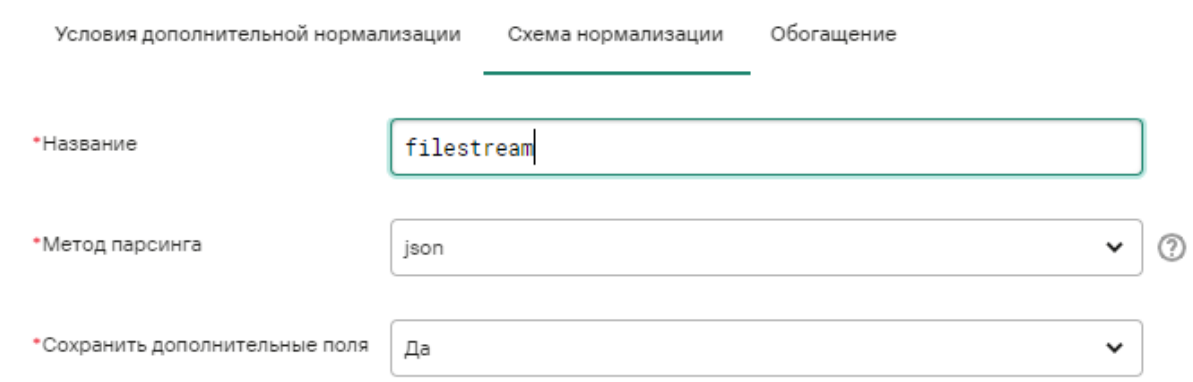
Поле из regex с наименованием route будет использоваться для маршрутизации по условию в нужный парсер, поле all необходимо для передачи полного содержимого в подпарсер. Структура парсера выглядит следующим образом:



Рассмотрим один подпарсер, например, filestream:



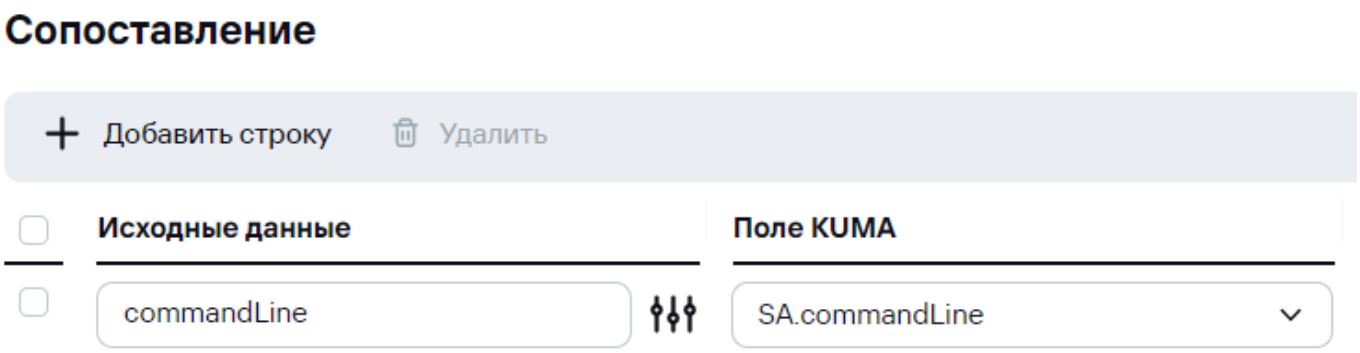
Тк общая структура сообщения формата JSON, используется соответсвующий коробочный парсер:



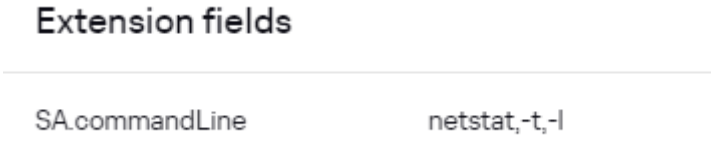
Парсинг массивов

> Актуально для KUMA 3.0+

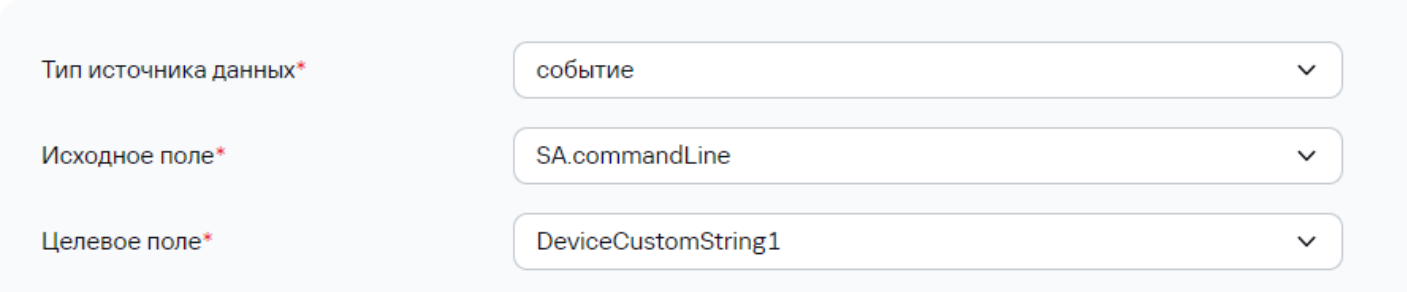
В KUMA 3.0.2 появилась возможность создания кастомных полей типа "массив" (SA, NA, FA), доступные для методов парсинг JSON и KV. Чтобы записать массив в дополнительное поле, достаточно его указать в маппинге:



В событии это будет выглядеть следующим образом:



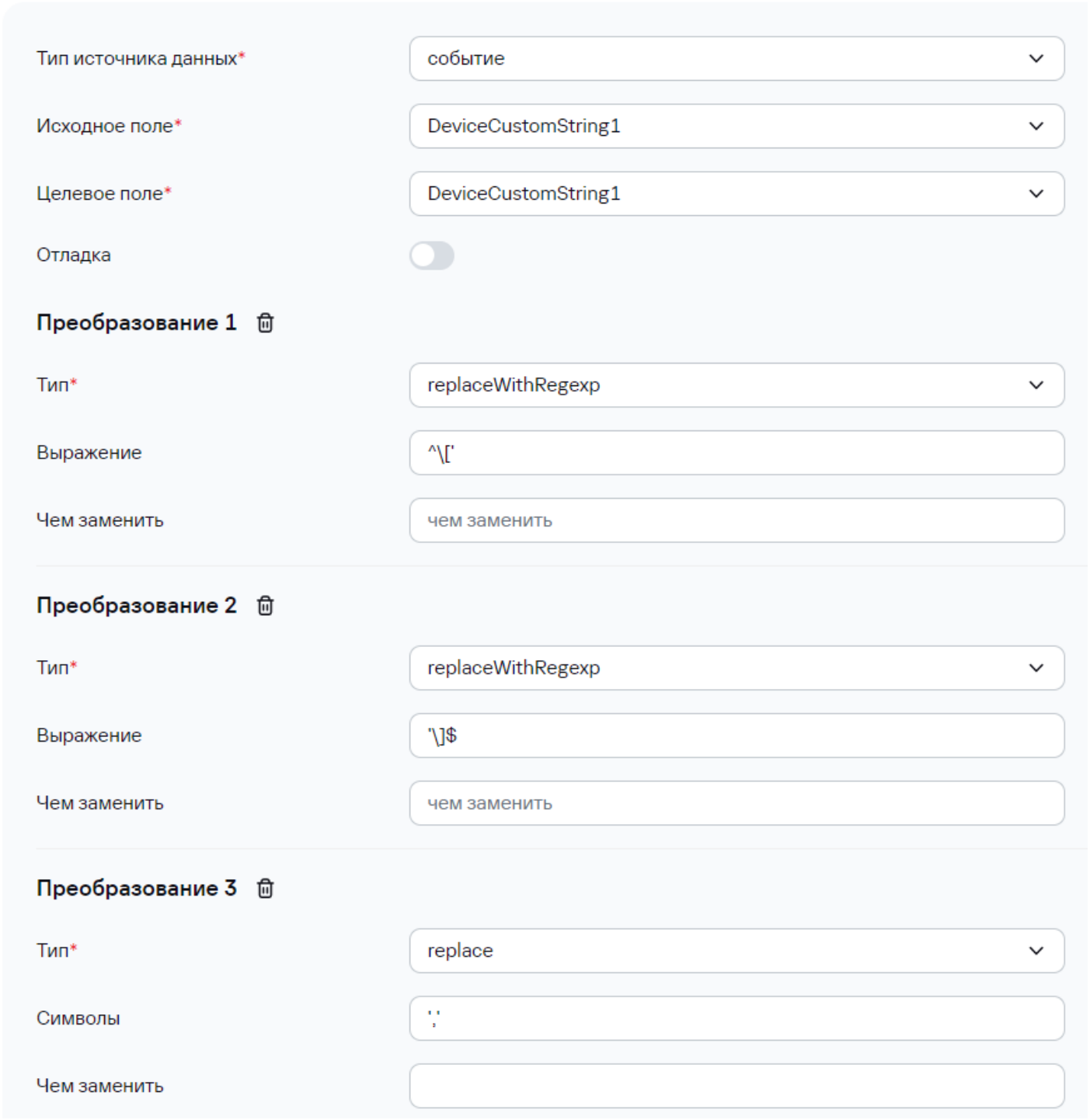
Если с массивом в таком случае работать не удобно и нужно все элементы из массива "склеить" через делиметр и записать в отдельное поле, можно воспользоваться обогащением. Для этого сначала массив мапится на строковое поле:



В таком случае в событии данное поле будет представлять собой массив переведенный в строку:

DeviceCustomString1	['netstat','-t','-l']

Чтобы привести ее в более "приятный" вид можно выполнить следующие преобразования:



После этого в DeviceCustomString1 будут записаны все элементы массива через выбранный в последнем (3) преобразовании делиметр (в данном примере это "пробел"):

DeviceCustomString1	netstat -t -l

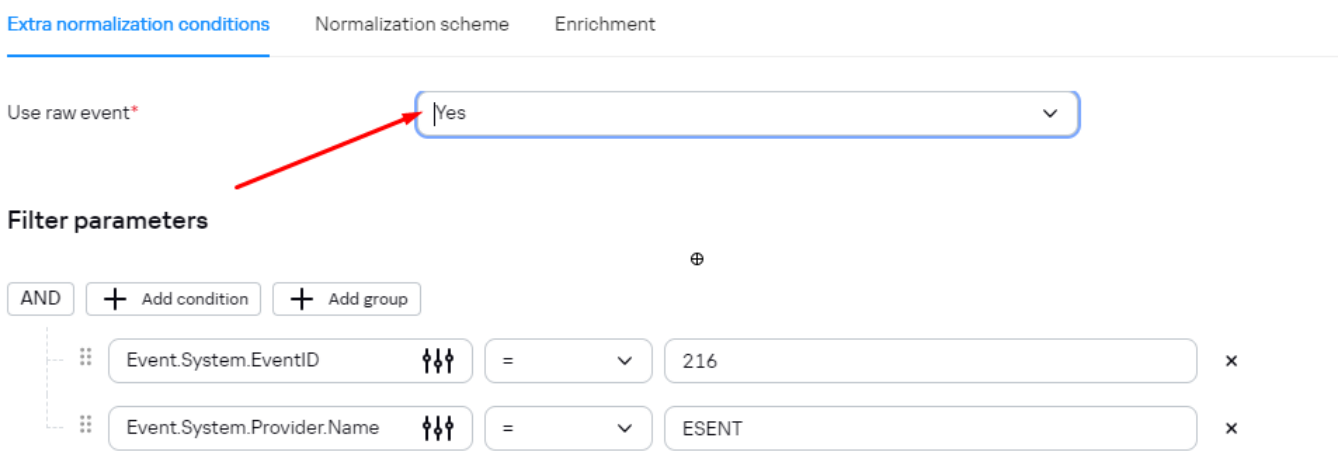
Передача сырого события в экстранормализатор, для доступа к элементам массива

> Актуально для KUMA 3.0+

Для передачи «сырого» события в экстра-нормализатор необходимо:

- открыть нормализатор событий;
- перейти в меню «Условия дополнительной нормализации»;
- активировать параметр «Использовать сырое событие».

> По умолчанию параметр «Использовать сырое событие» не активен.



Рекомендуется активировать параметр «Использовать сырое событие» в нормализаторах типа **«xml», «json»**.

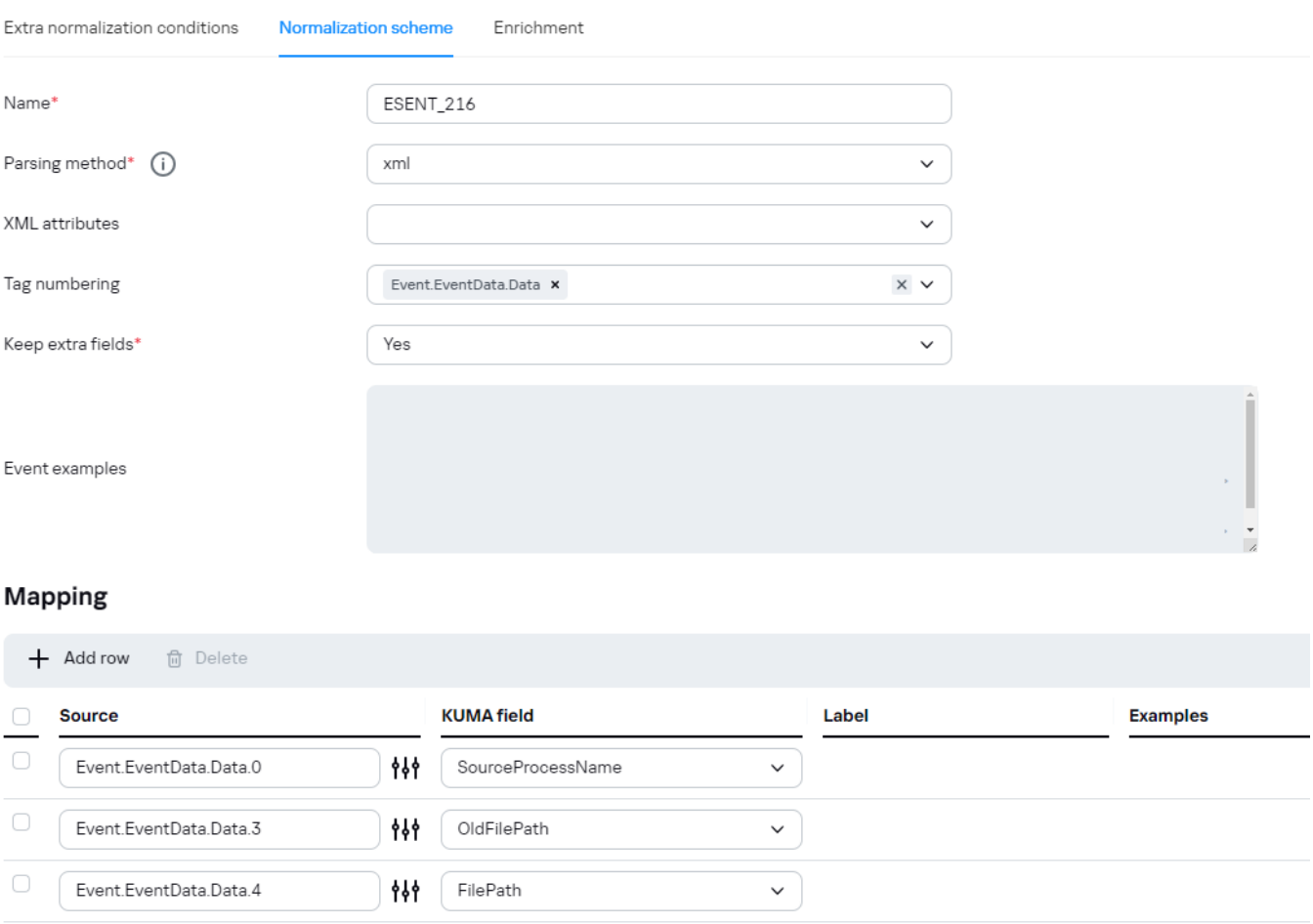
Для передачи «сырого» события в экстра-нормализатор второго, третьего и более глубоких уровней вложенности необходимо последовательно включить параметра «Использовать сырое событие» в каждом экстра-нормализаторе по пути следования события в целевой экстра-нормализатор и непосредственно в целевом экстра-нормализаторе.

В качестве примера работы данной функции вы можете обратиться к нормализатору Microsoft Products для KUMA 3.0.1: параметр «Использовать сырое событие» включен последовательно в экстра-нормализаторах «AD FS» и «424».

В качестве примера, событие:

<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='ESENT'/><EventID Qualifiers='0'>216</EventID><Level>4</Level><Task>3</Task><Keywords>0x80000000000000</Keywords><TimeCreated SystemTime='2024-01-20T20:06:07.144730300Z'/><EventRecordID>870234</EventRecordID><Channel>Application</Channel><Computer>COMPANY.COM</Computer><Security/></System><EventData><Data>lsass</Data><Data>724,R,98</Data><Data></Data><Data>C:\Windows\NTDS\ntds.dit</Data><Data>\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy50\Windows\NTDS\ntds.dit</Data></EventData></Event>

При парсинге ID события 216:



Будет корректно разбираться:

SourceProcessName	lsass
Service	BorisTest (tcp/5577)
ExternalID	1276974
FilePath	\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy47 \Windows\NTDS\ntds.dit
OldFilePath	C:\Windows\NTDS\ntds.dit
Type	Base

Revision #9
Created 2 November 2023 08:26:27 by Boris Rzr
Updated 7 July 2024 08:28:47 by Koala