Endpoint grouping

Kaa allows for aggregating endpoints related to the same application into endpoint groups. The *endpoint group* represents an independent managed entity which is defined by the profile filters assigned to the group. Those endpoints whose profiles match the profile filters of the specific endpoint group become automatically registered as members of this group. There is no restriction for endpoints on having membership in a number of groups at a time.

Endpoint group profile filters are predicate expressions which define characteristics of group members (endpoints). These filters are executed against the endpoint profile to figure out whether or not the endpoint belongs to the group.

> **NOTE**
> Different profile schema versions may require separate profile filters due to the schema structural differences.
> In case a group has no filter assigned for a specific profile schema version, the group does not apply to the endpoints that use the profile of this schema version.

Every endpoint group within the application has a unique weight value assigned to it. The weight value is used to resolve conflicts among endpoint groups: in general, the larger the weight, the higher the group priority.

Every Kaa application comes with the default, non-user-editable group "all", with the weight value 0. The profile filter of this group is automatically set to "true" for every profile schema version in the system. As a result, the "all" group contains every endpoint registered in the application. The "all" group is used to define the default configuration, default notification topics access list, and for some other special functions.

Profile filters

Profile filters in Kaa are based on the Spring Expression Language (SpEL). All filters must be specified as predicates (statements which may be either true or false).

For further reference on the filters syntax, please refer to the Spring documentation.

The following example illustrates the general idea of profile filters.

1. First, let's assume the following profile schema.

```
[
    {
        "name": "ExtendedEndpointProfileChild",
        "namespace": "org.kaaproject.kaa.common.endpoint.gen",
        "type": "record",
        "fields": [
            {
                "name": "otherSimpleField",
                "type": "int"
            },
            {
                "name": "stringField",
                "type": "string"
            },
            {
                "name": "otherMapSimpleField",
                "type": {
                    "type": "map",
                    "values": "long"
                }
            }
        ]
    },
    {
        "namespace": "org.kaaproject.kaa.common.endpoint.gen",
        "type": "record",
        "name": "ExtendedEndpointProfile",
        "fields": [
            {
                "name": "simpleField",
                "type": "string"
            },
            {
                "name": "recordField",
                "type": "org.kaaproject.kaa.common.endpoint.gen.ExtendedEndpointProfileChild"
            },
            {
                "name": "arraySimpleField",
```

```
                "type": {
                    "type": "array",
                    "items": "string"
                }
            },
            {
                "name": "arrayRecordField",
                "type": {
                    "type": "array",
                    "items": "org.kaaproject.kaa.common.endpoint.gen.ExtendedEndpointProfileChild"
                }
            },
            {
                "name": "mapSimpleField",
                "type": {
                    "type": "map",
                    "values": "long"
                }
            },
            {
                "name": "mapRecordField",
                "type": {
                    "type": "map",
                    "values": "org.kaaproject.kaa.common.endpoint.gen.ExtendedEndpointProfileChild"
                }
            },
            {
                "name": "nullableRecordField",
                "type": ["org.kaaproject.kaa.common.endpoint.gen.ExtendedEndpointProfileChild",
"null"]
            }
        ]
    }
]
```

2. Second, let's assume the following endpoint profile, which corresponds to the schema.

```
{
    "simpleField": "SIMPLE_FIELD",
    "recordField": {
        "otherSimpleField": 123,
        "stringField": "STRING_VALUE1",
        "otherMapSimpleField": {
            "KEY1": 1,
            "KEY2": 2
        }
    },
    "arraySimpleField": ["VALUE1", "VALUE2"],
    "arrayRecordField": [
        {
            "otherSimpleField": 456,
            "stringField": "STRING_VALUE2",
            "otherMapSimpleField": {
                "KEY3": 3,
                "KEY4": 4
            }
        },
        {
            "otherSimpleField": 789,
            "stringField": "STRING_VALUE3",
            "otherMapSimpleField": {
                "KEY5": 5,
                "KEY6": 6
            }
        }
    ],
    "mapSimpleField": {
        "KEY7": 7,
        "KEY8": 8,
        "KEY9": 9
    },
    "mapRecordField": {
        "SOME_KEY1": {
            "otherSimpleField": 987,
            "stringField": "STRING_VALUE4",
            "otherMapSimpleField": {
                "KEY10": 10,
                "KEY11": 11
            }
        },
        "SOME_KEY2": {
            "otherSimpleField": 654,
            "stringField": "STRING_VALUE5",
            "otherMapSimpleField": {
                "KEY12": 12,
                "KEY13": 13
            }
        }
    },
    "nullableRecordField": null
}
```

3. At last, the following filters will yield true when applied to the given endpoint.

Filter	Explanation
`simpleField=='SIMPLE_FIELD'`	The profile contains simpleField with the value 'SIMPLE_FIELD'.
`arraySimpleField[1] =='VALUE2'`	The profile contains arraySimpleField, which is an array containing the element 'VALUE2' in the position 1.
`arraySimpleField.size()==2`	The profile contains arraySimpleField, which is a collection containing two elements.

`recordField.` `otherSimpleField==123`	The profile contains recordField, which is a record containing otherSimpleField set to '123'.

`recordField.`
`otherMapSimpleField.size()`
`==2`
	The profile contains recordField, which is a record containing otherMapSimpleField, which is a collection containing two entries.

`arrayRecordField[1].` `otherSimpleField==789`	The profile contains arrayRecordField, which is an array. This array contains an element in the position 1, which is a record containing otherSimpleField set to '789'.

`arrayRecordField[1].`
`otherMapSimpleField[KEY5]`
`==5`
	The profile contains arrayRecordField, which is an array. This array contains an element in the position 1, which is a record containing otherMapSimpleField, which is a map. This map contains an entry with the key KEY5 and the value 5.

`mapSimpleField[KEY8]==8`	The profile contains mapSimpleField, which is a map containing an entry with the key KEY8 and the value 8.
`mapRecordField[SOME_KEY2].` `otherSimpleField==654`	The profile contains mapRecordField, which is a map containing an entry with the key SOME_KEY2 and the value that is a record. This record contains otherSimpleField with the value 654.

`mapRecordField[SOME_KEY2].`
`otherMapSimpleField[KEY12]`
`==12`
	The profile contains mapRecordField, which is a map containing an entry with the key SOME_KEY2 and the value that is a record. The record contains otherMapSimpleField, which is a map containing an entry with the key KEY12 and the value 12.

`mapRecordField[SOME_KEY2].`
`otherMapSimpleField[`
`new java.lang.String`
`('KEY13').toString()]==13`
	An example of how to use a new object instance in a query.

`nullableRecordField==null`	An example of how to check a field for the null value.

`arraySimpleField[1]`
`=='VALUE2' and`
`((arraySimpleField.size()`
`==2)`
`or (arrayRecordField[1].`
`otherMapSimpleField[KEY5]`
`==5))`
	An example of how to combine several conditions in a query.

`!arrayRecordField.?`
`[otherSimpleField==456].`
`isEmpty()`
	The arrayRecordField field is an array of records. It contains at least one element that contains otherSimpleField with the value 456.