

stats.js - WebRTC statistics parameters leap detection

- [Analyzing the source code](#)
- [Parameter leaps detection](#)
 - [1. Parameter initialization](#)
 - [2. Receiving WebRTC statistics data array and checking the parameter value](#)
- [Parameters list handling](#)
 - [1. Adding parameter to list](#)
 - [2. Removing parameter from list](#)
 - [3. Receiving WebRTC statistics data array and checking the values](#)

`stats.js` module contains the functions to check WebRTC statistics parameters and to detect their leaps above a thresholds defined

Analyzing the source code

To analyze the source code take the `stats.js` module version available [here](#)

Parameter leaps detection

1. Parameter initialization

`Threshold()` [code](#)

Where:

- parameter name and a maximum leap threshold value are stored
- Kalman filter is created for the parameter

```
const threshold = {
  parameter: parameter,
  maxLeap: maxLeap,
  filter: SFU.createFilter(),
  previousValue: -1,
  ...
}
```

2. Receiving WebRTC statistics data array and checking the parameter value

`Threshold.isReached()` [code](#)

Where:

- current parameter value is passed through the Kalman filter
- if the value exceeds the threshold, parameter leap is detected

```
isReached: function(stats) {
  let hasLeap = false;
  if (stats && parameter in stats) {
    let value = threshold.filter.filter(stats[parameter]);
    if (threshold.previousValue > -1) {
      if (Math.round(Math.abs(value - threshold.previousValue)) > maxLeap) {
        hasLeap = true;
      }
    }
    threshold.previousValue = value;
  }
  return hasLeap;
}
```

Parameters list handling

1. Adding parameter to list

Thresholds.add() [code](#)

```
add: function(parameter, maxLeap) {
  if (!thresholds.thresholds[parameter]) {
    thresholds.thresholds[parameter] = new Threshold(parameter, maxLeap);
  }
},
```

2. Removing parameter from list

Thresholds.remove() [code](#)

```
remove: function(parameter) {
  if (thresholds.thresholds[parameter]) {
    delete thresholds.thresholds[parameter];
  }
},
```

3. Receiving WebRTC statistics data array and checking the values

Thresholds.isReached() [code](#)

```
isReached: function(stats) {
  let result = false;
  Object.keys(thresholds.thresholds).forEach((key) => {
    result = result || thresholds.thresholds[key].isReached(stats);
  });
  return result;
}
```