**AlarmFilter Spec**

**3rd Feb 2020** / 2:00PM / Pete Reeves

# **Contents**

1. Foreword
2. Component Breakdown

2.1 Alarm Filter Module

2.2 Alarm Filter

2.3 Regex

2.4 DataList

2.5 And/Or/Not

# **Foreword**

This document describes the functionality and design of the alarm filter module developed by Pete Reeves, the purposes and advantages of using the software.

The implementation of alarm management in IQVision is largely dependent on the health of the site. Directing alarms from a properly managed site can be a breeze, and allows alarms to reach the relevant handlers. Unfortunately, not all sites have been correctly managed and this leaves a mess of alarms. With no common naming structure or setup, alarm filtering using the base kit of IQVision is near enough impossible without going through every controller and redesigning every alarmed point.

# **Component Breakdown**

This is where the alarm filter module comes in. The alarm filter allows you to build up a series of checks to filter alarms into their relevant categories.

**2.1 Alarm Module**

To be dropped into the alarm service and have any relevant alarm classes that you want to check wired into the routeAlarm slot. You can then drop any number of AlarmFilters onto it. This will pass the alarm to each filter so that they can attempt to match it. Note: this means the alarm may end up in multiple classes as it passes multiple checks. This is intentional but is something to be aware of when designing the checks.

**2.2 Alarm Filter**

This component combines the checks dragged onto it, and if the alarm that has been sent to it matches all of the checks directly underneath the filter, then it will direct the alarm into the correct alarm slot. This can then be wired up to a recipient (alarm console, email recipient, SMS etc) for output and processing.

**2.3** **RegexCheck**

This has two important fields; the Regex field and the properties field. The properties field is what property of the alarm you want to check against. A list of common fields and their relevant codes can be found in Appendix A

The other property is the Regex field. This is the regex function you want to compare the property to. Note this compares against the toString() method of the property you have selected. The majority of regex functions work here, and it is based upon the Match and Pattern methods built into Java.

**2.4 DataListCheck**

Similar to the Regex field, this component has a property field but instead of a regex function it has a data array. To edit the data array; invoke the update Array action on the dataList and enter the new array (items separated by commas).

This method then checks if the property selected contains any value in the array. This can be useful for selecting groups of alarms, such as those from particular controllers (as their source property will contain the controller name) or those with a set of different priorities.

**2.5 Check[And/Or/Not]**

These components have been grouped together as they all perform a similar function. They allow you to combine multiple RegexChecks and DataList checks together with basic logic.

**2.5.1 CheckAnd**

This checks that ALL checks underneath it are true.

**2.5.2 CheckOr**

This checks that ANY checks underneath it are true.

**2.5.3 CheckNot**

This returns the opposite of the first check underneath it. Unlike the others this one should only contain one item (though that item can contain more underneath it.

**Appendix A**

|  |  |  |
| --- | --- | --- |
| **Desired Property** | **Description** | **Code** |
| Timestamp | Time the alarm occurred | timestamp |
| Uuid | Unique alarm code | uuid |
| SourceState | Normal/Offnormal. What state the module is currently in. | sourceState |
| Ack State | Whether the alarm has been acked | ackState |
| Ack Required | Whether the alarm requires and ack or not to clear | ackRequired |
| Source | Where the alarm came from | source |
| Alarm Class | Which class the alarm was sent to | alarmClass |
| Priority | What priority the alarm was given | priority |
| Normal Time | What time the alarm returned to normal | normalTime |
| Ack Time | What time the alarm was acked | ackTime |
| User | Who acked the alarm | user |
| Alarm Value | What value the alarmed property had | alarmData.alarmValue |
| Device Name | What device generated the alarm | alarmData.deviceName |
| Escalated | If the alarm has been escalated | alarmData.escalated |
| Lan Name | What Lan the alarm was generated on | alarmData.lanName |
| Module Name | What Module the alarm was generated by | alarmData.moduleName |
| Notes | If any notes have been left on the alarm they will appear here | alarmData.notes |
| Source Name | The name of the source the generated the alarm. Similar to “Source” but more formatted | alarmData.sourceName |
| System Name | What System the alarm was generated on | alarmData.systemName |
| Time Zone | What time zone the device that generated the alarm is set to | alarmData.timeZone |
| Trend Encoding Format | What format the alarm has been sent in | alarmData.trendEncodingFormat |
| Units | Any units attached to the module (C°, lumens, volts etc) | alarmData.units |
| Alarm Transition | What state triggered the alarm | alarmTransition |
| Last Update | Time of the last update to the alarm | lastUpdate |