

# Help-Center Introduction

Welcome to the Providence Service Help page. This page explains all the important functions and features of the Providence Service.

This page includes the following topics:

- Monitoring Service About
- Monitoring Service Tree
- Administrative Operations
- Monitoring Internal Jobs
- API Documentation
- Postman Links
- Swagger Links

# About

## Providence Service Summary

The providence service is a custom monitoring solution built specifically to monitor multiple services.

The main purpose of the monitoring service is to provide a single source of information of the platforms current health state. The health state of the platform is calculated bottom-up from the state of the services (B2B-Services and Basic-Services) and their respective components.

The monitoring service contains a structure of the platform: Services are made of actions, which on the other hand are built out of components. Components can be Azure Resources or external services.

Services are green/OK unless an alert on one of their dependencies is triggered, which will then turn their state to WARNING or ERROR.

## Alerts

Alerts are defined in the platform and when triggered, they change the state of a service's health. The following table shows each state and its implication.

Component's state	Implication	Example
OK	No issues reported/ alerts triggered	-
WARNING	Operations team action required soon to prevent outage of the service	CPU load running continuously at > 80%
ERROR	Action required because a component is failing and auto-recovery was unsuccessful	Watchdog check for microservice failing, Data Factory Pipeline run failing

## No Current Data available message

To ensure that the monitoring service is receiving the newest updates of the platform, heartbeat signals are sent periodically from each environment to the service. If no new heartbeats arrive, the connection with the backend might be lost. In this case, data shown in the dashboard might not reflect the current health state of the environments. These are marked with "No current data available".

On the landing page, for each environment the last received heartbeat is shown in the environments tile.

# TestB2B

## Services

Remote Control

 OK

Eco Monitor

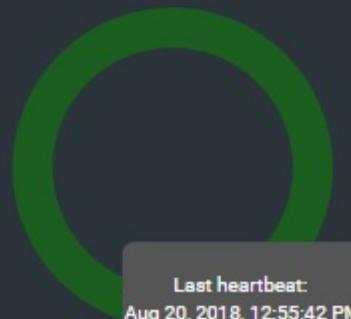
 OK

Fleet Communication

 OK

...

 OK 17  
 WARNING 0  
 ERROR 0



Last heartbeat:  
Aug 20, 2018, 12:55:42 PM

Heartbeat   


# Providence Service Tree

The Providence service tree is the basis for monitoring the environments.

The service tree is composed of elements which build up a tree with the environment as its root node.

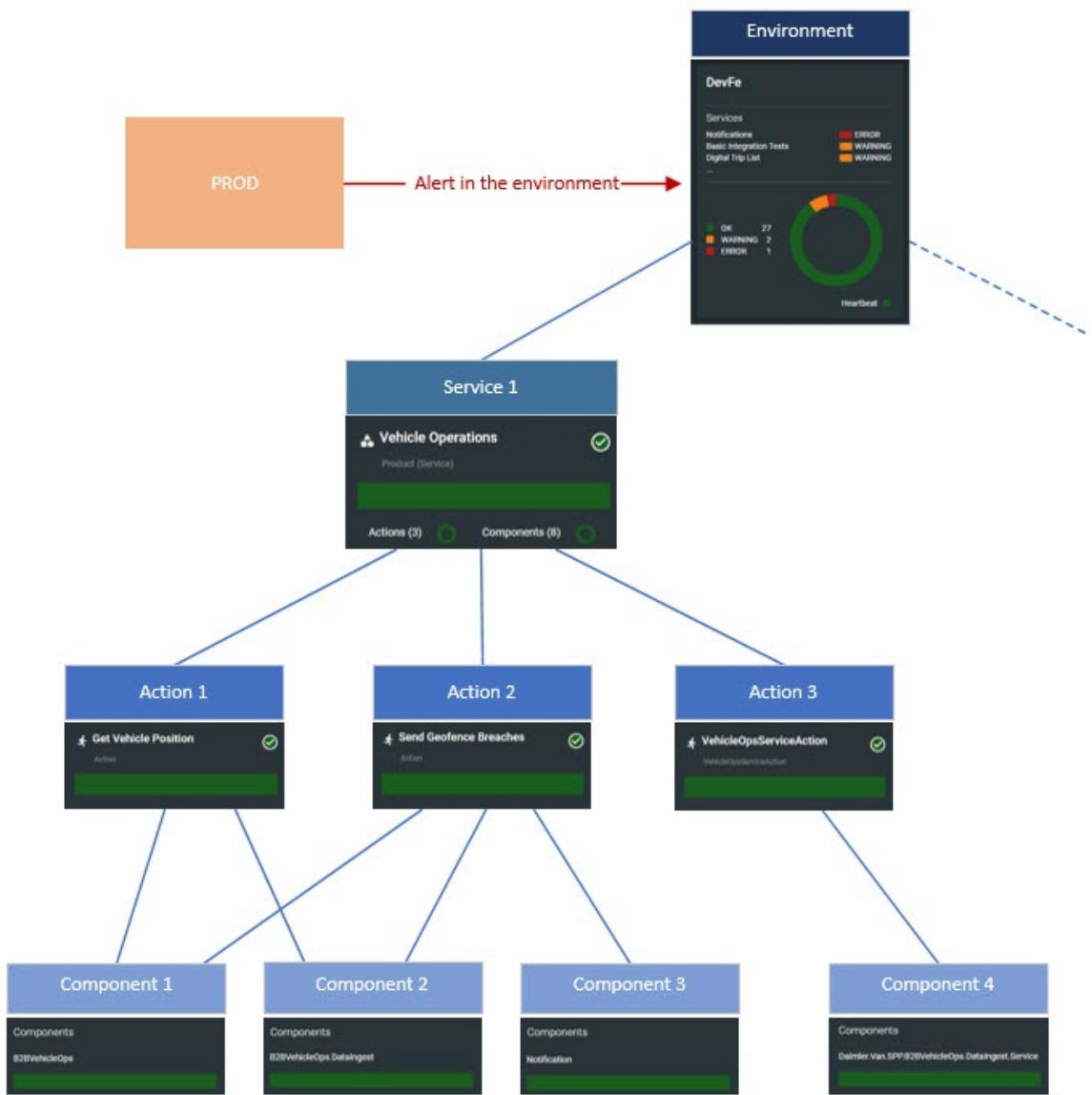
The following diagram shows an example of a service tree:



The root node is always the environment. An environment contains services, which are composed of actions. Each action has components, which are the leaf nodes of the service tree.

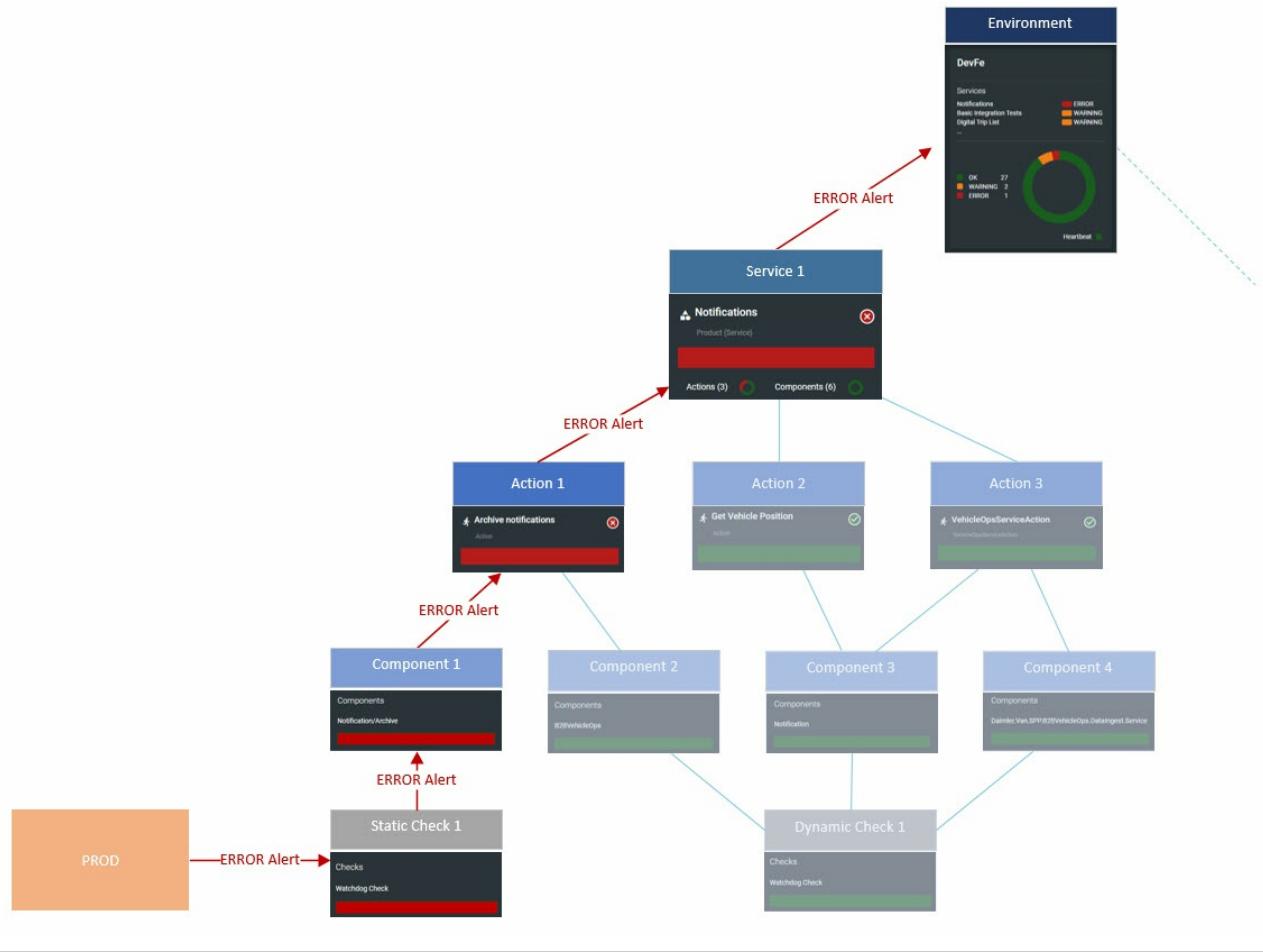
## Alerts

Alerts are triggered in each environment. For events to be visible in the Providence dashboard, a check must be created in the Providence service.



## State

State in the service tree is propagated bottom-up from the child nodes to their parent node.



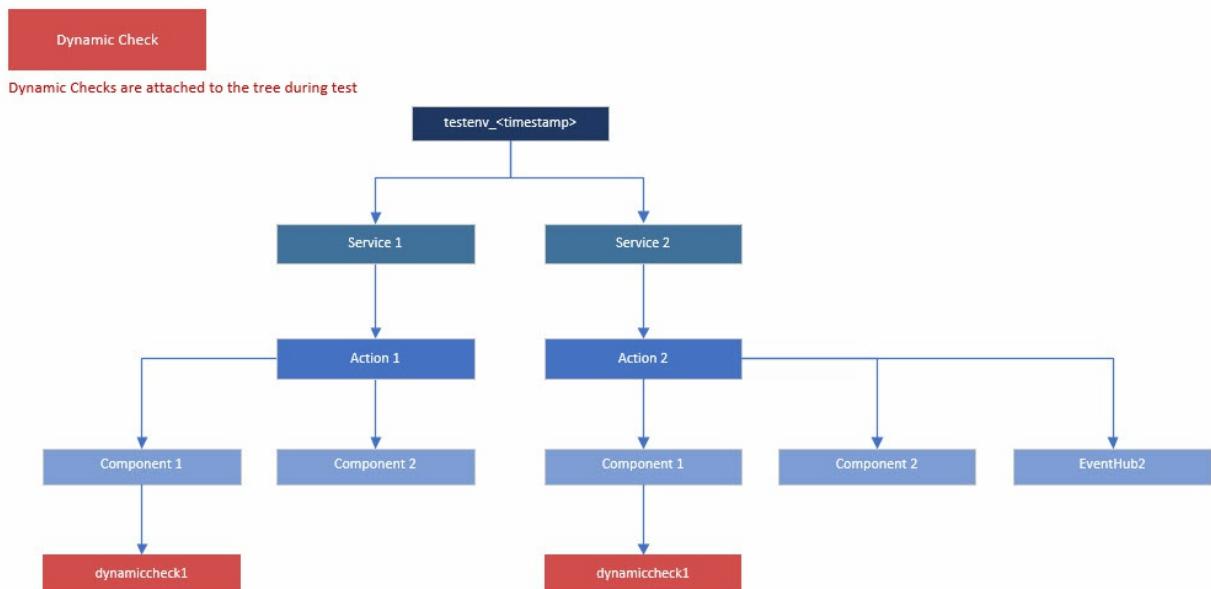
# Tree and use cases

## Alerting strategy

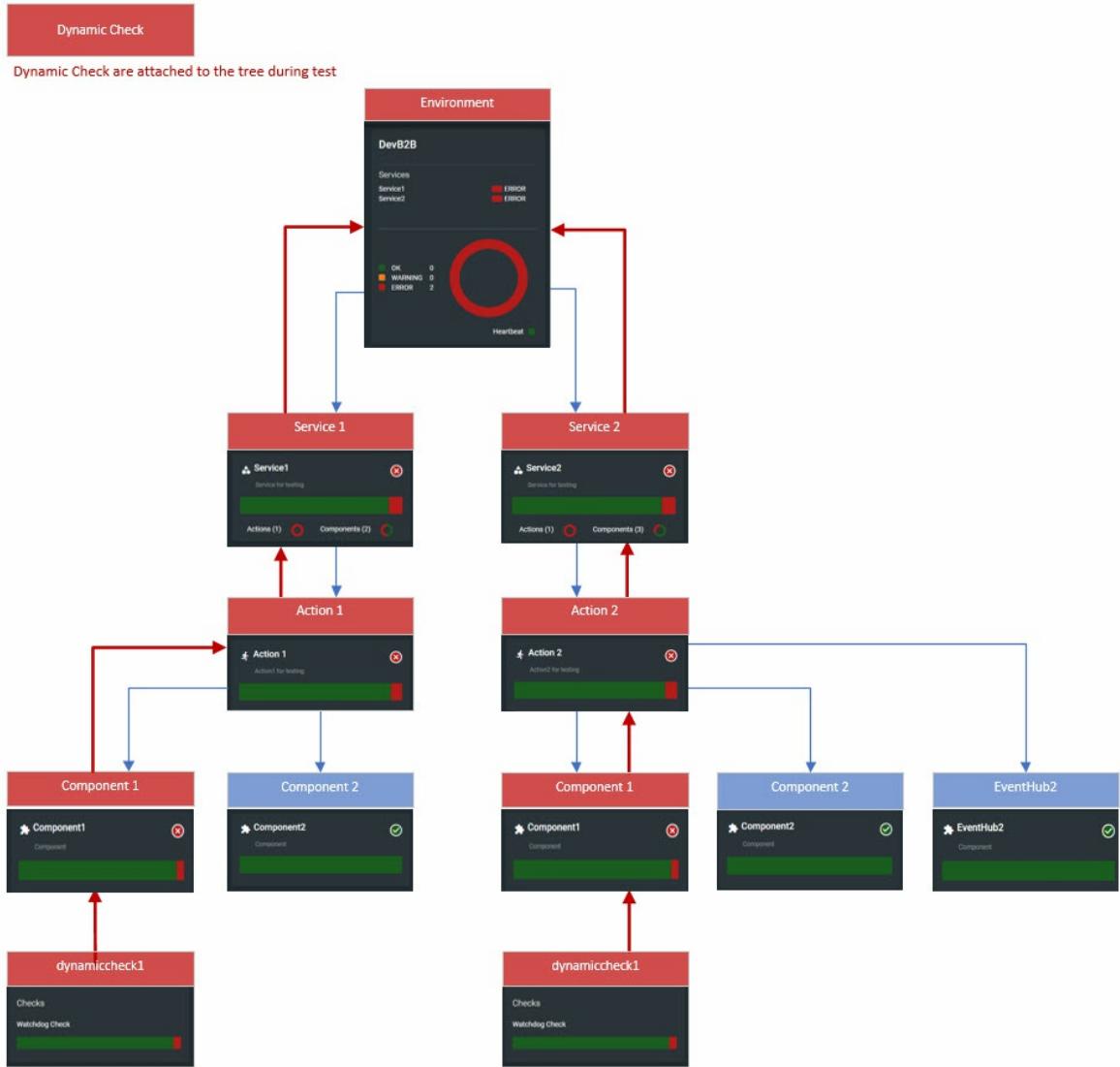
**You can't fix what you don't know is broken.**

Alerting on what matters is critical, it is underpinned by collecting and measuring the right metrics and logs, as well a monitoring tool that is capable of storing, aggregating, visualizing, analyzing, and initiating an automated response when conditions are met. Improving observability of your services and applications can only be accomplished if you fully understand its composition in order to map/translate that into a detailed monitoring configuration to be applied by the monitoring platform, including the predictable failure states (the symptoms not the cause of the failure) that make sense to alert on.

### 1. Alert for Checks (without AlertName)



For Example Dynamic Check:



## Alert

Attribute	Value
CheckId	Watchdog
ElementId	Component1
State	Error

## Procedure in background

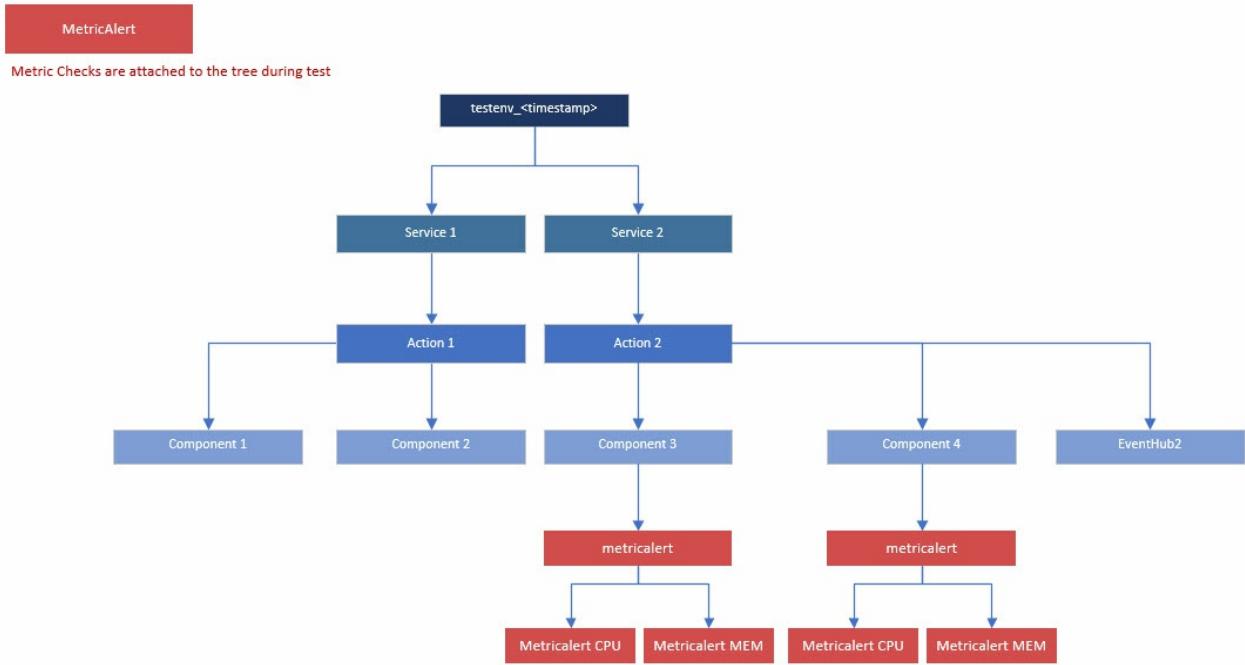
- search for node with ElementId
- attach Check to the node if not existant, else update
- set the state of the Check
- propagate the state transition towards the root of the tree

## Statetransitions

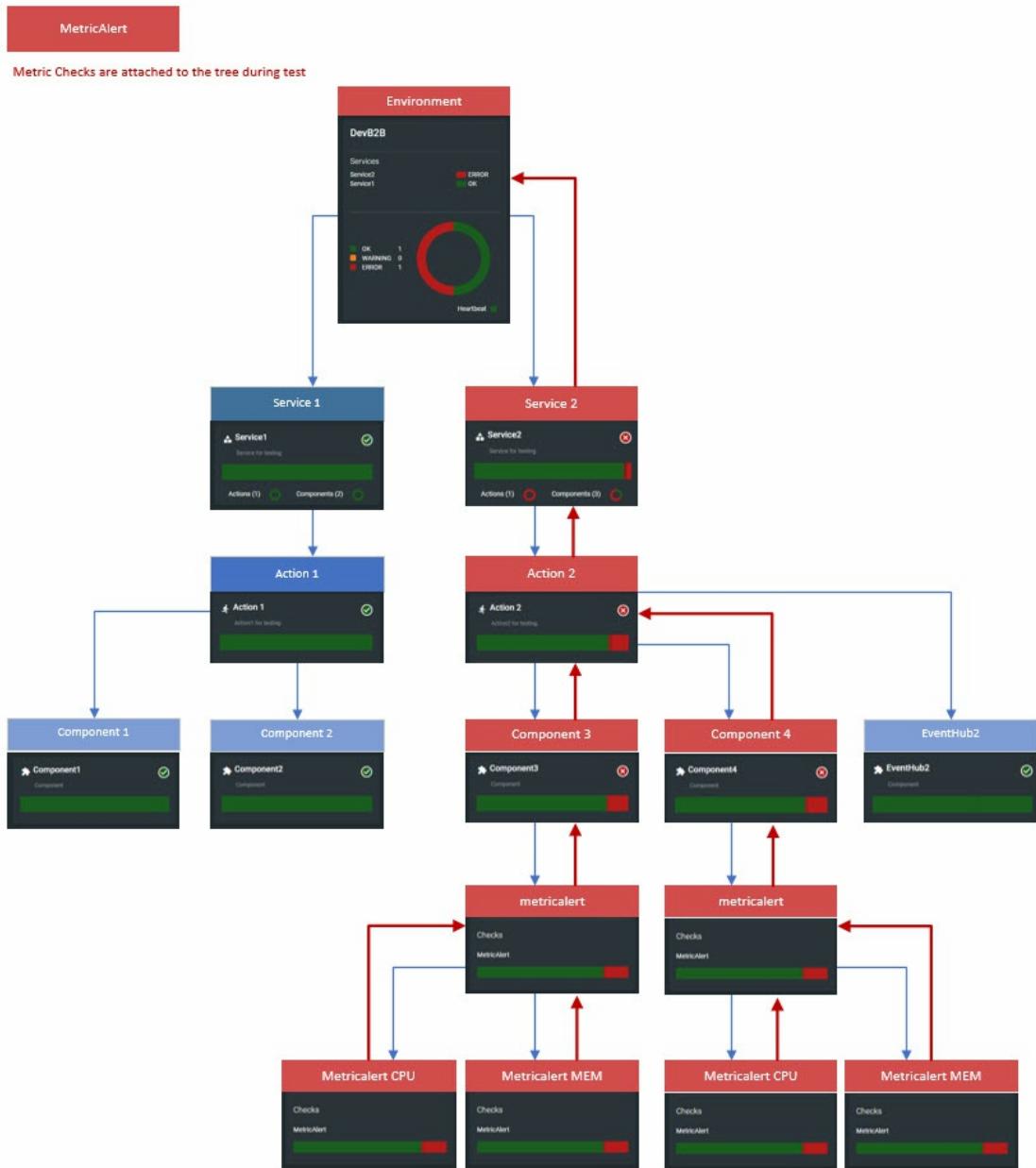
CheckId	ElementId	AlertName	State	Type
Watchdog	Component1	-	Error	Component
-	Action1	-	Error	Action
-	Service1	-	Error	Service
Watchdog	Component1	-	Error	Component
-	Action2	-	Error	Action

-	Service2	-	Error	Service
-	DevB2B	-	Error	Environment

## 2. Alert for Checks (with AlertName)



For Example Metric Check:



## Alert

Attribute	Value 1	Value 2
CheckId	MetricAlert	MetricAlert
ElementId	Component3	Component4
State	Error	Error
AlertName	CPU	MEM

## Procedure

- search for node with ElementId
- attach Check to the node if not existant, else update
- set the state of the Check
- propagate the state transition towards the root of the tree
- Attach Check for AlertName

## Statetransitions

CheckId	ElementId	AlertName	State	Type
metricalert	Component3	CPU	Error	Component

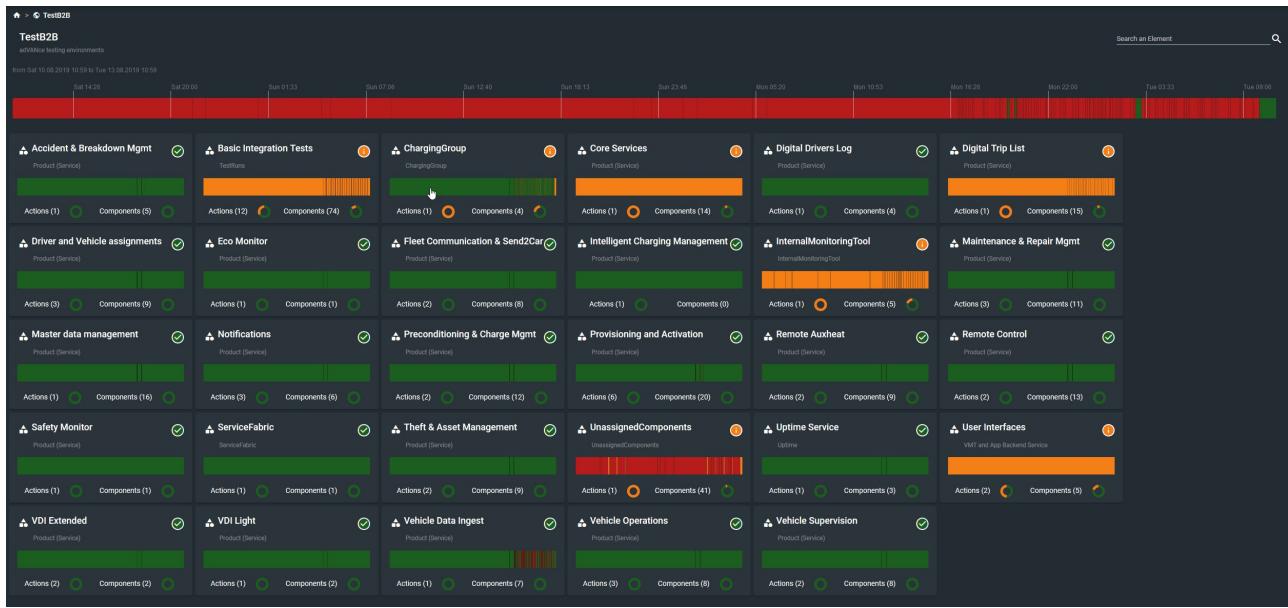
metricalert	Component3	MEM	Error	Component
-	Component3	-	Error	Component
metricalert	Component4	CPU	Error	Component
metricalert	Component4	MEM	Error	Component
-	Component4	-	Error	Component
-	Action2	-	Error	Action
-	Service2	-	Error	Service
-	DevB2B	-	Error	Environment

## # Graphical User Interface

### Dashboard

The dashboard includes information about the current status of the environment.

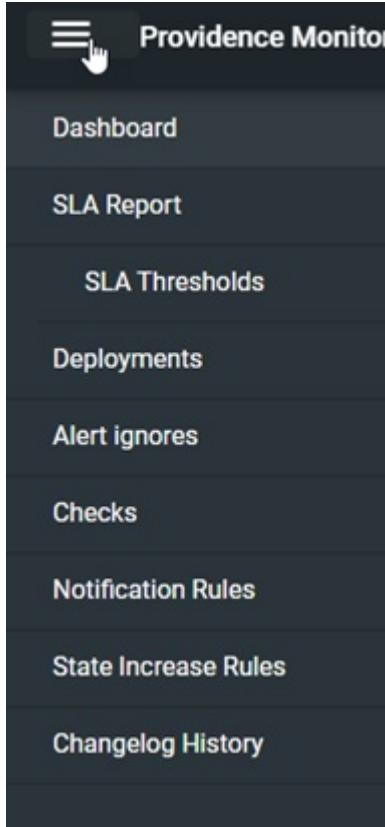
- Timeline of the environment
- Overview of all services monitored:
  - Service name
  - Status icon (OK, Warning, Error)
  - Service timeline shows the status in color
  - Quantity of actions
  - Quantity of components



### Operational Menu

The Operational menu includes the following areas:

- Dashboard
- SLA Report
  - SLA Thresholds
- Deployments
- Alert ignores
- Checks
- Notification Rules
- State Increase Rules
- Changelog History



## Timeline

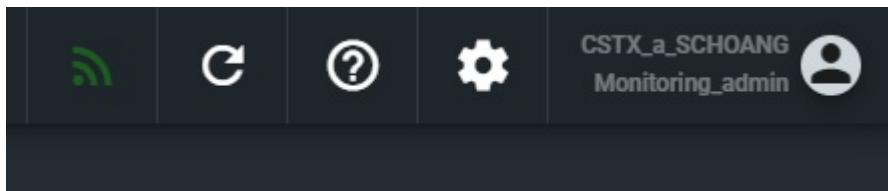
The timeline shows the status of the environment. The status is highlighted in color.

The timeline shows also the deployment slot. The deployment window is marked in blue color, means in that time a deployment of a service is running.



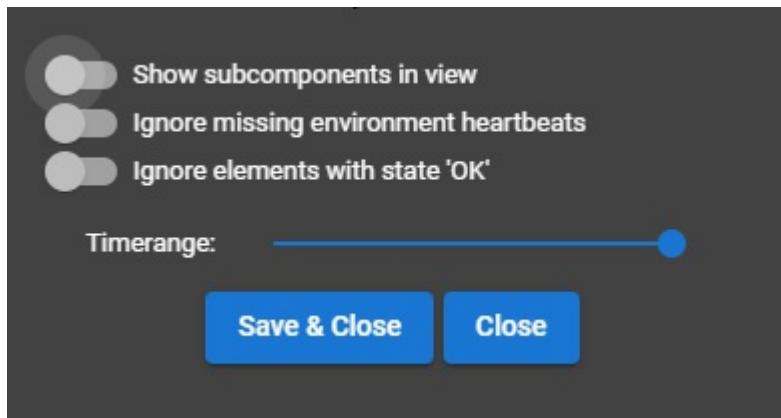
## Toolbar

In the upper right corner you can see the Toolbar



The following icons/buttons are displayed:

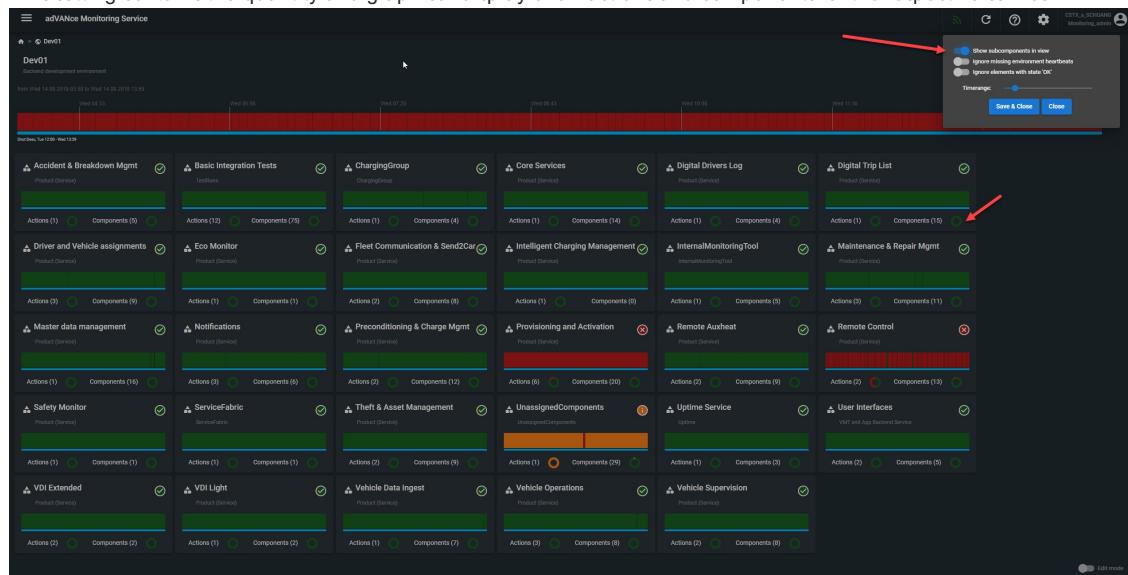
- Receiving (real time data from monitoring backend)
- Refresh (manual refresh)
- Help (Linkout to the Helpfunction)
- Setting (The settings reference to the dashboard and the timeline)



The following settings are available:

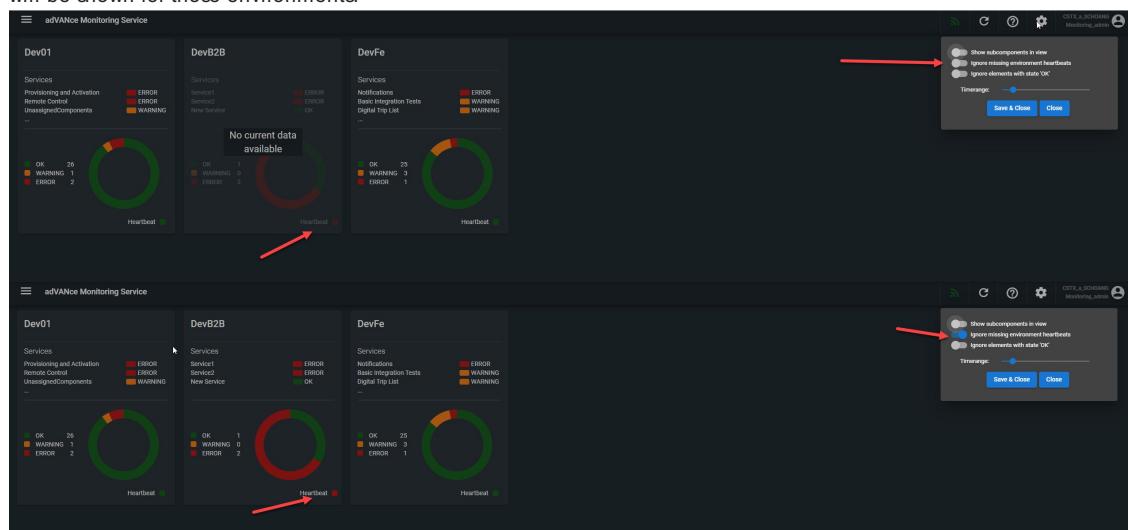
- Show subcomponents in view

This setting contains the quantity and graphical display of all actions and components for the respective service.



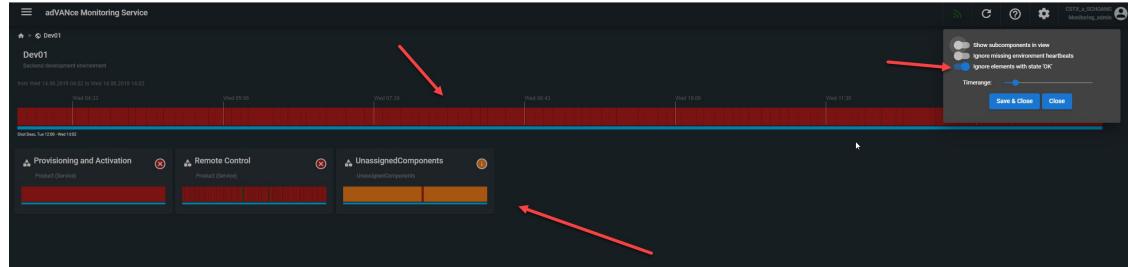
- Ignore missing environment heartbeats

This setting ignore missing heartbeats for all visible environments which means that no "No Data available" message will be shown for those environments.



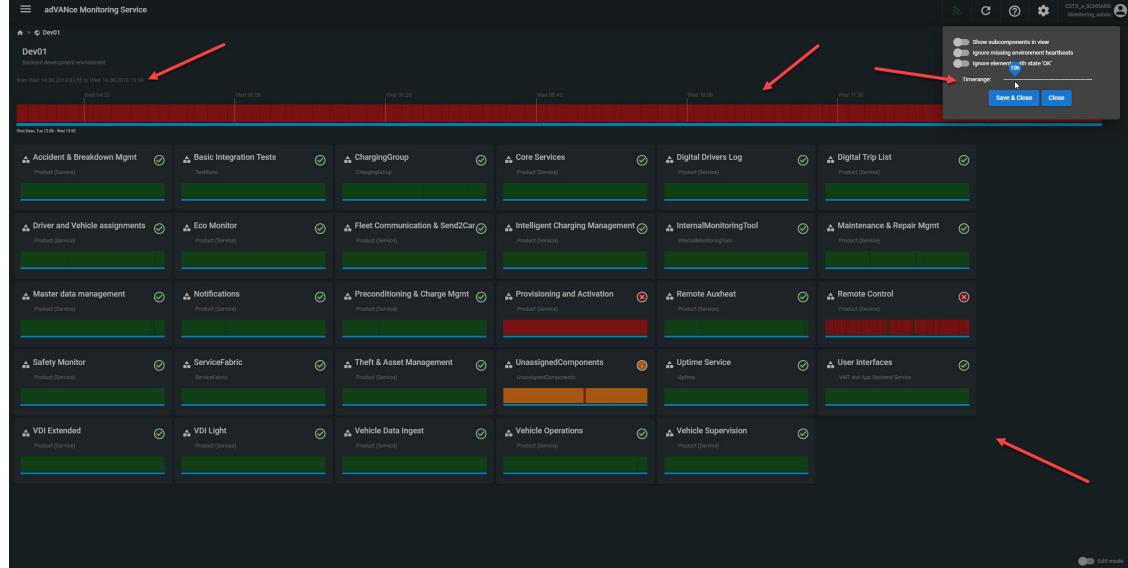
- Ignore elements with state "OK"

This setting shows only elements with warnings or error. All elements with the status "OK" are not listed in this overview.

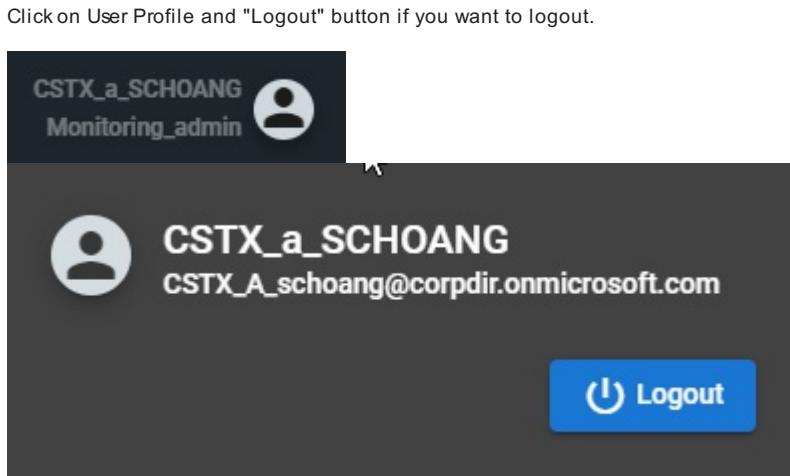


- Timerange setting

The timerange setting shows the timeline and all elements in the defined time period.



- Login / Logout User (User Profile)



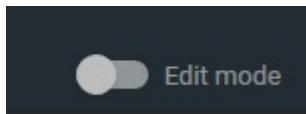
## Search an Element

The search function can be used to search for elements in the environment to step directly to this element.

🔍

## Administration

In the lower right corner you can see the toggle 'Edit mode', it is only visible for the administrator. He can create, modify and delete environments, services, actions and components.



\*\*\*

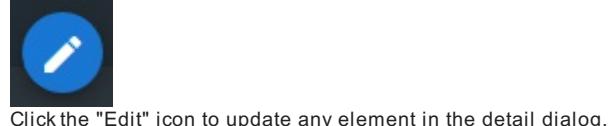
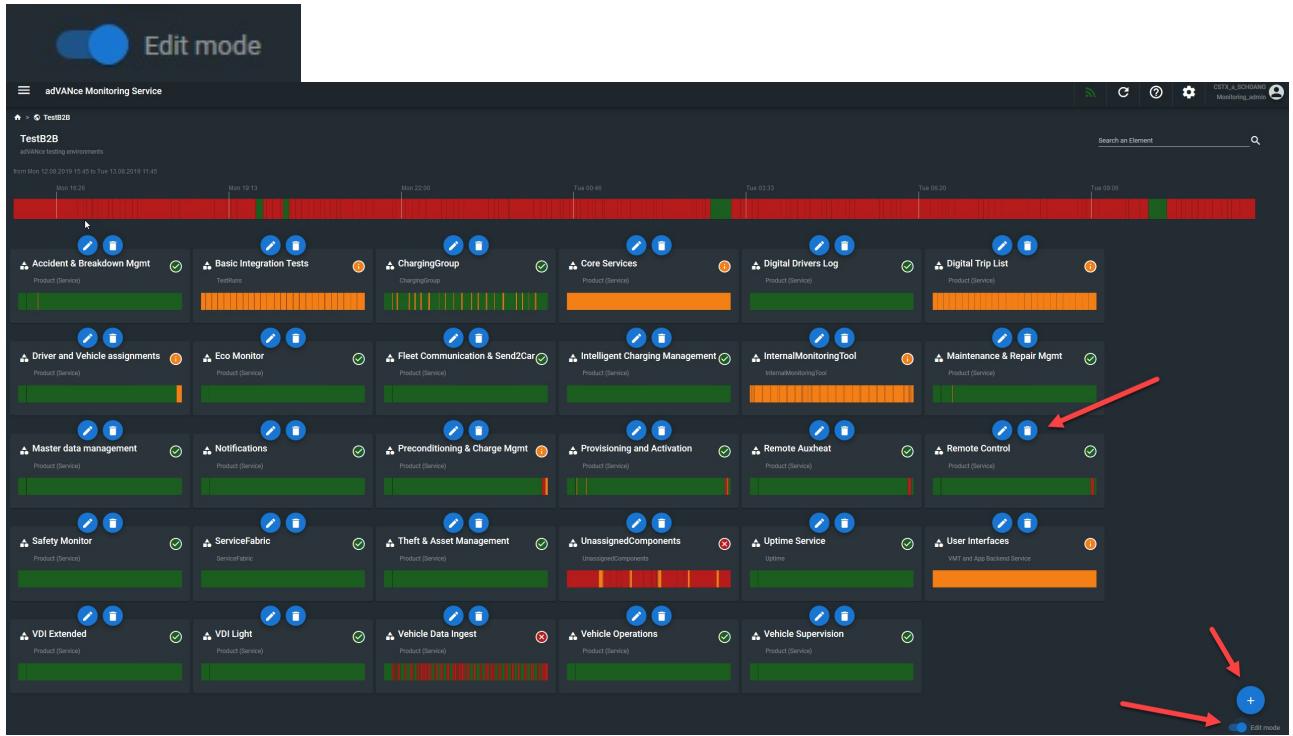
# Administrative Operations

## Administration

The Providence Service offers several ways to edit the dashboard. These are referred to as "administrative operations" and can be activated using the "Edit mode" switch. To see and activate the switch, a user must have the "Admin" role.

The admin can then create, modify or delete environments, services and components directly on the dashboard.

When the "Edit mode" is active, the following icons are visible:



Click the "Edit" icon to update any element in the detail dialog.

## Update Environment

Name \*

Dev01

Description

UFT\_Description1\_Modify

Element Id

Subscription Id

Demo environment?

**Save**      **Cancel**

## Update action

Name \*

UnassignedComponentsAction

Description

UnassignedComponentsAction

Element Id

UnassignedComponentsActiontestb2b

**Save**      **Cancel**

## Update component

Name \*

ChargingGroupMgmt/Service

Description

ChargingGroupMgmt/Service

Element Id

fabric:/Daimler.Van.SPP.ChargingGroupMgmt/Service

Component Type \*

Component

**Save**

**Cancel**



Click on the "Delete" icon to delete the selected element.

## Confirmation

**Do you really want to delete the element?**

**Delete**

**Cancel**



Click on the "Add" icon to add a new element.

## Create new environment

Name \*

|

Description

|

Element Id \*

Subscription Id \*

Demo environment?

**Save**      **Cancel**

## Create new service

Name \*

|

Description

Element Id \*

**Save**      **Cancel**

## Create new action

Name \*

|

Description

Element Id \*

**Save**      **Cancel**

## Create new component

Name \*

---

Description

---

Element Id \*

---

Component Type \*

Save

Cancel



Click on "Add existing component(s)" icon to add an existing component(s).  
This icon is only visible in the component level.

## Add existing component(s)

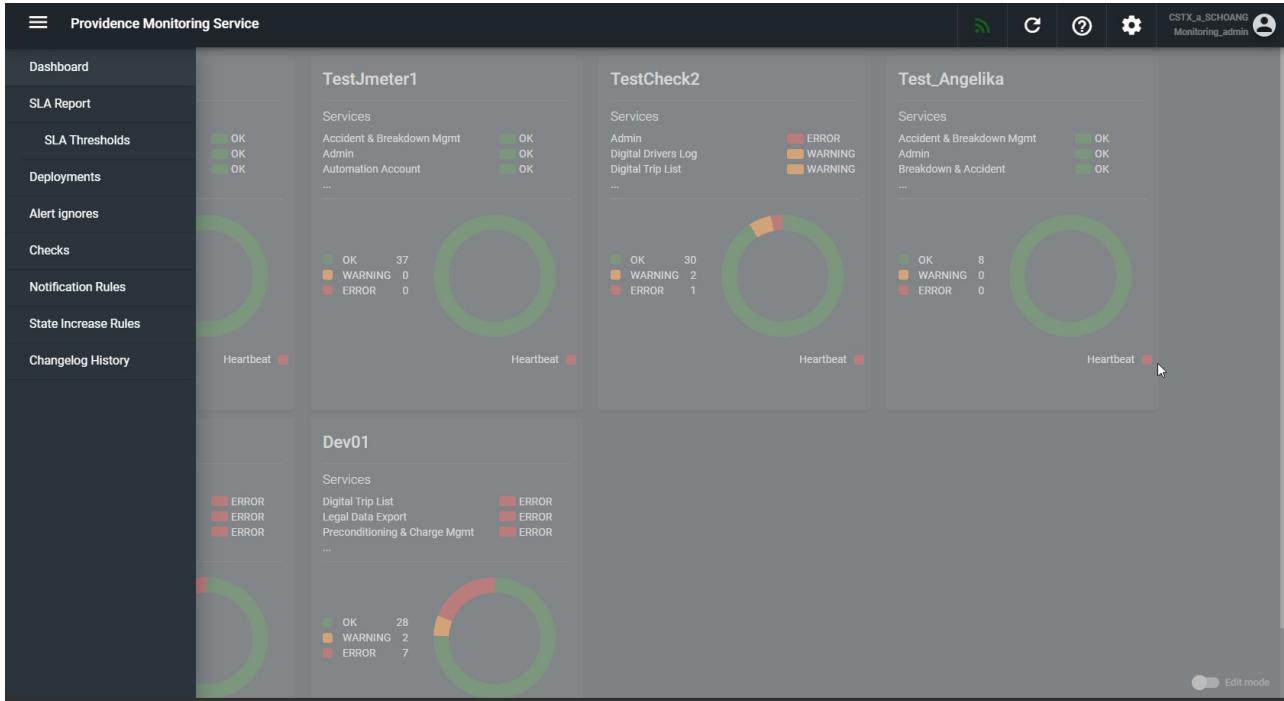
<input type="checkbox"/>	Name ↑	Element Id	Description	Orphaned
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No
<input type="checkbox"/>	/subscriptions/c30be90e...	/subscriptions/c30be90e-d5c2-4...	UnassignedComponents	No

Save

Cancel

# Operational Menu

Click on "menu" icon to make the column "operational menu" visible or to hide it again.



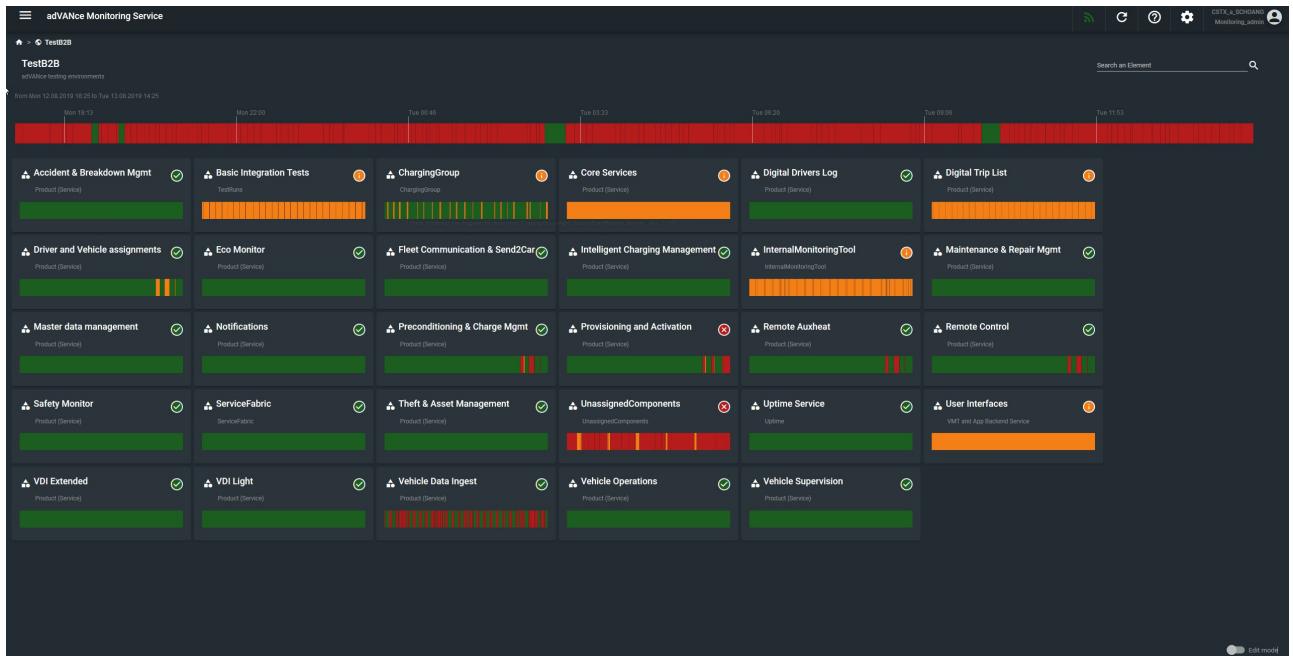
The operational menu includes the following pages:

- Dashboard Overview
- SLA Report
  - SLA Thresholds
- Deployment
- Alert ignores
- Checks
- Notification Rules
- State Increase Rules
- Changelog History

# Dashboard Overview The dashboard contains information about the current status of the environment.

- Timeline of the environment (The timerange is adjusted in the toolbar)
- List of all services to be monitored:

- Service name
- Status icon (OK, Warning, Error)   
- Service timeline in color representation of the status
- Quantity of actions with graphic display
- Quantity of components with graphic display



## # SLA Thresholds

The system monitoring works by "observing" the individual instances of an environment and "tracking" the various metrics within that instance. Each metric has two defined thresholds, one for minor (warnings) and one for major or critical problems (errors).

If a metric is above the error threshold it will be seen as error state.

If a metric is above the warning threshold but under the error threshold it will be seen as warning state

The thresholds are administered by the Administrator and can be visualized in the SLA report.

Hint: Define thresholds which reflect the expectations of your customers.

**Warning:** The experience of the enduser can be influenced badly without running in critical issues. With a warning you can indicate to fix things before your system is running in critical circumstances.

**Error:** If you are in the critical state then you need to fix issues immediately to come back to normal business again.

With administrating the thresholds you define following options:

- **Environment Name:**  
Choose the environment to administrate.
- **Source:**  
Choose between Option "Error" and "Error + Warning". This specifies whether alerts with the status "Warning" are to be taken into account when calculating the service downtimes.
- **Threshold Warning:**  
Define the threshold for the warning in %, this value needs to be higher than the error threshold.
- **Threshold Error:**  
Define the threshold for the state error in %

The screenshot shows the Providence Monitoring Service interface with the following details:

- Header:** Providence Monitoring Service with a navigation menu icon, RSS feed icon, refresh icon, help icon, settings icon, and user profile for CSTX\_a\_SCHOANG Monitoring\_admin.
- Section:** SLA Thresholds
- Sub-section:** Configure the params for the SLA Threshold Reports
- Environment Name:** TestCheck
- Source:** Error + Warning
- Threshold Warning:** 60%
- Threshold Error:** 50%
- Buttons:** Save SLA Threshold and Delete SLA Threshold

# SLA Report Using the SLA Reports you can monitor the compliance of the Service Level Agreement, and you can recognize incidents early.

#### Common example of availability and outages:

A good overview gives the following table showing the cumulative availability of a service per year in percent.

The calculation is made on following basis: 365 days \* 24 hours/day = 8.760 hours / year

Availability Percentage	Availability (Year)	Downtime (Year)
100 %	≈ 8.760 h	≈ 0 h
99 %	≈ 8.672 h	≈ 88 h
90 %	≈ 7.884 h	≈ 876 h
80 %	≈ 7.008 h	≈ 1.752 h
50 %	≈ 4.380 h	≈ 4.380 h
10 %	≈ 876 h	≈ 7.884 h

#### Calculation basis

Our calculations for the SLA-report are not calculated on the complete data, it will be calculated along a defined time range.

Also at first you need to define the time duration in which you want to have the SLA report. This time period is named "SLA-Period".

**Downtime** ≈ sum(Error-Duration) + sum(Warning-Duration) in SLA-period

**Uptime** ≈ 1 - (Downtime/SLA-period) [%]

**Sample:**

**SLA-period:** 01.01.2020 - 02.01.2020 -> SLA-period: 24 [h]

**Error:** 01.01.2020 05:00 - 01.01.2020 6:00 -> Error-Duration: 1 [h]

**Warning:** 01.01.2020 07:00 - 01.01.2020 8:00 -> Warning-Duration: 1 [h]

Downtime = (1 + 1) = 2 [h]

Uptime = 1 - (2 /24) = 0,91 [%]

## SLA Report Overview

In order to retrieve the SLA calculation for a specify environment you have to specify:

- The environment name of your environment
- The start and end date of the calculation

**Note:** If the start and end date of the SLA period is not given we take as default the last 3 days for the calculation.

The calculations are based on 24 hours / day.

The screenshot shows a dark-themed UI for creating an SLA report job. At the top, it says "SLA Reports" and "Create a new SLA Reports Job". Below that, there are three input fields: "Environment Name" set to "Dev01", "Start Date" set to "3/27/2021", and "End Date" set to "3/30/2021". At the bottom, there are two buttons: "Create Job" and "Cancel".

After starting a SLA calculation the queued job is shown in the SLA Jobs Overview

Environment	Type	Username	State	StateInformation	Startdate	Enddate	QueueDate	Filename	Action
4025db3d-bb6f-4de1-8495-36a843f29882	SLA	CSTX_a_RICKRIE@...	<input checked="" type="checkbox"/> Processed	Successfully calculated SLA data.	2021-03-27T00:00:00	2021-03-30T23:59:59	2021-03-30T11:41:18	CalculatedSla_18	<a href="#">Show data</a>
TestReview	SLA	CSTX_a_RICKRIE@...	<span>!</span> Error	Unexpected Error on SLA Calculat...	2021-03-27T00:00:00	2021-03-30T23:59:59	2021-03-30T11:41:18		<a href="#">Show data</a>

A queued job can have one of the following states:

- Queued
- Running
- Processed
- Error

## SLA Report Evaluation

After the SLA is finished and the state was updated to "Processed" you can click on the "Show data" button to open the SLA Overview.

Element Type	SLA State	SLA Value (uptime)

Here you can select the representation type and optionally a single element you want to retrieve the SLA calculation for.

**SLA Report**

Monitor and filter the SLA Report

Environment Name Dev01	Start Date 2021-03-27T00:00:00	Representation Value	Element Name
	End Date 2021-03-30T23:59:59	Pie Chart	
		Line Chart	

**Show Report** **Download CSV** **Download PDF**

Element Type	SLA State	SLA Value (uptime)
vehic Service	Ok	84.82%
vehic Action	Error	0%
vehic Action	Error	0%
vehic Service	Ok	84.63%
vehic Service	Ok	100%
user Action	Ok	99.9%
trigg Action	Warning	23.97%

After selecting those properties you can click on "Show Report" to show the calculated SLA values

Element Type	SLA State	SLA Value (uptime)
vehic Service	Ok	84.82%
vehic Action	Error	0%
vehic Action	Error	0%
vehic Service	Ok	84.63%
vehic Service	Ok	100%
user Action	Ok	99.9%
trigg Action	Warning	23.97%

The "SLA State" delivers the state of the corresponding element and can contain "Error", "Warning" and "OK".  
The corresponding value is shown as percent or as a chart (depends on representation format).  
If no SLA data is available this is shown in the field "ComponentType" as "No SLA Data found".

## SLA Representation formats

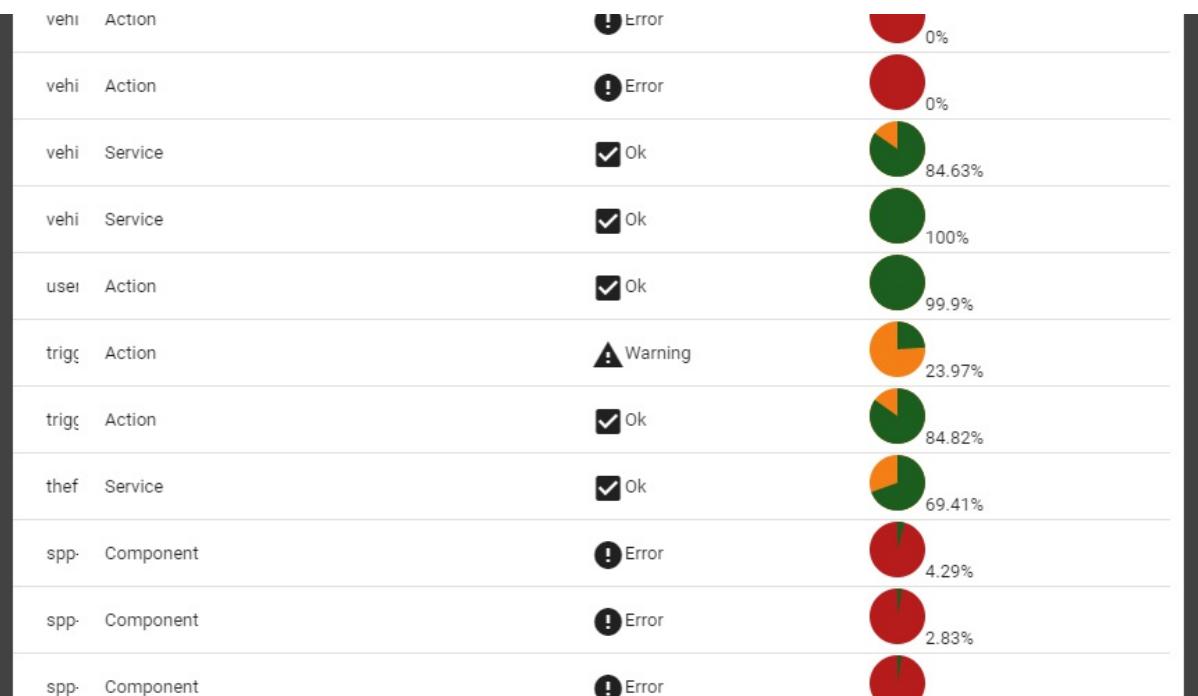
- **"Value" representation format**

The SLA value is always given in percent and correlates to the SLA uptime.

Element Type	SLA State	SLA Value (uptime)
vehi Service	<input checked="" type="checkbox"/> Ok	84.82%
vehi Action	<span style="color: red;">!</span> Error	0%
vehi Action	<span style="color: red;">!</span> Error	0%
vehi Service	<input checked="" type="checkbox"/> Ok	84.63%
vehi Service	<input checked="" type="checkbox"/> Ok	100%
user Action	<input checked="" type="checkbox"/> Ok	99.9%
trigc Action	<span style="color: orange;">!</span> Warning	23.97%

- "Pie Chart" representation format

The SLA Pie correlates also always to the SLA Uptime.

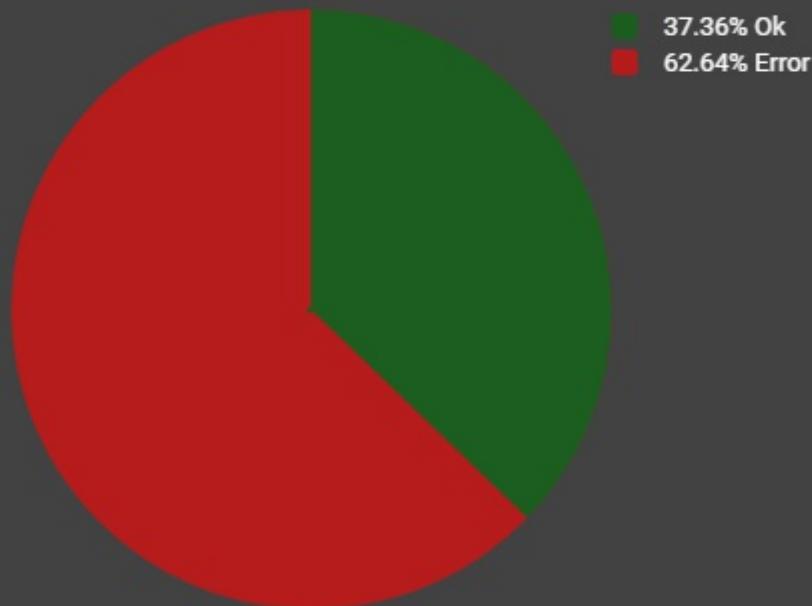


- Green part = Element was ..% available
- Yellow part = Element was with a ..% Warning
- Red part = Element was with a ..% Error

Mouse double click - on a corresponding pie shows a detail information of the data.

## SLA report

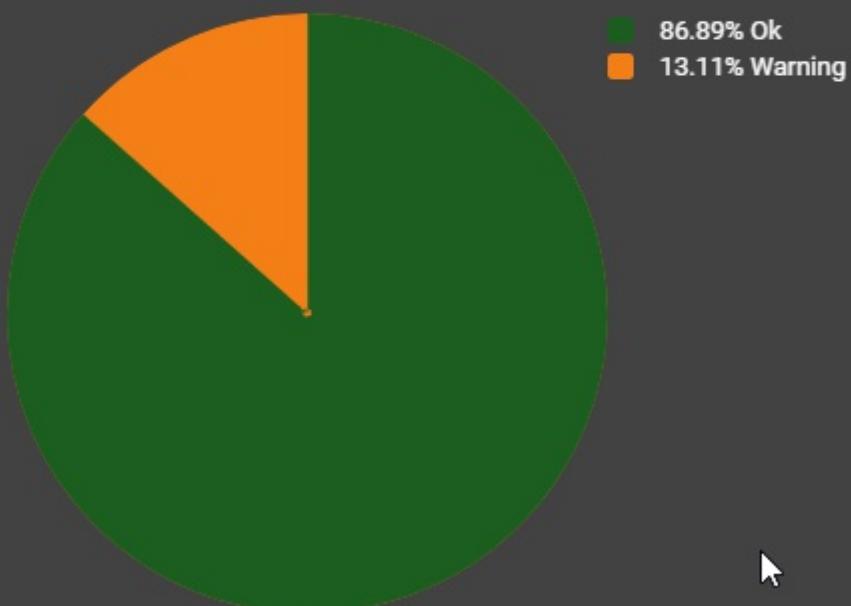
notificationsServiceDev01



[Close](#)

## SLA report

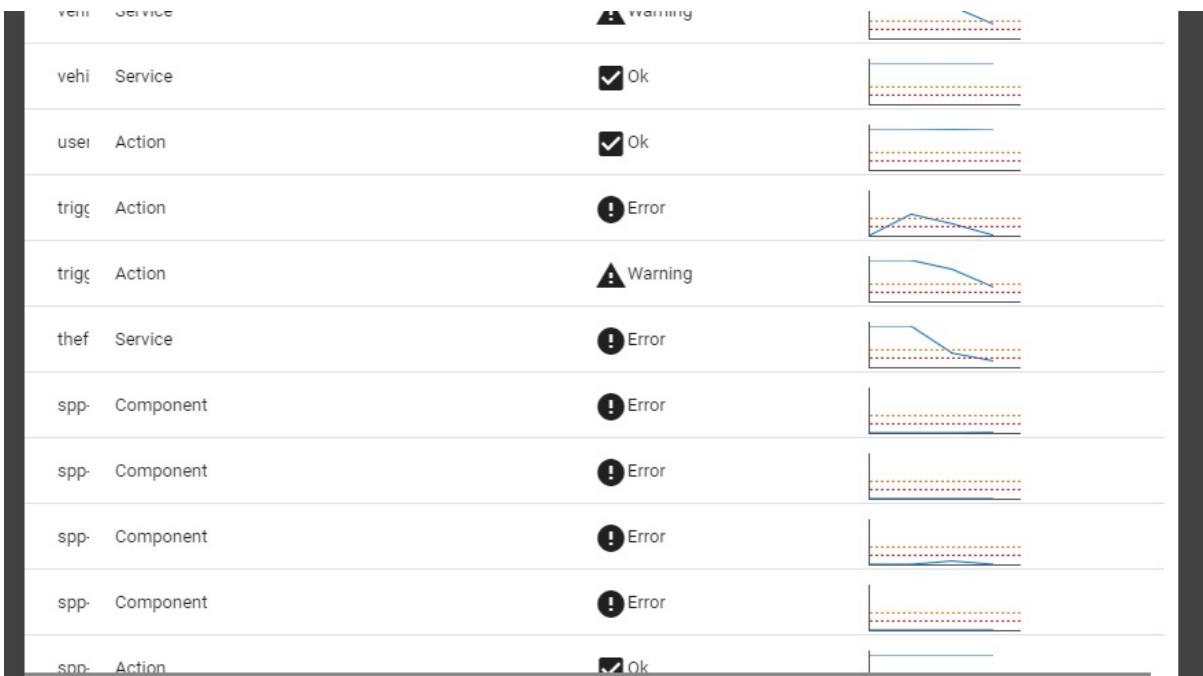
/subscriptions/4025db3d-bb6f-4de1-8485-  
36a843f29882/resourceGroups/spp-monitoring-healthstate-  
Dev01/providers/Microsoft.Logic/workflows/sppsendtoeventhubhourlyDev01



**Close**

- **"Line Chart" representation format**

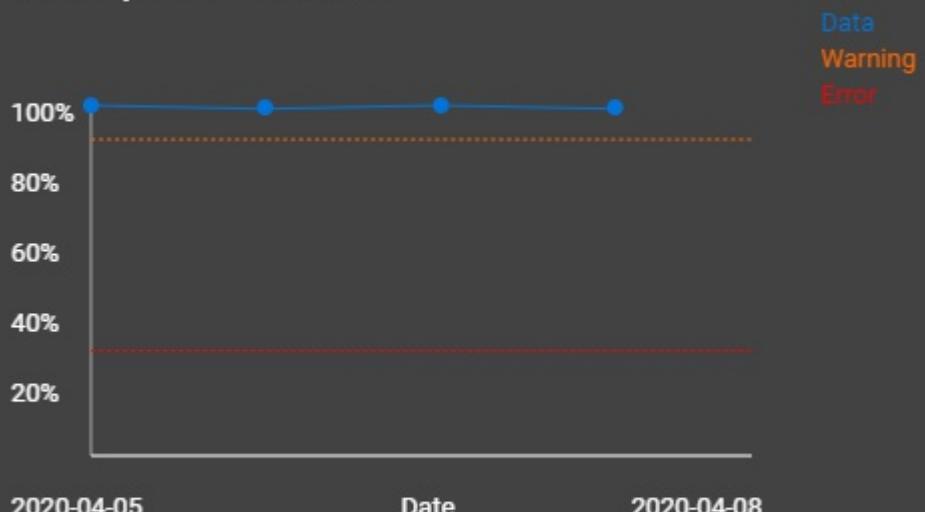
The SLA line chart correlates also on the SLA uptime.



**Mouse double click** - on a corresponding line chart shows a detail information of the data.

## SLA report

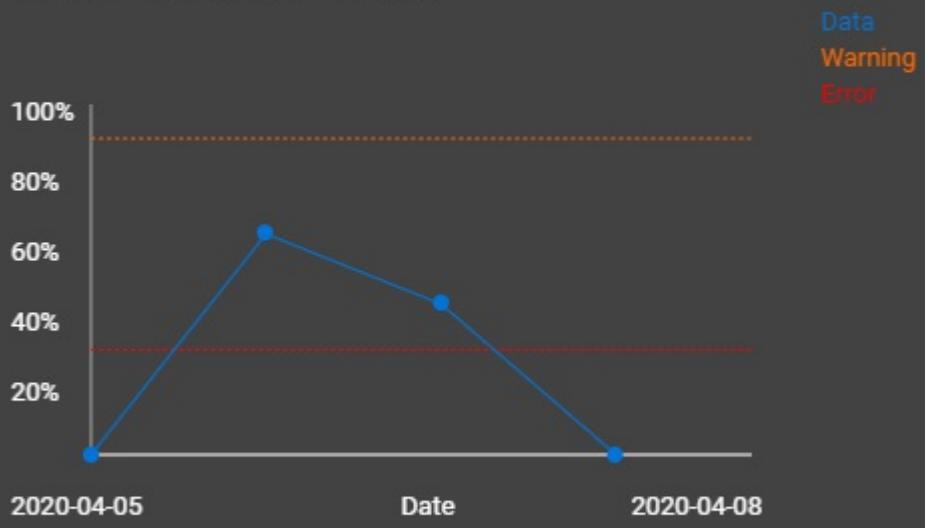
vehicleSupervisionServiceDev01



[Close](#)

## SLA report

sendEmailNotificationsActionDev01



[Close](#)

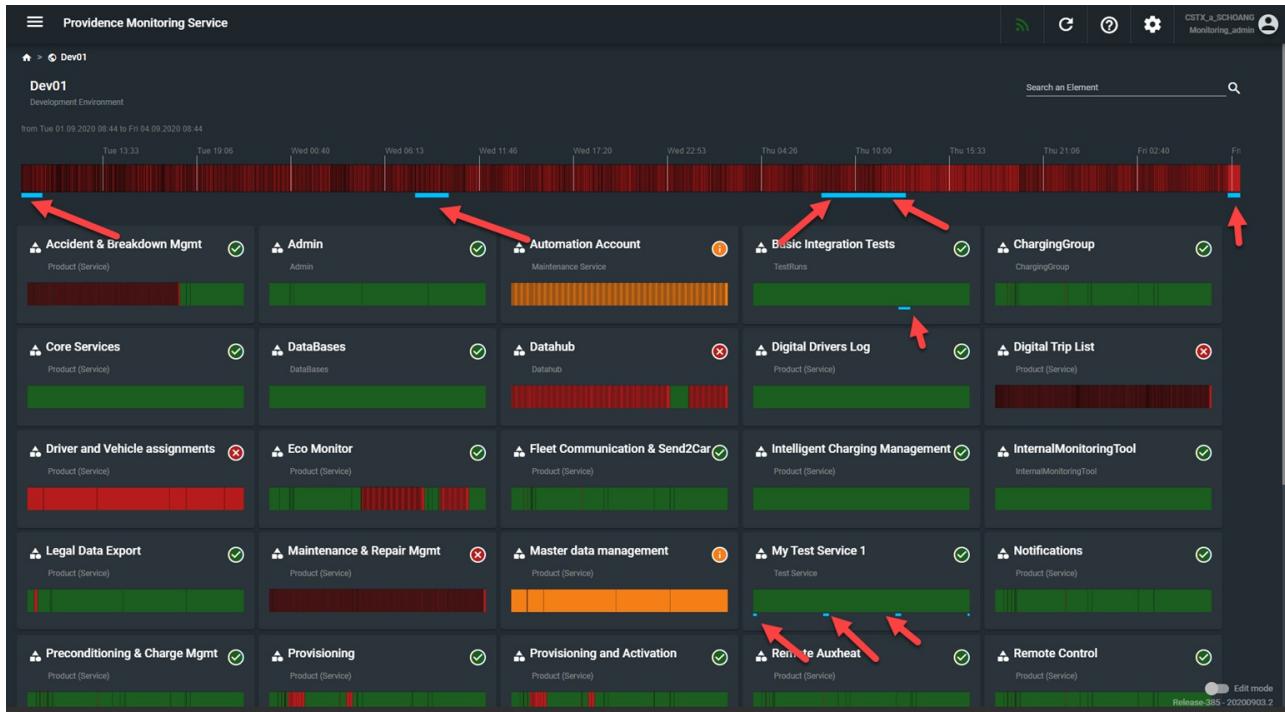
## ## Deployments

The Deployment notifies the UI when in the UI timeline a deployment window should be shown.

The sense for this behaviour is that the Operations-team can consider the deployments if a state of a element is in state "ERROR".

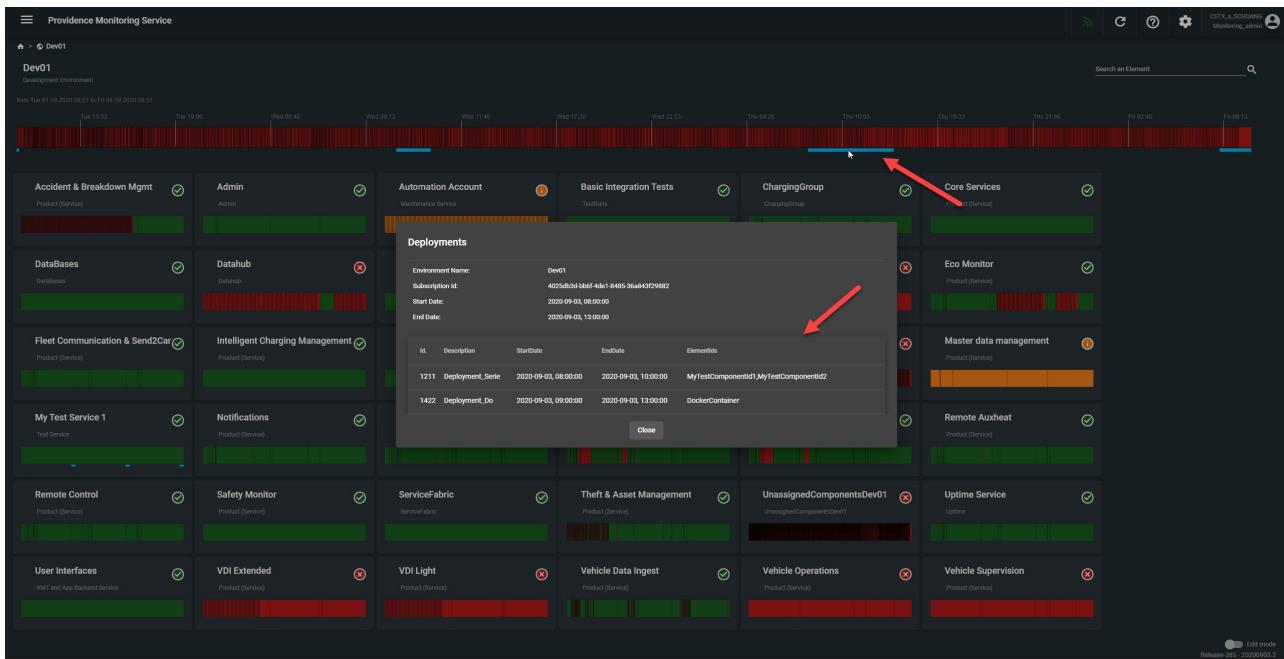
Screenshot of the timeline showing the deployment slot.

The deployment window is marked in blue color, means in that time a deployment of a service is running.



### Deployment Details:

With double click on the deployment line, following deployments details are displayed



## Deployments

Environment Name: Dev01  
 Subscription Id: 4025db3d-bb6f-4de1-8485-36a843f29882  
 Start Date: 2020-09-03, 08:00:00  
 End Date: 2020-09-03, 13:00:00

ID	Description	Start Date	End Date	Element IDs
1211	Deployment_Serie	2020-09-03, 08:00:00	2020-09-03, 10:00:00	MyTestComponentId1, MyTestComponentId2
1422	Deployment_Do	2020-09-03, 09:00:00	2020-09-03, 13:00:00	DockerContainer

[Close](#)

- Environment Name
- Subscription ID
- Start Date & Time
- End Date & Time
- Detail of individual Deployment:
  - Deployment ID
  - Description
  - Start Date & Time
  - End Date & Time
  - ElementID's

**Deployment Overview:**

Providence Monitoring Service

Deployment Window Overview

Start Date	Environment Name	Subscription Id	Deployment Id	Description	Short Description	Close Reason	End Date	Length	Occurs...	ElementIds
2020-09-03, 09:00	Dev01	4025db3d-bb6f-4d..	1422	Deployment_Do	Deployment_Do		2020-09-03, 13:00	4h	1	DockerContainer
2020-09-02, 08:00	Dev01	4025db3d-bb6f-4d..	1214	Deployment_Serie	Deployment_Serie		2020-09-02, 10:00	2h	2	MyTestComponen..
2020-09-01, 08:00	Dev01	4025db3d-bb6f-4d..	1212	Deployment_Serie	Deployment_Serie		2020-09-01, 10:00	2h	2	MyTestComponen..
2020-09-03, 08:00	Dev01	4025db3d-bb6f-4d..	1211	Deployment_Serie	Deployment_Serie		2020-09-03, 10:00	2h	2	MyTestComponen..
2020-09-04, 08:00	Dev01	4025db3d-bb6f-4d..	1210	Deployment_Serie	Deployment_Serie		2020-09-04, 10:00	2h	2	MyTestComponen..
2020-09-01, 02:00	Env_MK	ENV_MK	581	Deployment_Serie	Deployment_Serie		2020-09-01, 04:00	2h	2	Component2
2020-09-02, 02:00	Env_MK	ENV_MK	576	Deployment_Serie	Deployment_Serie		2020-09-02, 04:00	2h	2	Component2
2020-09-03, 02:00	Env_MK	ENV_MK	575	Deployment_Serie	Deployment_Serie		2020-09-03, 04:00	2h	2	Component2
2020-08-31, 02:00	Env_MK	ENV_MK	574	Deployment_Serie	Deployment_Serie		2020-08-31, 04:00	2h	2	Component2
2020-09-02, 18:00	Env_MK	ENV_MK	520	Deployment_Mittw..	Deployment_Mittw..		2020-09-02, 23:00	5h	1	Component3, Com..
2020-09-03, 09:00	Env_MK	ENV_MK	103	MyD5	MyD5		2020-09-03, 12:00	3h	1	Component3, Com..
2020-09-03, 08:00	Env_MK	ENV_MK	102	MyD4	MyD4		2020-09-03, 10:00	2h	1	Component3
2020-09-02, 14:00	Env_MK	ENV_MK	100	MyD3	MyD3		2020-09-02, 14:30	30min	1	Component3
2020-09-02, 13:00	Env_MK	ENV_MK	97	MyD2	MyD2		2020-09-02, 15:00	2h	1	Component2
2020-09-02, 08:00	Env_MK	ENV_MK	89	MyD1	MyD1		2020-09-02, 11:00	3h	1	Component1
2020-09-02, 02:00	Test_Angelika	9999999999999999	788	Deployment_Serie	Deployment_Serie		2020-09-02, 04:00	2h	2	fabric:/Daimler.Va..

### Add new Deployment:

Click on "Add" icon to add an new deployment.

## Deployment

Description \*

Short Description \*

Environment Name \*

ElementId(s)

Once     Serie

Start Date:

Start time  
For example: 12:33

End Date:

End time  
For example: 12:33

Close Reason

**Save**    **Cancel**

It is possible to either create one single deployment or to create a set of recurring deployments. If "Once" is chosen the following fields must be filled:

- Description
- Short Description
- Select the environment that is affected
- Select or Set the affected ElementId(s)
- Set the option "Once" deployment
- Set the start date or choose a date
- Set the start time
- Set the end date or choose a date
- Set the end time
- Set Reason for close

## Deployment

Description \*  
**Deployment\_Do**

Short Description \*  
**Deployment\_Do**

Environment Name  
Dev01

ElementId(s)  
**DockerContainer**

Once     Serie

---

Start Date:   Choose a date \*

End Date:   Choose a date \*

Close Reason

Start time  
09:00  
For example: 12:33

End time  
13:00  
For example: 12:33

### Add Deployment (recurring):

If "Serie" is chosen, to create a whole deployment set, the following fields must be filled:

- Description
- Short Description
- Select the environment that is affected
- Select or Set the affected ElementId(s)
- Set the option "Serie" deployment
- Set the start time
- Set the end time
- Set Reason for close
- Set the start date
- Choose the pattern of rescheduling ( Daily, Weekly, Monthly)  
(e.g. "Daily" and "every 2 days" for a deployment that will reoccur every two days)
  - Daily --> Choose day interval
  - Weekly --> Choose weeks interval (Mo - Su)
  - Monthly --> Choose monthly interval on the ... day of month (Mo - Su)
- Choose the duration date
- Choose "No Enddate"(default + 1 year) or the "End date"

Optional: choose an end date (otherwise the deployment will reoccur for one year)

## Deployment Set

Description \*  
**Deployment\_Serie**

Short Description \*  
**Deployment\_Serie**

Environment Name  
Dev01

Elementid(s)  
**MyTestComponentId1**  **MyTestComponentId2**

Once  Serie

---

### Deployment Duration

Start time \* 08:00  
For example: 12:33

End time \* 10:00  
For example: 12:33

Close Reason

---

### Pattern of rescheduling

Daily  
 Weekly  
 Monthly

Every 1 Weeks  
 Monday  Tuesday  Wednesday  Thursday  Friday  Saturday  
 Sunday

---

### Duration

Start Date:

No Enddate  
 End Date:

---

**Save** **Cancel**

#### Modify Deployment:

Double-click on the deployment or select the deployment and click on the "Edit" icon

adVANce Monitoring Service

Deployment Window Overview

Start Date	Environment N...	Subscription Id	Deployment Id	Description	Short Descript...	Close Reason	End Date	Length	Occurance
2019-12-04T12:00:00Z	DevFe	2609	Mo-MI_Deployme...	Mo-MI_Deployme...			2019-12-04T13:0...	1h	
2019-11-05T12:00:00Z	DevFe	2597	Mo-MI_Deployme...	Mo-MI_Deployme...			2019-11-05T13:0...	1h	
2019-11-04T12:00:00Z	DevFe	2460	Mo-MI_Deployme...	Mo-MI_Deployme...	test		2019-11-04T13:0...	1h	
2019-10-15T22:00:00Z	testenv_rk	2637	test	test			Ongoing...		1
2019-10-11T06:00:00Z	DevFe	2583	Once Deployment	Once Deployment			Ongoing...		1

Deployment "Once":

## Deployment

Description \*  
Deployment\_Do|

Short Description \*  
Deployment\_Do

Environment Name  
Dev01

Elementid(s)  
DockerContainer

Once  Serie

Start Date: Choose a date \*  
9/3/2020

End Date: Choose a date \*  
9/3/2020

Start-time  
09:00  
For example: 12:33

End time  
13:00  
For example: 12:33

Close Reason

Deployment "Serie":

## Deployment Set

Description \*

Deployment\_Serie

Short Description \*

Deployment\_Serie

Environment Name

Dev01

ElementId(s)

MyTestComponentId1

MyTestComponentId2

Once  Serie

### Deployment Duration

Start time \*

08:00

For example: 12:33

End time \*

10:00

For example: 12:33

Close Reason

### Pattern of rescheduling

Daily

Every 1 Weeks

Weekly

Monday  Tuesday  Wednesday  Thursday  Friday  Saturday

Monthly

Sunday

### Duration

Start Date:

Choose a date \*

9/2/2020



No Enddate

Choose a date \*

End Date:

9/4/2021



**Save**

**Cancel**

### Delete Deployment (Once/Serie):

It is not possible to delete a serial deployment as a whole, because these are created as individual ones per day of the week. Deletion must be done manually and individually.

Select the deployment and click on the "Delete" Icon

Providence Monitoring Service

Start Date	Environment Name	Subscription ID	Deployment ID	Description	Short Description	Close Reason	End Date	Length	Occurrences	Elements
2020-09-03, 09:00	Dev01	4025bd3d-bbaf-4de1-8485-36..	1422	Deployment_Do	Deployment_Do		2020-09-03, 19:00	4h	1	DockerContainer
2020-09-02, 08:00	Dev01	4025bd3d-bbaf-4de1-8485-36..	1214	Deployment_Serie	Deployment_Serie		2020-09-02, 10:00	2h	1	My/TestComponentId1, MyTes..
2020-09-01, 08:00	Dev01	4025bd3d-bbaf-4de1-8485-36..	1212	Deployment_Serie	Deployment_Serie		2020-09-01, 10:00	2h	1	My/TestComponentId1, MyTes..
2020-09-03, 08:00	Dev01	4025bd3d-bbaf-4de1-8485-36..	1211	Deployment_Serie	Deployment_Serie		2020-09-03, 10:00	2h	1	My/TestComponentId1, MyTes..
2020-09-04, 08:00	Dev01	4025bd3d-bbaf-4de1-8485-36..	1210	Deployment_Serie	Deployment_Serie		2020-09-04, 10:00	2h	1	My/TestComponentId1, MyTes..
2020-09-01, 02:00	Env_MK	ENV_MK	581	Deployment_Serie	Deployment_Serie		2020-09-01, 04:00	2h	1	Component2
2020-09-02, 02:00	Env_MK	ENV_MK	576	Deployment_Serie	Deployment_Serie		2020-09-02, 04:00	2h	1	Component2
2020-09-03, 02:00	Env_MK	ENV_MK	575	Deployment_Serie	Deployment_Serie		2020-09-03, 04:00	2h	1	Component2
2020-09-31, 02:00	Env_MK	ENV_MK	574	Deployment_Serie	Deployment_Serie		2020-09-31, 04:00	2h	1	Component2
2020-09-02, 18:00	Env_MK	ENV_MK	520	Deployment_Mittwoch	Deployment_Mittwoch		2020-09-02, 23:00	5h	1	Component1, Component1

## Delete this Deployment?

Are you sure you want to delete this Deployment?

**Delete**

**Cancel**

\*\*\*

# Alert Ignore Rules

The Alert Ignore Overview of the Providence Service supports operations for creating, updating and deleting alert ignore rules.

Alert ignore rules can be used to mute specific alerts. For example all alerts with a specific CheckId can be muted to avoid StateTransitions resulting from their occurrence in the Providence Dashboard.

## Alert Ignore Rule Overview:

The screenshot shows a table with columns: Description, Environment Name, Creation Date, Expiration Date, Component Id, Check Id, Alert Name, and Ignore Conditions. The table contains five rows of data:

Description	Environment Name	Creation Date	Expiration Date	Component Id	Check Id	Alert Name	Ignore Conditions
test	DevB2B	2019-08-12, 15:29	2019-08-13, 15:00	action2	metricalert	(AlertName"null","ComponentId"action..)	
CPU	DevB2B	2019-08-12, 14:53	2019-08-14, 13:15	component1	metricalert	CPU	(AlertName"CPU","ComponentId"com..)
Test_2	DevB2B	2019-08-12, 14:14	2019-08-13, 12:00	Test	Test	Test	(AlertName"Test","ComponentId"Test..)
CPU	DevB2B	2019-08-12, 14:11		action1	metricalert	(AlertName"null","ComponentId"action..)	

## Add new Alert Ignore Rule:

Click on "Add" icon to add new alert ignore rule.

The screenshot shows the same table as above, but with a red arrow pointing to the blue '+' icon in the top right corner of the header bar.

The following fields must be filled in:

- Description
- Select the environment that is affected
- Set the expiration date
- Select the alert property which shall be ignored and set a filter text (combination of multiple properties possible)

The screenshot shows the 'Ignore alert' configuration dialog with the following fields:

- Description: CPU
- Environment Name: DevB2B
- Expiration Date: 8/14/2019
- Specify the time: 13:15
- Add or remove filter criteria (table):

Alert ↑	Filter Text
alertName	CPU
checkId	metricalert
componentId	component1
customField1	
customField2	
customField3	
customField4	
- Save and Cancel buttons at the bottom.

## Modify Alert Ignore Rule:

Double-click on the rule you want to update or  
Select the alert filter and click on the "Edit" icon

Description	Environment Name	Creation Date	Expiration Date	Component ID	Check ID	Alert Name	Ignore Conditions
test	DevB2B	2019-08-12, 15:29	2019-08-13, 15:00	action2	metricalert		{AlertName":null,"ComponentId":action..}
<input checked="" type="checkbox"/> CPU	DevB2B	2019-08-12, 14:53	2019-08-14, 13:15	component1	metricalert	CPU	{AlertName":"CPU","ComponentId":com..}
Test_2	DevB2B	2019-08-12, 14:14	2019-08-13, 12:00	Test	Test	Test	{AlertName":Test,"ComponentId":Test..}
CPU	DevB2B	2019-08-12, 14:11		action1	metricalert		{AlertName":null,"ComponentId":action..}

Make changes and save to complete.

**Ignore alert**

Description \*  
CPU

Environment Name \*  
DevB2B

Expiration Date: 8/14/2019

Specify the time  
13:15

Add or remove filter criteria

Alert ↑	Filter Text
alertName	CPU
checkid	metricalert
componentid	component1
customField1	
customField2	
customField3	
customField4	

Save    Cancel

#### Delete Alert Ignore Rule:

Select the alert ignore rule and click on the "Delete" icon

Description	Environment Name	Creation Date	Expiration Date	Component ID	Check ID	Alert Name	Ignore Conditions
test	DevB2B	2019-08-12, 15:29	2019-08-13, 15:00	action2	metricalert		{AlertName":null,"ComponentId":action..}
<input checked="" type="checkbox"/> CPU	DevB2B	2019-08-12, 14:53	2019-08-14, 13:15	component1	metricalert	CPU	{AlertName":"CPU","ComponentId":com..}
Test_2	DevB2B	2019-08-12, 14:14	2019-08-13, 12:00	Test	Test	Test	{AlertName":Test,"ComponentId":Test..}
CPU	DevB2B	2019-08-12, 14:11		action1	metricalert		{AlertName":null,"ComponentId":action..}

Delete Alert Ignore?

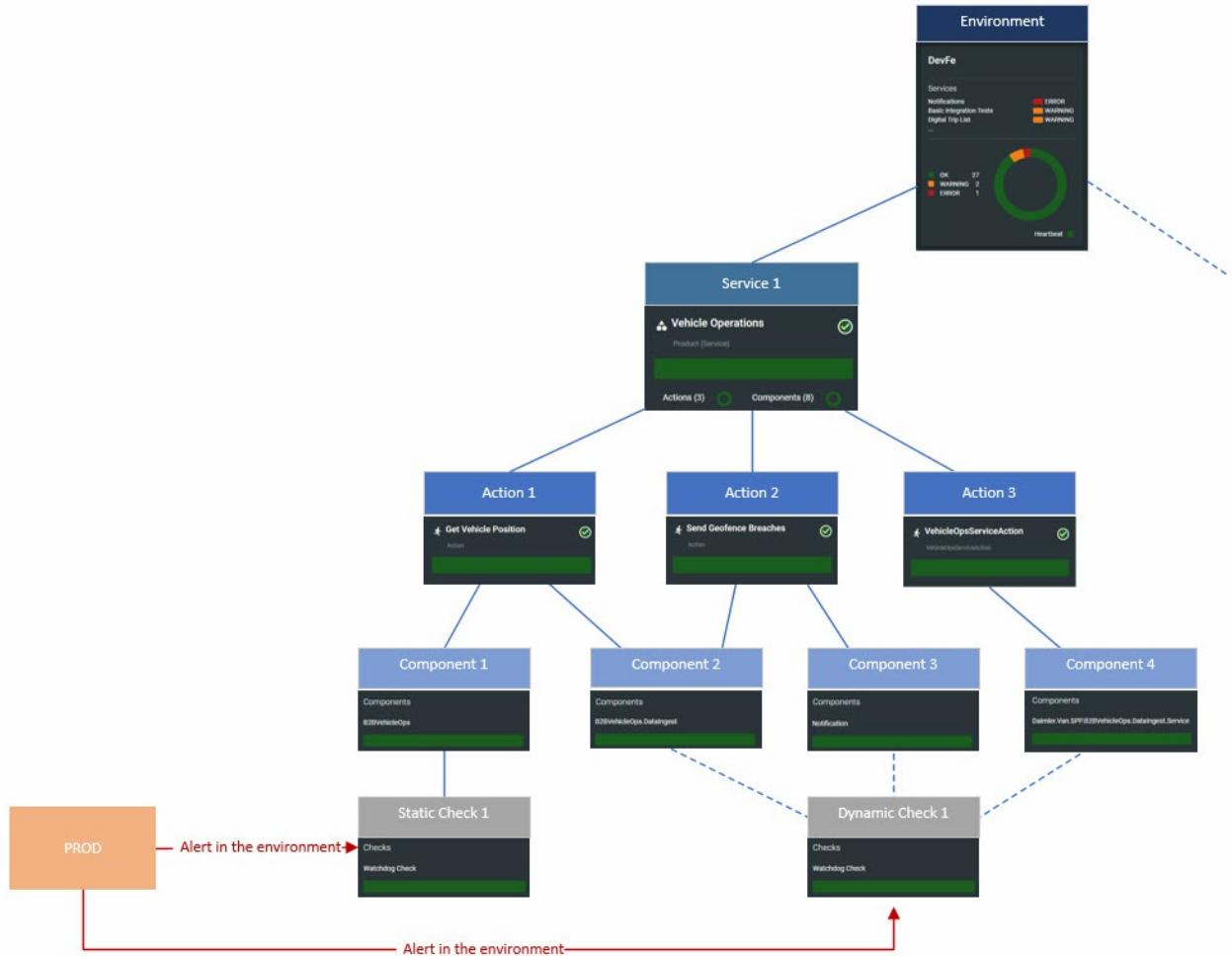
Are you sure you want to delete the Alert Ignore(s)?

**Delete**    Cancel

# Checks

Checks are used as a mechanism to avoid unknown data to be processed by the Providence service backend. Only alerts with a valid and known "CheckId" property are processed by the service. If a valid "CheckId" was found the alert changes the state of all elements which are linked to the Check within the environment. Depending on the state defined in the alert the state then changes to "ERROR", "WARNING" or "OK".

Besides the functionality as a "Validation Gateway" for alerts checks are also used to configure other different features like for example the "Reset to Green" feature. You can read more about this feature within the "Internal Jobs" section.



## Check Overview:

Screenshot of the "Van Check Overview" page in the adVANCE Monitoring Service. The page shows a table of checks across various environments.

Environment Name	Check ID	Check Name	Description	Frequency
DevFe	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	1050
DevFe	MetricAlert	Metric Check	Metric check for Azure resources	-1
DevFe	TestAutomation	TestAutomation	TestAutomation	-1
DevFe	Watchdog	Watchdog Check	Watchdog check for microservices	-1
TestB2B	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	1520
TestB2B	MetricAlert	Metric Check	Metric check for Azure resources	-1
TestB2B	TestAutomation	TestAutomation	TestAutomation	-1
TestB2B	Watchdog	Watchdog Check	Watchdog check for microservices	-1

## Add Check:

Click on "Add" icon to add a service check.

Environment Name	Check Id	Check Name	Description	Frequency
Dev_mock	checkAuxHeat	AuxHeatCheck	Heating check	60
Dev_mock	checkEventHub	EventHubCheck	check for the Eventhub	60
Dev_mock	DynamicCheck	DynamicCheck	Dynamic Check	30
Dev_mock	MetricAlertId	MetricAlertMock	MetricAlert Sum	60
Dev01	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	1520

The following fields must be filled in:

- Check Id
- Select the environment that is affected
- Name
- Description
- VSTS Link
- Select "Specify a reset frequency"
- Set the frequency in second (10 - 1800 seconds)
- Save the settings

**New Check**

Check Id \*

**checkAuxHeat**

Environment Name \*

**Dev\_mock**

Name \*

**AuxHeatCheck**

Description

**Heating check**

VSTS Link

**<https://vanstelmatics.visualstudio.com/>**

**Specify a reset frequency**

Frequency:  **60 s**

**Save**      **Cancel**

**Modify Check:**

Double-click on the service check or Select the service check and click on the "Edit" icon

Environment Name	Check Id	Check Name	Description	Frequency
Environment Name 1	checkId 1	AuxHeatCheck	Heating check	60
Dev_mock	checkAuthHeat	AuxHeatCheck	Heating check	60
Dev_mock	checkEventhub	EventhubCheck	check for the Eventhub	60
Dev_mock	DynamicCheck	DynamicCheck	Dynamic Check	30
<b>Dev_mock</b>	<b>MetricAlertOld</b>	<b>MetricAlertMock</b>	<b>MetricAlert Sum</b>	<b>60</b>
Dev01	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	1520

- Make changes and save to complete.

## Edit Check

Check Id  
**MetricAlertOld**

Environment Name  
**Dev\_mock**

Name \*  
**MetricAlertMock**

Description  
**MetricAlert Sum**

VSTS Link  
<https://vanstelematics.visualstudio.com/>

Specify a reset frequency

Frequency:  60 s

**Save**      **Cancel**

### Delete Check:

Select the service check and click on the "Delete" icon

Environment Name	Check Id	Check Name	Description	Frequency
Environment Name 1	checkId 1	AuxHeatCheck	Heating check	60
Dev_mock	checkAuthHeat	AuxHeatCheck	Heating check	60
Dev_mock	checkEventhub	EventhubCheck	check for the Eventhub	60
Dev_mock	DynamicCheck	DynamicCheck	Dynamic Check	30
<b>Dev_mock</b>	<b>MetricAlertOld</b>	<b>MetricAlertMock</b>	<b>MetricAlert Sum</b>	<b>60</b>
Dev01	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	GenericLogAnalyticsQueryAlert	1520

## Delete Check?

Are you sure you want to delete the Check(s)?

 Delete

Cancel

# Notification Rules

In order to receive emails from the Providence Service notification rules have to be configured.

These rules contain information about the user which wants to be notified, the elements the user want to be notified about and also a notification delay interval after what time he want to be notified about specific state changes.

## Notification Overview:

Environment Name	Levels	States	Addresses	Notification Interval (In Minutes)	Activation State
DevB2B	Environment, Service	ERROR	angellika kraenzler@daimler.com	30	true
DevB2B	Action, Component	WARNING	angellika.kraenzler@daimler.com	50	true
DevFe	Environment, Service	ERROR	Admin@byom.de	60	false

## Add notification rule:

Click on "Add" icon to add a notification rule.

Environment Name	Levels	States	Addresses	Notification Interval (In Minutes)	Activation State
DevB2B	Environment, Service	ERROR	angellika kraenzler@daimler.com	30	true
DevB2B	Action, Component	WARNING	angellika.kraenzler@daimler.com	50	true
DevFe	Environment, Service	ERROR	Admin@byom.de	60	false

The following fields must be filled:

- Toggle "Activation"
- Parameter settings:
  - Select environment
  - Set notification delay interval (from 1 - 90 minutes)
  - Select levels (Environment, Service, Action, Component)
  - Select states (Error, Warning)

## Notification Rule

Activation

**1 Params**

Environment Name \*

DevB2B

Notification Interval (min):

Levels States

<input checked="" type="checkbox"/> Environment	<input type="checkbox"/> ERROR
<input type="checkbox"/> Service	<input checked="" type="checkbox"/> WARNING
<input checked="" type="checkbox"/> Action	
<input type="checkbox"/> Component	

Save Cancel

- Address settings:

- Click on the "Edit" icon to set the Adresse:

## Notification Rule

Activation

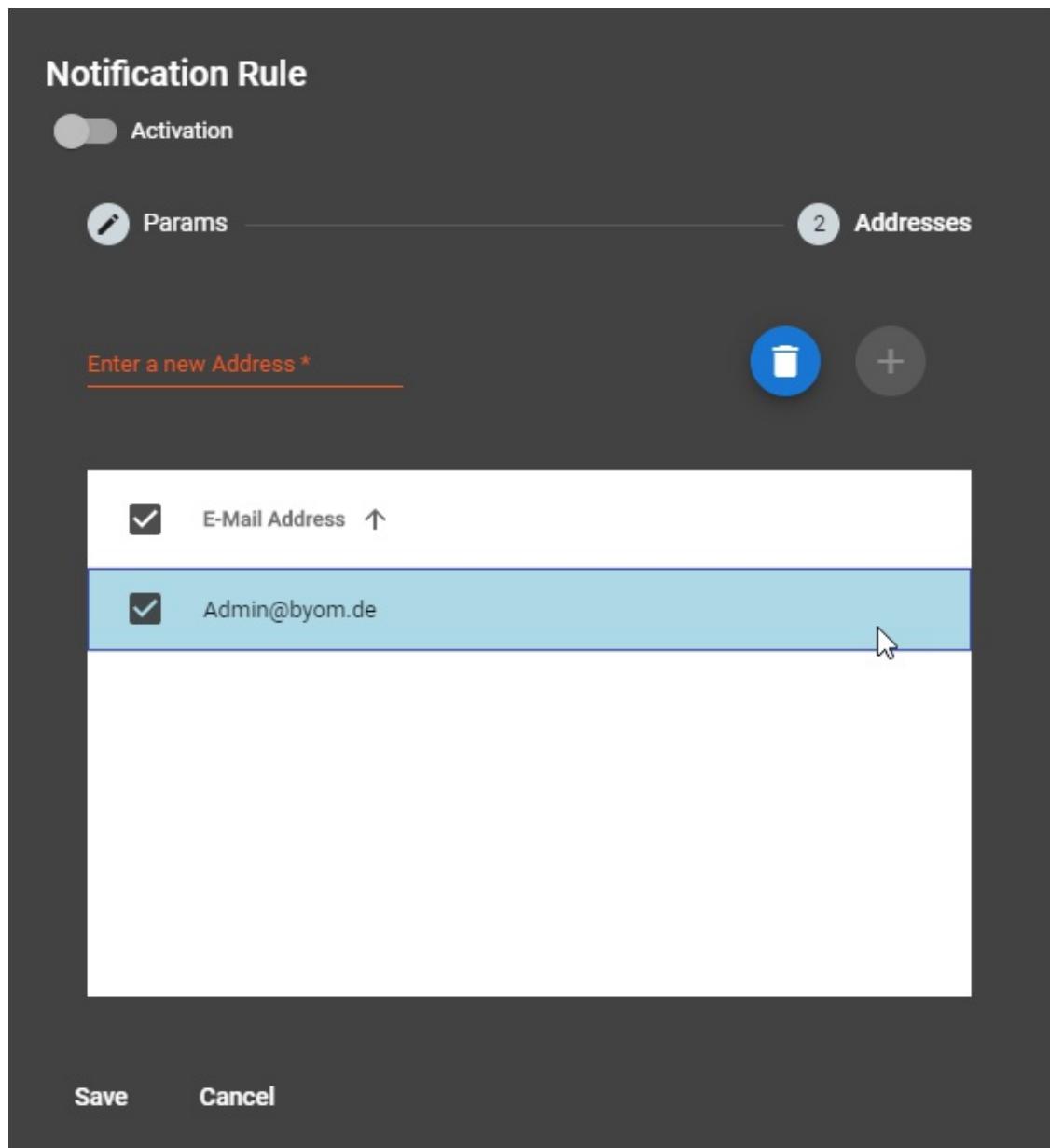
**1 Params**

Environment Name \*

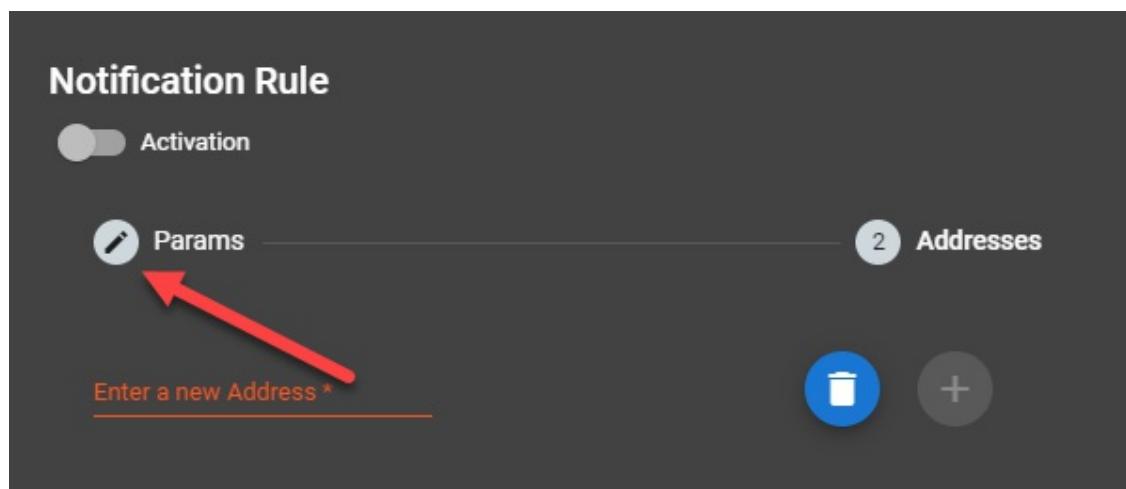
DevB2B

Addresses

- Enter a new e-mail adress and click on the "Add" icon
  - Select the new E-Mail address



- Click on the "Edit" icon to set the Parameter:



- Save the notification settings

## Notification Rule

Activation

1 Params Addresses

Environment Name \*  
DevFe

Notification Interval (min):

Levels	States
<input checked="" type="checkbox"/> Environment	<input checked="" type="checkbox"/> ERROR
<input checked="" type="checkbox"/> Service	<input type="checkbox"/> WARNING
<input type="checkbox"/> Action	
<input type="checkbox"/> Component	

**Save** **Cancel**

### Modify notification rule:

Double-click on the notification rule or

Select the service notification and click on the "Edit" icon

Environment Name	Levels	States	Addresses	Notification Interval (In Minutes)	Activation State
DevB2B	Environment, Service	ERROR	angela.kraenzle@daimler.com	30	false
<input checked="" type="checkbox"/> DevB2B	Action, Component	WARNING	angela.kraenzle@daimler.com	50	false
DevFe	Environment, Service	ERROR	Admin@byom.de	60	false

- Make changes and save to complete.

## Notification Rule

Activation

**1 Params** **2 Addresses**

Environment Name \*  
**DevB2B**

Notification Interval (min):

Levels	States
<input type="checkbox"/> Environment	<input type="checkbox"/> ERROR
<input type="checkbox"/> Service	<input checked="" type="checkbox"/> WARNING
<input checked="" type="checkbox"/> Action	
<input checked="" type="checkbox"/> Component	

**Save** **Cancel**



Delete service notification rule:

Select the notification rule and click on the "Delete" icon

adVANCE Monitoring Service

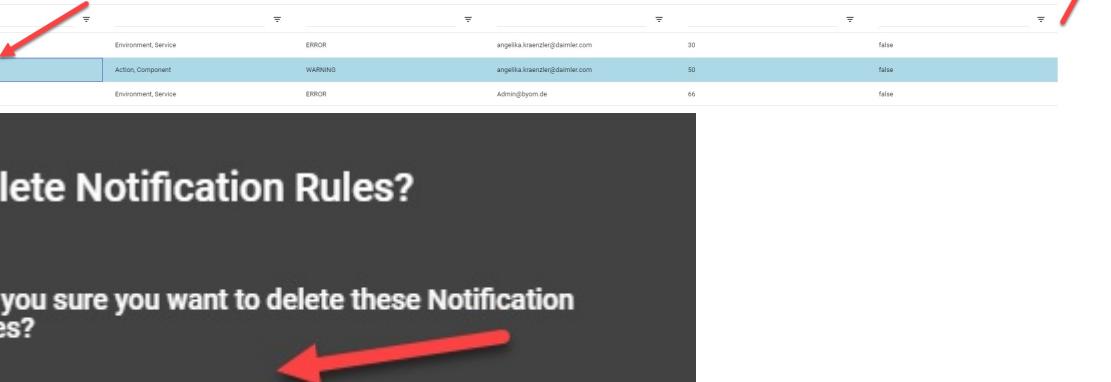
Notification Rule Overview

Environment Name	Levels	States	Addresses	Notification Interval (In Minutes)	Activation State
<input type="checkbox"/> DevB2B	Environment, Service	ERROR	angellika.kraenzer@daimler.com	30	false
<input checked="" type="checkbox"/> DevB2B	Action, Component	WARNING	angellika.kraenzer@daimler.com	50	false
<input type="checkbox"/> DevFe	Environment, Service	ERROR	Admin@byom.de	60	false

**Delete Notification Rules?**

Are you sure you want to delete these Notification Rules?

**Delete** **Cancel**



# State Increase Rules

The State Increase Rule overview of the Providence Service supports operations for creating, updating and deleting specific rules.

State Increase Rules can be used to increase the state of specific alerts after a specified period of time.

The state increase job checks in a specific period of time (configurable in app settings) if a check is in the state "WARNING" for a specific duration.

If so the job increases the state to "ERROR" to give this component more attention within the dashboard.

**State Increase Rule overview:**

The screenshot shows the 'adVANCE Monitoring Service' interface with the title 'State Increase Rule Overview'. On the left, there is a sidebar with navigation links: Dashboard, Deployments, Alert Ignores, Checks, Notification Rules, and State Increase Rules. The main area displays a table with two rows of data. The columns are: Active (checkbox), Title, Environment Name, Description, Component Id, Check Id, Alert Name, and Trigger Time [min]. The first row has 'true' in the Active column and 'Test StateIncrease...' in the Title column. The second row also has 'true' in the Active column and 'StateRule' in the Title column. The table has a header row with these column names.

Active	Title	Environment Name	Description	Component Id	Check Id	Alert Name	Trigger Time [min]
true	Test StateIncrease...	DevB2B	Rule to increase sp...	component1	metricalert	CPU	482
true	StateRule	DevB2B	Rule to increase sta...	component1	actioncheck2	Memory	202

**Add State Increase Rule:**

Click on "Add" icon to add a new rule.

The screenshot shows the same 'adVANCE Monitoring Service' interface as the previous one, but with a red arrow pointing to the blue '+' icon located at the top right of the main content area. This icon is used to add a new State Increase Rule.

The following fields must be filled:

- Title
- Description
- Select the environment
- Select the Component Id
- Select the Check Id
- Set the Alert Name
- Set "Activate this Rule" (yes/no)
- Set Trigger time (from 1 - 720 minutes)

## State Increase Rule

Title \*  
StateRule

Description \*  
Rule to increase state

Environment Name \*  
DevB2B

Component Id \*  
component1

Check Id \*  
actioncheck2

Alert Name  
Memory

Activate this Rule

Trigger Time:  202 min

**Save**    **Cancel**

### Modify State Increase Rule:

- Double-click on the rule or select the rule and click on the "Edit" icon

Active	Title	Environment Name	Description	Component Id	Check Id	Alert Name	Trigger Time [min]
<input checked="" type="checkbox"/>	StateRule	DevB2B	Rule to increase state	component1	actioncheck2	Memory	202
<input type="checkbox"/>	Test StateIncreaseRule	DevB2B	Rule to increase specify Alerts	component1	metricalert	CPU	482

- Make changes and save to complete.

## State Increase Rule

**Title \***  
**StateRule**

**Description \***  
**Rule to increase state**

**Environment Name \***  
**DevB2B**

**Component Id \***  
**component1**

**Check Id \***  
**actioncheck2**

**Alert Name**  
**Memory**

**Activate this Rule**

**Trigger Time:**  202 min

**Save**      **Cancel**

### Delete State Increase Rule:

Select the rule and click on the "Delete" icon

Active	Title	Environment Name	Description	Component Id	Check Id	Alert Name	Trigger Time [min]
<input checked="" type="checkbox"/>	StateRule	DevB2B	Rule to increase state	component1	actioncheck2	Memory	202
<input type="checkbox"/>	Test StateIncreaseRule	DevB2B	Rule to increase specify alerts	component1	metricalert	CPU	482

**Delete this State Increase Rule?**

Are you sure you want to delete this State Increase Rule?

**Delete**      **Cancel**

# Changelog History The changelog history displays all created, updated or deleted new elements of the Providence service.

All Create/Update/Delete config should be visible in some kind of history backlog.

- If somebody changes an "State Increase Rule" we want to be able to track down who it was.
- If somebody removes a "Alert Ignore Rule" we need to know who did it.

All changes in the model of tree elements (Environment/Service/Action/Component) are exclude because these are changed frequently by the users.

#### Changelog History overview:

The screenshot shows the 'Changelog History' section of the Providence Monitoring Service. The table lists 13 entries of deployment changes made by 'Deployment\_Cleanup\_Job' on 2020-12-09 at 09:51. The columns are: Environment Name, User Name, Change Date, Element Type, Element Id, Operation, Old Value, New Value, and Diff. The 'Diff' column shows the JSON representation of the environment changes.

Environment Name	User Name	Change Date	Element Type	Element Id	Operation	Old Value	New Value	Diff
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3617	Delete	{"Id":3617,"EnvironmentId":...	0	{"Id":3617,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3614	Delete	{"Id":3614,"EnvironmentId":...	0	{"Id":3614,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3612	Delete	{"Id":3612,"EnvironmentId":...	0	{"Id":3612,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3611	Delete	{"Id":3611,"EnvironmentId":...	0	{"Id":3611,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3610	Delete	{"Id":3610,"EnvironmentId":...	0	{"Id":3610,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3609	Delete	{"Id":3609,"EnvironmentId":...	0	{"Id":3609,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3608	Delete	{"Id":3608,"EnvironmentId":...	0	{"Id":3608,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3606	Delete	{"Id":3606,"EnvironmentId":...	0	{"Id":3606,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3603	Delete	{"Id":3603,"EnvironmentId":...	0	{"Id":3603,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3589	Delete	{"Id":3589,"EnvironmentId":...	0	{"Id":3589,0,0,"Environme...
Dev01	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3584	Delete	{"Id":3584,"EnvironmentId":...	0	{"Id":3584,0,0,"Environme...
Angelika-Test-Environment	Deployment_Cleanup_Job	2020-12-09, 09:51	Deployment	3374	Delete	{"Id":3374,"EnvironmentId":...	0	{"Id":3374,0,0,"Environme...

#### Filter Logs:

The filter can be used to limit the period:

- Set the start/end date or choose a date
- Set the start/end time

The screenshot shows the 'Filter Logs' dialog. It contains fields for 'Start Date' and 'End Date', each with a date picker and a time input field labeled 'Start time' and 'End time'. Below the fields are 'Apply Changes' and 'Clear Filter' buttons.

Filter Logs

Start Date: Choose a date \*  Start time  
For example 12:33

End Date: Choose a date \*  End time  
For example 12:33

Apply Changes Clear Filter

## **Internal Jobs**

All internal jobs of the Providence Service are explained below.

The Providence service includes the following internal jobs:

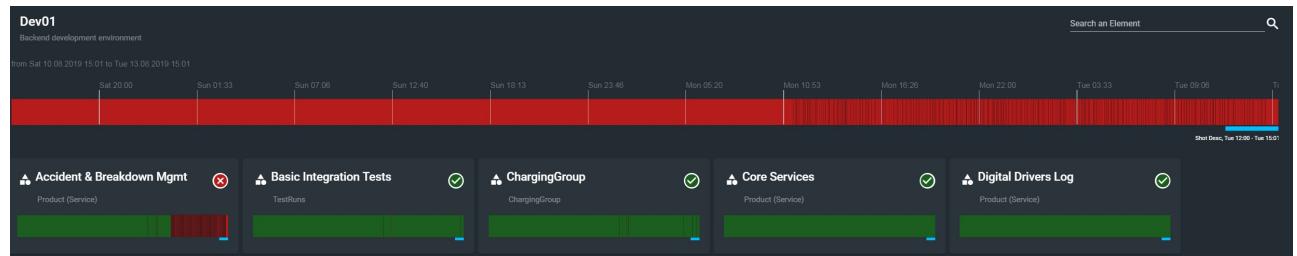
- Deployment Job
- State Increase Job
- Email notifications job
- Delete expired State transitions
- Auto refresh environments
- Reset to green

## Deployment Job

This job notifies the UI when a new deployment which shall be shown in the timeline was created.

The UI then takes this new deployment and sorts it into the timeline so that it is visible as a blue bar under the timeline.

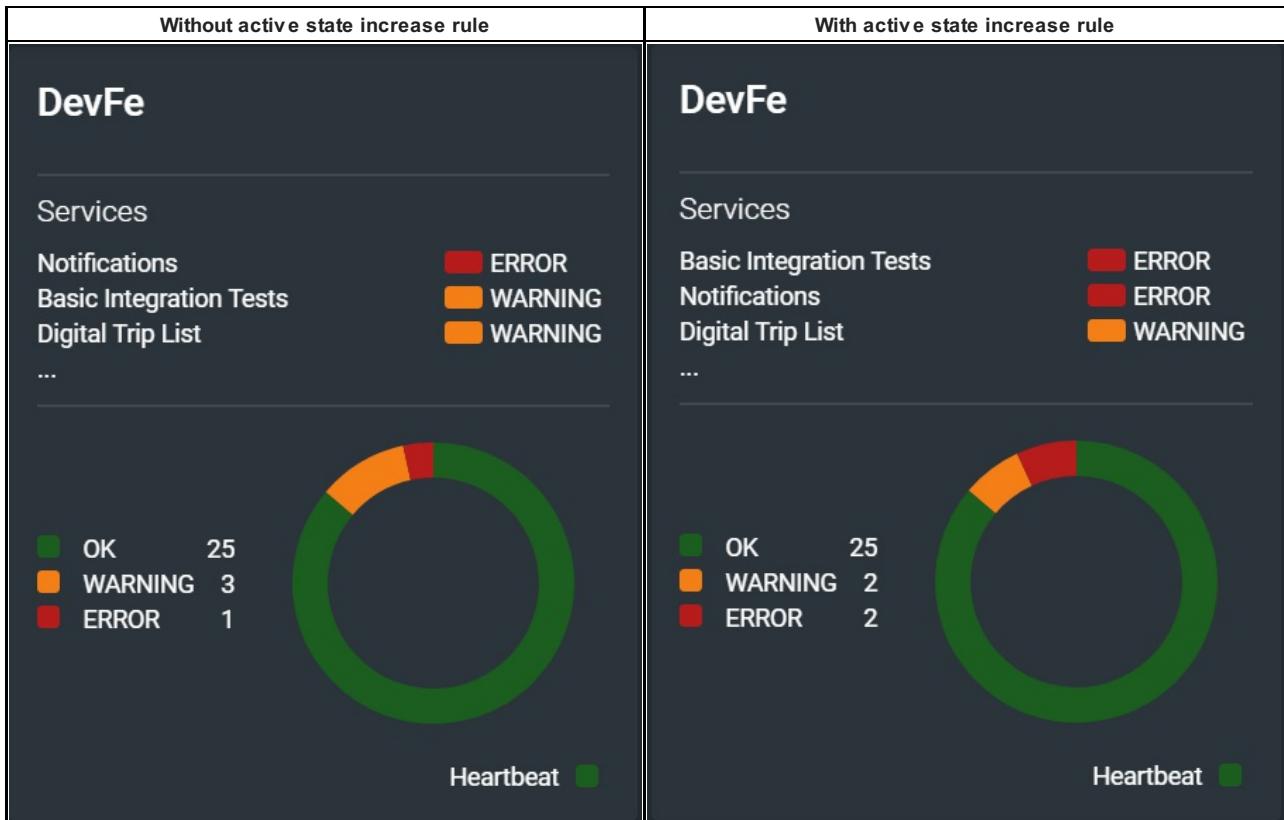
### Screenshot of the timeline showing the deployment slot



## State Increase Job

This job runs in a specified timeintervall and checks if there are elements with the state "Warning" which needs to be increased to the state "Error". The decission whether an element's state needs to be increased or not depends on the configured state increase rules of the system. If the criteria matches the job generates a new alert message to change the state of the element to "Error". In addition, all subsequent alerts that have the status "Warning" are automatically increased to alerts with the status "Error".

### Screenshot with and without an active state increase rule.



## Email Notifications Job This job runs in a specified timeintervall and checks if the states of an element matches the configuration of any notification rules. If so the job notifies the users mentioned in the rule via an email that a specific event occurred.

The emails contains the following information:

- Element (which element is affected)
  - Environment Name
  - Environment ID
  - Notification Level
  - Element ID
  - New State
  - URL to the Environment
- Alert (Which alert is responsible for the email)
  - State
  - Timestamp
  - Check ID
  - Affected component
  - Record ID
  - Alert Description

## Delete expired Elements Job

This job is used to delete expired elements stored in the Providence database after a specific time period (minimum 1 week) in order to reduce the data size within the state transition table and to increase the overall performance of the system.

The settings is environment specific and is configured within the app settings of the Providence service application.

Elements that are deleted by this job are:

- Deployments
- StateTransitions
- ChangeLog Entries
- Unassigned Components

## ## Refresh environments Job

This job is used to clean up the state managers held by the Providence backend. If necessary the job disposes and rebuilds the state managers within the backend. This is for example needed if changes were made directly on the database and need to be transferred into the state manager of a deployed instance.

The settings is environment specific and is configured within the app settings of the Providence service application.

## Reset to Green Job

This job runs in a specified timeintervall and checks if there are elements with the state "Warning" or "Error" which needs to be reseted to the state "Ok". The decission whether an element needs to be resetet or not depends on the alerts which lead to the current state and on the "reset frequency" property of the check which belongs to the received alert. If the criteria matches the job generates a new alert message to change the state of the element to "Ok".

When creating a check the "reset frequency" can be added. The frequency needs to be in the timerange of 10 min to 1440 min.

## # API Documentation

This sub pages include documentation of following Providence APIs:

- Export and Import API
- Notification API
- Deployment API
- Alert Ignore API
- State Increase Rule API

```
# Export and Import API
```

The Export/Import API of the Providence Service supports operations for exporting and importing whole environment trees.

## API Documentation

```
{ {HOST} } = https://spp-monitoringservice-X.azurewebsites.net where X stands for the environment the request is made to.
```

## Export Environment

This endpoint is accessible via: **GET: {{HOST}}/api/masterdata/environmentUpdate?environmentSubscriptionId={environmentSubscriptionId}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.OK** when the export was successful.
- **HttpStatusCode.BadRequest** when an invalid or no environmentSubscriptionId is provided.
- **HttpStatusCode.NotFound** when an unknown environmentSubscriptionId is provided.
- **HttpStatusCode.InternalServerError** when something went wrong during the export process.

## Import Environment

This endpoint is accessible via: **POST: {{HOST}}/api/masterdata/environmentUpdate?instance\_name={{environmentname}}&environmentSubscriptionId={{environmentsubscriptionid}}&replace={Replace}**

Parameters:

- **instance\_name** is a placeholder variable. All occurrences of {instance\_name} will be replaced with the values of these variable.
- The **replace** parameter can have following values:
  - **True**: Hereby all given elements in the payload will be replaced in the according environment. (If they don't exist they will be created)
  - **False**: Hereby the elements in the environment will be supplemented with the elements in the payload.
  - **All**: Hereby the elements in the environment will be overridden by the elements given in the payload.

The response of this endpoint call can be one of the following:

- **HttpStatusCode.OK** when the import was successful.
- **HttpStatusCode.BadRequest** when an invalid or no environmentSubscriptionId is provided.
- **HttpStatusCode.NotFound** when an unknown environmentSubscriptionId or environmentName is provided.
- **HttpStatusCode.InternalServerError** when something went wrong during the export process.

So for example it is possible to export the **dev** environment, replace in the response all appearances of "dev" with {environment\_name}, and after that during the import again all occurrences can be replaced by the right values with passing in the URL the value **environment\_name=test**. With that mechanism all elements are imported with the correct environment name.

The payload of the import has following structure:

```
{
  "Services": [
    {
      "actions": [""],
      "elementId": "",
      "name": "",
      "description": ""
    }
  ],
  "Actions": [
    {
      "components": [""],
      "elementId": "",
      "name": "",
      "description": ""
    }
  ],
  "Components": [
    {
      "componentType": "",
      "elementId": "",
      "name": "",
      "description": ""
    },
    {
      "componentType": "",
      "elementId": "",
      "name": "",
      "description": ""
    }
  ],
  "Checks": [
    {
      "vstsLink": "",
      "frequency": -1,
      "elementId": "",
      "name": "",
      "description": ""
    }
  ]
}
```

```
# Notification API
```

The Notification API of the Providence Service supports operations for creating, updating and deleting notification rules for specific users and element states.

## API Documentation

{ {HOST} } = https://spp-monitoringservice-X.azurewebsites.net where X stands for the environment the request is made to.

### Create Notification Rule

This endpoint is accessible via: **POST: {{HOST}}/api/notificationRules**

Following payload is used to create a notification rule.

#### Notice:

- The **levels** list shows for which kind of elements you want to be notified (To avoid spam this always notifies you about the highest state of this list)  
So for example if a **Component** goes to state **ERROR** and therefore also an **Action** goes to **ERROR** you receive a notification for the higher element (the Action) only.
- The **states** list show for which kind of states you want to be notified
- the **notificationInterval** shows after which period of time (in seconds) you want to be notified about a state change

```
{
  "environmentSubscriptionId": "XYZ",
  "levels": ["Environment", "Service", "Action", "Component"],
  "emailAddresses": "NotifyMeUser@daimler.com",
  "states": ["Error", "Warning"],
  "notificationInterval": 120,
  "isActive": true
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Created** when the notification rule was created successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the created notification rule or the environment with the specified "environmentSubscriptionId" was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the creation process.

### Update Notification Rule

This endpoint is accesible via: **PUT: {{HOST}}/api/notificationRules/{id}**

Following payload is used to update an existing notification rules (it is the same as when creating a new notification rule)

```
{
  "environmentSubscriptionId": "XYZ",
  "levels": ["Environment", "Service", "Action", "Component"],
  "emailAddresses": "NotifyMeUser@daimler.com",
  "states": ["Error", "Warning"],
  "notificationInterval": 120,
  "isActive": true
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.NoContent** when the notification rule was updated successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the created notification rule or the environment with the specified "environmentSubscriptionId" was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the update process.

### Delete Notification Rule

This endpoint is accessible via: **DELETE: {{HOST}}/api/notificationRules/{id}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Accepted** when the notification rule was deleted successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the notification rule to delete was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the deletion process.

## Get All Notification Rules

This endpoint is accessible via: **GET: {{HOST}}/api/notificationRules**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched notification rules were retrieved successfully.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

## Get Notification Rule by Id

This endpoint is accessible via: **GET: {{HOST}}/api/notificationRules/{id}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched notification rule were retrieved successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the searched notification rule was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

## Notification Logic

After the API is clear this section should explain how the notification rules work.

There are 3 options how a user can configure a notification rule:

- User wants to be notified about **WARNING** and **ERROR** states [1]
- User wants only to be notified about **WARNING** states [2]
- User wants only to be notified about **ERROR** states [3]

Depending on the rule configuration following scenarios may occur:

### NOTICE:

- "**Warning / Error occurred**" notifications are only send if the state of the specified element lasts longer than the **notificationInterval** configured in the rule
- "**Warning / Error resolved**" notifications are only send if a previous Warning / Error notification was send (to avoid spam)
- "**Warning / Error resolved**" notifications are send immediately and do not depend on the **notificationInterval** of a rule

### State Change: OK -> WARNING

- [1] User receives a "**Warning occurred: ...**" notification
- [2] User receives a "**Warning occurred: ...**" notification
- [3] User receives no notification

### State Change: OK -> ERROR

- [1] User receives a "**ERROR occurred: ...**" notification
- [2] User receives no notification
- [3] User receives a "**ERROR occurred: ...**" notification

### State Change: WARNING -> ERROR

- [1] User receives a "**ERROR occurred: ...**" notification
- [2] User receives no notification
- [3] User receives a "**ERROR occurred: ...**" notification

### State Change: WARNING -> ERROR -> WARNING

- [1] User receives a "**ERROR resolved: ...**" notification
- [2] User receives no notification
- [3] User receives a "**ERROR resolved: ...**" notification

### State Change: OK -> ERROR -> WARNING

- [1] User receives a "**ERROR resolved: ...**" notification

- [2] User receives a "**Warning occurred: ...**" notification
- [3] User receives a "**ERROR resolved: ...**" notification

#### **State Change: ERROR -> OK**

- [1] User receives a "**ERROR resolved: ...**" notification
- [2] User receives no notification
- [3] User receives a "**ERROR resolved: ...**" notification

#### **State Change: WARNING-> OK**

- [1] User receives a "**WARNING resolved: ...**" notification
- [2] User receives a "**WARNING resolved: ...**" notification
- [3] User receives no notification

```
# Deployment API
```

The Deployment API of the Providence Service supports operations for creating, updating and deleting deployment windows. Using the "repeatInformation" property of a Deployment you can configure recurring deployments. On this way you for example can create a deployment window for each Friday at 10PM.

## API Documentation

```
{ {HOST} } = https://spp-monitoringservice-X.azurewebsites.net where X stands for the environment the request is made to.
```

### Create Deployment

This endpoint is accessible via: **POST: {{HOST}}/api/deployments**

Following payload is used to create a deployment.

**Notice:**

- A deployment for an environment can only be created if there isn't already an ongoing deployment.
- Also the property startDate is optional. If no value is provided or the provided value points to the future, the current date and time will be set as value.

```
{
    "environmentsubscriptionId": "",
    "description": "",
    "shortDescription": "",
    "startDate": "", // StartDate of the Deployment Window
    "endDate" : "", // EndDate of the Deployment Window
    "closeReason": "",
    "elementIds": ["", ""],
    "repeatInformation": {
        "repeatType": 0, // 0 = Daily, 1 = Weekly, 2 = Monthly
        "repeatInterval": 0, // Repeat every X days
        "repeatOnSameWeekDayCount": false, // true if for example: Every 2nd Monday of a Month
        "repeatUntil": "" // EndDate of the recurring Deployment Window
    }
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Created** when the deployment was created successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the created deployment was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the creation process.

### Update Deployment

This endpoint is accessible via: **PUT: {{HOST}}/api/deployments/{environmentsubscriptionid}/{id}**

Following payload is used to update an existing deployment.

```
{
    "environmentsubscriptionId": "",
    "description": "",
    "shortDescription": "",
    "startDate": "", // StartDate of the Deployment Window
    "endDate" : "", // EndDate of the Deployment Window
    "closeReason": "",
    "elementIds": ["", ""],
    "repeatInformation": {
        "repeatType": 0, // 0 = Daily, 1 = Weekly, 2 = Monthly
        "repeatInterval": 0, // Repeat every X days
        "repeatOnSameWeekDayCount": false, // true if for example: Every 2nd Monday of a Month
        "repeatUntil": "" // EndDate of the recurring Deployment Window
    }
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.NoContent** when the deployment was updated successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the deployment to update was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the update process.

## Delete Deployment

This endpoint is accessible via: **DELETE: {{HOST}}/api/deployments/{environmentSubscriptionId}/{id}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Accepted** when the deployment was deleted successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the deployment to delete was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the deletion process.

## Get All Deployments

This endpoint is accessible via: **GET: {{HOST}}/api/deployments**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched deployment was retrieved successfully.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

## Get Deployment History for an Environment

This endpoint is accesible via: **GET: {{HOST}}/api/deployments/{environmentName}?startDate={startDate}&endDate={endDate}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched deployment were retrieved successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the searched deployment were not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

```
# Alert Ignore API The Alert Ignore API of the Providence Service supports operations for creating, updating and deleting "Alert Ignore Rules".
```

Alert ignore rules can be used to mute specific alerts. For example all alerts with a specific CheckId can be muted to avoid StateTransitions resulting from their occurrence in the Providence Service.

## API Documentation

{ {HOST} } = https://spp-monitoringservice-X.azurewebsites.net where X stands for the environment the request is made to.

### Create Alert Ignore Rule

This endpoint is accessible via: **POST: {{HOST}}/api/alertignores**

Following payload is used to create an alert ignore rule.

**Notice:** The value of **environmentSubscriptionId** is mandatory and if **subscriptionId** is set in the **ignoreCondition** property those two values have to be identical.

```
{
    "name": "",
    "environmentSubscriptionId": "",
    "creationDate": "",
    "expirationDate": "",
    "ignoreCondition": {
        "checkId" : "",
        "componentId" : "",
        "subscriptionId" : "",
        "description" : "",
        "state" : "",
        "customField1": "",
        "customField2": "",
        "customField3": "",
        "customField4": "",
        "customField5": "",
        "alertName": ""
    }
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Created** when the alert ignore was created successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the created alert ignore was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the creation process.

### Update Alert Ignore Rule

This endpoint is accessible via: **PUT: {{HOST}}/api/alertignores/{id}**

Following payload is used to create an alert ignore rule.

```
{
  "name": "",
  "environmentSubscriptionId": "",
  "creationDate": "",
  "expirationDate": "",
  "ignoreCondition": {
    "checkId" : "",
    "componentId" : "",
    "description" : "",
    "state": "",
    "customField1": "",
    "customField2": "",
    "customField3": "",
    "customField4": "",
    "customField5": "",
    "alertName": ""
  }
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.NoContent** when the alert ignore was updated successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the alert ignore to update was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the update process.

### Delete Alert Ignore Rule

This endpoint is accessible via: **DELETE: {{HOST}}/api/alertignores/{id}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Accepted** when the alert ignore was deleted successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the alert ignore to delete was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the deletion process.

### Get Alert Ignore Rule

This endpoint is accessible via: **GET: {{HOST}}/api/alertignores/{id}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched alert ignore was retrieved successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the searched alert ignore was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

### Get Alert Ignore Rules

This endpoint is accessible via: **GET: {{HOST}}/api/alertignores**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched alert ignore were retrieved successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the searched alert ignores were not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

```
# State Increase Rule API The State Increase Rule API of the Providence Service supports operations for creating, updating and deleting specific rules.
```

State increase rules can be used to increase the state of specific alerts after a specified period of time.

- For example a "warning" can be configured to be processed as an "error" within the Providence Service if the state "warning" does not change after X minutes.

## API Documentation

```
{ {HOST} } = https://spp-monitoringservice-X.azurewebsites.net where X stands for the environment the request is made to.
```

### Create State Increase Rule

This endpoint is accessible via: **POST: {{HOST}}/api/stateIncreaseRules**

Following payload is used to create a state increase rule.

**Notice:** Only the value of **alertName** is optional. All other values are **mandatory**

```
{
    "name": "",
    "description": "",
    "environmentSubscriptionId": "",
    "checkId": "",
    "alertName": "",
    "componentId": "",
    "triggerTime": int,
    "isActive": bool
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Created** when the State Increase Rule was created successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the specified environment was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the creation process.

### Update State Increase Rule

This endpoint is accessible via: **PUT: {{HOST}}/api/stateIncreaseRules/{id}**

Following payload is used to update a state increase rule.

```
{
    "name": "",
    "description": "",
    "environmentSubscriptionId": "",
    "checkId": "",
    "alertName": "",
    "componentId": "",
    "triggerTime": int,
    "isActive": bool
}
```

The response of this endpoint call can be one of the following:

- **HttpStatusCode.NoContent** when the State Increase Rule was updated successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the State Increase Rule to update was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the update process.

### Delete State Increase Rule

This endpoint is accessible via: **DELETE: {{HOST}}/api/stateIncreaseRules/{id}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Accepted** when the State Increase Rule was deleted successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.

- **HttpStatusCode.NotFound** when the State Increase Rule to delete was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the deletion process.

### Get State Increase Rule

This endpoint is accessible via: **GET: {{HOST}}/api/stateIncreaseRules/{id}**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched State Increase Rule was retrieved successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the searched State Increase Rule was not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

### Get State Increase Rules

This endpoint is accessible via: **GET: {{HOST}}/api/stateIncreaseRules**

The response of this endpoint call can be one of the following:

- **HttpStatusCode.Ok** when the searched State Increase Rule were retrieved successfully.
- **HttpStatusCode.BadRequest** when the payload/url contains invalid parameter.
- **HttpStatusCode.NotFound** when the searched State Increase Rules were not found in the database.
- **HttpStatusCode.InternalServerError** when something went wrong during the get process.

## Postman Collection

A full Postman Collection for calling the Providence endpoint can be downloaded on this page.

Postman Collection:

[Providence\\_Service.postman\\_collection.zip](#) ([./src/help/assets/help/assets/Providence\\_Service.postman\\_collection-04d5ca84-b9fe-4afe-92d7-a0c0bd31fa2b.zip](#))

## Swagger UI

The Swagger documentation of the Providence Service can be found under following url:

- <https://serviceurl/swagger>

