COOLPROFS

COOL MONITOR

Quick Reference

CoolProfs B.V. ▪ +31 (0)71 33 11 623 ▪ Schoolbaan 1 ▪ 2371 VJ Roelofarendsveen
www.coolprofs.com ▪ info@coolprofs.com ▪ KvK nr. 28097699

Table of Contents

Release notes

Version	Release notes
V2.02 (Elastic 8.3.1)	• added new dashboard "Users" which shows amount of users over time for Reactive and Tradional Screens • changed time field to instant to improve accuracy of visualizations • changed layout of "Failed Extension" and "Failed Timer" dashboard to align with other dashboards • translated fields from Dutch to English • improved naming of visualizations • added "OutSystems 11" to names of OutSystems dashboard differenatiate them • aligned number of digits after decimal point throughout the dashboards

Preface

Cool Monitor consists of a number of components, such as Discover, the set of standard dashboards, and, if desired, Application Performance Monitoring (APM) and Real User Monitoring (RUM).

This document describes two of the components, Discover and the standard dashboards, and explains their interface. Each explanation includes a screenshot with blue numbers for the components of the screenshot. These numbers are explained in the explanation below. The other components have separate documentation.

First, Discover is discussed, which is the way to navigate through the raw logs. Second, the standard dashboards are discussed, such as Slow SQL, Users, and Errors.

Many of the dashboards are based on log categories as they come from OutSystems—Integration, Extension, and Error. Others are subsets of logs—for example, SlowSQL is a subset of the General log.

Discover

"Discover" is the interface to navigate all the logs:

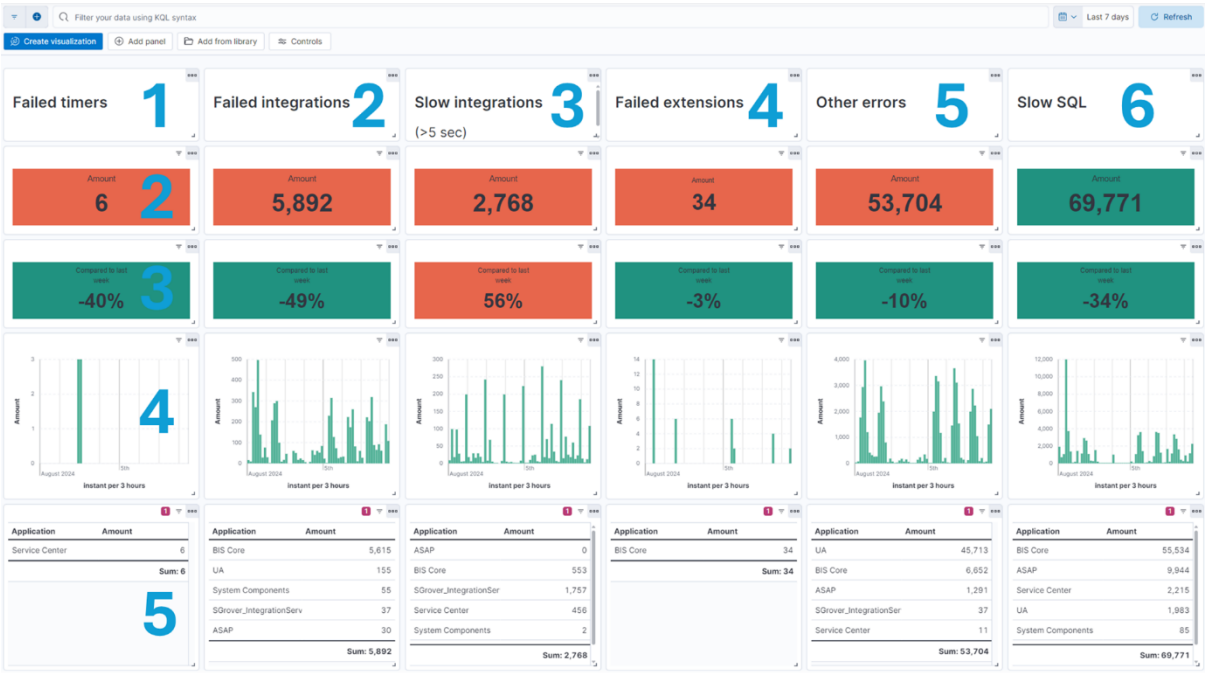
1. List of available fields.
2. Selecting a time range.
3. Entering a specific query for filtering (language = KQL).
4. Adding filters.
5. Selecting an index.
6. Bar chart showing the amount of logs through time.
7. Raw logs (the arrow opens the entire log).

Cool Monitor uses a standard index named "OutSystems", this is the place where the Outsystems logs go.

Dashboard overview

Dashboard	Inhoud
Application Overview	Overview of all applications with number of failed timers, failed integrations, slow integrations, failed extensions, errors, and Slow SQL notifications.
Timer	Detailed information about timers – number of runs, number of failed runs, and average duration.
Failed Timer	Detailed information about failed timers.
Extension	Detailed information about extensions – number of hits, number of failed hits, and average duration.
Failed Extension	Detailed information about failed extensions.
Integration	Detailed information about integrations (APIs) – number of calls, number of failed calls, and average duration.
Failed Integration	Detailed information about failed integrations.
Error	All error messages.
General	All general messages.
Web Request	Detailed information about actions on traditional sites – number of actions and average duration.
Mobile Request	Detailed information about actions on reactive sites – number of actions and average duration.
Slow SQL	Detailed information about SlowSQL queries – number of hits and average duration.
Users	Number of unique users of traditional and reactive sites.

Application Overview



Objective: Global overview of the state of the selected environment based on a number of main categories.

About the selected time period:

Column 1: No. of failed timers (timer fails when the log has an error_id).

Column 2: No. of failed integrations (integration fails when the log has an error_id)

Column 3: No. of slow integrations (slow is >5 sec).

Column 4: No. of failed extensions (extension fails when the log has an error_id).

Column 5: No. of other errors.

Column 6: No. of Slow SQL logs (standard logging when SQL call >200 ms).

Row 1: Category name.

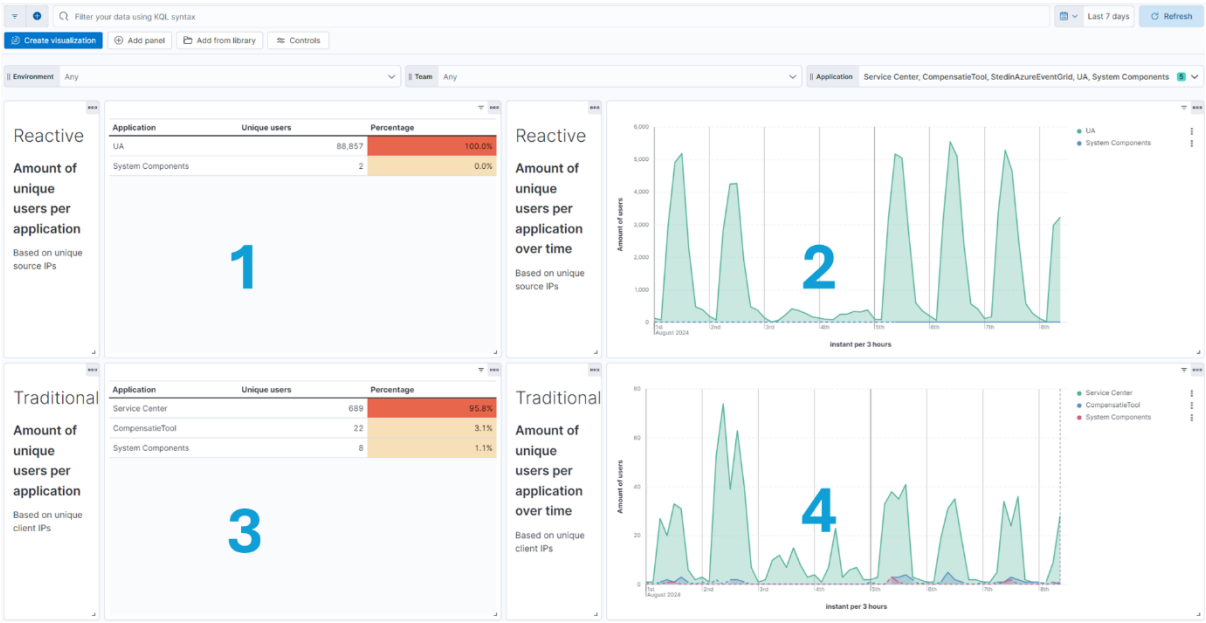
Row 2: No. of logs.

Row 3: No. of logs compared to last week in %. Positive is more, negative is fewer.

Row 4: No. of logs though time in a bar graph.

Row 5: No. of logs per application.

Users



Objective: Gaining insight into the number of users on the system.

This is divided into Reactive and Traditional applications. This data is based on the number of unique host/client IPs.

About the selected time period:

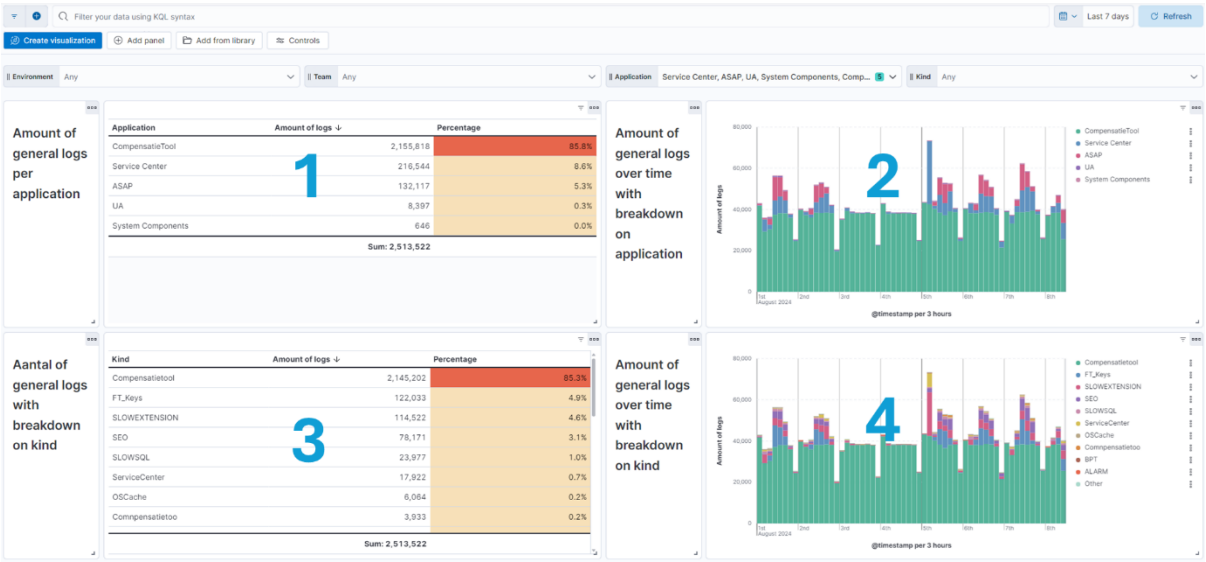
Balloon 1: No. of unique users per Reactive Application.

Balloon 2: No. of unique users per Reactive Application over time.

Balloon 3: No. of unique users per Traditional Application.

Balloon 4: No. of unique users per Traditional Application over time.

General



Objective: To gain insight into the quantity and type of general logs per application.

About the selected time period:

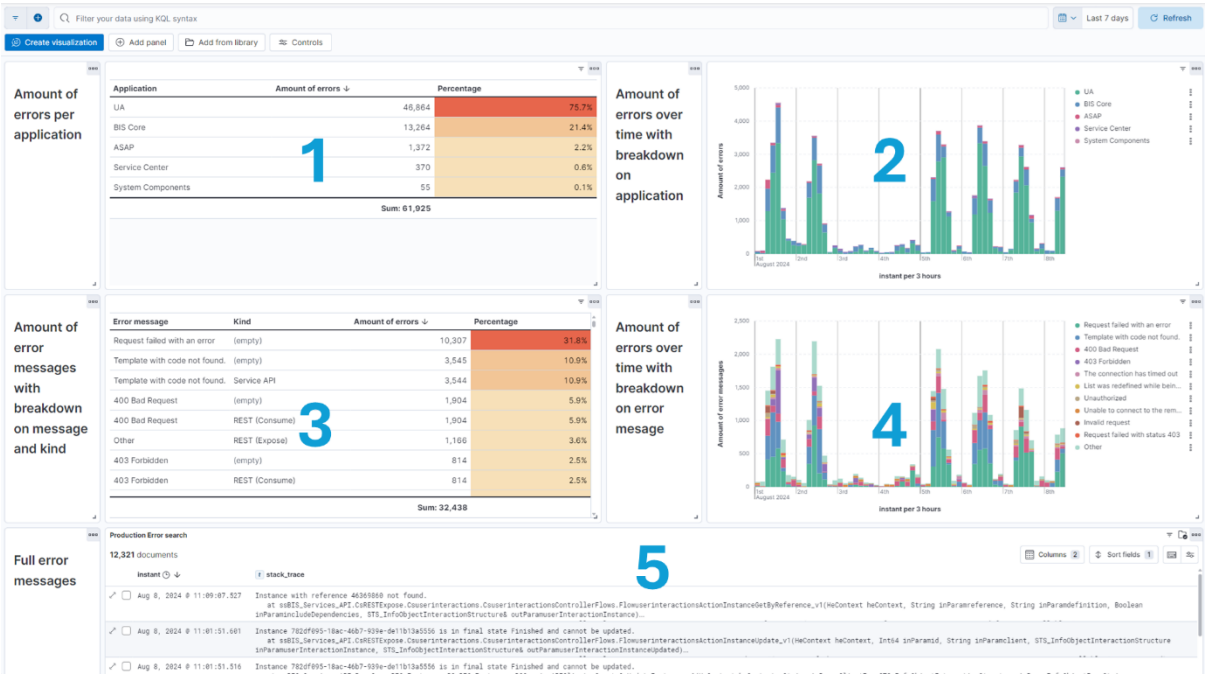
Balloon 1: No. of General logs per application and its share in the whole.

Balloon 2: Bar chart showing the number of general logs over time per application.

Balloon 3: Number of general logs per application per type (child) and what proportion of the total this represents.

Balloon 4: Bar chart showing the number of general logs over time per type (child).

Error



Objective: To gain insight into the quantity and type of error logs per application.

About the selected time period:

Balloon 1: No. of error logs per application and its share in the whole.

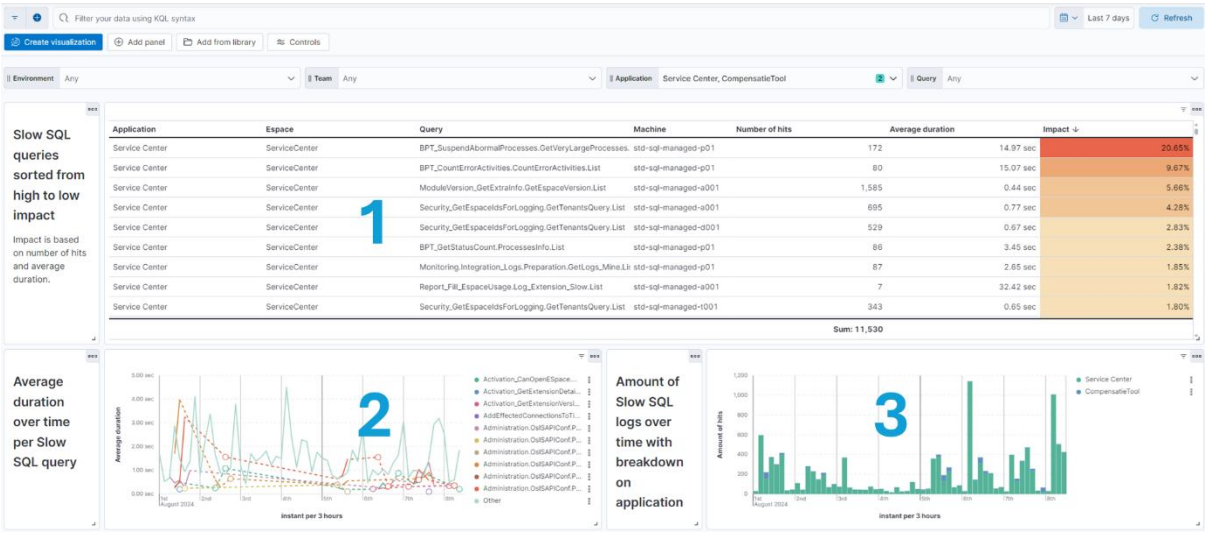
Balloon 2: Bar chart showing the number of error logs over time per application. This can indicate whether the number has increased or decreased after a deployment or update..

Balloon 3: Number of error logs per message and per type (child) and what proportion of the total this represents.

Balloon 4: Bar chart showing the number of error logs over time per error message. This can indicate that a particular error message has been added after a deployment or update..

Balloon 5: The complete stack trace of the error messages.

Slow SQL



Objective: To gain insight into the slowest SQL over time and its impact. OutSystems only logs SQL calls above a certain threshold (default 200 ms), which can be configured.

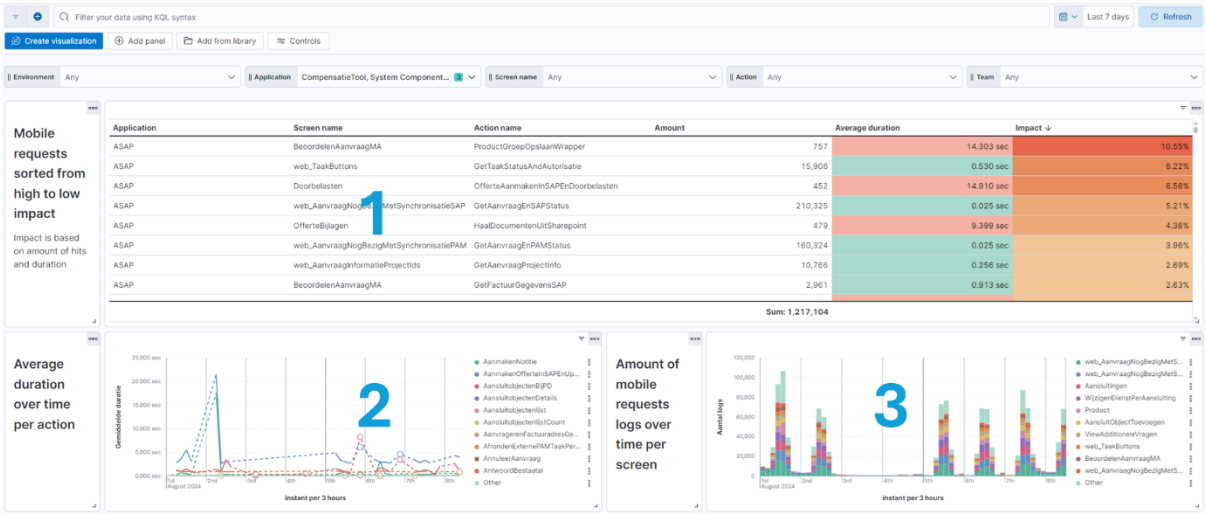
About the selected time period:

Balloon 1: Depending on your preferences, you can sort by the number of times the SQL occurs, the average duration, or the impact. Impact is based on the number of calls and the duration compared to the other calls. Here you can quickly see whether you need to analyze an SQL more deeply or not.

Balloon 2: The graph shows whether an SQL call is stable over time, or whether it is becoming slower or faster.

Balloon 3: Graph showing when the SQL calls take place. This can provide clues as to whether it is after a deployment, a timer, or a special case.

Mobile Request



Objective: Gain insight into the actions of Reactive screens over time and their impact.

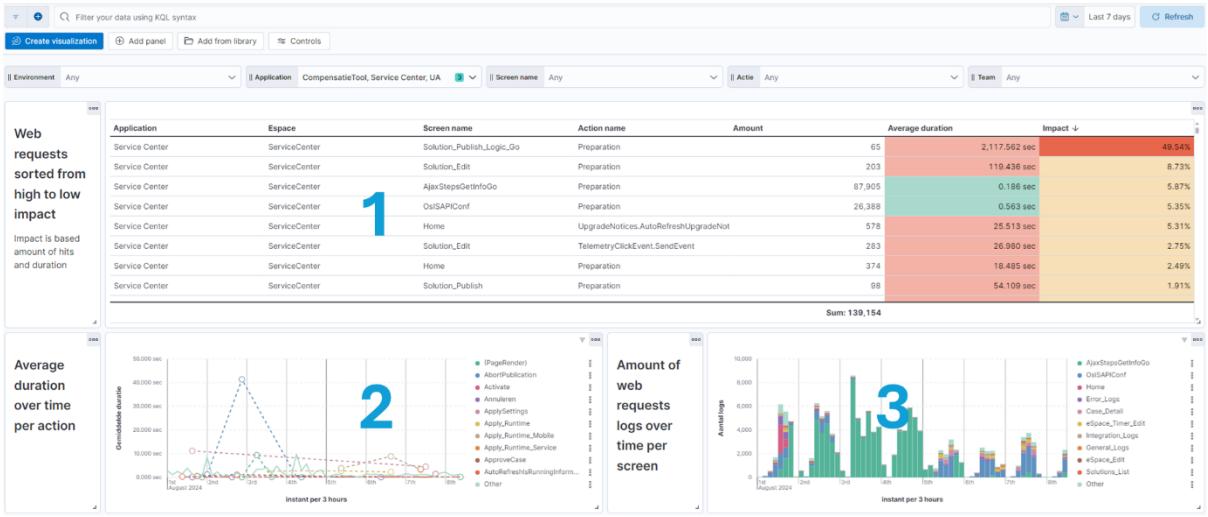
About the selected time period:

Balloon 1: Depending on your preferences, you can sort by the number of times an action occurs, the average duration, or the impact. Impact is based on the number of calls and the duration compared to the other calls. Here you can quickly see whether you need to analyze an action in more detail or not.

Balloon 2: In graph form, you can see whether a Reactive action is stable over time, or perhaps becoming slower or faster..

Balloon 3: Graph showing when the actions take place. This can provide clues as to whether the action has become faster or slower after a deployment or update.

Web Request



Objective: To gain insight into the actions of Traditional screens over time and their impact.

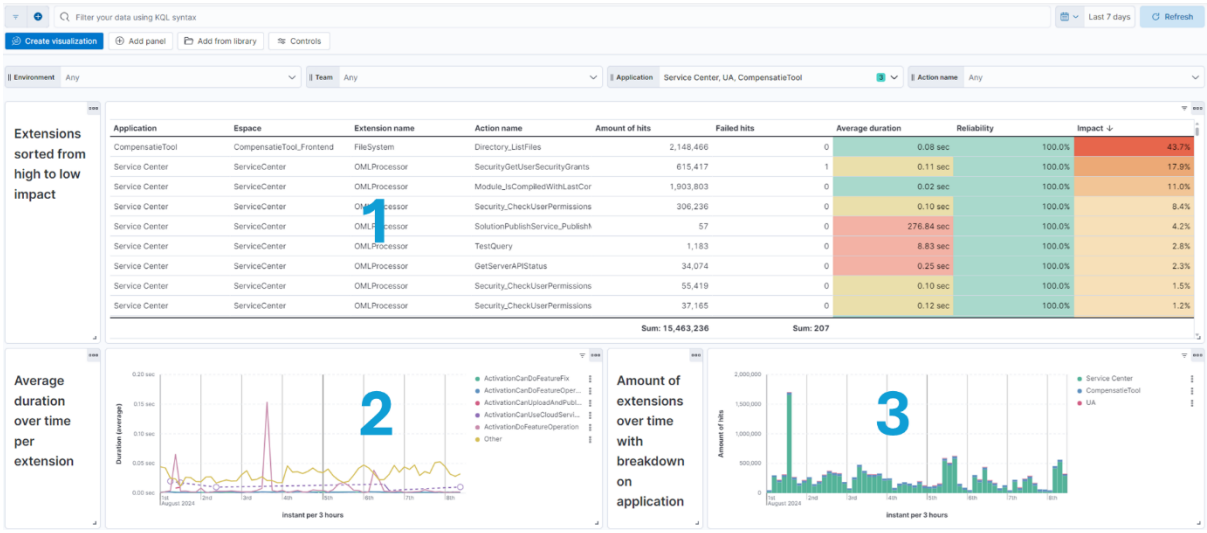
About the selected time period:

Balloon 1: Depending on your preferences, you can sort by the number of times a Traditional action occurs, the average duration, or the impact. Impact is based on the number of calls and the duration compared to the other calls. Here you can quickly see whether you need to analyze an action in more detail or not.

Balloon 2: In graph form, you can see whether a traditional action is stable over time, or perhaps becoming slower or faster.

Balloon 3: Graph showing when the actions take place. This can provide clues as to whether the action has become faster or slower after a deployment or update.

Extension



Objective: To gain insight into the duration and reliability of Extensions over time and their impact.

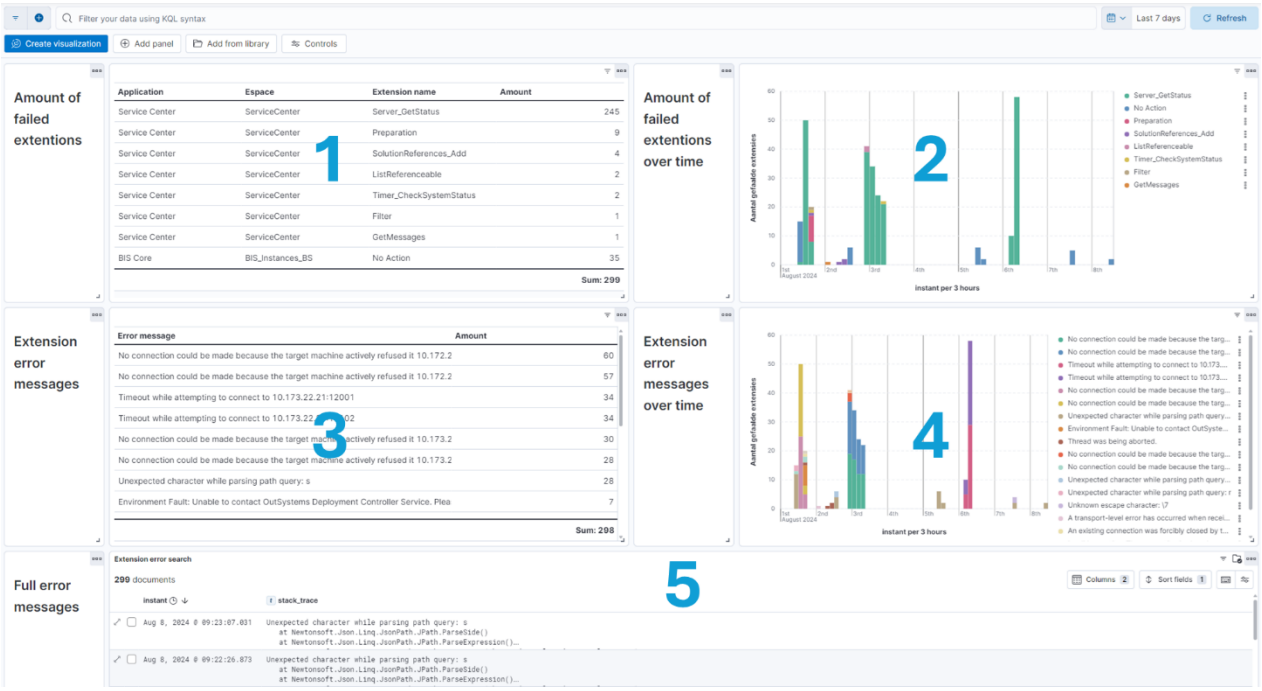
About the selected time period:

Balloon 1: Depending on your preferences, you can sort by the number of times the Extension is called, the average duration, how often the Extension fails, reliability, or impact. Impact is based on the number of calls and the duration compared to other calls. Reliability is based on the number of Extension logs with and without a related error_id. Here you can quickly see whether you need to analyze an Extension in more detail or not.

Balloon 2: The graph shows whether an extension is stable in terms of duration over time, or whether it is becoming slower or faster.

Balloon 3: Graph showing when extension calls take place. This can provide clues as to whether something has changed after a deployment, for example.

Failed Extension



Objective: To provide insight into the failed Extension and obtain detailed information about it in order to potentially take action.

About the selected time period:

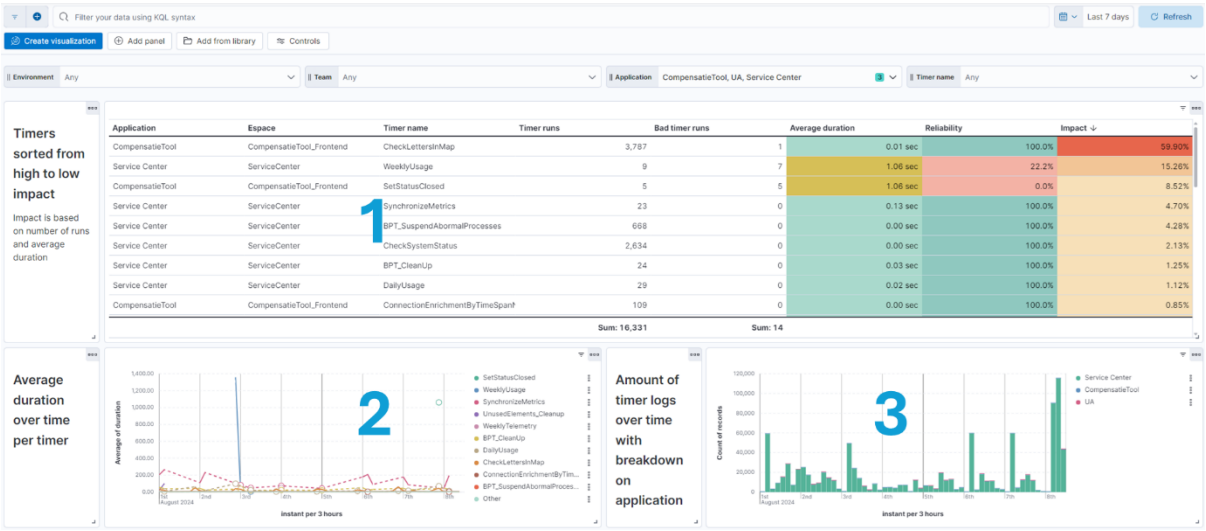
Balloon 1: Number of failed extensions, including information about which application and espace they come from.

Balloon 2: Number of failed extensions over time.

Balloon 3: The error message why the Extension has failed.

Balloon 4: The error message why the Extension has failed over time.

Timer



Objective: To gain insight into the duration and reliability of timers over time and their impact.

About the selected time period:

Balloon 1: Depending on your preferences, you can sort by the number of times the Timer is called, the average duration, how often the Timer fails, the reliability, or the impact. Impact is based on the number of calls and the duration compared to the other calls. Reliability is based on the number of Timer logs with and without a related error_id. Here you can quickly see whether you need to analyze a Timer in more detail or not.

Balloon 2: The graph shows whether a Timer is stable in terms of duration over time.

Balloon 3: Graph showing when the timers occur. This can provide clues as to whether it is after a deployment, a timer, or a special case.

Failed Timer

Objective: To provide insight into the failed Timers and obtain detailed information about them in order to potentially take action.

About the selected time period:

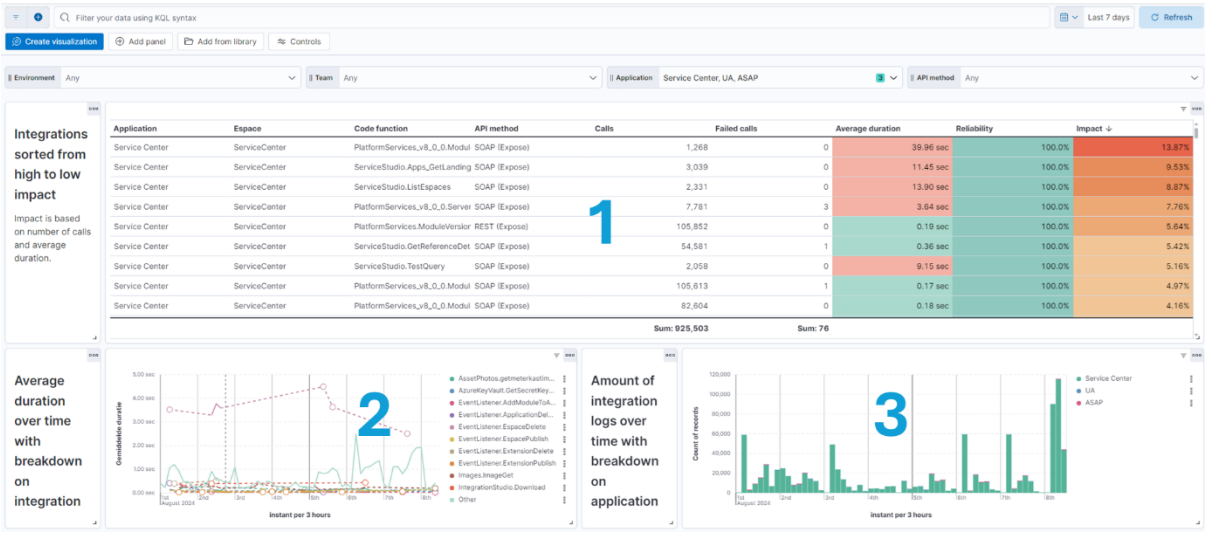
Balloon 1: No. of failed timers, including information about which application and espace they come from.

Balloon 2: No. of failed timers over time.

Balloon 3: The error messages why the timers have failed.

Balloon 4: The error message why the timers have failed over time.

Integration



Objective: To gain insight into the duration and reliability of integrations over time and their impact.

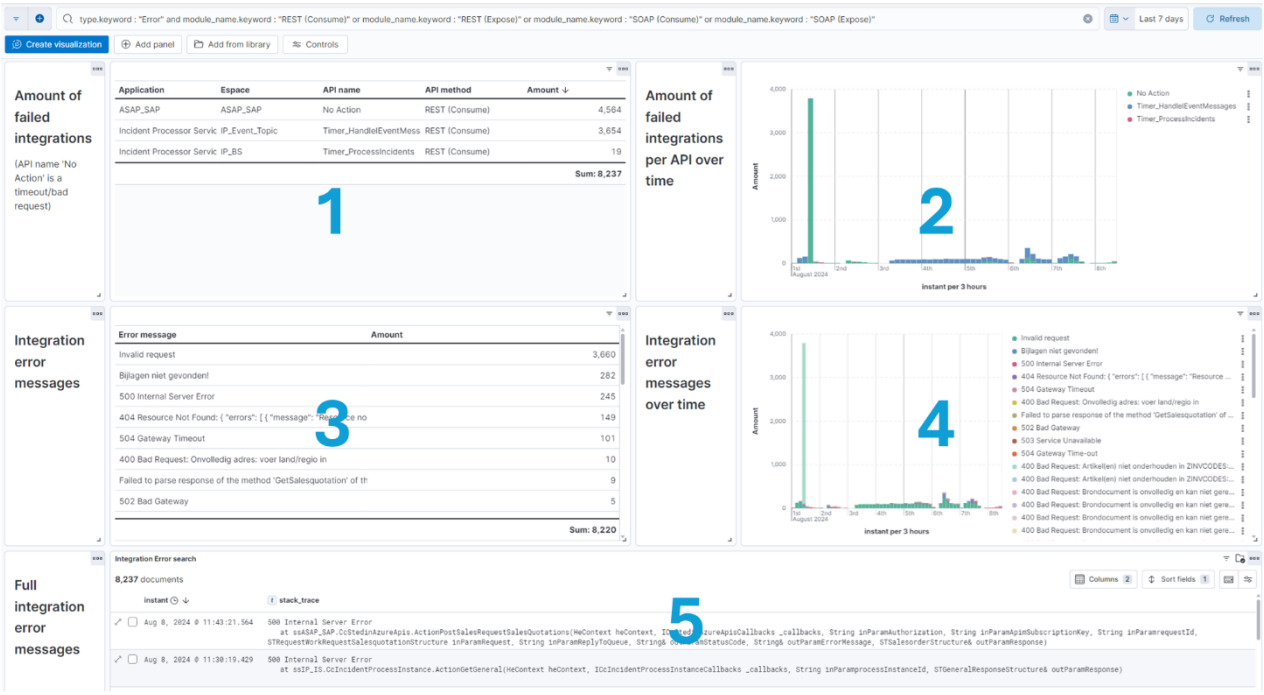
About the selected time period:

Balloon 1: Depending on your preferences, you can sort by the number of times the Integration is called, the average duration, how often the Integration fails, the reliability, or the impact. Impact is based on the number of calls and the duration compared to other calls. Reliability is based on the number of Integration logs with and without a related error_id. Here you can quickly see whether you need to analyze an Integration in more detail or not.

Balloon 2: The graph shows whether an integration is stable in terms of duration over time.

Balloon 3: Graph showing when the Integrations take place. This can provide clues as to whether it is after a deployment, an Integration, or a special case.

Failed Integration



Objective: To provide insight into failed integrations and obtain detailed information about them in order to potentially take action.

About the selected time period:

Balloon 1: No. of failed integrations, including information about which application and espace they come from.

Balloon 2: No. of failed integrations over time.

Balloon 3: The error messages why the integrations have failed.

Balloon 4: The error message why the integrations have failed over time.