



KissLog

On-premises

Installation and Configuration Guide

Table of contents

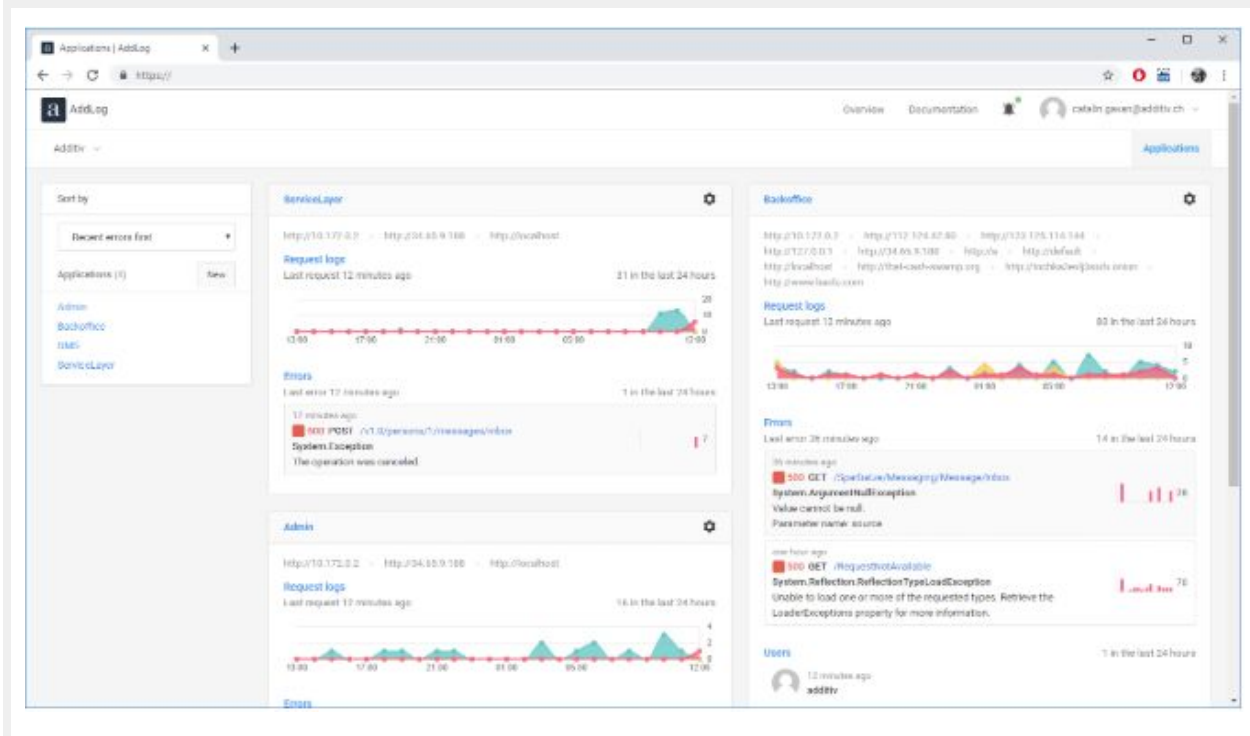
Table of contents	2
About	3
Links	3
Prerequisites	4
Artifacts	4
Services	4
Architecture	5
KissLog.Frontend	5
KissLog.Backend	5
Storage providers	5
MS-SQL / MySQL	5
MongoDB	5
Installation	7
MongoDB	7
MS-SQL / MySQL	7
IIS web applications	7
KissLog.Backend	11
Startup	11
Troubleshooting	13
Log files	13
/api/status	13
IIS logs	14
KissLog.Frontend	15
Startup	15
Troubleshooting	16
Log files	16
/api/status	16
IIS logs	17

About

This guide will walk through installation and configuration steps for deploying KissLog on-premises.

If you have any issues or questions, please feel free to send me an email to catalingavan@gmail.com and I will try to answer them as soon as possible.

KissLog application screen



Links

KissLog.net
<https://kisslog.net/>

KissLog SDK
<https://github.com/KissLog-net/KissLog.Sdk>

Prerequisites

Artifacts

1. KissLog.Backend.zip
2. KissLog.Frontend.zip

Services

KissLog on-premises requires the following services:

1. [MongoDB Community Server](#) (free)
2. [MS-SQL Server](#) or [MySQL Community Server](#)
3. IIS web server with [.NET Core Runtime](#)

Architecture

KissLog on-premises consists of two .NET Core 2.0 web applications:

KissLog.Frontend

User-interface application used by users (developers, IT administrators, application managers and implementation consultants) to visualise the captured errors, logs and other metrics data.

KissLog.Backend

Backend application responsible for managing the logs data. Exposes REST endpoints which can be used to save and to query the logs.

Storage providers

KissLog uses two storage providers:

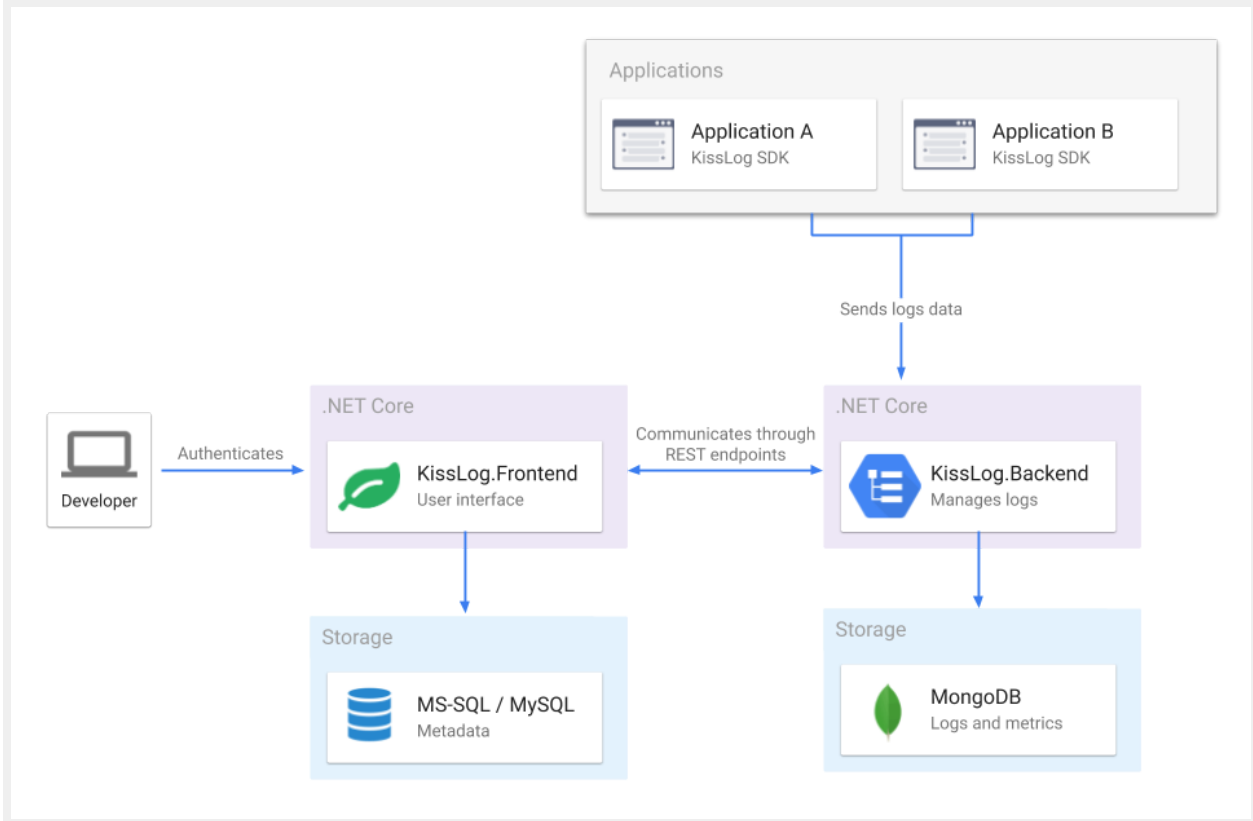
1. MS-SQL / MySQL

Used for storing metadata information in the database, including information about organizations, applications and other user-interface related properties.

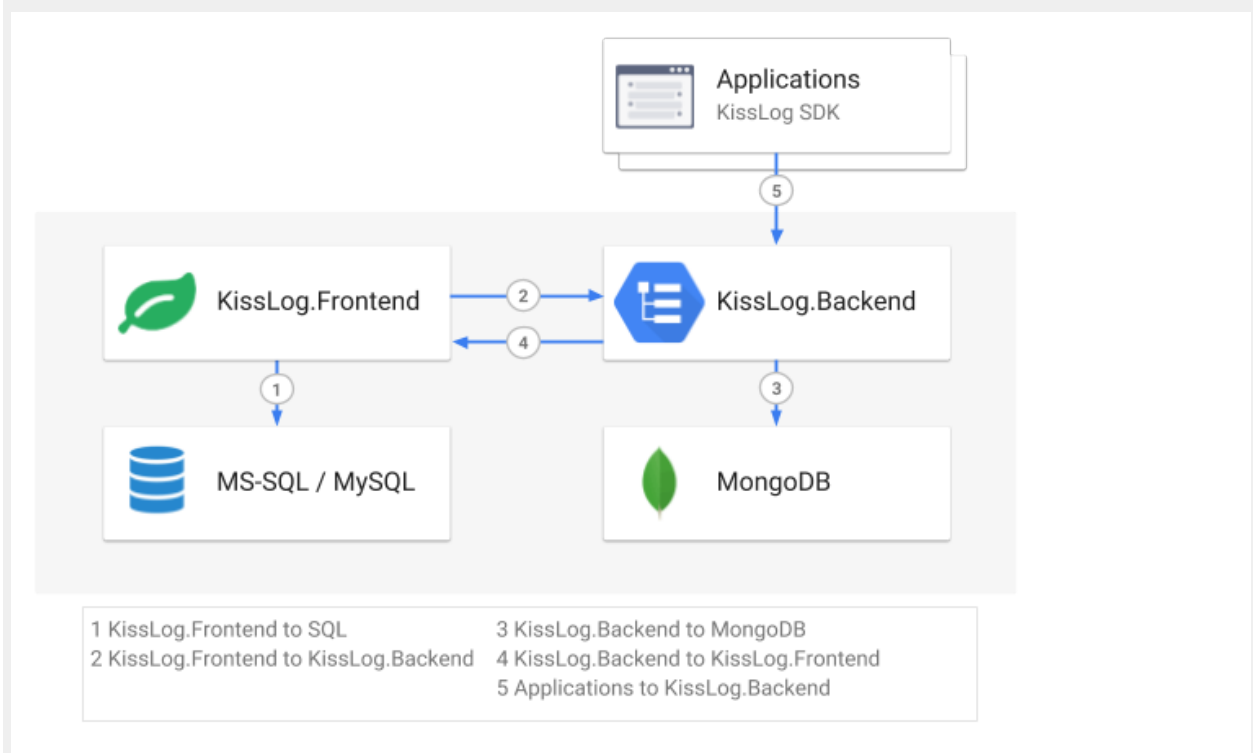
2. MongoDB

Used for storing logs, captured exceptions and other metrics data.

KissLog architecture



Network access



Installation

MongoDB

For installing MongoDB server, please check the [official tutorial](#).

MS-SQL / MySQL

We will not cover the installation guide for these services. There is a high possibility that the existing system will already have a running instance of MySQL or MS-SQL server.

IIS web applications

1. On the IIS server, install [.NET Core Runtime](#)
2. Create two IIS applications, one KissLog.Backend and the second KissLog.Frontend

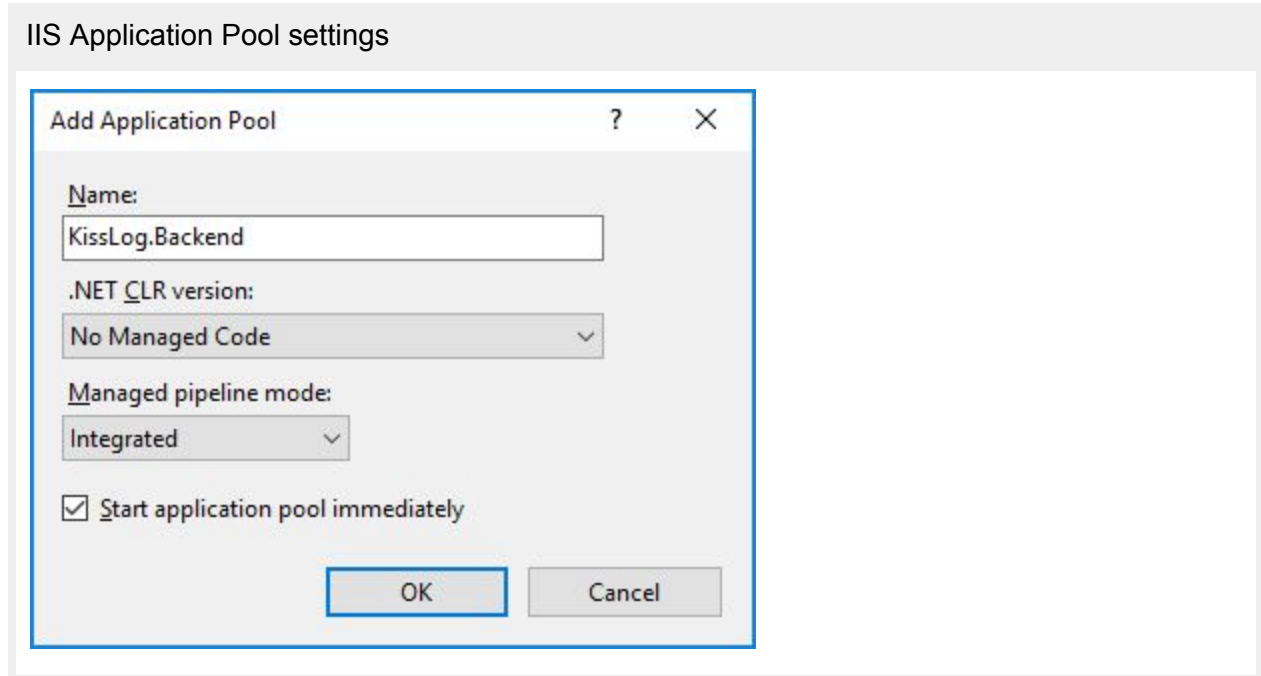
IIS write permissions

Make sure IIS has write permissions to the application folder.

3. Update the Application Pool settings for both of the applications to the following:

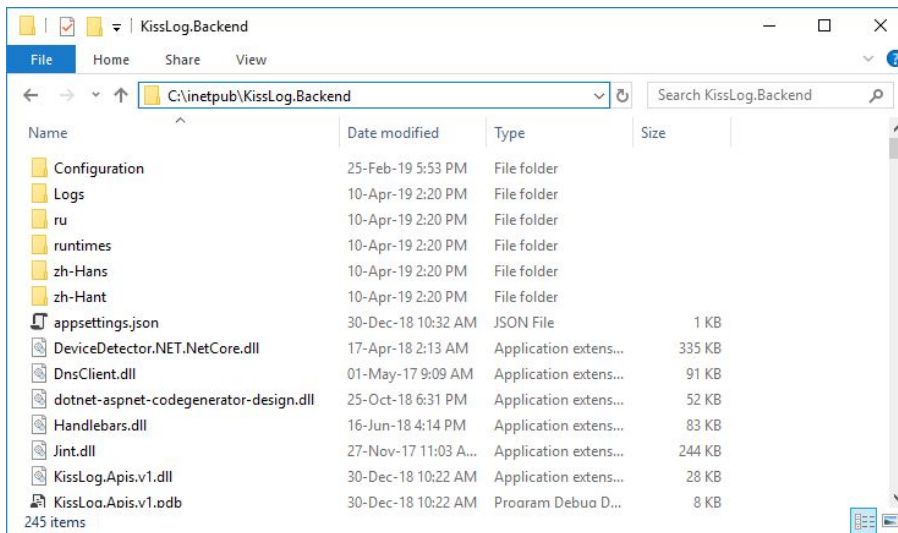
.NET CLR version: No Managed Code

Managed pipeline mode: Integrated

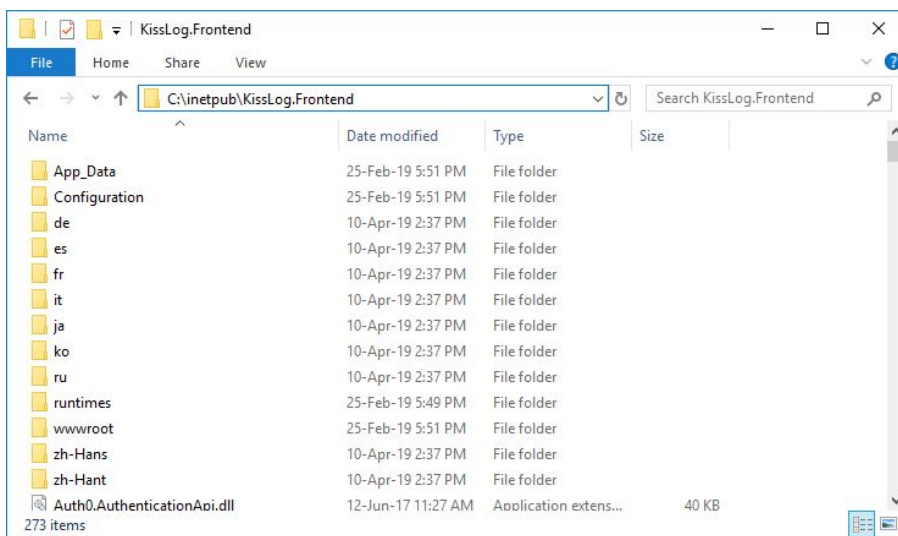


- Copy into each Site folder the corresponding deploy packages.

KissLog.Backend IIS folder



KissLog.Frontend IIS folder



-
5. Update the KissLog.json file for both applications:

[KissLog.Backend\Configuration\KissLog.json](#)

[KissLog.Frontend\Configuration\KissLog.json](#)

Configuration\KissLog.json backup

If you are updating an existing version of KissLog, make a backup of the KissLog.json file and replace it after the new files have been copied.

KissLog.Backend

Startup

After Configuration\KissLog.json file has been updated, KissLog.Backend application should be ready to use.

To initialize the startup process, make a single request to the application root URL ("/").

If the startup process went successful, a **200 OK "Running"** response will be returned.

This will bootstrap all the necessary components, including configuration validation and MongoDB database generation.

A **Logs** folder will be created in the application physical path.

IIS write permissions

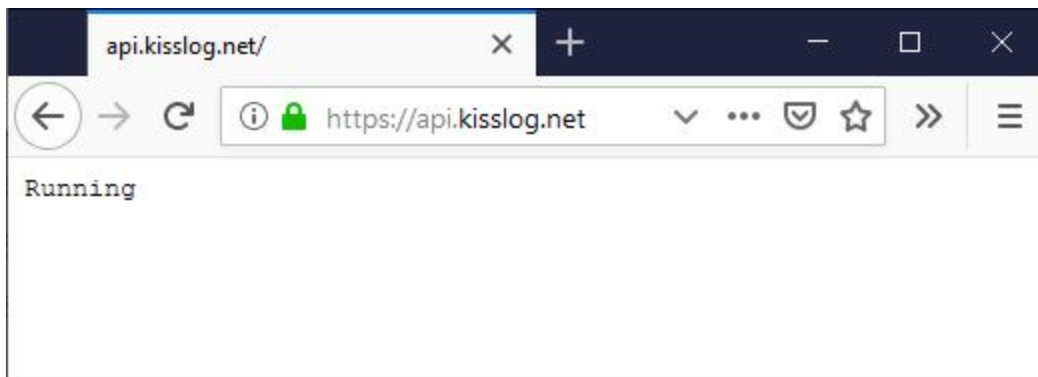
On the first run, the Logs folder will be automatically created, regardless if the startup process went successful or not.

If this folder has not been created, make sure IIS has write permissions to the application folder.

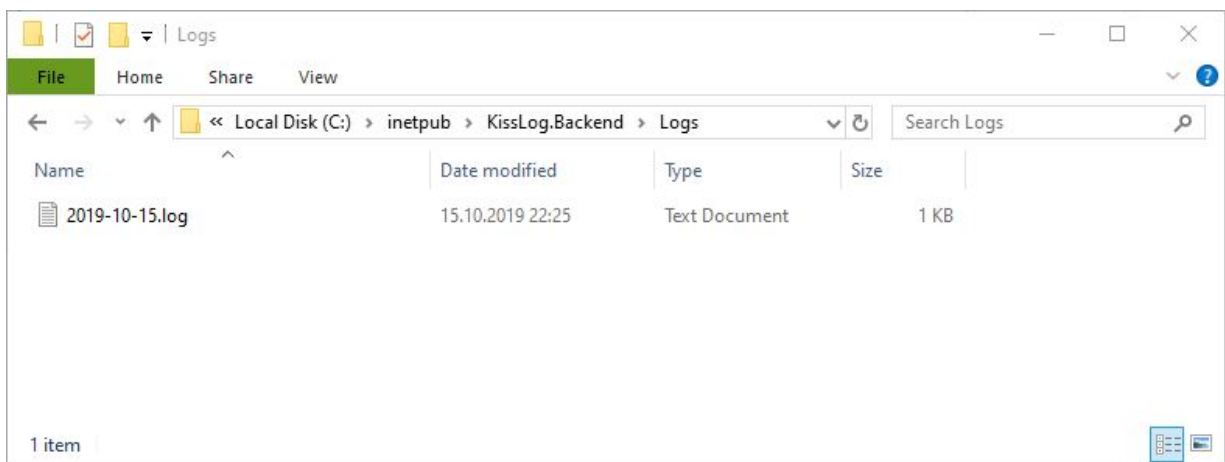
Updating KissLog.json

After a further update of the KissLog.json file, the IIS application needs to be restarted.

KissLog.Backend 200 OK Running



KissLog.Backend Logs



Troubleshooting

Log files

All the application errors are logged under the Logs folder.

In case of any problems, this should be the first place to check, as they will give a hint about the current issue.

/api/status

In addition to text logs, KissLog.Backend exposes GET **/api/status** endpoint, which provides useful information about the current status of the application.

GET /api/status

```
{
  "Success": true,
  "Steps": [
    {
      "Step": "CheckConfiguration",
      "Messages": [],
      "Success": true
    },
    {
      "Step": "CheckMongoDb",
      "Messages": [
        "DatabaseName: KissLogBackend",
        "MongoDb collections: 24"
      ],
      "Success": true
    }
  ]
}
```

IIS logs

If the application fails to start, and no logs messages can be found, we can activate and check the IIS logs.

To activate the IIS logs, edit the `web.config` file, and update `stdoutLogEnabled="true"`.

web.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <!-- code removed for simplicity -->
      <aspNetCore processPath="dotnet"
arguments=".\KissLog.Backend.AspNetCore.dll" stdoutLogEnabled="true"
stdoutLogFile=".\logs\stdout" />
    </system.webServer>
  </location>
</configuration>
```

After updating the `web.config`, restart the IIS application and initialize the startup process again.

The IIS logs will be added to `logs\stdout` folder and will provide more information about the current issue.

KissLog.Frontend

Startup

After Configuration\KissLog.json file has been updated, KissLog.Frontend application should be ready to use.

To initialize the startup process, make a single request to the application root URL ("/").

If the startup process went successful, a **200 OK** home page will be returned.

This will bootstrap all the necessary components.

A **Logs** folder will be created in the application physical path.

IIS write permissions

On the first run, the Logs folder will be automatically created, regardless if the startup process went successful or not.

If this folder has not been created, make sure IIS has write permissions to the application folder.

Updating KissLog.json

After a further update of the KissLog.json file, the IIS application needs to be restarted.

Troubleshooting

Log files

All the application errors are logged under the Logs folder.

In case of any problems, this should be the first place to check, as they will give a hint about the current issue.

/api/status

In addition to text logs, KissLog.Frontend exposes GET **/api/status** endpoint, which provides useful information about the current status of the application.

GET /api/status

```
{
  "Success":true,
  "Steps":[
    {
      "Step":"CheckConfiguration",
      "Messages":[ ],
      "Success":true
    },
    {
      "Step":"CheckDatabase",
      "Messages":[
        "KissLogDbContext created",
        "KissLogDbContext.Database.EnsureCreated() complete. New
database created: False"
      ],
      "Success":true
    },
    {
      "Step":"CheckKissLogBackend",
      "Messages":[
        "GET http://localhost:53335/api/status",
        "StatusCode: OK"
      ],
      "Success":true
    }
  ]
}
```

IIS logs

If the application fails to start, and no logs messages can be found, we can activate and check the IIS logs.

To activate the IIS logs, edit the `web.config` file, and update `stdoutLogEnabled="true"`.

web.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <!-- code removed for simplicity -->
      <aspNetCore processPath="dotnet"
arguments="..\KissLog.Frontend.AspNetCore.dll" stdoutLogEnabled="true"
stdoutLogFile="..\logs\stdout" />
    </system.webServer>
  </location>
</configuration>
```

After updating the `web.config`, restart the IIS application and initialize the startup process again.

The IIS logs will be added to `logs\stdout` folder and will provide more information about the current issue.