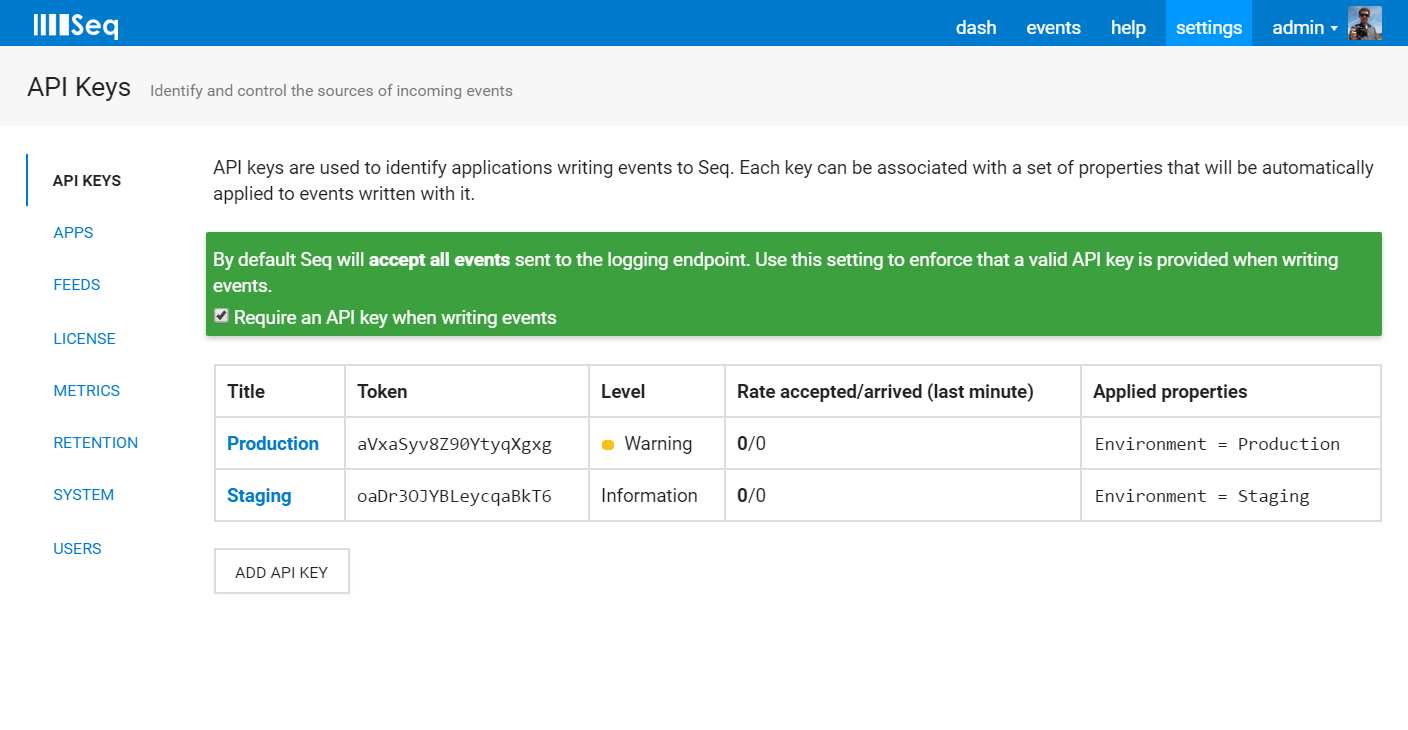
March/April 2017

**SeriLog and Seq**

Findings:

Seq is most useful and designed for **faster** access to **recent** data. Can use SeriLog and Seq on .NET core projects to get good index search and debugging capabilities such as

1. API Keys – Tag events, view incoming event rate, filter events on arrival, Notify client application of its desired logging level to reduce network traffic.
2. Trace an event end to end using by searching for its correlation Id
3. Filter for ‘Debug/Warning level’ logs to find out and pin point problems
4. The ‘Context’ and ‘Enrich’ functions of SeriLog help identify individual method scopes and projects (microservice) by and name making debugging easy. More

- Ambient logging contexts

-Contextual Logging

-Globally-attached properties (Enrich)

-Programmatic enrichment

{ https://nblumhardt.com/2016/08/context-and-correlation-structured-logging-concepts-in-net-5/}

1. The object de-structuring (@ sigil) feature is useful when we need a full class/object image in logs e.g. Inserting /Updating/Deleting a record, with the potential utility of ‘recovering’ such after the fact. On LoggerConfiguration, you can use e.g. Destructure.ByTransforming<post>(p => new { p.Name, p.Id }) to do this type-by-type. Alternatively get Destructurama.Attributed from Nuget.
2. Seq – Azure Integration:
   1. Is Azure Active Directory (AAD) supported? Yes, as of Seq 3.0.
   2. Azure Event hub Integration – via Nuget plugin Seq.App.Azure.EventHub
3. Seq - Serilog’s Event message templates/ or Microsoft.Extensions.Logging – message templates are hashed => highly optimising searching/filtering. More here

{ https://nblumhardt.com/2016/07/event-types-structured-logging-concepts-in-net-4/}

1. SeriLog Analyser – Roslyn-based Visual Studio Plugin- validates message syntax and checks binding to parameters. Useful when attempting to log parameter values succinctly.
2. SUGGESTED IDEA FOR PSC: When logging events it might be useful to use generic **EventIDs**. For example assign integer values for events from our controllers:
3. public class LoggingEvents
4. {
5. public const int PAGE\_LOADED = 1000;
6. public const int GRID\_LOADED = 1001;
7. public const int GET\_ITEM = 1002;
8. public const int INSERT\_ITEM = 1003;
9. public const int UPDATE\_ITEM = 1004;
10. public const int DELETE\_ITEM = 1005;
11. public const int DELETE\_ITEM = 1005;
12. public const int EXPORT\_EXCEL = 1005;
13. public const int EXPORT\_PDF = 1005;
14. public const int EXPORT\_ZIP = 1005;
15. public const int GET\_ITEM\_NOTFOUND = 4000;
16. public const int UPDATE\_ITEM\_NOTFOUND = 4001;
17. }

Problems:

1. Free license for only a single user.

To Investigate:

1. If there are advanced functions/frameworks that making debugging and tracking issues through many microservices easy.
2. How SeriLog can be used in an-all-cloud set up i.e. Seq/RavenDB on Azure, how to install, configure, access etc. (**Create experiment**) add Serilog to ms.Lib.Services.Log as SeriLogService.cs

Do we set up logging for each (Exitinterview, grade etc) item or once (in Lib)? How would ‘enrich’, ’context’ and ambient logging work ?

1. Experiment - Add SeriLog, Seq to a clone of current eservices Locally, add logging and demo debugging, filtering etc. in Seq Portal.

* ~~Setup,~~
* Configuration,
* Adding API key => requires license
* Seq syntax Practice,
* Plot charts & graphs,
* Creating and saving views,
* bookmarking events on dash => (sharing with other PSC developers),
* Assign event to specific User = > **Give developer ownership of errors => requires License**
* Creating dashboard watches => (big counter on dashboard for each watch event)
* log retention time

1. How logging in PSC’s microservice architecture would work – setup, best frameworks, how we would work with the Log
   1. Do we have to log into an online portal for debugging purposes like the localhost Seq endpoint?
   2. Are there frameworks with better native/first class support for a microservice setup?
   3. How good is the Seq query?
   4. Investigate further the properties/syntax of Seq and SeriLog.
2. Seq Apps – Install from Nuget (or PSC custom) e.g. (Nuget)**Threshold Detection** – set a count for a desired event, when threshold reached and additional threshold event will be emitted (Alert).

* Seq.App.Azure.EventHub - The Azure Event Hub app for Seq pushes data from Seq into an Azure Event Hub
* Seq.App.Email - The email app for Seq formats events as text and sends them via SMTP email
* Seq.App.EmailPlus - An improved SMTP email app for Seq that uses Handlebars.NET for templating
* Seq.App.FirstOfType - An app for Seq that raises an event whenever it detects that a raised event type has not been seen previously

1. We can connect a Seq instance to Azure event hubs.

Resources:

{.NET docs, Logging in ASP Core Primer, https://docs.microsoft.com/en-us/aspnet/core/fundamentals/logging}

{Articles, Structured logging concepts in .NET Series (1), <https://nblumhardt.com/2016/06/structured-logging-concepts-in-net-series-1/>}

{Course, <https://app.pluralsight.com/library/courses/modern-structured-logging-serilog-seq/table-of-contents>}

{Docs, Seq, <http://docs.getseq.net/docs/getting-started>}

{SeriLog, <https://serilog.net/>}

{SeriLog, Notes on pros/cons of structured logging , http://softwareengineering.stackexchange.com/questions/312197/benefits-of-structured-logging-vs-basic-logging}

{Seq, Epic Article -Some (#2) on Seq, http://www.wwwlicious.com/tag/seq/}

**Azure Application Insights**

**Experiment**: Create a .NET Core Project (Types,data, business, API) with VS2017. Add application Insights and publish on Azure. Use Jmeter (Postman/Manual) tools to call the api on Azure and investigate the capabilities of Application Insight.

**Problems**: Live Metrics for .Net Core apps not currently working. “your app is offline or using an older SDK”.

Findings: Useful tools

1. Emails – Azure Application Insights sends an email summarising an overview of the heath of the ‘resources’ (i.e. APIs, Webapps etc. Application Analytics automatically (by default)sends out emails to (configured) stakeholders if anomalous activities are detected. (uses machine learning and data analytics
2. Overview tab – shows A graph with Requests in time (e.g. past hour).
3. Application Insights tab –
   1. Shows slowest requests (last 24 hours) – useful
   2. Livestream of – requests, failures, duration etc. – Very useful

The following functions are inside Application Insights

1. **Alerts** – Create configurable alerts (e.g. failures etc) with email recipients and web end points (e.g. for custom monitoring. PSC can set up emails and/or a web end point to monitor alerts and get alerted/ahead of problems (e.g. High memory/CPU overload on servers, database etc…). we can add these alerts to PSC’s (future) health monitoring App/web app.
2. **Live Metrics Stream**\* - Live graphs of incoming, outgoing requests and Server Health. Visually intuitive diagnosis of overall live issues/performance.
3. **Smart Detection –** emails and/or list of alerts. Alerts are failure anomalies, performance anomalies and Azure Cloud Service Issues (e.g. run time crashes, frequent recycling etc.). Alert succinctly has three lines – When, What, Note.

(https://docs.microsoft.com/en-us/azure/application-insights/app-insights-proactive-diagnostics)

1. **Availability –** After deploying a web App/Rest API, set up test s to monitor its availability and responsiveness. It sends web requests at regular intervals and alerts if app doesn’t respond/responds slowly. Test can be
   1. URL ping test – simple test created an azure portal
   2. Multi-Step Web test- Created in Visual Studio Enterprise and upload to the portal.

(https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability)

1. **Application Map** – Is a visual layout of the dependency relationships of your application components. Can see API, database, we app, microservice as cards with arrows showing relationships, very similar to SQL table/UML diagrams. Has intuitive coloured symbols to find issues at a glance. It contains page view counts, percentage of failure, etc.

(https://docs.microsoft.com/en-us/azure/application-insights/app-insights-app-map)

