Log – your missing feature

KONRAD PIORUNEK

WE HAVE GREAT LOGS
IT'S FINE

Good log practices

You can google them.

Over first 3 results pages you can gather ~30 rules.

Can you memorize them?

I want to propose 2 concepts related to each other.*

*There's a lot of details to those concepts.

- Concepts should always be applied.
- Advice doesn't have to be applied in 100% cases.

Context

- Whether you like it or not logs allow solving multitude of problems and optimize many systems.
  - We discuss enterprise backend.
  - A lot of concepts will apply to other applications.
- We are not talking technical solution in this lecture. Focus on contents instead.
  - Most of it is easy to google.
  - I encourage using your own wrapper/interface for logging.
  - I encourage using log aggregation tooling if you can (Splunk, ELK, Loggly, Papertrail, Graylog, etc.)
- Logs aggregation and structured logging might be a topic for a separate lecture.
  - They provide statistical analysis and extensive querying options.

What is log for?

Typical story of a log in an application

- Use existing library for logging: log4net, slf4j, serilog, etc.
- Typical log entry:

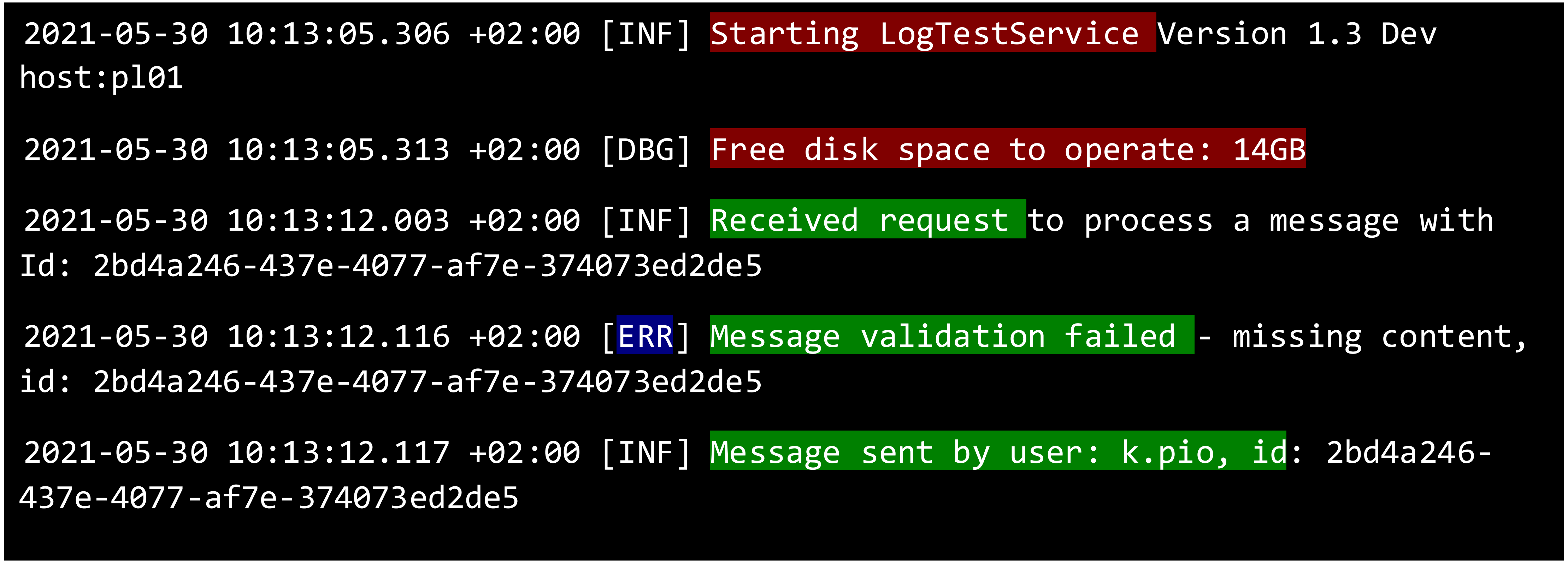
```
2021-03-22 11:49:33,441 [12] DEBUG [LogTestSample.Features.Domain.Validator] Reviewed
data received in a request...
```

- Contains:
  - Date
  - Time
  - Thread
  - Log level
  - Class
  - Log entry
  - …

Where do log go?

- There are various places where logs go:
  - Console
  - File
  - Dedicated service for logs aggregation
  - Windows OutputDebugString
  - Database
  - Browser console
- It’s all technical focus.
- But logs are used. The users should tell you what’s needed.

Logs help solving problems (1)



```
2021-05-30 10:13:05.306 +02:00 [INF] Starting LogTestService Version 1.3 Dev
host:pl01

2021-05-30 10:13:05.313 +02:00 [DBG] Free disk space to operate: 14GB

2021-05-30 10:13:12.003 +02:00 [INF] Received request to process a message with
Id: 2bd4a246-437e-4077-af7e-374073ed2de5

2021-05-30 10:13:12.116 +02:00 [ERR] Message validation failed - missing content,
id: 2bd4a246-437e-4077-af7e-374073ed2de5

2021-05-30 10:13:12.117 +02:00 [INF] Message sent by user: k.pio, id: 2bd4a246-
437e-4077-af7e-374073ed2de5
```

Logs help solving problems (1)

App log is an **evidence** of what app did.

Useful to **support** to inspect reported incident.

Can be used for active **monitoring** systems.

User activity documented.

Logs help solving problems (2)

```
2021-05-30 15:24:54.246 +02:00 [INF] Received request for article stats. reqId=asb128acf, artId=art324,
user=kpio01

2021-05-30 15:24:54.248 +02:00 [INF] Executing sql query to get stats. reqId=asb128acf

2021-05-30 15:24:54.569 +02:00 [INF] Sql query finished in 320ms. Received 301 rows. reqId=asb128acf

2021-05-30 15:24:54.569 +02:00 [INF] Sending request to get article metadata information. reqId=asb128acf,
 host=http://metadataservice.local.net/meta/?id=art324

2021-05-30 15:24:56.116 +02:00 [INF] Received metadata response. reqId=asb128acf

2021-05-30 15:24:56.117 +02:00 [INF] Formatting results for request. reqId=asb128acf

2021-05-30 15:24:56.119 +02:00 [INF] Sending response for request. reqId=asb128acf, responseSize=6724
```

Logs help solving problems (2)

You can get **usage stats** which can help finding which features are used the most or unused.

Log already contains **performance** measurements.

- Answer performance complaints.
- Service down analysis.
- Usual behavior patterns.

Debugging the application.

- Evidence of behavior for analysis when it can’t be predicted or simulated until real system usage starts.

Logs should be considered a feature

Short example showed numerous benefits.

IT IS WORTH INVESTMENT AS A FEATURE ON ITS OWN!

Log deals with the unknown, but often no one knows how to approach the unknown.

- Oftentimes no one knows what should be written as a log content.
- With simple concepts it can be validated if the log is better.
- Treated as a feature: it should be specified what should go to the log alongside other functional requirements.

What to write into the log?

TURN MANY ADVICE INTO SIMPLE CONCEPTS

Where it all begins?

Case: We receive performance complaint from the user of our service.

Problem 1 – Service requests

```
15:24:55.246 Request received /resource/id=1293

15:24:55.248 Request received /resource/id=1293

15:24:56.556 Sending response for id=1293

15:25:07.634 Sending response for id=1293
```

YOU KNOW NOTHING
FROM YOUR LOGS

Problem 1 – Service requests

```
15:24:55.246 Request received /resource/id=1293

15:24:55.248 Request received /resource/id=1293

15:24:56.556 Sending response for id=1293

15:25:07.634 Sending response for id=1293
```

Problem 1 – Service requests

```
15:24:55.246 Request received /resource/id=1293

15:24:55.248 Request received /resource/id=1293

15:24:56.556 Sending response for id=1293, size=600, total=208, db=0, cacheUsed=1

15:25:07.634 Sending response for id=1293, size=600, total=12388, db=12100, cacheUsed=0
```

Problem 1 – Service requests

```
15:24:55.246 Request received /resource/id=1293, user=A123043

15:24:55.248 Request received /resource/id=1293, user=B736222

15:24:56.556 Sending response for id=1293, size=600, total=208, db=0, cacheUsed=1

15:25:07.634 Sending response for id=1293, size=600, total=12388, db=12100, cacheUsed=0
```

Problem 1 – Service requests

```
15:24:55.246 Request received /resource/id=1293, user=A123043, requestId=210389

15:24:55.248 Request received /resource/id=1293, user=B736222, requestId=210390

15:24:56.556 Sending response for id=1293, size=600, total=208, db=0, cacheUsed=1, requestId=210390

15:25:07.634 Sending response for id=1293, size=600, total=12388, db=12100, cacheUsed=0, requestId=210389
```

Problem 1 – Log improvements

Advice 1: log total duration and each step breakdown for bottleneck detection.

Advice 2: preferably log everything you can. For HTTP: IP, headers, endpoint used, user, etc.

We also added a connection between log entries.

Correlation

A mutual relationship or connection between two or more things.

Links related log entries so we can be sure of the relation between them.

Correlation is used for ‚explicit correlation’ when made up ID is used to match actions or logs related with each other.

I want to extend this concept whenever possible.

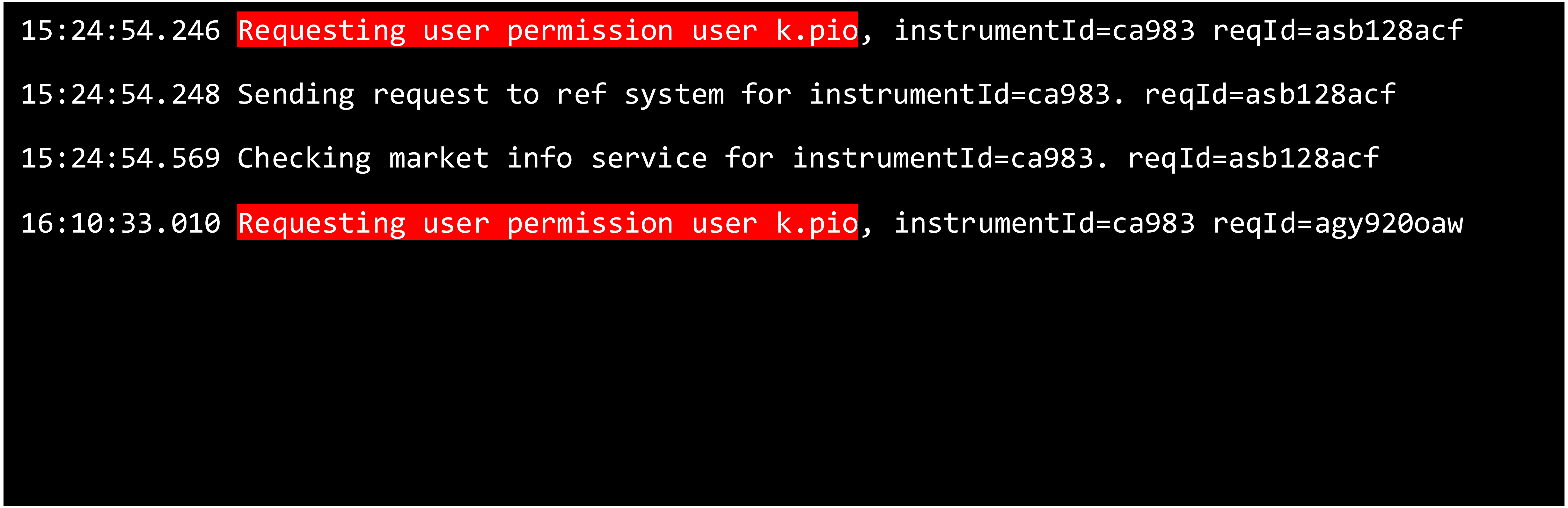
Concept 1: log should always be correlated.

Problem 2 – User activity

Case: We receive performance complaint from the user of our service.

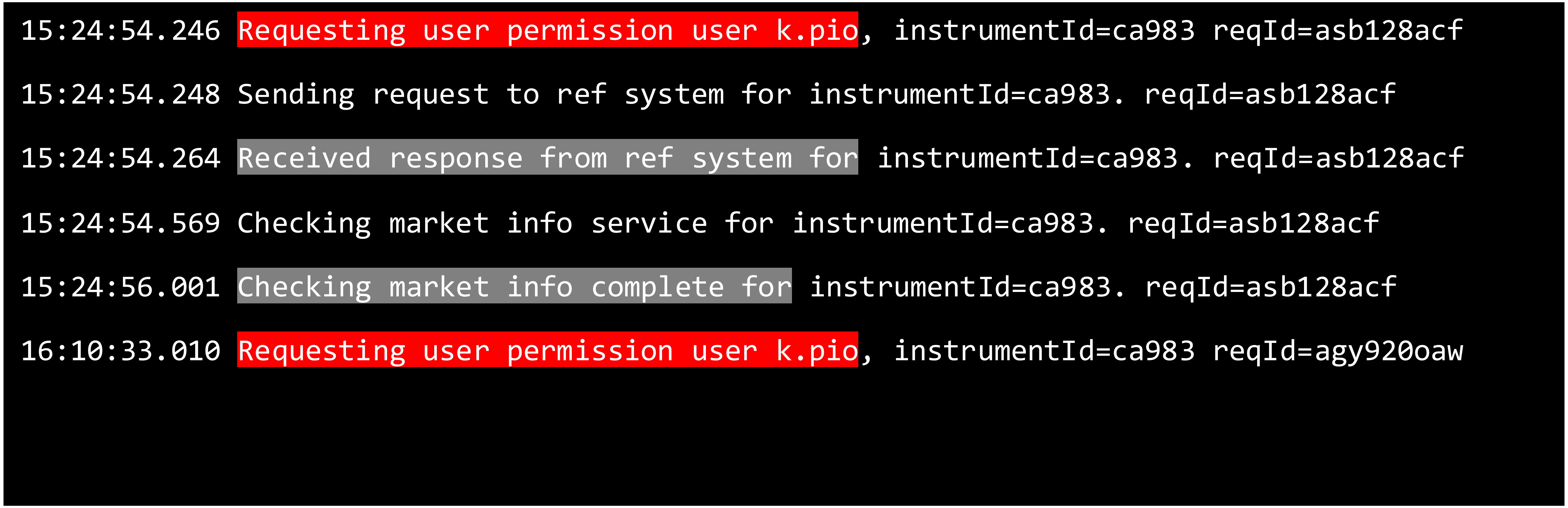
Namely: ‚My request didn’t finish imediately but it took 45mins to complete.’

Problem 2 – User activity



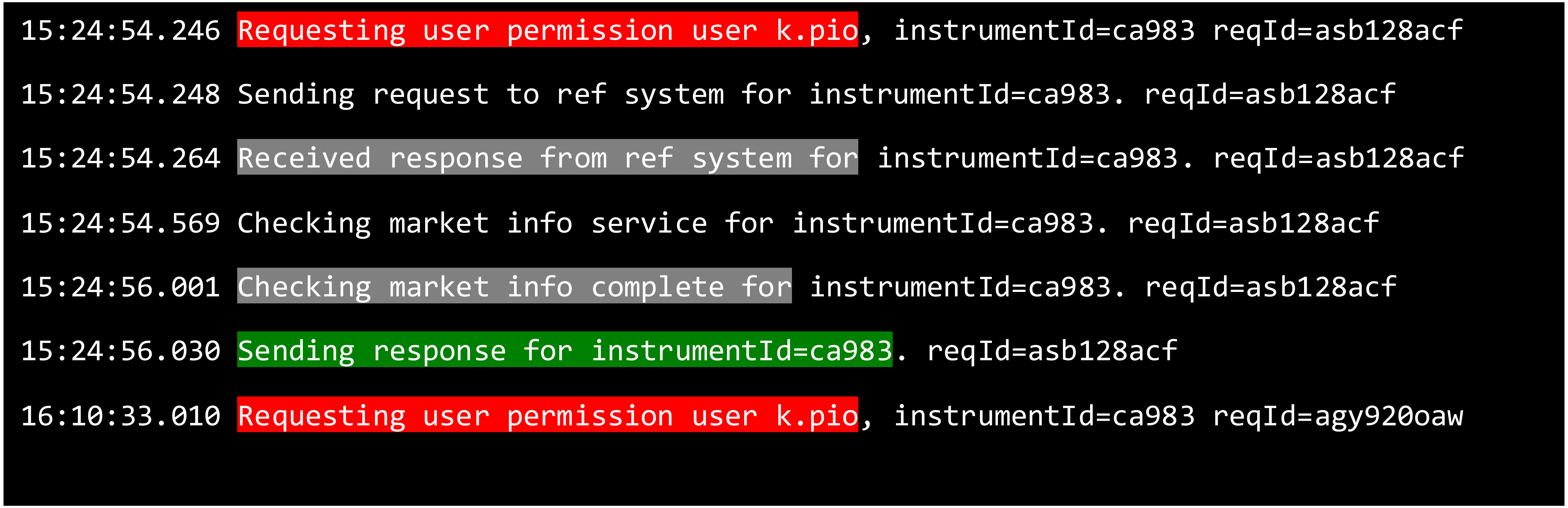
```
15:24:54.246 Requesting user permission user k.pio, instrumentId=ca983 reqId=asb128acf

15:24:54.248 Sending request to ref system for instrumentId=ca983. reqId=asb128acf

15:24:54.569 Checking market info service for instrumentId=ca983. reqId=asb128acf

16:10:33.010 Requesting user permission user k.pio, instrumentId=ca983 reqId=agy920oaw
```

Problem 2 – User activity



```
15:24:54.246 Requesting user permission user k.pio, instrumentId=ca983 reqId=asb128acf

15:24:54.248 Sending request to ref system for instrumentId=ca983. reqId=asb128acf

15:24:54.264 Received response from ref system for instrumentId=ca983. reqId=asb128acf

15:24:54.569 Checking market info service for instrumentId=ca983. reqId=asb128acf

15:24:56.001 Checking market info complete for instrumentId=ca983. reqId=asb128acf

16:10:33.010 Requesting user permission user k.pio, instrumentId=ca983 reqId=agy920oaw
```

Problem 2 – User activity



```
15:24:54.246 Requesting user permission user k.pio, instrumentId=ca983 reqId=asb128acf

15:24:54.248 Sending request to ref system for instrumentId=ca983. reqId=asb128acf

15:24:54.264 Received response from ref system for instrumentId=ca983. reqId=asb128acf

15:24:54.569 Checking market info service for instrumentId=ca983. reqId=asb128acf

15:24:56.001 Checking market info complete for instrumentId=ca983. reqId=asb128acf

15:24:56.030 Sending response for instrumentId=ca983. reqId=asb128acf

16:10:33.010 Requesting user permission user k.pio, instrumentId=ca983 reqId=agy920oaw
```

Problem 2 – User activity

By logging when each operation starts and ends we see what actually happened over time.

Helps with **support** and **user activity**.

Technique used – wrap with logs entire request and it’s steps: this is 2nd concept.

In and out rule

*Concept 2: When getting into or going out of the scope make log entry before and after.**

*This generalization has a lot of details.

*Effectively logs should go in pairs**

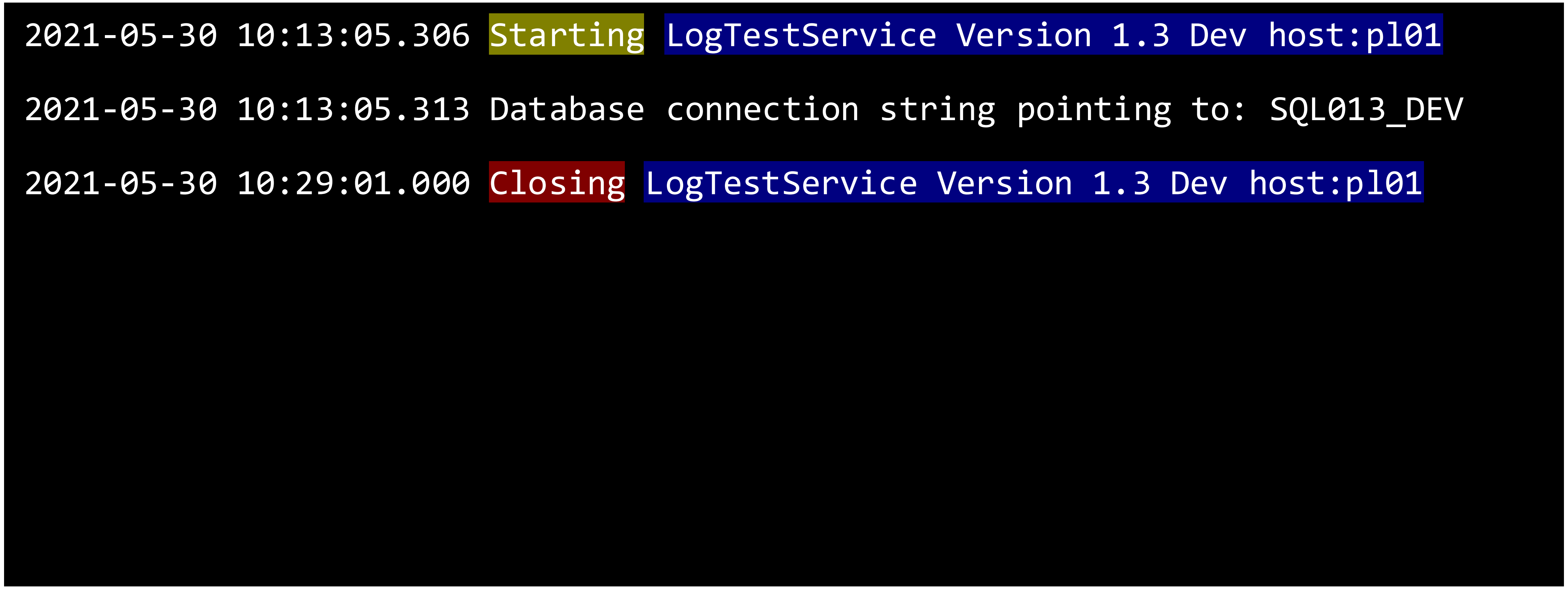
**Unless they don’t in some cases.

It should be used:

- Around important step of processing in the current process.
- When releasing control outside of the process: like db query or sending HTTP request.
- When receiving control from outside of the process: receiving HTTP request.

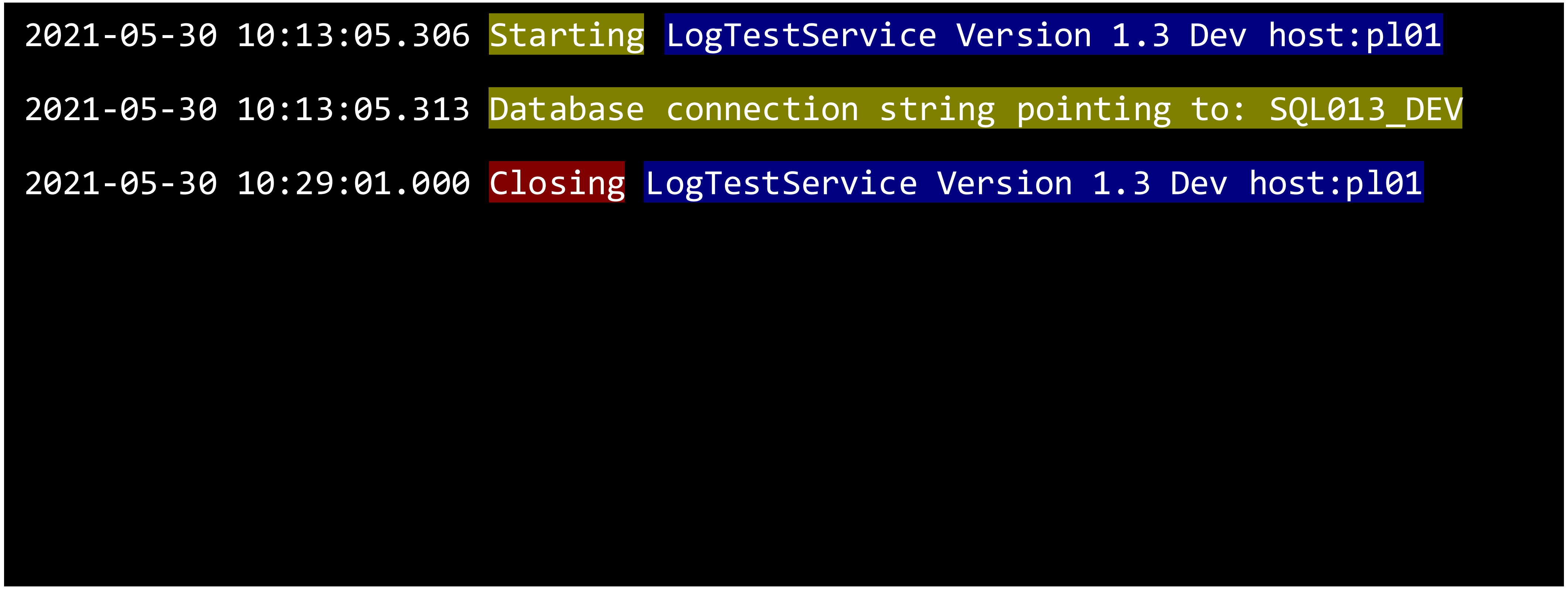
Application of the concepts

Program start mandatory logs



```
2021-05-30 10:13:05.306 Starting LogTestService Version 1.3 Dev host:pl01

2021-05-30 10:13:05.313 Database connection string pointing to: SQL013_DEV

2021-05-30 10:29:01.000 Closing LogTestService Version 1.3 Dev host:pl01
```

Program start mandatory logs



```
2021-05-30 10:13:05.306 Starting LogTestService Version 1.3 Dev host:pl01

2021-05-30 10:13:05.313 Database connection string pointing to: SQL013_DEV

2021-05-30 10:29:01.000 Closing LogTestService Version 1.3 Dev host:pl01
```

Program start mandatory logs

- **Advice 3**: Preferably always log the below:
  - App start/stop, environment, version, server name, configuration used.
- Such entries document what is used actually, where, and with what config.
- Detects errors in config/build/deployment
  - You can believe values were set, but in fact they are not.
- Log has correlation to server/instance/running version of the code, etc.
  Log uses in/out rule: beginning and end of program execution (important step).

Message in log entries

‚Should be informative and easy to understand’

- People and Internet says.
- But what it means?
- We actually use sentences only because logs are read by a human being.

- Self explanatory is convenient.
  - Who can evidence that these are self explanatory?
  - A non developer might read it and conclude a fix.
  - Works if they don't have to check the code.
- There is no rule. Half of advice on the internet can’t be applied.
  - ~~Log not too much / too little.~~
  - ~~Be descriptive~~

Message in log entries

```
2021-05-30 10:13:05.314 [INF] Scheduled batch job with JobId=52e8ce39-5a2c-41d4-870a-
60aa7338288d

2021-05-30 10:13:05.314 [DBG] Reading of file file=ftp\uploads\2021_04_12_data.csv started.
JobId=52e8ce39-5a2c-41d4-870a-60aa7338288d

2021-05-30 10:13:09.328 [ERR] Error processing a file. JobId=52e8ce39-5a2c-41d4-870a-
60aa7338288d, file=ftp\uploads\2021_04_12_data.csv
```

Message in log entries

```
2021-05-30 10:13:05.314 [INF] Scheduled batch job with JobId=52e8ce39-5a2c-41d4-870a-
60aa7338288d might not be understood

2021-05-30 10:13:05.314 [DBG] Reading of file file=ftp\uploads\2021_04_12_data.csv started.
JobId=52e8ce39-5a2c-41d4-870a-60aa7338288d a search term is split

2021-05-30 10:13:09.328 [ERR] Error processing a file. JobId=52e8ce39-5a2c-41d4-870a-
60aa7338288d, file=ftp\uploads\2021_04_12_data.csv there might be a lot of different
processing types
```

Message in log entries

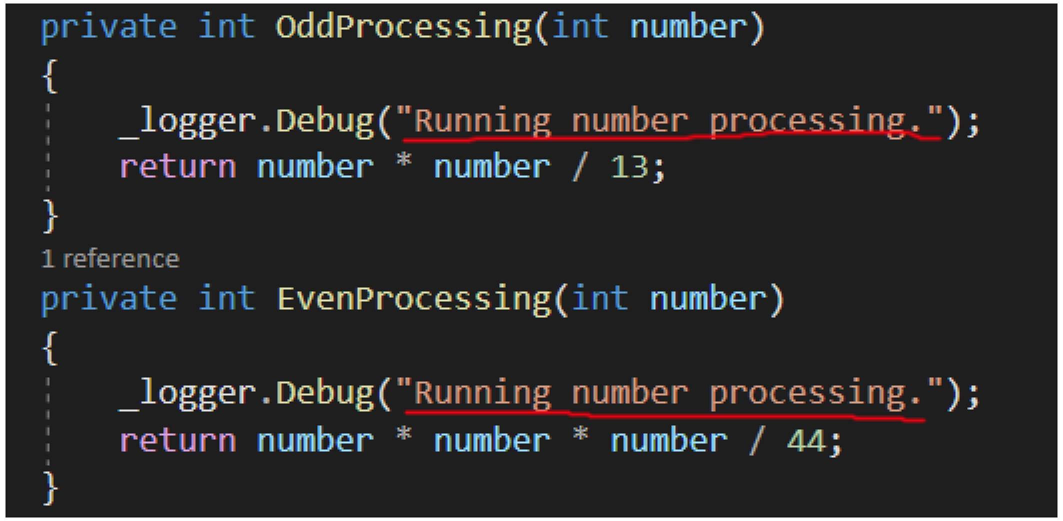
- **Advice 4**: Text in each code line should be unique for easy search.
  - Be able to check the code for specific entry.
  - Be able to search the log for specific entry.
- This doesn’t work when log search term is mixed with data and text search is impossible to find unique entry.
  - Transform the entry to have **unique part at the beginning.**

Log has correlation the line in code!

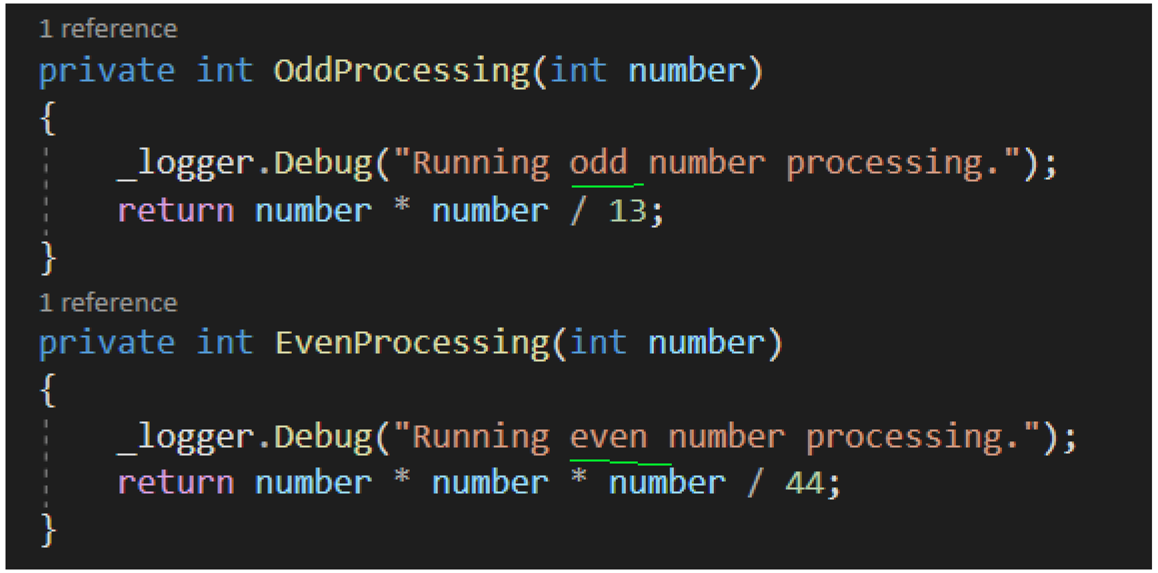
Message in log entries - fix

```
2021-05-30 10:13:05.314 +02:00 [INF] Scheduled batch job for csv delta with
JobId=52e8ce39-5a2c-41d4-870a-60aa7338288d

2021-05-30 10:13:05.314 +02:00 [DBG] Reading of csv delta file started
file=ftp\uploads\2021_04_12_data.csv. JobId=52e8ce39-5a2c-41d4-870a-
60aa7338288d

2021-05-30 10:13:09.328 +02:00 [ERR] Error processing a csv delta file.
JobId=52e8ce39-5a2c-41d4-870a-60aa7338288d,
file=ftp\uploads\2021_04_12_data.csv
```

Message in log entries – code fix

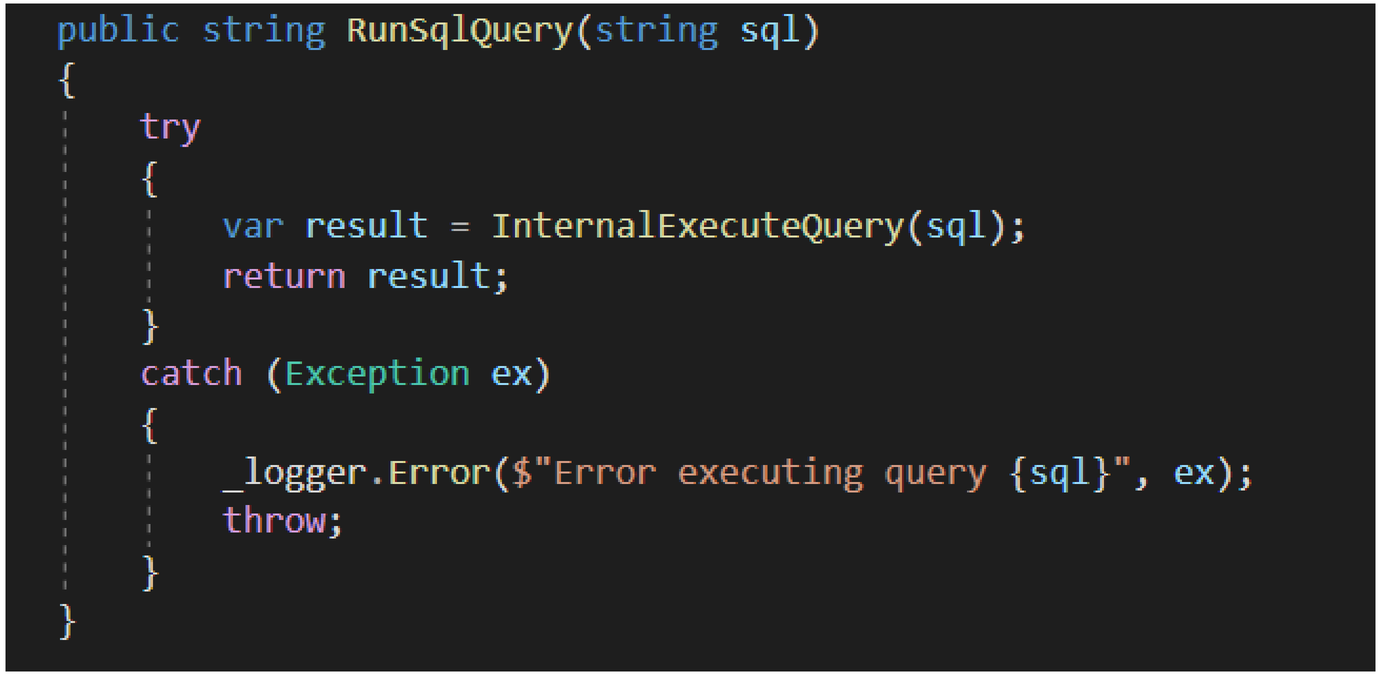


```
private int OddProcessing(int number)
{
    _logger.Debug("Running number processing.");
    return number * number / 13;
}
1 reference
private int EvenProcessing(int number)
{
    _logger.Debug("Running number processing.");
    return number * number * number / 44;
}
```



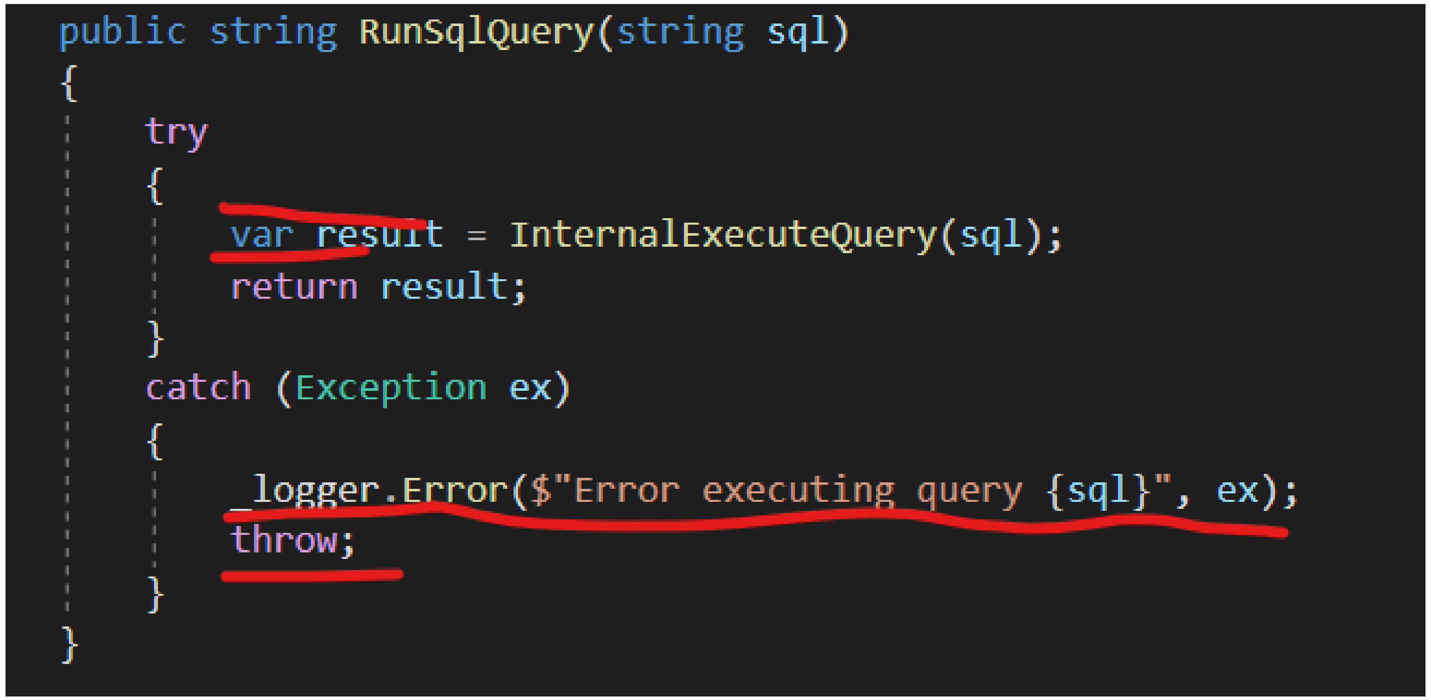
```
1 reference
private int OddProcessing(int number)
{
    _logger.Debug("Running odd number processing.");
    return number * number / 13;
}
1 reference
private int EvenProcessing(int number)
{
    _logger.Debug("Running even number processing.");
    return number * number * number / 44;
}
```

Exception logging – fix code



```
public string RunSqlQuery(string sql)
{
    try
    {
        var result = InternalExecuteQuery(sql);
        return result;
    }
    catch (Exception ex)
    {
        _logger.Error($"Error executing query {sql}", ex);
        throw;
    }
}
```

Exception logging – fix code

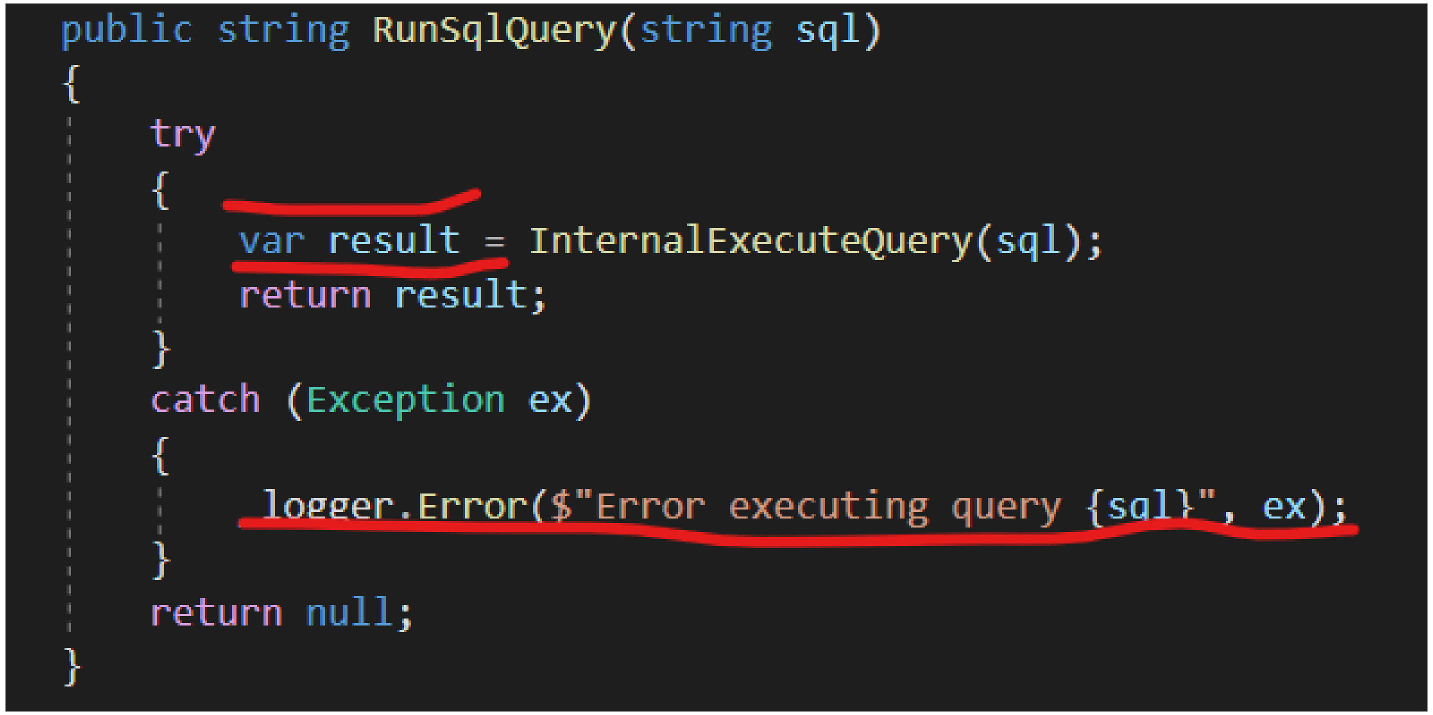


```
public string RunSqlQuery(string sql)
{
    try
    {
        var result = InternalExecuteQuery(sql);
        return result;
    }
    catch (Exception ex)
    {
        _logger.Error($"Error executing query {sql}", ex);
        throw;
    }
}
```

Exception logging

- Advice 5: Logged exception is a handled exception.
  - Intuitively exception is handled and logged immediately when it can occur.
- Rethrowing causes handling on top and… most likely logging again.
  - No need to rethrow the same exception in this case.
- This is a bit off topic, but I’ll count this advice.

Exception logging – no retrhow

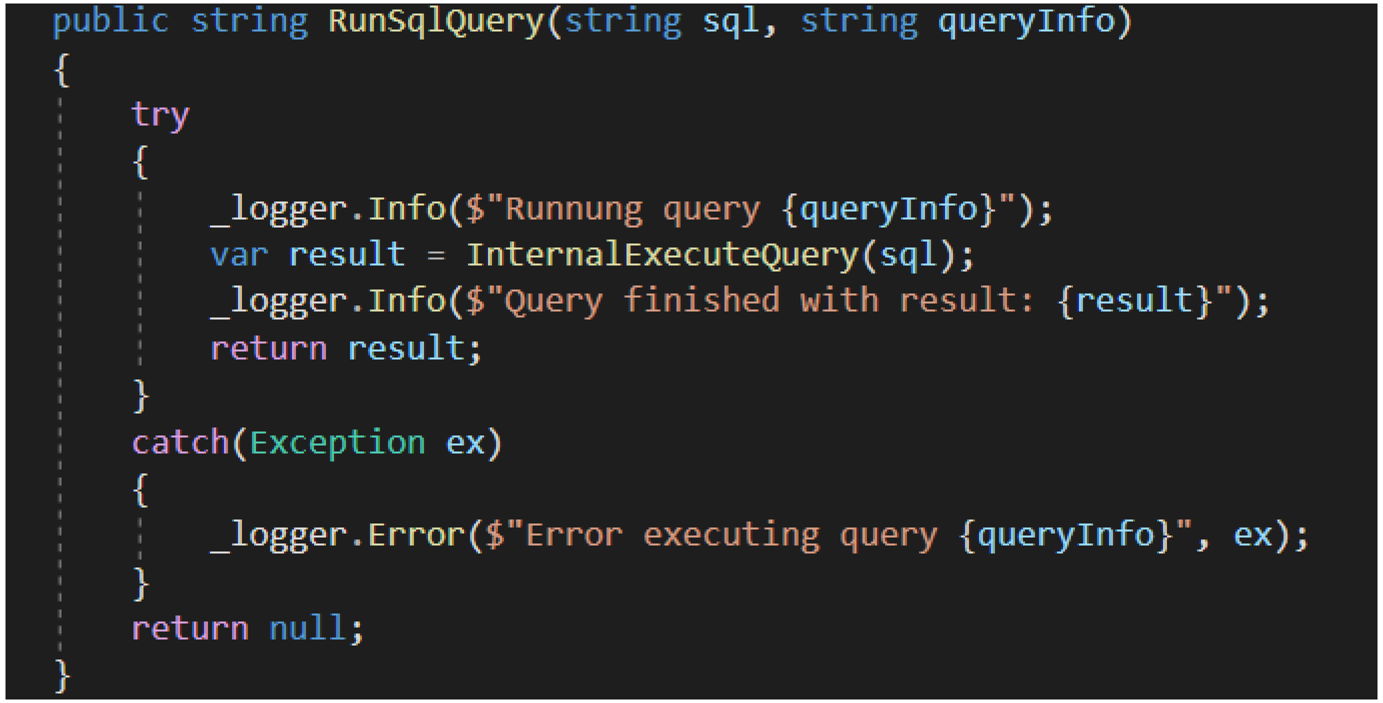


```
public string RunSqlQuery(string sql)
{
    try
    {
        var result = InternalExecuteQuery(sql);
        return result;
    }
    catch (Exception ex)
    {
        logger.Error($"Error executing query {sql}", ex);
    }
    return null;
}
```

Exception logging

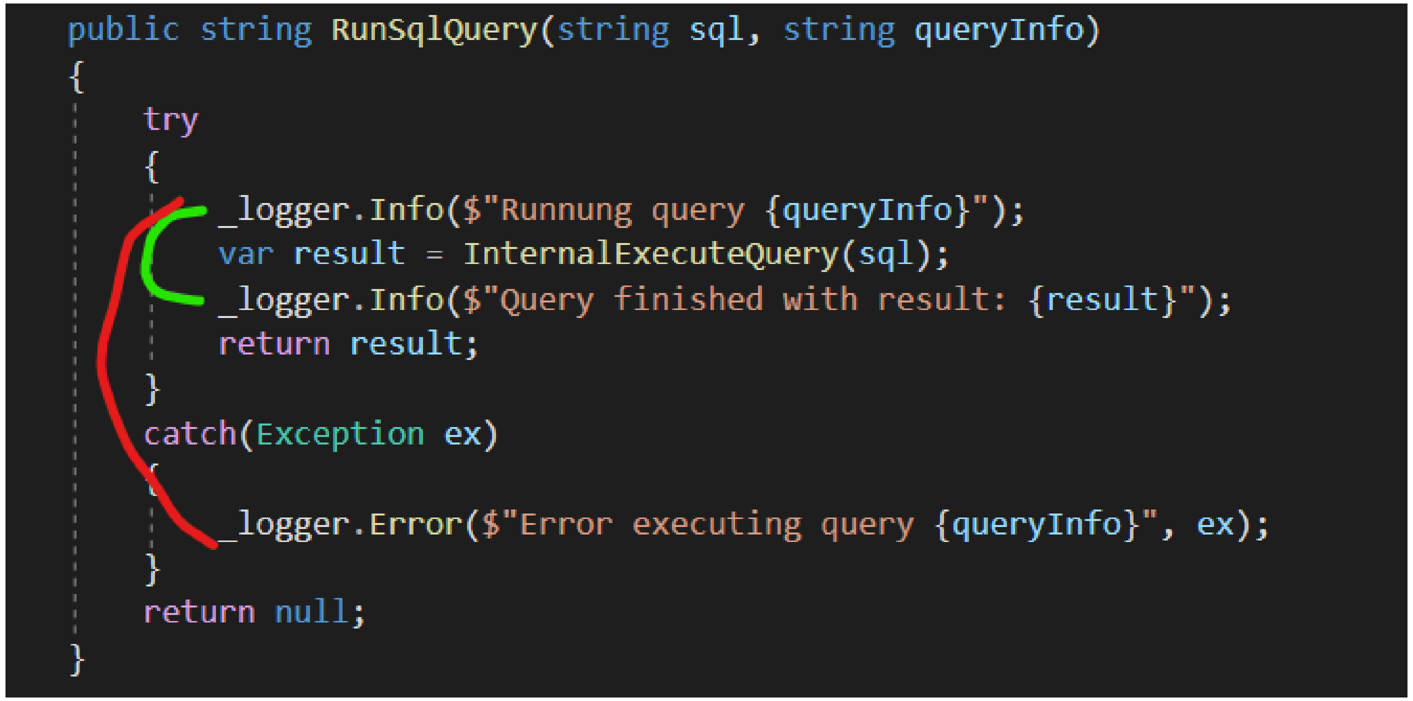
- Advice 6: Log successful and erroneous execution path.
  - This will evidence regular behavior.
  - Know your system!
- Logging success and error path is making the logs go in pairs – in/out rule.

Exception logging

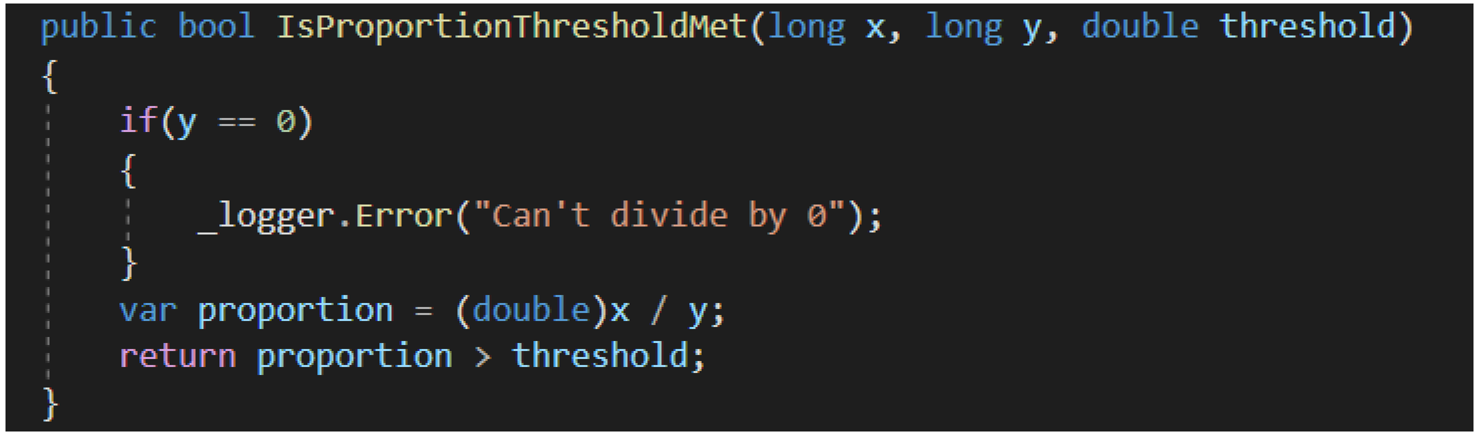


```
public string RunSqlQuery(string sql, string queryInfo)
{
    try
    {
        _logger.Info($"Runnung query {queryInfo}");
        var result = InternalExecuteQuery(sql);
        _logger.Info($"Query finished with result: {result}");
        return result;
    }
    catch(Exception ex)
    {
        _logger.Error($"Error executing query {queryInfo}", ex);
    }
    return null;
}
```

Exception logging


```
public string RunSqlQuery(string sql, string queryInfo)
{
    try
    {
        _logger.Info($"Runnung query {queryInfo}");
        var result = InternalExecuteQuery(sql);
        _logger.Info($"Query finished with result: {result}");
        return result;
    }
    catch(Exception ex)
    {
        _logger.Error($"Error executing query {queryInfo}", ex);
    }
    return null;
}
```

Algorithms and logging



```
public bool IsProportionThresholdMet(long x, long y, double threshold)
{
    if(y == 0)
    {
        _logger.Error("Can't divide by 0");
    }
    var proportion = (double)x / y;
    return proportion > threshold;
}
```

Write a test
Write a log

Algorithms and logging

Advice 7: Algorithmic code should contain no logs.

- Maybe logs are not the place for wrong results?

- Unit and integration tests are better tool to control the code behavior.

- Only code that uses algorithms should do the logging based on result.

- Logging here may lead to logging again on upper layer.

- It may lead to placing logs on hot path where it might be a bad idea.

Advice 8: Avoid logs on hot path.

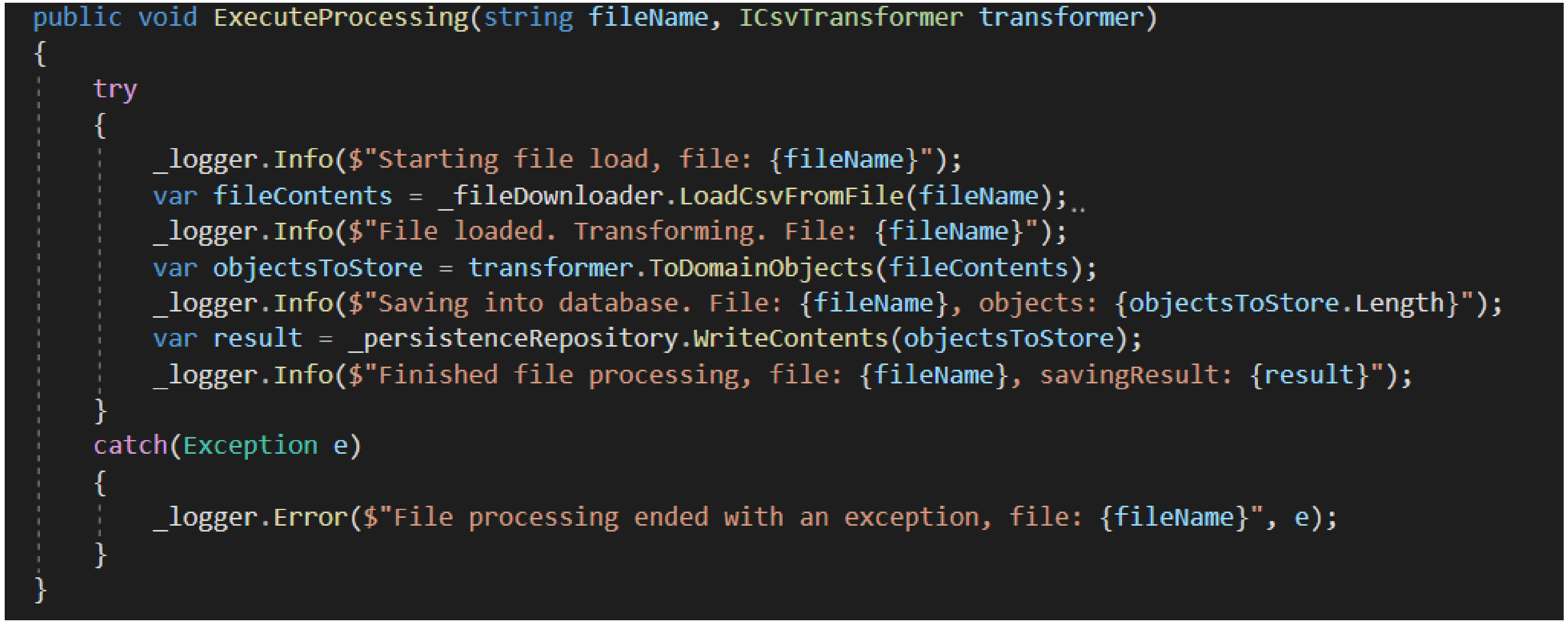
An in/out rule is violated. It is a way to show a ‚smell’ here like with exceptions.

How deep should logs go?

So how deep should logs go?

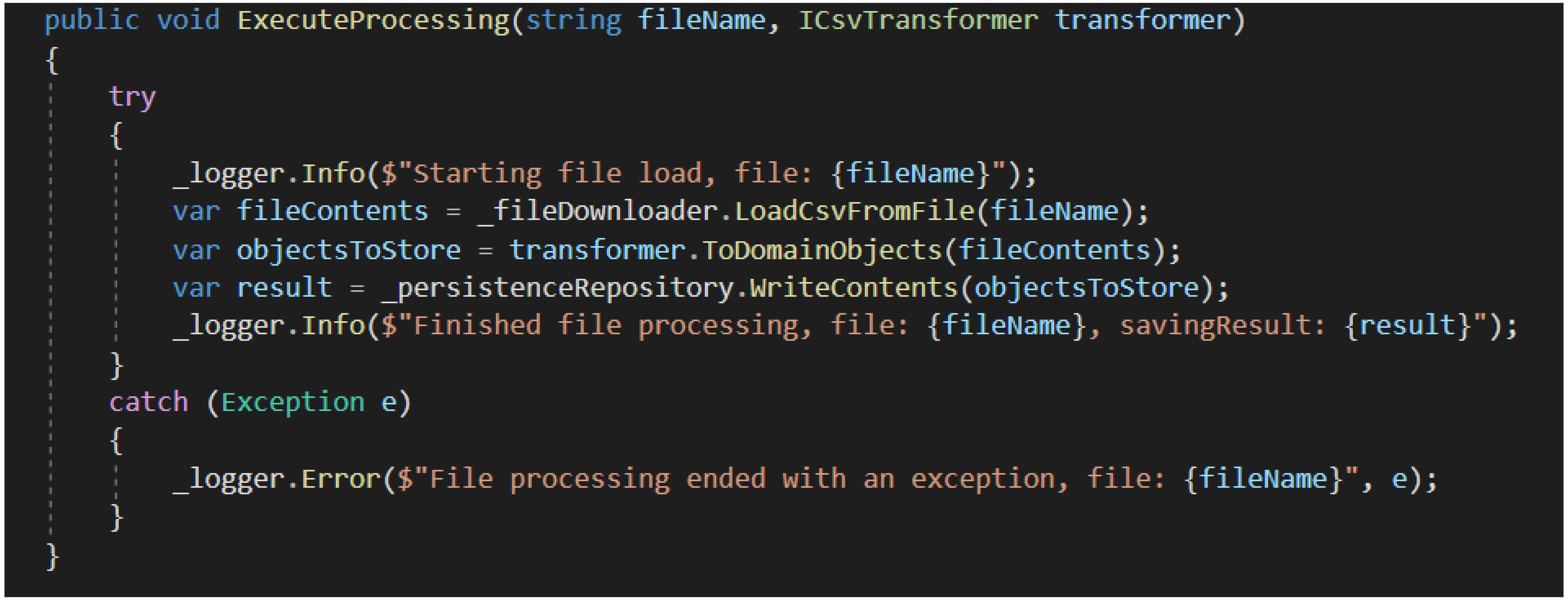
- **Advice 9**: Log information for big part or important step.
  - Your/team decision. No golden rule here.
- **Correlate** the steps with unique id, business id or any relevant data in the context.
- **In/out rule** optimised: a single entry between steps acts as first and last entry.
- In/out can be skipped for no duration important events. Like cache cleared event.

How deep? – Raw example



```
public void ExecuteProcessing(string fileName, ICsvTransformer transformer)
{
    try
    {
        _logger.Info($"Starting file load, file: {fileName}");
        var fileContents = _fileDownloader.LoadCsvFromFile(fileName);
        _logger.Info($"File loaded. Transforming. File: {fileName}");
        var objectsToStore = transformer.ToDomainObjects(fileContents);
        _logger.Info($"Saving into database. File: {fileName}, objects: {objectsToStore.Length}");
        var result = _persistenceRepository.WriteContents(objectsToStore);
        _logger.Info($"Finished file processing, file: {fileName}, savingResult: {result}");
    }
    catch(Exception e)
    {
        _logger.Error($"File processing ended with an exception, file: {fileName}", e);
    }
}
```

How deep? – Aspect/inner example



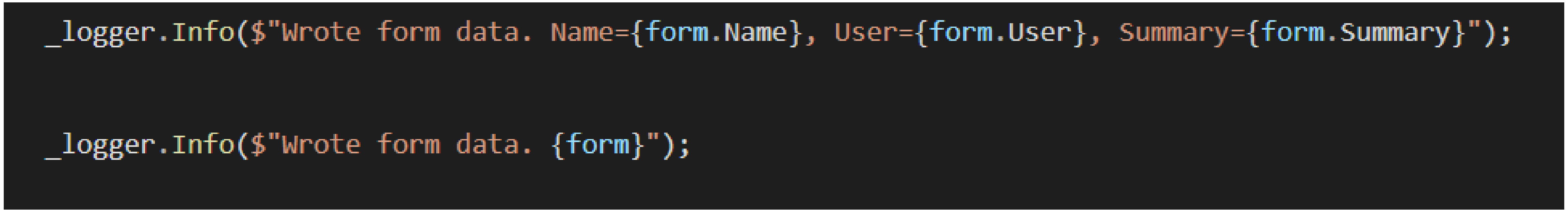
```
public void ExecuteProcessing(string fileName, ICsvTransformer transformer)
{
    try
    {
        _logger.Info($"Starting file load, file: {fileName}");
        var fileContents = _fileDownloader.LoadCsvFromFile(fileName);
        var objectsToStore = transformer.ToDomainObjects(fileContents);
        var result = _persistenceRepository.WriteContents(objectsToStore);
        _logger.Info($"Finished file processing, file: {fileName}, savingResult: {result}");
    }
    catch (Exception e)
    {
        _logger.Error($"File processing ended with an exception, file: {fileName}", e);
    }
}
```

How deep should logs go?

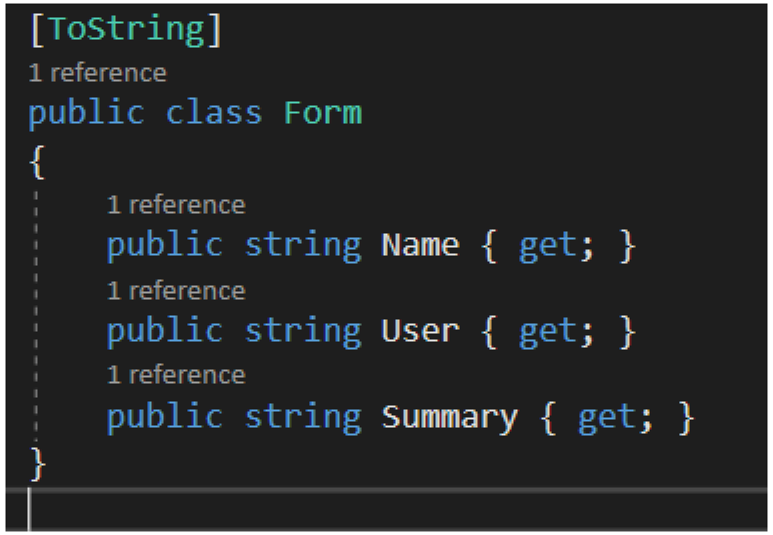
What can help writing loging code?

- Aspect approach – less code and cleaner via attriubte/middleware/DI/annotation.
  - Spring, Puresharp, Autofac, etc.
- Generate code for your objects automtically and use the objects ToString() in logs
  - Lombok / ToString.Fody, etc.

How deep? – ToString() usage



```
_logger.Info($"Wrote form data. Name={form.Name}, User={form.User}, Summary={form.Summary}");


_logger.Info($"Wrote form data. {form}");
```



```
[ToString]
1 reference
public class Form
{
    1 reference
    public string Name { get; }
    1 reference
    public string User { get; }
    1 reference
    public string Summary { get; }
}
```

In/out deep dive

Getting in/out of your own system to other ones should be evidenced:

- **Advice 10**: Evidence receiving or loss of control in the process: database query, system API, *other* service call, user input, message queue, a callback from scheduling, etc.
- **Advice 11**: Your own different services are also ‚*other* services'.
- **Advice 12**: Backend and frontend are also *other (separate)* services.
  - This advice is in case you thinking an API is not used by *another* service/user because it is just frontend.

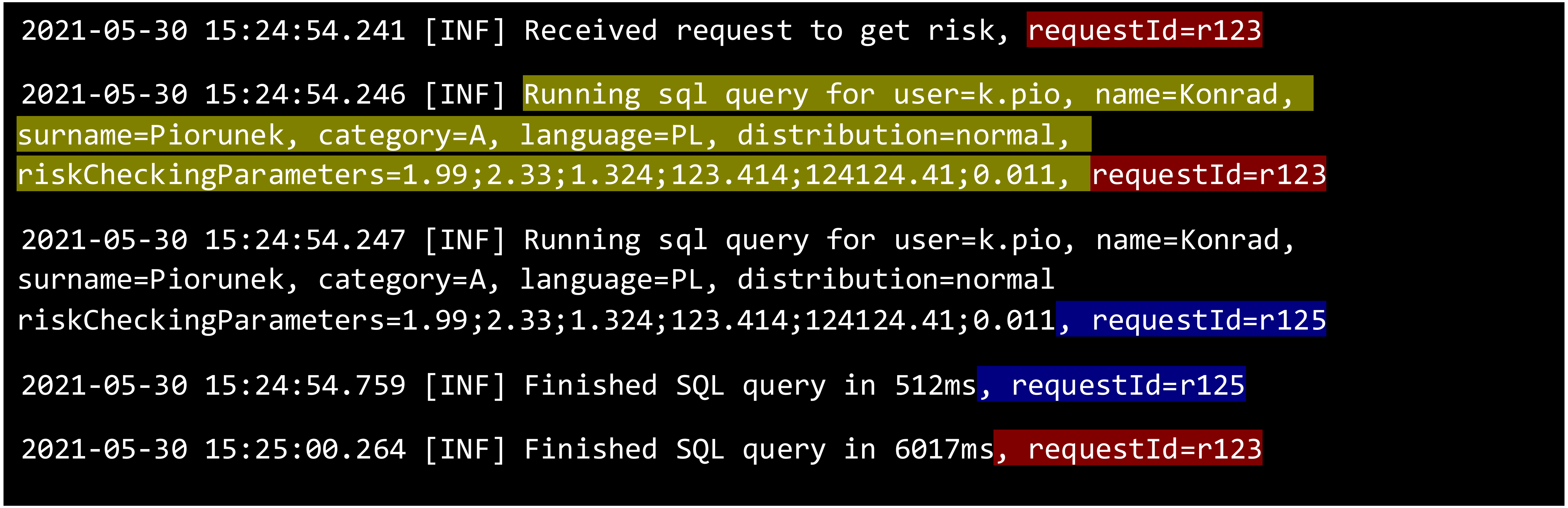
Explicit correlation deep dive

```
2021-05-30 15:24:54.246 [INF] Running sql query for user=k.pio, name=Konrad,
surname=Piorunek, category=A, language=PL, distribution=normal,
riskCheckingParameters=1.99;2.33;1.324;123.414;124124.41;0.011

2021-05-30 15:24:54.247 [INF] Running sql query for user=k.pio, name=Konrad,
surname=Piorunek, category=A, language=PL, distribution=normal
riskCheckingParameters=1.99;2.33;1.324;123.414;124124.41;0.011

2021-05-30 15:24:54.759 [INF] Finished SQL query in 512ms.

2021-05-30 15:25:00.264 [INF] Finished SQL query in 6017ms.
```

Explicit correlation deep dive

```
2021-05-30 15:24:54.246 [INF] Running sql query for user=k.pio, name=Konrad,
surname=Piorunek, category=A, language=PL, distribution=normal,
riskCheckingParameters=1.99;2.33;1.324;123.414;124124.41;0.011, requestId=r123

2021-05-30 15:24:54.247 [INF] Running sql query for user=k.pio, name=Konrad,
surname=Piorunek, category=A, language=PL, distribution=normal
riskCheckingParameters=1.99;2.33;1.324;123.414;124124.41;0.011, requestId=r125

2021-05-30 15:24:54.759 [INF] Finished SQL query in 512ms, requestId=r125

2021-05-30 15:25:00.264 [INF] Finished SQL query in 6017ms, requestId=r123
```

Explicit correlation deep dive



```
2021-05-30 15:24:54.241 [INF] Received request to get risk, requestId=r123

2021-05-30 15:24:54.246 [INF] Running sql query for user=k.pio, name=Konrad,
surname=Piorunek, category=A, language=PL, distribution=normal,
riskCheckingParameters=1.99;2.33;1.324;123.414;124124.41;0.011, requestId=r123

2021-05-30 15:24:54.247 [INF] Running sql query for user=k.pio, name=Konrad,
surname=Piorunek, category=A, language=PL, distribution=normal
riskCheckingParameters=1.99;2.33;1.324;123.414;124124.41;0.011, requestId=r125

2021-05-30 15:24:54.759 [INF] Finished SQL query in 512ms, requestId=r125

2021-05-30 15:25:00.264 [INF] Finished SQL query in 6017ms, requestId=r123
```

Explicit correlation deep dive

Correlation id provides context tracking for each line written that has meaning together.

Can use business identifier that is used in the system. Safer is to use explicit, made up id.

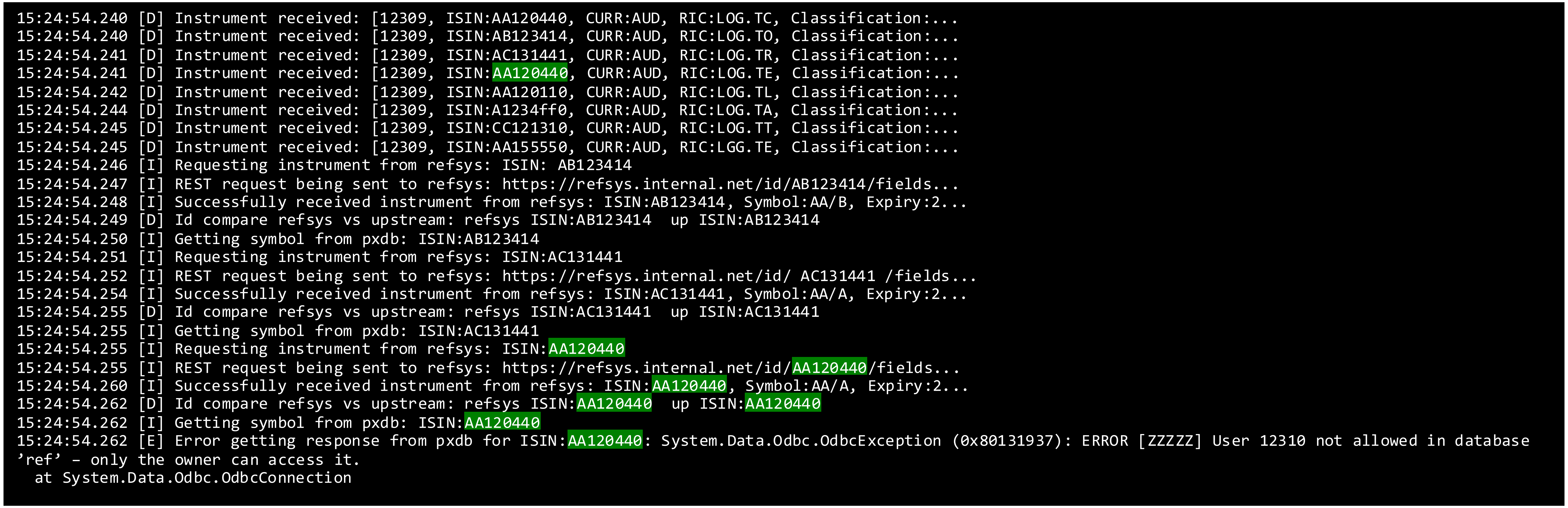
Can use explicit identifier just for logging purpose.

- Single line with business identifier can be matched and then full history can be searched by explicit id.
- A line in the log can be identified with explicit identifier and then related back to business id.
- You might already have a correlation using business Id.

Business id example

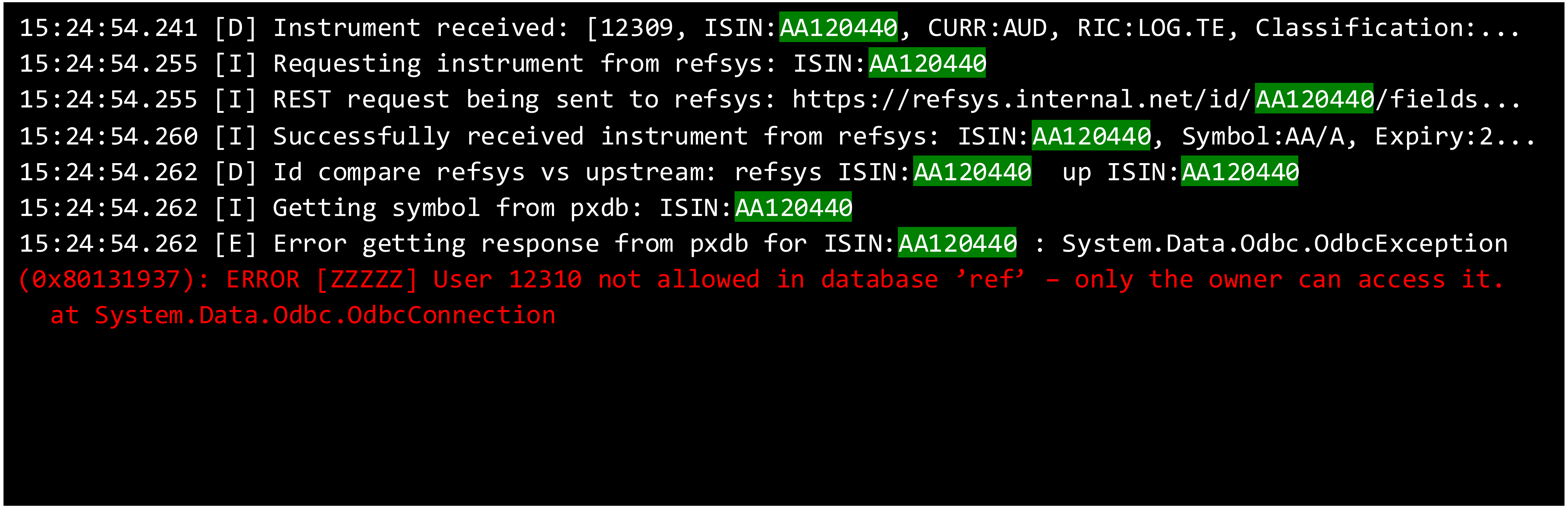
```
15:24:54.240 [D] Instrument received: [12309, ISIN:AA120440, CURR:AUD, RIC:LOG.TC, Classification:...
15:24:54.240 [D] Instrument received: [12309, ISIN:AB123414, CURR:AUD, RIC:LOG.TO, Classification:...
15:24:54.241 [D] Instrument received: [12309, ISIN:AC131441, CURR:AUD, RIC:LOG.TR, Classification:...
15:24:54.241 [D] Instrument received: [12309, ISIN:AA120440, CURR:AUD, RIC:LOG.TE, Classification:...
15:24:54.242 [D] Instrument received: [12309, ISIN:AA120110, CURR:AUD, RIC:LOG.TL, Classification:...
15:24:54.244 [D] Instrument received: [12309, ISIN:A1234ff0, CURR:AUD, RIC:LOG.TA, Classification:...
15:24:54.245 [D] Instrument received: [12309, ISIN:CC121310, CURR:AUD, RIC:LOG.TT, Classification:...
15:24:54.245 [D] Instrument received: [12309, ISIN:AA155550, CURR:AUD, RIC:LGG.TE, Classification:...
15:24:54.246 [I] Requesting instrument from refsys: ISIN: AB123414
15:24:54.247 [I] REST request being sent to refsys: https://refsys.internal.net/id/AB123414/fields...
15:24:54.248 [I] Successfully received instrument from refsys: ISIN:AB123414, Symbol:AA/B, Expiry:2...
15:24:54.249 [D] Id compare refsys vs upstream: refsys ISIN:AB123414  up ISIN:AB123414
15:24:54.250 [I] Getting symbol from pxdb: ISIN:AB123414
15:24:54.251 [I] Requesting instrument from refsys: ISIN:AC131441
15:24:54.252 [I] REST request being sent to refsys: https://refsys.internal.net/id/ AC131441 /fields...
15:24:54.254 [I] Successfully received instrument from refsys: ISIN:AC131441, Symbol:AA/A, Expiry:2...
15:24:54.255 [D] Id compare refsys vs upstream: refsys ISIN:AC131441  up ISIN:AC131441
15:24:54.255 [I] Getting symbol from pxdb: ISIN:AC131441
15:24:54.255 [I] Requesting instrument from refsys: ISIN:AA120440
15:24:54.255 [I] REST request being sent to refsys: https://refsys.internal.net/id/AA120440/fields...
15:24:54.260 [I] Successfully received instrument from refsys: ISIN:AA120440, Symbol:AA/A, Expiry:2...
15:24:54.262 [D] Id compare refsys vs upstream: refsys ISIN:AA120440  up ISIN:AA120440
15:24:54.262 [I] Getting symbol from pxdb: ISIN:AA120440
15:24:54.262 [E] Error getting response from pxdb for ISIN:AA120440: System.Data.Odbc.OdbcException (0x80131937): ERROR [ZZZZZ] User 12310 not allowed in database
’ref’ – only the owner can access it.
  at System.Data.Odbc.OdbcConnection
```

Business id example



```
15:24:54.240 [D] Instrument received: [12309, ISIN:AA120440, CURR:AUD, RIC:LOG.TC, Classification:...
15:24:54.240 [D] Instrument received: [12309, ISIN:AB123414, CURR:AUD, RIC:LOG.TO, Classification:...
15:24:54.241 [D] Instrument received: [12309, ISIN:AC131441, CURR:AUD, RIC:LOG.TR, Classification:...
15:24:54.241 [D] Instrument received: [12309, ISIN:AA120440, CURR:AUD, RIC:LOG.TE, Classification:...
15:24:54.242 [D] Instrument received: [12309, ISIN:AA120110, CURR:AUD, RIC:LOG.TL, Classification:...
15:24:54.244 [D] Instrument received: [12309, ISIN:A1234ff0, CURR:AUD, RIC:LOG.TA, Classification:...
15:24:54.245 [D] Instrument received: [12309, ISIN:CC121310, CURR:AUD, RIC:LOG.TT, Classification:...
15:24:54.245 [D] Instrument received: [12309, ISIN:AA155550, CURR:AUD, RIC:LGG.TE, Classification:...
15:24:54.246 [I] Requesting instrument from refsys: ISIN: AB123414
15:24:54.247 [I] REST request being sent to refsys: https://refsys.internal.net/id/AB123414/fields...
15:24:54.248 [I] Successfully received instrument from refsys: ISIN:AB123414, Symbol:AA/B, Expiry:2...
15:24:54.249 [D] Id compare refsys vs upstream: refsys ISIN:AB123414  up ISIN:AB123414
15:24:54.250 [I] Getting symbol from pxdb: ISIN:AB123414
15:24:54.251 [I] Requesting instrument from refsys: ISIN:AC131441
15:24:54.252 [I] REST request being sent to refsys: https://refsys.internal.net/id/ AC131441 /fields...
15:24:54.254 [I] Successfully received instrument from refsys: ISIN:AC131441, Symbol:AA/A, Expiry:2...
15:24:54.255 [D] Id compare refsys vs upstream: refsys ISIN:AC131441  up ISIN:AC131441
15:24:54.255 [I] Getting symbol from pxdb: ISIN:AC131441
15:24:54.255 [I] Requesting instrument from refsys: ISIN:AA120440
15:24:54.255 [I] REST request being sent to refsys: https://refsys.internal.net/id/AA120440/fields...
15:24:54.260 [I] Successfully received instrument from refsys: ISIN:AA120440, Symbol:AA/A, Expiry:2...
15:24:54.262 [D] Id compare refsys vs upstream: refsys ISIN:AA120440  up ISIN:AA120440
15:24:54.262 [I] Getting symbol from pxdb: ISIN:AA120440
15:24:54.262 [E] Error getting response from pxdb for ISIN:AA120440: System.Data.Odbc.OdbcException (0x80131937): ERROR [ZZZZZ] User 12310 not allowed in database
’ref’ – only the owner can access it.
  at System.Data.Odbc.OdbcConnection
```

Business id example



```
15:24:54.241 [D] Instrument received: [12309, ISIN:AA120440, CURR:AUD, RIC:LOG.TE, Classification:...
15:24:54.255 [I] Requesting instrument from refsys: ISIN:AA120440
15:24:54.255 [I] REST request being sent to refsys: https://refsys.internal.net/id/AA120440/fields...
15:24:54.260 [I] Successfully received instrument from refsys: ISIN:AA120440, Symbol:AA/A, Expiry:2...
15:24:54.262 [D] Id compare refsys vs upstream: refsys ISIN:AA120440  up ISIN:AA120440
15:24:54.262 [I] Getting symbol from pxdb: ISIN:AA120440
15:24:54.262 [E] Error getting response from pxdb for ISIN:AA120440 : System.Data.Odbc.OdbcException
(0x80131937): ERROR [ZZZZZ] User 12310 not allowed in database ’ref’ – only the owner can access it.
  at System.Data.Odbc.OdbcConnection
```

Correlation – incoming/outgoing request

Advice 13: Use explicit id for tracking each log entry for incoming API request.

- Request received.
- Request processing log entries.
- Request response sent.

Advice 14: Use explicit id for tracking each log entry for outgoing request.

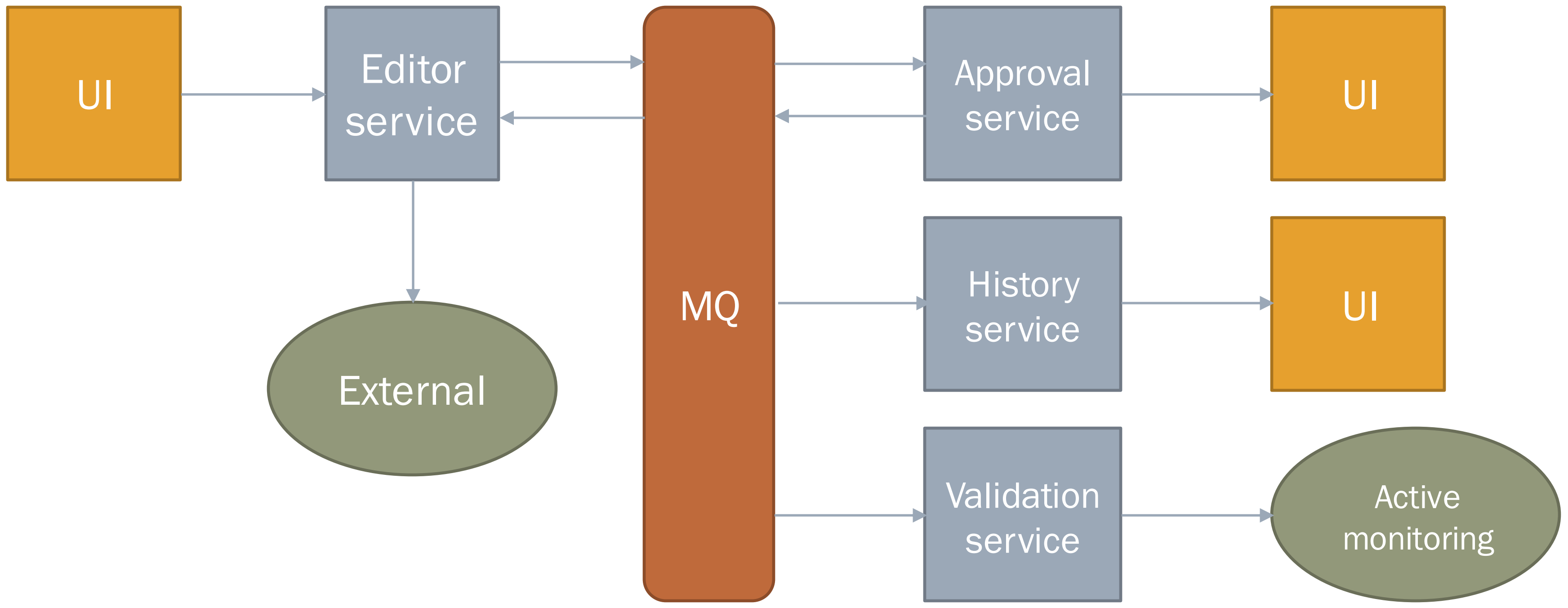
- Request sent.
- Request response received.
- ‚Outgoing request’ is database usage, HTTP calls, file operations, library usage, long lasting operation, etc. Follow in/out concept.

Correlation – microservices

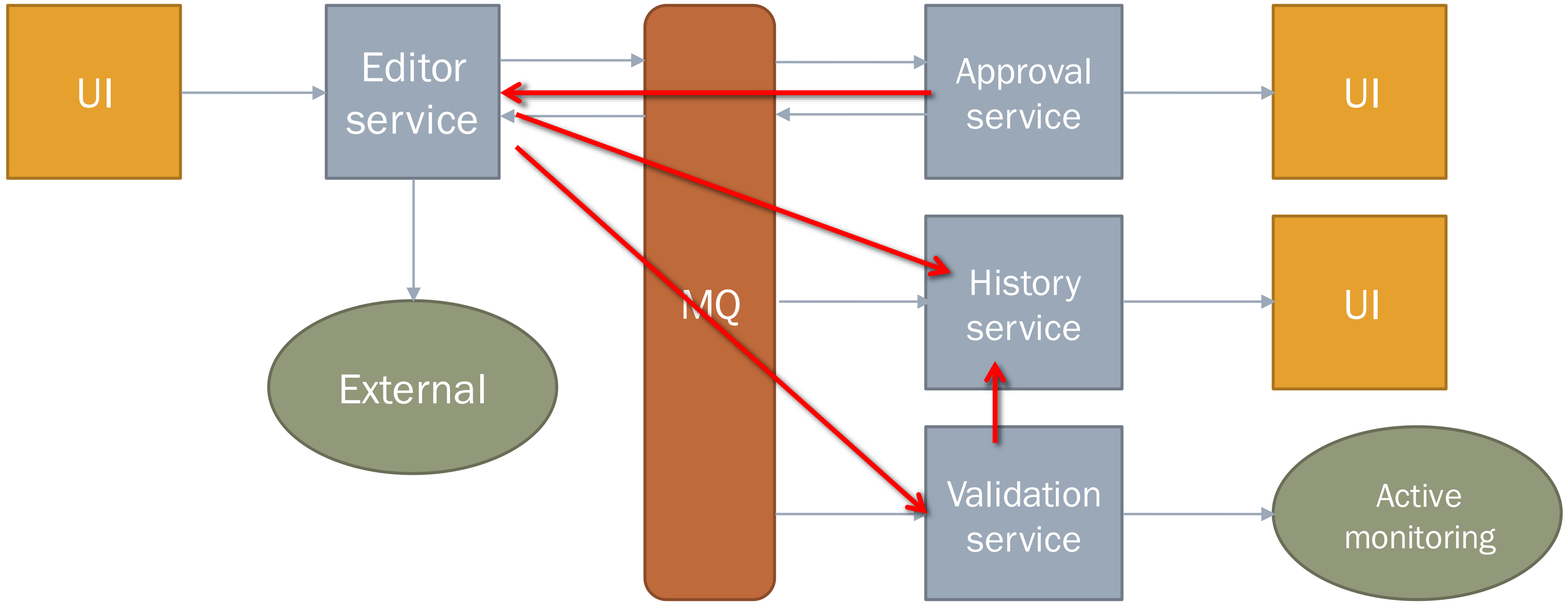
Advise 15: Use explicit id for tracking of a single context accross multiple services.

- Log in each service using the same id.
  - It most likely becomes a column/member and might be persisted.
  - If multiple teams are involved make sure all of them agree on logging approach.
  - Make sure logs go into single aggregation for seach purposes.
- This can correlate operations that are quick but also long lasting. An entire informative path evidencing all steps is useful.
- Aggregated log search will show entire tracking.

Correlation - microservices



Correlation - microservices



```
editor    Sending audit message requestId="8f88cea7", data: operationType="FI", isinCode="OS2492929399"...
editor    RabbitMQ successful sending message with 'queue history' corellationId: 15e4cf87, message:
              requestId="8f88cea7", msg:{data: operationType="FI", isinCode="OS2492929399"...
editor    Sending validation job message, requestId="8f88cea7",
              validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
approval  Processing approval message 8f88cea7 successfuly finished.
editor    RabbitMQ successful sending message with 'queue validation' corellationId: 1f2de92f,
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
editor    Processing message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: bf20fa4b, message:
              requestId="8f88cea7", msg:{data: operationType="FI", isinCode="OS2492929399"...
audit     Processing audit message successfuly finished for requestId="8f88cea7"
validator RabbitMQ successfuly deserialized message from 'queue validation' correlationId: 648a97d4, message:
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
validator Sending audit message requestId="8f88cea7", data: operationType="validation",
              text="Validation started for...
validator RabbitMQ successful sending message with 'queue history' corellationId: 435a6cf2, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
validator Processing validation message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: 17c3b29a, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
...
validator RabbitMQ successful sending message with 'queue history' corellationId: 6c3fb0ad, message:
              requestId="8f88cea7", data: operationType="validation", text="Completed...
```

```
editor    Sending audit message requestId="8f88cea7", data: operationType="FI", isinCode="OS2492929399"...
editor    RabbitMQ successful sending message with 'queue history' corellationId: 15e4cf87, message:
              requestId="8f88cea7", msg:{data: operationType="FI", isinCode="OS2492929399"...
editor    Sending validation job message, requestId="8f88cea7",
              validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
approval  Processing approval message 8f88cea7 successfuly finished.
editor    RabbitMQ successful sending message with 'queue validation' corellationId: 1f2de92f,
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
editor    Processing message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: bf20fa4b, message:
              requestId="8f88cea7", msg:{data: operationType="FI", isinCode="OS2492929399"...
audit     Processing audit message successfuly finished for requestId="8f88cea7"
validator RabbitMQ successfuly deserialized message from 'queue validation' correlationId: 648a97d4, message:
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
validator Sending audit message requestId="8f88cea7", data: operationType="validation",
              text="Validation started for...
validator RabbitMQ successful sending message with 'queue history' corellationId: 435a6cf2, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
validator Processing validation message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: 17c3b29a, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
...
validator RabbitMQ successful sending message with 'queue history' corellationId: 6c3fb0ad, message:
              requestId="8f88cea7", data: operationType="validation", text="Completed...
```

```
editor    Sending audit message requestId="8f88cea7", data: operationType="FI", isinCode="OS2492929399"...
editor    RabbitMQ successful sending message with 'queue history' corellationId: 15e4cf87, message:
              requestId="8f88cea7", msg:{data: operationType="FI", isinCode="OS2492929399"...
editor    Sending validation job message, requestId="8f88cea7",
              validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
approval  Processing approval message 8f88cea7 successfuly finished.
editor    RabbitMQ successful sending message with 'queue validation' corellationId: 1f2de92f,
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
editor    Processing message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: bf20fa4b, message:
              requestId="8f88cea7", msg:{data: operationType="FI", isinCode="OS2492929399"...
audit     Processing audit message successfuly finished for requestId="8f88cea7"
validator RabbitMQ successfuly deserialized message from 'queue validation' correlationId: 648a97d4, message:
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
validator Sending audit message requestId="8f88cea7", data: operationType="validation",
              text="Validation started for...
validator RabbitMQ successful sending message with 'queue history' corellationId: 435a6cf2, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
validator Processing validation message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: 17c3b29a, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
...
validator RabbitMQ successful sending message with 'queue history' corellationId: 6c3fb0ad, message:
              requestId="8f88cea7", data: operationType="validation", text="Completed...
```

```
editor    Sending audit message requestId="8f88cea7", data: operationType="FI", isinCode="OS2492929399"...
editor    RabbitMQ successful sending message with 'queue history' corellationId: 15e4cf87, message:
              requestId="8f88cea7", msg:{data: operationType="FI", isinCode="OS2492929399"...
editor    Sending validation job message, requestId="8f88cea7",
              validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
approval  Processing approval message 8f88cea7 successfuly finished.
editor    RabbitMQ successful sending message with 'queue validation' corellationId: 1f2de92f,
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
editor    Processing message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: bf20fa4b, message:
              requestId="8f88cea7", msg:{ata: operationType="FI", isinCode="OS2492929399"...
audit     Processing audit message successfuly finished for requestId="8f88cea7"
validator RabbitMQ successfuly deserialized message from 'queue validation' correlationId: 648a97d4, message:
              requestId="8f88cea7", validationDefinitions=[{fieldName="conversion",expectedValue=0.1,jPath=...}]
validator Sending audit message requestId="8f88cea7", data: operationType="validation",
              text="Validation started for...
validator RabbitMQ successful sending message with 'queue history' corellationId: 435a6cf2, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
validator Processing validation message successfuly finished for requestId="8f88cea7"
audit     RabbitMQ successfuly deserialized message from 'queue history' correlationId: 17c3b29a, message:
              requestId="8f88cea7", data: operationType="validation", text="Validation started for...
...
validator RabbitMQ successful sending message with 'queue history' corellationId: 6c3fb0ad, message:
              requestId="8f88cea7", data: operationType="validation", text="Completed...
```

Correlation – other services (3rd party)

Advise 16: Use unique id for tracking of a single context in conjuction with other services.

- Other service API might support it.
  - Headers are good place to provide metadata like this.
- Get agreement on using it on both sides.
- Invaluable cases when debugging:
  - Correlating logs by time might be difficult because of lag, logged time differences, framework induced delays.
  - Not enough business information might be logged to correlate. Correlation id is unique per operation/context.
  - Same requests sent multiple times will differ by correlation id.
  - Other side can rarely share detailed private information unless it's the same organisation.

Summary

2 concepts and 16 advises

It was a great presentation
CHANGE MY MIND