XFS4IoT SP-Dev Workgroup

2nd September 2025

XFS4IoT SP-Dev Workgroup agenda

- Recap from previous meeting
- Ahead of Time compilation PoC
- OpenTelemetry PoC
- What’s next?
- Next meeting

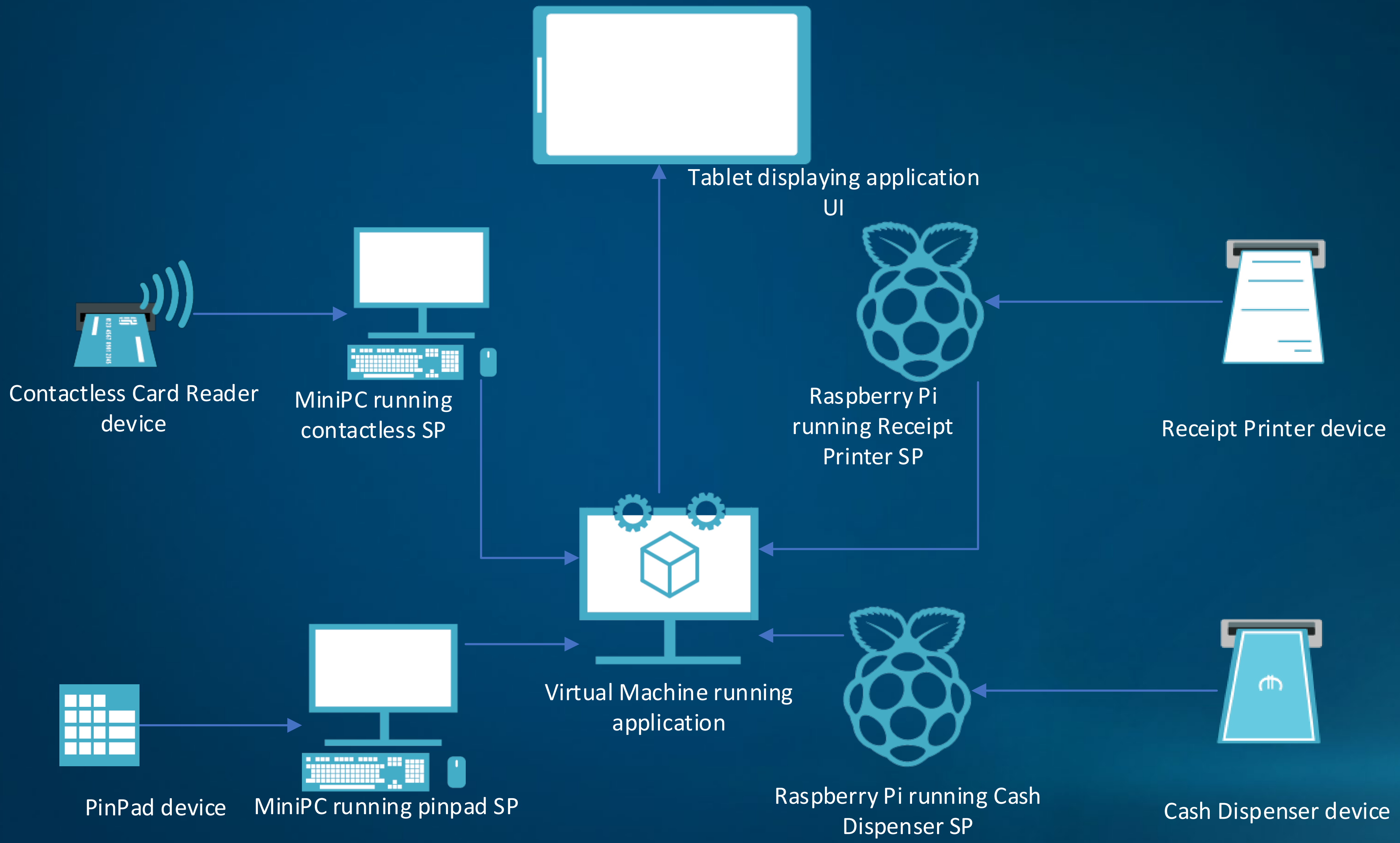
ATM Software

Recap from previous meeting

Recap from previous meeting

- Recap of previous hardware demos over the years
- XFS4IoT: different architectures
- Hybrid system big demo

Hybrid demo architecture


Tablet displaying application UI
Contactless Card Reader device
MiniPC running contactless SP
Raspberry Pi running Receipt Printer SP
Receipt Printer device
Virtual Machine running application
PinPad device
MiniPC running pinpad SP
Raspberry Pi running Cash Dispenser SP
Cash Dispenser device

ATM Software

Ahead of Time compilation

Ahead of Time

Aims from April 2025 (41st) meeting

- Remove reflection – use code generators
- Use “Ahead of Time” (AoT) compilation
- Use small footprint hardware…

Ahead of Time

Aims from April 2025 (41st) meeting

- Remove reflection – use code generators ✅
- Use "Ahead of Time" (AoT) compilation
- Use small footprint hardware…

Ahead of Time

Aims from April 2025 (41st) meeting

- Remove reflection – use code generators ✅
- Use "Ahead of Time" (AoT) compilation ✅
- Use small footprint hardware…

Ahead of Time

Aims from April 2025 (41st) meeting

- Remove reflection – use code generators ✅
- Use "Ahead of Time" (AoT) compilation ✅
- Use small footprint hardware… ❌

Ahead of Time – remove reflection

“Reflection” means dynamically creating code at run time. Not supported with AoT.

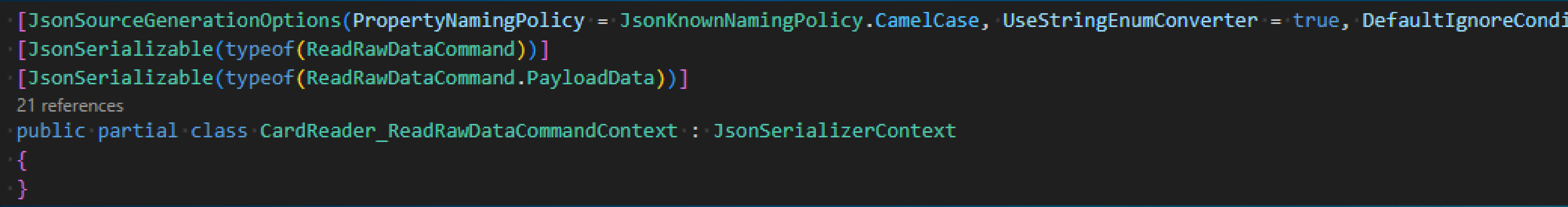
Instead, create all required code during development - code, or at compile time.

- Serialization - use new "Source Generator" support in .NET core
- Command handlers - update KAL code generator

Ahead of Time – Serialization

Mostly very simple to enable:

- Reference System.Text.Json.Serialization package and namespace
- Add a 'JsonSerializerContext partial class for each message – The .NET source generator then fills in the details



```
[JsonSourceGenerationOptions(PropertyNamingPolicy = JsonKnownNamingPolicy.CamelCase, UseStringEnumConverter = true, DefaultIgnoreCondi
[JsonSerializable(typeof(ReadRawDataCommand))]
[JsonSerializable(typeof(ReadRawDataCommand.PayloadData))]
21 references
public partial class CardReader_ReadRawDataCommandContext : JsonSerializerContext
{
}
```

Ahead of Time – Serialization

Problems:

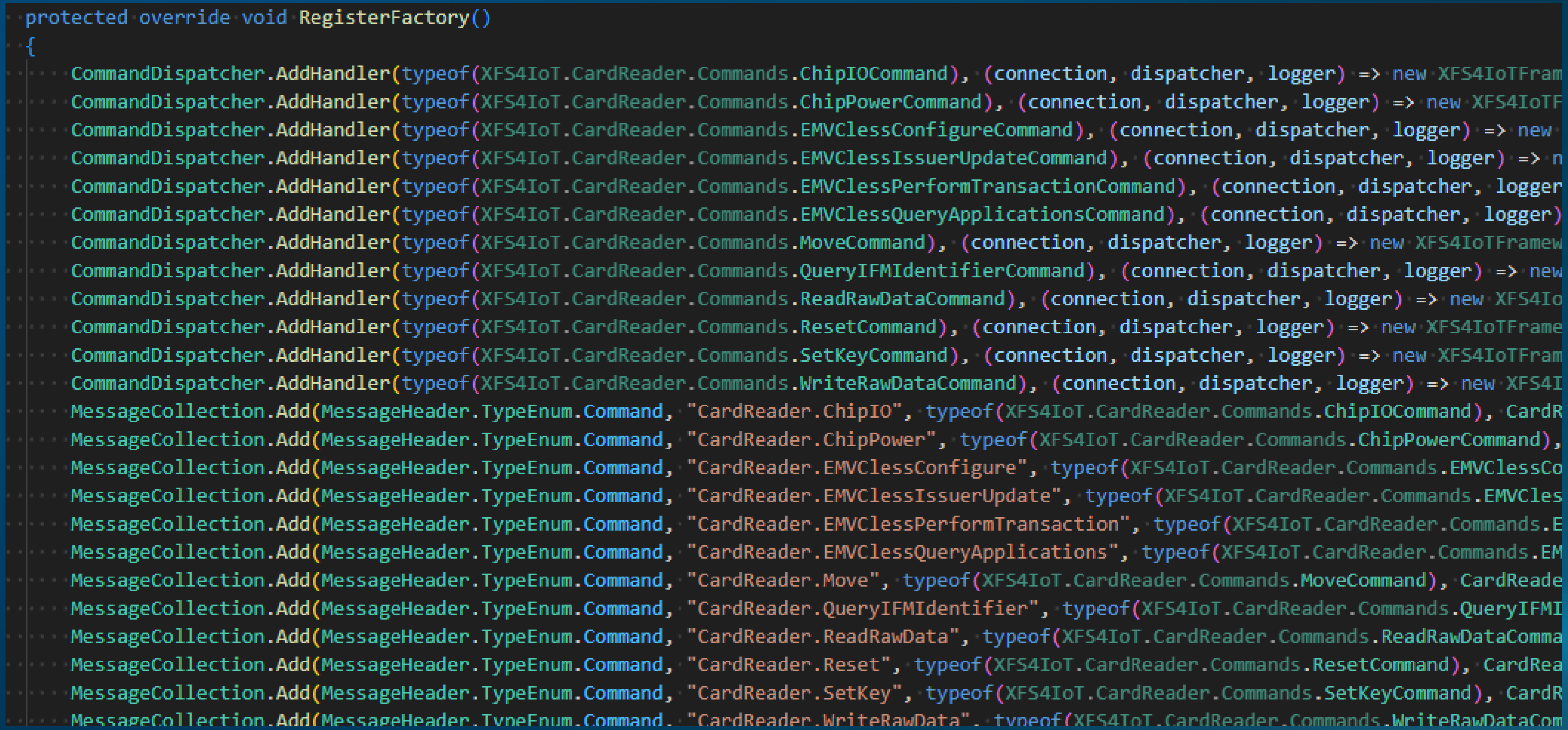
- We need custom support for Base64 encoded fields. We can handle that with a "Base64Converter" class, and add an attribute to each relevant field
- Namespace problems… https://github.com/dotnet/runtime/issues/72671

For now, delete everything except the card reader and rename relevant enums.

Ahead of Time – Command handlers

Much of the SP Dev framework is automatically generated from the YAML definition of the XFS4IoT specification created by the XFS committee, by a KAL utility.

We can update this utility to automatically create information on the supported commands:

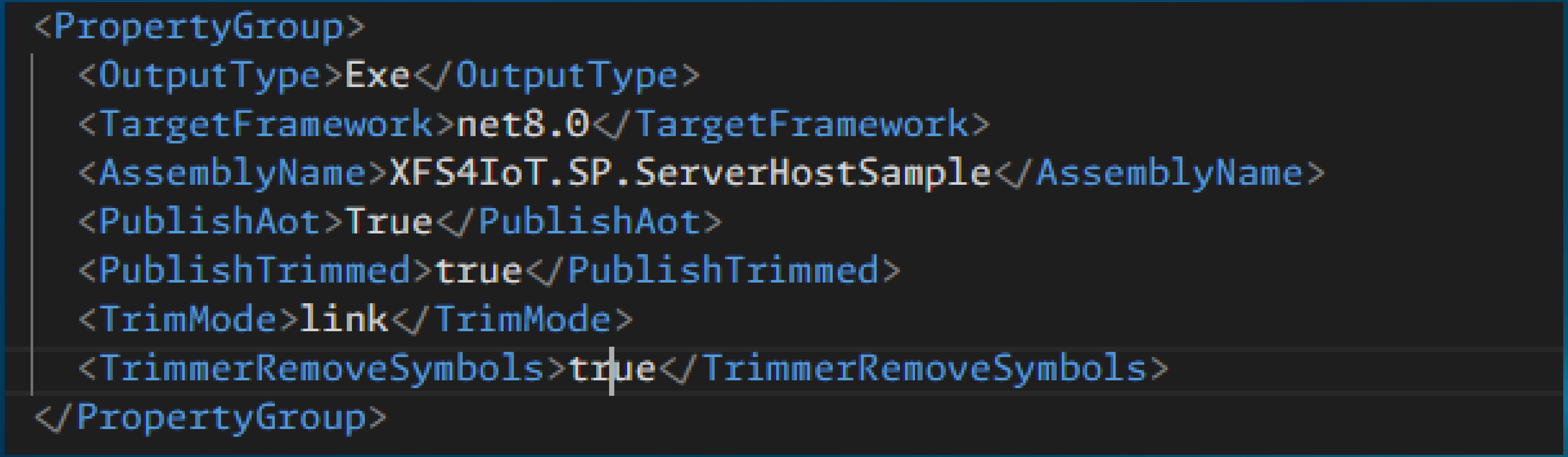


```
protected override void RegisterFactory()
{
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.ChipIOCommand), (connection, dispatcher, logger) => new XFS4IoTFram
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.ChipPowerCommand), (connection, dispatcher, logger) => new XFS4IoTF
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.EMVClessConfigureCommand), (connection, dispatcher, logger) => new
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.EMVClessIssuerUpdateCommand), (connection, dispatcher, logger) => n
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.EMVClessPerformTransactionCommand), (connection, dispatcher, logger
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.EMVClessQueryApplicationsCommand), (connection, dispatcher, logger)
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.MoveCommand), (connection, dispatcher, logger) => new XFS4IoTFramew
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.QueryIFMIdentifierCommand), (connection, dispatcher, logger) => new
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.ReadRawDataCommand), (connection, dispatcher, logger) => new XFS4Io
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.ResetCommand), (connection, dispatcher, logger) => new XFS4IoTFrame
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.SetKeyCommand), (connection, dispatcher, logger) => new XFS4IoTFram
    CommandDispatcher.AddHandler(typeof(XFS4IoT.CardReader.Commands.WriteRawDataCommand), (connection, dispatcher, logger) => new XFS4I
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.ChipIO", typeof(XFS4IoT.CardReader.Commands.ChipIOCommand), CardR
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.ChipPower", typeof(XFS4IoT.CardReader.Commands.ChipPowerCommand),
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.EMVClessConfigure", typeof(XFS4IoT.CardReader.Commands.EMVClessCo
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.EMVClessIssuerUpdate", typeof(XFS4IoT.CardReader.Commands.EMVCles
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.EMVClessPerformTransaction", typeof(XFS4IoT.CardReader.Commands.E
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.EMVClessQueryApplications", typeof(XFS4IoT.CardReader.Commands.EM
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.Move", typeof(XFS4IoT.CardReader.Commands.MoveCommand), CardReade
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.QueryIFMIdentifier", typeof(XFS4IoT.CardReader.Commands.QueryIFMI
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.ReadRawData", typeof(XFS4IoT.CardReader.Commands.ReadRawDataComma
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.Reset", typeof(XFS4IoT.CardReader.Commands.ResetCommand), CardRea
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.SetKey", typeof(XFS4IoT.CardReader.Commands.SetKeyCommand), CardR
    MessageCollection.Add(MessageHeader.TypeEnum.Command, "CardReader.WriteRawData", typeof(XFS4IoT.CardReader.Commands.WriteRawDataCom
```

Ahead of Time – Ahead of Time compilation

Now enable ‘AoT’

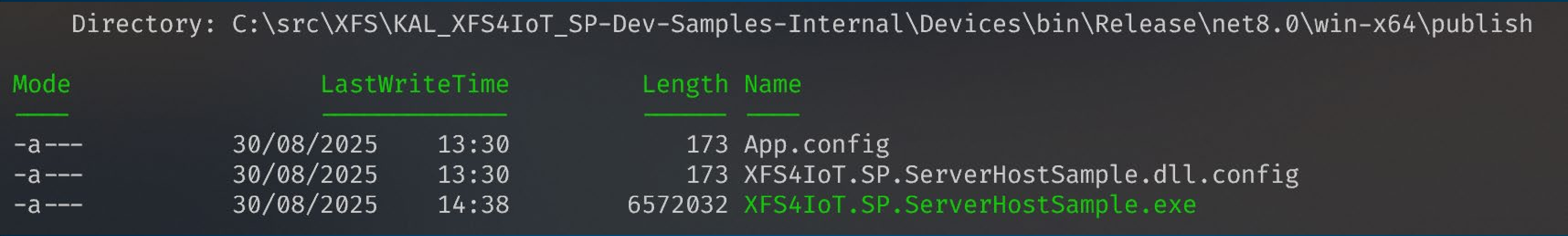
- Already using .NET 8
- Set required build project flags
- ‘Publish' project to get a single .exe



```
<PropertyGroup>
  <OutputType>Exe</OutputType>
  <TargetFramework>net8.0</TargetFramework>
  <AssemblyName>XFS4IoT.SP.ServerHostSample</AssemblyName>
  <PublishAot>True</PublishAot>
  <PublishTrimmed>true</PublishTrimmed>
  <TrimMode>link</TrimMode>
  <TrimmerRemoveSymbols>true</TrimmerRemoveSymbols>
</PropertyGroup>
```

Results – File size

Single file, ~7MB, including *all* dependencies

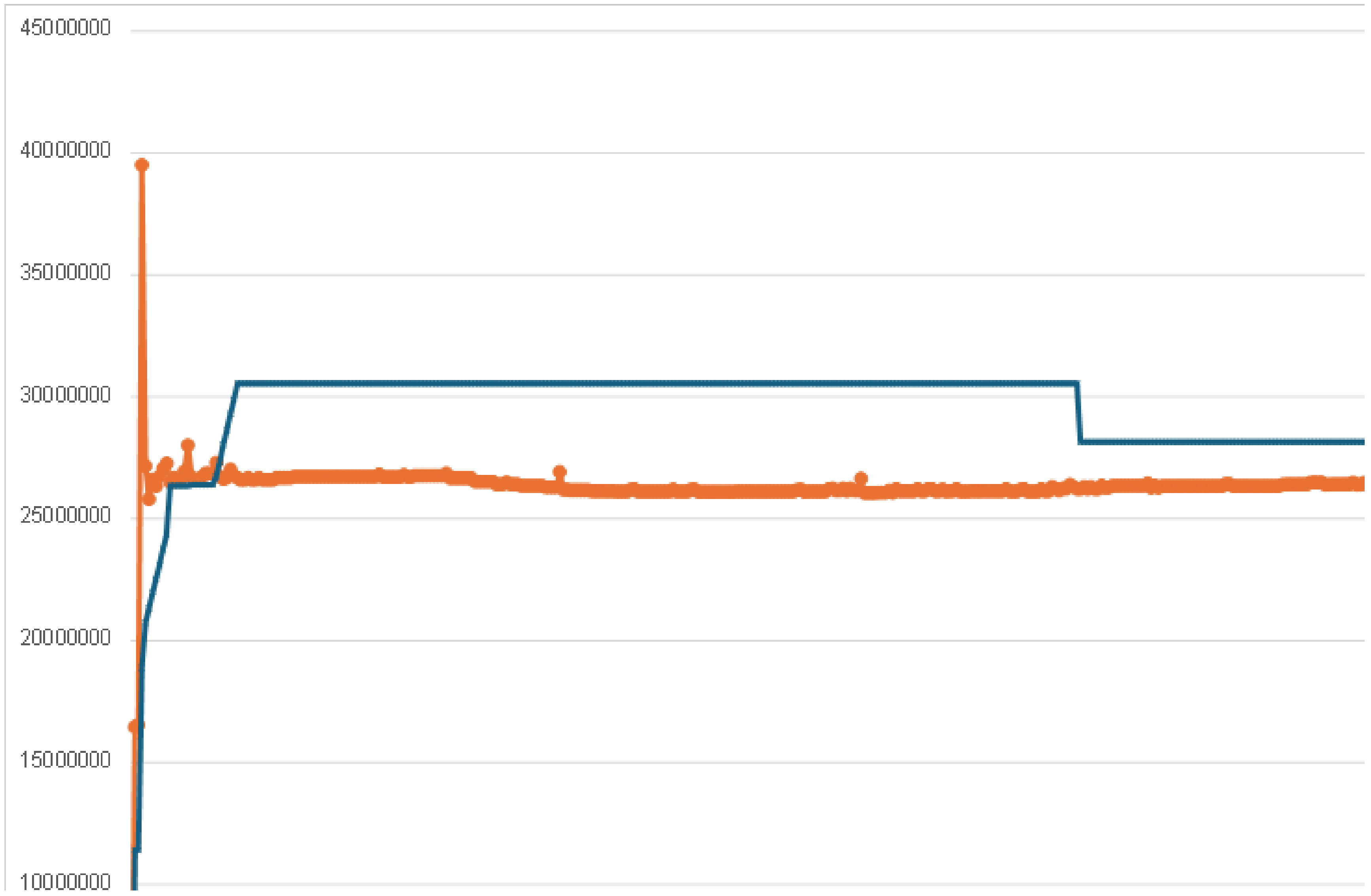


```
    Directory: C:\src\XFS\KAL_XFS4IoT_SP-Dev-Samples-Internal\Devices\bin\Release\net8.0\win-x64\publish

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          30/08/2025    13:30            173 App.config
-a---          30/08/2025    13:30            173 XFS4IoT.SP.ServerHostSample.dll.config
-a---          30/08/2025    14:38        6572032 XFS4IoT.SP.ServerHostSample.exe
```

Results – Memory

~30MB of
runtime memory



Next steps

- More performance testing
- More performance tuning
- Look for fixes to the serialization namespace problem

- **Embedded support with .NET nanoFramework**

ATM Software

OpenTelemetry PoC

OpenTelemetry

- Debug logging and tracing are important

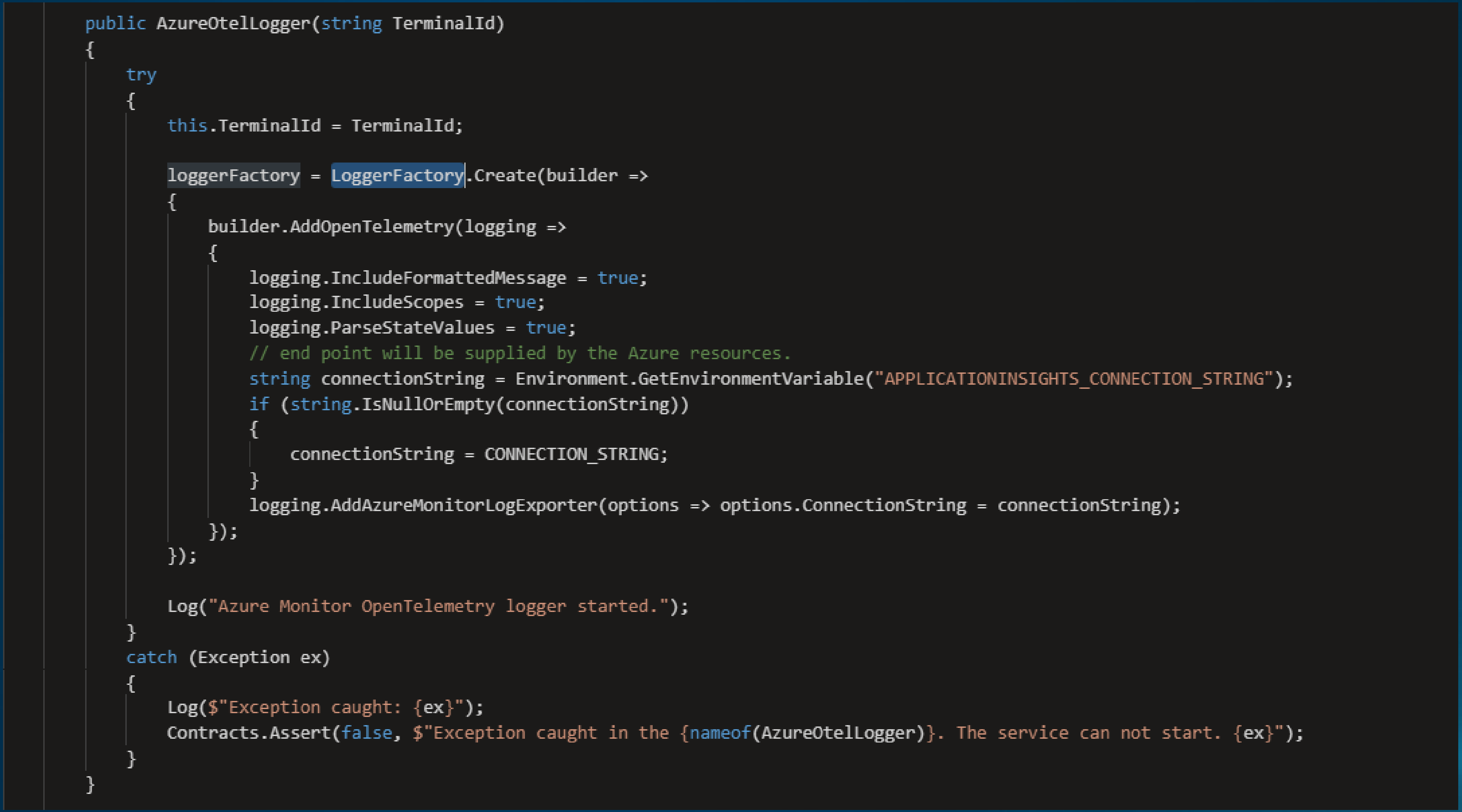
- XFS4IoT supports distributed applications – cloud-based, multi-machine, etc. – useful to support cross machine logging
- OpenTelemetry standard enables logging from multiple clients to a central location. For example, a client application and SP-Dev SP can update the same log
- Also useful where machine has no local storage
- Supported by 'ILogger' interface in SP-Dev framework
- Current default is console logging

OpenTelemetry with Azure

For a PoC, we used the "Azure Application Insight" service, which supports OpenTelemetry. This is very simple to do:

- Create an Application Insight resource in Azure. Get the connection string for the resource.
- Create a class that uses the Azure.Monitor.OpenTelemetry.Exporter and implements the ILogger interface. Use the connection string to make a secure connection to the service. (We store the string in an environment variable to avoid hard-coding it.)

OpenTelemetry - Code

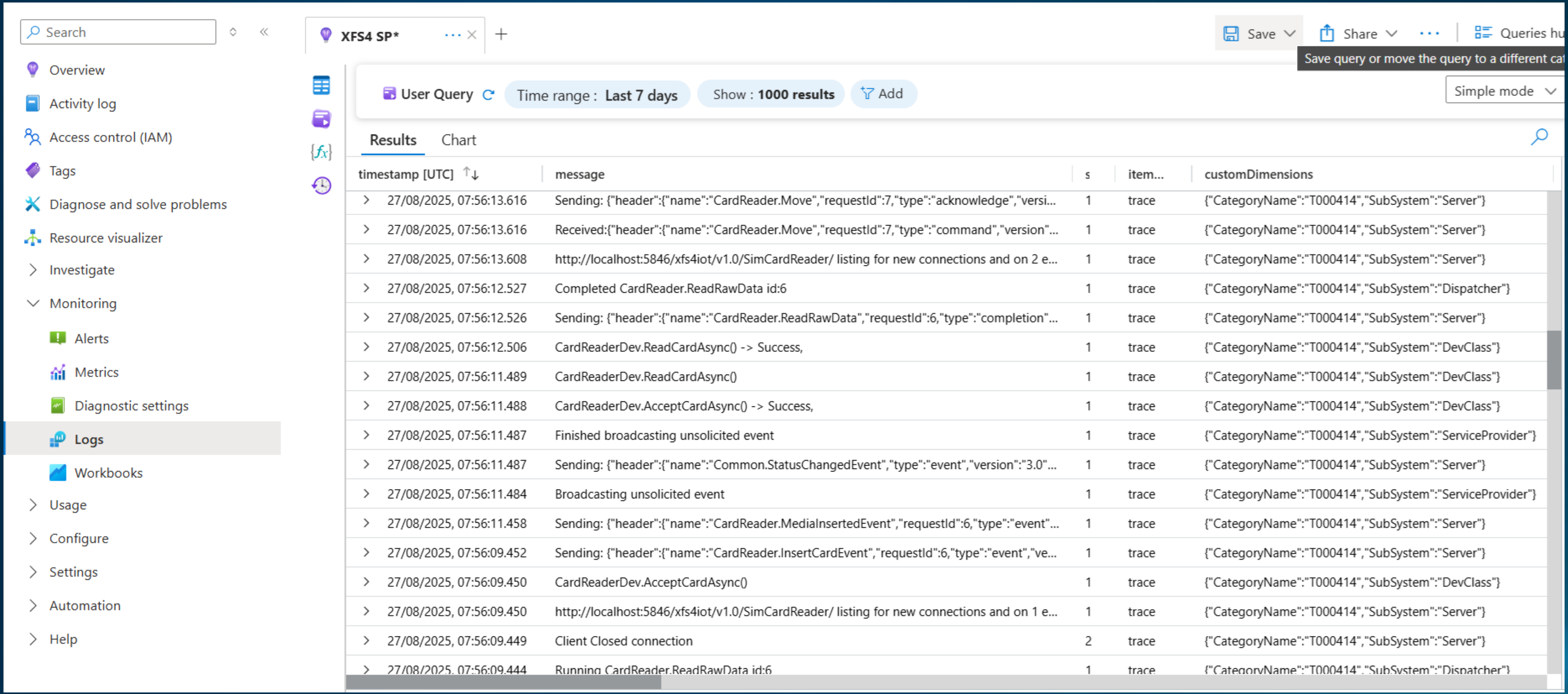


```
public AzureOtelLogger(string TerminalId)
{
    try
    {
        this.TerminalId = TerminalId;

        loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddOpenTelemetry(logging =>
            {
                logging.IncludeFormattedMessage = true;
                logging.IncludeScopes = true;
                logging.ParseStateValues = true;
                // end point will be supplied by the Azure resources.
                string connectionString = Environment.GetEnvironmentVariable("APPLICATIONINSIGHTS_CONNECTION_STRING");
                if (string.IsNullOrEmpty(connectionString))
                {
                    connectionString = CONNECTION_STRING;
                }
                logging.AddAzureMonitorLogExporter(options => options.ConnectionString = connectionString);
            });
        });

        Log("Azure Monitor OpenTelemetry logger started.");
    }
    catch (Exception ex)
    {
        Log($"Exception caught: {ex}");
        Contracts.Assert(false, $"Exception caught in the {nameof(AzureOtelLogger)}. The service can not start. {ex}");
    }
}
```

OpenTelemetry – Results



timestamp [UTC]	message	s	item...	customDimensions
27/08/2025, 07:56:13.616	Sending: {"header":{"name":"CardReader.Move","requestId":7,"type":"acknowledge","versi...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:13.616	Received:{"header":{"name":"CardReader.Move","requestId":7,"type":"command","version"...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:13.608	http://localhost:5846/xfs4iot/v1.0/SimCardReader/ listing for new connections and on 2 e...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:12.527	Completed CardReader.ReadRawData id:6	1	trace	{"CategoryName":"T000414","SubSystem":"Dispatcher"}
27/08/2025, 07:56:12.526	Sending: {"header":{"name":"CardReader.ReadRawData","requestId":6,"type":"completion"...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:12.506	CardReaderDev.ReadCardAsync() -> Success,	1	trace	{"CategoryName":"T000414","SubSystem":"DevClass"}
27/08/2025, 07:56:11.489	CardReaderDev.ReadCardAsync()	1	trace	{"CategoryName":"T000414","SubSystem":"DevClass"}
27/08/2025, 07:56:11.488	CardReaderDev.AcceptCardAsync() -> Success,	1	trace	{"CategoryName":"T000414","SubSystem":"DevClass"}
27/08/2025, 07:56:11.487	Finished broadcasting unsolicited event	1	trace	{"CategoryName":"T000414","SubSystem":"ServiceProvider"}
27/08/2025, 07:56:11.487	Sending: {"header":{"name":"Common.StatusChangedEvent","type":"event","version":"3.0"...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:11.484	Broadcasting unsolicited event	1	trace	{"CategoryName":"T000414","SubSystem":"ServiceProvider"}
27/08/2025, 07:56:11.458	Sending: {"header":{"name":"CardReader.MediaInsertedEvent","requestId":6,"type":"event"...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:09.452	Sending: {"header":{"name":"CardReader.InsertCardEvent","requestId":6,"type":"event","ve...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:09.450	CardReaderDev.AcceptCardAsync()	1	trace	{"CategoryName":"T000414","SubSystem":"DevClass"}
27/08/2025, 07:56:09.450	http://localhost:5846/xfs4iot/v1.0/SimCardReader/ listing for new connections and on 1 e...	1	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:09.449	Client Closed connection	2	trace	{"CategoryName":"T000414","SubSystem":"Server"}
27/08/2025, 07:56:09.444	Running CardReader.ReadRawData id:6	1	trace	{"CategoryName":"T000414","SubSystem":"Dispatcher"}

OpenTelemetry – Next steps

- Performance testing

Q: Is OpenTelemetry useful?

Is it interesting to make OpenTelemetry part of the XFS4IoT specification?

ATM Software

What's next?

What’s next?

- Quarterly meetings moving forward

- Framework updates and roadmap
- More PoC and experimental work
- Guest speakers
- DK specification
- More demos (biometrics and more)

Next call

Zoom

Meeting once every quarter
at 1300 UK time for 30 mins

Next call: 2nd December 2025
1300 UK, 0800 US EDT, 2200 Tokyo time

Calls are 30 mins long

We will continue to use Zoom
(Interpretation in Japanese, Chinese and Spanish is available using Zoom's interpretation feature)