

XFS4IoT SP-Dev Workgroup

3 August 2021

To add others from your company



Please email us at:

xfs4iot_sp-dev_info@kal.com

What have we done so far?

- Released support for:
 - Card reader
 - Cash dispenser
 - TTU
- Released sample code
- Demos with real hardware (videos on [YouTube](#))
- Experiments with various hardware including very small devices
- 7 meetings with great attendance

- Available **now** in "[KAL_XFS4IoT_SP-Dev](#)" repo
- All Dispenser commands are supported
- New End-to-End security feature is **not** yet included
- All framework code is available to:
 - Write test tools
 - Implement XFS4 SPs
 - Review
 - Test with our sample

- **We must not** agree to fix prices, coordinate bids, divide up territories, or agree not to compete.
- **We must not** discuss pricing strategy, terms and conditions of a deal, business strategy, deals won or lost, relationships with customers or partners, or share details of our company's technologies.
- **We must not** share competitively sensitive market information.
- **It is OK to** continue discussions on the development and adoption of XFS4IoT-based SPs.
- **We must** make access to workgroup products available on fair, reasonable and non-discriminatory terms.

Linux and XFS4IoT Containerization by Guest Speaker Daniel Schaefer of Red Hat

Containers for SP Developers

Daniel J. Schäfer

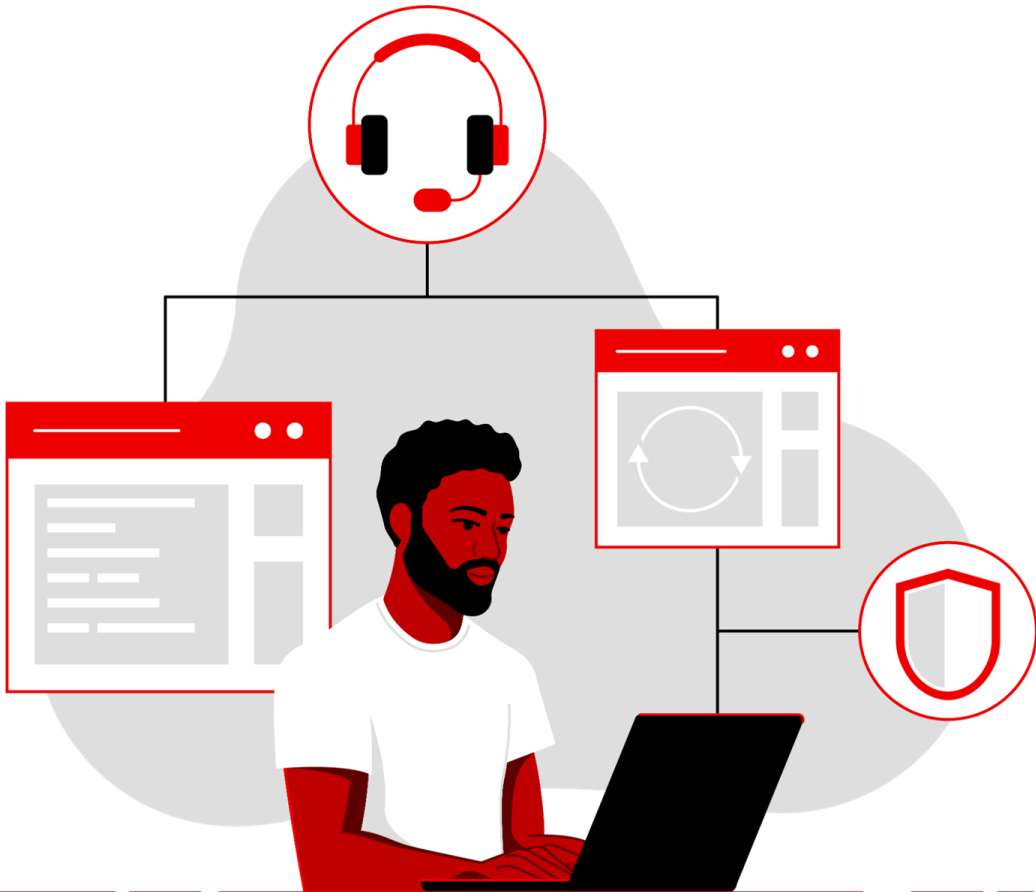
Senior Solutions Architect - EMEA Alliances, ISV

daniel.schaefer@redhat.com

Agenda

- ▶ (Enterprise) Linux as a secure OS alternative for ATMs
- ▶ Introduction to Linux Container Technology
- ▶ Walk-through demo: containerized SP example
- ▶ Benefits of containers for SP developers
- ▶ Outlook and Q&A

The value of a Red Hat subscription



Support and expertise



Security resources



Partner ecosystem



Lifecycle support and flexibility



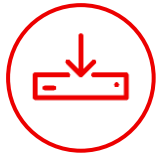
Product roadmaps



Proactive analytics

Red Hat Enterprise Linux Security Features

Secure foundation for running workloads in the open hybrid cloud



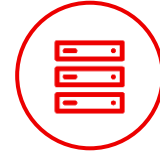
Mandatory Access Control with SELinux



Built-in compliance tools to meet security standards



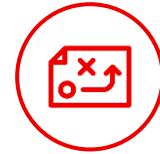
Automated patching and remediation without downtime



System-wide cryptography policies



Hardware root of trust support (TPM) and disk encryption



Application allowlisting prevents unwanted access

A consistent edge platform to meet your needs

Develop once,
deploy anywhere

Address diverse
use cases

Consistent
operations



Red Hat

Edge gateway/ edge
server



Small
bare metal
footprint



Infrastructure
virtualization



Public/private
cloud



Red Hat Enterprise Linux for edge

Ensured stability and deployment flexibility



Quick image generation

Easily create purpose-built OS images optimized for the architectural challenges inherent at edge.



Efficient over-the-air updates

Updates transfer significantly less data and are ideal for remote sites with limited or intermittent connectivity.



Remote device update mirroring

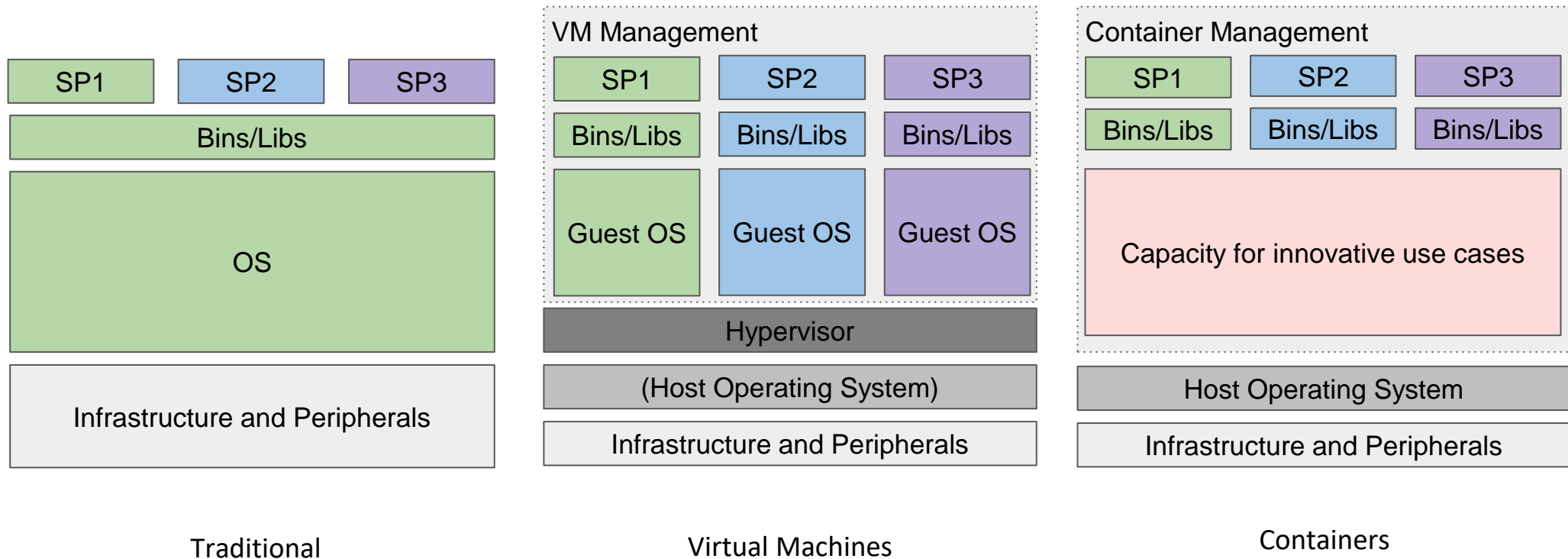
Staged and applied image updates occur at the next reboot or power cycle, ensuring minimal downtime.



Intelligent rollbacks

Application-specific health checks detect conflicts and automatically revert an OS update, preventing downtime.

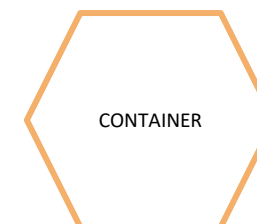
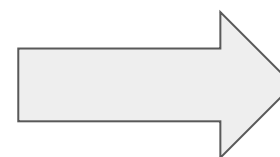
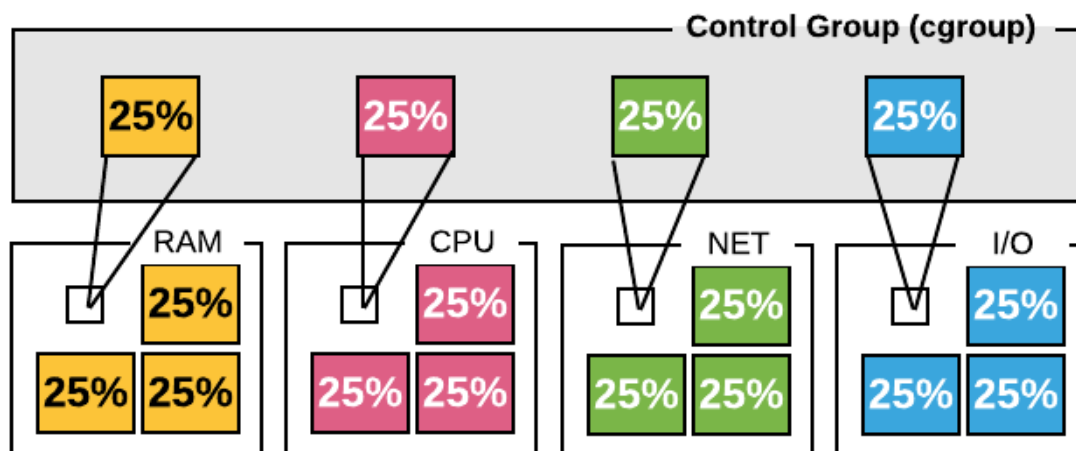
From Hosts to VM's to Containers



Container-Basics - Namespaces

A Linux kernel feature, not a container feature

- A container is a process!
- But in strong isolation!

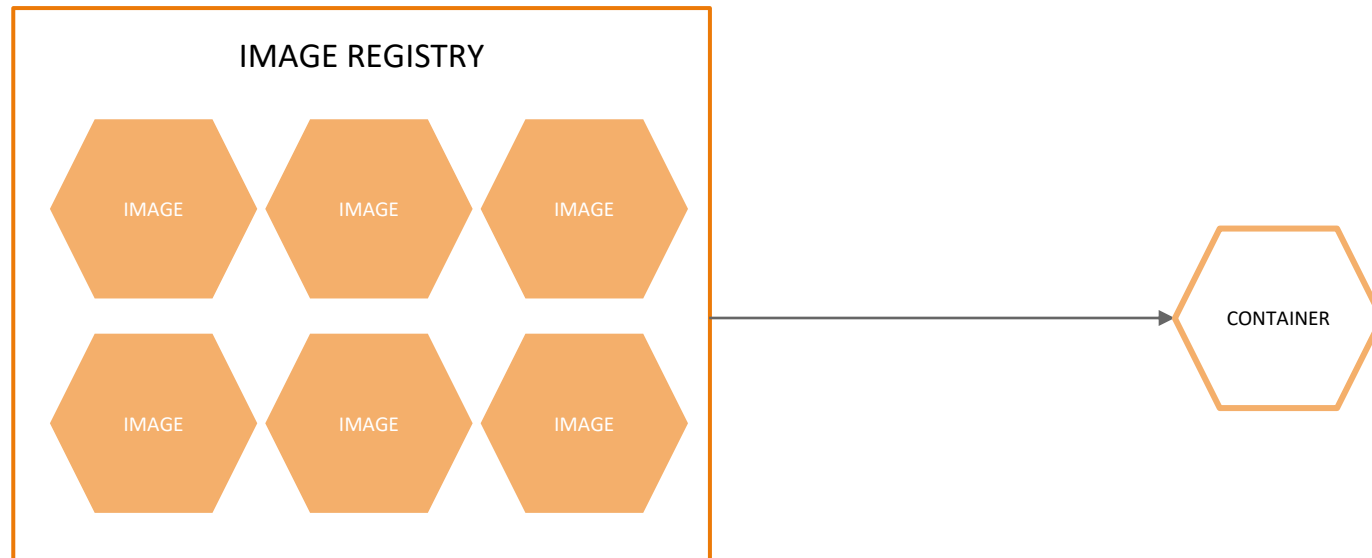


Container-Basics - Terminology

- Containers are the smallest compute unit. They are created from container images

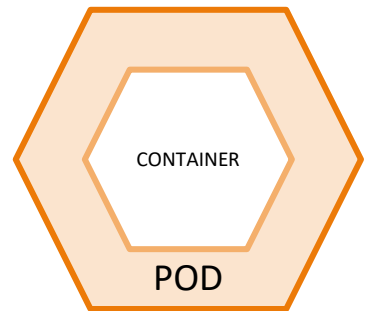


- Container images are stored in an image registry. Images have tags and version control.

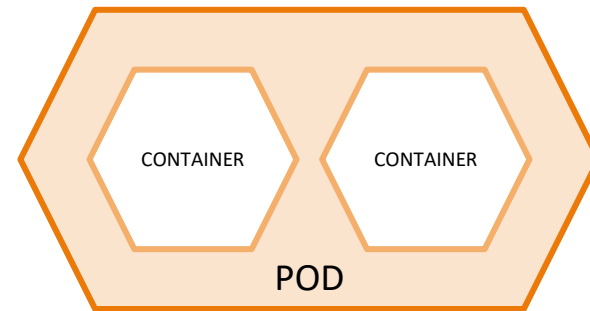


Container-Basics – POD's

- Containers are wrapped in pods which are units of deployment and management
- Containers within a pod can share resources (network, volumes, ...) - but are still derived from independent images



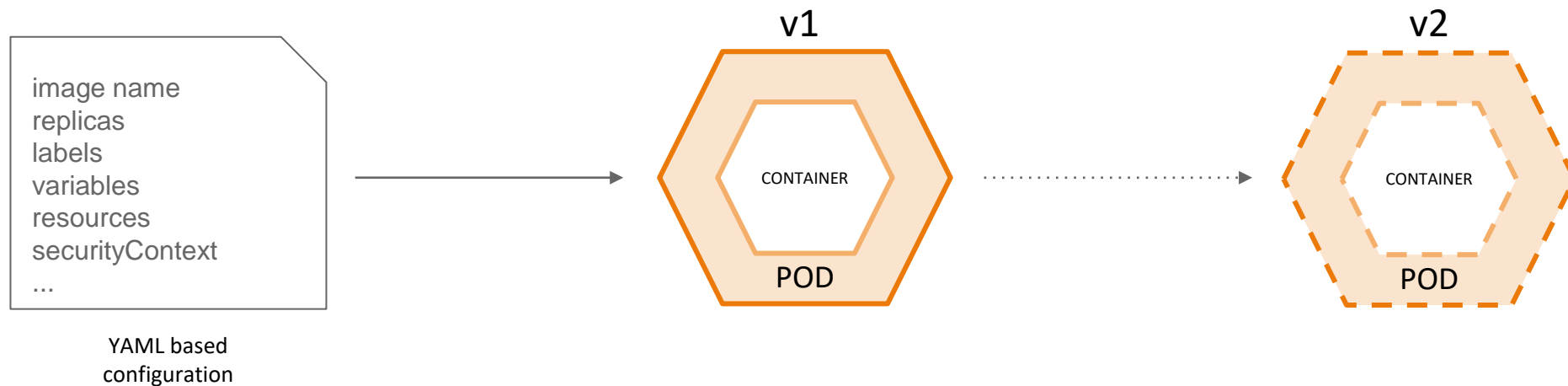
10.140.4.44



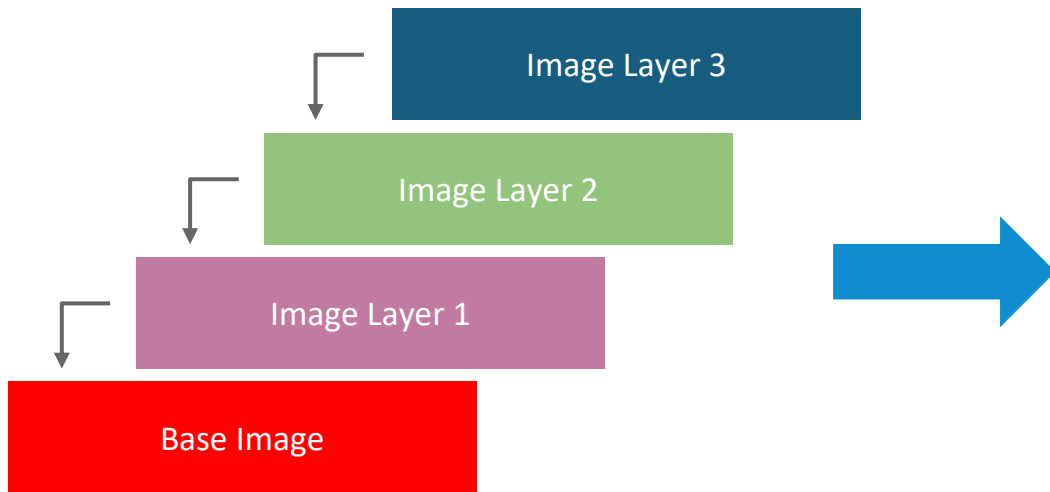
10.15.6.55

Container-Basics - YAML definitions

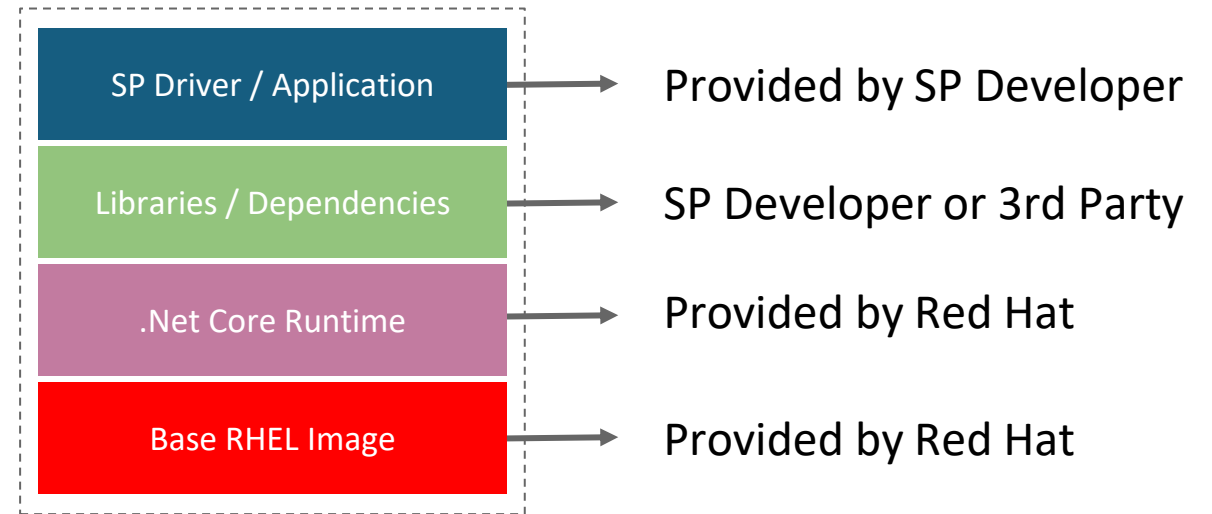
- Infrastructure-as-code concepts describe how to roll out Pods
- Standardize...
 - ... delivery formats across programming languages and technologies
 - ... language across functions: One format for Dev and Ops; for SW providers and operators
 - ... across environments: Developer workstation, bigger clusters in cloud environments, single edge server (ATM), and more



Container Images bundle all application dependencies

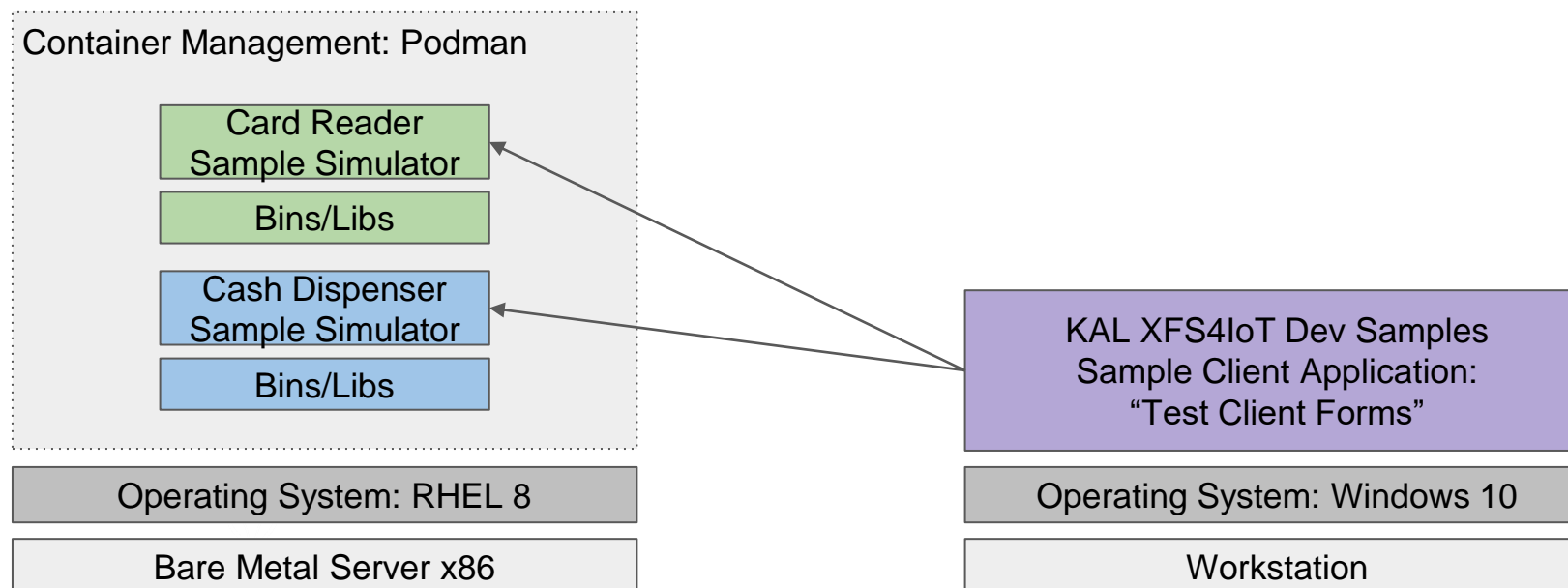


Container Image Layers



Example Container Image

Demo Environment Overview



2 independent Container Images (based on Red Hat Base Image with .Net 5 Runtime):

- KAL XFS4IoT Dev Samples for Cash Dispenser Simulator
- KAL XFS4IoT Dev Samples for Card Reader Simulator

Demo: Sample Dockerfile to build container image

```
# Based on Red Hat Universal Base Image for .Net 5.0. Image details at: https://catalog.redhat.com/software/containers/ubi8/dotnet-50/
FROM registry.access.redhat.com/ubi8/dotnet-50:latest

MAINTAINER Daniel Schaefer <daniel.schaefer@redhat.com>

# Here should be meaningful labels. At least "name", "vendor", "version", "release", "summary" and "description" are required for Red Hat image certification.
LABEL name="XFS4IoT_SP-Dev-Samples-Host" \
      vendor="KAL & Red Hat" \
      version="0.0.2-alpha.2" \
      release="SP-Dev 0.0.2" \
      summary="XFS4IoT server image with simulated card reader" \
      description="XFS4IoT server image with simulated card reader based on KAL XFS4IoT SP-Dev Framework samples."

USER root

# Update for latest security patches (certification recommendation)
RUN yum -y update-minimal --security --sec-severity=Important --sec-severity=Critical

# Include all license files (certification requirement)
COPY licenses /licenses

# Include application
COPY app /opt/app-root/app

# Exposed Ports
EXPOSE 5846-5856
EXPOSE 80
EXPOSE 443

# Good practice not to run as root
USER 1001

WORKDIR /opt/app-root/app

CMD ["dotnet", "XFS4IoT.SP.ServerHostSample.dll", "$@"]
```

Demo: Container Build Instructions

```
# cd cardreader-container-folder  
# podman build --layers=false --tag cardreader:0.0.2 .
```

```
[root@jenga container-CR]# podman build --layers=false --tag cardreader:0.0.2 .  
STEP 1: FROM registry.access.redhat.com/ubi8/dotnet-50:latest  
STEP 2: MAINTAINER Daniel Schaefer <daniel.schaefer@redhat.com>  
STEP 3: LABEL name="cardreader-simulator" vendor="KAL & Red Hat" version="0.0.2-alpha.2"  
release="SP-Dev 0.0.2" summary="XFS4IoT server image with simulated card reader" description="XFS4IoT server image with simulated card reader based on KAL XFS4IoT SP-Dev Framework samples."  
STEP 4: USER root  
STEP 5: RUN yum -y update-minimal --security --sec-severity=Important --sec-severity=Critical  
Updating Subscription Management repositories.  
Unable to read consumer identity  
Subscription Manager is operating in container mode.  
Red Hat Enterprise Linux 8 for x86_64 - BaseOS 21 MB/s | 33 MB 00:01  
Red Hat Enterprise Linux 8 for x86_64 - AppStre 22 MB/s | 31 MB 00:01  
Red Hat Universal Base Image 8 (RPMs) - BaseOS 3.6 MB/s | 786 kB 00:00  
Red Hat Universal Base Image 8 (RPMs) - AppStre 17 MB/s | 2.4 MB 00:00  
Red Hat Universal Base Image 8 (RPMs) - CodeRea 180 kB/s | 15 kB 00:00  
No security updates needed, but 10 updates available  
Dependencies resolved.  
Nothing to do.  
Complete!  
STEP 6: COPY licenses /licenses  
STEP 7: COPY app /opt/app-root/app  
STEP 8: EXPOSE 8088  
STEP 9: EXPOSE 5846-5856  
STEP 10: EXPOSE 80  
STEP 11: EXPOSE 443  
STEP 12: USER 1001  
STEP 13: WORKDIR /opt/app-root/app  
STEP 14: CMD ["dotnet", "XFS4IoT.SP.ServerHostSample.dll", "$@"]  
STEP 15: COMMIT cardreader:0.0.2  
Getting image source signatures  
Copying blob d7b836ae9a00 skipped: already exists  
Copying blob 5b2f0b9e81c7 skipped: already exists  
Copying blob fdd14009318c skipped: already exists  
Copying blob 4b78a7f12faf skipped: already exists  
Copying blob 48b0b9067531 done  
Copying config a9c2cc71d7 done  
Writing manifest to image destination  
Storing signatures  
--> a9c2cc71d72  
a9c2cc71d7264b5f791b4b68a3c3e71b2365becce241f52e88259f591e8d1c45
```

Demo: Container Run Instructions

```
# podman run -d --network=host --name cardreader1 cardreader:latest  
# podman run -d --network=host --name cashdispenser1 cashdispenser:latest
```

Use “--device” parameter to add an actual host device to the container
(e.g. physical card reader attached via USB)

Note:

This is a proof of concept. There are many more options to limit security context and resources per container that are recommended for hardened production setups.

The command above will let both pods share the host’s network in order to leave the service discovery functionality in tact.

Demo: Container Images and State

List container images:

```
# podman images
```

```
[root@jenga container-CR]# podman images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/cardreader	0.0.2	a9c2cc71d726	58 seconds ago	987 MB
localhost/cashdispenser	0.0.2	d55d8a7d4c4e	4 minutes ago	987 MB

List running container instances:

```
# podman ps
```

```
[root@jenga container-CR]# podman run -d --network=host --name cardreader1 localhost/cardreader:0.0.2
9e85f35d3316bf809bd46abf235364f68fb57c676dfc7e34c80ae7bd9f4a9798
[root@jenga container-CR]# podman run -d --network=host --name cashdispenser1 localhost/cashdispenser:0.0.2
2d15948163c47c507eb8000405ef508b7cbb16ff9de7eef5deb97bbb340b7586
[root@jenga container-CR]# podman ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9e85f35d3316	localhost/cardreader:0.0.2	dotnet XFS4IoT.SP...	26 seconds ago	Up 27 seconds ago		cardreader1
2d15948163c4	localhost/cashdispenser:0.0.2	dotnet XFS4IoT.SP...	2 seconds ago	Up 3 seconds ago		cashdispenser1

Demo: Container Runtime Information

Access container logs:

```
# podman logs cashdispenser1
```

```
[root@jenga container-CR]# podman logs --tail 10 cashdispenser1
XFS4IoT.Common.Commands.SetTransactionStateCommand => XFS4IoTFramework.Common.SetTransactionStateHandler Async:False
XFS4IoT.Common.Commands.StatusCommand => XFS4IoTFramework.Common.StatusHandler Async:TrueXFS4IoT.Common.Commands.SynchronizeCommandCommand => XFS4IoTFramework.Common.SynchronizeCommandHandler Async:False
12:17:24.090 (000.456): Listening on http://localhost:5847/xfs4iot/v1.0/SimCashDispenser/
12:17:24.091 (000.456): New endpoint at http://localhost:5847/xfs4iot/v1.0/SimCashDispenser/
12:17:24.102 (000.467): Exception caught on reading persistent data. XFS4IoTFramework.CashManagement.CashUnit, Could not find file '/opt/app-root/app/XFS4IoTFramework.CashManagement.CashUnit'.
12:17:24.102 (000.468): Failed to load persistent data. It could be a first run and no persistent exists on the file system.
12:17:24.108 (000.473): Exception caught on reading persistent data. XFS4IoTFramework.Dispenser.Mix, Could not find file '/opt/app-root/app/XFS4IoTFramework.Dispenser.Mix'.
12:17:24.111 (000.476): http://localhost:5847/xfs4iot/v1.0/ listing for new connections and on 0 existing connections
12:17:24.118 (000.484): http://localhost:5847/xfs4iot/v1.0/SimCashDispenser/ listing for new connections and on 0 existing connections
```

Directly log into container (e.g. for troubleshooting):

```
# podman exec -ti cardreader1 /bin/sh
```


Demo: Connection from remote Test Client

TestClientForms

Card Reader Dispenser

Service URI

Port Number

CardReader URI

Service Discovery

Device status

Media Status

Device type

Status

Capabilities

AcceptCard

EjectCard

CaptureCard

```
{"payload":{"timeout":60000},"headers":{"name":"Common.Status","requestId":5,"type":"command"}}
```

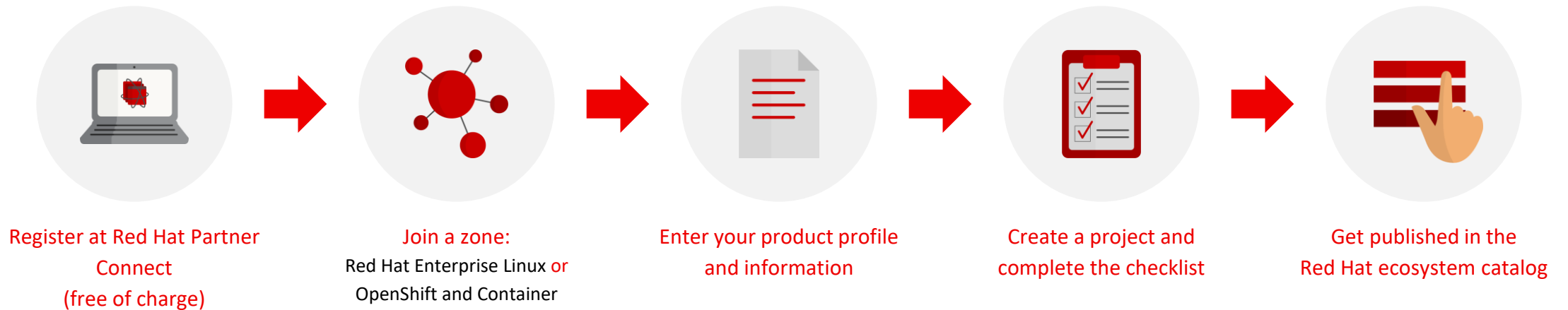
```
{"payload":{"common":{"device":"online","extra":[],"guideLights":[{"flashRate":"off","color":"green","direction":"off"}],"devicePosition":"inposition","powerSaveRecoveryTime":0,"antiFraudModule":"ok"},"cardReader":{"media":"notPresent","retainBin":"ok","security":"notSupported","numberCards":0,"chipPower":"poweredOff","chipModule":"ok","magWriteModule":"ok","frontImageModule":"ok","backImageModule":"ok","parkingStationMedia":[]},"completionCode":"success"},"headers":{"name":"Common.Status","requestId":5,"type":"completion"}}
```



WHAT IS RED HAT TECHNOLOGY CERTIFICATION?

Assurance that a product or solution interoperates with the Red Hat platform throughout its lifecycle, and is commercially supported by Red Hat and partners.

The path to certification and distribution by Red Hat



Get started at connect.redhat.com. Read the [partner guide](#) for more details.


Red Hat Container Catalog

Trusted source for Container Images:

- Signed Images by Red Hat and Certified Partners
- Health Index (A to F grade)*
- Security advisories & errata (patches)
- <https://catalog.redhat.com/software/containers/explore>


The screenshot shows the Red Hat Ecosystem Catalog interface. At the top, there's a navigation bar with the Red Hat logo, 'Red Hat Ecosystem Catalog', and links for 'Hardware', 'Software', and 'Cloud & service providers'. Below this, a breadcrumb trail reads 'Home > Software > Container images > .NET 5.0 SDK and Runtime'. The main content area is titled '.NET 5.0 SDK and Runtime' and is provided by Red Hat. It shows the architecture as 'amd64' and the tag as '5.0-16.20210622184043'. Below this, there's a section for 'latest' with a dropdown for '5.0' and a tag '5.0-16.20210622184043'. The 'Overview' tab is selected, showing a 'Description' of the SDK, 'Products using this container' (including 'Red Hat Universal Base Image 8'), and a 'Health index' of 'A'. Other details include 'Published 17 days ago', 'Release category: Generally Available', 'Size: 282.8 MB (790.9 MB uncompressed)', and 'Digest: d251096...'. A 'Have feedback?' button is at the bottom right.

Red Hat Container Certification: Sample


 Red Hat
Partner Connect

Certified Technology Portal

Product CertificationZones & ResourcesMy CompanySupport

 My Account

Projects >

 Container project


xfs4iot_sp-dev-samples-host

Actions ▾

OverviewImagesBuildsSettings


● Auto publish is off ● Auto rebuild is off ● Operational

Container Images


PID: ospid-c74b91dc-6e17-41d1-ab35-ced5efcfab0b 

Repository Grade: Unknown

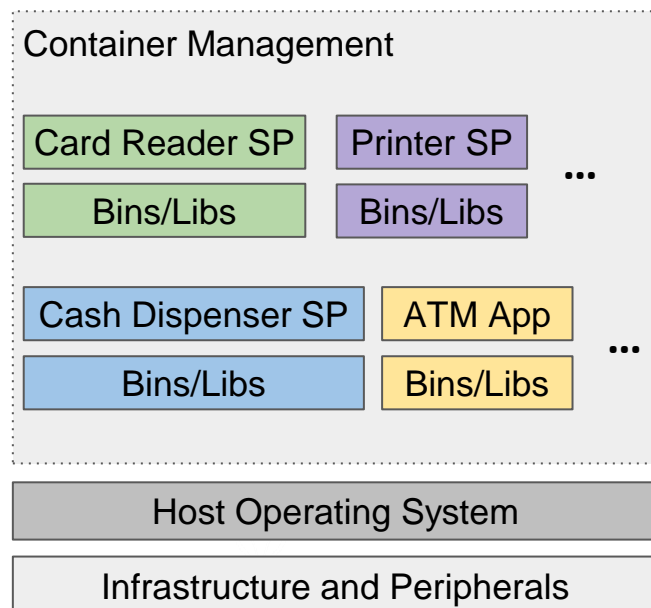
Filter by SHA Name or Tags

 Push Image Manually

1-1 of 1 ▾ < >

Image	Certification test	Health Index ⓘ	Architecture	Created ⓘ	Visibility	
> sha256:0ce86cc28f8e0fc05891c0ae1f34bf2161723c68a093544d311028889cbb057d  0.0.1-alpha.2	✓ Passed	A	amd64	25 minutes ago	Not published	⋮

Benefits of Containers for SP Developers



- **Independence of host, frameworks and other SPs:**
Bundle all your binaries and their requirements, libraries and runtimes into a container image to run isolated from other (interfering) SP's or applications. Goodbye to complex integration testing.
- **Versioning and certification:**
Enable infrastructure-as-code concepts for the SP and all its dependencies in one artefact that can be versioned, tested and even certified against the OS.
- **Security:**
Possibilities to limit security context to a minimum - e.g. a container with a printer SP can never see/access a cash dispenser or its SP.
- **Distribution:**
Leverage existing trusted distribution channels, such as the Red Hat Container Catalog. Enable new go-to-market models to offer software directly to clients.
- **Agility:**
Deliver updates faster and react faster to new business demands. In your development AND to production.

Outlook

Going from Proof of Concept to Pilot and identify additional use cases

- ▶ What's your feedback on the concept of containerization on ATMs?
- ▶ Feel free to build on the existing instructions and assets
- ▶ Reach out if you would like to try these concepts with real ATM hardware and peripherals!
 - Or if you just think about adopting containers for your own development environment
- ▶ Potential workstreams and topic to look into next:
 - Define good practices for ATM-specific setup of Linux (especially for security features and minimizing attack surface)
 - Define good practices for containerization of drivers/SPs/apps to decouple from hardware- and software-interdependencies
 - Management of the ATM software life cycle using containers and OS images to speed up time to market for new features
 - New and innovative workload on ATMs next to existing use cases or as part of hybrid cloud solution

Thank you!

 linkedin.com/company/red-hat

 youtube.com/user/RedHatVideos

 facebook.com/redhatinc

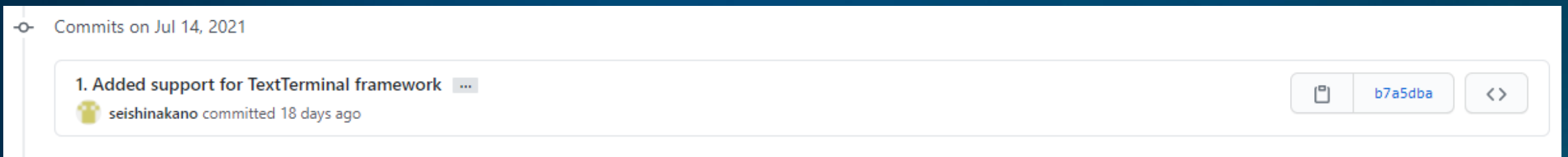
 twitter.com/RedHat

Framework progress and status

Text Terminal (TTU) framework



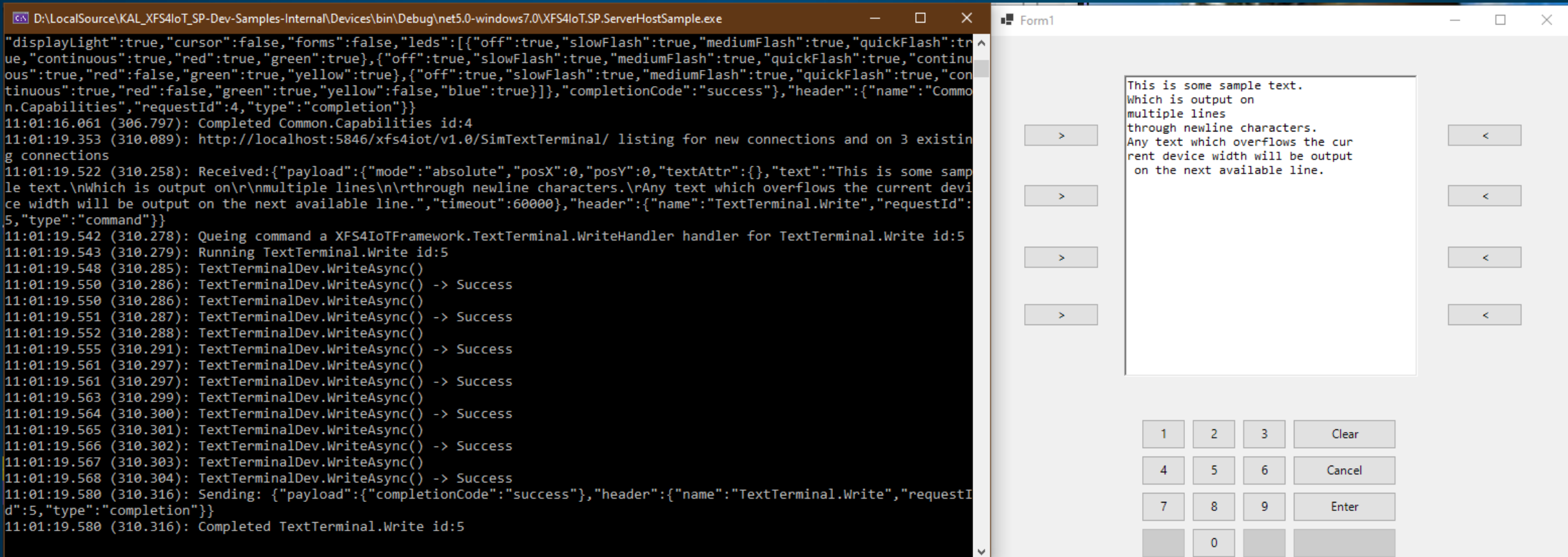
- Released in July
- Only forms are not supported *yet*
- Details of the changes can be seen in the commit



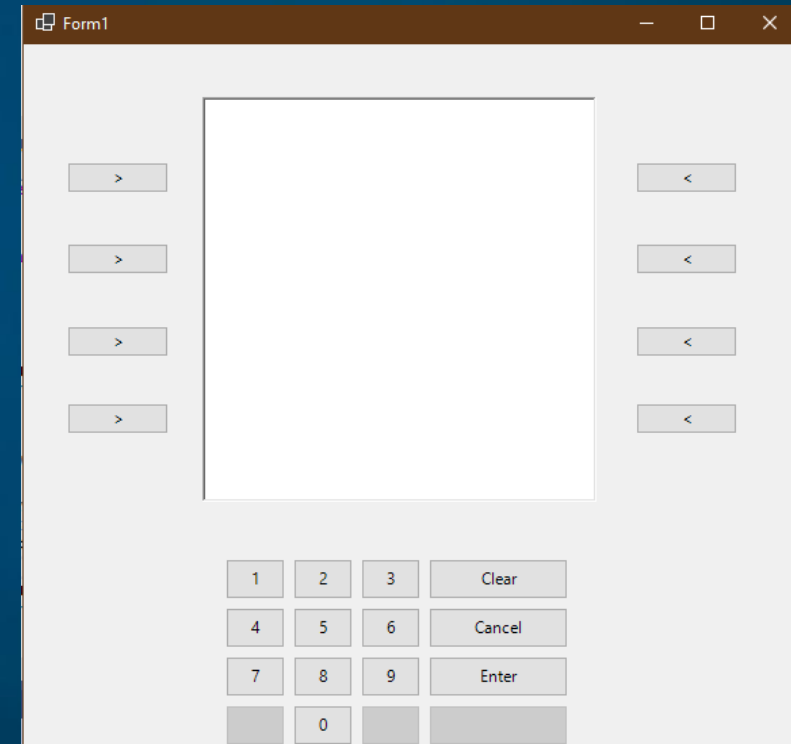
Text Terminal framework



- Sample code available
- Includes simple UI for the sample device



- Can be used as example for SP simulators
- No Cpp sample for TTU – Card reader Cpp sample available
- Ready to be implemented



- 3 devices already supported
 - Card Reader
 - Cash Dispenser
 - Text Terminal
- Sample code available for all supported devices
- Cpp sample code also available
- Demos with real devices

- Encryption related classes in progress
 - Key Management
 - Crypto
- Pinpad and keyboard classes
- Demos with real pinpad
- Key distribution with TR34
- E2E security proof of concept!

- KAL is following agile-like process. We plan to publish 2 months of roadmap
- Any suggestion on the next devices KAL should add is welcome!
- XFS4IoT CEN specification still in progress

CEN committee status of XFS4IoT and roadmap

- Very active CEN committee
- New members joined from the KAL Workgroup
- Sub-group meetings regularly (weekly, bi-weekly)
- CEN committee virtual meeting every month
- CEN specifications in a good state
- Some companies have started implementation...

- Not all classes from XFS3 will make it into the first XFS4IoT release
- Focus on most used devices such as Pinpad, Cash dispenser, Card reader, Printer...
- Devices not included will come soon after
- New device support is possible

- Sub-group work in progress for each device class
- Key CEN XFS dates:
 - End of September preview – close to final spec
 - End of October code freeze
 - Final review period by CEN committee members until mid-December
 - Target release of the XFS4IoT CEN specifications by end of December 2021!

Next meeting / Announcement

- XFS4IoT E2E security demo in **September**
 - Key Management and Crypto classes
 - TR34 support
 - Real device demo
- Other Pinpad-related classes (Keyboard & Pinpad) in **October**
- Cash dispenser E2E security feature support in **October**

MS Teams

- First Tuesday of each month at 1300 UK time

Next call: 7th September 2021, 1300 UK, 0800 US EST, 2100 Tokyo time