XFS4IoT SP-Dev Workgroup

2 November 2021

XFS4IoT workgroup progress

- Frameworks for most major ATM devices now available with C# and C++ sample code released, demos on YouTube
  - Card Reader (released May 2021)
  - Cash Dispenser (July 2021), without end-to-end security
  - Text Terminal Unit (July 2021)
  - EPP Key Management and Crypto classes (Sept 2021)
  - Keyboard and PinPad classes (October 2021)
  - End-to-end security partially complete (October 2021)
- Frameworks yet to be done in order to support a complete Cash Out ATM
  - Printer / Vendor Mode / Vendor Application / Auxiliaries

XFS4IoT workgroup progress

- Framework code updated regularly. Framework code is available to:
  - Implement XFS4 SPs
  - Review, test with our samples
  - Write test tools

CEN specification September preview – updating the framework

© 2021 KAL ATM Software GmbH (KAL)

Framework : latest updates

- New card reader interface:
  - Dispensing capabilities
  - “Move” command for all media transport
  - MoveHandler.cs

```
namespace XFS4IoTFramework.CardReader
{
    public partial class MoveHandler
    {
        private async Task<MoveCompletion.PayloadData> HandleMove(IMoveEvents events, MoveCommand move, CancellationToken cancel)
        {
            if (string.IsNullOrEmpty(move.Payload.From))
            {
                return new MoveCompletion.PayloadData(MessagePayload.CompletionCodeEnum.InvalidData,
                                                      "The property From is not specified.");
            }

            MoveCardRequest.MovePosition.MovePositionEnum fromPos = MoveCardRequest.MovePosition.MovePositionEnum.Storage;

            // First to check the specified storage is valid
            if (move.Payload.From != "exit" &&
                move.Payload.From != "transport")
            {
                if (!CardReader.CardUnits.ContainsKey(move.Payload.From))
                {
                    return new MoveCompletion.PayloadData(MessagePayload.CompletionCodeEnum.InvalidData,
                                                          $"Invalid StorageId supplied for From property. {move.Payload.From}");
                }
```

Framework : latest updates

- New Storage Interface
  - To be used with other device interfaces which requires storage
  - All bins and cash unit storage handling

main KAL_XFS4IoT_SP-Dev / Framework / ServiceClasses / StorageServiceProvider / **Handlers** /

seishinakano Updated SP framework to be aligned with the September preview. ...

..

EndExchangeHandler.cs	Updated SP framework to be aligned with the September preview.
EndExchangeHandler_g.cs	Updated SP framework to be aligned with the September preview.
GetStorageHandler.cs	Updated SP framework to be aligned with the September preview.
GetStorageHandler_g.cs	Updated SP framework to be aligned with the September preview.
SetStorageHandler.cs	Updated SP framework to be aligned with the September preview.
SetStorageHandler_g.cs	Updated SP framework to be aligned with the September preview.
StartExchangeHandler.cs	Updated SP framework to be aligned with the September preview.
StartExchangeHandler_g.cs	Updated SP framework to be aligned with the September preview.

Framework : latest updates

- Versioning information
  - Capabilities command
  - Per command
  - Per event
- *CommonSchemas* class

```
[DataContract]
public sealed class CommandsClass
{
    public CommandsClass(List<string> Versions = null)
    {
        this.Versions = Versions;
    }

    /// <summary>
    /// The versions of the command supported by the service. There will be one item for each major version
    /// supported. The minor version number qualifies the exact version of the message the service supports.
    /// <example>["1.3", "2.1", "3.0"]</example>
    /// </summary>
    [DataMember(Name = "versions")]
    [DataTypes(Pattern = @"^[1-9][0-9]*\.([1-9][0-9]*|0)$")]
    public List<string> Versions { get; init; }

}

/// <summary>
/// The commands supported by the service.
/// </summary>
[DataMember(Name = "commands")]
public Dictionary<string, CommandsClass> Commands { get; init; }

[DataContract]
public sealed class EventsClass
{
    public EventsClass(List<string> Versions = null)
    {
        this.Versions = Versions;
    }

    /// <summary>
    /// The versions of the event supported by the service. There will be one item for each major version
    /// supported. The minor version number qualifies the exact version of the message the service supports.
    /// <example>["1.3", "2.1", "3.0"]</example>
    /// </summary>
    [DataMember(Name = "versions")]
    [DataTypes(Pattern = @"^[1-9][0-9]*\.([1-9][0-9]*|0)$")]
    public List<string> Versions { get; init; }

}
```

XFS4IoT CEN specification

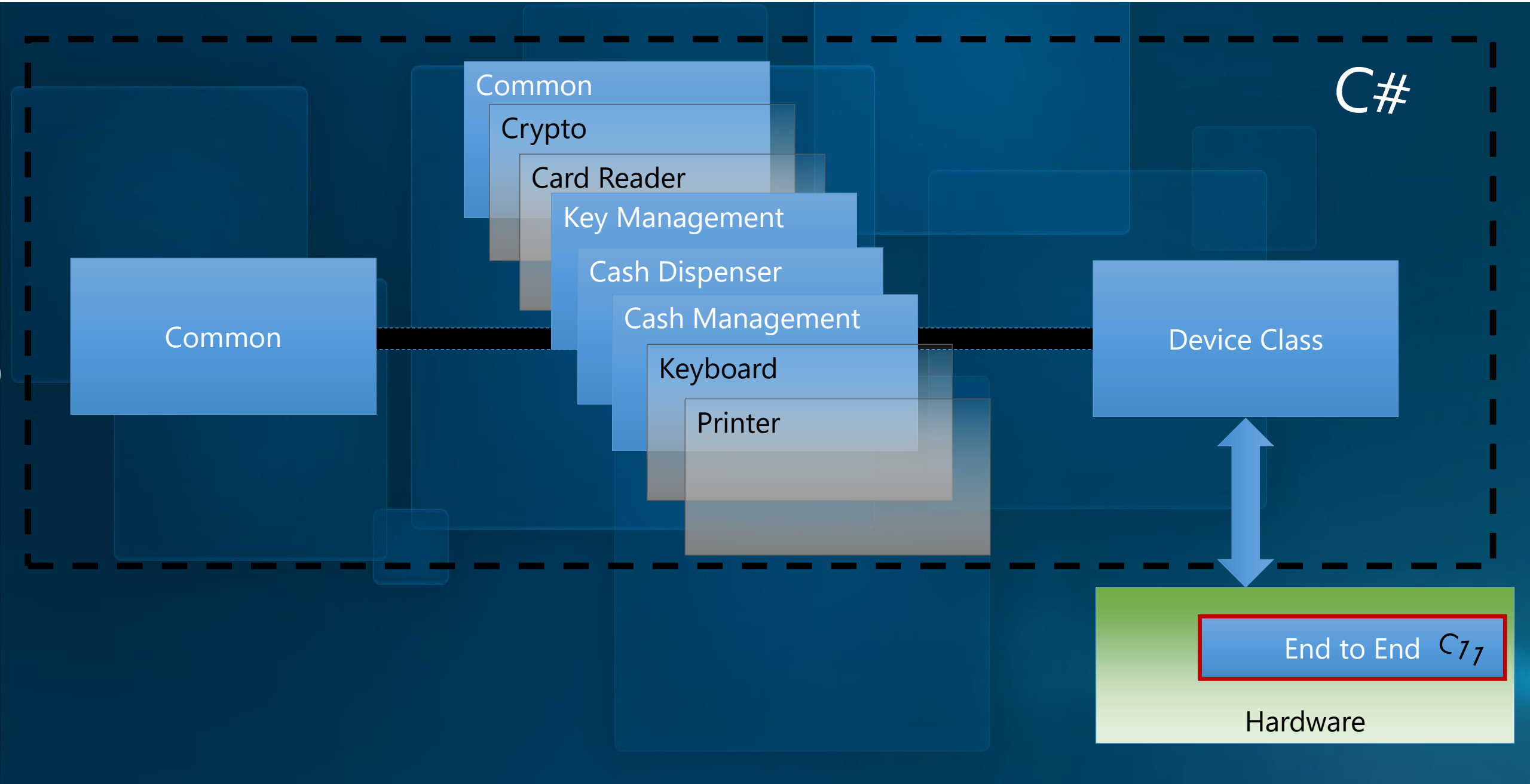
- Roadmap update:
  - Code freeze on specification
  - Next steps
    - Document generation
    - Final review period by CEN committee members
    - Final validation
    - Release process with CEN / Preview updated by **end of Dec 2021**

XFS4IoT SP-Dev E2E support

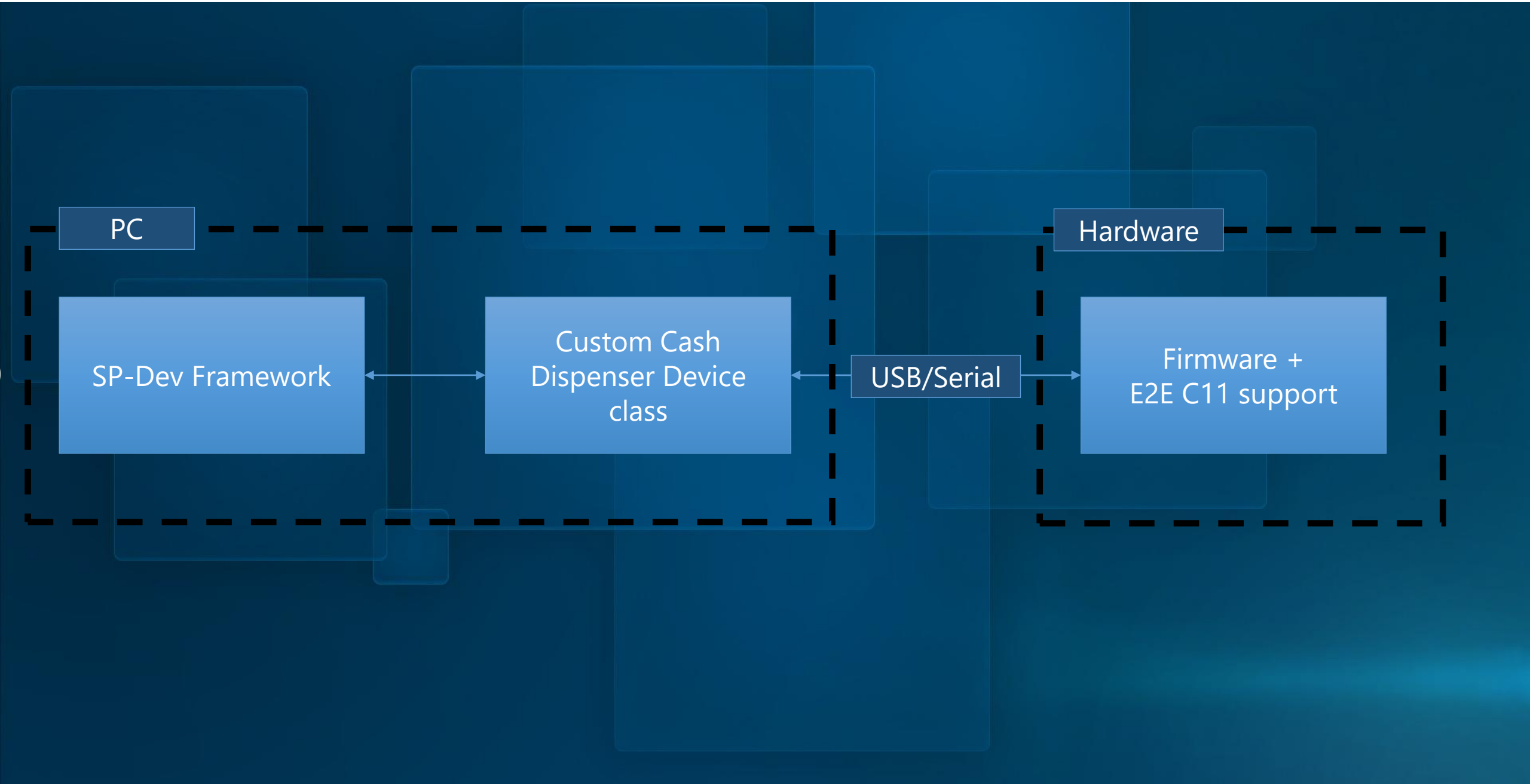
E2E support, November update

E2E security

Framework with End to End security



Components


PC
SP-Dev Framework
Custom Cash Dispenser Device class
USB/Serial
Hardware
Firmware + E2E C11 support

Specification change

Important change since the September preview was published:

- Previously : clear token on Present
- New proposal : only clear when total amount dispensed
  - Support partial dispense and retry on failure
  - Support dispense of large amounts in multiple operations

This is now mostly supported by the SP-Dev framework

Sample ‘firmware’ - October

```
bool KAL::XFS4IoTSP::CashDispenser::Sample::Firmware::VerifyAndDispense(System::String^ Token)
{
    // Convert the .net UTF16 string into a native UTF8 string.
    pin_ptr<const wchar_t> pinnedToken = PtrToStringChars(Token);
    wstring wideToken = pinnedToken;

    wstring_convert<codecvt_utf8<wchar_t>> utf8Converter;
    string utf8Token;
    try
    {
        utf8Token = utf8Converter.to_bytes(wideToken);
    }
    catch (range_error const &)
    {
        cout << "Error: Invalid token. Conversion to UTF8 failed after " << dec << utf8Token.size() << " characters:\n";
        for (auto c : utf8Token)
            cout << hex << showbase << c << '\n';
    }

    // Check that the token is valid.
    // Include null in token (buffer) size.
    auto tokenValid = ValidateToken(utf8Token.c_str(), utf8Token.size()+1);
    if (!tokenValid)
        return false;

    return true;
}

System::String^ KAL::XFS4IoTSP::CashDispenser::Sample::Firmware::GetCommandNonce() { ... }

void KAL::XFS4IoTSP::CashDispenser::Sample::Firmware::ClearCommandNonce() { ... }
```

Sample ‘firmware’ - November

```
bool KAL::XFS4IoTSP::CashDispenser::Sample::Firmware::VerifyAndDispense(System::String^ Token, System::String^ Currency, int Value )
{
    // Convert the .net UTF16 string into a native UTF8 string.
    pin_ptr<const wchar_t> pinnedToken = PtrToStringChars(Token);
    pin_ptr<const wchar_t> pinnedCurrency = PtrToStringChars(Currency);

    wstring_convert<codecvt_utf8<wchar_t>> utf8Converter;
    string utf8Token;
    string utf8Currency;
    try
    {
        utf8Token = utf8Converter.to_bytes(pinnedToken);
        utf8Currency = utf8Converter.to_bytes(pinnedCurrency);

    }
    catch (range_error const &)
    {
        cout << "Error: Conversion to UTF8 failed after " << dec << utf8Token.size() << " characters:\n";
        for (auto c : utf8Token)
            cout << hex << showbase << c << '\n';
    }

    // Check that the token is valid and authorises the requested dispense.
    // Include null in token (buffer) size.
    auto Authorised = AuthoriseDispenseAgainstToken(utf8Token.c_str(), utf8Token.size() + 1, Value, 0, utf8Currency.c_str());
    if (!Authorised)
        return false;

    auto TokenValues = GetDispenseKeyValues();
    cout << "Got dispense token values: Currency:" << string(TokenValues->Currency, 3) << " value:" << TokenValues->Value << " cents:" << TokenValues->Fraction << "\n";

    // Simulate the actual dispense
    cout << "dispensing: " << TokenValues->Value << "." << TokenValues->Fraction << string(TokenValues->Currency, 3) << "\n";
    Sleep(1000);
    cout << "dispensed\n";


    return true;
}
```

Core firmware check

```
/// ...
bool AuthoriseDispenseAgainstToken(char const* Token, size_t TokenLength, unsigned int Value, unsigned int Fraction, char const Currency[3])
{
    LogV("AuthoriseDispenseAgainstToken( Token=\"%.1024s\", TokenSize=%d, Value=%d, Fraction=%d, Currency=\"%c%c%c\"  )", Token, TokenLength, Value, Fr

    bool result = false;
    bool ExistingToken = CurrentTokenSet;

    result = ValidateToken(Token, TokenLength);
    if (!result) { ... }

    if (!ExistingToken)
    {
        result = ParseDispenseToken(Token, TokenLength);
        if (!result) { ... }
    }

    result = AuthoriseDispense(Value, Fraction, Currency);
    if (!result) { ... }

    LogV("AuthoriseDispenseAgainstToken: ⇒ true");
    return true;
}
```

Firmware functions

```
bool AuthoriseDispenseAgainstToken( char const *const Token, size_t TokenLength,
                                    unsigned int Value, unsigned int Fraction, char const Currency[3]);
bool InvalidateToken();

bool ValidateToken(char const *const Token, size_t TokenLength);
bool ParseDispenseToken(char const *const Token, size_t TokenSize);
bool AuthoriseDispense(unsigned int UnitValue, unsigned int SubUnitValue, char const Currency[3]);
```

Test Client

```
token = MakeToken(await DoGetCommandNonce(connection), "300.00EUR");
await DoDispenseCash(connection, 100, "EUR", token); // Success
await DoPresentCash(connection);
await DoDispenseCash(connection, 100, "EUR", token); // Success
await DoPresentCash(connection);
await DoDispenseCash(connection, 100, "EUR", token); // Success
await DoPresentCash(connection);
await DoDispenseCash(connection, 100, "EUR", token); // Invalid Token
```

Status

- Complete
  - Added firmware C library support (including unit tests.)
  - Framework support for Common.GetCommandNonce and CashDispenser.Dispense token commands.
  - Sample SP implementation using ‘firmware’ code and dispenser implementation
  - (Command line) test client showing dispense and present sequence
  - Read and track token keys – such as dispense amount
  - Enforce dispense values
  - Automatically delete the nonce/invalidate tokens
  - UI test client

To do

- Support failed dispense/partial dispense and retry

- Support for generating response tokens
- GetPresentStatus token support

Not included

- Key handling – KeyManagement, but needs hardware support
- Cryptography – must be done in hardware
- Random number or Persistent storage – needs to be done in hardware

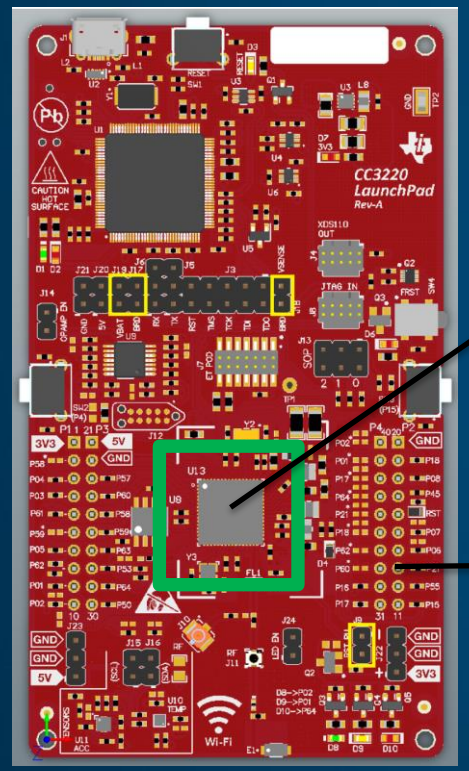
- Not planned
  - Support for multiple currencies

E2E process on real hardware

© 2021 KAL ATM Software GmbH (KAL)

E2E security on real hardware

Hardware components



Ti LaunchPad development board

SPI protocol

- Arm® Cortex®-M4 core
- 256KB of RAM
- UART, I2S, I2C, SPI, SD, ADC
- Cost less than 5USD (Volume 1 – Retail)

- Trusted Platform Module
- Version 2.0 of TCG.
- (SPI) Protocol up to 36 MHz
- RNG, HMAC, AES, SHA-256, ECC, and RSA
- Cost less than 2USD (Volume 1 – Retail)

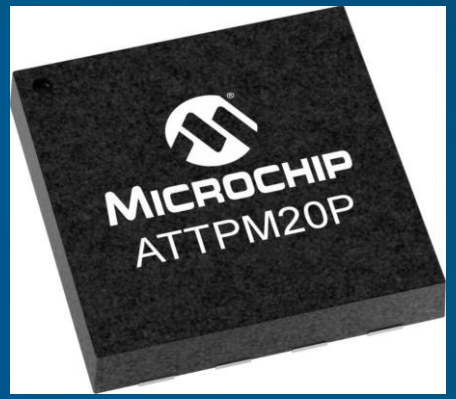
References:
https://www.ti.com/launchpad
https://www.microchip.com/en-us/product/ATTPM20P

E2E security on real hardware

TPM → Trusted Platform Module

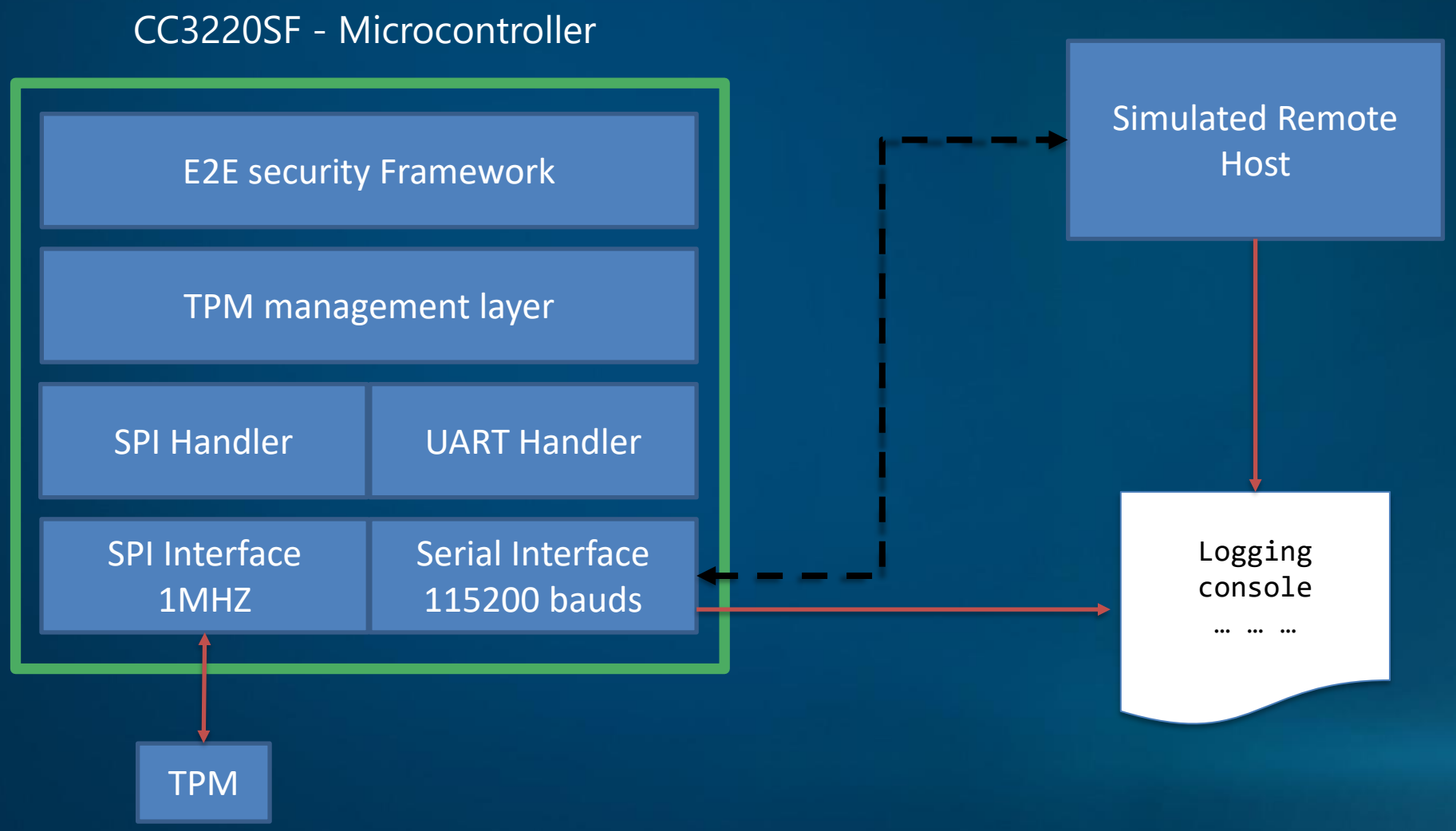
- Resource constrained chip
- Secure crypto-processor
- Secure key generation and key storage
- Hardware Asymmetric/Symmetric Crypto Engine
- Cryptographic Support for: RNG, HMAC, AES, SHA, ECC, RSA, etc.
- Two generations: TPM1.2 and TPM2.0
- Trusted Computing Group compliance

TPM = Security



E2E security on real hardware

Software Demo components



E2E security on real hardware

End to End security Framework – Implemented Functions

- **CompareNonce**

```
extern bool CompareNonce(char const* const CommandNonce, size_t NonceLength)
{
    //Variables
    int i;
    uint8_t     *nonceHex;
    size_t      nonceHexLen = 0;
    uint32_t    result = 0;

    //1. Calculate the length in HEX bytes
    nonceHexLen = NonceLength/2;
    if(NonceLength % 2 != 0)
        nonceHexLen += 1;
    nonceHex = (uint8_t *) malloc( sizeof(uint8_t)*nonceHexLen);

    //2. Convert to HEX
    ConvertoHex((void *)CommandNonce,NonceLength,nonceHex);

    //3. Compare the nonces (constant time comparison)
    Log("Calculated vs. Received\n\r");
    for(i = 0 ; i < getRandomOut.randomBytesSize ; i++)
    {
        LogV("DEVICE >> (%02X - %02X)\n\r", *(getRandomOut.randomBytes + i), *(nonceHex + i));
        result |= *(getRandomOut.randomBytes + i) ^ *(nonceHex + i);
    }
    free(nonceHex);

    //4: Accept or Decline the token.
    return result == 0;
}
```

E2E security on real hardware

End to End security Framework – Implemented Functions

- **ClearNonce**

```
extern void ClearNonce()
{
    // Clean nonce variables
    getRandomOut.randomBytesSize = 0;

    if(getRandomOut.randomBytes != NULL)
        free(getRandomOut.randomBytes);
}
```

- **NewNonce**

```
extern void NewNonce( char const ** Nonce )
{
    size_t  length = 20;

    // Step 1: Generate a RNG using the TPM!!
    if(!TPM2_GetRandom(length))
    {
        Log("Random Number generation error");
        while(1);
    }
}
```

- **CheckHMAC**

```
extern bool CheckHMAC(char const *const Token, unsigned int TokenLength, char const *const TokenHMAC)
{
    //Variables
    int             i;
    uint32_t        result = 0;

    // Procedure
    Log("CheckHMAC validation!");

    //1: Compute HMAC in TPM
    TPM2_HMAC(TPM20_AUTH_PASSWORD_SESSION, (void *)Token, TokenLength);

    Log("Calculated vs. Received\n\r");
    //2: Compare the computed HMAC value with the one received in the token (constant time comparison)
    for(i = 0 ; i < 32 ; i++)
    {
        LogV("DEVICE >> (%02X - %02X)\n\r", *(hmacOut.outHMAC + i), TokenHMAC[i]);
        result |= *(hmacOut.outHMAC + i) ^ TokenHMAC[i];
    }

    //3: Accept or Decline the token.
    return result == 0;
}
```

E2E security on real hardware

End to End security Framework

- Required Functions
  - ClearNonce
  - NewNonce
  - CheckHMAC
  - CompareNonce

E2E security on real hardware

TPM Management Layer

- Initialization Sequence

- Provisioning Process

E2E security on real hardware

TPM Management Layer

- TPM crypto facilities

Generate Nonce (RNG)

Table 66 — TPM2_GetRandom Command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS if an audit or encrypt session is present; otherwise, TPM_ST_NO_SESSIONS
UINT32	commandSize	
TPM_CC	commandCode	TPM_CC_GetRandom
UINT16	bytesRequested	number of octets to return

Table 67 — TPM2_GetRandom Response

Type	Name	Description
TPM_ST	tag	see clause 6
UINT32	responseSize	
TPM_RC	responseCode	
TPM2B_DIGEST	randomBytes	the random octets

```
bool TPM2_GetRandom(uint16_t numBytesReq)
{
    //Variables
    uint32_t        res = TPM_FAIL;
    size_t          inBufSize = 12;
    size_t          inBufOS = 0;
    size_t          outBufSize = 0;
    uint8_t         *TPM2_GetRandom_cmd;
    int             i;
    bool            Ready = false;

    //Check the TIS registry status before sending data
    Ready = PrepareTISRegister();

    if(Ready)
    {
        UART_PRINT("\n\rDEVICE >> TPM2_GetRandom!");
        //Create TPM2_LoadExternal input buffer
        inBufOS = 0;
        TPM2_GetRandom_cmd = (uint8_t *) malloc( sizeof(uint8_t)*inBufSize);
        //Populate the data
        /*HEADER*/
        AppendConstToBuffer(TPM2_GetRandom_cmd, TPM_ST_NO_SESSIONS, 2, &inBufOS, true);
```

HMAC Calculation

Table 64 — TPM2_HMAC Command

Type	Name	Description
TPMI_ST_COMMAND_TAG	tag	TPM_ST_SESSIONS
UINT32	commandSize	
TPM_CC	commandCode	TPM_CC_HMAC
TPMI_DH_OBJECT	@handle	handle for the symmetric signing key providing the HMAC key Auth Index: 1 Auth Role: USER
TPM2B_MAX_BUFFER	buffer	HMAC data
TPMI_ALG_HASH+	hashAlg	algorithm to use for HMAC

Table 65 — TPM2_HMAC Response

Type	Name	Description
TPM_ST	tag	see clause 6
UINT32	responseSize	
TPM_RC	responseCode	
TPM2B_DIGEST	outHMAC	the returned HMAC in a sized buffer

```
bool TPM2_HMAC(uint8_t AuthSessionType, void *inputDataHmac, size_t inputDataHmacLen)
{
    //Variables
    uint32_t        res = TPM_FAIL;
    size_t          inBufSize = 0;
    size_t          inBufOS = 0;
    uint8_t         *TPM2_HMAC_cmd;
    uint8_t         *AuthStructure;
    int             i;
    bool            Ready = false;

    //Check the TIS registry status before sending data
    Ready = PrepareTISRegister();

    if(Ready)
    {
        UART_PRINT("\n\rDEVICE >> TPM2_HMAC!");

        if(AuthSessionType == TPM20_AUTH_PASSWORD_SESSION)  // Password Session
        {
            //Create authorization structure
            AuthStructure = (uint8_t *) malloc( sizeof(uint8_t)*29);
```

References: TPM2.0 Parts 1 to 4, on the TCG website: https://www.trustedcomputinggroup.org

E2E security on real hardware

Peripherals Interfaces

SPI Interface

```c
SPI_Handle      masterSpi;

void InitSPI(void)
{
    SPI_Params      spiParams;

    // Init SPI Module
    SPI_init();

    /* Open SPI as master and configure SPI bus */
    SPI_Params_init(&spiParams);
    spiParams.mode = SPI_MASTER;
    spiParams.frameFormat = SPI_POL0_PHA0;
    spiParams.bitRate = 1000000;
    spiParams.dataSize = 8;

    // Open SPI handler
    masterSpi = SPI_open(CONFIG_SPI_0, &spiParams);
    if (masterSpi == NULL) {
        UART_PRINT("\n\rError initializing master SPI");
    }
    else
        UART_PRINT("\n\rMaster SPI initialized\n\r");
}
```

UART Interface

```c
static UART_Handle uartHandle;

UART_Handle InitUart(void)
{
    UART_Params uartParams;

    UART_init();
    UART_Params_init(&uartParams);

    uartParams.writeDataMode = UART_DATA_BINARY;
    uartParams.readDataMode = UART_DATA_BINARY;
    uartParams.readReturnMode = UART_RETURN_FULL;
    uartParams.readEcho = UART_ECHO_OFF;
    uartParams.baudRate = 115200;
    uartParams.dataLength = 8;

    uartHandle = UART_open(CONFIG_UART_0, &uartParams);
    /* remove uart receive from LPDS dependency */
    UART_control(uartHandle, UART_CMD_RXDISABLE, NULL);

    return(uartHandle);
}
```

E2E security on real hardware

Demo Simulated Host

- Console Application.
- Connected through UART (COM4 port)

Idle – Waiting serial Data

Compute Token
Wait Request

Compute BAD Token

HOST

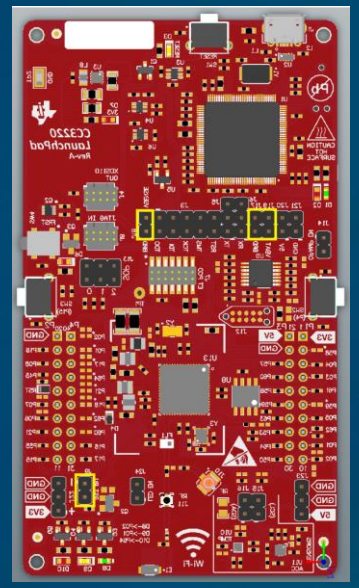
Send Random Nonce

Token Request 01

GOOD token response

Token Request 02

BAD token response



DEVICE

Start Initialization Sequence
Generates random nonce (TPM)

Initialization completed

Token Validation **OK**
Parse Token and Blink LED

Token Validation **FAILS**

E2E security on real hardware

DEMO VIDEO

Demo with real hardware

Demo video will be available on YouTube.

All previous demo videos can be found on the KAL ATM Software YouTube channel:

https://www.youtube.com/user/ATMsoftware/videos

What’s next?

© 2021 KAL ATM Software GmbH (KAL)

What's next?

- Support more classes:
  - Printer
  - Vendor Mode/Vendor Application
  - Auxiliaries

→ *Everything needed to support a complete Cash Out ATM…*

Next call

MS Teams

- First Tuesday of each month at 1300 UK time

Next call: 7th December 2021, 1300 UK, 0800 US EST, 2200 Tokyo time