PosDeviceListener Doc

1.1 Containers:

1.1.1 CrytpoVison Containers:

```
public class InitTse
{
    //8Bit For Cryptovision / 8 Bits Swissbit
    public byte[] AdminPin { get; set; }
    //10 Bytes for Cryptovision / 8 Bits for Swissbit
    public byte[] AdminPuk { get; set; }
    //8Bit For Cryptovision / 8 Bits Swissbit
    public byte[] TimeAdminPin { get; set; }
    //10 Bytes for Cryptovision / 8 Bits for Swissbit
    public byte[] TimeAdminPuk { get; set; }
    public string DevicePath { get; set; }
    //Millisecond Epoch
    public long Date { get; set; }
    //Cash Register Name
    public string ClientId { get; set; }
    public bool SwissbitTse { get; set; }
}
```

```
public class TseStatus
{
    public string Exception { get; set; }
    public int MaxClients { get; set; }
    public int MaxConcurrentTransactions { get; set; }
    public int UpdateVariants { get; set; }
    public long SyncExpirationDate { get; set; }
    public int CurrentClients { get; set; }
    public int CurrentTransactions { get; set; }
    public int SyncVariants { get; set; }
    public int SyncInterval { get; set; }
    public int WearIndicator { get; set; }
    public string SerialNumber { get; set; }
    public long ExpiryDate { get; set; }
    public string PublicKey { get; set; }
    public string[] ErsMappings { get; set; }
    public string SignatureAlgorithm { get; set; }
    public int TransactionCounter { get; set; }
    public int SignatureCounter { get; set; }
    public int[] OpenTransactions { get; set; }
    public long TotalMemory { get; set; }
    public long AvailableMemory { get; set; }
```

```
}
```

```
public class TseTransactionRequest
{
    public string ProcessData { get; set; }
    public string ClientId { get; set; }
    public int TransactionId { get; set; }
    public long TransactionDate { get; set; }
}
```

```
public class TransactionResult
{
    public long LogTime { get; set; }
    public string SerialNumber { get; set; }
    public int SignatureCounter { get; set; }
    public string SignatureValue { get; set; }
    public int TransactionId { get; set; }
    public int Result { get; set; }
    public long SyncExpirationDate { get; set; }
}
```

```
public class TseExportRequest
{
    public string ClientId { get; set; }
    public uint TransactionNumber { get; set; } = uint.MaxValue;
    public uint StartTransactionNumber { get; set; } = 0;
    public uint EndTransactionNumber { get; set; } = uint.MaxValue;
    public long StartDate { get; set; } = 0;
    public long EndDate { get; set; } = long.MaxValue;
    public int MaximumNumberRecords = int.MaxValue;
}
```

```
public class UnlockUser
{
    public string UserName { get; set; }
    public byte[] Puk { get; set; }
    public byte[] NewPin { get; set; }
    public string DevicePath { get; set; }
}
```

```
public class ErsToKey
{
    public string RegisterId { get; set; }
    public byte[] SerialNumber { get; set; }
    public string DevicePath { get; set; }
    public string TimeAdminAuth { get; set; }
}
```

1.2 Methods

1.2.1 CryptoVision API-Calls

Method: POST

Path: “/tseFirstBoot”
Request Object: InitTse
Return Object: TseStatus

Kann ohne Initialisierung ausgeführt werden.

1. Setzt die Pins und Puks
2. Liest Seriennummer aus
3. Überprüfet Logins aus Funktionalität
4. Initialisiert die TSE
5. Setzt zum ersten Mal die Zeit
6. Registriert den Kassennamen mit dem TSE Modul
7. Aktiviert die TSE.

Anmerkung: Aktivierung Ist nicht das gleiche wie Initialisierung, Die Initialisierung kann nur einmalig ausführet werden, wenn die TSE sich im Auslieferungszustand befindet, die TSE kann jedoch beliebig oft aktiviert und deaktiviert werden. Die Deaktivierung sperrt die TSE nur und hat keinen Einfluss auf die gespeicherten Transaktionen

Method: POST
Path: „/tseInit“
Request Object: InitTse
Return Object: TseStatus

Die Initialisierung hier unterscheidet sich von der Initialisierung beim Erstbetrieb.

Diese Funktion sollte immer vor dem Ausführen einer Transaktion und beim Starten der Kasse ausgeführt werden um die Funktionalität des TSE Modules zu prüfen, sollte die Initialisierung fehlschlagen wird dennoch ein TSE Status Objekt zurückgegeben, In diesem wird der vorliegende Fehler im Feld „string Exception“ notiert, bei einer erfolgreichen Initialisierung bleibt dieses Feld leer.

Method: GET

Path: „/tseStatus“

Return Object: TseStatus

Liefert den derzeitigen Status der TSE, diese Funktion kann nur erfolgreich ausgeführt werden wenn die TSE im Betrieb bereits Initialisiert wurde (siehe Oben), andernfalls wird im Feld Exception „Not Initialised“ angegeben.

Method: POST

Path: „/tseSetTime“

Request Object long/int64 (Millisecond Epoch)

Response Object: TseStatus

Setzt manuell die Uhrzeit auf der TSE

Method: POST

Path: „/tseExecuteTransaction“

Request Object: TseTransactionRequest

Response Object: TransactionResult

Füllt eine vollständige Transaktion aus (Transaction Start und Transaction Finish in einem)

Method: POST

Path: „/tseStartTransaction“

Request Object: TseTransactionRequest

Response Object: TransactionResult

Startet eine Transaktion, dass Response Object hat eine Transaktionsnummer, die bei einem Update oder Finish angegeben werden MUSS!!

Method: POST

Path: „/tseUpdateTransaction“

Request Object: TseTransactionRequest

Response Object: TransactionResult

Aktualisiert eine Transaktion, dass Response Object der letzten Transaktion enthielt eine Transaktionsnummer, die hier angegeben werden MUSS!!

Method: POST

Path: „/tseFinishTransaction“

Request Object: TseTransactionRequest

Response Object: TransactionResult

Endet eine Transaktion, dass Response Object der letzten Transaktion enthielt eine Transaktionsnummer, die hier angegeben werden MUSS!!

Method: POST

Path: „/tseGetExport“

Request Object: TseExportRequest

Response Object: byte[]

Liefert ein byte array welches als .tar Datei auf den Speicher geschrieben werden muss.

Method: GET

Path: „/tseGetCertificates“

Response Object: byte[]

Liefert die Zertifikate in einem byteArray.

Method: POST

Path: „/unblockUser“

Request Object: UnblockUser

Return Object: int (Enum)

Diese Funktion wird verwendet, um den Admin oder den TimeAdmin mithilfe des PUKs bei mehrfacher falsch Eingabe des PINs wieder freizuschalten das Ergebnis wird als int wert zurückgegeben und bezieht sich auf folgendes enum:

```
public enum UnblockResult {
    UnblockOk = 0,
    UnblockFailed,
    UnblockPinIsBlocked,
    UnblockUnknownUserId,
    UnblockError
}
```

Die PUKs der jeweiligen Nutzer können nicht gesperrt werden, diese werden nur für einige Minuten gesperrt, um einen Brute-Force angriff entgegenzuwirken.

Method: Post

Path „/tseMapErsToKey“

Request Object: ErsToKey

Return Object: bool

Diese Funktion bindet einen Kassennamen and Die TSE

ACHTUNG:

Derzeit wird nur ein einziger Kassenname PER Cryptovision TSE unterstützt, hierbei handelt es sich um eine Limitierung des verwendeten TSE Chips, diese Funktion sollte NICHT selbstständig ausgeführt werden da bei dem Inbetriebnehmen der TSE bereits ein Name vergeben wird. Sie wird hier trotzdem zu zwecken der Fehlerbeseitigung bei einer unvollständigen Aktivierung gelistet.

Method: GET

Path: „/tseActivate“

ResponseObject int

Aktiviert eine Deaktivierte TSE

Method: GET

Path: „/tseDectivate“

ResponseObject int

Deaktiviert eine Aktive TSE

1.3 Exceptions/Return Codes

1.3.1 Cryptovision Return codes

```
ExecutionOk = 0, // 0000 no error
ErrorMissingParameter = 0x1000, // 1000 a mandatory input parameter is NULL
ErrorFunctionNotSupported, // 1001 the function is not supported
ErrorAllocationfailed, // 1002 the memory allocation for an output parameter failed
ErrorFileNotFound, // 1003 transport I/O configuration file not found
ErrorSECommunicationFailed, // 1004 communication with Secure Element failed
ErrorTSECommandDataInvalid, // 1005 invalid TSE command data
ErrorTSEResponseDataInvalid, // 1006 invalid TSE response data
ErrorTSEUnknownError, // 1007 unknown TSE error code
ErrorStreamWrite, // 1008 write to output stream failed
ErrorIO, // 1009 transport I/O connection error
ErrorTSETimeout, // 100A transport I/O timeout error
ErrorBufferTooSmall, // 100B transport I/O buffer too small
ErrorCallback, // 100C callback function failed (e.g. se_writer_t export data callback)
ErrorTSEFirmwareVersion, // 100D TSE firmware incompatible with API version
ErrorTransport, // 100E data fragmentation on transport layer

ErrorAuthenticationFailed = 0x2000, // 2000 return value for the SE API function
authenticateUser, authentication failed
ErrorUnblockFailed, // 2001 return value for the SE API function unblockUser, unblock
failed

ErrorRetrieveLogMessageFailed = 0x3000, // 3000 the retrieving of the log message parts
that have been created by Secure Element most recently failed
ErrorStorageFailure, // 3001 storing of the log message in the storage failed
ErrorUpdateTimeFailed, // 3002 the execution of the Secure Element functionality for
setting the time failed
ErrorParameterMismatch, // 3003 there is a mismatch regarding the particular parameters
that have been provided in the context of the export of stored data
ErrorIdNotFound, // 3004 no data has been found for the provided clientID in the context of
the export of stored data
ErrorNoSuchKey, // 3005 unknown key serial number (hash of public key)
ErrorERSalreadyMapped, // 3006 the serial number of an ERS is already mapped to a signature
key
ErrorNoERS, // 3007 unknown ERS
ErrorNoKey, // 3008 unknown signature key
ErrorTransactionNumberNotFound, // 3009 no data has been found for the provided transaction
number(s) in the context of the export of stored data
ErrorNoDataAvailable, // 300A no data has been found for the provided selection in the
context of the export of stored data
ErrorTooManyRecords, // 300B the amount of requested records exceeds the passed value for
the maximum number of records in the context of the export of stored data
ErrorStartTransactionFailed, // 300C the execution of the Secure Element functionality to
start a transaction failed
ErrorUpdateTransactionFailed, // 300D the execution of the Secure Element functionality for
updating a transaction failed
ErrorFinishTransactionFailed, // 300E the execution of the Secure Element functionality for
finishing a transaction failed
ErrorRestoreFailed, // 300F the restore process in the context of a restoring from a backup
in form of exported data failed
ErrorStoringInitDataFailed, // 3010 the storing of the initialization data during the
commissioning of the SE API by the application operator failed
ErrorExportCertFailed, // 3011 the collection of the certificates for the export failed
ErrorNoLogMessage, // 3012 no log message parts have been found in the Secure Element
ErrorReadingLogMessage, // 3013 the retrieving of the log message parts that have been
created from Secure Element most recently failed
ErrorNoTransaction, // 3014 no transaction is known to be open under the provided
transaction number
ErrorSeApiNotInitialized, // 3015 the SE has not been initialized
ErrorSeApiDeactivated, // 3016 the SE is temporary deactivated
```

```
ErrorSeApiNotDeactivated, // 3017 the SE is not deactivated
ErrorTimeNotSet, // 3018 the managed data/time in the Secure Element has not been updated
after the initialization of the SE API or a period of absence of current for the Secure
Element
ErrorCertificateExpired, // 3019 the certificate with the public key for the verification
of the appropriate type of log messages is expired
ErrorSecureElementDisabled, // 301A SE API functions are invoked although the Secure
Element has been disabled
ErrorUserNotAuthorized, // 301B the user who has invoked a restricted SE API function is
not authorized to execute this function
ErrorUserNotAuthenticated, // 301C the user who has invoked a restricted SE API function
has not the status "authenticated"
ErrorDescriptionNotSetByManufacturer, // 301D the function initialize has been invoked
without a value for the input parameter description although the description of the SE API
has not been set by the manufacturer
ErrorDescriptionSetByManufacturer, // 301E the function initialize has been invoked with a
value for the input parameter description although the description of the SE API has been
set by the manufacturer
ErrorExportSerialNumbersFailed, // 301F collection of the serial number(s) failed
ErrorGetMaxNumberOfClientsFailed, // 3020 determination of the maximum number of clients
that could use the SE API simultaneously failed
ErrorGetCurrentNumberOfClientsFailed, // 3021 determination of the current number of
clients using the SE API failed
ErrorGetMaxNumberTransactionsFailed, // 3022 determination of the maximum number of
transactions that can be managed simultaneously failed
ErrorGetCurrentNumberOfTransactionsFailed, // 3023 determination of the number of currently
opened transactions failed
ErrorGetSupportedUpdateVariantsFailed, // 3024 identification of the supported variant(s)
for updating transactions failed
ErrorDeleteStoredDataFailed, // 3025 deletion of the data from the storage failed
ErrorUnexportedStoredData, // 3026 deletion of data from the storage failed because the
storage contains data that has not been exported
ErrorSigningSystemOperationDataFailed, // 3027 determination of the log message parts for
the system operation data by the Secure Element failed
ErrorUserIdNotManaged, // 3028 userId is not managed by the SE API
ErrorUserIdNotAuthenticated, // 3029 userId has not the status authenticated
ErrorDisableSecureElementFailed // 302A deactivation of the Secure Element failed
```

```
public enum SePinStateFlags
{
   stateInitialized = 0,
   adminPinTransportState = 1,
   adminPukTransportState = 2,
   timeAdminPinTransportState = 4,
   timeAdminPukTransportState = 8
}
```

```
public enum SeSyncVariants : int // Represents the supported time formats of the TSE.
{
   noInput = 0,
   utcTime = 1,
   generalizedTime = 2,
   unixTime = 3
}
```

```
public enum SeUpdateVariants :
   int // Represents the variants that are supported by the Secure Element to update
transactions.
{
   signedUpdate = 0,
```

```
   unsignedUpdate,
   signedAndUnsignedUpdate
}
```

```
public enum SeAuthenticationResult : int // Represents the result of an authentication
attempt.
{
   authenticationOk = 0, // 0 indicates that the authentication has been successful.
   authenticationFailed, // 1 indicates that the authentication has failed.
   authenticationPinIsBlocked, // 2 indicates that the PIN entry for the userId was blocked
before the authentication attempt.
   authenticationUnknownUserId, // 3 indicates that the passed userId Is Not managed by the
SE API.
   authenticationError // 4 indicates that an error has occurred during the execution of
the function unblockUser.
}
```

```
public enum SeUnblockResult : int // Represents the result of the unblock process.
{
   unblockOk = 0, // 0 The value unblockOk SHALL indicate that the unblocking has been
successful.
   unblockFailed, // 1 The value unblockFailed SHALL indicate that the unblocking has
failed.
   unblockPinIsBlocked, // 2 The value unblockPinIsBlocked SHALL indicate that the PIN
entry for the PUK was blocked before the authentication attempt.
   unblockUnknownUserId, // 3 The value unblockUnknownUserId SHALL indicate that the passed
userId Is Not managed by the SE API.
}
```

```
public enum SeLCS : int // Represents the Life Cycle State of the TSE.
{
   lcsUnknown = 0, // 0 indicates an unknown LCS.
   lcsNotInitialized, // 1 indicates that TSE.Initialize() has not been called yet.
   lcsNoTime, // 2 indicates that the TSE time is not set.
   lcsActive, // 3 indicates that the TSE is ready for transactions.
   lcsDeactivated, // 4 indicates that TSE.DeactivateTSE() has been called.
   lcsDisabled // 5 indicates that TSE.DisableSecureElement() has been called.
}
```