15 Electronic Value Reader/Writer

15.1 General

This Chapter defines the Electronic Value Reader / Writer device category.

15.2 Summary

Properties (UML attributes)

Common	*Type*	*Mutability*	*Version*	*May Use After*
AutoDisable:	*boolean*	{ read-write }	1.12	open
CapCompareFirmwareVersion:	*boolean*	{ read-only }	1.12	open
CapPowerReporting:	*int32*	{ read-only }	1.12	open
CapStatisticsReporting:	*boolean*	{ read-only }	1.12	open
CapUpdateFirmware:	*boolean*	{ read-only }	1.12	open
CapUpdateStatistics:	*boolean*	{ read-only }	1.12	open
CheckHealthText:	*string*	{ read-only }	1.12	open
Claimed:	*boolean*	{ read-only }	1.12	open
DataCount:	*int32*	{ read-only }	1.12	open
DataEventEnabled:	*boolean*	{ read-write }	1.12	open
DeviceEnabled:	*boolean*	{ read-write }	1.12	open & claim
FreezeEvents:	*boolean*	{ read-write }	1.12	open
OutputID:	*int32*	{ read-only }	1.12	open
PowerNotify:	*int32*	{ read-write }	1.12	open
PowerState:	*int32*	{ read-only }	1.12	open
State:	*int32*	{ read-only }	1.12	--
DeviceControlDescription:	*string*	{ read-only }	1.12	--
DeviceControlVersion:	*int32*	{ read-only }	1.12	--
DeviceServiceDescription:	*string*	{ read-only }	1.12	open
DeviceServiceVersion:	*int32*	{ read-only }	1.12	open
PhysicalDeviceDescription:	*string*	{ read-only }	1.12	open
PhysicalDeviceName:	*string*	{ read-only }	1.12	open

Specific	*Type*	*Mutability*	*Version*	*May Use After*
CapActivateService:	*boolean*	{ read-only }	1.12	open
CapAdditionalSecurityInformation:	*boolean*	{ read-only }	1.15	open
CapAddValue:	*boolean*	{ read-only }	1.12	open
CapAuthorizeCompletion:	*boolean*	{ read-only }	1.15	open
CapAuthorizePreSales:	*boolean*	{ read-only }	1.15	open
CapAuthorizeRefund:	*boolean*	{ read-only }	1.15	open
CapAuthorizeVoid:	*boolean*	{ read-only }	1.15	open
CapAuthorizePreSales:	*boolean*	{ read-only }	1.15	open
CapCancelValue:	*boolean*	{ read-only }	1.12	open
CapCardSensor:	*int32*	{ read-only }	1.12	open
CapCashDeposit:	*boolean*	{ read-only }	1.15	open
CapCenterResultCode:	*boolean*	{ read-only }	1.15	open
CapCheckCard:	*boolean*	{ read-only }	1.15	open
CapDailyLog:	*int32*	{ read-only }	1.14	open
CapDetectionControl:	*int32*	{ read-only }	1.12	open
CapElectronicMoney:	*boolean*	{ read-only }	1.12	open
CapEnumerateCardServices:	*boolean*	{ read-only }	1.12	open
CapIndirectTransactionLog:	*boolean*	{ read-only }	1.12	open
CapInstallments:	*boolean*	{ read-only }	1.15	open
CapLockTerminal:	*boolean*	{ read-only }	1.12	open
CapLogStatus:	*boolean*	{ read-only }	1.12	open
CapMediumID:	*boolean*	{ read-only }	1.12	open
CapMembershipCertificate	*boolean*	{ read-only }	1.14.1	open
CapPaymentDetail:	*boolean*	{ read-only }	1.15	open
CapPINDevice:	*boolean*	{ read-only }	1.14	open
CapPoint:	*boolean*	{ read-only }	1.12	open
CapSubtractValue:	*boolean*	{ read-only }	1.12	open
CapTaxOthers:	*boolean*	{ read-only }	1.15	open
CapTrainingMode:	*boolean*	{ read-only }	1.14	open
CapTransaction:	*boolean*	{ read-only }	1.12	open
CapTransactionLog:	*boolean*	{ read-only }	1.12	open
CapTransactionNumber:	*boolean*	{ read-only }	1.15	open
CapUnlockTerminal:	*boolean*	{ read-only }	1.12	open
CapUpdateKey:	*boolean*	{ read-only }	1.12	open
CapVoucher:	*boolean*	{ read-only }	1.12	open
CapWriteValue:	*boolean*	{ read-only }	1.12	open

AccountNumber:	*string*	{ read-only }	1.12	open
AdditionalSecurityInformation:	*string*	{ read-write }	1.12	open
Amount:	*currency*	{ read-write }	1.12	open
ApprovalCode:	*string*	{ read-write }	1.12	open
AsyncMode:	*boolean*	{ read-write }	1.12	open
Balance:	*currency*	{ read-only }	1.12	open
BalanceOfPoint:	*currency*	{ read-only }	1.12	open
CardCompanyID:	*string*	{ read-only }	1.15	open
CardServiceList:	*string*	{ read-only }	1.12	open
CenterResultCode:	*string*	{ read-only }	1.15	open
CurrentService:	*string*	{ read-write }	1.12	open
DailyLog:	*string*	{ read-write }	1.15	open
DetectionControl:	*boolean*	{ read-write }	1.12	open
DetectionStatus:	*int32*	{ read-only }	1.12	open
ExpirationDate:	*string*	{ read-only }	1.12	open
LastUsedDate:	*string*	{ read-only }	1.12	open
LogStatus:	*int32*	{ read-only }	1.12	open
MediumID:	*string*	{ read-write }	1.12	open
PaymentCondition:	*int32*	{ read-only }	1.15	open
PaymentDetail:	*string*	{ read-only }	1.15	open
PaymentMedia:	*int32*	{ read-write }	1.15	open
PINEntry:	*int32*	{ read-write }	1.14	open
Point:	*currency*	{ read-write }	1.12	open
ReaderWriterServiceList:	*string*	{ read-only }	1.12	open
ServiceType	*int32*	{ read-only }	1.14.1	open
SequenceNumber:	*int32*	{ read-only }	1.12	open
SettledAmount:	*currency*	{ read-only }	1.12	open
SettledPoint:	*currency*	{ read-only }	1.12	open
SlipNumber:	*string*	{ read-only }	1.15	open
TrainingModeState	*int32*	{ read-write }	1.14	open
TransactionLog:	*string*	{ read-only }	1.12	open
TransactionNumber:	*string*	{ read-only }	1.15	open
TransactionType:	*int32*	{ read-only }	1.15	open
VoucherID:	*string*	{ read-write }	1.12	open
VoucherIDList:	*string*	{ read-write }	1.12	open

Methods (UML operations)

Common

Name	*Version*
open (logicalDeviceName: *string* **):** **void { raises-exception }**	1.12
close ( ): **void { raises-exception, use after open }**	1.12
claim (timeout: *int32* **):** **void { raises-exception, use after open }**	1.12
release ( ): **void { raises-exception, use after open, claim }**	1.12
checkHealth (level: *int32* **):** **void { raises-exception, use after open, enable }**	1.12
clearInput ( ): **void { }**	1.12
clearInputProperties ( ): **void { }**	1.12
clearOutput ( ): **void { }**	1.12
directIO (command: *int32*, **inout data:** *int32*, **inout obj:** *object* **):** **void { raises-exception, use after open }**	1.12
compareFirmwareVersion (firmwareFileName: *string*, **out result:** *int32* **):** **void { raises-exception, use after open, claim, enable }**	1.12
resetStatistics (statisticsBuffer: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.12
retrieveStatistics (inout statisticsBuffer: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.12
updateFirmware (firmwareFileName: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.12
updateStatistics (statisticsBuffer: *string* **):** **void { raises-exception, use after open, claim, enable }**	1.12

Specific

Name	
accessDailyLog (sequenceNumber: *int32*, **type:** *int32*, **timeout:** *int32* **):** **void { raises-exception, use after open, claim, enable }**	1.15
accessData (dataType: *int32*, **inout data:** *int32*, **inout obj:** *object* **):** **void { raises-exception, use after open, claim, enable }**	1.14.1
accessLog (sequenceNumber: *int32*, **type:** *int32*, **timeout:** *int32* **):** **void { raises-exception, use after open, claim, enable }**	1.12
activateEVService (inout data: *int32*, **inout obj:** *object* **):** **void { raises-exception, use after open, claim, enable }**	1.14.1
activateService (inout data: *int32*, **inout obj:** *object* **):** **void { raises-exception, use after open, claim, enable }**	1.12

addValue (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

authorizeCompletion (sequenceNumber: *int32*, **amount: currency, taxOthers: currency, timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

authorizePreSales (sequenceNumber: *int32*, **amount: currency, taxOthers: currency, timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

authorizeRefund (sequenceNumber: *int32*, **amount: currency, taxOthers: currency, timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

authorizeSales (sequenceNumber: *int32*, **amount: currency, taxOthers: currency, timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

authorizeVoid (sequenceNumber: *int32*, **amount: currency, taxOthers: currency, timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

authorizeVoidPreSales (sequenceNumber: *int32*, **amount: currency, taxOthers: currency, timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

beginDetection (type: *int32*, **timeout:** *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

beginRemoval (timeout: *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

cancelValue (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

captureCard ( ): 1.12
void { raises-exception, use after open, claim, enable }

cashDeposit (sequenceNumber: *int32*, **amount: currency, timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

checkCard (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.15
void { raises-exception, use after open, claim, enable }

checkServiceRegistrationToMedium(sequenceNumber: *int32*, **timeout:** *int32* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

clearParameterInformation ( ): 1.14
void { raises-exception, use after open, claim, enable }

closeDailyEVService (inout data: *int32*, **inout obj:** *object* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

deactivateEVService (inout data: *int32*, **inout obj:** *object* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

endDetection ( ): 1.12
void { raises-exception, use after open, claim, enable }

endRemoval ( ): 1.12
void { raises-exception, use after open, claim, enable }

enumerateCardServices ( ): 1.12
void { raises-exception, use after open, claim, enable }

lockTerminal ( ): 1.12
void { raises-exception, use after open, claim, enable }

openDailyEVService (inout data: *int32*, **inout obj:** *object* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

queryLastSuccessfulTransactionResult ( ): 1.14
void { raises-exception, use after open, claim, enable }

readValue (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

registerServiceToMedium (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

retrieveResultInformation (name: *string,* **inout value:** *string* **):** 1.14
void { raises-exception, use after open, claim }

setParameterInformation (name: *string,* **value:** *string* **):** 1.14
void { raises-exception, use after open, claim }

subtractValue (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

transactionAccess (control: *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

unlockTerminal ( ): 1.12
void { raises-exception, use after open, claim, enable }

unregisterServiceToMedium (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

updateData (dataType: *int32*, **inout data:** *int32*, **inout obj:** *object* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

updateKey (inout data: *int32*, **inout obj:** *object* **):** 1.12
void { raises-exception, use after open, claim, enable }

writeValue (sequenceNumber: *int32*, **timeout:** *int32* **):** 1.12
void { raises-exception, use after open, claim, enable }

openDailyEVService (inout data: *int32*, **inout obj:** *object* **):** 1.14.1
void { raises-exception, use after open, claim, enable }

Events (UML interfaces)

Name	*Type*	*Mutability*	*Version*
upos::events::DataEvent			1.12
Status:	*int32*	{ read-only }	
upos::events::DirectIOEvent			1.12
EventNumber:	*int32*	{ read-only }	
Data:	*int32*	{ read-write }	
Obj:	*object*	{ read-write }	
upos::events::ErrorEvent			1.12
ErrorCode:	*int32*	{ read-only }	
ErrorCodeExtended:	*int32*	{ read-only }	
ErrorLocus:	*int32*	{ read-only }	
ErrorResponse:	*int32*	{ read-write }	
upos::events::OutputCompleteEvent			1.12
OutputID:	*int32*	{ read-only }	
upos::events::StatusUpdateEvent			1.12
Status:	*int32*	{ read-only }	
upos::events::TransitionEvent			1.14
EventNumber:	*int32*	{ read-only }	
pData:	*int32*	{ read-write }	
pString:	*string*	{ read-write }	

15.3 General Information

The Electronic Value Reader / Writer programmatic name is “ElectronicValueRW.”

This device was introduced in Version 1.12 of the specification.

Electronic value is defined as a collection of services such as electronic money, points, and voucher/ticket, maintained on a contact-less or contact IC card (this is referred to as ‘card’ in the following sections). The Electronic Value Reader / Writer device is a device that offers the capability to hold the settlement addition, subtraction, setting, and reading electronically.

The electronic money service supports the post-paid type electronic money settlement, pre-paid type electronic money settlement, the credit card settlement, and the debit card settlement.

The point service maintains (can add or subtract) points directly on the card. Alternatively, the points may be stored in another location and only a reference is maintained on the card.

The voucher/ticket service maintains two or more identifiers that validate the card holder. The card holder can receive and exchange the value at any time. The service provider can provide value to the card holder at its discretion.

15.3.1 Capabilities

The Electronic Value Reader / Writer (EVR/W) has the following set of capabilities:

• Access the card for the settlement.

• Read/write the content of electronic value that can be used for the settlement from the card.

• Execute the settlement service using electronic value.

• Accumulate the result of the settlement in the device as a log.

15.3.2 Added in Release 1.14

The following functionality was added for **Release 1.14**.

The EVR/W specification up to release 1.13 did not define the syntax and semantics of the settlement information specified as a device or service. Each device has the ability to define the syntax of the settlement information in the **AdditionalSecurityInformation** property. Release 1.14 adds the syntax and semantics necessary to convey the settlement information which previously was available only through the **DirectIO** method and event structures. This hindered compatibility and with the following properties, methods, and events serves to rectify this shortcoming.

In addition to updates to the device category, the following Properties, Methods, and Events are added:

• A **CapPINDevice** property to indicated if the EVR/W is equipped with a PIN pad entry device.

• A **CapTrainingMode** property to indicated if the EVR/W supports an operator training function mode.

• A **PINEntry** property which defines the PIN functionality supported by the EVR/W device.

• A **TrainingModeState** property which provides information if the device is in training mode or run mode.

• A **clearParameterInformation** method to clear all device tag values.

• A **queryLastSuccessfulTransactionResult** method that is used to refresh the property values from the last device function operation.

• A **retrieveResultInformation** method that associates a tag name with a data value that is read.

• A **setParameterInformation** method that is used to associate a tag name with additional data value parameters for a card.

• A **TransitionEvent** which is a new event only for the EVR/W device in order to support communicating asynchronous I/O operation status between the application and the EVR/W device.

In addition to updates to the device category, the following Properties were updated:

• The **MediumID** property which is used to specify unique information about the card.

• The **SettledAmount** property which contains the real amount of the settlement by the electronic money service.

15.3.3 Added in Release 1.14.1

After the release of 1.14, additional changes were required based upon extensive testing of the updated specification. These include the following:

• Updated the Model to include new services: Point, Voucher/Ticket, Membership Certificate, and Common along with their service capabilities and corresponding methods dependability.

• Addition of a description of the Life cycle of a Sub-Service.

• Addition of description of the variations of the service dependent upon behavior of a store or a location.

• Addition of description of how the EVR/W device interacts with a payment center.

• Added an updated Error model that more completely describes the EVR/W error conditions and reporting structure.

• Added the **CapMembershipCertificate** capability property.

• Updated the **CardServiceList** property variations description.

• Updated the **CurrentService** property variations description.

• Added the **ServiceType** property.

• Updated the **ReaderWriterServiceList** property variations description.

• Added the **accessData** method.

• Updated the **accessLog** method consistency information.

• Added the **activateEVService** method.

• Added the **checkServiceRegistrationToMedium** method.

• Added the **closeDailyEVService** method.

• Added the **deactivateEVService** method.

• Updated the **lockTerminal** method.

• Added the **openDailyEVService** method.

• Added the **registerServiceToMedium** method.

• Updated the **retrieveResultInformation** method by additional tags and values and enumeration tag values.

• Updated the **unlockTerminal** method with changes to the Remarks section.

• Added the **unregisterServiceToMedium** method.

• Added the **updateData** method.

• Updated the **updateKey** method.

• Updated the **TransitionEvent** by adding two new event type identifiers.

• Corrected formatting issues throughout the chapter.

15.3.4 Added in Release 1.15

In order to support devices supporting credit payment function, version 1.15 included the CAT specification in the electronic value reader / writer specification.

The following added properties and methods conform to the CAT specification, so please refer to the description of the CAT device specification.

• Added the **CapAdditionalSecurityInformation** capability property.

• Added the **CapAuthorizeCompletion** capability property.

• Added the **CapAuthorizePreSales** capability property.

• Added the **CapAuthorizeRefund** capability property.

• Added the **CapAuthorizeVoid** capability property.

• Added the **CapAuthorizeVoidPreSales** capability property.

• Added the **CapCashDeposit** capability property.

• Added the **CapCenterResultCode** capability property.

• Added the **CapCheckCard** capability property.

• Added the **CapDailyLog** capability property.

• Added the **CapInstallments** capability property.

• Added the **CapPaymentDetail** capability property.

• Added the **CapTaxOthers** capability property.

• Added the **CapTransactionNumber** capability property.

• Added the **CardCompanyID** property.

• Added the **CenterResultCode** property.

• Added the **DailyLog** property.

• Added the **LogStatus** property.

• Added the **PaymentCondition** property.

• Added the **PaymentDetail** property.

• Added the **PaymentMedia** property.

• Added the **SlipNumber** property.

• Added the **TransactionNumber** property.

• Added the **TransactionType** property.

• Added the **accessDailyLog** method.

• Added the **authorizeCompletion** method.

• Added the **authorizePreSales** method.

• Added the **authorizeRefund** method.

• Added the **authorizeSales** method.

• Added the **authorizeVoid** method.

• Added the **authorizeVoidPreSales** method.

• Added the **cashDeposit** method.

• Added the **checkCard** method.

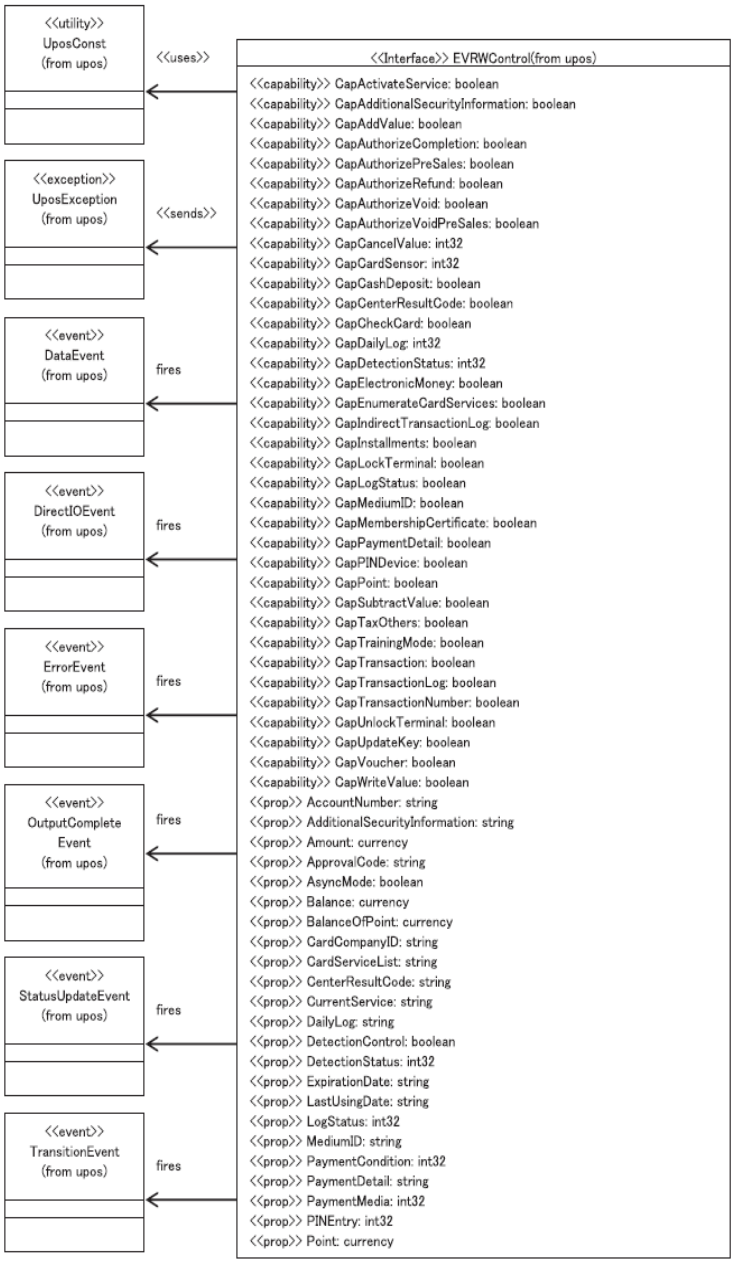
The TrainingMode property of the CAT specification corresponds to the TrainingModeState property defined in the electronic value reader / writer specification. To deal with credit processing, the following tag definitions and TransitionEvent event definitions have been added.

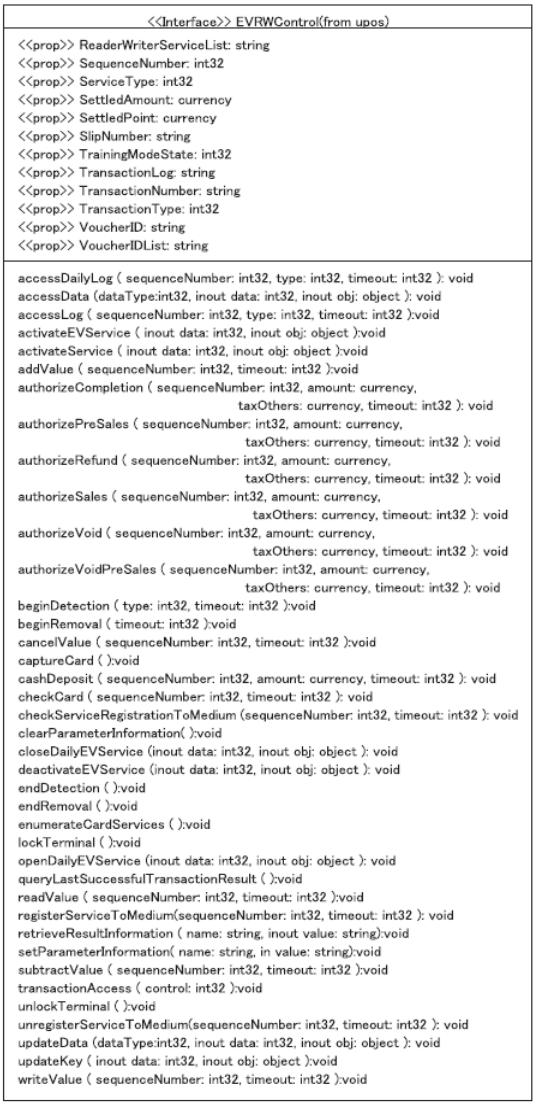
• Updated the **retrieveResultInformation** method by adding additional tags, values and enumeration tag values.

• Updated the **TransitionEvent** by adding five new event type values.

15.3.5 EVRW Class Diagram

The following diagram shows the relationships between the EVR/W classes. ***Updated in Release 1.15***




<<Interface>> EVRWControl(from upos)
<<prop>> ReaderWriterServiceList: string
<<prop>> SequenceNumber: int32
<<prop>> ServiceType: int32
<<prop>> SettledAmount: currency
<<prop>> SettledPoint: currency
<<prop>> SlipNumber: string
<<prop>> TrainingModeState: int32
<<prop>> TransactionLog: string
<<prop>> TransactionNumber: string
<<prop>> TransactionType: int32
<<prop>> VoucherID: string
<<prop>> VoucherIDList: string
accessDailyLog (sequenceNumber: int32, type: int32, timeout: int32): void
accessData (dataType:int32, inout data: int32, inout obj: object): void
accessLog (sequenceNumber: int32, type: int32, timeout: int32):void
activateEVService (inout data: int32, inout obj: object):void
activateService (inout data: int32, inout obj: object):void
addValue (sequenceNumber: int32, timeout: int32):void
authorizeCompletion (sequenceNumber: int32, amount: currency,
taxOthers: currency, timeout: int32): void
authorizePreSales (sequenceNumber: int32, amount: currency,
taxOthers: currency, timeout: int32): void
authorizeRefund (sequenceNumber: int32, amount: currency,
taxOthers: currency, timeout: int32): void
authorizeSales (sequenceNumber: int32, amount: currency,
taxOthers: currency, timeout: int32): void
authorizeVoid (sequenceNumber: int32, amount: currency,
taxOthers: currency, timeout: int32): void
authorizeVoidPreSales (sequenceNumber: int32, amount: currency,
taxOthers: currency, timeout: int32): void
beginDetection (type: int32, timeout: int32):void
beginRemoval (timeout: int32):void
cancelValue (sequenceNumber: int32, timeout: int32):void
captureCard ( ):void
cashDeposit (sequenceNumber: int32, amount: currency, timeout: int32): void
checkCard (sequenceNumber: int32, timeout: int32): void
checkServiceRegistrationToMedium (sequenceNumber: int32, timeout: int32): void
clearParameterInformation( ):void
closeDailyEVService (inout data: int32, inout obj: object): void
deactivateEVService (inout data: int32, inout obj: object): void
endDetection ( ):void
endRemoval ( ):void
enumerateCardServices ( ):void
lockTerminal ( ):void
openDailyEVService (inout data: int32, inout obj: object): void
queryLastSuccessfulTransactionResult ( ):void
readValue (sequenceNumber: int32, timeout: int32):void
registerServiceToMedium(sequenceNumber: int32, timeout: int32): void
retrieveResultInformation (name: string, inout value: string):void
setParameterInformation(name: string, in value: string):void
subtractValue (sequenceNumber: int32, timeout: int32):void
transactionAccess (control: int32):void
unlockTerminal ( ):void
unregisterServiceToMedium(sequenceNumber: int32, timeout: int32): void
updateData (dataType:int32, inout data: int32, inout obj: object): void
updateKey (inout data: int32, inout obj: object):void
writeValue (sequenceNumber: int32, timeout: int32):void

15.3.6 Model

The EVR/W supports the following services and methods.

Services	Service Capabilities	Corresponding Methods
Common	Deploy	**activateEVService** method
	Open	**openDailyEVService** method
	Maintenance	**accessData** method **updateData** method **accessLog** method **updateKey** method
	Close	**closeDailyEVService** method
	Remove	**deactivateEVService** method
Electronic Money	Balance Inquiry	**readValue** method **Balance** property
	Payment	**subtractValue** method **Amount** property **SettledAmount** property
	Deposit	**addValue** method **Amount** property **SettledAmount** property
	Cancel	**cancelValue** method **ApprovalCode** property
Membership certificate	Registering service to medium	**registerServiceToMedium** method **checkServiceRegistrationToMedium** method
	Unregistering service to medium	**unregisterServiceToMedium** method
	Inquiry	**readValue** method
	Updating	**writeValue** method

Services	Service Capabilities	Corresponding Methods
Point	Registering service to medium	**registerServiceToMedium** method **checkServiceRegistrationToMedium** method **Point** property
	Unregistering service to medium	**unregisterServiceToMedium** method
	Inquiry	**readValue** method **BalanceOfPoint** property
	Deposit	**addValue** method **Point** property **SettledPoint** property
	Redeem	**subtractValue** method **Point** property **SettledPoint** property
	Updating	**writeValue** method **Point** property
	Cancel	**cancelValue** method **ApprovalCode** property
Voucher/Ticket	Registering service to medium	**registerServiceToMedium** method **checkServiceRegistrationToMedium** method
	Unregistering service to medium	**unregisterServiceToMedium** method
	Inquiry/ Enumeration	**readValue** method **VoucherIDList** property
	Issue	**addValue** method **VoucherID** property
	Redeem	**subtractValue** method **VoucherID** property

The general model of the EVR/W is as follows:

Input Model

The **readValue** method follows the UnifiedPOS Input model.

When the application is ready to receive the data from the EVR/W, the **readValue** method is called. Then, when input data is received, a **DataEvent** event is enqueued. When the application sets the **DataEventEnabled** property to true, the **DataEvent** event will be delivered to the application.

If an error occurs while reading the data, an **ErrorEvent** is enqueued instead of the **DataEvent**. When the application sets the **DataEventEnabled** property to true, the **ErrorEvent** event will be delivered to the application.

The application can obtain the number of enqueued data events by reading the **DataCount** property.

If **AutoDisable** is true, then the device is automatically disabled when a **DataEvent** is enqueued.

All input data that is queued can be cleared by executing the **clearInput** method.

Output Model

The **accessLog**, **addValue**, **cancelValue**, **subtractValue**, **transactionAccess**, and **writeValue** methods can be executed asynchronously or synchronously depending on the value of the **AsyncMode** property as defined by the UnifiedPOS output model.

When **AsyncMode** is true, methods cannot be issued immediately after issuing a prior method; only one outstanding asynchronous method is allowed at a time. However, **clearOutput** is an exception because its purpose is to cancel an outstanding asynchronous method.

When asynchronous processing completes, an **OutputCompleteEven**t is delivered to the application.

Support of Sub-Service Use

When one EVR/W provides two or more electronic value services, and an
EVR/W Service corresponding to each service provider exists, then they can be used as sub-service.

If the **open** method is executed, the **open** method of all sub-services is called, and the sub-service is enumerated by the **ReaderWriterServiceList** property. The **close**, **claim**, and **release** methods operate in the same manner on all the sub-services.

The application selects the sub-service to be used by setting the **CurrentService** property. All method and property operations thereafter effect that sub-service.

CAT Device used for the EVR/W device: *Added in Release 1.15*

• The general model for the CAT control used for the EVR/W device is shown below:

• The CAT control used for the EVR/W device basically follows the output device model. However, multiple methods cannot be issued for asynchronous output; only one outstanding asynchronous request is allowed.

• The CAT control used for the EVR/W device issues requests to the EVR/W device for different types of authorization by invoking the following methods.

Function	Method name	Corresponding **Cap** property
Purchase	**authorizeSales**	None
Cancel Purchase	**authorizeVoid**	**CapAuthorizeVoid**
Refund Purchase	**authorizeRefund**	**CapAuthorizeRefund**
Authorization Completion	**authorizeCompletion**	**CapAuthorizeCompletion**
Pre-Authorization	**authorizePreSales**	**CapAuthorizePreSales**
Cancel Pre-Authorization	**authorizeVoidPreSales**	**CapAuthorizeVoidPreSales**

• The CAT control used for the EVR/W device issues requests to the EVR/W device for special processing local to the EVR/W device by invoking the following methods.

Function	Method name	Corresponding **Cap** property
Card Check	**checkCard**	**CapCheckCard**
Daily log	**accessDailyLog**	**CapDailyLog**

• The CAT control used for the EVR/W device stores the authorization results in the following properties when an authorization operation successfully completes:

Description	Property Name	Corresponding **Cap** Property
Credit Account number	**AccountNumber**	None
Additional information	**AdditionalSecurityInformation**	**CapAdditionalSecurityInformation**
Approval code	**ApprovalCode**	None
Card company ID	**CardCompanyID**	None
Cod from the approval agency	**CenterResultCode**	**CapCenterResultCode**
Payment condition	**PaymentCondition**	None
Payment detail	**PaymentDetail**	**CapPaymentDetail**
Sequence number	**SequenceNumber**	None
Slip number	**SlipNumber**	None
Center transaction number	**TransactionNumber**	**CapTransactionNumber**
Transaction type	**TransactionType**	None

• The accessDailyLog method sets the following property

Description	Property Name	Corresponding **Cap** Property
Daily log	**DailyLog**	**CapDailyLog**

Electronic Money Device: ***Added in Release 1.9***

• The CAT Control used for the EVR/W device requires the Electronic Money Device to track each settlement and closing in the **DealingLog**.

Function	Method name	Corresponding **Cap** property
Settlement	**authorizeSales**	None
Charge	**cashDeposit**	**CapCashDeposit**
Inquiry for the balances	**checkCard**	**CapCheckCard**
Closing **DealingLog**	**accessDailyLog**	**CapDailyLog**
Setting security lock	**lockTerminal**	**CapLockTerminal**
Releasing security lock	**unlockTerminal**	**CapUnlockTerminal**

• When the CAT Control used for the EVR/W device receives the settlement results from the Electronic Money Device it stores these results in the following properties:

Description	Property Name	Corresponding **Cap** Property
Card ID	**AccountNumber**	None
Additional information	**AdditionalSecurityInformation**	**CapAdditionalSecurityInformation**
Approval code	**ApprovalCode**	None
Settled amount	**SettledAmount**	None
Balance	**Balance**	None
Sequence number	**SequenceNumber**	None
Transaction type	**TransactionType**	None

• The **accessDailyLog** method sets the following property.

Description	Property Name	Corresponding **Cap** Property
DealingLog	**DailyLog**	**CapDailyLog**

• Sequence numbers are used to validate that the properties set at completion of a method are indeed associated with the completed method. An incoming **SequenceNumber** argument for each method is compared with the resulting **SequenceNumber** property after the operation associated with the method has completed. If the numbers do not match, or if an application fails to identify the number, there is no guarantee that the values of the properties listed in the two tables correspond to the completed method.

• The **AsyncMode** property determines if methods are run synchronously or asynchronously.

• When **AsyncMode** is false, methods will be executed synchronously and their corresponding properties will contain data when the method returns.

• When **AsyncMode** is true, methods will return immediately to the application. When the operation associated with the method completes, each corresponding property will be updated by the CAT control used for the EVR/W device prior to an **OutputCompleteEvent**. When **AsyncMode** is true, methods cannot be issued immediately after issuing a prior method; only one outstanding asynchronous method is allowed at a time. However, **clearOutput** is an

exception because its purpose is to cancel an outstanding asynchronous method.

The methods supported and their corresponding properties vary depending on the CAT control used for the EVR/W device implementation. Applications should verify that particular **Cap** properties are supported before utilizing the capability dependent methods and properties.

• Results of synchronous calls to methods and writable properties will be stored in **ErrorCode**. Results of asynchronous processing will be indicated by an **OutputCompleteEvent** or returned in the **Errorcode** argument of an **ErrorEvent**. If **ErrorCode** or the **ErrorCode** argument is E_EXTENDED, detailed device specific information may be stored to
ErrorCodeExtended in synchronous mode and stored to **ErrorEvent** argument **ErrorCodeExtended** in asynchronous mode. The error code from the approval agency will be stored in **CenterResultCode** in either mode.

• An outstanding asynchronous method can be canceled via the **clearOutput** method.

• The Daily log can be collected by the **accessDailyLog** method. Collection will be run either synchronously or asynchronously according to the value of **AsyncMode.**

• Following is the general usage sequence of the CAT control.

Synchronous Mode:

- **open**
- **claim**
- **setDeviceEnabled** (true)
- Definition of the argument SequenceNumber
- Set **PaymentMedia** ***Added in Version 1.5***
- **authorizeSales**()
- Check UposException of the authorizeSales method
- Verify that the **SequenceNumber** property matches the value of the **authorizeSales**() *sequenceNumber* argument
- Access the properties set by **authorizeSales**()
- **setDeviceEnabled** (false)
- **release**
- **Close**

Asynchronous Mode:

- **open**
- **claim**
- **setDeviceEnabled** (true)
- **setAsyncMode** (true)
- Definition of the argument *SequenceNumber*
- Set **PaymentMedia** ***Added in Version 1.5***
- **authorizeSales**()
- Check UposException of the authorizeSales method
- Wait for **OutputCompleteEvent**
- Check the argument *ErrorCode*
- Verify that the **SequenceNumber** property matches the value of the **authorizeSales**() SequenceNumber argument
- Access the properties set by **authorizeSales**()

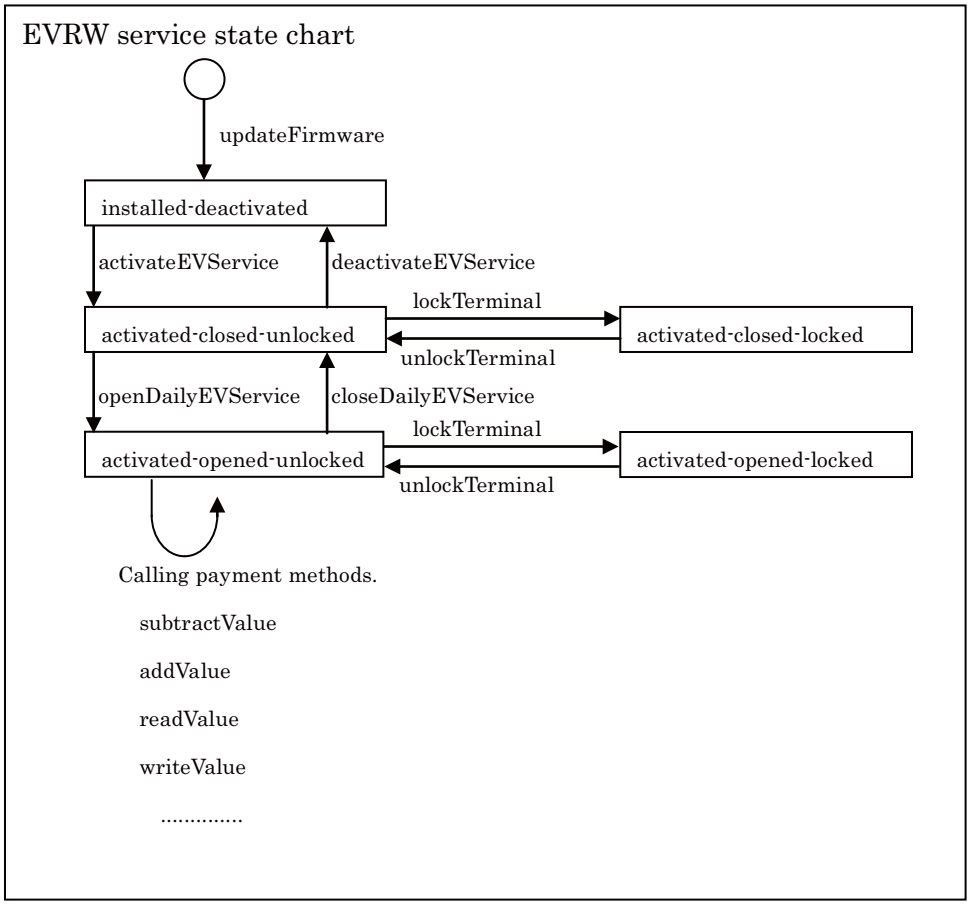
- **setDeviceEnabled** (false)

- **release**

- **close**

15.3.7 Life Cycle of Sub-Service *Added in Release 1.14.1*

The life cycle of a Sub-Service is illustrated below.



• Installed-deactivated state:
It is in the state which is invoked by the **updateFirmware** method and is not activated by **activateEVService** method.

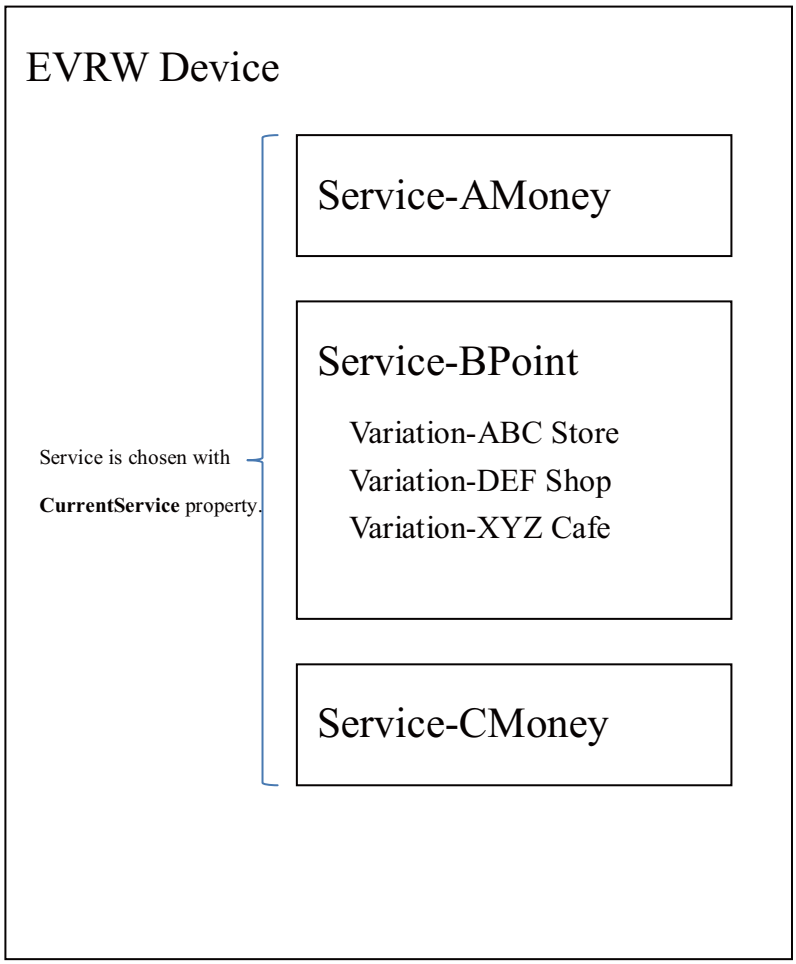
• Activated-closed-unlocked state:
It is in the state where Sub-Service was activated by the **activateEVService** method. In order to use Sub-Service, it is necessary to open by the **openDailyEVService** method.

• Activated-opened-unlocked state:
It is in the state where the Sub-Service was opened by the **openDailyEVService** method.

• Activated-closed-locked/activated-opened-locked state:
It is in the state where Sub-Service was locked by the **lockTerminal** method. In order to unlock Sub-Service, it is necessary to use the **unlockTerminal** method.

15.3.8 The Service with Variations *Added in Release 1.14.1*

The service can have variations depending upon the store or location which can alter the services required behavior.

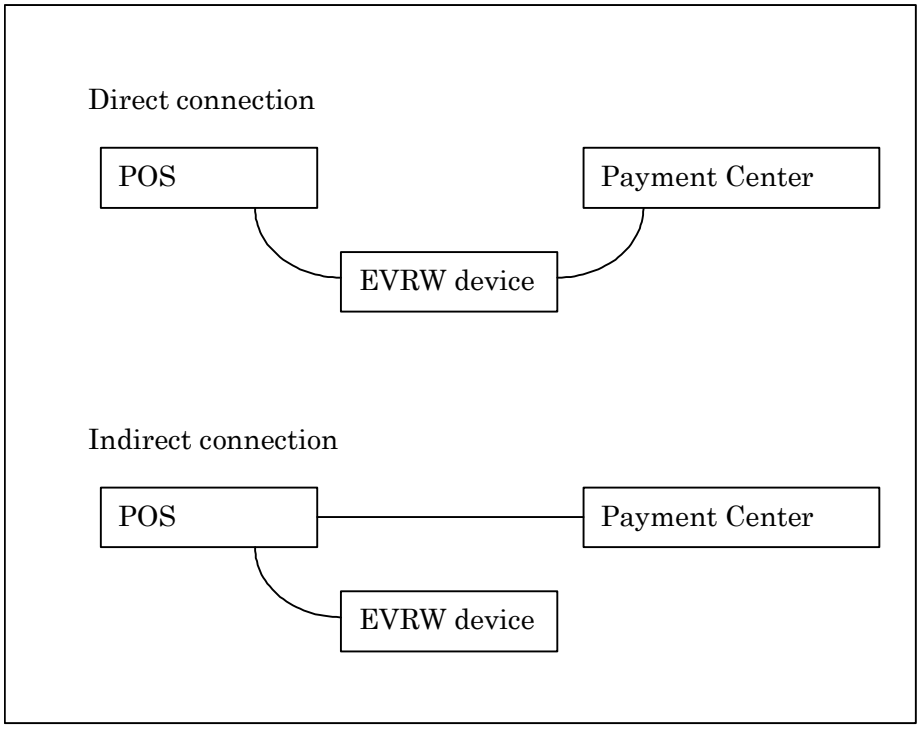


15.3.9 The Connection Model of EVR/W Devices and Payment Center

Added in Release 1.14.1

There are two ways of connecting an EVR/W device to a payment center.

Method	Definition
Direct Connection	The EVR/W device is directly connected to the Payment Center.
Indirect Connection	The EVR/W device is connected through a POS system to the Payment Center.



15.3.10 Transaction Mode Support

Transaction mode is comprised of multiple method calls and property accesses. Operations that can be included in the batch processing is a invocation of the writeValue, addValue, subtractValue, and cancelValue methods and all properties. When these methods are executed in transaction mode, their validation is confirmed first. If it is valid, the operation is added to the transaction mode buffer prior to execution. No update has yet been performed to the card.

Executing the **transactionAccess** method with a *control* value of EVRW_TA_NORMAL will cause all buffered commands to be processed.

The **AsyncMode** property also influences the execution of the transaction mode.

If the transaction is processed synchronously and an exception is not raised, then the entire transaction process was successful. If the transaction is processed asynchronously, then the asynchronous process rules listed above are followed. If an error occurs and the Error Event handler causes a retry, the entire transaction is retried.

15.3.11 Device Sharing

The EVR/W is an exclusive-use device, as follows:

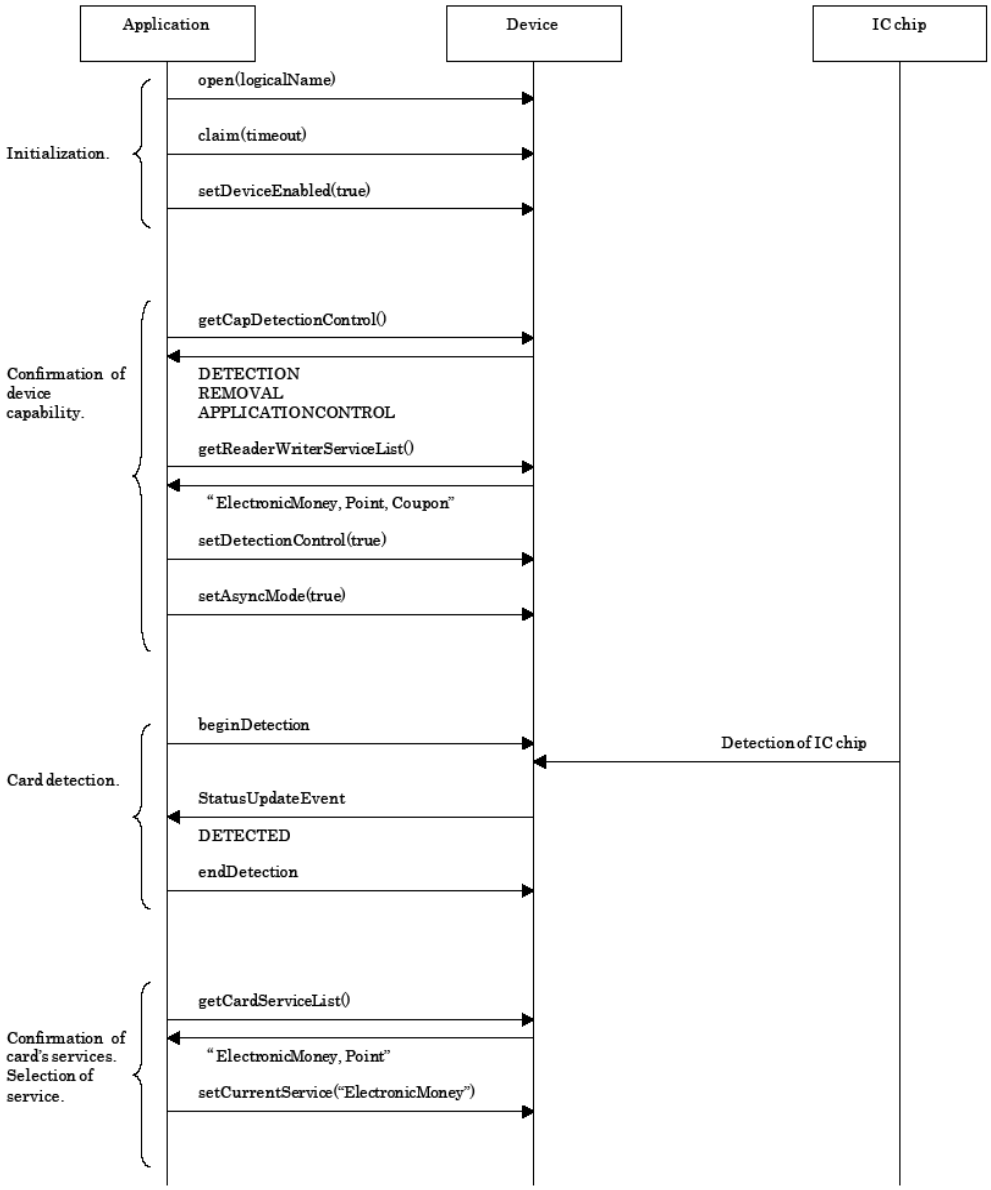
• The application must claim the device before enabling it.

• The application must claim and enable the device before calling methods that manipulate the device.

See the “Summary” table for precise usage prerequisites.

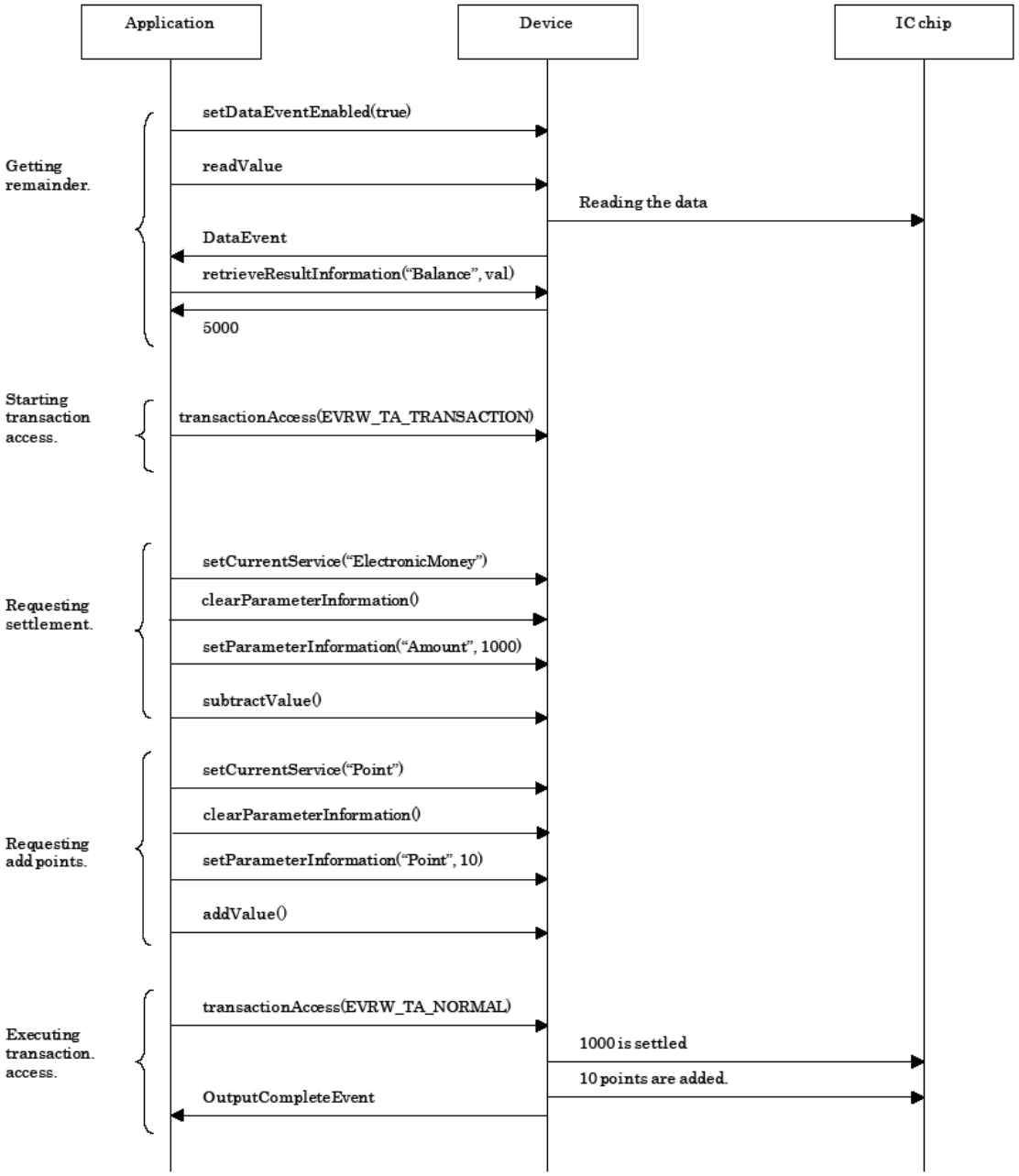
15.3.12 EVRW Sequence Diagram

The following sequence diagram shows the typical usage of the EVR/W device. ***Updated in Release 1.14.1***



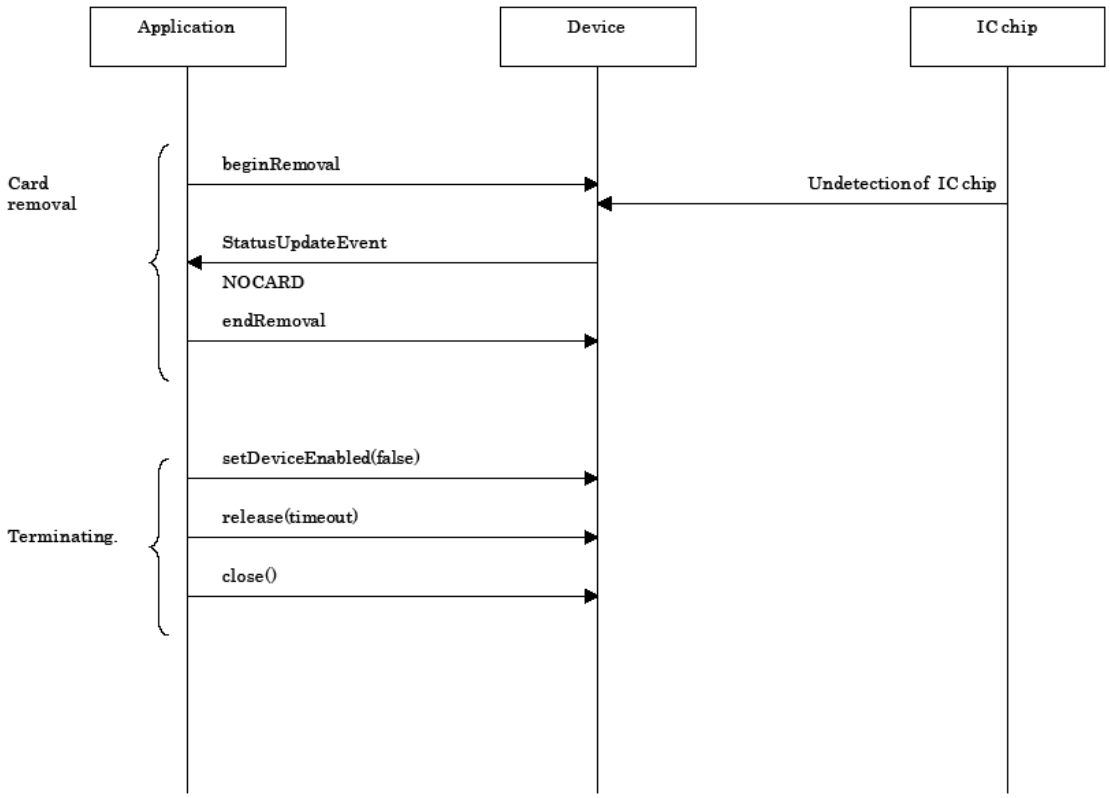
The following sequence diagram shows the continuation of the typical usage of the EVR/W device.

Updated in Release 1.14.1



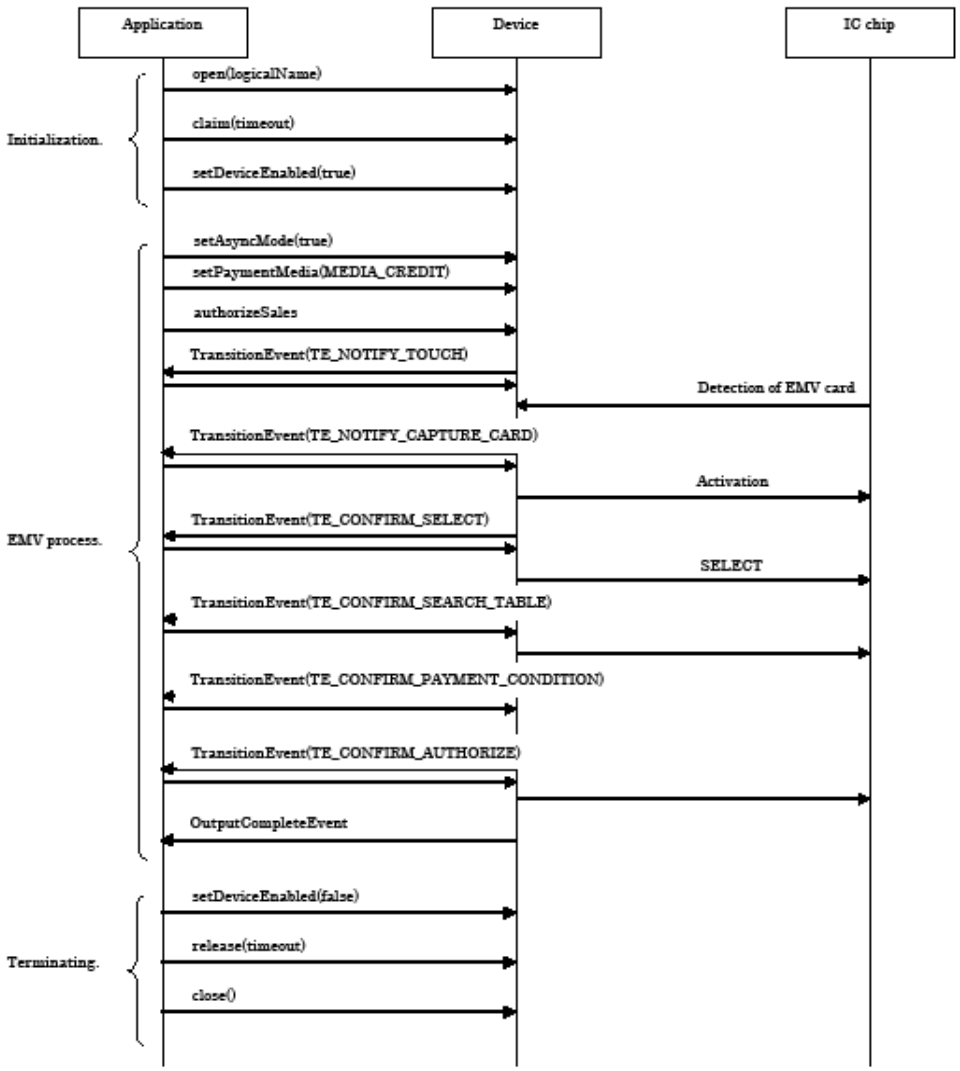
The following sequence diagram shows the continuation of the typical usage of the EVR/W device.

Updated in Release 1.14.1



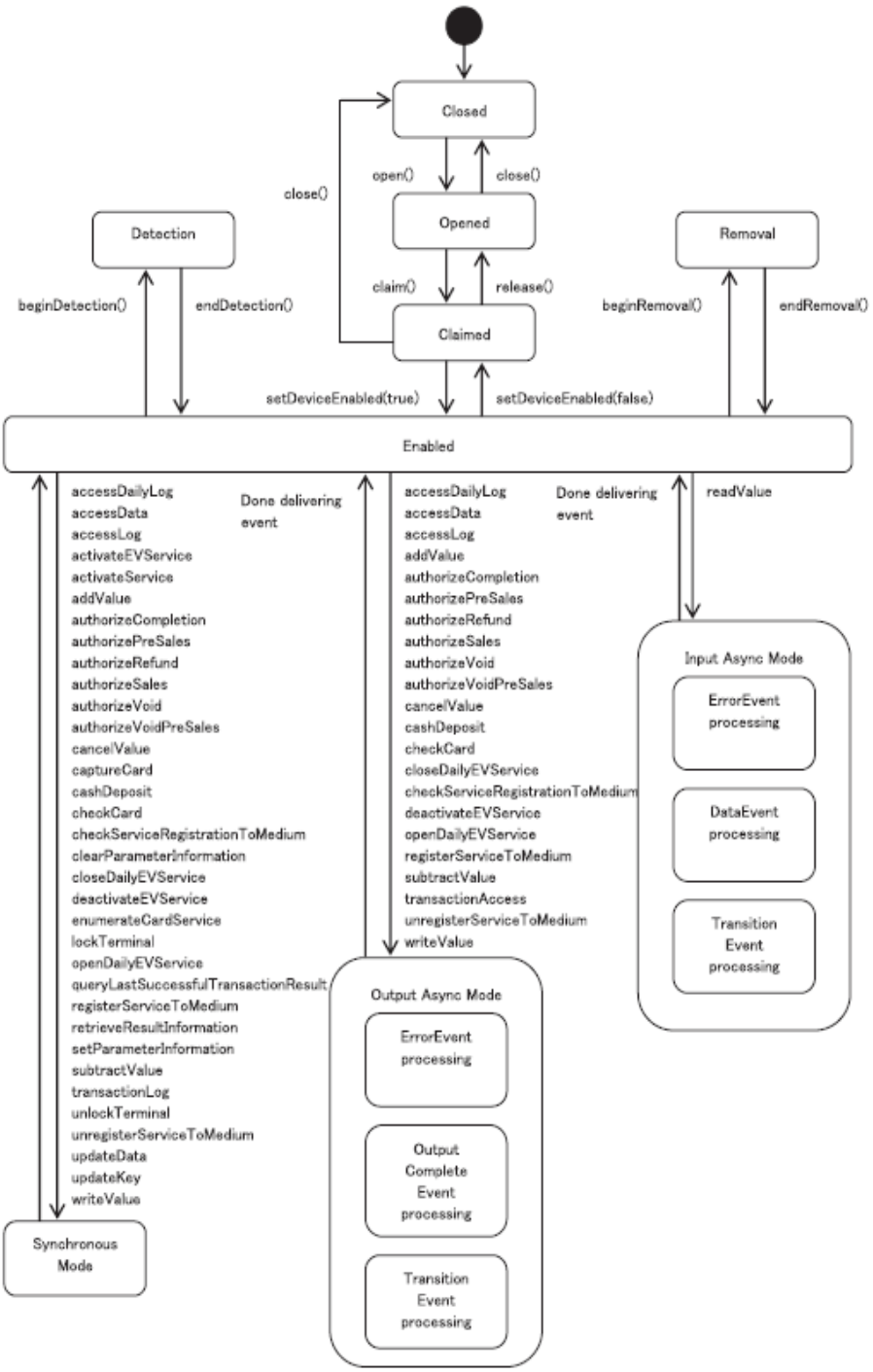
The following sequence diagram shows the CAT(EMV) usage that is used as EVR/W device.

Updated in Release 1.15



15.3.13 EVRW State Diagram

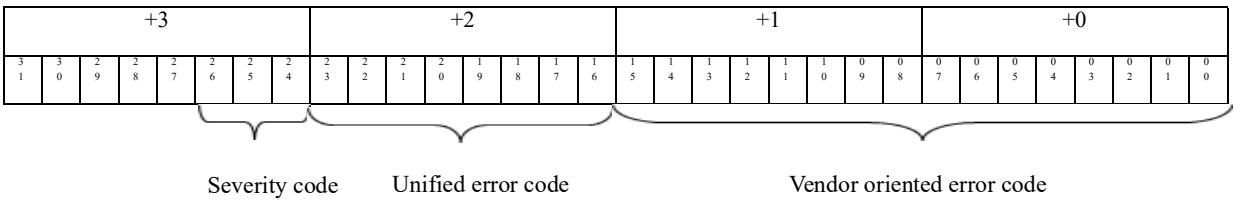
The following state diagram depicts the EVR/W device model. ***Updated in Release 1.15***



15.3.14 Error Model *Updated in Release 1.14.1*

The EVR/W error reporting model is as follows:

Most of the EVR/W device error conditions are reported by setting the UposException's (or ErrorEvent's) **ErrorCode** to E_EXTENDED and then setting **ErrorCodeExtended** as indicated in the following tables.



Bit assign	Size	Item	Description
31 – 27	5	Undefined	
26 – 24	3	Severity Code	Severity of the error condition.
23 – 16	8	Unified error code	Error code which defined by UPOS specification
15-0	16	Vendor oriented error code	Error code which oriented by vendor

Severity code indicates the severity condition and operation recovered from the error condition.

No.	Value	Description	Remarks
0	NORMAL	No need to recover	
1	BLOCKED	Need to recover by maintenance engineer	May need to replace the device
2	RECOVERABLE	Recoverable state which can be recovered by retrying with changing condition.	Ex) Operation timeout
3	RECOVERABLE_ASK_CARDHOLDER	Recoverable state which can be recovered by retrying with changing condition which the card holder determines.	Deficiency Transaction incomplete Over deposit
4	RECOVERABLE_ASK_OPERATOR	Recoverable state which can be recovered by retrying with changing condition which the POS operator determines.	Log full Mode mismatch

Unified error code indicates the type of error condition.

Value	Item	Description
EEVRW_ABORTED	Canceling from POS.	Transaction was aborted by the request from POS.
EEVRW_DEFICIENT	Amount is deficient.	Transaction cannot perform because the balance is insufficient.
EEVRW_DETECTION_ TIMEOUT	Medium detection timeout.	Medium could not be detected within the specified time.
EEVRW_HOST_ CANNOT_CLOSE	Payment center cannot close.	Transaction cannot perform because the payment center cannot close.
EEVRW_HOST_ CANNOT_OPEN	Payment center cannot open.	Transaction cannot perform because the payment center cannot open.
EEVRW_HOST_ CANNOT_OPERATE	The error occurred in payment center.	Transaction cannot perform because the error occurred in the payment center.
EEVRW_HOST_ REFUSAL	Transaction is refused by the payment center.	Transaction cannot perform because the request from transaction is refused by the payment center.
EEVRW_IN_PROGRESS	Transaction is in progress.	Transaction was already progressing and it was not able to perform the request.
EEVRW_INVALID_ MEDIUM	Invalid medium is detected.	Transaction cannot perform because invalid medium is detected.
EEVRW_INVALID_ MEDIUM_ABORTED	The error occurred in medium.	Transaction cannot perform because the error occurred in medium.
EEVRW_INVALID_ MEDIUM_ABORTED_ EXISTS	The error occurred in medium.	Transaction cannot perform because the service is already existing in medium.
EEVRW_INVALID_ MEDIUM_ABORTED_ NOSERVICE	The error occurred in medium.	Transaction cannot perform because the service is not present in medium.
EEVRW_INVALID_ MEDIUM_ABORTED_ NOSPACE	The error occurred in medium.	Transaction cannot perform because there is not enough memory space in medium.
EEVRW_INVALID_ MEDIUM_EXPIRED	Medium has expired.	Transaction cannot perform because medium has expired.
EEVRW_LOG_ OVERFLOW	Transaction log overflowed.	Transaction cannot perform because transaction log overflowed.
EEVRW_MEDIUM_ CANNOT_AUTHORIZE	Medium cannot authorize.	Medium detected by EVR/W cannot authorize.
EEVRW_MESSAGE_ FORMAT	Message format is invalid.	Transaction cannot perform because the message format is invalid.

EEVRW_OVERDEPOSIT	The balance after charging is exceeding a amount limit.	Transaction cannot perform because the balance after charging is exceeding a amount limit.
EEVRW_OVERDEPOSIT_TO_POINT	The point balance after adding is exceeding a amount limit.	Transaction cannot perform because the point balance after adding is exceeding a amount limit.
EEVRW_PAYMENT_RESTRICTION	Transaction is restricted.	Transaction cannot perform because transaction includes restricted item.
EEVRW_RW_LOCKED	EVR/W device is locked.	Transaction cannot perform because EVR/W device is locked.
EEVRW_RW_OUT	Permanent error on a device.	Transaction cannot perform because of a permanent error on a device.
EEVRW_RW_OUT_TEMPORARY_OUT	Temporary recoverable error on a device.	Transaction cannot perform because of a temporary recoverable error on a device.
EEVRW_RW_OUT_TEMPORARY_OUT_NEED_TO_RESET	Reset request from EVR/W.	EVR/W needs to be reset.
EEVRW_TRANSACTION_INCOMPLETE	Transaction incomplete.	The problem occurred during transaction and transaction was aborted in the unknown state.
EEVRW_UNREACHABLE_HOST	Payment center cannot be reached.	Transaction cannot perform because the payment center cannot be reached.
EEVRW_UPOS114_COMPATIBLE	For compatibility with the error code defined by UPOS older version.	The error code defined by the ResultCodeExtended property of UPOS1.14 is set to a Vendor oriented error code.

A vendor oriented error code is a code from which a definition differs by the device or a service and which shows a detailed error condition.

The contents of a vendor oriented error code are dependent on vendors.

15.4 Properties (UML attributes)

15.4.1 AccountNumber Property *Updated in Release 1.14*

Syntax **AccountNumber:** ***string*** **{ read-only, access after open }**

Remarks Information for the service provider such as card number, member number, etc.; specifies the user (owner) of the card from data set information on the card.

Note as of Release 1.14: The **AccountNumber** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **AccountNumber** *property wherever possible.*

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors**" on page 1- 16.

15.4.2 AdditionalSecurityInformation Property

Syntax **AdditionalSecurityInformation:** ***string*** **{ read-write, access after open }**[1]

Remarks An application can send data to the EVR/W device by setting this property before issuing an authorization method. Also, data obtained from the EVR/W device and not stored in any other property as the result of an authorization operation can be provided to an application by storing it in this property. Since the data stored here is device specific, this should not be used for any development that requires portability.

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors**" on page 1- 16.

15.4.3 Amount Property *Updated in Release 1.14*

Syntax **Amount:** ***currency*** **{ read-write, access after open }**

Remarks Holds the payment amount on the electronic money service.

Note as of Release 1.14: The **Amount** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **Amount** *property wherever possible.*

This property is initialized to zero by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors**" on page 1- 16.

---

1. In the **OPOS** environment, the format of this data depends upon the value of the **BinaryConversion** property. See **BinaryConversion** property in Annex A.

15.4.4 ApprovalCode Property

Syntax **ApprovalCode:** ***string*** **{ read-write, access after open }**

Remarks Holds the payment approval code.

The content of the approval code depends on implementation the device. When a unique number is issued to the processing done with the device, the information is set.

This property is set to specify the cancellation of the payment when the device supports cancellation of the payment and the **cancelValue** method is executed.

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors**" on page 1- 16.

15.4.5 AsyncMode Property

Syntax **AsyncMode:** ***boolean*** **{ read-write, access after open }**

Remarks If true, the **writeValue**, **addValue**, **subtractValue**, **cancelValue**, **accessLog**, and **transactionAccess** methods will be performed asynchronously.

If false, these methods will be performed synchronously.

This property is initialized to false by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors**" on page 1- 16.

15.4.6 Balance Property *Updated in Release 1.14*

Syntax **Balance:** ***currency*** **{ read-only, access after open }**

Remarks Holds the balance on the electronic money service.

Note as of Release 1.14: The **Balance** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **Balance** *property wherever possible.*

This property is initialized to zero by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors**" on page 1- 16.

15.4.7 BalanceOfPoint Property *Updated in Release 1.14*

Syntax **BalanceOfPoint:** ***currency*** **{ read-only, access after open }**

Remarks Holds the point balance on the point service.

Note as of Release 1.14: The **BalanceOfPoint** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **BalanceOfPoint** *property wherever possible.*

This property is initialized to zero by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors**" on page 1- 16.

15.4.8 CapActivateService Property

Syntax **CapActivateService:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the activation processing is supported; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.9 CapAdditionalSecurityInformation Property *Added in Release 1.15*

Syntax **CapAdditionalSecurityInformation:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **AdditionalSecurityInformation** property may be utilized; otherwise it is false.

This property is initialized by **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **AdditionalSecurityInformation property**.

15.4.10 CapAddValue Property

Syntax **CapAddValue:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the addition of electronic value is supported; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.11 CapAuthorizeCompletion Property *Added in Release 1.15*

Syntax **CapAuthorizeCompletion:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **authorizeCompletion** method has been implemented; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **authorizeCompletion** method.

15.4.12 CapAuthorizePreSales Property *Added in Release 1.15*

Syntax **CapAuthorizePreSales:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **authorizePreSales** method has been implemented; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **authorizePreSales** method.

15.4.13 CapAuthorizeRefund Property *Added in Release 1.15*

Syntax **CapAuthorizeRefund:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **authorizeRefund** method has been implemented; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **authorizeRefund** method.

15.4.14 CapAuthorizeVoid Property *Added in Release 1.15*

Syntax **CapAuthorizeVoid:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **authorizeVoid** method has been implemented; otherwise it is false**.**

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **authorizeVoid** Method.

15.4.15 CapAuthorizeVoidPreSales Property *Added in Release 1.15*

Syntax **CapAuthorizeVoidPreSales:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **authorizeVoidPreSales** method has been implemented; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **authorizeVoidPreSales** Method.

15.4.16 CapCancelValue Property

Syntax **CapCancelValue:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the cancellation of the operation to the electronic value is supported; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.17 CapCrdSensor Property

Syntax **CapCardSensor:** ***int32*** **{ read-only, access after open }**

Remarks Contains a bit mask indicating the types of card detection supported. When the sensor exists, the detection is set to the **DetectionStatus** property and a **StatusUpdateEvent** is delivered.

This property is set to the logical OR of one or more of the following values:

Value	Meaning
EVRW_CCS_ENTRY	There is an insertion slot sensor.
EVRW_CCS_DETECT	There is a card detection sensor.
EVRW_CCS_CAPTURE	There is a stock space sensor.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **DetectionStatus** Property, **StatusUpdateEvent**.

15.4.18 CapCashDeposit Property *Added in Release 1.15*

Syntax **CapCashDeposit:** ***boolean*** **{ read-only, access after open }**

Remarks Show the device has charged method by **cashDeposit** method or not. If true, the **cashDeposit** method is implemented, otherwise false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **cashDeposit** method.

15.4.19 CapCenterResultCode Property *Added in Release 1.15*

Syntax **CapCenterResultCode:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **CenterResultCode** property has been implemented; otherwise it is false.

This property is initialized by the **open** method**.**

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **CenterResultCode** property.

15.4.20 CapCheckCard Property *Added in Release 1.15*

Syntax **CapCheckCard:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **checkCard** method has been implemented; otherwise it is false.

This property is initialized by the open method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **checkCard** method.

15.4.21 CapDailyLog Property *Added in Release 1.14*

Syntax **CapDailyLog:** ***int32*** **{ read-only, access after open }**

Remarks Shows the daily log ability of the device.

Value	Meaning
EVRW_DL_NONE	The EVRW device does not have the daily log functions.
EVRW_DL_REPORTING	The EVRW device only has an intermediate total function which reads the daily log but does not erase the log.
EVRW_DL_SETTLEMENT	The EVRW device only has the "final total" and "erase daily log" functions.
EVRW_DL_REPORTING_SETTLEMENT	The EVRW device has both the intermediate total function and the final total and erase daily log function.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **DailyLog property, accessDailyLog method.**

15.4.22 CapDetectionControl Property

Syntax **CapDetectionControl:** ***int32*** **{ read-only, access after open }**

Remarks It is shown whether the detection processing of the card, the ejection processing of the card, the storing processing of the card and these processing can be controlled from the application or the EVR/W.

This property is set to the logical OR of one or more of the following values:

Value	Meaning
EVRW_CDC_RWCONTROL	Control is possible by the EVR/W device.
EVRW_CDC_APPLICATIONCONTROL	Control is possible by the application.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **DetectionControl** Property, **DetectionStatus** Property.

15.4.23 CapElectronicMoney Property

Syntax **CapElectronicMoney:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the electronic money service is supported; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.24 CapEnumerateCardServices Property

Syntax **CapEnumerateCardServices:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the enumeration of service in the card is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.25 CapIndirectTransactionLog Property

Syntax **CapIndirectTransactionLog:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the transaction log is accessed as a file; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.26 CapInstallments Property *Added in Release 1.15*

Syntax **CapInstallments:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the item "Installments" which is stored in the **DailyLog** property as the result of **accessDailyLog** will be provided; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **DailyLog property**.

15.4.27 CapLockTerminal Property

Syntax **CapLockTerminal:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the security lock setting is supported; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **lockTerminal** Method.

15.4.28 CapLogStatus Property

Syntax **CapLogStatus:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the reporting of the status of the transaction log is supported; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **StatusUpdateEvent**.

15.4.29 CapMediumID Property

Syntax **CapMediumID:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the specification of the medium identifier is supported; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.30 CapMembershipCertificate Property *Added in Release 1.14.1*

Syntax **CapMembershipCertificate:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the membership certificate service is supported otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.31 CapPaymentDetail Property *Added in Release 1.15*

Syntax **CapPaymentDetail:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **PaymentDetail** property has been implemented; otherwise it is false.

This property is initialized by **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **PaymentDetail property**.

15.4.32 CapPINDevice Property *Added in Release 1.15*

Syntax **CapPINDevice:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the EVR/W is equipped with a PIN device.
If false, the EVR/W is not equipped with a PIN device. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.33 CapPoint Property

Syntax **CapPoint:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the point service is supported otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.34 CapSubtractValue Property

Syntax **CapSubtractValue:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the subtraction of electronic value is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.35 CapTaxOthers Property *Added in Release 1.15*

Syntax **CapTaxOthers:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the item "TaxOthers" which is stored in the **DailyLog** property as the result of **access DailyLog** will be provided; otherwise it is false.

Note that this property is not related to the "TaxOthers" argument used with the authorization methods.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **DailyLog property**.

15.4.36 CapTrainingMode Property *Added in Release 1.14*

Syntax **CapTrainingMode:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the EVR/W supports a training mode.
If false, the EVR/W does not support a training mode. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.37 CapTransaction Property

Syntax **CapTransaction:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the transaction mode is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.38 CapTransactionLog Property

Syntax **CapTransactionLog:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the transaction log is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.39 CapTransactionNumber Property *Added in Release 1.15*

Syntax **CapTransactionNumber:** ***boolean*** **{ read-only, access after open }**

Remarks If true, the **TransactionNumber** property has been implemented; otherwise it is false.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **TransactionNumber property**.

15.4.40 CapUnlockTerminal Property

Syntax **CapUnlockTerminal: *boolean* { read-only, access after open }**

Remarks If true, releasing of the security lock is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **unlockTerminal** Method.

15.4.41 CapUpdateKey Property

Syntax **CapUpdateKey: *boolean* { read-only, access after open }**

Remarks If true, the update of key information is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.42 CapVoucher Property

Syntax **CapVoucher: *boolean* { read-only, access after open }**

Remarks If true, the voucher/ticket service is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.43 CapWriteValue Property

Syntax **CapWriteValue: *boolean* { read-only, access after open }**

Remarks If true, the writing of electronic value is supported; otherwise it is false. This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.44 CardCompanyID Property *Added in Release 1.15*

Syntax **CardCompanyID:** ***string*** **{ read-only, access after open }**

Remarks This property is updated when an authorization operation successfully completes. It shows credit card company ID.

The length of the ID string varies depending upon the EVRW device.

This property is initialized to an empty string by the **open** method

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.45 CardServiceList Property *Updated in Release 1.14.1*

Syntax **CardServiceList:** ***string*** **{ read-only, access after open }**

Remarks Holds the comma-separated (CSV) list of services supported by the card. This list is populated by the **enumerateCardServices** method.

For example, when the character string that identifies A electronic money service is "MoneyA" and the character string that identifies B electronic point service is "PointB," the **CardServiceList** property becomes "MoneyA,PointB."

Note as of Release 1.14.1: In case service has variation

When a service has some variations, a string value of this property can be specified with the following rules.

"*service* [:*variation* [:*additional*]]"

Service is required. *Variation* with separator ":" and *Additional* with separator ":" are optional. Separator characters such as ",", and ":" cannot be used for a *Service*, *Variation*, and *Additional* identifier.

Example:
Service "XYZCustomerPoint" offers two variations, "ABCStore" and "DEFShop", as a variation. In this case, it will be set to a **ReaderWriterServiceList** property as "XYZCustomerPoint:ABCStore, XYZCustomerPoint:DEFShop."

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **enumerateCardServices** Method.

15.4.46 CenterResultCode Property *Added in Release 1.15*

Syntax **CenterResultCode:** ***string*** **{ read-only, access after open }**

Remarks Contains the code from the approval agency. Check the approval agency for the actual codes to be stored.

This property is initialized to an empty string by the **open** method and is updated when an authorization operation successfully completes.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.47 CurrentService Property *Updated in Release 1.14.1*

Syntax **CurrentService:** ***string*** **{ read-write, access after open }**

Remarks Holds the character string that identifies the currently selected service.
This value is guaranteed to be one of the set of services specified by the **ReaderWriterServiceList** property.

The character string being enumerated by the **ReaderWriterServiceList** property can be set.
If an empty string ("") is set, it enters the state that no service has been selected.
In this state, depending on the device, an application can operate directly to the device.
When a valid string is set, the service is selected and started.

If the service supports the sub-service, the execution of the method and the setting of property are done to the sub-service of the service that property shows. And only the event fires from the sub-service which is selected by this property.

Note as of Release 1.14.1: In case service has variation

When a service has some variations, a string value of this property can be specified with the following rules.

"*service* [:*variation* [:*additional*]]"

Service is required. *Variation* with separator ":" and *Additional* with separator ":" are optional. Separator characters such as ",", and ":" cannot be used for a *Service*, *Variation*, and *Additional* identifier.

Example:
Service "XYZCustomerPoint" offers two variations, "ABCStore" and "DEFShop", as a variation.
In this case, it will be set to a **ReaderWriterServiceList** property as
"XYZCustomerPoint:ABCStore, XYZCustomerPoint:DEFShop".

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **ReaderWriterServiceList** Property.

15.4.48 DailyLog Property *Added in Release 1.15*

Syntax **DailyLog:** ***string*** **{ read-only, access after open }**

Remarks Stores the result of the **accessDailyLog** method. The data is delimited by CR(13 decimal)+LF(10 decimal) for each transaction and is stored in ASCII code. The detailed data of each transaction is comma separated [i.e., delimited by "," (44)].
The details of one transaction are shown as follows:

No	Item	Property	Corresponding **Cap** Property
1	Card company ID	**CardCompanyID**	None
2	Transaction type	**TransactionType**	None
3	Transaction date Note 1)	None	None
4	Transaction number Note 3)	**TransactionNumber**	**CapTransactionNumber**
5	Payment condition	**PaymentCondition**	None
6	Slip number	**SlipNumber**	None
7	Approval code	**ApprovalCode**	None
8	Purchase date Note 5)	None	None
9	Account number	**AccountNumber**	None
10	Amount Note 4)	The argument Amount of the authorization method or the amount actually approved.	None
11	Tax/others Note 3)	The argument TaxOthers of the authorization method.	**CapTaxOthers**
12	Installments Note 3)	None	**CapInstallments**
13	Additional data Note 2)	**AdditionalSecurityInformation**	**CapAdditionalSecurity Information**

Notes from the previous table:

1) Format

Item	Format
Transaction date	YYYYMMDDHHMMSS
Purchase date	MMDD

Some EVRW devices may not support seconds by the internal clock. In that case, the second field of the transaction date is filled with "00"

2) Additional data

The area where the EVRW device stores the vendor specific data. This enables an application to receive data other than that defined in this specification. The data stored here is vendor specific and should not be used for development which places an importance on portability.

3) If the corresponding Cap property is false

Cap property is set to false if the EVRW device provides no corresponding data. In such instances, the item cannot be displayed so the next comma delimiter immediately follows. For example, if "Amount" is 1234 yen and "Tax/others" is missing and "Installments" is 2, the description will be "1234,,2". This makes the description independent of Cap property and makes the position of each data item consistent.

4) Amount

Amount always includes "Tax/others" even if item 11 is present.

5) Purchase date

The date manually entered for the purchase transaction after approval.

Example: An example of daily log content is shown below.

Item	Description	Meaning
Card company ID	102	JCB
Transaction type	EVRW_TRANSACTION_SALES	Purchase
Transaction date	19980116134530	1/16/199813:45:30
Transaction number	123456	123456
Payment condition	EVRW_PAYMENT_INSTALLMENT_1	Installment 1
Slip number	12345	12345
Approval code	0123456	0123456
Purchase date	None	None
Account number	1234123412341234	1234-1234-1234-1234
Amount	12345	12345JPY
Tax/others	None	None
Number of payments	2	2
Additional data	12345678	Specific information

The actual d

The acutal data stored in DailyLog will be as follows:

```
102,10,19980116134530,123456,61,12345,0123456,,12341234123
41234,12345,,2,12345678[CR][LF]
```

Electronic Money Device: Setting DealingLog which is a result of the Electronic Money Device which does not have the communication module for closing processing done closing processing. It may be the device which is enciphered DealingLog to everything except for Center.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **CapDailyLog** Property, **accessDailyLog** Method.

15.4.49 DetectionControl Property

Syntax **DetectionControl:** ***boolean*** **{ read-write, access after open }**

Remarks If true, the detection processing of the card by the **beginDetection**/**endDetection** methods and the card ejection processing by the **beginRemoval**/**endRemoval** methods are controlled by the application.

This property can only be set true by the application when **CapDetectionControl** is set to EVRW_CDC_APPLICATIONCONTROL.

If false, neither detection nor the ejection processing of the card are controlled from the application. Invocation of the **beginDetection**/**endDetection** methods and the **beginRemoval**/**endRemoval** methods from the application is invalid. When EVRW_CDC_RWCONTROL is specified for the **CapDetectionControl** property, it is possible to set it.

This property is initialized to false by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **CapDetectionControl** Property, **beginDetection** Method, **beginRemoval** Method, **endDetection** Method, **endRemoval** Method.

15.4.50 DetectionStatus Property

Syntax **DetectionStatus:** ***int32*** **{ read-only, access after open }**

Remarks Holds the state of card detection.

Value	Meaning
EVRW_DS_NOCARD	No card. The card detection sensor does not detect a card.
EVRW_DS_DETECTED	There is a card in the device. The card detection sensor detects the card.
EVRW_DS_ENTERED	Card remaining at the insertion slot. The insertion slot sensor detects the card.
EVRW_DS_CAPTURED	The card is in the stock space. The stock space sensor detects the card.

This property is initialized to EVRW_DS_NOCARD by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.51 ExpirationDate Property *Updated in Release 1.14*

Syntax **ExpirationDate:** ***string*** **{ read-only, access after open }**

Remarks Holds the expiration date in the format "YYYYMMDD".

Note as of Release 1.14: The **ExpirationDate** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **ExpirationDate** *property wherever possible.*

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.52 LastUsedDate Property *Updated in Release 1.14*

Syntax **LastUsedDate:** ***string*** **{ read-only, access after open }**

Remarks Holds the last used date in the format "YYYYMMDDHHMMSS".

Note as of Release 1.14: The **LastUsedDate** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **LastUsedDate** *property wherever possible.*

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.53 LogStatus Property

Syntax **LogStatus:** ***int32*** **{ read-only, access after open }**

Remarks Holds the state of transaction log.

Value	Meaning
EVRW_LS_OK	Transaction Log has enough capacity.
EVRW_LS_NEARFULL	Transaction Log is nearly full.
EVRW_LS_FULL	Transaction Log is full.

If transaction log becomes full, depending on the device, the settlement processing may not be able to operate.

After this property is initialized, it is kept current as long as the device is enabled.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.54 MediumID Property *Updated in Release 1.14*

Syntax **MediumID:** ***string*** **{ read-write, access after open }**

Remarks Holds the medium identifier of the card.

The medium identifier is information (manufacturer's serial number, etc.) to specify the card uniquely, and its content depends on implementation for the card.

The following methods are processed to the card with the medium identifier specified by this property:

- **addValue**
- **beginDetection**
- **cancelValue**
- **readValue**
- **subtractValue**
- **writeValue**

The application can specify the card to be operated on by setting the medium identifier to this property before the method call is issued. Setting an empty string ("") for this property means the operation can be performed with any card.

The medium identifier of the card is set when the method that have relation to the card succeeds.

Note as of Release 1.14: The **MediumID** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **MediumID** *property wherever possible.*

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **addValue** Method, **beginDetection** Method, **cancelValue** Method, **readValue** Method, **subtractValue** Method, **writeValue** Method.

15.4.55 PaymentCondition Property *Updated in Release 1.15*

Syntax **PaymentCondition:** ***int32*** **{ read-only, access after open }**

Remarks Holds the payment condition of the most recent successful authorization operation.
This property will be set to one of the following values. See **PaymentDetail** for the detailed payment string that correlates to the following **PaymentCondition** values.

Value	Meaning
EVRW_PAYMENT_LUMP	Lump-sum
EVRW_PAYMENT_BONUS_1	Bonus 1
EVRW_PAYMENT_BONUS_2	Bonus 2
EVRW_PAYMENT_BONUS_3	Bonus 3
EVRW_PAYMENT_BONUS_4	Bonus 4
EVRW_PAYMENT_BONUS_5	Bonus 5
EVRW_PAYMENT_INSTALLMENT_1	Installment 1
EVRW_PAYMENT_INSTALLMENT_2	Installment 2
EVRW_PAYMENT_INSTALLMENT_3	Installment 3
EVRW_PAYMENT_BONUS_COMBINATION_1	Bonus combination payments 1
EVRW_PAYMENT_BONUS_COMBINATION_2	Bonus combination payments 2
EVRW_PAYMENT_BONUS_COMBINATION_3	Bonus combination payments 3
EVRW_PAYMENT_BONUS_COMBINATION_4	Bonus combination payments 4
EVRW_PAYMENT_ REVOLVING	Revolving
EVRW_PAYMENT_DEBIT	Debit card
EVRW_PAYMENT_ELECTRONIC_MONEY	Electronic Money

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **PaymentDetail property**

15.4.56 PaymentDetail Property *Added in Release 1.15*

Syntax **PaymentDetail:** ***string*** **{ read-only, access after open }**

Remarks Contains payment condition details as the result of an authorization operation. Payment details vary depending on the value of **PaymentCondition**. The data will be stored as comma separated ASCII code. An empty string means that no data is stored and represents a string with zero length data.

PaymentCondition	PaymentDetail
EVRW_PAYMENT_LUMP	An empty string
EVRW_PAYMENT_BONUS_1	An empty string
EVRW_PAYMENT_BONUS_2	Number of bonus payments
EVRW_PAYMENT_BONUS_3	1st bonus month
EVRW_PAYMENT_BONUS_4*	Number of bonus payments, 1st bonus month, 2nd bonus month, 3rd bonus month, 4th bonus month, 5th bonus month, 6th bonus month
EVRW_PAYMENT_BONUS_5*	Number of bonus payments, 1st bonus month, 1st bonus amount, 2nd bonus month, 2nd bonus amount, 3rd bonus month, 3rd bonus amount, 4th bonus month, 4th bonus amount, 5th bonus month, 5th bonus amount, 6th bonus month, 6th bonus amount
EVRW_PAYMENT_INSTALLMENT_1	1st billing month, Number of payments
EVRW_PAYMENT_INSTALLMENT_2*	1st billing month, Number of payments, 1st amount, 2nd amount, 3rd amount, 4th amount, 5th amount, 6th amount
EVRW_PAYMENT_INSTALLMENT_3	1st billing month, Number of payments, 1st amount
EVRW_PAYMENT_BONUS_COMBINATION_1	1st billing month, Number of payments
EVRWT_PAYMENT_BONUS_COMBINATION_2	1st billing month, Number of payments, bonus amount
EVRW_PAYMENT_BONUS_COMBINATION_3*	1st billing month, Number of payments, number of bonus payments, 1st bonus month, 2nd bonus month, 3rd bonus month, 4th bonus month, 5th bonus month, 6th bonus month

EVRW_PAYMENT_BONUS_COMBINATION_4*	1st billing month, Number of payments, number of bonus payments, 1st bonus month, 1st bonus amount, 2nd bonus month, 2nd bonus amount, 3rd bonus month, 3rd bonus amount, 4th bonus month, 4th bonus amount, 5th bonus month, 5th bonus amount, 6th bonus month, 6th bonus amount
EVRW_PAYMENT_REVOLVING	An empty string
EVRW_PAYMENT_DEBIT	An empty string
EVRW_PAYMENT_ELECTRONIC_MONEY	An empty string

*Maximum 6 installments

The payment types and names vary depending on the EVRW device. The following are the payment types and terms available for EVRW devices. Note that there are some differences between UnifiedPOS terms and those used by the EVRW devices. The goal of this table is to synchronize these terms.

General Payment Category	Entry item	PaymentCondition Value	CAT Name	CAT (Old CAT)	G-CAT	JET-S	SG-CAT	Master-T
			Credit Card	Not specified	Not specified	JCB	VISA	MASTER
			UnifiedPOS Term	Card Company Terms				
Lump-sum	(None)	10	Lump-sum	Lump-sum	Lump-sum	Lump-sum	Lump-sum	Lump-sum
Bonus	(None)	21	Bonus 1	Bonus 1	Bonus 1	Bonus 1	Bonus 1	Bonus 1
	Number of bonus payments	22	Bonus 2	Bonus 2	Bonus 2	Bonus 2	Bonus 2	Bonus 2
	Bonus month(s)	23	Bonus 3	Bonus 3	Does not exist.	Does not exist.	Bonus 3	Bonus 3

	Number of bonus payments Bonus month (1) Bonus month (2) Bonus month (3) Bonus month (4) Bonus month (5) Bonus month (6)	24	Bonus 4	Bonus 4	Bonus 3	Bonus 3	Bonus 4 (Up to two entries for bonus month)	Bonus 4
	Number of bonus payments Bonus month (1) Bonus amount (1) Bonus month (2) Bonus amount (2) Bonus month (3) Bonus amount (3) Bonus month (4) Bonus amount (4) Bonus month (5) Bonus amount (5) Bonus month (6) Bonus amount (6)	25	Bonus 5	Bonus 5	Does not exist.	Does not exist.	Does not exist.	Bonus 5
Installment	Payment start month Number of payments	61	Installment 1	Installment 1	Installment 1	Installment 1	Installment 1	Installment 1

	Payment start month Number of payments Installment amount(1) Installment amount(2) Installment amount(3) Installment amount(4) Installment amount(5) Installment amount(6)	62	Installment 2	Installment 2	Does not exist.	Does not exist.	Does not exist.	Does not exist.
	Payment start month Number of payments Initial amount	63	Installment 3	Installment 3	Installment 2	Installment 2	Does not exist.	Installment 2
Combination	Payment start month Number of payments	31	Bonus Combination 1	Bonus Combination 1	Bonus Combination 1	Bonus Combination 1	Bonus Combination 1	Bonus Combination 1

	Payment start month Number of payments Bonus amount	32	Bonus Combination 2	Bonus Combination 2	Does not exist.	Does not exist.	Bonus Combination 2	Bonus Combination 2
	Payment start month Number of payments Number of bonus payments Bonus month (1) Bonus month (2) Bonus month (3) Bonus month (4) Bonus month (5) Bonus month (6)	33	Bonus Combination 3	Bonus Combination 3	Does not exist.	Does not exist.	Bonus Combination 3 (Up to two entries for bonus month)	Bonus Combination 3

	Payment start month Number of payments Number of bonus payments Bonus month (1) Bonus amount(1) Bonus month (2) Bonus amount(2) Bonus month (3) Bonus amount(3) Bonus month (4) Bonus amount(4) Bonus month (5) Bonus amount(5) Bonus month (6) Bonus amount(6)	34	Bonus Combination 4	Bonus Combination 4	Bonus Combination 2	Bonus Combination 2	Bonus Combination 4 (Up to two entries for bonus month and amount)	Bonus Combination 4

Revolving	(None)	80	Revolving	Revolving	Revolving	Revolving	Revolving	Revolving
Debit	(None)	110	Debit	(Support depends on the actual device)	(Support depends on the actual device)	(Support depends on the actual device)	(Support depends on the actual device)	(Support depends on the actual device)

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **CapPaymentDetail property**.

15.4.57 PaymentMedia Property *Added in Release 1.15*

Syntax **PaymentMedia:** ***int32*** **{ read-write, access after open }**

Remarks Holds the payment media type that the approval method should approve.

The application sets this property to one of the following values before issuing an approval method call. "None specified" means that payment media will be determined by the EVRW device, not by the POS application.

Value	Meaning
EVRW_MEDIA_UNSPECIFIED	None specified.
EVRW_MEDIA_CREDIT	Credit card.
EVRW_MEDIA_DEBIT	Debit card.
EVRW_MEDIA_ELECTRONIC_MONEY	Electronic Money.

This property is initialized to EVRW_MEDIA_UNSPECIFIED by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.58 PINEntry Property *Added in Release 1.14*

Syntax **PINEntry:** ***int32*** **{ read-write, access after open }**

Remarks The PIN entry functionality that is supported by the EVR/W.

Value	Meaning
EVRW_PIN_ENTRY_NONE	PIN input is not supported.
EVRW_PIN_ENTRY_EXTERNAL	The EVR/W is not equipped with the PIN input device. When PIN input is required, it is necessary to use an external PIN pad device.
EVRW_PIN_ENTRY_INTERNAL	The EVR/W is equipped with an internal PIN input device for PIN number entry.
EVRW_PIN_ENTRY_UNKNOWN	The PIN entry may be supported by the EVR/W device but the **CurrentService** property is set to empty string (““) and the it is not clear where the PIN entry is to occur.

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **“Errors”** on page 1- 16.

15.4.59 Point Property *Updated in Release 1.14*

Syntax **Point:** ***currency*** **{ read-write, access after open }**

Remarks Holds the settlement point on the point service.

Note as of Release 1.14: The **Point** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **Point** *property wherever possible.*

This property is initialized to zero by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **“Errors”** on page 1- 16.

15.4.60 ReaderWriterServiceList Property *Updated in Release 1.14.1*

Syntax **ReaderWriterServiceList:** ***string*** **{ read-only, access after open }**

Remarks Holds the comma-separated list of services that are supported by the EVR/W device.

For example, when the character string that identifies 'A' electronic money service is "MoneyA" and the character string that identifies 'B' electronic point service is "PointB," the **ReaderWriterServiceList** property becomes "MoneyA,PointB."

If the service supports the sub-service, the **open** method succeeds, the service that all the sub-services provides is enumerated.

If the EVR/W service does not support the sub-service and an EVR/W service supports many services, those services are enumerated by this property.

This property is initialized by the **open** method. The initialization value depends on what services are supported; e.g., if the EVR/W device supports "MoneyA" and "PointB" services, this property is initialized to "MoneyA, PointB."

Note as of Release 1.14.1:

When a service has some variations, a string value of this property can be specified using the following rules.

"*service* [:*variation* [:*additional*]]"

Service is required. *Variation* with separator ":" and *Additional* with separator ":" are optional. Separator characters such as ",", and ":" cannot be used for a *Service*, *Variation*, and *Additional* identifier.

Expamle:
Service "XYZCustomerPoint" offers two variations, "ABCStore" and "DEFShop" as a variation. In this case, it will be set to a **ReaderWriterServiceList** property as "XYZCustomerPoint:ABCStore, XYZCustomerPoint:DEFShop."

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.61 SequenceNumber Property

Syntax **SequenceNumber:** ***int32*** **{ read-only, access after open }**

Remarks Holds a "sequence number" as the result of each method call. This number needs to be checked by an application to see if it matches with the argument *sequenceNumber* of the originating method.

This property is initialized to zero by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.62 ServiceType Property *Updated in Release 1.15*

Syntax **ServiceType:** ***int32*** **{ read-only, access after open }**

Remarks This property is initialized by the **open** method and updated when the **CurrentService** property is updated.

Value	Meaning
EVRW_ST_ELECTRONIC_MONEY	Electronic money service
EVRW_ST_POINT	Point service
EVRW_ST_VOUCHER	Voucher/Ticket service
EVRW_ST_MEMBERSHIP	Membership certificate service
EVRW_ST_UNSPECIFIED	Nothing is set to **CurrentService**
EVRW_ST_CAT	Credit Authorization Terminal service

This property is initialized by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **CurrentService** Property.

15.4.63 SettledAmount Property *Updated in Release 1.14*

Syntax **SettledAmount:** ***currency*** **{ read-only, access after open }**

Remarks Sets the real amount of the settlement on the electronic money service.

Note as of Release 1.14: The **SettledAmount** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **SettledAmount** *property wherever possible.*

This property is initialized to zero by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.64 SettledPoint Property

Syntax **SettledPoint:** ***currency*** **{ read-only, access after open }**

Remarks Sets the settlement point on the point service.

This property is initialized to zero by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.65 SlipNumber Property *Added in Release 1.15*

Syntax **SlipNumber:** ***string*** **{ read-only, access after open }**

Remarks Stores a "slip number" as the result of each authorization operation.

This property is initialized to an empty string by the **open** method and is updated when an authorization operation successfully completes.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.66 TrainingModeState Property *Added in Release 1.14*

Syntax **TrainingModeState:** ***int32*** **{ read-write, access after open }**

Remarks The current state of the EVR/W device to indicate if the device is in training mode or not.

Value	Meaning
EVRW_TM_FALSE	The training mode is not selected, therefore normal operation is the current state.
EVRW_TM_TRUE	The training mode is selected.
EVRW_TM_UNKNOWN	The training mode may be supported by the EVR/W device but the **CurrentService** property is set to empty string ("") and the it is not clear what is the current state of the training mode.

This property is initialized to one of the these values by the **open** method.

Errors If **TrainingModeState** is set to EVRW_TM_TRUE but the device does not support training mode, a UposException with E_ILLEGALmay be thrown.
A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **CapTrainingMode** Property.

15.4.67 TransactionLog Property

Syntax **TransactionLog:** ***string*** **{ read-only, access after open }**

Remarks Stores the result of the **accessLog** method.

If the **CapIndirectTransactionLog** property is true, the **TransactionLog** property shows URL that shows the position such as files where the transaction log is stored. The content of the transaction log depends on the device and service. This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

See Also **CapIndirectTransactionLog** Property, **TransactionLog** Property, **accessLog** Method.

15.4.68 TransactionNumber Property *Added in Release 1.15*

Syntax **TransactionNumber:** ***string*** **{ read-only, access after open }**

Remarks Stores a "transaction number" as the result of each authorization operation. This property is initialized to an empty string by the **open** method and is updated when an authorization operation successfully completes.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.69 TransactionType Property *Added in Release 1.15*

Syntax **TransactionType:** ***int32*** **{ read-only, access after open }**

Remarks Stores a "transaction type" as the result of each authorization operation.

This property is initialized to zero by the **open** method and is updated when an authorization operation successfully completes.This property will be set to one of the following values.

Value	Meaning
EVRW_TRANSACTION_SALES	Sales
EVRW_TRANSACTION_VOID	Cancellation
EVRW_TRANSACTION_REFUND	Refund purchase
EVRW_TRANSACTION_COMPLETION	Purchase after approval
EVRW_TRANSACTION_PRESALES	Pre-authorization
EVRW_TRANSACTION_CHECKCARD	Card Check
EVRW_TRANSACTION_VOIDPRESALES	Cancel pre-authorization approval
EVRW_TRANSACTION_CASHDEPOSIT	Charge

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.70 VoucherID Property *Updated in Release 1.14*

Syntax **VoucherID:** ***string*** **{ read-write, access after open }**

Remarks Sets the ID of voucher/ticket on the voucher/ticket service.

It consists of pairs of the identifier and the number which validate the card holder.

For example, six tickets of identifier "001" are shown by the character string "001:6". The ":" is a separator between the identifier and the number of sheets.

Note as of Release 1.14: The **VoucherID** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **VoucherID** *property wherever possible.*

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.4.71 VoucherIDList Property *Updated in Release 1.14*

Syntax **VoucherIDList:** ***string*** **{ read-write, access after open }**

Remarks Sets the IDs of voucher/ticket are enumerated on the voucher/ticket service.

If six tickets of identifier "001", one ticket of identifier "002", two tickets of identifier "034" are maintained, this is expressed by the CSV character string in the format "001:6,002:1,034:2". The "," is a separator when two or more rights are maintained.

Note as of Release 1.14: The **VoucherIDList** *property may contain some of the same information found in the tag values used by the* **setParameterInformation** *and* **retrieveResultInformation** *methods. The tag values should be used instead of the* **VoucherIDList** *property wherever possible.*

This property is initialized to an empty string ("") by the **open** method.

Errors A UposException may be thrown when this property is accessed. For further information, see **"Errors"** on page 1- 16.

15.5 Methods (UML operations)

15.5.1 accessDailyLog Method *Added in Release 1.15*

Syntax **accessDailyLog (sequenceNumber:** ***int32*****, type:** ***int32*****, timeout:** ***int32*** **): void { raises-exception, use after open-claim-enable }**

Parameter	Description
sequenceNumber	The sequence number to get daily log.
type	Specify whether the daily log is intermediate total or final total and erase.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Gets daily log from EVRW.

Daily log will be retrieved and stored in **DailyLog** as specified by *sequenceNumber*.

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Application must specify one of the following values for *type* for daily log type (either intermediate total or adjustment). Legal values depend upon the **CapDailyLog** value.

Electronic Money Device: Gets the **DealingLog** from the Electronic Money Device to send to the Center. If the Electronic Money Device has communication capabilities, the **DealingLog** will be sent from the Electronic Money Device to the Center and nothing is stored in the **DailyLog**. Otherwise, the **DealingLog** is stored in the **DailyLog** Property.

Value	Meaning
EVRW_DL_REPORTING	Intermediate total.
EVRW_DL_SETTLEMENT	Final total and erase. **Electronic Money Device:** Closing **DealingLog** of the Electronic Money device.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported *type* or *timeout* parameter was specified, or **CapDailyLog** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in ErrorCode Extended.
E_BUSY	The EVRE device canot accept any commands now.

See Also **CapDailyLog** property, **DailyLog** property.

15.5.2 accessData Method *Added in Release 1.14.1*

Syntax **accessData (dataType:*int32*, inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
dataType	Type of the data which accesses

Value	Meaning
EVRW_AD_KEY	Key information.
EVRW_AD_NEGATIVE_LIST	Negative list.
EVRW_AD_OTHERS	Other information.
data	An array of one mutable integers whose specific values or usage vary by service.
obj	Additional data whose usage varies by service.

Remarks Data other than a transaction log is accessed from an EVR/W. It is supported when an EVR/W has accessible data besides a transaction log accessible by **AccessLog** method.

The contents of data are dependent on service.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

For consistency, a Service must always fire at least one **TransitionEvent** with an incomplete progress completion percentage (i.e., a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a **TransitionEvent** with a progress of 100. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have the activation.
E_BUSY	The device cannot accept any commands now.

See Also **accessLog** Method, **updateData** Method, **TransitionEvent**.

15.5.3 accessLog Method *Updated in Release 1.14.1*

Syntax **accessLog (sequenceNumber: *int32*, type: *int32,* timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	The sequence number to get transaction log.
type	Specifies whether the transaction log is intermediate total or the last total. (see values below)
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Gets transaction log from device. Gets transaction log on demand by s*equenceNumber,* and it is stored in the **TransactionLog** property.

When *timeout* is FOREVER(-1), a timeout never occurs and it waits indefinitely until it receives a response from the device. If EVR/W device needs the last total processing of a transaction, it performs this method. The last total processing might be cleared in the transaction log, this depends on the implementation. However, the intermediate total must not be cleared in the transaction log.

It depends on the implementation if the transaction log will be passed to the service center directly and not to the application. The application must specify one of the following values for *type* of transaction (either intermediate total or the last total).

Value	Meaning
EVRW_AL_REPORTING	Gets transaction log as an intermediate total.
EVRW_AL_SETTLEMENT	The transaction log for the device is fixed and erased. (Whether it is erased or not depends on the implementation.)

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Added in Release 1.14.1: For consistency, a Service must always fire at least one **TransitionEvent** with an incomplete progress completion percentage (i.e. a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a **TransitionEvent** with a progress of 100. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16. Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *type* or *timeout* parameter was specified. Or transaction log function is unsupported.
E_TIMEOUT	No response was received from device during the specified *timeout* (in milliseconds).
E_BUSY	The device cannot accept any commands while asynchronously processing.

See Also **TransactionLog** Property, **accessData** Method, **TransitionEvent**.

15.5.4 activateEVService Method *Added in Release 1.14.1*

Syntax **activateEVService (inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
data	An array of one mutable integer whose specific values or usage vary by service.
obj	Additional data whose usage varies by service.

Remarks Executes the device activation process.

If the device has the activation process function, it is supported.

The activation process is the initial process performed when newly installing a device or service, or when enabling the function disabled at the time of factory shipment.

The contents of processing and the contents of the parameter are dependent on service.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

For consistency, a Service must always fire at least one **TransitionEvent** with an incomplete progress completion percentage (i.e. a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a **TransitionEvent** with a progress of 100. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have the activation.
E_BUSY	The device cannot accept any commands now.

See Also **deactivateEVService** Method, **TransitionEvent**.

15.5.5 activateService Method

Syntax **activateService (inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Remarks Executes the device activation process.

If the device has the activation process function, it is supported.

The activation process is initialization or installation of device. The details of process contents and parameters depend on implementation.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have the activation.
E_BUSY	The device cannot accept any commands now.

See Also **CapActivateService** Property.

15.5.6 addValue Method

Syntax **addValue (sequenceNumber: *int32*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Electronic value is added to the card as specified by *sequenceNumber* on demand.

When *timeout* is FOREVER(-1), timeout never occurs and it waits indefinitely until it receives a response from the device.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *Timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **CapAddValue** Property, **cancelValue** Method, **readValue** Method, **subtractValue** Method, **writeValue** Method.

15.5.7 authorizeCompletion Method *Added in Release 1.15*

Syntax **authorizeCompletion (sequenceNumber: *int32*, amount: *currency*, taxOthers: *currency*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number for approval.
amount	Purchase amount for approval.
taxOthers	Tax and other amounts for approval.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Purchase after approval is intended.

Sales after approval for *amount* and *taxOthers* are intended as the approval specified by *sequenceNumber*.

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified, or **CapAuthorizeCompletion** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

See Also **CapAuthorizeCompletion** property.

15.5.8 authorizePreSales Method *Added in Release 1.15*

Syntax **authorizePreSales (sequenceNumber:** ***int32*****, amount:** ***currency*****, taxOthers:** ***currency*****, timeout:** ***int32*** **):**
void { raises-exception, use after open-claim-enable }

Value	Meaning
sequenceNumber	Sequence number for approval.
amount	Purchase amount for approval.
taxOthers	Tax and other amounts for approval.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Makes a pre-authorization.

Pre-authorization for *amount* and *taxOthers* is made as the approval specified by *sequenceNumber*.

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified, or **CapAuthorizePreSales** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

See Also **CapAuthorizePreSales** property.

15.5.9 authorizeRefund Method *Added in Release 1.15*

Syntax **authorizeRefund (sequenceNumber: *int32*, amount: *currency*, taxOthers: *currency*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number for approval.
amount	Purchase amount for approval.
taxOthers	Tax and other amounts for approval.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Refund purchase approval is intended.

Refund purchase approval for *amount* and *taxOthers* is intended as the approval specified by *sequenceNumber*.

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified, or **CapAuthorizeRefund** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

See Also **CapAuthorizeRefund** property.

15.5.10 authorizeSales Method *Added in Release 1.15*

Syntax **authorizeSales (sequenceNumber: *int32*, amount: *currency*, taxOthers: *currency*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number for approval.
amount	Purchase amount for approval.
taxOthers	Tax and other amounts for approval.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Normal purchase approval is intended.

Normal purchase approval for *amount* and *taxOthers* is intended as the approval specified by *sequenceNumber*.

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

15.5.11 authorizeVoid Method *Added in Release 1.15*

Syntax **authorizeVoid (sequenceNumber: *int32*, amount: *currency*, taxOthers: *currency*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number for approval.
amount	Purchase amount for approval.
taxOthers	Tax and other amounts for approval.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Purchase cancellation approval is intended.

Cancellation approval for *amount* and *taxOthers* is intended as the approval specified by *sequenceNumber*.

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified, or **CapAuthorizeVoid** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

See Also **CapAuthorizeVoid** property.

15.5.12 authorizeVoidPreSales Method *Added in Release 1.15*

Syntax **authorizeVoidPreSales (sequenceNumber: *int32*, amount: *currency*, taxOthers: *currency*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number for approval.
amount	Purchase amount for approval.
taxOthers	Tax and other amounts for approval.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Pre-authorization cancellation approval is intended.

Pre-authorization cancellation approval for *amount* and *taxOthers* is intended as the approval specified by s*equenceNumber*.

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Normal cancellation could be used for EVRW control and EVRW devices which have not implemented the pre-authorization approval cancellation. Refer to the documentation supplied with EVRW device and / or EVRW control.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified, or **CapAuthorizeVoidPreSales** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

See Also **CapAuthorizeVoidPreSales property**.

15.5.13 beginDetection Method

Syntax **beginDetection (type: *int32,* timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Executes the card detection process.

If the *timeout* parameter value is zero, the method initiates the detection mode immediately. If a value is set (in milliseconds), the card detection process will wait for this time period if necessary. If a value of FOREVER(-1) is specified, the method initiates the card detection process and then waits as long as necessary until either the card is detected or an error occurs.

The *type* parameter specifies the type of the detected card. The value that can be specified is as follows:

Value	Meaning
EVRW_BD_ANY	The content of the detected card can be anything.
EVRW_BD_SPECIFIC	When this method is called, only the card that corresponds to the identifier in the **MediumID** property can be detected.

Remarks Starts the card detection process in the device slot.

Supports the both contactless and contact IC card devices.

When called, the device starts a card detection process, and initiates the card detection in the device. This method is called together with the **endDetection** method that ends the card detection process.

If the device cannot be set to the detection process, an error exception will be fired such as E_TIMEOUT. However, the device stays in the detection mode until the **endDetection** method is called.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	Cannot execute while asynchronous processing.
E_ILLEGAL	An invalid *timeout* parameter was specified.
E_TIMEOUT	The specified timeout has elapsed without the card being properly detected.

See Also **MediumID** Property, **endDetection** Method.

15.5.14 beginRemoval Method

Syntax **beginRemoval (timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Executes the removal process

If the *timeout* parameter value is zero, the method initiated the detection mode immediately. If its value is set (milliseconds), the card detection process will be wait until time is due. If its value is FOREVER(-1), the method initiates the card removal process and then waits as long as necessary until either the card is removed or an error occurs.

Remarks Starts the card ejection process.

If the device is a contactless IC card device, when this method is called, device starts the card ejection process and ejects the card and this method ends successfully at any time.

If the device is a contact IC card device with card detection sensor, this method completes when card ejection was detected.

If the device is a contact IC card device without card detection sensor, this method completes when this method is executed.

This method is called together with the **endRemoval** method that ends the card ejection process.

If the device cannot be set to the card ejection mode, an error exception will be fired, e.g., E_TIMEOUT. However, the device will remain in card ejection mode until **endRemoval** method is called.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_BUSY	cannot execute while asynchronous processing.
E_ILLEGAL	An invalid *timeout* parameter was specified.
E_TIMEOUT	The specified *timeout* has elapsed without the card being properly removed.

See Also **endRemoval** Method.

15.5.15 cancelValue Method

Syntax **cancelValue (sequenceNumber: *int32*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Cancels the Electronic value related operation specified by *sequenceNumber* on demand. The targeted cancellation operation is identified by the settlement number that is contained in the **ApprovalCode** property.

When *timeout* is FOREVER(-1), timeout never occurs and it waits indefinitely until it receives a response from the device.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **ApprovalCode** Property, **CapCancelValue** Property, **addValue** Method, **readValue** Method, **subtractValue** Method, **writeValue** Method.

15.5.16 captureCard Method

Syntax **captureCard ( ):**
void { raises-exception, use after open-claim-enable }

Remarks The card left in the slot is removed.

This method is effective, if the device is equipped with a card detection sensor. When the card insertion slot sensor detects the card, since the card ejection process was executed, application may call this method to keep and maintain the card.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_FAILURE	The device cannot capture the card.

See Also **DetectionStatus** Property.

15.5.17 cashDeposit Method *Added in Release 1.15*

Syntax **cashDeposit (sequenceNumber: *int32*, amount: *currency*, timeout: *int32*): void { raises-exception, use after open-claim-enable }**

Parameter	Description
sequenceNumber	Sequence number for charge.
amount	Amount of money for charge.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Charging amounts.

The *amount* is stored on the Electronic Money Device.

If *timeout* is FOREVER(-1), a timeout will not occur and the process will wait forever until the Electronic Money Device responds.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified, or **CapCashDeposit** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

See Also **CapCashDeposit** property.

15.5.18 checkCard Method *Added in Release 1.15*

Syntax **checkCard (sequenceNumber: *int32*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number for approval.
timeout	The maximum waiting time (in milliseconds) until the response is received from the EVRW device. FOREVER (-1), 0 and positive values can be specified.

Remarks Card Check is intended.

Card Check will be made as specified by *sequenceNumber*.

Electronic Money Device:
The check of the **Balance** will be done by the specified *sequenceNumber*. The **Balance** will be stored in the **Balance**

When *timeout* is FOREVER (-1), timeout never occurs and the device waits until it receives response from the EVRW.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid *timeout* parameter was specified, or **CapCheckCard** is false.
E_TIMEOUT	No response was received from EVRW during the specified *timeout* time in milliseconds.
E_EXTENDED	The detail code has been stored in *ErrorCodeExtended*.
E_BUSY	The EVRW device cannot accept any commands now.

See Also **Balance property, CapCheckCard property.**

15.5.19 checkServiceRegistrationToMedium Method *Added in Release 1.14.1*

Syntax **checkServiceRegistrationToMedium**
(sequenceNumber: *int32*, timeout: *int32*):
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks To a medium, it is checked whether electronic value service can be registered.

An UposException with E_EXTENDED is thrown when service cannot register to medium.

When *timeout* is FOREVER(-1), timeout never occurs and it waits indefinitely until it receives a response from the device.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **registerServiceToMedium** Method.

15.5.20 clearParameterInformation Method *Added in Release 1.14*

Syntax **clearParameterInformation ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Used to clear the all the tag values for the control set previously stored by the **setParameterInformation** method.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

See Also **setParameterInformation** Method.

15.5.21 closeDailyEVService Method *Added in Release 1.14.1*

Syntax **closeDailyEVService (inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
data	An array of one mutable integer whose specific values or usage vary by service.
obj	Additional data whose usage varies by service.

Remarks Executes the closing process of the service selected by **CurrentService** property..

If the device has the closing process function, it is supported.

The contents of processing are dependent on service.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

For consistency, a Service must always fire at least one **TransitionEvent** with an incomplete progress completion percentage (i.e. a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a **TransitionEvent** with a progress of 100. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The service does not have the closing process.
E_BUSY	The device cannot accept any commands now.

See Also **openDailyEVService** Method, **TransitionEvent** .

15.5.22 deactivateEVService Method *Added in Release 1.14.1*

Syntax **deactivateEVService (inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
data	An array of one mutable integer whose specific values or usage vary by service.
obj	Additional data whose usage varies by service.

Remarks Executes the device deactivation process.

If the device has the deactivation process function, it is supported.

The deactivation process is the terminate process performed when uninstalling a service or removing a device.

The contents of processing and the contents of the parameter are dependent on service.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

For consistency, a Service must always fire at least one **TransitionEvent** with an incomplete progress completion percentage (i.e. a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a **TransitionEvent** with a progress of 100. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have the deactivation.
E_BUSY	The device cannot accept any commands now.

See Also **activateEVService** Method, **TransitionEvent**.

15.5.23 endDetection Method

Syntax **endDetection ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Ends the card detection process.

When called, the device ends card detection mode. If the card is correctly detected in the device control is returned to the application. If the card cannot be detected an exception is delivered with its *ErrorCodeExtended* property set to EVRW_NOCARD.

This method is called together with the **beginDetection** method.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device is not in card detection mode.
E_EXTENDED	*ErrorCodeExtended*=EVRW_NOCARD: No card has been detected.

See Also **beginDetection** Method.

15.5.24 endRemoval Method

Syntax **endRemoval ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Ends the card removal process.

When called, the device ends the card removal mode. If the card is not detected in the device, control is returned to the application. If the card remains in the device, an exception is delivered with its *ErrorCodeExtended* property set to EVRW_RELEASE. If the device is contactless IC card, it is not necessary to implement this and control is always returned to the application without any exceptions.

This method is called together with the **beginRemoval** method for the card removal processing.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device is not in card removal mode.
E_EXTENDED	*ErrorCodeExtended*=EVRW_RELEASE: The card remains in the device.

See Also **beginRemoval** Method.

15.5.25 enumerateCardServices Method

Syntax **enumerateCardServices ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Enumerates the services which are used in the card and sets the **CardServiceList** property.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

See Also **CardServiceList** Property.

15.5.26 lockTerminal Method *Updated in Release 1.14.1*

Syntax **lockTerminal ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Sets the security lock on the device or the service. If the device or the service is locked, the device or the service cannot accept any commands except for unlockTerminal method.

AdditionalSecurityInformation property is set if key information is required to lock for the authentication.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have a security lock function. **CapLockTerminal** is false.
E_BUSY	The device cannot accept any commands now.

See Also **AdditionalSecurityInformation** Property, **CapLockTerminal** Property, **unlockTerminal** Method.

15.5.27 openDailyEVService Method *Added in Release 1.14.1*

Syntax **openDailyEVService (inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
data	An array of one mutable integer whose specific values or usage vary by service.
obj	Additional data whose usage varies by service.

Remarks Executes the opening process of the service selected by **CurrentService** property. If the device has the opening process function, it is supported. The contents of processing are dependent on service.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

For consistency, a Service must always fire at least one **TransitionEvent** with an incomplete progress completion percentage (i.e. a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a **TransitionEvent** with a progress of 100. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The service does not have the opening process.
E_BUSY	The device cannot accept any commands now

See Also **closeDailyEVService** Method, **TransitionEvent**.

15.5.28 queryLastSuccessfulTransactionResult Method *Added in Release 1.14*

Syntax **queryLastSuccessfulTransactionResult ( ):**
void { raises-exception, use after open-claim-enable }

Remarks This method is used to refresh the property values that resulted from last successful **readValue**, **writeValue**, **addValue**, **subtractValue**, **cancelValue**, and **accessLog** methods calls.

When the **readValue** method was last successfully executed, the property values will indicate the status at the time the **DataEvent** event or **ErrorEvent** event was sent. The tag name "TransactionType" will be set to the value of last successful transaction method call.

The **queryLast SuccessfulTransactionResult** method is necessary because there may be situations where a transaction result is unknown. This could be due to power failure or network communication interruptions occurring just before the transaction result updated. Some EVR/W devices support a function to provide the last transaction results from the device and this method utilizes this functionality.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

15.5.29 readValue Method

Syntax **readValue (sequenceNumber: *int32*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Reads the electronic value from the card.

Electronic value is read from the card specified by *sequenceNumber* on demand.

When *timeout* is FOREVER(-1), a timeout never occurs and the Service waits indefinitely until it receives a response from the device.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *Timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **addValue** Method, **cancelValue** Method, **subtractValue** Method, **writeValue** Method.

15.5.30 registerServiceToMedium Method *Added in Release 1.14*

Syntax **registerServiceToMedium**
(sequenceNumber: *int32*, timeout: *int32*):
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Electronic value service is registered to a medium.

When *timeout* is FOREVER(-1), timeout never occurs and it waits indefinitely until it receives a response from the device.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **checkServiceRegistrationToMedium** Method,
unregisterServiceToMedium Method.

15.5.31 retrieveResultInformation Method *Updated in Release 1.15*

Syntax **retrieveResultInformation (name: string, inout value: string):**
void { raises-exception, use after open, claim }

Parameter	Description
name	The tag name whose value is to be retrieved.
value	The string value for the tag specified by the *name* parameter. If the *name* parameter is not recognized or not supported for the current card type, the value returned will be an empty string (““).

Remarks The **retrieveResultInformation** method is used to associate a tag *name* with the data *value* that comes from the card that is being read.

The following table defines the tag *name* and associated information on its *value* and usage.

Tag *name*	Type** of String and Description
AccessLogLastDateTime	The ***Datetime*** of obtaining the last transaction log.
AccountNumber	Account ID ***String*** for electronic value service. Although it has the same information in a property, it is recommended to use this tag name/value.
Amount	Settlement ***Currency*** amount requested to the EVR/W. Although it has the same information in a property, it is recommended to use this tag name/value.
AmountForPoint	The ***Currency*** amount targeted for calculating points. The amount will be specified when the EVR/W device calculates the point values to be added at the same time as settlement, but there are some products not targeted for points.
AuthenticationStatus	The ***Enumerated*** number for the status of authentication.
AutoCharge	***Boolean*** for request to conduct an automatic charge at the time of issuing a method, or the result of automatic charge at the time of completing the process.
Balance	The ***Currency*** balance of electronic value service. Although it has the same information in a property, it is recommended to use this tag name/value.
BalanceOfPoint	The ***Currency*** balance of point service.Although it has the same information in a property, it is recommended to use this tag name/value.
BusinessUnitID	ID ***String*** for a store.
CardCompanyName	The ***String*** name of a company issuing electronic value media (card or mobile phone).
CardTransactionLogID	The ID ***String*** for transaction details stored in electronic value service media (card or mobile phone).
CardTransactionNumber	The transaction ***Number*** assigned and controlled by electronic value service media (card or mobile phone).
ChargeableAmount	The ***Currency*** amount for which charging is possible
ChargeableCount	The ***Number*** of times in which charging is possible.

ChargeMethod	The ***Enumerated*** value for the method to charge an electronic value service: 1. Cash 2. Exchanging points
DateTime	The ***Datetime*** of issuing a method, notifying an event, or completing a process.
EffectiveDaysOfKey	The ***Number*** of days the Key value is effective.
EndAccountID	The ending point specified by an account ID ***String*** when requesting closing or summary to the EVR/W.
EndDateTime	The ending point specified by the ***Datetime*** when requesting closing or summary to the EVR/W.
EndEVRWTransactionNumber	The ending point ***Number*** specified by the EVR/W transaction sequential number when requesting closing or summary to the EVR/W.
EndPOSTransactionNumber	The ending point ***Number*** specified by a POS transaction number when requesting closing or summary to the EVR/W.
EVRWApprovalCode	The approval code ***String*** for processing assigned and controlled by the EVR/W.
EVRWDataUpdateDateTime	The ***Datetime*** when the internal data of the EVR/W was updated.
EVRWDateTime	The ***Datetime*** managed by the EVR/W.
EVRWID	The ID ***Number*** of the EVR/W
EVRWTransactionLogID	The ID ***String*** for transaction details stored in the EVR/W
EVRWTransactionNumber	The transaction ***Number*** assigned and controlled by the EVR/W.
ExpirationDate	The expiration ***DateTime*** of the medium. Although it has the same information in a property, it is recommended to use this tag name/value.
ExpiredAccountID	The ***String*** description provided when information is held for an account already expired in the electronic value service media (card or mobile phone).
ForceOnlineCheck	***Boolean*** Specifies request to force the center to check online/offline status at the time of settlement.
InsufficientAmount	Insufficient ***Currency*** amount when the balance is found insufficient by the EVR/W.
ItemCode	The item code ***String*** for the product handled in the settled transaction.
KeyExpirationDateTime	The ***DateTime*** when the key expires.
KeyUpdateDateTime	The ***DateTime*** when the key of the EVR/W was last updated.
LastTimeBalance	***Currency*** Balance before settlement
LastTimeCardTransaction-LogID	The ID ***String*** for last time transaction details stored in electronic value service media (card or mobile phone).
LastTimeEVRWTransaction-LogID	The ID ***String*** for last time transaction details stored in the EVR/W.
LastUsedDateTime	The most recent used ***DateTime*** of the medium. Although it has the same information in a property, it is recommended to use this tag name/value.

LogCheck	***Boolean*** The flag to specify whether to check the transaction log when voiding the settlement.
MediaData	Information ***String*** data for electronic value media (card or mobile phone) that is not related to POS. The content can be freely set by service providers or vendors.
MediumID	The ID ***Number*** for electronic value service media (card or mobile phone). Although it has the same information in a property, it is recommended to use this tag name/value.
MediumIssuerInformation	The ***String*** containing the information on the issuer of the medium.
MemberInformation	The ***String*** containing the information of the membership certificate.
MerchantID	The ***String*** containing the merchant identification information.
ModuleID	The ID ***Number*** for individual settlement modules or applications that exist in the EVR/W that provides multiple services.
NegativeInformationType	The ***Enumerated*** value indicating the type of negative transaction information.
NegativeInformationUpdateDateTime	The ***DateTime*** when the negative information of the EVR/W was updated.
NumberOfAddition	The ***Number*** of charge settlement transactions
NumberOfEVRWTransactionLog	The ***Number*** of transaction details stored in the EVR/W.
NumberOfFreeEVRWTransactionLog	The ***Number*** value of free space for transaction details stored in the EVR/W
NumberOfRecord	The ***Number*** of records
NumberOfSentEVRWTransactionLog	The ***Number*** of transaction details that are stored in the EVR/W and have been sent to the settlement center.
NumberOfSubtraction	The ***Number*** of settlement transactions.
NumberOfTransaction	The total ***Number*** of transactions
NumberOfUncompletedAddition	The ***Number*** of transactions uncompleted due to communication error between the EVR/W and electronic value media (card or mobile phone) during the charge settlement transaction.
NumberOfUncompletedSubtraction	The ***Number*** of transactions uncompleted due to communication error between the EVR/W and electronic value media (card or mobile phone) during the settlement transaction.
NumberOfUncompletedVoid	The ***Number*** of transactions uncompleted due to communication error between the EVR/W and electronic value media (card or mobile phone) during voiding transaction.
NumberOfVoid	The ***Number*** of voiding transactions
OtherAmount	The ***Currency*** amount for extra payment when it is used for the transaction together with a regular settlement.
PaymentCondition	The ***Enumerated*** number for the type of payment for the settlement amount in case of post-pay type electronic value services.

PaymentDetail	The ***String*** data of the type of payment for the settlement amount in case of post-pay type electronic value services.
PaymentMethod	The ***Enumerated*** number for the amount required by the EVR/W, it specifies the type of settlement of transaction amount: 1. Full settlement 2. Settlement combined with another payment method.
PaymentMethodForPoint	The ***Enumerated*** value that represents the settlement method that is targeted for calculating points.
Point	The point value ***Number*** requested to the EVR/W from POS. Although it has the same information in a property, it is recommended to use this tag name/value.
POSDateTime	The ***Datetime*** of accounting managed by POS.
POSTransactionNumber	The sequential ***Number*** that identifies the POS transaction.
RegistrableServiceCapacity	The ***Number*** indicating the quantity of services that can be registered.
RequestedAutoChargeAmount	The ***Currency*** amount requested for automatic charge.
ResponseCode1	The primary result code ***Number*** for processing.The content can be freely set by service providers or vendors.
ResponseCode2	The secondary result code ***Number*** for detailed processing.The content can be freely set by service providers or vendors.
ResultOnSettlement	The ***Enumerated*** number for the result status of the settlement transaction between the EVR/W and electronic value media (card or mobile phone)
RetryTimeout	Timeout ***Number*** (in milliseconds) until the EVR/W is touched by electronic value media (card or mobile phone) when it is necessary to retry processing between the EVR/W and electronic value media (card or mobile phone)
SettledAmount	The ***Currency*** amount actually settled with the EVR/W. Although it has the same information in a property, it is recommended to use this tag name/value.
SettledAutoChargeAmount	The automatic charge ***Currency*** value actually settled by the EVR/W
SettledMemberInformation	The ***String*** which contains the member information in the membership certificate after it has been updated.
SettledOther-Amount	The actual ***Currency*** amount of extra payment when an electronic value service is used with other settlement methods.
SettledPoint	The point value ***Number*** actually settled by the EVR/W.
SetttledVoucherID	The ***String*** which contains the updated voucher ID.
SettlementNumber	The sequential ***Number*** for the clearing process.
SignatureFlag	***Boolean*** The flag to specify whether or not it is necessary to sign after settlement.
SoundAssistFlag	***Boolean*** The flag specifying whether or not to activate voice navigation.
StartAccountID	The starting point specified by a ***String*** account ID when requesting closing or summary to the EVR/W.
StartDateTime	The starting point specified by the ***Datetime*** when requesting closing or summary to the EVR/W.

StartEVRWTransactionNum-ber	The starting point ***Number*** specified by the EVR/W transaction sequential number when requesting closing or summary to the EVR/W.
StartPOSTransactionNumber	The starting point ***Number*** specified by a POS transaction number when requesting closing or summary to the EVR/W.
SummaryTermType	The ***Enumerated*** number that specifies the term for the summary process.
TargetService	The ***String*** which contains the information about the target service.
TaxOthers	Tax and other ***Currency*** amounts included in the settlement amount required by the EVR/W.
TotalAmountOfAddition	The total ***Currency*** amount of charge settlement transactions
TotalAmountOfSubtraction	Total ***Currency*** amount of settlement transactions.
TotalAmountOfTransaction	The total ***Currency*** amount of the transactions.
TotalAmountOfUncomplete-dAddition	The total ***Currency*** amount of transactions not completed due to communication errors between the EVR/W and electronic value media (card or mobile phone) during the charge settlement transaction.
TotalAmountOfUncompleted-Subtraction	The total ***Currency*** amount of transactions not completed due to communication errors between the EVR/W and electronic value media (card or mobile phone) during the transaction settlement.
TotalAmountOfUncompleted-Void	The total ***Currency*** amount of transactions not completed due to communication errors between the EVR/W and electronic value media (card or mobile phone) during voiding transactions.
TotalAmountOfVoid	The total ***Currency*** amount of voided transactions.
TouchTimeout	Timeout ***Number*** (in milliseconds) until the EVR/W is touched by electronic value media (card or mobile phone).
TransactionType	The ***Enumerated*** number for the type of transaction for the electronic value service.
UILCDControl	***Boolean*** Specifies whether or not a LCD is controlled if the EVR/W has a LCD.
UILEDControl	***Boolean*** Specifies whether or not a LED iscontrolled if the EVR/W has a LCD.
UISOUNDControl	***Boolean*** Specifies whether or not sound is controlled if EVR/W has sounds.
VOIDorRETURN	The ***Enumerated*** value for how a transaction is voided: 1. Void 2. Return
VoidTransactionType	The ***Enumerated*** value for the type of transaction to be voided: 1. Cash 2. Exchanging points
VoucherID	The ID ***String*** of the voucher/ticket.
VoucherIDList	The enumerated IDs ***String*** of the voucher/ticket.
WorkstationID	ID ***String*** for POS.

WorkstationMaker	The ***String*** which identifies the manufacturer's code of the workstation manufacturer.
WorkstationSerialNumber	The ***String*** which contains the manufacturer's serial number or the identification code of the POS workstation.

All the values for the tags are typed as character strings. The following table shows the format for the string values.

Type**	**Format**
String	Text character string.
Number	32 bit Integer value represented by text characters.
Currency	64 bit Integer value represented by text characters.The 4 fixed numbers of digits define below a decimal point. For example, if the integer is "1234567," then the currency value is "123.4567."
Datetime	Datetime format is: yyyy '-' mm '-' dd 'T' hh ':' mm ':' ss '.' sss zzzzzz where '-' is the character separator between the date elements. yyyy is a 4-digits numeral representing the year. mm is a 2-digits numeral representing the month (from 01 to 12) . dd is a 2-digits numeral representing the day of the month (from 01 to 31). 'T' is the character separator between the date and the time. ':' is the character separator between the time elements. hh is a 2-digits numeral representing the hours (from 00 to 23). mm (the second one) is a 2-digits numeral representing the minute (from 00 to 59). ss is a 2-digits numeral representing the integer part of the seconds (from 00 to 59). '.' is the character separator between the time and the fractional seconds. sss is a 1-digit to 3-digits numeral representing the fractional seconds. zzzzzz represent the time zone which is the character 'Z' for a GMT time, or the delta from the GMT time, with a string of the form (('+' | '-') hh ':' mm) where '+' represent a positive delta from the GMT time '-' represent a negative delta from the GMT time hh is a 2-digits numeral representing the delta hours (from 00 to 14) mm is a 2-digits numeral representing the delta minute (from 00 to 59) Requesting a mandatory time zone resolves the problem of Daylight Saving Time or Summer Time, because the time is absolute. Examples 2008-04-12T23:20:50.275 represents the date of 12 April 2008 on the local time of 20 minutes, 50 seconds and 275 milliseconds past 23 hours. 2008-04-12T22:20:50.275+01:00 represents the same date and time in Geneva. 2008-04-12T17:20:50.275-05:00 represents the same date and time in New-York.
Boolean	A logical type of string value "True" or "False."
Enumerated	One of the text character strings defined by each tag.

The following values are used for the ***Enumerated*** tags.

Tag	Definition	Remarks
Authentication Status	EVRW_TAG_AS_AUTHENTICATED	Authenticated
	EVRW_TAG_AS_UNAUTHENTICATED	Unauthenticated
Cancel Transaction Type	EVRW_TAG_CTT_CANCEL	Canceling
	EVRW_TAG_CTT_CHARGE	Canceling charge
	EVRW_TAG_CTT_RETURN	Return
	EVRW_TAG_CTT_SALES	Canceling sales
Charge Method	EVRW_TAG_CM_CASH	Charge by cash
	EVRW_TAG_CM_CREDIT	Charge by credit
	EVRW_TAG_CM_POINT	Charge by points
Negative Information Type	EVRW_TAG_NIT_ALL	Full list of negative settlement information.
	EVRW_TAG_NIT_UPDATED	Updated list of negative settlement information

Payment Condition	EVRW_TAG_PC_ INSTALLMENT_2	Installment 2
	EVRW_TAG_PC_ INSTALLMENT_3	Installment 3
	EVRW_TAG_PC_BONUS_1	Bonus 1
	EVRW_TAG_PC_BONUS_2	Bonus 2
	EVRW_TAG_PC_BONUS_3	Bonus 3
	EVRW_TAG_PC_BONUS_4	Bonus 4
	EVRW_TAG_PC_BONUS_5	Bonus 5
	EVRW_TAG_PC_BONUS_COMBINATION_1	With extra payment by bonus 1
	EVRW_TAG_PC_BONUS_COMBINATION_2	With extra payment by bonus 2
	EVRW_TAG_PC_BONUS_COMBINATION_3	With extra payment by bonus 3
	EVRW_TAG_PC_BONUS_COMBINATION_4	With extra payment by bonus 4
	EVRW_TAG_PC_INSTALLMENT_1	Installment 1
	EVRW_TAG_PC_LUMP	Lump-sum
	EVRW_TAG_PC_REVOLVING	Revolving
Payment Method	EVRW_TAG_PM_COMBINED	Settlement combined with other payment
	EVRW_TAG_PM_FULL_SETTLEMENT	Full settlement
Payment Method ForPoint	EVRW_TAG_PMFP_CASH	Cash
	EVRW_TAG_PMFP_CREDIT	Credit card
	EVRW_TAG_PMFP_EM	Electronic money
	EVRW_TAG_PMFP_OTHER	Other
ResultOnSet-tlement	EVRW_TAG_ROS_NG	Abnormal termination
	EVRW_TAG_ROS_OK	Normal termination
	EVRW_TAG_ROS_UNKNOWN	Unidentified

Summary TermType	EVRW_TAG_STT_1	From the previous type of summary result to current.
	EVRW_TAG_STT_2	From the summary result before the previous type of result to the previous summary result.
	EVRW_TAG_STT_3	From the summary result two times before the previous type of summary result to the summary result before the previous result.
Transaction-Type	EVRW_TAG_TT_ADD	Adding (Charge)
	EVRW_TAG_TT_CANCEL_CHARGE	Canceling charge
	EVRW_TAG_TT_CANCEL_RETURN	Canceling/Return
	EVRW_TAG_TT_CANCEL_SALES	Canceling sales
	EVRW_TAG_TT_COMPLETION	Authorizing com-pletion
	EVRW_TAG_TT_GET_LOG	Acquiring a transaction log
	EVRW_TAG_TT_PRE-SALES	Authorizing pre-sales
	EVRW_TAG_TT_READ	Reading (Reference)
	EVRW_TAG_TT_RETURN	Return
	EVRW_TAG_TT_SUBTRACT	Subtracting (Sales)
	EVRW_TAG_TT_WRITE	Writing

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

15.5.32 setParameterInformation Method *Added in Release 1.14*

Syntax **setParameterInformation (name: string, value: string):void**
{ raises-exception, use after open, claim }

Parameter	Description
name	The tag used to identify the specific card data item.
value	The string value associated with the tag *name*. If the *name* parameter is not recognized or not supported for the current card type, the value returned will be an empty string (““).

Remarks The **setParameterInformation** method is used to associate a tag *name* with additional the data *value* parameters that are associated with the card that is being read. Refer to explanation of a **retrieveResultInformation** method for the tags and values that can be used.

The application can call a **clearParameterInformation** method which will set the *value* to the empty string (““).

Errors A UposException may be thrown when this method is invoked. For further information, see **“Errors”** on page 1- 16.

See Also **clearParameterInformation** Method, **retrieveResultInformation** Method.

15.5.33 subtractValue Method

Syntax **subtractValue (sequenceNumber: *int32*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Subtracts the electronic value from the card.

Electronic value is subtracted from the card specified by *sequenceNumber* on demand.

When *timeout* is FOREVER(-1), timeout never occurs and the Service waits indefinitely until it receives a response from the device.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **CapSubtractValue** Property, **addValue** Method, **cancelValue** Method, **readValue** Method, **writeValue** Method.

15.5.34 transactionAccess Method

Syntax **transactionAccess (control: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	**Description**
control	The transaction control, can be set to one of the following values:

Value	**Meaning**
EVRW_TA_TRANSACTION	Begin a transaction
EVRW_TA_NORMAL	End the transaction mode by executing the buffer operation.

Remarks Enters or exits transaction mode.

If *control* is EVRW_TA_TRANSACTION, then transaction mode is entered. Subsequent calls to **readValue**, **writeValue**, **addValue**, **subtractValue**, and **cancelValue** will buffer the data until **transactionAccess** is called with the *control* parameter set to EVRW_TA_NORMAL. It depends on the implementation if buffering is done in the EVR/W device or buffering is done within the Service.

If *control* is EVRW_TA_NORMAL, then transaction mode is exited. If some requests were buffered by calls to the methods **readValue**, **writeValue**, **addValue**, **subtractValue**, and **cancelValue**, then the buffered requests will be executed.

The entire transaction requests are treated as one message. This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Calling the **clearOutput** method cancels transaction mode. Any buffered print lines are also cleared.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

See Also **AsyncMode** Property, **CapTransaction** Property, **addValue** Method, **cancelValue** Method, **readValue** Method, **subtractValue** Method, **writeValue** Method.

15.5.35 unlockTerminal Method *Updated in Release 1.14.1*

Syntax **unlockTerminal ( ):**
void { raises-exception, use after open-claim-enable }

Remarks Releases the security lock on the device or the service. When the device has a security lock function, it is supported. **AdditionalSecurityInformation** property is set when key information is required to release the lock.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16. Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have a security lock function. **CapUnlockTerminal** is false.
E_BUSY	The device cannot accept any commands now.

See Also **AdditionalSecurityInformation** Property, **CapUnlockTerminal** Property **lockTerminal** Method.

15.5.36 unregisterServiceToMedium Method *Added in Release 1.14.1*

Syntax **unregisterServiceToMedium**
(sequenceNumber: *int32*, timeout: *int32*):
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Electronic value service is deleted from a medium.

When *timeout* is FOREVER(-1), timeout never occurs and it waits indefinitely until it receives a response from the device. This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **registerService** Method.

15.5.37 updateData Method *Added in Release 1.14.1*

Syntax **updateData (dataType:*int32*, inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
dataType	Type of the data which accesses

Value	Meaning
EVRW_AD_KEY	Key information.
EVRW_AD_NEGATIVE_LIST	Negative list.
EVRW_AD_OTHERS	Other information.

data	An array of one mutable integer whose specific values or usage vary by service.
obj	Additional data whose usage varies by service.

Remarks The data of an EVR/W is updated.

The contents of data are dependent on service.

This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

For consistency, a Service must always fire at least one **TransitionEvent** with an incomplete progress completion percentage (i.e., a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a **TransitionEvent** with a progress of 100. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have the activation.
E_BUSY	The device cannot accept any commands now.

See Also **accessData** Method, **TransitionEvent**.

15.5.38 updateKey Method *Updated in Version 1.14.1*

Syntax **updateKey (inout data: *int32*, inout obj: *object*):**
void { raises-exception, use after open-claim-enable }

Remarks Updates the key information in the device. If the device has the function to the key information, it is supported.
The content of processing and the content of the parameter depend on the implementation.

Added in Release 1.14.1:*For consistency, a Service must always fire at least one* ***TransitionEvent*** *with an incomplete progress completion percentage (i.e., a percentage between 1 and 99), even if the device cannot physically report the progress of the process. If the process completes successfully, the Service must fire a* ***TransitionEvent*** *with a progress of 100. These Service*

requirements allow applications using this method to be designed to always expect some level of progress notification.

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	The device does not have the update function of key information.
E_BUSY	The device cannot accept any commands now.

See Also **TransitionEvent**

15.5.39 writeValue Method

Syntax **writeValue (sequenceNumber: *int32*, timeout: *int32*):**
void { raises-exception, use after open-claim-enable }

Parameter	Description
sequenceNumber	Sequence number
timeout	The maximum waiting time (in milliseconds) until the response is received from the device. FOREVER(-1), 0, and positive values can be specified.

Remarks Writes the electronic value in the card. Electronic value is written in the card specified by *sequenceNumber* on demand. When *timeout* is FOREVER(-1), timeout never occurs and it waits indefinitely until it receives a response from the device. This method is performed synchronously if **AsyncMode** is false, and asynchronously if **AsyncMode** is true**.**

Errors A UposException may be thrown when this method is invoked. For further information, see **"Errors"** on page 1- 16.

Some possible values of the exception's *ErrorCode* property are:

Value	Meaning
E_ILLEGAL	Invalid or unsupported parameter was specified.
E_TIMEOUT	No response was received from device during the specified *timeout* in milliseconds.
E_BUSY	The device cannot accept any commands now.

See Also **CapWriteValue** Property, **addValue** Method, **cancelValue** Method, **readValue** Method, **subtractValue** Method.

15.6 Events (UML interfaces)

15.6.1 DataEvent

<< event >> upos::events::DataEvent
Status: *int32* { read-only }

Description Notifies the application about the available input data from the device.

Attributes This event contains the following attribute:

Attributes	Type	Description
Status	*int32*	The *Status* parameter contains zero.

Remarks Before this event is delivered, the data is set into the appropriate property.

See Also **"Events"** on page 1- 15.

15.6.2 DirectIOEvent

<< event >> upos::events::DirectIOEvent
EventNumber: *int32* { read-only }
Data: *int32* { read-write }
Obj: *object* { read-write }

Description Provides Service information directly to the application. This event provides a means for a vendor-specific EVR/W Service to provide events to the application that are not otherwise supported by the Control.

Attributes This event contains the following attributes:

Attribute	Type	Description
EventNumber	*int32*	Event number whose specific values are assigned by the Service.
Data	*int32*	Additional numeric data. Specific values vary by the *EventNumber* and the Service. This property is settable.
Obj	*Object*	Additional data whose usage varies by the *EventNumber* and Service. This property is settable.

Remarks This event is to be used only for those types of vendor specific functions that are not otherwise described. Use of this event may restrict the application program from being used with other vendor's EVR/W devices which may not have any knowledge of the Service's need for this event.

See Also **"Events"** on page 1- 15, **directIO** Method.

15.6.3 ErrorEvent

<< event >> upos::events::ErrorEvent
ErrorCode: *int32* { read-only }
ErrorCodeExtended: *int32* { read-only }
ErrorLocus: *int32* { read-only }
ErrorResponse: *int32* { read-write }

Description Notifies the application that an EVR/W error has been detected and a suitable response by the application is necessary to process the error condition.

Attributes This event contains the following attributes:

Attributes	Type	Description
ErrorCode	*int32*	Error code causing the error event.
ErrorCodeExtended	*int32*	Extended Error code causing the error event. If *ErrorCode* is E_EXTENDED, then see values below. Otherwise, it may contain a Service-specific value.
ErrorLocus	*int32*	Location of the error. See values below.
ErrorResponse	*int32*	Error response, whose default value may be overridden by the application. (i.e., this property is settable). See values below.

If *ErrorCode* is E_EXTENDED, then *ErrorCodeExtended* has one of the following values:

Value	Meaning
EVRW_CENTERERROR	An error was returned from the approval agency.
EVRW_COMMANDERROR	The command sent to the device is wrong. This error is never returned so long as device control is working correctly.
EVRW_RESET	The device was stopped during processing by device reset key (stop key) and so on.
EVRW_COMMUNICATIONERROR	Communication error has occurred between the approval agency (center) and device.
EVRW_LOGOVERFLOW	Transaction log was too big to be stored. Getting transaction log has been stopped and the value of **TransactionLog** is uncertain.
EVRW_DAILYLOGOVERFLOW	Try to processing, a failure will occur if the transaction log on the device is full and cannot be settle.
EVRW_DEFICIENT	Because the balance is insufficient, it cannot be subtracted.
EVRW_OVERDEPOSIT	Because the amount that was able to be charged was exceeded, it cannot be added.

The *ErrorLocus* property may be one of the following:

Value	Meaning
EL_OUTPUT	Error occurred while processing asynchronous output.
EL_INPUT	Error occurred while gathering or processing event-driven input. No previously buffered input data is available.
EL_INPUT_DATA	Error occurred while gathering or processing event-driven input, and some previously buffered data is available.

The contents of the *ErrorResponse* property are preset to a default value, based on the *ErrorLocus*. The application's error processing may change *ErrorResponse* to one of the following values:

Value	Meaning
ER_RETRY	Typically valid only when locus is EL_OUTPUT. Retry the asynchronous output. The error state is exited. May be valid when locus is EL_INPUT. Default when locus is EL_OUTPUT.
ER_CLEAR	Clear all buffered output data (including all asynchronous output) or buffered input data. The error state is exited. Default when locus is EL_INPUT.
ER_CONTINUEINPUT	Used only when locus is EL_INPUT_DATA. Acknowledges the error and directs the Control to continue processing. The Control remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and the **DataEventEnabled** property is again set to true, then another **ErrorEvent** is delivered with locus EL_INPUT. Default when locus isEL_INPUT_DATA.

Remarks Notifies when the error is detected when a method is asynchronously executed, and the state of the control moves to the error state.

Input error events are generated when errors occur while reading the data from a card, directed by **readValue** method. These error events are not delivered until the **DataEventEnabled** property is set to true so as to allow proper application sequencing. All error information is placed into the applicable properties before this event is delivered.

Output error events are generated and delivered when errors occur during asynchronous output processing. The errors are placed into the applicable properties before the events are delivered.

See Also "**Events**" on page 1- 15.

15.6.4 OutputCompleteEvent

<< event >> upos::events::OutputCompleteEvent
OutputID: *int32* { read-only }

Description Notifies the application that the queued asynchronous output request associated with the *OutputID* attribute has completed successfully.

Attributes This event contains the following attribute:

Attributes	Type	Description
OutputID	*int32*	The ID number of the asynchronous output request that is complete.

Remarks This event is enqueued after the request's data has been both sent and the Service has confirmation that it was processed by the device successfully.

See Also **"Device Output Models"** on page 1- 21.

15.6.5 StatusUpdateEvent

<< event >> upos::events::StatusUpdateEvent
Status: *int32* { read-only }

Description Notifies the application when the device detects a status change.

Attributes This event contains the following attribute:

Attribute	Type	Description
Status	*int32*	The status condition of the EVR/W.

The *Status* attribute has one of the following values:

Value	Description
EVRW_SUE_LS_OK	The transaction log has enough capacity.
EVRW_SUE_LS_NEARFULL	The transaction log is nearly full.
EVRW_SUE_LS_FULL	The transaction log is full.
EVRW_SUE_DS_NOCARD	The card detection sensor does not detect the card.
EVRW_SUE_DS_DETECTED	The card detection sensor detected the card.
EVRW_SUE_DS_ENTERED	The insertion slot sensor detected the card.
EVRW_SUE_DS_CAPTURED	The stock space sensor detected the card.

Note that Release 1.3 added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability, added in ***Release 1.9***, added additional *Status* values for communicating the status/progress of an asynchronous update firmware process.

See description **"StatusUpdateEvent"** in Chapter 1.

Remarks This event is enqueued when a EVR/W detection undergoes a change or if Power State Reporting is enabled and a change in the power state is detected.

The state of the transaction log is reported only if **CapLogStatus** is true.

See Also **CapLogStatus** Property, **LogStatus** Property, "**Events**" on page 1- 15.

15.6.6 TransitionEvent *Updated in Release 1.15*

<< event >> upos::events::TransitionEvent

EventNumber: int32 { read-only }

pData:int32{ read-write }

pString:string{ read-write }

Description Notifies the application that an important device process condition has occurred during an asynchronous I/O operation and a suitable response is necessary by the application.

Note: *In the OPOS environment, the format of this data depends upon the value of the* ***BinaryConversion*** *property. See* ***BinaryConversion*** *property in Annex A.*

Attributes This event contains the following attribute:

Attribute	Type	Description
EventNumber	int32	The ID number of the asynchronous I/O device process condition that is the cause for the event.
pData	int32	Additional information about appropriate response which is dependent upon the specific process condition.
pString	string	Information about the specific event that has occurred.

The *EventNumber* attribute has one of the following values:

Value	Description
EVRW_TE_NOTIFY_TOUCH_RETRY	Update retry notification Notification of retouching request (Retouching cannot be canceled until a certain period of time passes)
EVRW_TE_NOTIFY_TOUCH_RETRY_CANCELABLE	Update retry notification (can be canceled) Notification of retouching request (Retouching can be canceled at any time)
EVRW_TE_CONFIRM_TOUCH_RETRY	Confirmation of update retry (continued or canceled) At the time of completing the event, it specifies in *pData* whether to continue waiting for retouching (1), or to cancel (0).
EVRW_TE_CONFIRM_CANCEL	Confirmation of process cancellation At the time of completing the event, it specifies in *pData* whether to cancel the process (1), or to continue (0).
EVRW_TE_NOTIFY_INVALID_OPERATION	Notification of issuing an invalid operation The event code is set in *pData*

EVRW_TE_CONFIRM_INVALID_OPERATION
Confirmation of invalid operation
The event code is set in *pData*. Specifies whether to continue the process (1), or to terminate the process abnormally (0).

EVRW_TE_CONFIRM_REMAINDER_SUBTRACTION
Confirmation of insufficient funds and the deductible amount from the balance.
The balance is set in **Balance** property during notification. After completing the event, specify in *pData* whether to deduct all the balance (1), or to cancel (0).

EVRW_TE_CONFIRM_CENTER_CHECK
Confirmation of a center check
At the time of completing the event, specify in *pData* whether to conduct a center check (1), or not (0).

EVRW_TE_CONFIRM_TOUCH_TIMEOUT
Confirmations of timeout to wait for touching
At the time of completing the event, specify in *pData* whether to continue touching (1) or not (0).

EVRW_TE_CONFIRM_AUTO_CHARGE
Confirmation of automatic charge
At the time of completing the event, specify in *pData* whether to continue touching (1) or not (0).

EVRW_TE_NOTIFY_CAPTURE_CARD
Notification of card detection

EVRW_TE_NOTIFY_CENTER_CHECK
Notification of center checkis being conducted.

EVRW_TE_NOTIFY_COMPLETE
Notification of process completion.
Used when it is necessary to provide this information before same information is available through an **OutputCompleteEvent** event.

EVRW_TE_NOTIFY_PIN Notification that PIN input data is available in the PIN input status

EVRW_TE_NOTIFY_TOUCH
Status Notification of waiting for touching.

EVRW_TE_NOTIFY_BUSY
Status Notification that a processis underway requires some time before it is completed.

EVRW_TE_CONFIRM_CENTER_CHECK_COMPLETE
The confirmation that a center check has been completed.
After the check is completed, specify in *pData* whether to continue the process after the completion (1) or cancel the process (0).

EVRW_TE_CONFIRM_SELECT

Confirmation of settlement option when there are options available for settlement. Options are set in *pString* in CSV format.
After completing the event, specify in *pData* the selected element number, starting with number 1).

EVRW_TE_NOTIFY_LOCK

Notification that unlocking card or device is required.
Notifies that a user must unlock the card (mobile phone) which is currently in a locked state.

EVRW_TE_NOTIFY_CENTER_CHECK_COMPLETE

Notifies that a center check has finished.

EVRW_TE_NOTIFY_PROGRESS_1_TO_100

Notification of process progress The process has successfully completed 1 to 100 percent of the total operation.

EVRW_TE_CONFIRM_DEVICE_DATA

The required confirmation of a data event.
The confirmation of a data event occurs when an EVR/W device requires the delivery of data during processing of a method call. The data is delivered by using the **AddditionalSecurityInformation** property.

EVRW_TE_CONFIRM_PIN_ENTRY_BY_OUTER_PINPAD

Requesting PIN input from an external device.Confirmation of PIN input request from an external PIN input device. The *pData* is used to specify whether to cancel the process at the time of event completion (0), or to continue the process (1).
To continue the process, specify in *pString* the PIN data acquired from the PIN pad device. When the effective PIN is not obtained from a PIN pad, (2) it is returned in *pData*.

EVRW_TE_CONFIRM_SEARCH_TABLE

Confirmation of table search request.
The encrypted information block is passed through the AdditionalSecurityInformation property. The content of the information block and the method of encryption are implementation dependent.

EVRW_TE_CONFIRM_PAYMENT_CONDITION

Confirmation of payment method selection request.
At event notification, pString lists selectable payment method strings in CSV format. The character string indicating the payment method is the value of the enumerator that can be specified in the PaymentCondition tag. At the end of the event, specify both the PaymentCondition tag enumerator that indicates the payment method in the pData argument and the payment type details in the CSV format as the pString argument. The CSV format that defines the details of the payment type follows the specification of the PaymentCondition property.

EVRW_TE_CONFIRM_AUTHORIZE

Confirmation of authorization communication request.
The encrypted information block is passed through the AdditionalSecurityInformation property. The content of the information block and the method of encryption are implementation dependent.

EVRW_TE_NOTIFY_CHECK_CARD

Notification of card check.

EVRW_TE_NOTIFY_SELECT_PAYMENT_CONDITION

Notification of payment method selection.

The event codes specified in *pData* during the *EventNumber*(s) EVRW_TE_NOTIFY_INVALID_OPERATION and EVRW_TE_CONFIRM_INVALID_OPERATION have the following meanings.

***PData* Parameter**	**Description**
1	Mismatch of a retouched card
2	Card authentication error
3	An uncompleted process occurs again when requesting re-touching.
4	Failure of PIN input
5	Requests processing after a detailed check.
6	Mismatch of cards
7	Detects multiple cards
8	Detects a card with the balance at 0.

Remarks This event is enqueued when the EVR/W process requires notification of application or device service of impending activity that requires immediate action or response.

See Also "**Events**" on page 1- 15.