**UnifiedPOS Retail Peripheral Architecture**

**Version 1.16 RCSD**

**International Standard**

**For Implementation of Point Of Service Peripherals**

**OMG Document Number: retail/2020-02-xx**

**Original submission date: February 18th, 2019**

**Standard document URL: https://www.omg.org/spec/UPOS/**

**This proposal adds to and extends the UPOS 1.15 standard.**

Copyright © 2017, Object Management Group, Inc.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

IPR Mode

This specification is issued under the RAND Mode based on the OMG IPR Policy.

OMG IPR Policy https://www.omg.org/cgi-bin/doc.cgi?ipr

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 109 Highland Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

IMM®, MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

**OMG’s Issue Reporting Procedure**

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/report\_issue.)

Document Submitter

VINX Corp.

Document Publishing Supporters

OPOS-J

Sorimachi Giken Co. Ltd.

Microsoft Japan Ltd.

SEIKO EPSON Corp.

Toshiba TEC Corp.

Star Micronics Corp.

Fujitsu Frontec Corp.

NCR Corporation

Sharp Corporation

Omron Social Solutions Corp.

NEC Platforms Corp.

Transaction Media Networks Inc.

[Preface 13](#_Toc32697236)

[UPOS 1.16 RCSD Specification Overview 16](#_Toc32697237)

[**Updated Items in Release 1.16** 16](#_Toc32697238)

[**Updated Items in CHAPTER 21 Lights** 16](#_Toc32697239)

[**Updated Items in CHAPTER 29 POS Power** 16](#_Toc32697240)

[**Added Chapters in Release 1.16** 16](#_Toc32697241)

[Chapter 21 17](#_Toc32697242)

[Lights 17](#_Toc32697243)

[Summary 17](#_Toc32697244)

[General Information 20](#_Toc32697245)

[**Capabilities** 20](#_Toc32697246)

[**Device Sharing** 20](#_Toc32697247)

[**Lights Class Diagram** 21](#_Toc32697248)

[**Lights Sequence Diagram** 22](#_Toc32697249)

[Properties (UML attributes) 24](#_Toc32697250)

[**CapAlarm Property** 24](#_Toc32697251)

[**CapBlink Property** 24](#_Toc32697252)

[**CapColor Property** 24](#_Toc32697253)

[**CapPattern Property *Added in Release 1.16*** 25](#_Toc32697254)

[**MaxLights Property** 25](#_Toc32697255)

[Methods (UML operations) 26](#_Toc32697256)

[**switchOff Method** 26](#_Toc32697257)

[**switchOffPattern Method** 26](#_Toc32697258)

[**switchOn Method *Updated in Release 1.16*** 27](#_Toc32697259)

[**switchOnMultiple Method** ***Added in Release 1.16*** 28](#_Toc32697260)

[**switchOnPattern Method** ***Added in Release 1.16*** 29](#_Toc32697261)

[Events (UML interfaces) 30](#_Toc32697262)

[**DirectIOEvent** 30](#_Toc32697263)

[**StatusUpdateEvent** 30](#_Toc32697264)

[Chapter 29 31](#_Toc32697265)

[POS Power 31](#_Toc32697266)

[Summary 31](#_Toc32697267)

[General Information 35](#_Toc32697268)

[**Capabilities** 35](#_Toc32697269)

[**Device Sharing** 35](#_Toc32697270)

[Model 36](#_Toc32697271)

[**POSPower Class Diagram** ***Updated in Release 1.16*** 37](#_Toc32697272)

[**POSPower Sequence Diagram** 38](#_Toc32697273)

[**POSPower Standby Sequence Diagram** 39](#_Toc32697274)

[**POSPower State Diagram** 40](#_Toc32697275)

[**POSPower PowerState Diagram - Part 1** 41](#_Toc32697276)

[**POSPower PowerState Diagram - Part 2** 42](#_Toc32697277)

[**POSPower PowerState Diagram - Part 3** 43](#_Toc32697278)

[**POSPower State Chart Diagram for Fan and Temperature** 44](#_Toc32697279)

[**POSPower Battery State Diagram** 45](#_Toc32697280)

[**POSPower Power Transitions State Diagram** 46](#_Toc32697281)

[Properties (UML attributes) 47](#_Toc32697282)

[**BatteryCapacityRemaining Property** 47](#_Toc32697283)

[**BatteryCapacityRemainingInSeconds Property** ***Added in Release 1.16*** 47](#_Toc32697284)

[**BatteryCriticallyLowThreshold Property** 47](#_Toc32697285)

[**BatteryCriticallyLowThresholdInSeconds Property**   ***Added in Release 1.16*** 48](#_Toc32697286)

[**BatteryLowThreshold Property** 48](#_Toc32697287)

[**BatteryLowThresholdInSeconds Property**  ***Added in Release 1.16*** 48](#_Toc32697288)

[**CapBatteryCapacityRemaining Property** 49](#_Toc32697289)

[**CapBatteryCapacityRemainingInSeconds Property**   ***Added in Release 1.16*** 49](#_Toc32697290)

[**CapChargeTime Property**   ***Added in Release 1.16*** 49](#_Toc32697291)

[**CapFanAlarm Property** 49](#_Toc32697292)

[**CapHeatAlarm Property** 50](#_Toc32697293)

[**CapQuickCharge Property** 50](#_Toc32697294)

[**CapRestartPOS Property** 50](#_Toc32697295)

[**CapShutdownPOS Property** 50](#_Toc32697296)

[**CapStandbyPOS Property** 51](#_Toc32697297)

[**CapSuspendPOS Property** 51](#_Toc32697298)

[**CapUPSChargeState Property** 51](#_Toc32697299)

[**CapVariableBatteryCriticallyLowThreshold Property** 52](#_Toc32697300)

[**CapVariableBatteryCriticallyLowThresholdInSeconds Property**   ***Added in Release 1.16*** 52](#_Toc32697301)

[**CapVariableBatteryLowThreshold Property** 52](#_Toc32697302)

[**CapVariableBatteryLowThresholdInSeconds Property**   ***Added in Release 1.16*** 52](#_Toc32697303)

[**ChargeTime Property**   ***Added in Release 1.16*** 53](#_Toc32697304)

[**EnforcedShutdownDelayTime Property** 53](#_Toc32697305)

[**PowerFailDelayTime Property** 53](#_Toc32697306)

[**PowerSource Property** 54](#_Toc32697307)

[**QuickChargeMode Property** 54](#_Toc32697308)

[**QuickChargeTime Property** 54](#_Toc32697309)

[**UPSChargeState Property** 55](#_Toc32697310)

[Methods (UML operations) 56](#_Toc32697311)

[**restartPOS Method** 56](#_Toc32697312)

[**shutdownPOS Method** 56](#_Toc32697313)

[**standbyPOS Method** 57](#_Toc32697314)

[**suspendPOS Method** 57](#_Toc32697315)

[Events (UML Interfaces) 58](#_Toc32697316)

[**DirectIOEvent** 58](#_Toc32697317)

[**StatusUpdateEvent** 58](#_Toc32697318)

[Chapter 39 61](#_Toc32697319)

[Video Capture 61](#_Toc32697320)

[**Summary** 61](#_Toc32697321)

[General Information 66](#_Toc32697322)

[**Capabilities** 66](#_Toc32697323)

[**Video Capture Class Diagram** 67](#_Toc32697324)

[Model 68](#_Toc32697325)

[**Modes** 68](#_Toc32697326)

[**Device behaviors** 68](#_Toc32697327)

[**Photo Mode** 68](#_Toc32697328)

[**Video Mode** 69](#_Toc32697329)

[**Device Sharing** 70](#_Toc32697330)

[Properties (UML attributes) 71](#_Toc32697331)

[**AutoExposure Property** 71](#_Toc32697332)

[**AutoFocus Property** 71](#_Toc32697333)

[**AutoGain Property** 71](#_Toc32697334)

[**AutoWhiteBalance Property** 72](#_Toc32697335)

[**Brightness property** 72](#_Toc32697336)

[**CapAssociatedHardTotalsDevice Property** 72](#_Toc32697337)

[**CapAutoExposure Property** 72](#_Toc32697338)

[**CapAutoFocus Property** 73](#_Toc32697339)

[**CapAutoGain Property** 73](#_Toc32697340)

[**CapAutoWhiteBalance Property** 73](#_Toc32697341)

[**CapBrightness Property** 73](#_Toc32697342)

[**CapContrast Property** 73](#_Toc32697343)

[**CapExposure Property** 74](#_Toc32697344)

[**CapGain Property** 74](#_Toc32697345)

[**CapHorizontalFlip Property** 74](#_Toc32697346)

[**CapHue Property** 74](#_Toc32697347)

[**CapPhoto Property** 74](#_Toc32697348)

[**CapPhotoColorSpace Property** 75](#_Toc32697349)

[**CapPhotoFrameRate Property** 75](#_Toc32697350)

[**CapPhotoResolution Property** 75](#_Toc32697351)

[**CapPhotoType Property** 75](#_Toc32697352)

[**CapSaturation Property** 75](#_Toc32697353)

[**CapStorageTarget Property** 76](#_Toc32697354)

[**CapVerticalFlip Property** 76](#_Toc32697355)

[**CapVideo Property** 76](#_Toc32697356)

[**CapVideoColorSpace Property** 77](#_Toc32697357)

[**CapVideoFrameRate Property** 77](#_Toc32697358)

[**CapVideoResolution Property** 77](#_Toc32697359)

[**CapVideoType Property** 77](#_Toc32697360)

[**Contrast Property** 78](#_Toc32697361)

[**Exposure Property** 78](#_Toc32697362)

[**Gain Property** 78](#_Toc32697363)

[**HorizontalFlip Property** 79](#_Toc32697364)

[**Hue Property** 79](#_Toc32697365)

[**PhotoColorSpace Property** 79](#_Toc32697366)

[**PhotoColorSpaceList Property** 80](#_Toc32697367)

[**PhotoFrameRate Property** 80](#_Toc32697368)

[**PhotoMaxFrameRate Property** 80](#_Toc32697369)

[**PhotoResolution Property** 81](#_Toc32697370)

[**PhotoResolutionList Property** 81](#_Toc32697371)

[**PhotoType Property** 81](#_Toc32697372)

[**PhotoTypeList Property** 82](#_Toc32697373)

[**RemainingRecordingTimeInSec Property** 82](#_Toc32697374)

[**Saturation Property** 82](#_Toc32697375)

[**StorageTarget Property** 83](#_Toc32697376)

[**VerticalFlip Property** 83](#_Toc32697377)

[**VideoCaptureMode Property** 84](#_Toc32697378)

[**VideoColorSpace Property** 85](#_Toc32697379)

[**VideoColorSpaceList Property** 85](#_Toc32697380)

[**VideoFrameRate Property** 86](#_Toc32697381)

[**VideoMaxFrameRate Property** 86](#_Toc32697382)

[**VideoResolution Property** 86](#_Toc32697383)

[**VideoResolutionList Property** 87](#_Toc32697384)

[**VideoType Property** 87](#_Toc32697385)

[**VideoTypeList Property** 87](#_Toc32697386)

[Note: Video Capture Device Property Value Relationship 88](#_Toc32697387)

[Methods (UML operations) 89](#_Toc32697388)

[**startVideo Method** 89](#_Toc32697389)

[**stopVideo Method** 90](#_Toc32697390)

[**takePhoto Method** 90](#_Toc32697391)

[Events (UML interfaces) 91](#_Toc32697392)

[**DirectIOEvent** 91](#_Toc32697393)

[**ErrorEvent** 92](#_Toc32697394)

[**StatusUpdateEvent** 93](#_Toc32697395)

[Chapter 40 94](#_Toc32697396)

[Individual Recognition 94](#_Toc32697397)

[Summary 94](#_Toc32697398)

[General Information 97](#_Toc32697399)

[**Capabilities** 97](#_Toc32697400)

[**Individual Recognition Class Diagram** 97](#_Toc32697401)

[Model 98](#_Toc32697402)

[**Input Model** 98](#_Toc32697403)

[**Device Sharing** 98](#_Toc32697404)

[**IndividualRecognitionFilter** 99](#_Toc32697405)

[**IndividualRecognitionFilter Property Example Format** 100](#_Toc32697406)

[ Basic Items 100](#_Toc32697407)

[ Face Recognition device example 101](#_Toc32697408)

[**IndividualRecognition Information Property Example Format** 103](#_Toc32697409)

[ Basic Items 103](#_Toc32697410)

[ Face Recognition Device Example 104](#_Toc32697411)

[Properties (UML attributes) 105](#_Toc32697412)

[**CapIndividualList Property** 105](#_Toc32697413)

[**IndividualIDs Property** 105](#_Toc32697414)

[**IndividualRecognitionFilter Property** 105](#_Toc32697415)

[**IndividualRecognitionInformation Property** 106](#_Toc32697416)

[Events (UML interfaces) 107](#_Toc32697417)

[**DataEvent** 107](#_Toc32697418)

[**DirectIOEvent** 107](#_Toc32697419)

[**ErrorEvent** 108](#_Toc32697420)

[**StatusUpdateEvent** 109](#_Toc32697421)

[Chapter 41 110](#_Toc32697422)

[Sound Recorder 110](#_Toc32697423)

[Summary 110](#_Toc32697424)

[General Information 114](#_Toc32697425)

[**Capabilities** 114](#_Toc32697426)

[**Sound Recorder Class Diagram** 114](#_Toc32697427)

[Model 115](#_Toc32697428)

[**Device Sharing** 116](#_Toc32697429)

[Properties(UML attributes) 117](#_Toc32697430)

[**CapAssociatedHardTotalsDevice Property** 117](#_Toc32697431)

[**CapChannel Property** 117](#_Toc32697432)

[**CapSamplingRate Property** 117](#_Toc32697433)

[**CapSoundType Property** 117](#_Toc32697434)

[**CapStorageTarget Property** 118](#_Toc32697435)

[**CapRecordingLevel Property** 118](#_Toc32697436)

[**Channel Property** 118](#_Toc32697437)

[**ChannelList Property** 119](#_Toc32697438)

[**RecordingLevel Property** 119](#_Toc32697439)

[**RemainingRecordingTimeInSec Property** 119](#_Toc32697440)

[**SamplingRate Property** 119](#_Toc32697441)

[**SamplingRateList Property** 120](#_Toc32697442)

[**SoundData Property** 120](#_Toc32697443)

[**SoundType Property** 120](#_Toc32697444)

[**SoundTypeList Property** 120](#_Toc32697445)

[**StorageTarget Property** 121](#_Toc32697446)

[Methods(UML operations) 122](#_Toc32697447)

[**startRecording Method** 122](#_Toc32697448)

[**stopRecording Method** 123](#_Toc32697449)

[Events(UML interfaces) 124](#_Toc32697450)

[**DataEvent** 124](#_Toc32697451)

[**DirectIOEvent** 124](#_Toc32697452)

[**ErrorEvent** 125](#_Toc32697453)

[**StatusUpdateEvent** 126](#_Toc32697454)

[Chapter 42 127](#_Toc32697455)

[Voice Recognition 127](#_Toc32697456)

[Summary 127](#_Toc32697457)

[General Information 131](#_Toc32697458)

[**Capabilities** 131](#_Toc32697459)

[**Voice Recognition Class Diagram** 131](#_Toc32697460)

[Model 132](#_Toc32697461)

[**Device Sharing** 133](#_Toc32697462)

[Properties (UML attributes) 134](#_Toc32697463)

[**CapLanguage Property** 134](#_Toc32697464)

[**HearingDataPattern Property** 134](#_Toc32697465)

[**HearingDataWord Property** 134](#_Toc32697466)

[**HearingDataWordList Property** 135](#_Toc32697467)

[**HearingResult Property** 136](#_Toc32697468)

[**HearingStatus Property** 137](#_Toc32697469)

[**LanguageList Property** 137](#_Toc32697470)

[Methods (UML operations) 138](#_Toc32697471)

[**startHearingFree Method** 138](#_Toc32697472)

[**startHearingSentence Method** 139](#_Toc32697473)

[**startHearingWord Method** 140](#_Toc32697474)

[**startHearingYesNo Method** 141](#_Toc32697475)

[**stopHearing Method** 141](#_Toc32697476)

[Events (UML interfaces) 142](#_Toc32697477)

[**DataEvent** 142](#_Toc32697478)

[**DirectIOEvent** 142](#_Toc32697479)

[**ErrorEvent** 143](#_Toc32697480)

[**StatusUpdateEvent** 144](#_Toc32697481)

[Chapter 43 145](#_Toc32697482)

[Sound Player 145](#_Toc32697483)

[Summary 145](#_Toc32697484)

[General Information 148](#_Toc32697485)

[**Capabilities** 148](#_Toc32697486)

[**Sound Player Class Diagram** 148](#_Toc32697487)

[Model 149](#_Toc32697488)

[**Device Sharing** 150](#_Toc32697489)

[Properties(UML attributes) 151](#_Toc32697490)

[**CapAssociatedHardTotalsDevice Property** 151](#_Toc32697491)

[**CapMultiPlay Property** 151](#_Toc32697492)

[**CapSoundTypeList Property** 151](#_Toc32697493)

[**CapStorageTarget Property** 151](#_Toc32697494)

[**CapVolume Property** 152](#_Toc32697495)

[**DeviceSoundList Property** 153](#_Toc32697496)

[**OutputIDList Property** 153](#_Toc32697497)

[**StorageTarget Property** 153](#_Toc32697498)

[**Volume Property** 153](#_Toc32697499)

[Methods (UML operations) 154](#_Toc32697500)

[**playSound Method** 154](#_Toc32697501)

[**stopSound Method** 154](#_Toc32697502)

[Events (UML interfaces) 155](#_Toc32697503)

[**DirectIOEvent** 155](#_Toc32697504)

[**ErrorEvent** 156](#_Toc32697505)

[**OutputCompleteEvent** 158](#_Toc32697506)

[**StatusUpdateEvent** 158](#_Toc32697507)

[Chapter 44 159](#_Toc32697508)

[Speech Synthesis 159](#_Toc32697509)

[Summary 159](#_Toc32697510)

[General Information 163](#_Toc32697511)

[**Capabilities** 163](#_Toc32697512)

[**Speech Synthesis Class Diagram** 163](#_Toc32697513)

[Model 164](#_Toc32697514)

[**Device Sharing** 165](#_Toc32697515)

[Properties (UML attributes) 166](#_Toc32697516)

[**CapLanguage Property** 166](#_Toc32697517)

[**CapPitch Property** 166](#_Toc32697518)

[**CapSpeed Property** 166](#_Toc32697519)

[**CapVoice Property** 166](#_Toc32697520)

[**CapVolume Property** 167](#_Toc32697521)

[**Language Property** 167](#_Toc32697522)

[**LanguageList Property** 167](#_Toc32697523)

[**OutputIDList Property** 167](#_Toc32697524)

[**Pitch Property** 168](#_Toc32697525)

[**Speed Property** 168](#_Toc32697526)

[**Voice Property** 168](#_Toc32697527)

[**VoiceList Property** 169](#_Toc32697528)

[**Volume Property** 169](#_Toc32697529)

[Methods (UML operations) 170](#_Toc32697530)

[**speak Method** 170](#_Toc32697531)

[**speakImmediate Method** 171](#_Toc32697532)

[**stopCurrentSpeaking Method** 172](#_Toc32697533)

[**stopSpeaking Method** 172](#_Toc32697534)

[Events (UML interfaces) 173](#_Toc32697535)

[**DirectIOEvent** 173](#_Toc32697536)

[**ErrorEvent** 174](#_Toc32697537)

[**OutputCompleteEvent** 176](#_Toc32697538)

[**StatusUpdateEvent** 176](#_Toc32697539)

[Chapter 45 177](#_Toc32697540)

[Gesture Control 177](#_Toc32697541)

[Summary 177](#_Toc32697542)

[General Information 181](#_Toc32697543)

[**Capabilities** 181](#_Toc32697544)

[**Gesture Control Class Diagram** 181](#_Toc32697545)

[Model 182](#_Toc32697546)

[**Automatic control** 182](#_Toc32697547)

[**Pose / Motion** 183](#_Toc32697548)

[**Device Sharing** 183](#_Toc32697549)

[Properties (UML attributes) 184](#_Toc32697550)

[**AutoMode Property** 184](#_Toc32697551)

[**AutoModeList Property** 184](#_Toc32697552)

[**CapAssociatedHardTotalsDevice Property** 184](#_Toc32697553)

[**CapMotion Property** 185](#_Toc32697554)

[**CapMotionCreation Property** 185](#_Toc32697555)

[**CapPose Property** 185](#_Toc32697556)

[**CapPoseCreation Property** 185](#_Toc32697557)

[**CapStorageTarget Property** 186](#_Toc32697558)

[**MotionList Property** 186](#_Toc32697559)

[**PoseCreationMode Property** 186](#_Toc32697560)

[**PoseList Property** 187](#_Toc32697561)

[**StorageTarget Property** 187](#_Toc32697562)

[**JointList Property** 188](#_Toc32697563)

[**Table of Gesture Control Device Listed Items in Property** 188](#_Toc32697564)

[Methods (UML operations) 189](#_Toc32697565)

[**createMotion Method** 189](#_Toc32697566)

[**createPose Method** 189](#_Toc32697567)

[**getPosition Method** 190](#_Toc32697568)

[**setPosition Method** 191](#_Toc32697569)

[**setSpeed Method** 192](#_Toc32697570)

[**startMotion Method** 193](#_Toc32697571)

[**startPose Method** 193](#_Toc32697572)

[**stopControl Method** 194](#_Toc32697573)

[Events (UML interfaces) 195](#_Toc32697574)

[**DirectIOEvent** 195](#_Toc32697575)

[**ErrorEvent** 196](#_Toc32697576)

[**OutputCompleteEvent** 197](#_Toc32697577)

[**StatusUpdateEvent** 198](#_Toc32697578)

[Chapter 46 199](#_Toc32697579)

[Device Monitor 199](#_Toc32697580)

[**Summary** 199](#_Toc32697581)

[General Information 202](#_Toc32697582)

[**Capabilities** 202](#_Toc32697583)

[**Device Monitor Class Diagram** 202](#_Toc32697584)

[Model 203](#_Toc32697585)

[**Device Sharing** 203](#_Toc32697586)

[Properties (UML attributes) 204](#_Toc32697587)

[**DeviceData Property** 204](#_Toc32697588)

[**DeviceList Property** 205](#_Toc32697589)

[**MonitoringDeviceList Property** 206](#_Toc32697590)

[Methods (UML operations) 207](#_Toc32697591)

[**addMonitoringDevice Method** 207](#_Toc32697592)

[**deleteMonitoringDevice Method** 209](#_Toc32697593)

[**clearMonitoringDevices Method** 209](#_Toc32697594)

[**getDeviceValue method** 209](#_Toc32697595)

[Events (UML interfaces) 210](#_Toc32697596)

[**DataEvent** 210](#_Toc32697597)

[**DirectIOEvent** 210](#_Toc32697598)

[**ErrorEvent** 211](#_Toc32697599)

[**StatusUpdateEvent** 213](#_Toc32697600)

[Chapter 47 214](#_Toc32697601)

[Graphic Display 214](#_Toc32697602)

[Summary 214](#_Toc32697603)

[General Information 218](#_Toc32697604)

[**Capabilities** 218](#_Toc32697605)

[**Graphics Display Class Diagram** 218](#_Toc32697606)

[Model 219](#_Toc32697607)

[Image Display Mode 219](#_Toc32697608)

[Video Display Mode 219](#_Toc32697609)

[Web Display Mode 220](#_Toc32697610)

[**Device Sharing** 221](#_Toc32697611)

[Properties (UML attributes) 222](#_Toc32697612)

[**Brightness Property** 222](#_Toc32697613)

[**CapAssociatedHardTotalsDevice Property** 222](#_Toc32697614)

[**CapBrightness Property** 222](#_Toc32697615)

[**CapImageType Property** 222](#_Toc32697616)

[**CapStorageTarget Property** 223](#_Toc32697617)

[**CapURLBack Property** 223](#_Toc32697618)

[**CapURLForward Property** 223](#_Toc32697619)

[**CapVideoType Property** 224](#_Toc32697620)

[**CapVolume Property** 224](#_Toc32697621)

[**DisplayMode Property** 225](#_Toc32697622)

[**ImageType Property** 226](#_Toc32697623)

[**ImageTypeList Property** 226](#_Toc32697624)

[**LoadStatus Property** 226](#_Toc32697625)

[**StorageTarget Property** 227](#_Toc32697626)

[**URL Property** 227](#_Toc32697627)

[**VideoType Property** 228](#_Toc32697628)

[**VideoTypeList Property** 228](#_Toc32697629)

[**Volume Property** 228](#_Toc32697630)

[Methods (UML operations) 229](#_Toc32697631)

[**cancelURLLoading Method** 229](#_Toc32697632)

[**goURLBack Method** 229](#_Toc32697633)

[**goURLForward Method** 229](#_Toc32697634)

[**loadImage Method** 230](#_Toc32697635)

[**loadURL Method** 230](#_Toc32697636)

[**playVideo Method** 231](#_Toc32697637)

[**stopVideo Method** 231](#_Toc32697638)

[**updateURLPage Method** 232](#_Toc32697639)

[Events (UML interfaces) 233](#_Toc32697640)

[**DirectIOEvent** 233](#_Toc32697641)

[**ErrorEvent** 233](#_Toc32697642)

[**OutputCompleteEvent** 235](#_Toc32697643)

[**StatusUpdateEvent** 235](#_Toc32697644)

[Appendix K 236](#_Toc32697645)

[Relationship to other OMG specification and activities 236](#_Toc32697646)

[**Robotics Domain Task Force** 236](#_Toc32697647)

[Activities in Robotics Domain Task Force 236](#_Toc32697648)

[RoIS Specification 236](#_Toc32697649)

[Scope of RoIS specification 236](#_Toc32697650)

[Robot Service Ontology [RoSO] RFP 237](#_Toc32697651)

[**Interoperability between UPOS RCSD and RoIs** 238](#_Toc32697652)

[Rleationsihp between UPOS RCSD and RoIS 238](#_Toc32697653)

[Document History 240](#_Toc32697654)

[**Version History** 240](#_Toc32697655)

[Glossary 241](#_Toc32697656)

# Preface

**OMG**

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG’s specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG’s specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.　More information on the OMG is available at http://www.omg.org/.

**OMG Specifications**

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:

*http://www.omg.org/spec*

Specifications are organized by the following categories:

Business Modeling Specifications

Middleware Specifications

1. CORBA/IIOP
2. Data Distribution Services
3. Specialized CORBA

IDL/Language Mapping Specifications

Modeling and Metadata Specifications

1. UML, MOF, CWM, XMI
2. UML Profile

Modernization Specifications

Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

1. CORBAServices
2. CORBAFacilities

OMG Domain Specifications

CORBA Embedded Intelligence Specifications

CORBA Security Specifications

All of OMG’s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters  
109 Highland Avenue  
Needham, MA 02494  
USA  
Tel: +1-781-444-0404  
Fax: +1-781-444-0320  
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

**Typographical Conventions**

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

**Issues**

The reader is encouraged to report any technical or editing issues/problems with this specification to <http://www.omg.org/report_issue.htm>.

# UPOS 1.16 RCSD Specification Overview

## **Updated Items in Release 1.16**

Chapter sections 23 and 38 from UPOS1.15 are included with annotations denoting the changes necessary for supporting the addition of the Retail Communications Service Devices. Chapters 39-47 are new chapters for devices being added to UPOS v1. The following is a list of the properties, methods and chapters.

## **Updated Items in CHAPTER 21 Lights**

**Properties**

CapPattern **Property**

**Methods**

switchOn **Method**

switchOnMultipl**e Method**

switchOnPattern **Method**

switchOffPattern **Method**

## **Updated Items in CHAPTER 29 POS Power**

**Properties**

CapBatteryCapacityRemainingInSeconds **Property**

CapVariableBatteryCriticallyLowThresholdInSeconds **Property**

CapVariableBatteryLowThresholdInSeconds **Property**

CapChargeTime **Property**

BatteryCapacityRemainingInSeconds **Property**

BatteryCriticallyLowThresholdInSeconds **Property**

BatteryLowThresholdInSeconds **Property**

ChargeTime **Property**

## **Added Chapters in Release 1.16**

CHAPTER 39 Video Capture

CHAPTER 40 Individual Recognition

CHAPTER 41 Sound Recorder

CHAPTER 42 Voice Recognition

CHAPTER 43 Sound Player

CHAPTER 44 Speech Synthesis

CHAPTER 45 Gesture Control

CHAPTER 46 Device Monitor

CHAPTER 47 Graphic Display

# Chapter 21

# Lights

This Chapter defines the Lights device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.12 | Not Supported |
| **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 | Not Supported |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.12 | Not Supported |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.12 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.12 | open |
| **OutputID:** | *int32* | {read-only} | 1.12 | Not Supported |
| **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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **CapAlarm:** | *int32* | {read-only} | 1.12 | open |
| **CapBlink:** | *boolean* | {read-only} | 1.12 | open |
| **CapColor:** | *int32* | {read-only} | 1.12 | open |
| **CapPattern:** | *int32* | {read-only} | 1.16 | open |
| **MaxLights:** | *int32* | {read-only} | 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 { }** | ***Not supported*** |
| **clearInputProperties ( ): void { }** | ***Not supported*** |
| **clearOutput ( ): void { }** | ***Not supported*** |
| **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, enable}** | 1.12 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | 1.12 |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | 1.12 |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, enable}** | 1.12 |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, enable}** | 1.12 |
| ***Specific*** |  |
| ***Name*** |  |
| **switchOff (lightNumber*:*** *int32***):  void {raises-exception, use after open, claim, enable}** | 1.12 |
| **switchOn (lightNumber:** *int32*,**blinkOnCycle:** *int32*, **blinkOffCycle:** *int32*,**color:** *int32*, **alarm:** *int32***):  void {raises-exception, use after open, claim, enable}** | 1.12 |
| **switchOnMultiple (lightNumbers: string, blinkOnCycle: int32,**  **blinkOffCycle: int32, color: int32, alarm: int32):**  **void {raises-exception, use after open, claim, enable}** | 1.16 |
| **switchOnPattern (pattern: int32, alarm: int32):**  **void {raises-exception, use after open, claim, enable}** | 1.16 |
| **switchOffPattern ( ):**  **void {raises-exception, use after open, claim, enable}** | 1.16 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
|  |  |  |  |
| **upos::events::DataEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.12 |
| **EventNumber:** | *int32* | {read-only} |  |
| **Data:** | *int32* | {read-write} |  |
| **Obj:** | *object* | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.12 |
| **Status:** | *int32* | {read-only} |  |

# General Information

The Lights programmatic name is “Lights”.

This device category was added to Version 1.12 of the specification.

## **Capabilities**

* The Lights device control has the following capability:
* Supports commands to “switch on” and “switch off” a light.
* The Lights device control may have the following additional capabilities:
  + Supports device-level blinking at adjustable blink cycles.
  + Support multiple lights.
  + Supports different colors of a light.
  + Supports different alarms

## **Device Sharing**

Lights is an exclusive-use device. Its device sharing rules are:

* The application must claim the device before enabling it.
* The application must claim and enable the device before accessing some of the properties and methods, or receiving events.
* See the “Summary” table for precise usage prerequisites.

## **Lights Class Diagram**

***Updated in Release 1.16***

The following diagram shows the relationships between the Lights classes

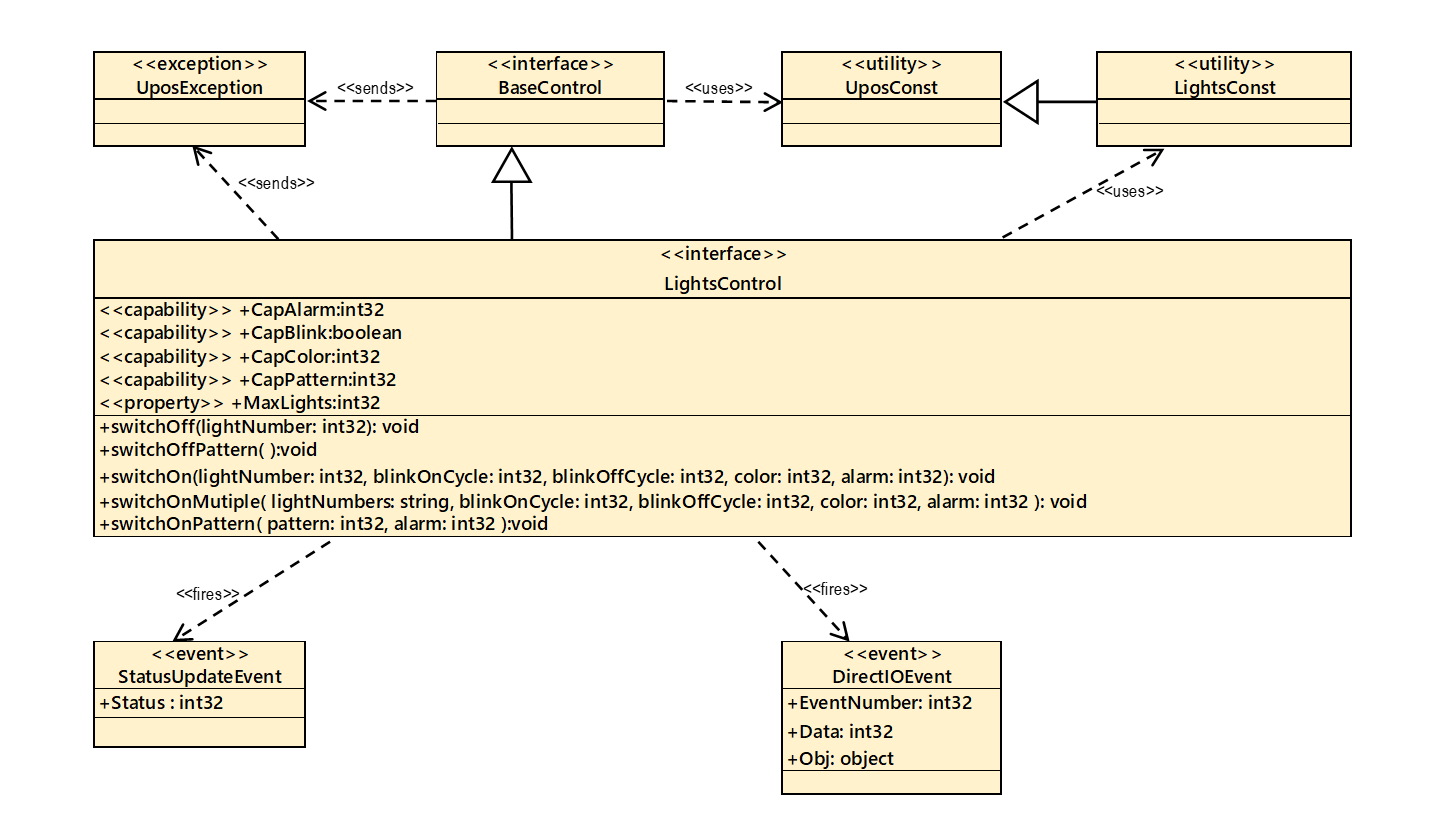


Fig. Chap. 21-1 Lights Class Diagram

## **Lights Sequence Diagram**

The following sequence diagram show the typical usage of the Lights device illustrating the handling of the media entry indicator lights.



Fig. Chap. 21-2 Lights Sequence Diagram (handling of the media entry indicator lights)

The following sequence diagram show the typical usage of the Lights device illustrating the handling of the pole lights.



Fig. Chap. 21-3 Lights Sequence Diagram (handling of the pole lights)

# Properties (UML attributes)

## **CapAlarm Property**

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

**Remarks** This capability indicates if the device supports different alarms.

**CapAlarm** is a logical OR combination of any of the following values:

**Value Meaning**

LGT\_ALARM\_NOALARM Alarms are not supported.

LGT\_ALARM\_SLOW Supports a slow beep.

LGT\_ALARM\_MEDIUM Supports a medium beep.

LGT\_ALARM\_FAST Supports a fast beep.

LGT\_ALARM\_CUSTOM1 Supports 1st custom alarm.

LGT\_ALARM\_CUSTOM2 Supports 2nd custom alarm.

This property is initialized by the **open** method. If the device does not support alarms, it is initialized to LGT\_ALARM\_NOALARM.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

## **CapBlink Property**

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

**Remarks** If true, a blinking capability is supported. It may be either a physical capability of the device or emulated by the 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 Intro-20.

## **CapColor Property**

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

**Remarks** This capability indicates if the device supports different colors.

**CapColor** is a logical OR combination of any of the following values:

**Value Meaning**

LGT\_COLOR\_PRIMARY Supports Primary Color (Usually Green).

LGT\_COLOR\_CUSTOM1 Supports 1st Custom Color (Usually Red).

LGT\_COLOR\_CUSTOM2 Supports 2nd Custom Color (Usually Yellow).

LGT\_COLOR\_CUSTOM3 Supports 3rd Custom Color.

LGT\_COLOR\_CUSTOM4 Supports 4th Custom Color.

LGT\_COLOR\_CUSTOM5 Supports 5th Custom Color.

This property is initialized by the **open** method. If the device supports only one color, it is initialized to LGT\_COLOR\_PRIMARY.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

## **CapPattern Property *Added in Release 1.16***

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

**Remarks** This capability indicates if the device supports different lighting patterns.

**CapPattern** is a logical OR combination of any of the following values:

**Value Meaning**

LGT\_PATTERN\_NOPATTERN Lighting patterns are not supported.

LGT\_PATTERN\_CUSTOM 1~32 Supports 1st to 32th Lighting Pattern.

This property is initialized by the **open** method. If the device does not support lighting pattern, it is initialized to LGT\_PATTERN\_NOPATTERN.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also switchOnPattern** Method.

## **MaxLights Property**

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

**Remarks** **MaxLights** specifies the maximum number of lights that the device can support.

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 Intro-20.

# Methods (UML operations)

## **switchOff Method**

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

**Parameter Description**

*lightNumber* Specifies the light number. Valid light numbers are 1 through **MaxLights**.

**Remarks** Switches off the light specified by *lightNumber*.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20**.**

A possible value of the exception’s *ErrorCode* property is:

**Value Meaning**

E\_ILLEGAL The *lightNumber* parameter exceeds **MaxLights**.

**See Also MaxLights** Property**.**

## **switchOffPattern Method**

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

**Remarks** Switches off the pattern lighting.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20**.**

A possible value of the exception’s *ErrorCode* property is:

**Value Meaning**

E\_ILLEGAL Pattern lighting is not executed.

**See Also switchOnPattern** Method.

## **switchOn Method *Updated in Release 1.16***

**Syntax switchOn (lightNumber: *int32,* blinkOnCycle: *int32,* blinkOffCycle: *int32,* color: *int32,* alarm: *int32*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*lightNumber* Specifies the light number. Valid light numbers are 1 through **MaxLights**.

*blinkOnCycle* A zero (0) value indicates no blink cycle. A positive value indicates the blink on cycle time in milliseconds. Negative values are not allowed.

*blinkOffCycle* A zero (0) value indicates no blink cycle. A positive value indicates the blink off cycle time in milliseconds. Negative values are not allowed.

*color* Specifies the color of the light, must be one of the colors defined by **CapColor**.

*alarm* Specifies the used alarm type, must be one of the alarms defined by **CapAlarm**.

**Remarks** Switches on the light specified by *lightNumber* or let it blink.

If *blinkOnCycle* and *blinkOffCycle* are zero (0) or **CapBlink** is false, then the parameters *blinkOnCycle* and *blinkOffCycle* will be ignored and the light will only be switched on.

If **CapBlink** is true and *blinkOnCycle* and *blinkOffCycle* are positive, then the light will blink.

If **CapColor** is LGT\_COLOR\_PRIMARY the light does not support different colors and *color* is ignored, otherwise **switchOn** will use the color specified by *color*.

If **CapAlarm** is LGT\_ALARM\_NOALARM the light does not support different alarms and *alarm* is ignored, otherwise **switchOn** will use the alarm specified by *alarm*.

Subsequent calls to **switchOn** will change the blink cycles, the color or the alarm type of the light.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20**.**

A possible value of the exception’s *ErrorCode* property is:

**Value Meaning**

E\_ILLEGAL The *lightNumber* parameter exceeds **MaxLights**, an invalid *color* or *alarm* was specified.

**See Also** **CapAlarm** Property, **CapBlink** Property, **CapColor** Property, **MaxLights** Property**.**

## **switchOnMultiple Method** ***Added in Release 1.16***

**Syntax switchOnMultiple (lightNumbers: *string,* blinkOnCycle: *int32,* blinkOffCycle: *int32,* color: *int32,* alarm: *int32*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*lightNumbers* Specifies the comma-delimited list of light number. Valid light numbers are 1 through **MaxLights**.

*blinkOnCycle* A zero (0) value indicates no blink cycle. A positive value indicates the blink on cycle time in milliseconds. Negative values are not allowed.

*blinkOffCycle* A zero (0) value indicates no blink cycle. A positive value indicates the blink off cycle time in milliseconds. Negative values are not allowed.

*color* Specifies the color of the light, must be one of the colors defined by **CapColor**.

*alarm* Specifies the used alarm type, must be one of the alarms defined by **CapAlarm**.

**Remarks** This method does the same as swicthOn but in a synchronized way such that all lights are switched on / blinking synchronously. Switches on the multiple lights specified by *lightNumbers* or let it blink.

If *blinkOnCycle* and *blinkOffCycle* are zero (0) or **CapBlink** is false, then the parameters *blinkOnCycle* and *blinkOffCycle* will be ignored and the light will only be switched on.

If **CapBlink** is true and *blinkOnCycle* and *blinkOffCycle* are positive, then the light will blink.

If **CapColor** is LGT\_COLOR\_PRIMARY the light does not support different colors and *color* is ignored, otherwise **switchOnMultiple** will use the color specified by *color*.

If **CapAlarm** is LGT\_ALARM\_NOALARM the light does not support different alarms and *alarm* is ignored, otherwise **switchOnMultiple** will use the alarm specified by *alarm*.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20**.**

A possible value of the exception’s *ErrorCode* property is:

**Value Meaning**

E\_ILLEGAL The *lightNumbers* parameter exceeds **MaxLights**, an invalid value was specified.

**See Also CapAlarm** Property, **CapBlink** Property, **CapColor** Property, **MaxLights** Property**.**

## **switchOnPattern Method** ***Added in Release 1.16***

**Syntax switchOnPattern (pattern: *int32,* alarm: *int32*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*pattern* Specifies the lighting pattern, must be one of the patterns defined by **CapPattern**.

*alarm* Specifies the used alarm type, must be one of the alarms defined by **CapAlarm**.

**Remarks** Switches on the light specified by *pattern*.

If **CapAlarm** is LGT\_ALARM\_NOALARM the light does not support different alarms and *alarm* is ignored, otherwise **switchOn** and **swithOnPattern** will use the alarm specified by *alarm*.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20**.**

A possible value of the exception’s *ErrorCode* property is:

**Value Meaning**

E\_ILLEGAL An invalid value was specified, or unsupported operation with the Device.

**See Also CapAlarm** Property, **CapPattern** Property**.**

# Events (UML interfaces)

## **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 Lights Service to provide events to the application that are not otherwise supported by the device 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 Lights devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** Method.

## **StatusUpdateEvent**

**<< event >> upos::events::StatusUpdateEvent  
Status : *int32* {read-only}**

**Description** Notifies the application that there is a change in the power status of a light.

**Attributes** This event contains the following attribute:

**Attribute Type Description**

*Status int32* Reports a change in the power status of a light.

***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 “**StatusUpdateEvent**” description on page 1-34.

**Remarks** Enqueued when the light detects a power state change.

**See Also** **“Events"** on page Intro-19.

# Chapter 29

# POS Power

This Chapter defines the POS Power device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.5 | Not Supported |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.9 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.3 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.8 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.9 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.8 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.5 | open |
| **Claimed:** | *boolean* | {read-only | 1.5 | open |
| **DataCount:** | *int32* | {read-only} | 1.5 | Not Supported |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.5 | Not Supported |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.5 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.5 | open |
| **OutputID:** | *int32* | {read-only} | 1.5 | Not Supported |
| **PowerNotify:** | *int32* | {read-write} | 1.5 | open |
| **PowerState:** | *int32* | {read-only} | 1.5 | open |
| **State:** | *int32* | {read-only} | 1.5 | -- |
| **DeviceControlDescription:** | *string* | {read-only} | 1.5 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.5 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.5 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.5 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.5 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.5 | open |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **BatteryCapacityRemaining:** | *int32* | {read-only} | 1.9 | open |
| **BatteryCapacityRemainingInSeconds:** | *int32* | {read-only} | 1.16 | open |
| **BatteryCriticallyLowThreshold:** | *int32* | {read-write} | 1.9 | open |
| **BatteryCriticallyLowThresholdInSeconds:** | *int32* | {read-write} | 1.16 | open |
| **BatteryLowThreshold:** | *int32* | {read-write} | 1.9 | open |
| **BatteryLowThresholdInSeconds:** | *int32* | {read-write} | 1.16 | open |
| **CapBatteryCapacityRemaining:** | *boolean* | {read-only} | 1.9 | open |
| **CapBatteryCapacityRemainingInSeconds:** | *boolean* | {read-only} | 1.16 | open |
| **CapChargeTime:** | *boolean* | {read-only} | 1.16 | open |
| **CapFanAlarm:** | *boolean* | {read-only} | 1.5 | open |
| **CapHeatAlarm:** | *boolean* | {read-only} | 1.5 | open |
| **CapQuickCharge:** | *boolean* | {read-only} | 1.5 | open |
| **CapRestartPOS:** | *boolean* | {read-only} | 1.9 | open |
| **CapShutdownPOS**: | *boolean* | {read-only} | 1.5 | open |
| **CapStandbyPOS:** | *boolean* | {read-only} | 1.9 | open |
| **CapSuspendPOS:** | *boolean* | {read-only} | 1.9 | open |
| **CapUPSChargeState:** | *int32* | {read-only} | 1.5 | open |
| **CapVariableBatteryCriticallyLowThreshold:** | *boolean* | {read-only} | 1.9 | open |
| **CapVariableBatteryCriticallyLowThresholdInSeconds:** | *boolean* | {read-only} | 1.16 | open |
| **CapVariableBatteryLowThreshold:** | *boolean* | {read-only} | 1.9 | open |
| **CapVariableBatteryLowThresholdInSeconds:** | *boolean* | {read-only} | 1.16 | open |
| **ChargeTime:** | *int32* | {read-only} | 1.16 | open |
| **EnforcedShutdownDelayTime:** | *int32* | {read-write} | 1.5 | open |
| **PowerFailDelayTime:** | *int32* | {read-only} | 1.5 | open |
| **PowerSource:** | *int32* | {read-only} | 1.9 | open |
| **QuickChargeMode:** | *boolean* | {read-only} | 1.5 | open |
| **QuickChargeTime:** | *int32* | {read-only} | 1.5 | open |
| **UPSChargeState:** | *int32* | {read-only} | 1.5 | open, claim & enable |

|  |  |
| --- | --- |
| **Methods (UML operations)** |  |
| ***Common*** |  |
| ***Name*** | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | 1.5 |
| **close ( ): void {raises-exception, use after open}** | 1.5 |
| **claim (timeout:** *int32* **): void {raises-exception, use after open}** | 1.5 |
| **release ( ): void {raises-exception, use after open, claim}** | 1.5 |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | 1.5 |
| **clearInput ( ): void { }** | ***Not supported*** |
| **clearInputProperties ( ): void { }** | ***Not supported*** |
| **clearOutput ( ): void { }** | ***Not supported*** |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | 1.5 |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, claim, enable}** | 1.9 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, claim, enable}** | 1.8 |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, claim, enable}** | 1.8 |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, claim, enable}** | 1.9 |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, claim, enable}** | 1.8 |
| ***Specific*** |  |
| ***Name*** |  |
| **restartPOS ( ):  void {raises-exception, use after open, enable}** | 1.9 |
| **shutdownPOS ( ):  void {raises-exception, use after open, enable}** | 1.5 |
| **standbyPOS (reason: int32 ):  void {raises-exception, use after open, enable}** | 1.9 |
| **suspendPOS (reason: int32 ):  void {raises-exception, use after open, enable}** | 1.9 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
|  |  |  |  |
| **upos::events::DataEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.5 |
| **EventNumber:** | ***int32*** | {read-only} |  |
| **Data:** | ***int32*** | {read-write} |  |
| **Obj:** | ***object*** | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.5 |
| **Status:** | ***int32*** | {read-only} |  |
|  |  |  |  |

# General Information

The POS Power programmatic name is “POSPower”.

## **Capabilities**

The POSPower device class has the following capabilities:

* Supports a command to “shut down” the system.
* Supports a command to restart the system.
* Supports a command to “suspend” the system.
* Supports a command to have the system go to standby.
* Supports accessing a power handling mechanism of the underlying operating system and hardware.
* Informs the application if a power fail situation has occurred.
* Informs the application about battery level.
* Informs the application if the UPS charge state has changed.
* Informs the application about high CPU temperature.
* Informs the application about stopped CPU fan.
* Informs the application if an operating system dependent enforced shutdown mechanism is processed.
* Allows the application after saving application data locally or transferring application data to a server to shut down the POS terminal.
* Informs the application about an initiated shutdown.

## **Device Sharing**

The POSPower is a sharable device. Its device sharing rules are:

* After opening and enabling the device, the application may access all properties and methods and will receive status update events.
* If more than one application has opened and enabled the device, all applications may access its properties and methods. Status update events are fired to all of the applications.
* If one application claims the POSPower, then only that application may call the **shutdownPOS, standbyPOS, or suspendPOS** methods. This feature provides a degree of security, such that these methods may effectively be restricted to the main POS application if that application claims the device at startup.
* See the “Summary” table for precise usage prerequisites.

# Model

The general model of POSPower is based on the power model of each device in version 1.3 or later. The same common properties are used but all states relate to the POS terminal itself and not to a peripheral device.

There are three states of the POSPower:

* ONLINE. The POS terminal is powered on and ready for use. This is the “operational” state.
* OFF. The POS terminal is powered off or detached from the power supplying net. The POS terminal runs on battery power support. This is the powerfail situation.
* OFFLINE. The POS terminal is powered on but is running in a “lower-power-consumption” mode. It may need to be placed online by pressing a button or key or something else which may wake up the system.

Power reporting only occurs while the device is open, enabled and power notification is switched on.

In a powerfail situation - that means the POSPower is in the state OFF - the POS terminal will be shut down automatically after the last application has closed the POSPower device or the time specified by the **EnforcedShutdownDelayTime** property has been elapsed.

A call to the **shutdownPOS** method will always shut down the POS terminal independent of the system power state.

**Version 1.9 or later**

Support of battery powered devices is added. In addition to adding properties to report battery levels and power sources, properties are added to allow for the setting of low and critically low battery levels. The POSPower device also includes the ability to request or respond to request to enter the standby and suspend states. The model does not attempt to duplicate other power management models such as APM and ACPI, but leaves those implementation details to the provider. As a rule, the suspend state will consume less power than the standby state, which in turn will consume less power than the on state. A suggested mapping of these states to other power management models is:

|  |  |  |  |
| --- | --- | --- | --- |
| ***State*** | ***ACPI*** | ***APM*** | ***Description*** |
| On | S0 | ON | Active, Powered On |
| Standby | S1 | SUSPEND | Displays and drives off, CPU, RAM and fans powered on |
| Suspend | S3 | SUSPEND | Only RAM powered |
| Off | S5 | OFF | Completely powered off |

## **POSPower Class Diagram 　　　　*Updated in Release 1.16***

The following diagram shows the relationships between the POSPower classes.

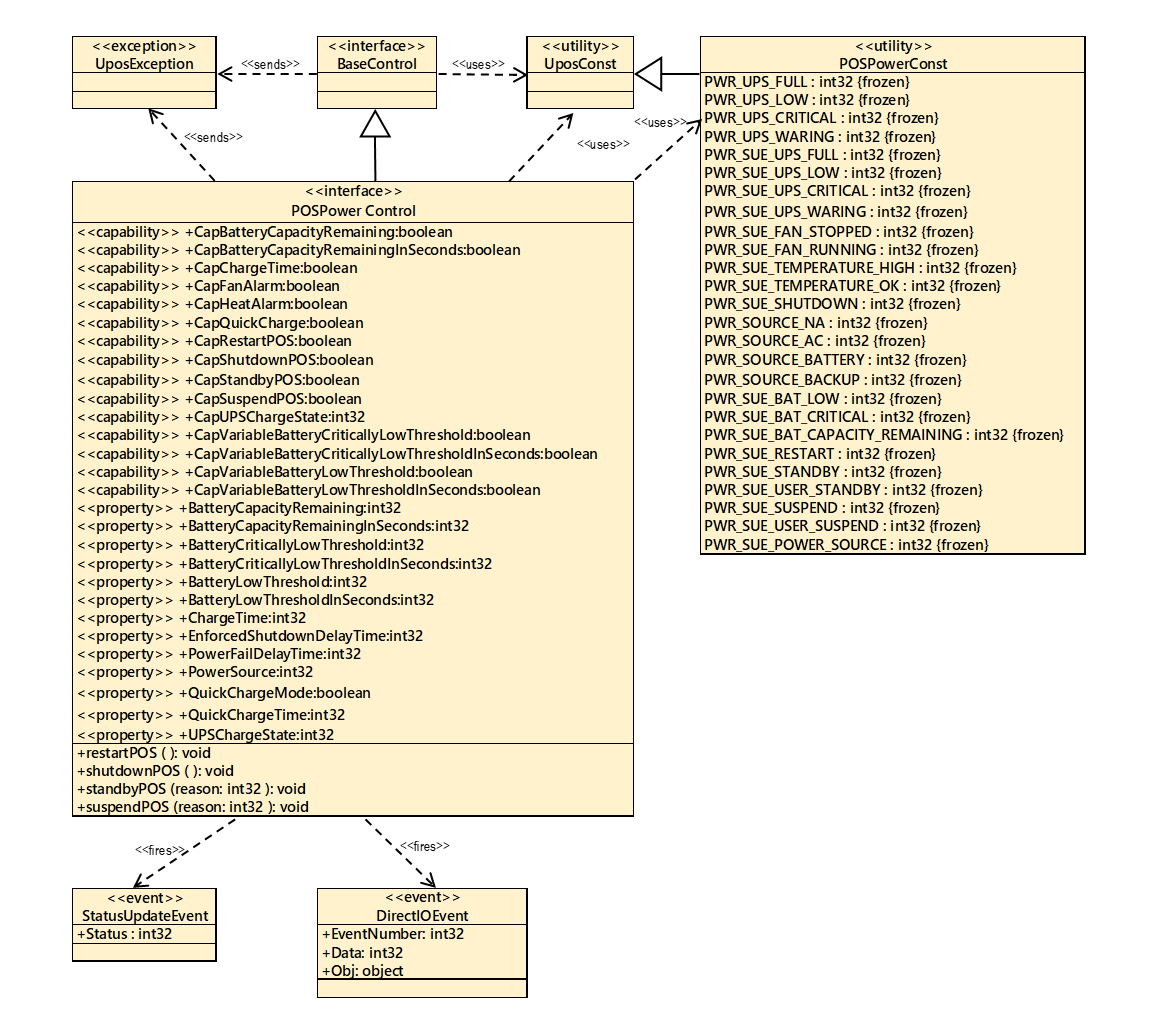


Fig. Chap.29-1 POSPower Class Diagram

## **POSPower Sequence Diagram**

The following sequence diagram shows the typical usage of the POSPower device for registering for **StatusUpdateEvent**s and an atypical case of initiating a **shutdownPOS** call.

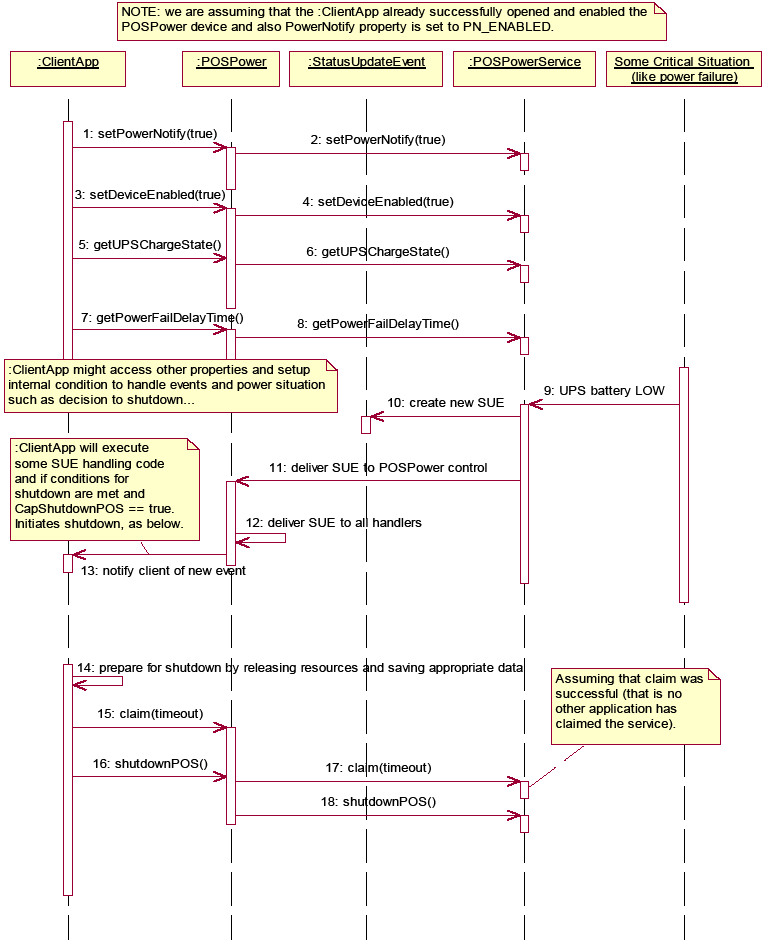


Fig. Chap. 29-2 POSPower Sequence Diagram

## **POSPower Standby Sequence Diagram**

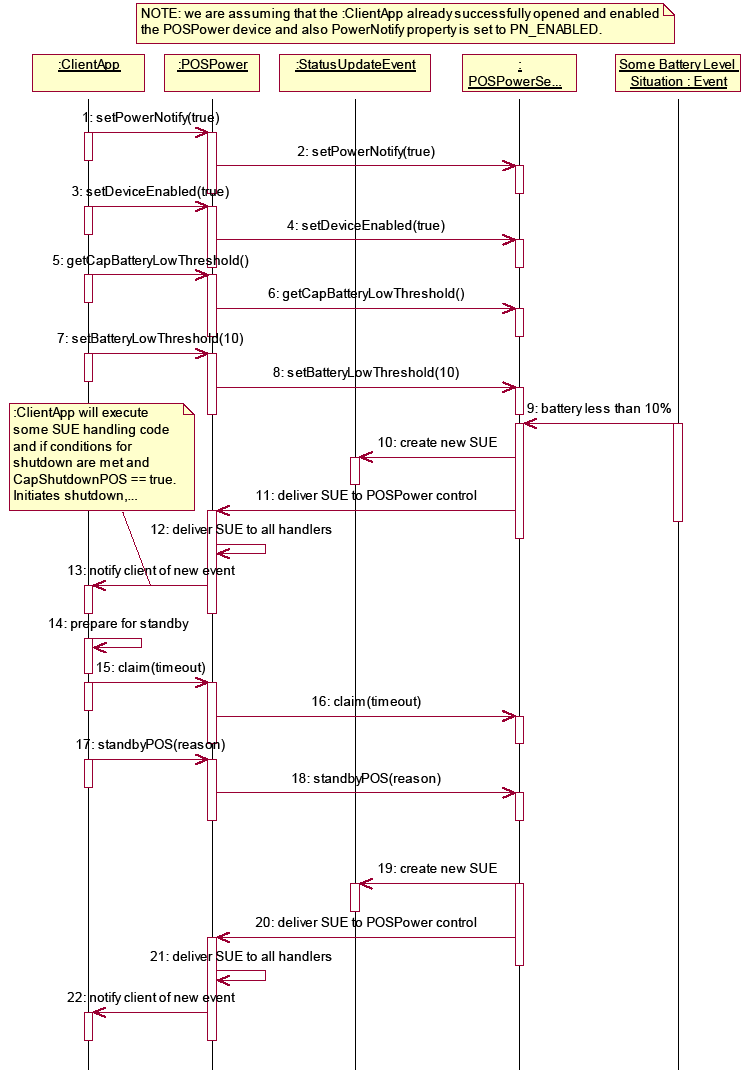


Fig. Chap. 29-3 POSPower Standby Sequence Diagram

## **POSPower State Diagram**

The following state diagram depicts the POSPower Control device model.

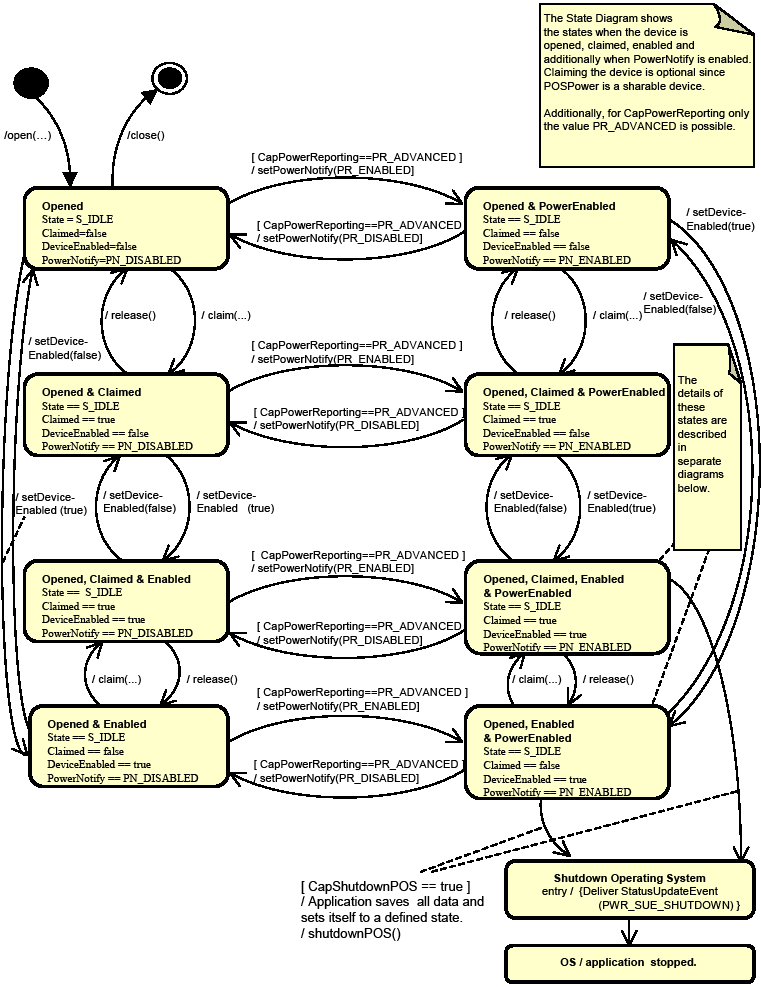


Fig. Chap. 29-4 Power State Diagram (POSPower Control Device Model)

## **POSPower PowerState Diagram - Part 1**

The following state diagram depicts the POSPower Power States.

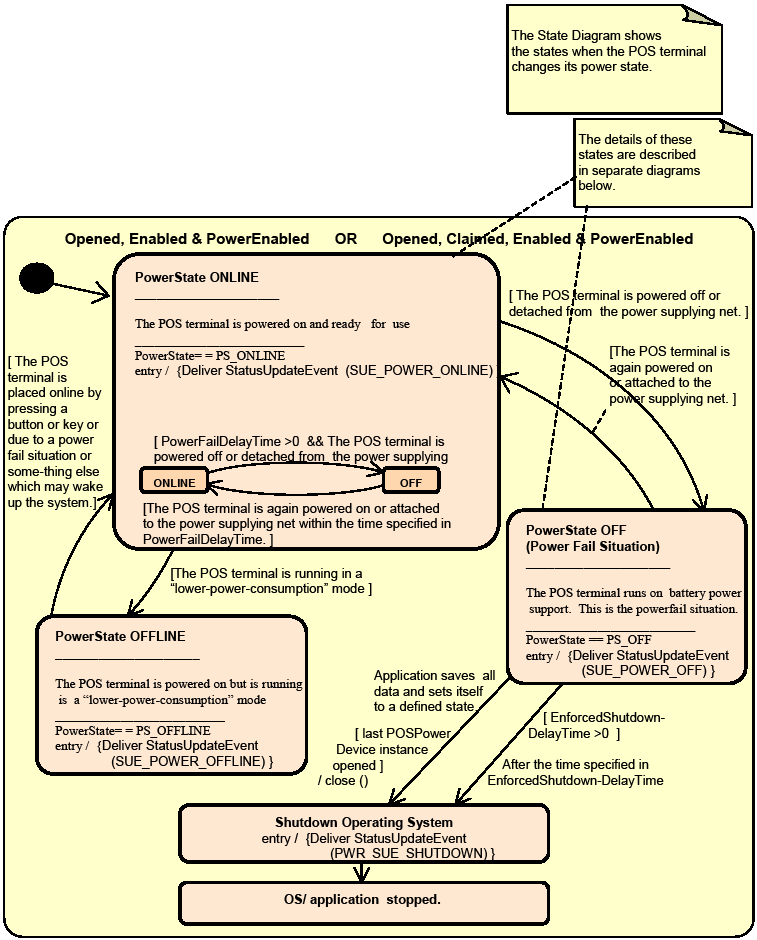


Fig. Chap. 29-5 POSPower PowerState Diagram (Part 1)

## **POSPower PowerState Diagram - Part 2**

The following state diagram depicts the POSPower PowerState ONLINE.

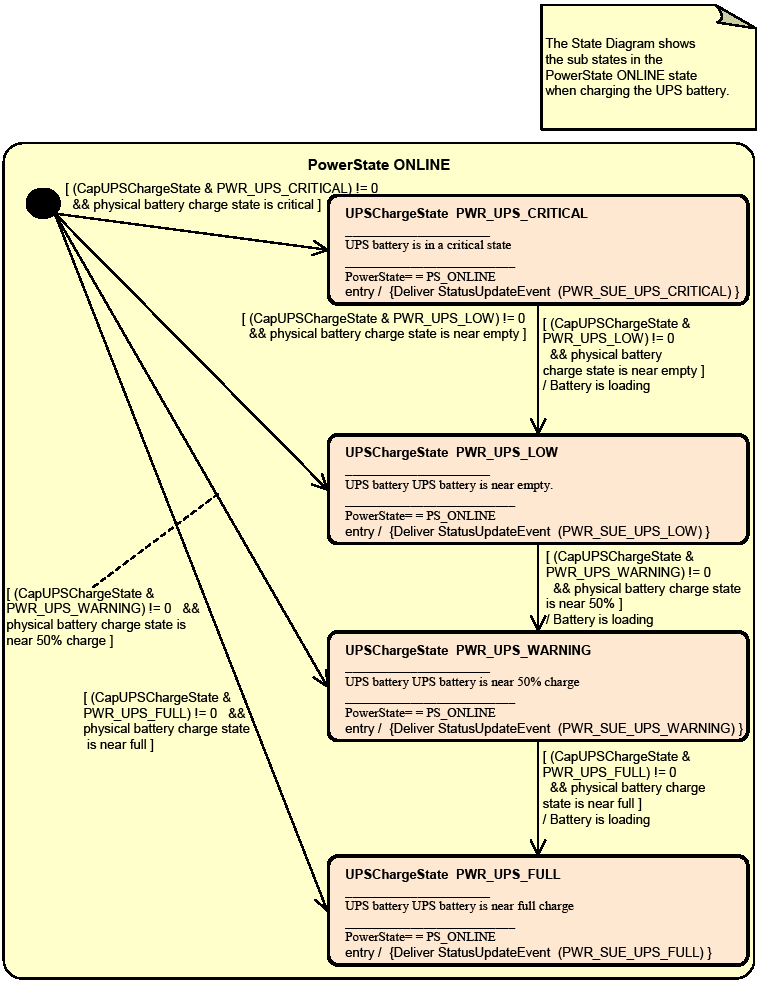


Fig. Chap. 29-6 POSPower PowerState Diagram (Part 2)

## **POSPower PowerState Diagram - Part 3**

The following state diagram depicts the POSPower PowerState OFF.

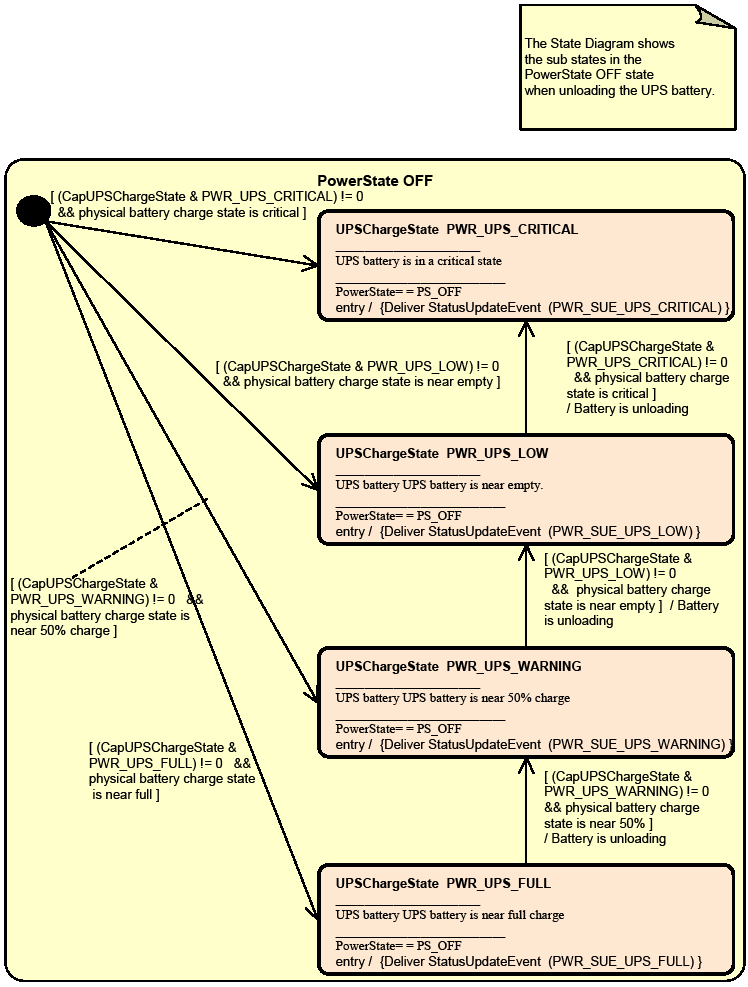


Fig. Chap. 29-7 POSPower PowerState Diagram (Part 3)

## **POSPower State Chart Diagram for Fan and Temperature**

The following state diagram depicts the handling of fan and temperature alarms.

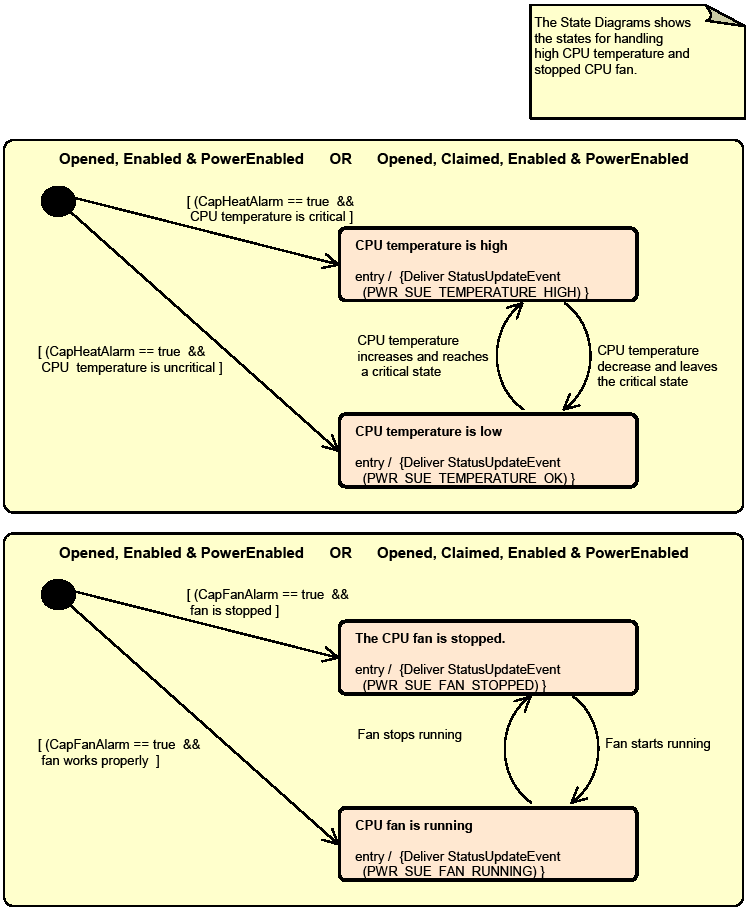


Fig. Chap. 29-8 POSPower State Chart Diagram (Fan and Temperature)

## **POSPower Battery State Diagram**

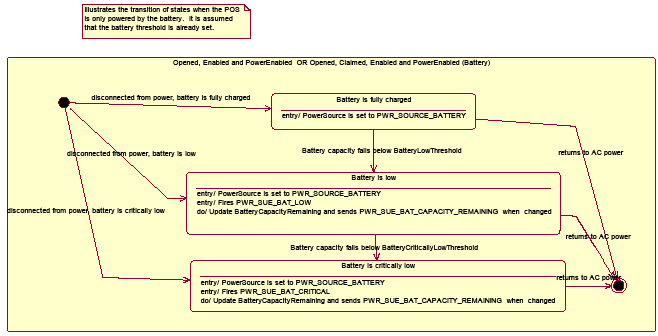


Fig. Chap. 29-9 POSPower Battery State Diagram

## **POSPower Power Transitions State Diagram**

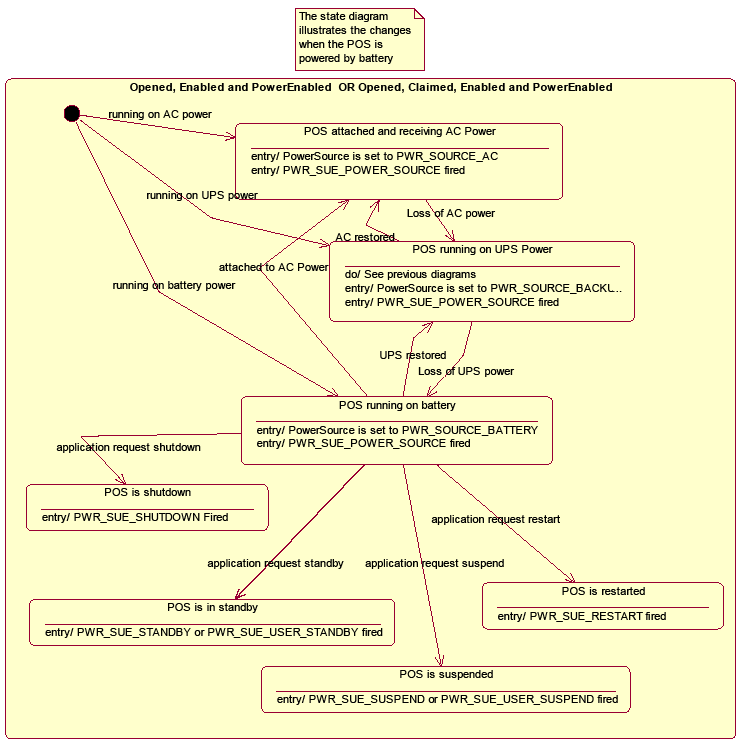


Fig. Chap. 29-10 POSPower Power Transitions State Diagram

# Properties (UML attributes)

## **BatteryCapacityRemaining Property**

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

**Remarks** A value of 0 to 100 represents percent of battery capacity remaining.

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 Intro-20.

**See Also** **CapBatteryCapacityRemaining** Property

## **BatteryCapacityRemainingInSeconds Property *Added in Release 1.16***

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

**Remarks** A value of battery capacity remaining in seconds.

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 Intro-20.

**See Also** **CapBatteryCapacityRemainingInSeconds** Property

## **BatteryCriticallyLowThreshold Property**

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

**Remarks** If not zero, this property holds the threshold at which a PWR\_SUE\_BAT\_CRITICAL **StatusUpdateEvent** is generated. The values 1 through 99 represent the percentage of the capacity remaining. The value 0 indicates that Battery Critically Low reporting is not supported or is disabled.

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 Intro-20.

**See Also** **CapVariableBatteryCriticallyLowThreshold** Property**, StatusUpdateEvent**

## **BatteryCriticallyLowThresholdInSeconds Property *Added in Release 1.16***

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

**Remarks** If not zero, this property holds the threshold at which a PWR\_SUE\_BAT\_CRITICAL **StatusUpdateEven**t is generated. The values of seconds of the capacity remaining. The value 0 indicates that Battery Critically Low reporting is not supported or is disabled.

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 Intro-20.

**See Also** **CapVariableBatteryCriticallyLowThresholdInSeconds** Property**, StatusUpdateEvent**

## **BatteryLowThreshold Property**

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

**Remarks** If not zero, this property holds the threshold at which a PWR\_SUE\_BAT\_LOW **StatusUpdateEvent** is generated. The value 1 to 99 represents the percent capacity remaining. The value 0 indicates that battery low reporting is not supported or is disabled. If variable battery low threshold is supported, setting a value between 1 and 99 sets the threshold to that value. Setting a value of zero disables battery low reporting.

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 Intro-20.

**See Also** **CapVariableBatteryLowThreshold** Property**, StatusUpdateEvent**

## **BatteryLowThresholdInSeconds Property *Added in Release 1.16***

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

**Remarks** If not zero, this property holds the threshold at which a PWR\_SUE\_BAT\_LOW **StatusUpdateEvent** is generated. The value of seconds of the capacity remaining. The value 0 indicates that battery low reporting is not supported or is disabled. If variable battery low threshold is supported, setting a value of seconds sets the threshold to that value. Setting a value of zero disables battery low reporting.

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 Intro-20.

**See Also** **CapVariableBatteryLowThresholdInSeconds** Property**, StatusUpdateEvent**

## **CapBatteryCapacityRemaining Property**

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

**Remarks** If true, the device is able to provide battery capacity information. 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 Intro-20.

**See Also** **BatteryCapacityRemaining** Property

## **CapBatteryCapacityRemainingInSeconds Property *Added in Release 1.16***

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

**Remarks** If true, the device is able to provide battery capacity information seconds. 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 Intro-20.

**See Also** **BatteryCapacityRemainingInSeconds** Property

## **CapChargeTime Property *Added in Release 1.16***

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

**Remarks** If true, the device is able to acquire the remaining time until full charging. 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 Intro-20.

**See Also** **ChargeTime** Property.

## **CapFanAlarm Property**

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

**Remarks** If true, the device is able to detect whether the CPU fan is stopped. 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 Intro-20.

## **CapHeatAlarm Property**

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

**Remarks** If true the device is able to detect whether the CPU is running at too high of a temperature. 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 Intro-20.

## **CapQuickCharge Property**

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

**Remarks** If true, the power management allows the charging of the UPS battery in quick mode. The time for charging the battery is shorter than usual. 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 Intro-20.

**See Also** **QuickChargeMode** Property, **QuickChargeTime** Property.

## **CapRestartPOS Property**

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

**Remarks** If true the device is able to explicitly restart the POS. 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 Intro-20.

**See Also** **restartPOS** Method.

## **CapShutdownPOS Property**

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

**Remarks** If true the device is able to explicitly shut down the POS. 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 Intro-20.

**See Also** **shutdownPOS** Method.

## **CapStandbyPOS Property**

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

**Remarks** If true, the device is able to request that the POS System enter the Standby state. 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 Intro-20.

**See Also** **standbyPOS** Method.

## **CapSuspendPOS Property**

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

**Remarks** If true, the device is able to request that the POS System enter the Suspend state. 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 Intro-20.

**See Also** **suspendPOS** Method.

## **CapUPSChargeState Property**

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

**Remarks** If not equal to zero, the UPS can deliver one or more charge states. It can contain any of the following values logically ORed together.

**Value Meaning**

PWR\_UPS\_FULL UPS battery is near full charge.

PWR\_UPS\_WARNING UPS battery is near 50% charge.

PWR\_UPS\_LOW UPS battery is near empty. Application shutdown should be started to ensure that is can be completed before the battery charge is depleted. A minimum of 2 minutes of normal system operation can be assumed when this state is entered unless this is the first state reported upon entering the “Off” power state.

PWR\_UPS\_CRITICAL UPS battery is in a critical state and could be disconnected at any time without further warning. 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 Intro-20.

**See Also** **UPSChargeState** Property.

## **CapVariableBatteryCriticallyLowThreshold Property**

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

**Remarks** If true, the device supports a variable threshold for critically low battery. 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 Intro-20.

**See Also** **BatteryCriticallyLowThreshold** Property**, StatusUpdateEvent**

## **CapVariableBatteryCriticallyLowThresholdInSeconds Property 　　　　　　　　　　　　　　　*Added in Release 1.16***

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

**Remarks** If true, the device supports a second’s variable threshold for critically low battery. 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 Intro-20.

**See Also** **BatteryCriticallyLowThresholdInSeconds** Property**, StatusUpdateEvent**

## **CapVariableBatteryLowThreshold Property**

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

**Remarks** If true, the device supports a variable threshold for battery low. 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 Intro-20.

**See Also** **BatteryLowThreshold** Property**, StatusUpdateEvent**

## **CapVariableBatteryLowThresholdInSeconds Property *Added in Release 1.16***

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

**Remarks** If true, the device supports a second’s variable threshold for battery low. 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 Intro-20.

**See Also** **BatteryLowThresholdInSeconds** Property**, StatusUpdateEvent**

## **ChargeTime Property *Added in Release 1.16***

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

**Remarks** Indicates the time remaining until the battery is fully charged in seconds.

If equal to zero the battery is not charging or not supported.

This property is only set if **CapChargeTime** is true.

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 Intro-20.

**See Also** **CapChargeTime** Property.

## **EnforcedShutdownDelayTime Property**

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

**Remarks** If not equal to zero the system has a built-in mechanism to shut down the POS terminal after a determined time in a power fail situation. This property contains the time in milliseconds when the system will shut down automatically after a power failure. A power failure is the situation when the POS terminal is powered off or detached from the power supplying net and runs on UPS.   
If zero no automatic shutdown is performed and the application has to call itself the **shutdownPOS** method. 　　　Applications will be informed about an initiated automatic shutdown**.** 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 Intro-20.

**See Also** **shutdownPOS** Method.

## **PowerFailDelayTime Property**

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

**Remarks** This property contains the time in milliseconds for power fail intervals which will not create a power fail situation. In some countries the power has sometimes short intervals where the power supply is interrupted. Those short intervals are in the range of milliseconds up to a few seconds and are handled by batteries or other electric equipment and should not cause a power fail situation. The power fail interval starts when the POS terminal is powered off or detached from the power supplying net and runs on UPS. The power fail interval ends when the POS terminal is again powered on or attached to the power supplying net. However, if the power fail interval is longer than the time specified in the **PowerFailDelayTime** property a power fail situation is created.

Usually this parameter is a configuration parameter of the underlying power management. So, the application can only read this property.

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 Intro-20.

## **PowerSource Property**

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

**Remarks** This property holds the current power source if power source reporting is available. A StatusUpdateEvent is generated each time this property is updated.

**Value Meaning**

PWR\_SOURCE\_NA Power source reporting is not available.

PWR\_SOURCE\_AC The current power source is the AC line.

PWR\_SOURCE\_BATTERY The current power source is a system battery. This value is only presented for systems that operate normally on battery.

PWR\_SOURCE\_BACKUP The current power source is a backup source such as an UPS or backup battery.

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 Intro-20.

**See Also** **StatusUpdateEvent**

## **QuickChargeMode Property**

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

**Remarks** If true, the UPS battery is being recharged in a quick charge mode.  
If false, it is being charged in a normal mode.

This property is only set if **CapQuickCharge** is true.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also** **CapQuickCharge** Property, **QuickChargeTime** Property.

## **QuickChargeTime Property**

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

**Remarks** This time specifies the remaining time for charging the UPS battery in quick charge mode. After the time has elapsed, the UPS battery charging mechanism of power management usually switches into normal mode.

This time is specified in milliseconds.

This property is only set if **CapQuickCharge** is true.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also** **CapQuickCharge** Property, **QuickChargeTime** Property.

## **UPSChargeState Property**

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

**Remarks** This property holds the actual UPS charge state.

It has one of the following values:

**Value Meaning**

PWR\_UPS\_FULL UPS battery is near full charge.

PWR\_UPS\_WARNING UPS battery is near 50% charge.

PWR\_UPS\_LOW UPS battery is near empty. Application shutdown should be started to ensure that is can be completed before the battery charge is depleted. A minimum of 2 minutes of normal system operation can be assumed when this state is entered unless this is the first state reported upon entering the “Off” power state.

PWR\_UPS\_CRITICAL UPS battery is in a critical state and could be disconnected at any time without further warning.

This property is initialized and kept current while the device is enabled.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20

**See Also** **CapUPSChargeState** Property.

# Methods (UML operations)

## **restartPOS Method**

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

**Remarks** Call to restart the POS terminal. This method will always restart the system independent of the system power state.

If the POSPower is claimed, only the application which claimed the device is able to restart the POS terminal.

Applications will be informed about an initiated restart.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20

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

**Value Meaning**

E\_ILLEGAL This method is not supported (see the **CapRestartPOS** property**)**

**See Also** **CapRestartPOS** Property

## **shutdownPOS Method**

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

**Remarks** Call to shut down the POS terminal. This method will always shut down the system independent of the system power state.

If the POSPower is claimed, only the application which claimed the device is able to shut down the POS terminal.

Applications will be informed about an initiated shutdown.

It is recommended that in a power fail situation an application has to call this method after saving all data and setting the application to a defined state.  
If the **EnforcedShutdownDelayTime** property specifies a time greater than zero and the application did not call the **shutdownPOS** method within the time specified in **EnforcedShutdownDelayTime**, the system will be shut down automatically. This mechanism may be provided by an underlying operating system to prevent the battery from being emptied before the system is shut down.  
This method is only supported if **CapShutdownPOS** is true.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20

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

**Value Meaning**

E\_ILLEGAL This method is not supported. 　　(See the **CapShutdownPOS** property**)**

**See Also** **CapShutdownPOS** Property, **EnforcedShutdownDelayTime** Property.

## **standbyPOS Method**

**Syntax standbyPOS (reason: *int32*):   
　　　　void{raises-exception, use after open, enable}**

**Remarks** Call to request that the system be placed into the Standby state or to respond to a request from the system, OS or other application that the system be put into Standby state.

The *reason* parameter indicates the reason the POS terminal should enter a standby state:

**Value Description**

PWR\_REASON\_REQUEST Call is to request that the system enter the standby state.

PWR\_REASON\_ALLOW Call is a response to a standby Status Update Event and specifies that the request should be allowed.

PWR\_REASON\_DENY Call is a response to a standby Status Update Event and specifies that the request should be denied.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20

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

**Value Meaning**

E\_ILLEGAL This method is not supported (see the **CapStandbyPOS** property**)**

**See Also** **CapStandbyPOS** Property.

## **suspendPOS Method**

**Syntax suspendPOS (reason: *int32*):   
void{raises-exception, use after open, enable}**

**Remarks** Call to request that the system be placed into the Suspend state or to respond to a request from the system, OS or other application that the system be put into Suspend state.

The *reason* parameter indicates the reason the POS terminal should enter a standby state:

**Value Description**

PWR\_REASON\_REQUEST Call is to request that the system enter the suspend state.

PWR\_REASON\_ALLOW Call is a response to a suspend Status Update Event and specifies that the request should be allowed.

PWR\_REASON\_DENYCall is a response to a suspend Status Update Event and specifies that the request should be denied.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20

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

**Value Meaning**

E\_ILLEGAL This method is not supported (see the **CapSuspendPOS** property**)**

**See Also** **CapSuspendPOS** Property.

# Events (UML Interfaces)

## **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 POSPower Service to provide events to the application that are not otherwise supported by the device control.

**Attributes** This event contains the following attributes:

**Attributes 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 E*ventNumber* 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 POSPower devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Errors"** on page Intro-20, **directIO** Method.

## **StatusUpdateEvent**

**<< event >> upos::events::StatusUpdateEvent  
Status : *int32* {read-only}**

**Description** Delivered when **UPSChargeState** changes or an alarm situation occurs.

**Attributes** This event contains the following attribute:

**Attributes Type Description**

*Status int32* See below.

The *Status* property contains the updated power status or alarm status.

**Value Meaning**

PWR\_SUE\_UPS\_FULL UPS battery is near full charge. Can be returned if **CapUPSChargeState** contains PWR\_UPS\_FULL.

PWR\_SUE\_UPS\_WARNING

UPS battery is near 50% charge. Can be returned if CapUPSChargeState contains PWR\_UPS\_WARNING.

PWR\_SUE\_UPS\_LOW UPS battery is near empty. Application shutdown should be started to ensure that it can be completed before the battery charge is depleted. A minimum of 2 minutes of normal system operation can be assumed when this state is entered unless this is the first charge state reported upon entering the “Off” state. Can be returned if **CapUPSChargeState** contains PWR\_UPS\_LOW.

PWR\_SUE\_UPS\_CRITICAL

UPS is in critical state, and will in short time be disconnected. Can be returned if **CapUPSChargeState** contains PWR\_UPS\_CRITICAL.

PWR\_SUE\_FAN\_STOPPED

The CPU fan is stopped. Can be returned if **CapFanAlarm** is true.

PWR\_SUE\_FAN\_RUNNING

The CPU fan is running. Can be returned if **CapFanAlarm** is true.

PWR\_SUE\_TEMPERATURE\_HIGH

The CPU is running on high temperature. Can be returned if **CapHeatAlarm** is true.

PWR\_SUE\_TEMPERATURE\_OK

The CPU is running on normal temperature. Can be returned if **CapHeatAlarm** is true.

PWR\_SUE\_SHUTDOWN The system will shut down immediately.

PWR\_SUE\_BAT\_LOW The system remaining battery capacity is at or below the low battery threshold and the system is operating from the battery.

PWR\_SUE\_BAT\_CRITICAL

The system remaining battery capacity is at or below the critically low battery threshold and the system is operating from the battery.

PWR\_SUE\_BAT\_CAPACITY\_REMAINING.

The **BatteryCapacityRemaining** property has been updated

PWR\_SUE\_BAT\_CAPACITY\_REMAINING\_IN\_SECONDS

The **BatteryCapacityRemainingInSeconds** property has been updated

PWR\_SUE\_RESTART The system will restart immediately.

PWR\_SUE\_STANDBY The system is requesting a transition to the **Standby** state

PWR\_SUE\_USER\_STANDBY

The system is requesting a transition to the **Standby** state as a result of user input.

PWR\_SUE\_SUSPEND The system is requesting a transition to the **Suspend** state.

PWR\_SUE\_USER\_SUSPEND

The system is requesting a transition to the **Suspend** state as a result of user input.

PWR\_SUE\_PWR\_SOURCE

The **PowerSource** property has been updated.

***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 “**StatusUpdateEvent**” description on page 1-34.

**See Also** **CapFanAlarm** Property, **CapHeatAlarm** Property, **CapUPSChargeState** Property**,** **UPSChargeState** Property.

# Chapter 39

# Video Capture

This Chapter defines the Video Capture device category.

## **Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.16 | open |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | 1.16 | open |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.16 | open |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | Not Supported |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **CapAssociatedHardTotalsDevice:** | *string* | {read-only} | 1.16 | open |
| **CapAutoExposure:** | *boolean* | {read-only} | 1.16 | open |
| **CapAutoFocus:** | *boolean* | {read-only} | 1.16 | open |
| **CapAutoGain:** | *boolean* | {read-only} | 1.16 | open |
| **CapAutoWhiteBalance:** | *boolean* | {read-only} | 1.16 | open |
| **CapBrightness:** | *boolean* | {read-only} | 1.16 | open |
| **CapContrast:** | *boolean* | {read-only} | 1.16 | open |
| **CapExposure:** | *boolean* | {read-only} | 1.16 | open |
| **CapGain:** | *boolean* | {read-only} | 1.16 | open |
| **CapHorizontalFlip:** | *boolean* | {read-only} | 1.16 | open |
| **CapHue:** | *boolean* | {read-only} | 1.16 | open |
| **CapPhoto:** | *boolean* | {read-only} | 1.16 | open |
| **CapPhotoColorSpace:** | *boolean* | {read-only} | 1.16 | open |
| **CapPhotoFrameRate:** | *boolean* | {read-only} | 1.16 | open |
| **CapPhotoResolution:** | *boolean* | {read-only} | 1.16 | open |
| **CapPhotoType:** | *boolean* | {read-only} | 1.16 | open |
| **CapSaturation:** | *boolean* | {read-only} | 1.16 | open |
| **CapStorageTarget:** | *int32* | {read-only} | 1.16 | open |
| **CapVerticalFlip:** | *boolean* | {read-only} | 1.16 | open |
| **CapVideo:** | *boolean* | {read-only} | 1.16 | open |
| **CapVideoColorSpace:** | *boolean* | {read-only} | 1.16 | open |
| **CapVideoFrameRate:** | *boolean* | {read-only} | 1.16 | open |
| **CapVideoResolution:** | *boolean* | {read-only} | 1.16 | open |
| **CapVideoType:** | *boolean* | {read-only} | 1.16 | open |
| **AutoExposure:** | *boolean* | {read-write} | 1.16 | open, claim & enable |
| **AutoFocus:** | *boolean* | {read-write} | 1.16 | open, claim & enable |
| **AutoGain:** | *boolean* | {read-write} | 1.16 | open, claim & enable |
| **AutoWhiteBalance:** | *boolean* | {read-write} | 1.16 | open, claim & enable |
| **Brightness:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **Contrast:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **Exposure:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **Gain:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **HorizontalFlip:** | *boolean* | {read-write} | 1.16 | open, claim & enable |
| **Hue:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **PhotoColorSpace:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **PhotoColorSpaceList:** | *string* | {read-only} | 1.16 | open |
| **PhotoFrameRate:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **PhotoMaxFrameRate:** | *int32* | {read-only} | 1.16 | open |
| **PhotoResolution:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **PhotoResolutionList:** | *string* | {read-only} | 1.16 | open |
| **PhotoType:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **PhotoTypeList:** | *string* | {read-only} | 1.16 | open |
| **RemainingRecordingTimeInSec:** | *int32* | {read-only} | 1.16 | open, claim & enable |
| **Saturation:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **StorageTarget:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **VerticalFlip:** | *boolean* | {read-write} | 1.16 | open, claim & enable |
| **VideoCaptureMode:** | *int32* | {read-only} | 1.16 | open, claim & enable |
| **VideoColorSpace:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **VideoColorSpaceList:** | *string* | {read-only} | 1.16 | open |
| **VideoFrameRate:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **VideoMaxFrameRate:** | *int32* | {read-only} | 1.16 | open |
| **VideoResolution:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **VideoResolutionList:** | *string* | {read-only} | 1.16 | open |
| **VideoType:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **VideoTypeList:** | *string* | {read-only} | 1.16 | open |

|  |  |
| --- | --- |
| **Methods (UML operations)** |  |
| ***Common*** |  |
| ***Name*** | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | 1.16 |
| **close ( ): void {raises-exception, use after open}** | 1.16 |
| **claim (timeout:** *int32***): void {raises-exception, use after open}** | 1.16 |
| **release ( ): void {raises-exception, use after open, claim}** | 1.16 |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | 1.16 |
| **clearInput ( ): void { }** | 1.16 |
| **clearInputProperties ( ):   void { }** | 1.16 |
| **clearOutput ( ): void { }** | Not  supported |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | 1.16 |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***):  void {raises-exception, use after open, enable}** | 1.16 |
| **resetStatistics (statisticsBuffer:** *string***):  void {raises-exception, use after open, enable}** | 1.16 |
| **retrieveStatistics (inout statisticsBuffer:** *string***):  void {raises-exception, use after open, enable}** | 1.16 |
| **updateFirmware (firmwareFileName:** *string***):  void {raises-exception, use after open, enable}** | 1.16 |
| **updateStatistics (statisticsBuffer:** *string***):  void {raises-exception, use after open, enable}** | 1.16 |
| ***Specific*** |  |
| ***Name*** |  |
| **startVideo (fileName:** *string***, overwrite:** *boolean***, recordingTime:** *int32***):  void {raises-exception, use after open, claim, enable}** | 1.16 |
| **stopVideo ():  void {raises-exception, use after open, claim, enable}** | 1.16 |
| **takePhoto (fileName:** *string***, overwrite:** *int32***):  void {raises-exception, use after open, claim, enable}** | 1.16 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
| **upos::events::DataEvent** |  | Not supported |  |
| **Status:** |  |  |  |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.16 |
| **EventNumber:** | *int32* | {read-only} |  |
| **Data:** | *int32* | {read-write} |  |
| **Obj:** | *object* | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  |  | 1.16 |
| **ErrorCode:** | *int32* | {read-only} |  |
| **ErrorCodeExtended:** | *int32* | {read-only} |  |
| **ErrorLocus:** | *int32* | {read-only} |  |
| **\*pErrorResponse** | *int32* | {read-write} |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  | Not Supported |  |
|  |  |  |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |

# General Information

The Video Capture Device name is “Video Capture”.

## **Capabilities**

Video capture device class has the following capabilities:

* Take a photo and record it as a file in a host and may store it in the targeted storage device.
* Take a video and record it as a file in a host and may store it in the targeted storage device.
* May read the encoded data from the bar code label with the hydra connected scanner device.
* May detect the individuals faces and/or objects with the hydra connected individual recognition device.

## **Video Capture Class Diagram**

The following diagram shows the relationships between the Video Capture classes.

Fig. Chap. 39-1 Video Capture Class Diagram

# Model

## **Modes**

The Video Capture Device has two operation modes.

・Photo Mode

・Video Mode

The operation of each mode is as follows.

・Photo Mode

Photo Mode may capture a photo image and may save it in a host as the image data file format, if **CapPhoto** property is true. Its’ capable data file format is indicated in the **PhotoType** property and all of the capable values are listed in the **PhotoTypeList** property. And the device may save the file in the targeted storage device that is specified by the **StorageTarget** property, if **CapStorageTarget** value is CST\_HARDTOTAL\_ONLY or CST\_ALL.

・Video Mode

Video Mode may capture a video image data and may save it in a host as the video image data file format, if **CapVideo** property is true. Its’ capable data file format is indicated in the **VideoType** property and all of the capable values are listed in the **VideoTypeList** property. And the device may save the file in the targeted storage device that is specified by the **StorageTarge**t property, if **CapStorageTarget** value is CST\_HARDTOTAL\_ONLY or CST\_ALL.

## **Device behaviors**

“Video capture device” device control follows the device behavior as follows. 　 They are different in each mode as described below.

## **Photo Mode**

If **CapPhoto** property is true, this mode can be executed. Prior to start this mode, “**Video Capture Device**” device control needs to set the **VideoCaptureMode** property as to be VCAP\_VCMODE\_PHOTO. And each of **CapPhotoColorSpace**, **CapPhotoFrameRate, CapPhotoResolution**, **CapPhotoType** property is true and these **PhotoColorSpaceLis**t, **PhotoMaxFrameRate, PhotoResolutionList** and **PhotoTypeList** should have the appropriate values to be used as the photo file data in this targeted device. And then it needs to set the appropriate values in the each of **PhotoColorSpace** property, **PhotoFrameRate** property, **PhotoResolution** property and **PhotoType** property.

It starts photo capturing by executing the **takePhoto** method. Then, “**Video Capture Device**” device control may capture a photo image and may save it in a host as an image data file format specified by the value of **PhotoType** property that is listed in the **PhotoTypeList** property. And may store it in the storage device specified by the **StorageTarget** property, if **CapStorageTarget** value is CST\_HARTTOTAL\_ONLY or CST\_ALL. Then the file name is set by the **takePhoto** method parameter and can deliver the photo data file to the application. 　If device needs to be able to write the image data file to an associated HardTotals device, the **CapAssociatedHardTotalsDevice** property holds the open name of the associated HardTotals device.

This method is executed synchronously. Only one call to **takePhoto** method can be in progress at a time. An attempt to nest video capture device operations will result in an **UPOSException** being thrown.

When it exceeded the specified parameter time out or when photo file generation is finished or when **clearInput** method is executed, the taking photo process will be ended.

**StatusUpdateEvent** with status VCAP\_SUE\_START\_PHOTO is evoked when **takePhoto** method is executed to notify the application that recording state has started.

When the taking photo is finished, or the specified time out has been exceeded, a **StatusUpdateEvent** with status VCAP\_SUE\_END\_PHOTO is evoked to notify the application that photo taking has been ended.

An **ErrorEven**t event (or events) is enqueued if an error occurs while gathering or processing input, and is delivered to the application when **DataEventEnabled** property is true and other event delivery requirements are met. The **DataCount** property may be read to obtain the total number of enqueued events.

If **ErrorEvent** response is ER\_CONTINUEINPUT, the process of input processing continues. However, as long as the cause of the error is not resolved, the ErrorEvent will occur again immediately.

If **ErrorEvent** is ER\_CLEAR, the input processing process is terminated and the taking photo is discarded.

All enqueued input may be deleted by calling **clearInput** method. See the **clearInput** method description for more details.

All data properties that are populated as a result of firing an **ErrorEvent** event can be set back to their default values by calling the **clearInputProperties** method.

## **Video Mode**

Prior to start this mode, “**Video Capture Device**” device control needs to set the **VideoCaptureMode** property as to be VCAP\_VCMODE\_VIDEO. And each of **CapVideoColorSpace**, **CapVideoFrameRate, CapVideoResolution** and **CapVideoType** property is true and these V**ideoColorSpaceList**, **VideoMaxFrameRate, VideoResolutionList** and **VideoTypeList** should have the appropriate values to be used as the video image data file in this targeted device. And then it needs to set the appropriate values in the each of **VideoColorSpace** property, **VideoFrameRate** property, **VideoResolution** property and **VideoType** property.

It starts video image capturing by executing the **startVideo** method. Then “**Video Capture Device**” device control captures a video image and save it in a host with the filename specified value of **VideoType** property that is listed in the **VideoTypeList** property. And may store it in the storage device specified by the **StorageTarget** property, if **CapStorageTarget** value is CST\_HARTTOTAL\_ONLY or CST\_ALL. And the file name is set by the **startVideo** method parameter and can deliver the video image data file to the application. This method is executed synchronously.

The video capturing ends after the specified time has elapsed or when **stopVideo** method is called or when **clearInput** method is called.

The remaining video capture recording time in seconds can be obtained from the property **RemainingRecordingTimeInSec**.

**StatusUpdateEvent** with status VCAP\_SUE\_START\_VIDEO is evoked when **startVideo** method is executed to notify the application that taking video has been started.

When the taking video is finished, or the specified time out has been exceeded, a **StatusUpdateEvent** with status VCAP\_SUE\_END\_VIDEO is evoked to notify the application that taking video has been ended.

If the time specified by the **startVideo** method is FOREVER(-1), execution will continue until the **stopVideo** method is called. When **stopVideo** is called, the previous taking video data may be recorded in a host and deliver to the targeted storage device specified by the **StorageTarget** property, if **CapStorageTarget** property value is CST\_HARTTOTAL\_ONLY or CST\_ALL. And it can be delivered to the application with the specified file name that is set by the **startVideo** method.

Only one call to **startVideo** method can be in progress at a time. An attempt to nest taking video operations will result in an UPOSException being thrown.

If Error occurs during the execution of the **startVideo** method, application may call the **stopVideo** method to terminate the taking video process or cancel the taking video process by calling the **clearInput** method before ending the **ErrorEvent** processing. After this when the **stopVideo** method is called, the video file data until just before the **ErrorEvent** occur is stored to the host and targeted storage device that is specified by the **StorageTarget** property, if **CapStorageTarget** property value is CST\_HARTTOTAL\_ONLY or CST\_ALL, and can be delivered to the application.

If **ErrorEvent** response is ER\_CONTINUEINPUT, the process of input processing continues. However, as long as the cause of the error is not resolved, the **ErrorEvent** will occur again immediately.

If **ErrorEvent** is ER\_CLEAR, the input processing process is terminated and the taking video is discarded.

An **ErrorEven**t event (or events) is enqueued if an error occurs while gathering or processing input, and is delivered to the application when **DataEventEnabled** property is true and other event delivery requirements are met. The **DataCount** property may be read to obtain the total number of enqueued events.

If there is no error during the execution of **startVideo** method, it is possible to terminate the taking video process and can stop the taking video anytime. When the **stopVideo** method is called, the video data until just before the method is called, may be recorded in the host and targeted storage device that is specified by the **StorageTarget** property if **CapStorageTarget** property is CST\_HARTTOTAL\_ONLY or CST\_ALL, and can deliver it to the application.

All enqueued input may be deleted by calling **clearInput** method. See the clearInput method description for more details.

All data properties that are populated as a result of firing an **ErrorEvent** event can be set back to their default values by calling the **clearInputProperties** method.

## **Device Sharing**

Video capture 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 accessing many video capture-specific properties.
* The application must claim and enable the device before calling methods that manipulate the device.
* See the “Summary” table for precise usage prerequisites.

# Properties (UML attributes)

## **AutoExposure Property**

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

**Remarks** If true, auto exposure of camera is enabled. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also** **CapAutoExposure** Property

## **AutoFocus Property**

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

**Remarks** If true, auto focus of camera is enabled. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E*\_*ILLEGAL An invalid value was specified. Or it does not support this function.

**See also CapAutoFocus** Property

## **AutoGain Property**

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

**Remarks** If true, auto gain of camera is enabled. Otherwise it is false.

When this property is true, it is possible to read the value of **Gain** property. However, it is not possible to write and change the value of **Gain** property. If **AutoGain** property is false, then, it is possible to read, write and change the value of **Gain** property.

　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 Intro-20. Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also** **CapAutoGain** Property **Gain** Property

## **AutoWhiteBalance Property**

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

**Remarks** If true, auto white balance of camera is enabled. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also** **CapAutoWhiteBalance** Property

## **Brightness property**

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

**Remarks** Indicate the brightness of camera. Valid values range from 0 to 100. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning** E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See Also CapBrightness** Property

## **CapAssociatedHardTotalsDevice Property**

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

**Remarks** Holds the open name of the associated HardTotals device if the device is able to write to such devices which is the case if **CapStorageTarget** is either CST\_ALL or CST\_HARDTOTALS\_ONLY. If **CapStorageTarget** is CST\_HOST\_ONLY this property value must be the empty string.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapStorageTarget** Property

## **CapAutoExposure Property**

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

**Remarks** If true, can change the auto exposure of camera. 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 Intro-20.

**See also** **AutoExposure** Property

## **CapAutoFocus Property**

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

**Remarks** If true, can change the auto focus of camera. 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 Intro-20.

**See also AutoFocus** Property

## **CapAutoGain Property**

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

**Remarks** If true, automatic gain change of the camera is possible. 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 Intro-20.

**See also AutoGain** Property

## **CapAutoWhiteBalance Property**

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

**Remarks** If true, auto white balance of camera is possible. 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 Intro-20.

**See also** **AutoWhiteBalance** Property

## **CapBrightness Property**

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

**Remarks** If true, the brightness of camera can be changed. 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 Intro-20.

**See also** **Brightness** Property

## **CapContrast Property**

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

**Remarks** If true, can change the contrast of camera. Otherwise, it if 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 Intro-20.

**See also** **Contrast** Property

## **CapExposure Property**

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

**Remarks** If true, can change the exposure of camera. 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 Intro-20.

**See also Exposure** Property

## **CapGain Property**

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

**Remarks** If true, can change the gain of camera. 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 Intro-20.

**See also Gain** Property

## **CapHorizontalFlip Property**

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

**Remarks** If true, can change the horizontal flip of camera. 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 Intro-20.

**See also** **HorizontalFlip** Property

## **CapHue Property**

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

**Remarks** If true, the hue of the camera can be changed. 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 Intro-20.

**See also** **Hue** Property

## **CapPhoto Property**

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

**Remarks** If true, it supports the photo function and can take a photo. And to activate the photo mode, the **VideoCaptureMode** property value needs to set VCAP\_VCMODE\_PHOTO. If false, it’s not supporting the photo 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 Intro-20.

**See also** **VideoCaptureMode** Property

## **CapPhotoColorSpace Property**

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

**Remarks** If true, can handle and change the photo color space. 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 Intro-20

**See also** **PhotoColorSpace** Property

## **CapPhotoFrameRate Property**

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

**Remarks** If true, can handle and change the capture frame rate. 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 Intro-20.

**See also** **PhotoFrameRate** Property

## **CapPhotoResolution Property**

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

**Remarks** If true, taking photo resolution is handled and can be changed. 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 Intro-20.

**See also PhotoResolution** Propert**y**

## **CapPhotoType Property**

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

**Remarks** If true, photo image format type can be changed. 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 Intro-20.

## **CapSaturation Property**

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

**Remarks** If true, can change the saturation of camera. 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 Intro-20.

**See also** **Saturation** Property

## **CapStorageTarget Property**

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

**Remarks** This is an enumeration and announces where the device is able to write the recorded sound data file to.  
It holds one of the following values.

**Value　　　　　　　　　　　 Meaning**

CST\_HARDTOTAL\_ONLY　　 Only an associate **HardTotals** device is　supported.

CST\_HOST\_ONLY Only the host’s file system is supported.

CST\_ALL Both, the associated **HardTotals** device and the host’s file system is supported.

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

If a HardTotals device is supported the storage target, the property value should be CST\_HARDTOTAL\_ONLY or CST\_ALL, and the property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **StorageTarget** Property, **CapAssociatedHardTotalsDevice** Property

## **CapVerticalFlip Property**

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

**Remarks** If true, can change the vertical flip of camera. 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 Intro-20.

**See also** **VerticalFlip** Property

## **CapVideo Property**

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

**Remarks** If true, video function is supported. Otherwise, it is false. If this property is true, taking video and recording can be done by calling the **startVideo** method. And to activate the video mode, the **VideoCaptureMode** property value needs to set VCAP\_VCMODE\_VIDEO. If false, taking video and recording cannot be performed. 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 Intro-20.

**See also** **StartVideo** Method, **VideoCaptureMode** Property

## **CapVideoColorSpace Property**

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

**Remarks** If true, can change the color space when taking the video. 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 Intro-20

**See also** **VideoColorSpace** Property

## **CapVideoFrameRate Property**

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

**Remarks** If true, can change the video frame rate from 1 to up to **VideoMaxFrameRate** property value. 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 Intro-20.

**See also** **VideoMaxFrameRate** Property, **VideoFrameRate** Property

## **CapVideoResolution Property**

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

**Remarks** If true, taking video resolution can be changed and all of possible values are listed in the **VideoResolutionList** property values. If false, taking video resolution cannot be changed. 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 Intro-20.

**See also VideoResolutionList** Property, **VideoResolution** Property

## **CapVideoType Property**

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

**Remarks** If true, taking video type can be changed and all of possible values are listed in the **VideoTypeList** values. 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 Intro-20.

**See also VideoTypeList** Property, **VideoType** Property

## **Contrast Property**

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

**Remarks** Indicate the contrast of the camera. Valid values range from 0 to 100. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning** E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See Also CapContrast** Property

## **Exposure Property**

**Syntax** **Exposure: *int32* {read-write, access after open} Remarks** Indicate the exposure of camera. Valid values range from 0 to 100. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also** **CapExposure** Property

## **Gain Property**

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

**Remarks** Indicate the gain of camera. Valid values range from 0 to 100. If **AutoGain** property is true, it is possible to read the value of **Gain** property. However, it is not possible to write and change the value of **Gain** property. If **AutoGain** property is false, then, it is possible to read, write and change the value of **Gain** property. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also CapGain** Property, **AutoGain** Property

## **HorizontalFlip Property**

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

**Remarks** If true, horizontal flip of camera is enabled and it is possible to reverse the camera captured image horizontally. Otherwise, it is false. There is a similar property called **VerticalFlip** property. However, each **VerticalFlip** property and **HorizontalFlip** property value can be set independently. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare: **Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See Also** **CapHorizontalFlip** property, **VerticalFlip** property,  
 **CapVerticalFlip** property

## **Hue Property**

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

**Remarks** Indicate the hue of camera. Valid values range from 0 to 100. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also** **CapHue** Property

## **PhotoColorSpace Property**

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

**Remarks** Indicates the photo color space ID of the frame data to be acquired by the Video Capture Device, if **CapPhotoColorSpace** property is true. Valid values are one of the values listed in the **PhotoColorSpaceList** property. 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 Intro-20. Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See also** **PhotoColorSpaceList** Property, **CapPhotoColorSpace** Property

## **PhotoColorSpaceList Property**

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

**Remarks** Photo Color space information supported by the device is indicated in a comma-separated list. Each color space information is composed of the following information and is shown in the following order separated by a colon (":"). This property is initialized by the **open** method.

**Parameter Description**

*Color space ID* ID for identifying the color space of RGB, YUV 422, etc. And they are indicating like “RGB;YUV422;…..”

*Depth* Number of bits per 1 pixel

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

***See also*** **PhotoColorSpace** Property

## **PhotoFrameRate Property**

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

**Remarks**  Indicates the frame rate of frame data to be acquired by the Video Capture Device. Valid values range from 1 to **PhotoMaxFrameRate** property. 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 Intro-20.

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See also** **CapPhotoFrameRate** Property, **PhotoMaxFrameRate** Property,

## **PhotoMaxFrameRate Property**

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

**Remarks** Indicates the maximum frame rate that can be set for the **PhotoFrameRate** property. 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 Intro-20.

**See also PhotoFrameRate** Property

## **PhotoResolution Property**

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

**Remarks** It shows the resolution of the frame data acquired by the Video Capture Device and the photo taken and recorded with the **takePhoto** method. Valid values are one of those listed in **PhotoResolutionList** property. 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 Intro-20. Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See also** **PhotoResolutionList** Property, **VideoCaptureMode** Property, **takePhoto** Method

## **PhotoResolutionList Property**

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

**Remarks** Indicating the comma-separated list of possible resolutions for the **PhotoResolution** property. Resolution is indicated in "horizontal x height" format. For example, when you support 320x240, 640x480, 640x360, it is the following. "320x240,640x480,640x360".  
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 Intro-20.

**See also PhotoResolution** Propert**y**

## **PhotoType Property**

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

**Remarks** Indicates the data format of photo taken with the **takePhoto** method. Valid values are one of the values listed in the **PhotoTypeList** property.  
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 Intro-20.  
Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning** E\_ILLEGAL An invalid value was specified.

**See also PhotoTypeList** Property**, takePhoto** Method

## **PhotoTypeList Property**

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

**Remarks** A comma-separated list of photo image format values that can be set for the **PhotoType** property. For example, when supporting BMP and JPEG, it is the following. "BMP,JPEG"

Note: The notation contents may be different depending on the 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 Intro-20.

**See also** **PhotoType** Property

## **RemainingRecordingTimeInSec Property**

**Syntax RemainingRceordingTimeInSec: *int32* {read-only, access after open-claim-enable}**

**Remarks** This property holds the remaining recording time in seconds if a video recording is ongoing. If no video recording is ongoing its value is 0. When a call to method **startVideo** returns, this property initially holds the time passed as argument *recordingTime* to that call. If this argument value is FOREVER (-1), this property also holds this value unchanged until **stopVideo** method has been called. This property is initialized during device set **DeviceEnabled** method to 0.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also startVideo** Method**, stopVideo** Method

## **Saturation Property**

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

**Remarks** Indicate the saturation of camera. Valid values range from 0 to 100. 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 Intro-20.Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also** **CapCameraSaturation** Property

## **StorageTarget Property**

**Syntax StorageTarget: *string* {read-write, access after open-claim-enable}**

**Remarks** Holds the open name of HardTotals device, if **CapStorageTarget** property value is CST\_HARTTOTAL\_ONLY or CST\_ALL. If **CapStorageTarget** property value is CST\_HOST\_ONLY, this property value must be the empty string. This property is initialized by the **open** method according to the value hold by **CapStorageTarget**.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also CapStorageTarget** Property

## **VerticalFlip Property**

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

**Remarks** If true, vertical flipping of the video is enabled and it is possible to reverse the video or photo image capturing vertically. Otherwise, it is false. There is a similar property called **HorizontalFlip** property and each **VerticalFlip** property and **HorizontalFlip** property value can be set independently. 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 Intro-20. Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or it does not support this function.

**See also** **CapVerticalFlip** Property, **HorizontalFlip** Property,  
 **CapHorizontalFlip** Property

## **VideoCaptureMode Property**

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

**Remarks** Indicate the operation mode of video capture device.  
Valid values are as follows

**Parameter Description**

VCAP\_VCMODE\_PHOTO

This mode is for taking photo and their data recording. Can be set when **CapPhoto** property is true.  
The values of the **PhotoType** property, **PhotoColorSpace** property**, PhotoResolution** property **PhotoFrameRate** property are applied to the taking photo image formats list in the **PhotoTypeList** property, the color space values list in the **PhotoColorSpaceList** property**,** the resolution values list in the **PhotoResolutionList** property, and the frame rate values within the values of **PhotoMaxFrameRate** property. And taking photo is executed by the **takePhoto** method.

VCAP\_VCMODE\_VIDEO

This mode is for taking the videos and their data recording. Can be set when **CapVideo** property is true. The value of the **VideoTyp**e property, **VideoColorSpace** property, **VideoResolution** property and **VideoFrameRate** property are applied to the taking video image format list in the **VideoTypeList** property, the color space values list in the **VideoColorSpaceList** property,theresolution values list in the **VideoResolutionList** propertyand frame rate values within the values of **VideoMaxFrameRate** property. Taking the videos and their data recording will be executed by the **startVideo** methodandends taking the video by using the **stopVideo** method.

This property is initialized by the by the **open** method. The default value of this property is VCAP\_VCMODE\_PHOTO.

**Errors** A UposException may be thrown when this property is accessed.  
 For further information, see “**Errors**" on page Intro-20.

**See also PhotoColorSpace** Property, **VideoColorSpace** Property, **PhotoResolution** Property, **VideoResolution** Property, **VideoFrameRate** Property, **PhotoFrameRate** Property, **CapPhotoColorSpace** Property, **CapVideoColorSpace** Property, **CapPhotoResolution** Property, **CapVidoeResolution** Property, **VideoMaxFrameRate** Property,  
 **PhotoMaxFrameRate** Property, **VideoCaptureMode** Property, **CapPhoto** Property, **CapVideo** Property, **VideoType** Property, **VideoTypeList** Property **PhotoType** Property, **PhotoTypeList** Property, **takePhoto** Method, **startVideo** Method, **stopVideo** Method.

## **VideoColorSpace Property**

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

**Remarks** Indicates the video color space ID of the frame data to be used by **startVideo** method. Valid values are one of the values listed in the **VideoColorSpaceList** property. This property is referred to when **VideoCaptureMode** property value is VCAP\_VCMODE\_VIDEO and **CapVideo** is true. 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 Intro-20. Some possible values of the exception’s*ErrorCode* propertyare:

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See also** **VideoColorSpaceList** Property, **VideoCaptureMode** Property, **stardVideo** Method

## **VideoColorSpaceList Property**

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

**Remarks** Video Color space information supported by the device is indicated in a comma-separated list. Each color space information is composed of the following information and is shown in the following order separated by a colon (":"). This property is initialized by the **open** method.

**Parameter Description**

*Color space ID* ID for identifying the color space of RGB, YUV 422, etc. And they are indicating like“RGB;YUV422;…..”

*Depth* Number of bits per 1 pixel

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See also** **CapVideoColorSpace** Property, **VideoColorSpace** Property

## **VideoFrameRate Property**

**Syntax** **VideoFrameRate;** ***int32* {read-write, access after open}**

**Remarks** Indicates the frame rate of the frame data recorded by the Video Capture Device and the video image capturing and recorded with the **startVideo** method. Valid values range from 1 to **VideoMaxFrameRate** property and **CapVideo** property is true. This property is only applied when VCAP\_VCMODE\_VIDEO is set in **VideoCaptureMode** property. 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 Intro-20. Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning**  E\_ILLEGAL An invalid value was specified.

**See also** **VideoMaxFrameRate** Property, **CapVideo** Property  
 **VideoCaptureMode** Property, **startVideo** Method

## **VideoMaxFrameRate Property**

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

**Remarks** Indicates the maximum video recording frame rate that can be set in **VideoFrameRate** property. 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 Intro-20.

**See also** **VideoFrameRate** Property

## **VideoResolution Property**

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

**Remarks** Indicates the resolution of video image data acquired by the Video Capture Device and recorded with the execution of **startVideo** method. Valid values are one of the values listed in the **VideoResolutionList** property. This property is only applied when VCAP\_VCMODE\_VIDEO is set in **VideoCaptureMode** property and if **CapVide**o property is true. 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 Intro-20. Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning** E\_ILLEGAL An invalid value was specified.

**See also** **VideoResolutionList** Property, **CapVideo** Property  
 **VideoCaptureMode** Property, **startVideo** Method

## **VideoResolutionList Property**

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

**Remarks** A comma-separated list of possible resolutions for the **VideoResolution** property. Resolution is indicated by "Horizontal resolution number x Vertical resolution number" format. For example, when it supports 320x240, 640x480, 640x360, it is the following. "320x240,640x480,640x360" 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 Intro-20.

**See also CapVideoResolution** Property, **VideoResolution** Property

## **VideoType Property**

**Syntax** **VideoType;** ***string* {read-write, access after open}**

**Remarks** Indicate the shape of the taking video and recorded with the **startVideo** method. Valid values are one of those listed in **VideoTypeList**  property. This property is applied when VCAP\_VCMODE\_VIDEO is set in **VideoCaptureMode** property and if **CapVideo** property is true. 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 Intro-20. Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning**  E\_ILLEGAL An invalid value was specified.

**See also** **VideoCaptureMode** Property, **CapVideo** Property, **VideoTypeList** Property, **startVideo** Method

## **VideoTypeList Property**

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

**Remarks** A comma-separated list of image format values that can be set for the **VideoType** property. For example, when AVI\_IYUV, AVI\_MJPG is supported, it is the following "AVI\_IYUV,AVI\_MJPG".  
 Note: The notation contents may be different depending on the 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 Intro-20.

**See also** **CapVideoType** Property, **VideoType** Property

# Note: Video Capture Device Property Value Relationship

Properties listed below are related within each Photo / Video Mode group, and if any value change occurs, other values may change accordingly.

**Photo Mode Group Properties**

**PhotoType, PhotoColorSpace, PhotoColorSpaceList, PhotoFrameRate, PhotoMaxFrameRate, PhotoResolution, PohtoResolutionList**

**Video Mode Group Properties**

**VideoType, VideoColorSpace, VideoColorSpaceList, VideoFrameRate, VideoMaxFrameRate, VideoResolution, VideoResolutionList**

# Methods (UML operations)

## **startVideo Method**

**Syntax startVideo (fileName : string, overwrite: *boolean*, recordingTime: *int32*): void{raises-exception, use after open-claim-enable}  
Parameter Description**

filename Specify the name of the video file to be recorded. Overwrite Specify the behavior when the same name file exists. If true, it is overwritten. If false, it will raise the UposException. recordingTime Specify the time for video recording in seconds. If FOREVER (-1) is specified, recording will continue until the **stopVideo** method is called.

**Remarks** Before calling this method, it needs to set the **VideoCaptureMode** property to VCAP\_VCMODE\_VIDEO and **CapVideo** property is true. Video capturing and recording starts with the setting contents of the **VideoColorSpace** property**, VideoResolution** property, **VideoFrameRate** property and **VideoType** property. This method is executed asynchronously. **StatusUpdateEvent** will notify the application that there is a change in the power status or a state change during video capturing and recording.  
When the time specified in recordingTime has elapsed, or by calling the **stopVideo** method, recording is completed and the video file specified by fileName is recorded and can deliver to the application  
Also, S\_BUSY is set in the **Status** property during video capturing and recording. The place where video files are recorded is the area as the value of **StorageTarget** Property.

**Errors** A UposException may be thrown when this method is invoked.  
For further information, see “**Errors**" on page Intro-20.  
Some possible values of the exception's *ErrorCode* property are:

**Value Meaning** E\_ILLEGAL fileName is too long or contains characters that cannot be used, or 0 is specified for recordingTime. **VideoCaptureMode** property is not VCAP\_VCMODE\_VIDEO E\_EXISTS fileName already exists. (If overwrite is false) E\_BUSY Cannot execute because it is recording.

**See also VideoColorSpace** Property**, VideoResolution** Property**, VideoFrameRate** Property**, VideoType** Property**, stopVideo** Method, **StatusUpdateEvent** Event, **VideoCaptureMode** Property

## **stopVideo Method**

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

**Remarks** The video capturing and recording process started by the **startVideo** method has been ended and the taking video is completed. This method processed synchronously. **StatusUpdateEven**t will notify the application that there is a change in the power status or a state change during taking video and recording.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20. Some possible values of the exception's *ErrorCode* property are:

**Value Meaning**

E\_ILLEGAL It is not recorded.

**See also startVideo** Method, **StatusUpdateEvent** Event

## **takePhoto Method**

**Syntax takePhoto (fileName: *string*, overwrite: *boolean*, timeout: *int32*): void{raises-exception, use after open-claim-enable}**

**Parameter Description**

fileName Specify the image file name to be recorded. overwrite Specify the behavior when the same name file exists. If true it overwrites. If false, UposException is thrown.

**Remarks** Take photo and record with setting contents of **PhotoColorSpace** property, **PhotoResolution** property, **PhotoFrameRate** Property and **PhotoType** property. Before calling this method, it needs to set the **VideoCaptureMode** property to VCAP\_VCMODE\_PHOTO and this method can be executed if **CapPhoto** property is true. This method is executed synchronously. The location where photo files are recorded is the area managed by the associated “**Hard Totals**" service device.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20. Some possible values of the exception's *ErrorCode* property are:

**Value Meaning**

E\_ILLEGAL One of the following occurred. FileName is too long or contains unusable characters. **VideoCaptureMode** property is not VCAP\_VCMODE\_PHOTO and **CapPhoto** property is not true.

E\_EXISTS fileName already exist. (When overwrite=false)

**See also** **VideoCaptureMode** Property**, PhotoColorSpace** Property**,  
PhotoResolution** Property**, CapPhoto** Property, **PhotoType** Property, **PhotoFrameRate** Property, **StatusUpdateEvent** Event

# Events (UML interfaces)

## **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 Video Capture Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** method

## **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 a Video Capture Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is re-tried, and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is re-tried, and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period of time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **StatusUpdateEvent**

**<< event >> upos::events::StatusUpdateEvent**

**Status : *int32* {read-only}**

**Description** *Notifies the application that there is a change in the power status or a state change of the Video Capture device.*

**Attributes**  This event contains the following attribute:

**Attributes Type Description**

*Status* *int32* Indicates a change in the power status or a sate change of the unit.

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process. See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

VCAP\_SUE\_START\_VIDEO\_RECORDING  
 It will be notified when video recording starts.

VCAP\_SUE\_STOP\_VIDEO\_RECORDING  
 It will be notified when video recording stops.

VCAP\_SUE\_START\_PHOTO It will be notified when photo capturing starts.

VCAP\_SUE\_END\_PHOTO It will be notified when photo capturing ends.

**Remarks** Enqueued when the Video Capture Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# 

# Chapter 40

# Individual Recognition

This Chapter defines the Individual Recognition device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.16 | Open |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | Open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | 1.16 | open |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.16 | open |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | Not Supported |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **CapIndividualList:** | *string* | {read-only} | 1.16 | open |
| **IndividualIDs:** | *string* | {read-only} | 1.16 | open, claim & enable |
| **IndividualRecognitionFilter:** | *string* | {read-write} | 1.16 | open |
| **IndividualRecognitionInformation** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Methods (UML operations)** | | |  | | |
| ***Common*** | | |  | | |
| ***Name*** | | | ***Version*** | | |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | | | 1.16 | | |
| **close ( ): void {raises-exception, use after open}** | | | 1.16 | | |
| **claim (timeout:** *int32***): void {raises-exception, use after open}** | | | 1.16 | | |
| **release ( ): void {raises-exception, use after open, claim}** | | | 1.16 | | |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | | | 1.16 | | |
| **clearInput ( ): void { }** | | | 1.16 | | |
| **clearInputProperties ( ): void { }** | | | 1.16 | | |
| **clearOutput ( ): void { }** | | | Not  supported | | |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, enable}** | | | 1.16 | | |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | | | 1.16 | | |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | | | 1.16 | | |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | | | 1.16 | | |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, enable}** | | | 1.16 | | |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, enable}** | | | 1.16 | | |
| **Events (UML interfaces)** |  |  | |  |
| ***Name*** | ***Type*** | ***Mutability*** | | ***Version*** |
|  |  |  | |  |
| **upos::events::DataEvent** |  |  | | 1.16 |
| **Status:** | *int32* | {read-only} | |  |
|  |  |  | |  |
| **upos::events::DirectIOEvent** |  |  | | 1.16 |
| **EventNumber:** | *int32* | {read-only} | |  |
| **Data:** | *int32* | {read-write} | |  |
| **Obj:** | *object* | {read-write} | |  |
|  |  |  | |  |
| **upos::events::ErrorEvent** |  |  | | 1.16 |
| **ErrorCode:** | *int32* | {read-only} | |  |
| **ErrorCodeExtended:** | *int32* | {read-only} | |  |
| **ErrorLocus:** | *int32* | {read-only} | |  |
| **\*pErrorResponse:** | *int32* | {read-write} | |  |
|  |  |  | |  |
| **upos::events::OutputCompleteEvent** |  | ***Not Supported*** | | 1.16 |
|  |  |  | |  |
| **upos::events::StatusUpdateEvent** |  |  | | 1.16 |
| **Status:** | *int32* | {read-only} | |  |

# General Information

The Individual Recognition programmatic name is “Individual Recognition”.

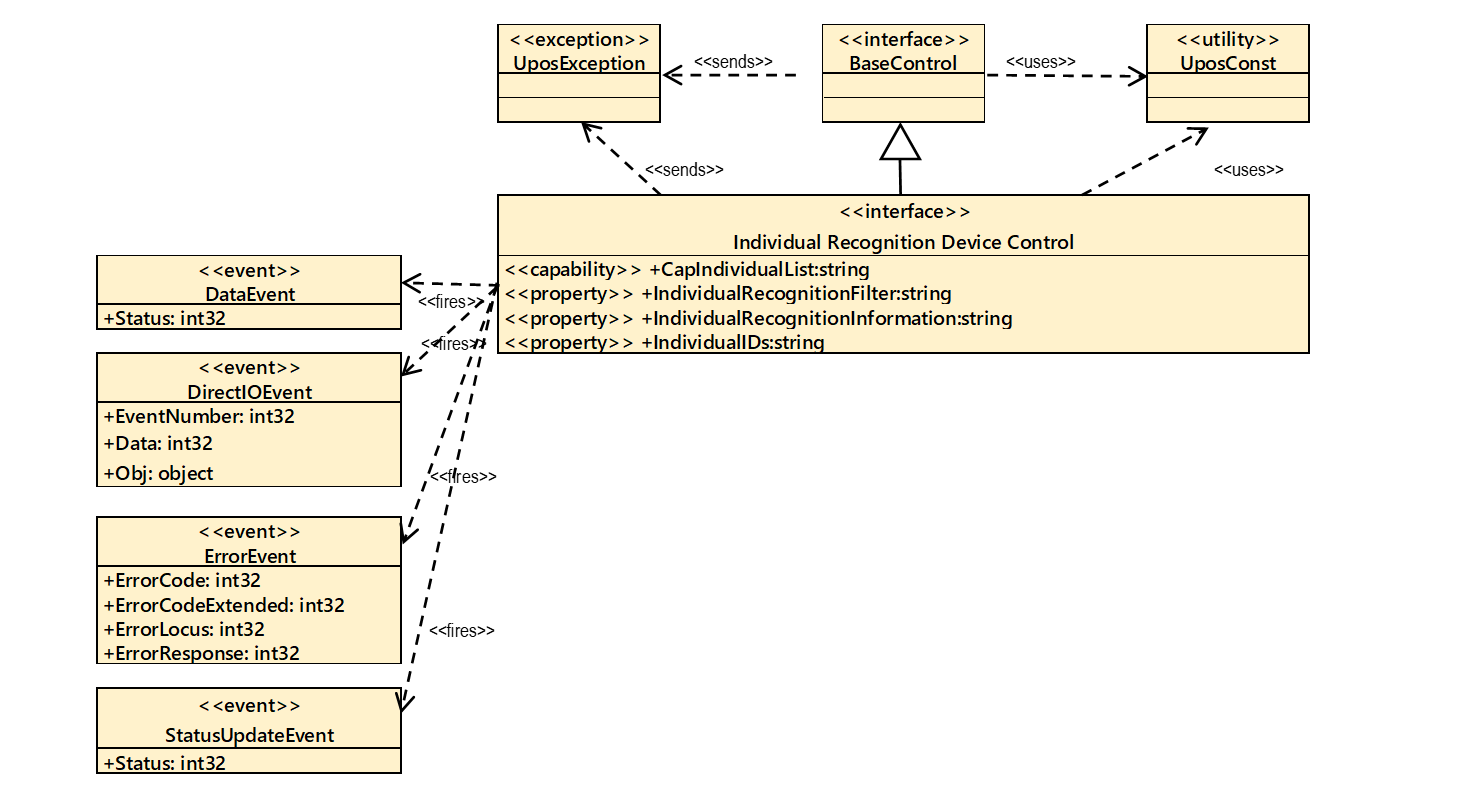
## **Capabilities**

The Individual Recognition has the following set of capabilities:

Analyzes the image of the camera and recognizes individuals such as people and listed goods.

## **Individual Recognition Class Diagram**

The following diagram shows the relationships between the Individual Recognition classes.



**Fig.** Chap.40-1 Individual Recognition Class Diagram

# Model

The Individual Recognition follows the general “Device Input Model” for event-driven input:

## **Input Model**

* When an individual is recognized by this device, a **DataEvent** is delivered to the application after the **IndividualIDs** property was set to indicate the recognized individuals.
* Identifiable individuals are indicated by the **CapIndividualList** property.
* Check the functions supported by the device, set validity / invalidity, etc. with the **IndividualRecognitionInformation** property.
* Recognized data is stored in the **IndividualRecognitionInformatio**n property, **IndividualIDs** property.
* How to recognize the individuals depends on the IndividualRecognitionFilter function therefore, please refer to the IndividualRecognitionFiler section.
* Other device behavior about this device supports the general device input model as listed below.
* If the **AutoDisable** property is true, then the device automatically disables itself when a **DataEvent** is enqueued.
* An enqueued **DataEvent** can be delivered to the application when the **DataEventEnabled** property is true and other event delivery requirements are met. Just before delivering this event, data is copied into corresponding properties, and further data events are disabled by setting **DataEventEnabled** to false. This causes subsequent input data to be enqueued while the application processes the current input and associated properties. When the application has finished processing the current input and is ready for more data, it reenables events by setting **DataEventEnabled** to true.
* An **ErrorEvent** (or events) is enqueued if an error occurs while gathering or processing input, and is delivered to the application when **DataEventEnabled** is true and other event delivery requirements are met.
* The **DataCount** property may be read to obtain the total number of enqueued **DataEvent**s.
* All enqueued input may be deleted by calling **clearInput** method. See the **clearInput** method description for more details.
* All data properties that are populated as a result of firing a **DataEvent** or **ErrorEvent** can be set back to their default values by calling the **clearInputProperties** method.
* The application will be informed about any status change with a **StatusUpdateEvent,** also all corresponding status properties will be updated before event delivery.

## **Device Sharing**

The Individual Recognition 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 the device begins reading input.
* See the “Summary” table for precise usage prerequisites.

## **IndividualRecognitionFilter**

The **IndividualRecognitionFilter** property defines the following data as information for the individual recognition function of Individual Recognition Device.

･ Various support function existence or not.   
 (Supported functions are defined by the device)

･ Enable, disable status of various functions.

･ Types handled by various functions (examples: “male”, “female” of gender recognition)

･ Filter setting of various functions.

The following data is defined in the IndividualRecognitionInformation property

･ Individual Recognition input data

The device defines the individual recognition function information and the individual recognition input data.

The application refers to these contents to determine the support range and so on.

In addition, the application changes the enabled / disabled state of various functions, the filter setting, and controls each function.

The enabled / disabled state of the various functions set by the application, and the filter settings are applied by setting the DeviceEnabled property to true and enabling the individual recognition function.

When the application set various functions, it is possible to specify and set only the target ones.

The device fires a DataEvent based on the content set by the application and stores the input data in **IndividualRecognitionInformation** property.

## **IndividualRecognitionFilter Property Example Format**

The IndividualRecognitionFilter property of the individual recognition device may define various information. Here is the example described by using the JSON format.

* Basic Items

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | | | | Value | Value change capability | Explanation |
| IndividualRecognitionFilter | | | | object | N | Information for the various individual recognition. Target device define the supporting individual recognition object. |
|  | [IndividualRecognitionID] | | | object | N | Recognizable individual recognition information. Key name is the ID of recognized individual |
|  | Enabled | | boolean | Y | Enable or disable state of target individual recognition.  Application can control the target individual recognition by referring or changing. |
|  | Properties | | object | N | Property information of the target individual recognition.  Application control the target individual recognition by referring or changing the defined property value. |
|  |  | [Property01] | - | - |  |
|  |  | [Property02] | - | - |  |
|  | Filters | | object | N | Input data filter setting information. Application filter the target individual recognition input data by changing the defined value. |
|  |  | [Filter01] | - | - |  |
|  |  | [Filter02] | - | - |  |

* Face Recognition device example

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Key | | | | | | Value | Value change capability | Explanation |
| IndividualRecognitionFilter | | | | | | object | N |  |
|  | Face | | | | | object | N |  |
|  | Enabled | | | | boolean | Y |  |
| Properties | | | | object | N |  |
|  | FaceImageNamePrefix | | | string | Y | Output image file prefix for face recognition |
| Gender | | | object | N | Information on gender recognition |
|  | Enabled | | boolean | Y | Gender recognition enable, disable state |
| CapTypeList | | array | N | Type list (“female”, “male”) |
| Age | | | object | N | Information on age recognition. |
|  | Enabled | | boolean | Y | Age recognition enable, disable state |
| Facial Expression | | | object | N | Information on facial expression recognition |
|  | Enabled | | boolean | Y | Facial expression recognition enable, disable state. |
| CapTypeList | | array | N | Type list (“smile”, “angry”,…) |
| Gaze | | | object | N | Information on gaze recognition |
|  | Enabled | | boolean | Y | Gaze recognition enable, disable state. |
| CapTypeList | | array | N | Type list (“gaze”, “nogaze”) |
| Distance | | | object | N | Information on distance recognition |
|  | Enabled | | boolean | Y | Distance recognition enable, disable state |
| CapTypeList | | array | N | Type list (“near”, “far”, “very far”,…) |
| NearLength | | number | Y | Distance to recognize as “near”. A recognition event is fired when a person is recognized in the range from 0 to Near Length. |
| FarLength | | number | Y | Distance to recognize as “far”, “very far”. A recognition event is fired when a person is recognized in the range from Near Length to Far Length. A recognition event is fired when a person is recognized in the range more than Far Length. |
| Authentication | | | object | N | Information on face authentication |
|  | Enabled | | boolean | Y | Face authentication enable, disable state. |
| ImageList | | array | Y | Image file name list for comparison. Event is fired when it matches the image specified here. (Wild card can be specified) |
| Filters | | | | object | N |  |
|  | Gender | | | object | N | Information on gender recognition filter. |
|  | TypeList | | array | Y | Target Filter TypeList. Valid values are defined by  CapTypeList. Recognition target is specified. To disable the filter, null should be assigned in its value. |
| Score | | number | Y | Recognition score. Valid values are from 0 to 100. The range of the score specified here is the recognition target. To disable the filter, -1 should be assigned in its value. |
| Age | | | object | N | Information on age recognition. |
|  | | Min | number | Y | Minimum age. The age below the specified is not a recognition target. To disable the filter -1 should be specified in its value. |
| Max | number | Y | Maximum age. The age above the specified is not a recognition target. To disable the filter -1 should be specified in its value. |
| Expression | | | object | N | Information on facial expression recognition filter. |
|  | | TypeList | array | Y | Filter target type list. Valid values are defined in CapTypeList.  Recognition target type is specified. To disable the filter null should be assigned in its value. |
| Score | number | Y | Recognition score. Valid values are from 0 to 100. The range of the score specified here is to be recognized. To disable the filter -1 should be assigned in its value. |
| Gaze | | | object | N | Information on gaze recognition filter |
|  | | TypeList | array | Y | Filter target type list. Valid values are defined by CapTypeList.  Recognition target is specified. To disable the filter, null should be assigned in its value. |
| Distance | | | object | N | Information on distance recognition filter |
|  | | TypeList | array | Y | Filter target type list. Valid values are defined by CapTypeList.  Recognition target is specified. To disable the filter, null should be assigned in its value. |

## **IndividualRecognition Information Property Example Format**

IndividualRecognitionInformation property of individual recognition device may define various information and here is the example format described by JSON.

* Basic Items

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Key | | | | Value | Value change capability | Explanation |
| IndividualRecognitionInformation | | | | object | N | Various Individual recognition input data. |
|  | [IndividualRecognitionID] | | | object | N | Store the input data of individual recognition.  Key name is ID of individual recognition. |
|  |  | Properties | | Array <object> | N | Input data list of target individual recognition.  The content of the data is different for each device or function. |
|  |  | [Data01] | - | - |  |
|  |  | [Data02] | - | - |  |

* Face Recognition Device Example

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Key | | | | | Value | Value change capability | Explanation |
| IndividualRecognitionInformation | | | | | object | N |  |
|  | Face | | | | object | N |  |
|  | DataLists | | | array  <object> | N |  |
|  | FaceID | | string | N | ID assigned to the recognized face |
| FaceImageName | | string | N | Recognized face image file name |
| Gender | | object | N | Recognized gender information |
|  | Type | string | N | Recognized type |
| Score | number | N | Confidence score of recognized type. |
| Age | | object | N | Recognized age information |
|  | Age | number | N | Recognized age |
| Expression | | object | N | Recognized facial expression information |
|  | Type | string | N | Recognized type.  One of CapTypeList items is set. |
| Score | number | N | Confidence score of recognized type. |
| Gaze | | object | N | A gaze list for each recognized face ID. |
|  | Type | string | N | Recognized type |
| Distance | | object | N | Recognized distance information |
|  | Type | string | N | Recognized type.  One of CapTypeList items is set. |
| Authentication | | object | N | Authentication result information |
|  | ImageName | string | N | Matched image file name |

# Properties (UML attributes)

## **CapIndividualList Property**

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

**Remarks** Recognizable Individual information is indicated by the list separated by a separator ",".

Each Individual information consists of the following information and is shown in the following order, separated with a colon (":").

**Parameter Meaning**

IndividualID An ID indicated an identifiable Individual

IndividualName A Name of an Individual.

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 Intro-20.

**See Also** **IndividualIDs** Property.

## **IndividualIDs Property**

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

**Remarks** Set the IndividualIDs recognizable Individual recognition device.

IndividualIDs values are indicated by separated with a colon (":").

Its value is set prior to a **DataEvent** being delivered to the application.

**Errors** A UposException may be thrown when this property is accessed.  
 For further information, see “**Errors**" on page Intro-20.

**See Also** **CapIndividualList** Property

## **IndividualRecognitionFilter Property**

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

**Remarks** Individual Recognition Function Information:

* Supporting the various functions (Refer to the Individual Recognition Filter Example Format written by JSON and supported function examples.).
* Various Valid / Invalid State functions.
* Various handled function types (e.g., "male" "female" in gender recognition, etc.).
* Various filter function settings.

All Individual Recognition function data information is defined by the device. By referring to these contents, the application can determine the supporting scope. Thereby, the application can control each function by changing the valid / invalid state and / or the various filter function settings.

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 Intro-20.

## **IndividualRecognitionInformation Property**

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

**Remarks** Holds data indicating the following. Individual recognition input data. All Individual recognition input data is defined by the device.

**Errors** A UposException may be thrown when this property is accessed.  
 For further information, see “**Errors**" on page Intro-20.

# Events (UML interfaces)

## **DataEvent**

**<<event>> upos::events::DataEvent**

**Status : *int32*{read-only}**

**Description** Notifies the application when data from the Individual Recognition device is available to be read.

**Attributes** This event contains the following attributes:

**Attribute Type Description**

*Status int32 Set to 0.*

**Remarks** Before this event is delivered, the individual recognition information is enqueued into the area that is indicated by the **readValue** method. Since the stored individual recognition information might be managed by the associated “**Hard Totals**” device service, therefore, the application might also support the “**Hard Totals**” service.

**See Also** **VideoCaptureMode** Property, **“Events"** on page Intro-19, **readFrame** Method , **directIO** method.

## **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 Individual Recognition Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **"Events"** on page Intro-19, **directIO** method.

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent**

**ErrorCode : *int32* {read-only}  
ErrorCodeExtended : *int32* {read-only}  
ErrorLocus : *int32* {read-only}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that an Individual Recognition Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is re-tried, and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is re-tried, and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period of time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
Status : *int32* {read-only}**

**Description** *Notifies the application that there is a change in the power status or a status*

o*f the Individual Recognition device.*

**Attributes** This event contains the following attribute:

**Attribute Type Description**

*Status int32* Indicates a change in the power status of the unit.

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values

for communicating the status/progress of an asynchronous update firmware

process. See “**StatusUpdateEvent**” description on page 1-34.

**Remarks** Enqueued when the Individual Recognition Device detects a power state

change or a status change.

**See Also** **“Events"** on page Intro-19

# Chapter 41

# Sound Recorder

This Chapter defines the Sound Recorder device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties(UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.16 | open |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | 1.16 | open |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.16 | open |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | Not Supported |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | open |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **CapAssociatedHardTotalsDevice:** | *string* | {read-only} | 1.16 | open |
| **CapChannel:** | *boolean* | {read-only} | 1.16 | open |
| **CapSamplingRate:** | *boolean* | {read-only} | 1.16 | open |
| **CapSoundType:** | *boolean* | {read-only} | 1.16 | open |
| **CapStorageTarget** | *boolean* | {read-only} | 1.16 | open |
| **CapRecordingLevel:** | *boolean* | {read-only} | 1.16 | open |
| **Channel:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **ChannelList:** | *string* | {read-only} | 1.16 | open |
| **RecordingLevel:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **RemainingRecordingTimeInSec:** | *int32* | {read-only} | 1.16 | open, claim & enable |
| **SamplingRate:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **SamplingRateList:** | *string* | {read-only} | 1.16 | open |
| **SoundData:** | *binary* | {read-only} | 1.16 | open |
| **SoundType:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **SoundTypeList:** | *string* | {read-only} | 1.16 | open |
| **StorageTarget** | *int32* | {read-write} | 1.16 | open, claim & enable |

|  |  |
| --- | --- |
| **Methods(UML operations)** |  |
| ***Common*** |  |
| ***Name*** | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | 1.16 |
| **close ( ): void {raises-exception, use after open}** | 1.16 |
| **claim (timeout:** *int32***): void {raises-exception, use after open}** | 1.16 |
| **release ( ): void {raises-exception, use after open, claim}** | 1.16 |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | 1.16 |
| **clearInput ( ): void { }** | 1.16 |
| **Me Methods (UML operations)(continued)** |  |
| ***Common*** |  |
| ***Name*** | ***Version*** |
| **clearInputProperties ( ): void { }** | 1.16 |
| **clearOutput ( ): void { }** | *Not*  *supported* |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, enable}** | 1.16 |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | 1.16 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | 1.16 |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | 1.16 |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, enable}** | 1.16 |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, enable}** | 1.16 |
|  |  |
| ***Specific*** |  |
| ***Name*** | ***Version*** |
| **startRecording (FileName:** *string***, OverWrite:** *boolean***, RecordingTime:***int32***):**  **void {raises-exception, use after open, claim, enable}** | 1.16 |
| **stopRecording ( ):**  **Void {raises-exception, use after open, claim, enable}** | 1.16 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
|  |  |  |  |
| **upos::events::DataEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.16 |
| **EventNumber:** | *int32* | {read-only} |  |
| **Data:** | *int32* | {read-write} |  |
| **Obj:** | *object* | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  |  | 1.16 |
| **ErrorCode:** | *int32* | {read-only} |  |
| **ErrorCodeExtended:** | *int32* | {read-only} |  |
| **ErrorLocus:** | *int32* | {read-only} |  |
| **\*pErrorResponse:** | *int32* | {read-write} |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  | ***Not Supported*** | 1.16 |
|  |  |  |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |
| **upos::events::TransitionEvent** |  | ***Not Supported*** | 1.16 |

# General Information

The Sound Recorder programmatic name is "Sound Recorder".

## **Capabilities**

The Sound Recorder has the following capability:

Record the real-time audio to a file, deliver the recorded sound data to the property that application may read and / or retrieve, and save the recorded sound data file to device memory and / or other storage devices.

## **Sound Recorder Class Diagram**

　　The following diagram shows the relationships between the Sound Recorder classes.



Fig. Chap. 41-1 Sound Recorder Class Diagram

# Model

Sound Recorder Control follows a general “Device Input Model” in a broad sense. One point of difference is that the Sound Recorder device required the execution of methods to start and stop the sound recording process and creates a sound data file in real time, deliver the data to the property and save the file in device and / or associated storage device.

The Sound Recorder Model defines the following behavior: Sound Recorder device controls the Sound Recorder device to set the input (recording) conditions, specifies the start / end of input data acquisition by the method. And makes the sound data file in real time from the acquired audio and delivers the data to the appropriate property. At the same time, saves the recorded data file in device and /or associated storage devices.

“Sound Recorder” device control starts recording with the **startRecording** method. Prior to execute the **startRecording** method each value setting of **Channe**l property, **SamplingRate** property, and **RecordingLevel** property are required, if each of **CapChannel** property **CapSamplingRate** property is true. And also need to set the **DataEventEnabled** property to true. At the same time, the recording format setting starts with the **SoundType** property value, if **CapSoundType** property is true.

The recording ends after the specified time has elapsed or when **stopRecording** method is called or when **clearInput** method is called. The generated sound data file will be recorded for either the host file or the HardTotals device or both, after the end of recording. And generated sound data will be delivered to the **SoundData** property. Just after the delivery of sound data to the property, when the DataEventEnabled property is true, the **DataEvent** is enqueued and delivered to the application.

If the **AutoDisable** property is true, the device will automatically disable itself after the **DataEvent** is enqueued.

The remaining recording time in seconds can be obtained from the property **RemainingRecordingTimeInSec**

**StatusUpdateEvent** with status SERC\_SUE\_START\_SOUND\_RECORDING is evoked when **startRecording** method is executed to notify the application that recording state with has started.

When the sound recording is finished, if the specified time of **startRecording** method has elapsed or **stopRecording** method has been called, a **StatusUpdateEvent** with status SERC\_SUE\_STOP\_SOUND\_RECORDING is evoked to notify the application that recording has been stopped.

An enqueued **DataEvent** can be delivered to the application when the **DataEventEnabled** property is true and other event delivery requirements are met. Just before enqueuing this event, the device provides the recorded data to the **SoundData** property, and disables further data events by setting the **DataEventEnabled** property to false. This causes subsequent input data to be buffered by the device while the application processes the current input and associated properties. When the application has finished processing the current input and is ready for more data, it re-enables events by setting **DataEventEnabled** to true.

If **ErrorEvent** response is ER\_CONTINUEINPUT, the process of input processing continues. However, as long as the cause of the error is not resolved, the **ErrorEvent** will occur again immediately.

If **ErrorEvent** is ER\_CLEAR, the input processing process is terminated, and the record is discarded.

If the time specified by the **startRecording** method is FOREVER (-1), execution will continue until the **stopRecording** method is called in the application. When **stopRecording** is called, the previous recording data is recorded to the host file, the HardTotals device~~,~~ or both, with the specified file name, and the sound data will be delivered to the **SoundData** property. When DataEventEnabled property is true, the **DataEvent** is enqueued and delivered to the application.

Only one call to **startRecording** method can be in progress at a time. An attempt to nest sound recorder operations will result in an **UposException** being thrown.

If Error occurs during the execution of the **startRecording** method application should call the stopRecording method to terminate the recording process or cancel the recording process by calling the **clearInput** method before ending the **ErrorEvent** processing. After this when the **stopRecording** method is called, the recording data until just before the **ErrorEvent** occurs is recorded to the host file, the HardTotals device, or both. When **DataEventEnabled** property is true, the **DataEvent** is enqueued and delivered to the application.

If there is no Error during the execution of **startRecording** method can terminate the recording process and can stop the recording at any time. When the **stopRecording** method is called, the recording data until just before the method call is recorded to the host file, the HardTotals or both. When **DataEventEnabled** property is true, the **DataEvent** is enqueued and delivered to the application.

All input data enqueued by the device may be deleted by calling the **clearInput** method．All data properties that are populated as a result of a **DataEvent** or **Error Event** can be set back to their default values by calling the **clearInputProperties** method.

The device may have the ability to write encoded sound data files to either the HardTotals devices or the host file system, or both, and the **CapStorageTarget** property will show the device’s data storage location capability.

If device supports either or both HardTotals devices and the host file system, the application should set the **StorageTarget** property accordingly to tell where to write the encoded sound data file.

If device needs to be able to write the encoded sound data to an associated HardTotals device, the **CapAssociatedHardTotalsDevice** property holds the open name of the associated HardTotals device.

## **Device Sharing**

The Sound Recorder 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 accessing some properties or calling methods that update the device.
* See the “Summary” table for precise usage prerequisites.
* The image display mode of the graphics device control is as follows.

# Properties(UML attributes)

## **CapAssociatedHardTotalsDevice Property**

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

**Remarks** Holds the open name of the associated HardTotals device if the device is able to write to such devices which is the case if **CapStorageTarget** is either CST\_ALL or CST\_HARDTOTALS\_ONLY. If **CapStorageTarget** is CST\_HOST\_ONLY this property value must be the empty string.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapStorageTarget** Property

## **CapChannel Property**

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

**Remarks** If true, the application can change the channel.  
 If false, the application cannot change the channel.  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **Channel** Property

## **CapSamplingRate Property**

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

**Remarks** If true, the application can change the sampling rate.  
If false, the application cannot change the sampling rate.  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
For further information, see “Errors" on page Intro-20.

**See Also** **SamplingRate** Property.

## **CapSoundType Property**

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

**Remarks** If true, the application can change the sound file type.  
If false, the application cannot change the sound file type.  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **SoundType** Property.

## **CapStorageTarget Property**

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

**Remarks** This is an enumeration and announces where the device is able to write the recorded sound data file to.  
It holds one of the following values.

**Value Meaning**

CST\_HARDTOTAL\_ONLY Only an associate **HardTotals** device is supported.

CST\_HOST\_ONLY Only the host’s file system is supported.

CST\_ALL Both, the associated **HardTotals** device and the host’s file system is supported.

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

If a HardTotals device is supported the **StorageTarget** the property value should be CST\_HARDTOTAL\_ONLY or CST\_ALL, and the property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **StorageTarget** Property, **CapAssociatedHardTotalsDevice** Property

## **CapRecordingLevel Property**

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

**Remarks** If true, the application can change the recording level.  
If false, the application cannot change the recording level.  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapRecordingLevel** Property.

## **Channel Property**

**Syntax Channel : *string* {read-write, access after open-claim-enable}**

**Remarks** Holds the channel during recording.  
Valid values are one of the values listed in the **ChannelList** property.  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapChannel** Property, **ChannelList** Property

## **ChannelList Property**

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

**Remarks** Contains the comma-delimited list of channels that is supported by the device.

For example, if the device only supports channel1and channel2 and channel4, then this property should be set to "1,2,4".  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **Channel** Property.

## **RecordingLevel Property**

**Syntax RecordingLevel : *int32* {read-write, access after open- claim-enable}**

**Remarks** Holds the recording level during recording.  
Legal values range from zero through 100.  
This property is initialized by theopen **method.**

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **CapRecordingLevel** Property

## **RemainingRecordingTimeInSec Property**

**Syntax RemainingRceordingTimeInSec: *int32* {read-only, access after open-claim-enable}**

**Remarks** This property holds the remaining recording time in seconds if a recording is ongoing. If no recording is ongoing its value is 0. When a call to method **startRecording** returns, this property initially holds the time passed as argument *recordingTime* to that call. If this argument value is FOREVER, this property also holds this value unchanged until **stopRecording** has been called.

This property is initialized during device **setDeviceEnbaled** method to 0.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also startRecording** Method**, stopRecording** Method

## **SamplingRate Property**

**Syntax SamplingRate : *string*{read-write, access after open-claim-enable}**

**Remarks** Holds the sampling rate during recording.  
Valid values are one of the values listed in the **SamplingRateList** property.  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapSamplingRate** Property, **SamplingRateList** Property

## **SamplingRateList Property**

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

**Remarks** Contains the comma-delimited list of sampling rate that are supported by the device.  
For example, if the device only supports 44.1kHz and 48kHz and 96kHz, then this property should be set to "44100,48000,96000".  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **SamplingRate** Property.

## **SoundData Property**

**Syntax 　SoundData: *binary* { read-only, access after open }**

**Remarks** This property is used to store the sound data after the recording time elapse of startRecording method or stopRecording method is called. If no recorded sound data was available, the **SoundData** property will be set to zero length (or empty). Its value is set prior to a **DataEvent** to be enqueued**.** This property is initialized to zero length by the **open** method.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-21.

**See Also startRecording** Method, **stopRecording** Method, **DataEvent**.

## **SoundType Property**

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

**Remarks** Holds the audio file format to be recorded.  
Valid values are one of the values listed in the **CapSoundTypeList** property.  
**This property is initialized by the** open **method**.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapSoundType** Property, **CapSoundTypeList** Property.

## **SoundTypeList Property**

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

**Remarks** Contains the comma-delimited list of sound file type that is supported by the device.  
For example, if the device only supports WAV and OGG, then this property should be set to "WAV,OGG".  
This property is initialized by the **open** method.

**Errors** UposException may be thrown when this property is accessed.  
For further information, see “Errors" on page Intro-20.

**See Also** **SoundType** Property.

## **StorageTarget Property**

**Syntax StorageTarget: *int32* {read-write, access after open-claim-enable}**

**Remarks** This is an enumeration and defines where the device writes the recorded sound data file to. Should be set before a call to **startRecording**.  
It holds one of the following values.

**Value Meaning**

ST\_HARDTOTAL The encoded data file is written to the associated HardTotals device. The property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

ST\_HOST The encoded data file is written to the host’s file system.

ST\_HOST&HARDTOTAL

The encoded data file is written to the associated **HardTotals** device and host’s file system. The property **CapAssociatedHardTotalsDevice** holds the open name of the associated HardTotals device.

This property is initialized by the **open** method according to the value hold by **CapStorageTarget**. If **CapStorageTarget** has the value CST\_ALL it is initialized to ST\_HOST.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**Value Meaning**

E\_ILLEGAL An invalid value was specified, or recording is ongoing.

**See Also CapStorageTarget** Property, **CapAssociatedHardTotalsDevice** Property

# Methods(UML operations)

## **startRecording Method**

**Syntax** **startRecording (fileName:** *string***, overWrite:** *boolean***, recordingTime :** *int32***):  
 void{raises-exception, use after open-claim-enable}**

**Parameter Description**

*fileName* Specify the file name of the sound to be recorded.

*overWrite* Specify the behavior when the same name file exists. If it is true it will be overwritten and if false it will raise the UPOSException.

*recordingTime* Specify the time for recording in seconds. If OPOS\_FOREVER (-1) is specified, recording will continue until the **stopRecording** method is called.

**Remarks** Sound recording starts with the settings of the **Channel** property, **SamplingRate** property, and **RecordingLevel** property and need to set DataEventEnabled property to true. At the same time, recording format setting starts with the **SoundType** property. When this method is called, if specified recording time is elapsed, recording process will be ended and recorded sound data is provided at the **SoundData** property that the application may read it and / or process the stored sound data file given as *filename* argument. When the DataEventEnabled property is true, the **DataEvent** is enqueued and delivered to the application. **StatusUpdateEvent** with state SREC\_SUE\_START\_SOUND\_RECORDING is evoked when **startRecording** method is executed to notify the application, the recording has started. When the sound recording is finished, if the specified time of **startRecording** method has elapsed or **stopRecording** method has been called, the value of **StatusUpDateEvent** with state SREC\_SUE\_STOP\_SOUND\_RECORDING is evoked to notify the application, the recording has stopped

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20

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

**Value Meaning**

E\_ILLEGAL FileName is too long or contains characters that cannot be used, or 0 is specified for RecordingTime.

E\_EXISTS FileName already exists. (When OverWrite is FALSE)

E\_BUSY It cannot be executed as it is recording.

**See Also** **Channel** Property, **SamplingRate** Property, **SoundData** Property, **SoundType** Property, **RecordingLevel** Property, **stopRecording** Method, **StatusUpdateEvent** Event

## **stopRecording Method**

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

**Remarks** When this method is called the sound recording process that started by **startRecording** method is ended and the recording is finished. This method is processed synchronously. After recording and decoding process has been finished, the recorded sound data will be provided at the **SoundData** property prior to the Data Event is enqueued, when DataEventEnabled property is true. When **stopRecording** method is called, a **StatusUpdateEvent** with status SREC\_SUE\_STOP\_SOUND\_RECORDING is evoked to notify the application, the recording has stopped.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20 Some possible values of the exception’s *ErrorCode* property are:

**Value Meaning**

E\_ILLEGAL It is not recorded.

**See Also** **StartRecording** Property, **SoundData** Property, **StatusUpdateEvent**. Event

# Events(UML interfaces)

## **DataEvent**

**<<event>> upos::events::DataEvent**

**Status :*int32*{read-only}**

**Description** Notifies the application when data from the Sound Recorder device is available to be read.

**Attributes** This event contains the following attributes:

**Attribute Type Description**

*Status int32 Set to 0.*

**Remarks** Before this event is delivered, the Sound Recorder information is enqueued into the area that is indicated by the startRecording method. Since the stored sound recorder device information might be managed by the associated “Hard Totals” device service, therefore, the application might also support the “Hard Totals” service.

**See Also Channel** Property, **SamplingRate** Property, **SoundType** property, **RecordingLevel** Property, **stopRecording** Method, **startRecording** Method

## **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 Individual Recognition Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **"Events"** on page Intro-19, **directIO** method.

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent  
ErrorCode : *int32*{read-write}  
ErrorCodeExtended : *int32*{read-write}  
ErrorLocus : *int32*{read-write}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that a Sound Recorder Device error has been detected and suitable response by the application is necessary to process the error condition.

**Attributes** This event contains following attributes.

**Attributes Type Description**

*Error Code int32* Error Code causing the error event. See the list of Error Code.

*ErrortCodeExtended int32* Error Code causing the error event. These values are device category specific.

*ErrorLocus* *int32* Location of the error. See values below.

*pErrorResponse* *int32* Error response, whose default value may be overridden by the application (i.e., this attribute is settable). See values below.

*The ErrorLocus attribute has one of the following values:*

**Value Meaning**

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.

*If ResultCode is E\_EXTENDED,* ***ResultCodeExtended*** *is set to one of the following values.*

**Value Meaning**

ETOT\_NOROOM There is not enough space to create the file.

*The application’s error event handler can set the ErrorResponse attribute to one of the following values:*

**Value Meaning**

ER\_CLEAR I will try its asynchronous output again.

The error condition is exited.

ER\_CONTINUEINPUT

Only valid when the locus is EL\_INPUT\_DATA.

Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional DataEvents as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEven**t is delivered with locus EL\_INPUT. This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** It notifies you when an error is detected during recording. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

**See Also** **Status, Error code, State model**

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
Status : *int32* {read-only}**

**Description** *Notifies the application that there is a change in the power status or a status of the Sound Recorder device.*

**Attributes**  This event contains the following attribute:

**Attributes Type Description**

*Status* *int32* Indicates a change in the power status or a status of the unit.

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process. See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

SREC\_SUE\_START\_SOUND\_RECORDING It will be notified when sound recording starts.

SREC\_SUE\_STOP\_SOUND\_RECORDING It will be notified when sound recording stops.

**Remarks** Enqueued when the Sound Recorder Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# Chapter 42

# Voice Recognition

This Chapter defines the Voice Recognition device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.16 | open |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | 1.16 | open |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.16 | open |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | Not Supported |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  | |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** | |
| **CapLanguage:** | *boolean* | {read-only} | 1.16 | open | |
| **HearingDataPattern:** | *string* | {read-only} | 1.16 | open, claim & enable | |
| **HearingDataWord:** | *string* | {read-only} | 1.16 | open, claim & enable | |
| **HearingDataWordList:** | *string* | {read-only} | 1.16 | open, claim & enable | |
| **HearingResult:** | *int32* | {read-only} | 1.16 | open, claim & enable | |
| **HearingStatus:** | *int32* | {read-only} | 1.16 | open, claim & enable | |
| **LanguageList:** | *string* | {read-only} | 1.16 | open | |
|  |  |  |  |  | |
| **Me** **Methods (UML operations)** | | | | |  |
| ***Common*** | | | | |  |
| ***Name*** | | | | | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | | | | | 1.16 |
| **close ( ): void {raises-exception, use after open}** | | | | | 1.16 |
| **claim ( timeout:** *int32* **): void {raises-exception, use after open}** | | | | | 1.16 |
| **release ( ): void {raises-exception, use after open, claim}** | | | | | 1.16 |
| **checkHealth ( level:** *int32* **): void {raises-exception, use after open, enable}** | | | | | 1.16 |
| **clearInput ( ): void { }** | | | | | 1.16 |
| **clearInputProperties ( ): void { }** | | | | | 1.16 |
| **clearOutput ( ): void { }** | | | | | Not  supported |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | | | | | 1.16 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **retrieveStatistics ( inout statisticsBuffer:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **Me Methods (UML operations)(continued)** | | | | |  |
| ***Common*** | | | | |  |
| ***Name*** | | | | | ***Version*** |
| **updateFirmware ( firmwareFileName:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **updateStatistics ( statisticsBuffer:** *string***):** **void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| ***Specific*** | | | | |  |
| ***Name*** | | | | |  |
| **startHearingFree (language:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **startHearingSentence (language:** *string*, **wordList:** *string,* **patternList:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **startHearingWord (language:** *string*, **wordList:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **startHearingYesNo (language:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **stopHearing (): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
|  |  |  |  |
| **upos::events::DataEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.16 |
| **EventNumber:** | *int32* | {read-only} |  |
| **Data:** | *int32* | {read-write} |  |
| **Obj:** | *object* | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  |  | 1.16 |
| **ErrorCode:** | *int32* | {read-only} |  |
| **ErrorCodeExtended:** | *int32* | {read-only} |  |
| **ErrorLocus:** | *int32* | {read-only} |  |
| **\*pErrorResponse:** | *int32* | {read-write} |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |

# General Information

The Voice Recognition programmatic name is "Voice Recognition".

## **Capabilities**

The Voice Recognition has the following capability:

* Convert spoken words to strings.

## **Voice Recognition Class Diagram**

The following diagram shows the relationships between the Voice Recognition classes.

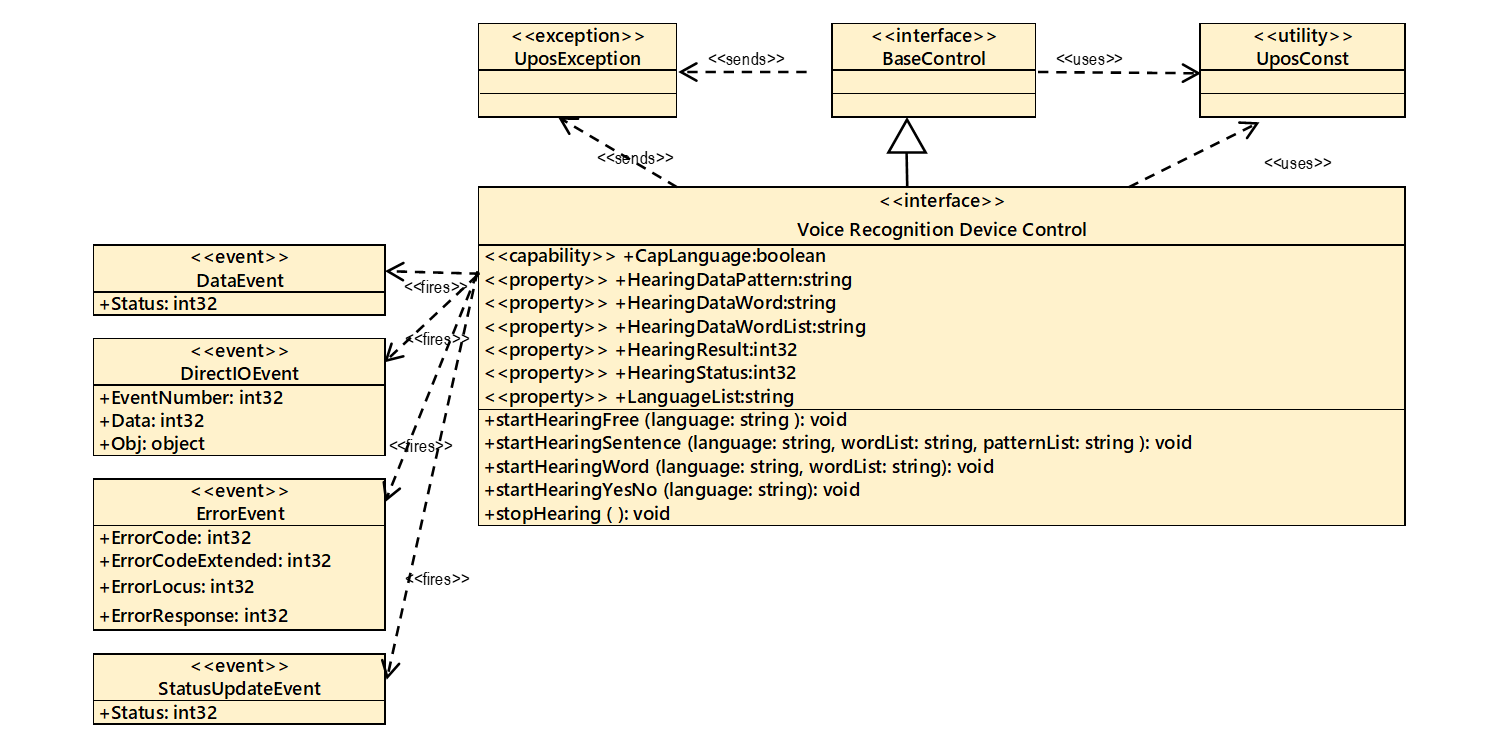


Fig. Chap. 42-1 Voice Recognition Class Diagram

# Model

The Voice Recognition follows the general “Device Input Model” for event-driven input:

Device control starts voice recognition with the **startHearingYesNo** method, **startHearingSentence** method, etc., and generates **DataEvent** when recognizing voice.

If the **AutoDisable** property is true, then the device automatically disables itself when a **DataEvent** is enqueued.

An enqueued **DataEvent** can be delivered to the application when the **DataEventEnabled** property is true and other event delivery requirements are met. Just before delivering this event, data is copied into corresponding properties, and further data events are disabled by setting **DataEventEnabled** to false. This causes subsequent input data to be enqueued while the application processes the current input and associated properties. When the application has finished processing the current input and is ready for more data, it reenables events by setting **DataEventEnabled** to true.

An **ErrorEvent** (or events) is enqueued if an error occurs while gathering or processing input, and is delivered to the application when **DataEventEnabled** is true and other event delivery requirements are met.

The **DataCount** property may be read to obtain the total number of enqueued DataEvents.

All enqueued input may be deleted by calling **clearInput** method. See the **clearInput** method description for more details.

All data properties that are populated as a result of firing a **DataEvent** or **ErrorEvent** can be set back to their default values by calling the **clearInputProperties** method.

The application will be informed about any status change with a **StatusUpdateEvent**, also all corresponding status properties will be updated before event delivery.

**Types of voice recognition**

Voice recognition is mainly a method of specifying word candidates to be recognized and waiting for those words.

There are the following four types of voice recognition.

**Yes/No/Cancel recognition**

It listens to the sound of words classified as Yes / No / Cancel defined by the device.

For example, the voice ""OK."" is classified as Yes.

The recognized content is set in the HearingDataWord property.

For details, refer to the **startHearingYesNo** method.

**Word recognition**

The application specifies a list of words and listens for the voice of that word.

The recognized content is set in the **HearingDataWord** property.

For details, refer to the **startHearingWord** method.

**Sentence recognition**

The application specifies a word and a list of patterns of the sentences using it and awaits the sound of the sentence.

The recognized content is set in the HearingDataWordList property, **HearingDataPattern** property.

For details, see the **startHearingSentence** method.

**Free recognition**

Voice recognition leave to the device is performed without specifying the word to wait.

It does not specify waiting words and performs voice recognition entrusted to the device.

The recognized content is set in the **HearingDataWord** property.

For details, see the **startHearingFree** method.

When recognizing voice, the kind of recognition was stored in the **HearingResult** property.

## **Device Sharing**

The Voice Recognition 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 accessing some properties or calling methods that update the device.
* See the “Summary” table for precise usage prerequisites.

# Properties (UML attributes)

## **CapLanguage Property**

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

**Remarks** If true, the application can change the language. If false, the application cannot change the language.  
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 Intro-20.

## **HearingDataPattern Property**

**Syntax HearingDataPattern: *string* {read-only, access after open-claim-enable}**

**Remarks** The pattern ID recognized by the **startHearingSentence** method is set.  
This property is set by the device control just before the **DataEvent** is enqueued.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also startHearingSentence** Method

## **HearingDataWord Property**

**Syntax HearingDataWord: *string* {read-only, access after open-claim-enable}**

**Remarks** The content of voice recognition is set.  
This property is set as input data of the following method. To know which method it is for, check the HearingResult property.

**Methods Meaning**

**startHearingYesNo** Method

The recognized word is set.

**startHearingWord** Method

Recognized words are set among the word candidates

specified by the **startHearingWord** method.

**startHearingFree** Method

Recognized words and sentences are set.

The alphabet 's uppercase letters, Japanese kanji,

hiragana, katakana, etc., the contents to be set varies

depending on the device.

This property is set by the device control just before the **DataEvent** is enqueued..

**Errors** A **UposException** may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also HearingResult** Property**, startHearingYesNo** Method**,**

**startHearingWord** Method**, startHearingFree** Method

## **HearingDataWordList Property**

**Syntax HearingDataWordList: *string* {read-only, access after open-claim-enable}**

**Remarks** Comma-separated list of word information recognized by the **startHearingSentence** method.  
Each word information consists of the following information and is shown in the following order separated by a colon (":").

**Parameter Description**

*WordGoupID* Recognized word group ID

*Word* Recognized words. The content defined in the word

group is set.

For example, in the **startHearingSentence** method, set candidates as follows, Word list: "item:coffee:tea, count:a:two:three"  
Pattern list: "P1:[count] cup of [item], P2:[item]"  
**startHearingSentence** ("en-US", "item:coffee:tea, count:a:two", "P1:[count] cup of [item],P2:[item]")

If you speak "Give me two cups of coffee", device recognize “Pattern” as "P1" and “WordList” as "item:coffee, count:two".

The properties are set as follows,  
HearingDataPattern="P1";  
HearingDataWordList="item:coffee, count:two";

This property is set by the device control just before the **DataEvent** is notified.

**Errors** A **UposException** may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also startHearingSentence** Method

## **HearingResult Property**

**Syntax HearingResult: *int32* {read-only, access after open-claim-enable}**

**Remarks** A value indicating the voice recognition result is set.  
The parameters to be set are as follows.

**Value Meaning**

VRCG\_HRESULT\_YESNO\_YES

Voice recognition result of **StartHearingYesNo**

method. Also, Device got an answer that is classified as YES. The recognized content is set in the **HearingDataWord** property.

VRCG\_HRESULT\_YESNO\_NO

Voice recognition result of **startHearingYesNo**

method. Also, Device got an answer that

is classified as NO. The recognition content is set in

the **HearingDataWord** property.

VRCG\_HRESULT\_YESNO\_CANCEL

Voice recognition result of **startHearingYesNo**

method. Also, Device got responses that are classified

as CANCEL. The recognition content is set in the

**HearingDataWord** property.

VRCG\_HRESULT\_WORD

Recognition result of **startHearingWord** method. The recognition content is set in the **HearingDataWord** property.

VRCG\_HRESULT\_SENTENCE

Recognition result of **startHearingSentence** method.

The recognition content is set in the

**HearingDataWordList** property and

**HearingDataPattern** property.

VRCG\_HRESULT\_FREE

Recognition result of **startHearingFree** method. The

recognition content is set in the **HearingDataWord**

property.

This property is set by the device control just before the **DataEvent** is enqueued.

**Errors** A **UposException** may be thrown when this property is accessed.  
For further information, see **“Errors"** on page Intro-20.

**See Also HearingDataWord** Property**, HearingDataWordList** Property**, HearingDataPattern** Property**, startHearingYesNo** Method**, startHearingWord** Method**, startHearingSentence** Method**, startHearingFree** Method.

## **HearingStatus Property**

**Syntax HearingStatus: *int32* {read-only, access after open-claim-enable}**

**Remarks** A value indicating the voice recognition status is set.

**Value Meaning**

VRCG\_HSTATUS\_NONE Voice recognition is not running.

VRCG\_HSTATUS\_YESNO Voice recognition by the **startHearingYesNo** method is in progress.

VRCG\_HSTATUS\_WORD Voice recognition by the **startHearingWord** method is in progress.

VRCG\_HSTATUS\_SENTENCE

Voice recognition by the **startHearingSentence** method is in progress.

VRCG\_HSTATUS\_FREE Voice recognition by the **startHearingFree**

method is in progress.

This property is initialized by the **open** method. Also, it is set by the device control just before the voice recognition state changes.

**Errors** A **UposException** may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also startHearingYesNo** Method**, startHearingWord** Method**, startHearingSentence** Method**, startHearingFree** Method

## **LanguageList Property**

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

**Remarks** Contains the comma-delimited list of language that are supported by the device.  
The value representing the language is a value consisting of the language and country code defined in RFC 4664.  
For example, when the device supports US / English, Japan / Japanese, it will be as follows.  
"en-US, ja-JP"  
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 Intro-20.

**See Also startHearingYesNo** Method**, startHearingWord** Method**, startHearingSentence** Method**, startHearingFree** Method.

# Methods (UML operations)

## **startHearingFree Method**

**Syntax** **startHearingFree (language: *string*):**

**void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*Language* Specify the language to recognize. Specify one of the

values listed in the **LanguageList** property.

**Remarks** This method can make a voice recognition from the listed language in the **LanguageList** property. In addition, this method can be called without specifying the word candidate to be recognized from the application, however recognized word depends on the word recognizing device capability. When this method is called, proper values are set in the **HearingDataWord** property, **HearingResult** property and **HearingStatus** property just before the **DataEvent** issuing. This method is executed asynchronously . Voice recognition ends when **stopHearing** method is called.

**Errors** A **UposException** may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported language was specified.

E\_BUSY Voice recognition in progress so it cannot be executed.

**See Also** **HearingDataWord** Property, **HearingResult** Property, **HearingStatus** Property, **LanguageList** Property**, stopHearing** Method.

## **startHearingSentence Method**

**Syntax** **startHearingSentence (language: *string*, wordList: *string*, patternList: *string*):**

**void {raises-exception, use after open-claim-enable}  
Parameter Description**

*language* Specify the language to recognize. Specify one of the values listed in the **LanguageList** property.

*wordList* Specify word candidates to be waited on in a comma-separated list.

*patternList* Specify the sentence pattern information to be waited for in a comma-separated list.

Each word information specified in wordList consists of the following information and is shown in the following order, separated by a colon (":").

**Parameter Description**

*wordGroupID* ID to identify word list

*wordList* A word candidate to be waited for being separated by a colon (":")

For example, to specify word candidates "one" and "two" for word candidate’s "coffee" "tea" and word group "number" in the single item group "product", specify as follows. "item:coffee:tea, number:one:two"

Each word information specified in patternList consists of the following information, and it is shown in the following order separated by a colon (":").

**Parameter Description**

*patternID* ID to identify the pattern

*pattern* A sentence pattern to wait. To add the word list specified in wordList to the candidate, enclose the word group ID with "[" and "]". Example: "[word group ID1]" [word group ID2] "

Example: You can order coffee or tea. You can also specify how many cups you need. If you want to recognize it by voice, do as follows.

Set the **startHearingSentence** method parameter as follows:  
WordList:"item:coffee:tea, count:a:two:three" Coffee, Tea -> item:coffee:tea  
How many cups -> count:a:two:three

Invoke the method.  
**startHearingSentence** ("en-US", "item:coffee:tea,count:a:two", "P1:[count] cup of [item],P2:[item]")  
**HearingStatus**=VRCG\_HSTATUS\_SENTENCE;

People talk to "Give me two cups of coffee"

Speech recognition is performed, properties are set, and an event is notified.  
**HearingResult**=VRCG\_HRESULT\_SENTENCE;  
**HearingDataPattern**="P1";  
**HearingDataWordList**="item:coffee,count:two";  
raise **DataEvent**(0);

**Remarks** This method can make a voice recognition from the listed language in the **LanguageList** property. In addition, this method can recognize the words and sentences that are defined in *wordList* and *patternList* as parameter. When this method is called, proper values are set in the **HearingDataWord** property. **HearingResult** property and **HearingStatus** property, just before **DataEven**t issuing. This method is executed asynchronously. Voice recognition ends when **stopHearing** method is called.

**Errors** A **UposException** may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported

language was specified.

E\_BUSY Voice recognition in progress so it cannot be executed.

**See Also** **HearingDataWord** Property, **HearingResult** Property, **HearingStatus** Property, **LanguageList** Property**, stopHearing** Method

## **startHearingWord Method**

**Syntax** **startHearingWord (language: *string*, wordList: *string*):**

**void {raises-exception, use after open-claim-enable}  
Parameter Description**

*language* Specify the language to recognize. Specify one of the values listed in the **LanguageList** property.

*wordList* Specify word candidates to be waited on in a comma-separated list.  
Example: "word1, word2, word3"

**Remarks** This method can make a voice recognition from the listed language in the **LanguageList** property. In addition, this method can recognize the words that are defined in wordList as parameter. When this method is called, proper values are set in the **HearingDataWord** property, **HearingResult** property and **HearingStatus** property just before **DataEvent** issuing. This method is executed asynchronously. Voice recognition ends when **stopHearing** method is called.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported

language was specified.

E\_BUSY Voice recognition in progress so it cannot be

executed.

**See Also** **HearingDataWord** Property, **HearingResult** Property, **HearingStatus** Property, **LanguageList** Property**, stopHearing** Method.

## **startHearingYesNo Method**

**Syntax** **startHearingYesNo (language: *string*):**

**void {raises-exception, use after open-claim-enable}  
Parameter Description**

*language* Specify the language to recognize. Specify one of the

values listed in the **LanguageList** property.

**Remarks** This method can make a voice recognition from the listed language in the **LanguageList** property. In addition, this method can recognize the words that are defined in the device as the recognition candidate corresponding to "Yes" "No" "Cancel". When this method is called, proper values are set in the **HearingDataWord** property, **HearingResult** property and **HearingStatus** property, just before **DataEvent** issuing. This method is executed asynchronously. Voice recognition ends when **stopHearing** method is called.

**Errors** A **UposException** may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported

language was specified.

E\_BUSY Voice recognition in progress so it cannot be executed.

**See Also** **LanguageList** Property**, HearingDataWord** Property, **Hearing Result** Property, **LanguageList** Property**, stopHearing** Method.

## **stopHearing Method**

**Syntax** **stopHearing ( ):**

**void {raises-exception, use after open-claim-enable}  
Remarks** Voice Recognition ends when this property called. This method is executed synchronously.

**Errors** A **UposException** may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported

language was specified.

# Events (UML interfaces)

## **DataEvent**

**<<event>> upos::events::DataEvent**

**Status : *int32*{read-only}**

**Description** Notifies the application when data from the Voice Recognition device is available to be read.

**Attributes** This event contains the following attributes:

**Attribute Type Description**

*Status int32 Set to 0.*

**Remarks** Before this event is delivered, the voice recognition information is enqueued into the area that is indicated by the startHearingXXX kinds of methods.

**See Also** HearingResult Property, **“Events"** on page Intro-19, **StartHearingYesNo,StartHearingWord,StartHearingSentence,StartHearingFree** Method , **directIO** method.

## **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 Voice Recognition Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** method.

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent  
ErrorCode : *int32*{read-write}  
ErrorCodeExtended : *int32*{read-write}  
ErrorLocus : *int32*{read-write}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that a Voice Recognition Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is re-tried, and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is re-tried, and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period of time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
Status : *int32* {read-only}**

**Description** *Notifies the application that there is a change in the power status or a status of the Voice Recognition device.*

**Attributes**  This event contains the following attribute:

**Attributes Type Description**

*Status* *int32* Indicates a change in the power status of the unit.

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process. See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

VRCG\_SUE\_START\_HEARING \_FREE  
 It will be notified when hearing free starts.

VRCG\_SUE\_START\_HEARING \_SENTENCE  
 It will be notified when hearing sentence starts.

VRCG\_SUE\_START\_HEARING \_WORD  
 It will be notified when hearing word starts.

VRCG\_SUE\_START\_HEARING \_YESNO  
 It will be notified when hearing yesno starts.

VRCG\_SUE\_STOP\_HEARING  
 It will be notified when hearing stops.

**Remarks** Enqueued when the Voice Recognition Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# Chapter 43

# Sound Player

This Chapter defines the Sound Player device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.16 | Not Supported |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | 1.16 | Not supported |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.16 | Not supported |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | open |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | - |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | - |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **CapAssociatedHardTotalsDevice** | *string* | {read-write} | 1.16 | open |
| **CapMultiPlay:** | *boolean* | {read-only} | 1.16 | open |
| **CapSoundTypeList:** | *string* | {read-only} | 1.16 | open |
| **CapStorageTarget** | *int32* | {read-only} | 1.16 | open |
| **CapVolume:** | *boolean* | {read-only} | 1.16 | open |
| **DeviceSoundList:** | *string* | {read-only} | 1.16 | open |
| **OutputIDList:** | *string* | {read-only} | 1.16 | open, claim & enable |
| **StorageTarget** | *string* | {read-write} | 1.16 | open, claim & enable |
| **Volume:** | *int32* | {read-write} | 1.16 | open, claim & enable |

|  |  |
| --- | --- |
| **Me Methods (UML operations)** |  |
| ***Common*** |  |
| ***Name*** | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | 1.16 |
| **close ( ): void {raises-exception, use after open}** | 1.16 |
| **claim (timeout:** *int32***): void {raises-exception, use after open}** | 1.16 |
| **release ( ): void {raises-exception, use after open, claim}** | 1.16 |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | 1.16 |
| **clearInput ( ): void { }** | ***Not supported*** |
| **clearInputProperties ( ): void { }** | ***Not supported*** |
| **clearOutput ( ): void { }** | 1.16 |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | 1.16 |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, claim, enable}** | 1.16 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, claim, enable}** | 1.16 |
| **retrieveStatistics ( inout statisticsBuffer:** *string* **): void {raises-exception, use after open, claim, enable}** | 1.16 |
| **Me Methods (UML operations)(continued)** |  |
| ***Common*** |  |
| ***Name*** | ***Version*** |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, claim, enable}** | 1.16 |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, claim, enable}** | 1.16 |
|  |  |
| ***Specific*** |  |
| ***Name*** | ***Version*** |
| **playSound(fileName:** *string***, loop:** *boolean***):**  **void { raises-exception, use after open, claim, enable}** | 1.16 |
| **stopSound(outputID:***int32***):**  **void {raises-exception, use after open, claim, enable}** | 1.16 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
|  |  |  |  |
| **upos::events::DataEvent** |  | ***Not Supported*** | 1.16 |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.16 |
| **EventNumber:** | *int32* | {read-only} |  |
| **Data:** | *int32* | {read-write} |  |
| **Obj:** | *object* | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  |  | 1.16 |
| **ErrorCode:** | *int32* | {read-only} |  |
| **ErrorCodeExtended:** | *int32* | {read-only} |  |
| **ErrorLocus:** | *int32* | {read-only} |  |
| **\*pErrorResponse:** | *int32* | {read-write} |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  |  | 1.16 |
| **OutputID:** | *int32* | {read-only} |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |

# General Information

The Sound Player programmatic name is "Sound Player".

## **Capabilities**

The Sound Player has the following capability:

* Play audio file.

## **Sound Player Class Diagram**

The following diagram shows the relationships between the Sound player classes.



Fig. Chap.43-1 Sound Player Class Diagram

# Model

The Sound Player follows the general device behavior model for asynchronous output devices:

* The Device validates the method parameters and produces an error condition immediately if necessary. If the validation is successful, the Device does the following:
* Audio files will be played sequentially. When **playSound** method is called, device starts the playing sound that is specified by the method parameters and the requested sound file data placed in a queue and corresponding OutputID is stored at **OutputID** property and added to the **OutputIDList** property as a listed value. And sets the **OutputID** property to a unique integer identifier for this request.
* When the sound playing starts **StatusUpdateEvent** is evoked as the value of SPLY\_SUE\_START\_PLAY\_SOUND. When the sound playing is finished an **OutputCompleteEvent** is enqueued for the delivery to the application and corresponding OutputID is stored in **OutputID** property. At the same time, **StatusUpdateEvent** is evoked as the value of SPLY\_SUE\_STOP\_PLAY\_SOUND. The application should compare the returned **OutputCompleteEvent** property **OutputID** value with the **OutputID** value set by the asynchronous process method call used to send the data in order to track what data has been successfully sent to the device.
* When **stopSound** method is called, device stop the playing sound according to the OutputID property value and the current playing sound is terminated and enqueued sound file data is cleared. After this method is executed, corresponding **OutputID** property and **OutputIDList** values are not changed. No **OutputCompleteEvent** is fired and only **StatusUpdateEvent** will be evoked the value of SPLY\_SUE\_STOP\_PLAY\_SOUND.
* If an error occurs while processing a request, an **ErrorEvent** is enqueued which will be delivered to the application after the events already enqueued, including **OutputCompleteEvent**. No further asynchronous output will occur until the event has been delivered to the application. If the response is ER\_CLEAR, then outstanding asynchronous output is cleared. If the response is ER\_RETRY, then output is retried; note that if several outputs were simultaneously in progress at the time that the error was detected, then the Service may need to retry all of these outputs.
* Asynchronous output is always performed on a first-in first-out basis. If the device supports concurrent playback, the request will be executed simultaneously. To check if the device supports simultaneous playback, check the **CapMultiPlay** property.
* If the request is terminated before completion, due to reasons such as the application calling the **clearOutput** method, then no **OutputCompleteEvent** is delivered.
* Application can also delete the output individually by calling the **stopSound** method. Also, in this case **OutputCompleteEvent** will not be notified."
* The **CapSoundTypeList** property lists audio file types that the device can play.
* The application will be informed about any status change with a **StatusUpdateEvent**, also all corresponding status properties will be updated before event delivery.

* If device supports either or both of HardTotals devices and the host file system, the application should set the **StorageTarget** property accordingly to tell where to access the data file.
* If device needs to be able to access the audio files played with **playSound** method from a HardTotals device, the **CapAssociatedHardTotalsDevice** property holds the open name of the associated HardTotals device.

## **Device Sharing**

The Sound Player 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 accessing some properties or calling methods that update the device.
* See the “Summary” table for precise usage prerequisites.

# Properties(UML attributes)

**CapAssociatedHardTotalsDevice Property**

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

**Remarks** Holds the open name of the associated HardTotals device if the device is able to write to such devices which is the case if **CapStorageTarget** is either CST\_ALL or CST\_HARDTOTALS\_ONLY. If **CapStorageTarget** is CST\_HOST\_ONLY this property value must be the empty string.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapStorageTarget** Property

## **CapMultiPlay Property**

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

**Remarks** If true, the application can play sound simultaneously.

If false, the application cannot play sound simultaneously.

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 Intro-20.

**See Also** **playSound** Method.

## **CapSoundTypeList Property**

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

**Remarks** Contains the comma-delimited list of file type that is supported by the device.

For example, if the device only supports WAV and OGG, then this property should be set to “WAV, OGG". 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 Intro-20.

**See Also** **playSound** Method

**CapStorageTarget Property**

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

**Remarks** This is an enumeration and announces where the device is able to write the recorded sound data file to.  
It holds one of the following values.

**Value Meaning**

CST\_HARDTOTAL\_ONLY Only an associate **HardTotals** device is supported.

CST\_HOST\_ONLY Only the host’s file system is supported.

CST\_ALL Both, the associated **HardTotals** device and the host’s file system is supported.

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

If a HardTotals device is supported the storage target, the property value should be CST\_HARDTOTAL\_ONLY or CST\_ALL and the property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

## **CapVolume Property**

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

**Remarks** If true, the application can change the volume during playback.

If false, the application cannot change the volume during playback.

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 Intro-20.

**See Also** **Volume** Property.

## **DeviceSoundList Property**

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

**Remarks** Contains the comma-delimited list of device sound ID that is supported by the 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 Intro-20.

**See Also** **playSound** Method

## **OutputIDList Property**

**Syntax OutputIDList : *string* {read-only, access after open, claim-enable}**

**Remarks** Contains the comma-delimited list of OutputID that is output by the **playSound** method. 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 Intro-20.

**See Also** **playSound** Method

**StorageTarget Property**

**Syntax StorageTarget: *string* {read-write, access after open-claim-enable}**

**Remarks** Holds the open name of HardTotals device, if **CapStorageTarget** property value is CST\_HARTTOTAL\_ONLY or CST\_ALL. If **CapStorageTarget** property value is CST\_HOST\_ONLY, this property value must be the empty string. This property is initialized by the **open** method according to the value hold by **CapStorageTarget**.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also CapStorageTarget** Property

## **Volume Property**

**Syntax Volume : *int32* {read-write, access after open-claim-enable}**

**Remarks** Holds the volume at playing sound.

Legal values range from zero through 100.

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 Intro-20.

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **playSound** Method

# Methods (UML operations)

## **playSound Method**

**Syntax** **playSound (fileName :*****string*, loop : *boolean*):  
 void{raises-exception, use after open-claim-enable}**

**Parameter Description**

*fileName* Specifies the file name of audio file. Or, specifies one of the sound ID defined by **DeviceSoundList**.

*loop* When true is specified, loop playback is performed, and if false is specified, loop playback will not be performed.

**Remarks** Play audio file specified by fileName or device definition sound.

Audio files might be located in the area managed by "**Hard Totals**" service.

This method will be performed asynchronously. To stop playback, call the **stopSound** method.

**Errors** A UposException may be thrown when this method is invoked. For further information , see “Errors” on page Intro-20. Some possible values of the exception’s ErrorCode property are:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported sound file was specified.

E\_NOEXIST File does not exist.

**See Also** **CapSoundType** Property, **DeviceSoundList** Property, **stopSound** Method

## **stopSound Method**

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

**Parameter Description**

*outputID* Specify the outputID of the sound to stop.

**Remarks** Terminates specified audio playback according to the **OutputID** property value.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “Errors” on page Intro-20. Some possible values of the exception’s ErrorCode property are:

**Value Meaning**

E\_ILLEGAL The specified sound is not being played.

**See Also** **OutputID** Property, **startSound** Method

# Events (UML interfaces)

## **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 Sound Player Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** method

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent  
ErrorCode : *int32*{read-write}  
ErrorCodeExtended : *int32*{read-write}  
ErrorLocus : *int32*{read-write}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that a Sound Player Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is re-tried, and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is re-tried, and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period of time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **OutputCompleteEvent**

**<<event>> upos::events::OutputCompleteEvent   
OutputID : *int32*{read-only}**

**Description** Notify the application that the queued output request associated with the *outputID* property has completed successfully.

**Attributes** This event contains the following attributes:

**Attribute 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 Intro-25

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
Status : *int32* {read-only}**

**Description** *Notifies the application that there is an operation status change or a status of the sound player device.*

**Attributes** This event contains the following attribute:

**Attributes Type Description**

*Status*  *int32* Indicates a change of operation status of sound player 　 device.

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process.

See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

SPLY\_SUE\_START\_PLAY\_SOUND It will be notified when sound playing start.

SPLY\_SUE\_STOP\_PLAY\_SOUND It will be notified when sound playing stop.

**Remarks** Enqueued when the Sound Player Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# Chapter 44

# Speech Synthesis

This Chapter defines the Speech Synthesis device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.16 | Not Supported |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | 1.16 | Not Supported |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.16 | Not Supported |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | open |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Properties (Continued)** | |  |  |  |  | |
| ***Specific*** | | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** | |
| **CapLanguage:** | | *boolean* | {read-only} | 1.16 | open | |
| **CapPitch:** | | *boolean* | {read-only} | 1.16 | open | |
| **CapSpeed:** | | *boolean* | {read-only} | 1.16 | open | |
| **CapVoice:** | | *boolean* | {read-only} | 1.16 | open | |
| **CapVolume:** | | *boolean* | {read-only} | 1.16 | open | |
| **Language:** | | *string* | {read-write} | 1.16 | open, claim & enable | |
| **LanguageList:** | | *string* | {read-only} | 1.16 | open | |
| **OutputIDList:** | | *string* | {read-only} | 1.16 | open, claim & enable | |
| **Pitch:** | | *int32* | {read-write} | 1.16 | open, claim & enable | |
| **Speed:** | | *int32* | {read-write} | 1.16 | open, claim & enable | |
| **Voice:** | | *string* | {read-write} | 1.16 | open, claim & enable | |
| **VoiceList:** | | *string* | {read-only} | 1.16 | open | |
| **Volume:** | | *int32* | {read-write} | 1.16 | open, claim & enable | |
|  | |  |  |  |  | |
| **Me Methods (UML operations)** | | | | |  | |
| ***Common*** | | | | |  | |
| ***Name*** | | | | | ***Version*** | |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | | | | | 1.16 | |
| **close ( ): void {raises-exception, use after open}** | | | | | 1.16 | |
| **claim ( timeout:** *int32***): void {raises-exception, use after open}** | | | | | 1.16 | |
| **release ( ): void {raises-exception, use after open, claim}** | | | | | 1.16 | |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | | | | | 1.16 | |
| **clearInput ( ): void { }** | | | | | Not  supported | |
| **clearInputProperties ( ): void { }** | | | | | Not  supported | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | |  | |
| **Methods (UML operations)(continued)** | | | | |  | |
| **clearOutput ( ): void { }** | | | | | 1.16 | |
| ***Common*** | | | | |  | |
| ***Name*** | | | | | ***Version*** | |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | | | | | 1.16 | |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| ***Specific*** | | | | |  | |
| ***Name*** | | | | |  | |
| **speak (text:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| **speakImmediate (text:** *string***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| **stopCurrentSpeaking ( ): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |
| **stopSpeaking (outputID:** *int32***): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
|  |  |  |  |
| **upos::events::DataEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.16 |
| **EventNumber:** | *int32* | {read-only} |  |
| **Data:** | *int32* | {read-write} |  |
| **Obj:** | *object* | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  |  | 1.16 |
| **ErrorCode:** | *int32* | {read-only} |  |
| **ErrorCodeExtended:** | *int32* | {read-only} |  |
| **ErrorLocus:** | *int32* | {read-only} |  |
| **\*pErrorResponse:** | *int32* | {read-write} |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  |  | 1.16 |
| **OutputID:** | *int32* | {read-only} |  |
|  |  |  |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |

# General Information

The Speech Synthesis programmatic name is "Speech Synthesis".

## **Capabilities**

The Speech Synthesis has the following capability:

* Convert text to speech and read it aloud.

## **Speech Synthesis Class Diagram**

The following diagram shows the relationships between the Speech Synthesis classes.

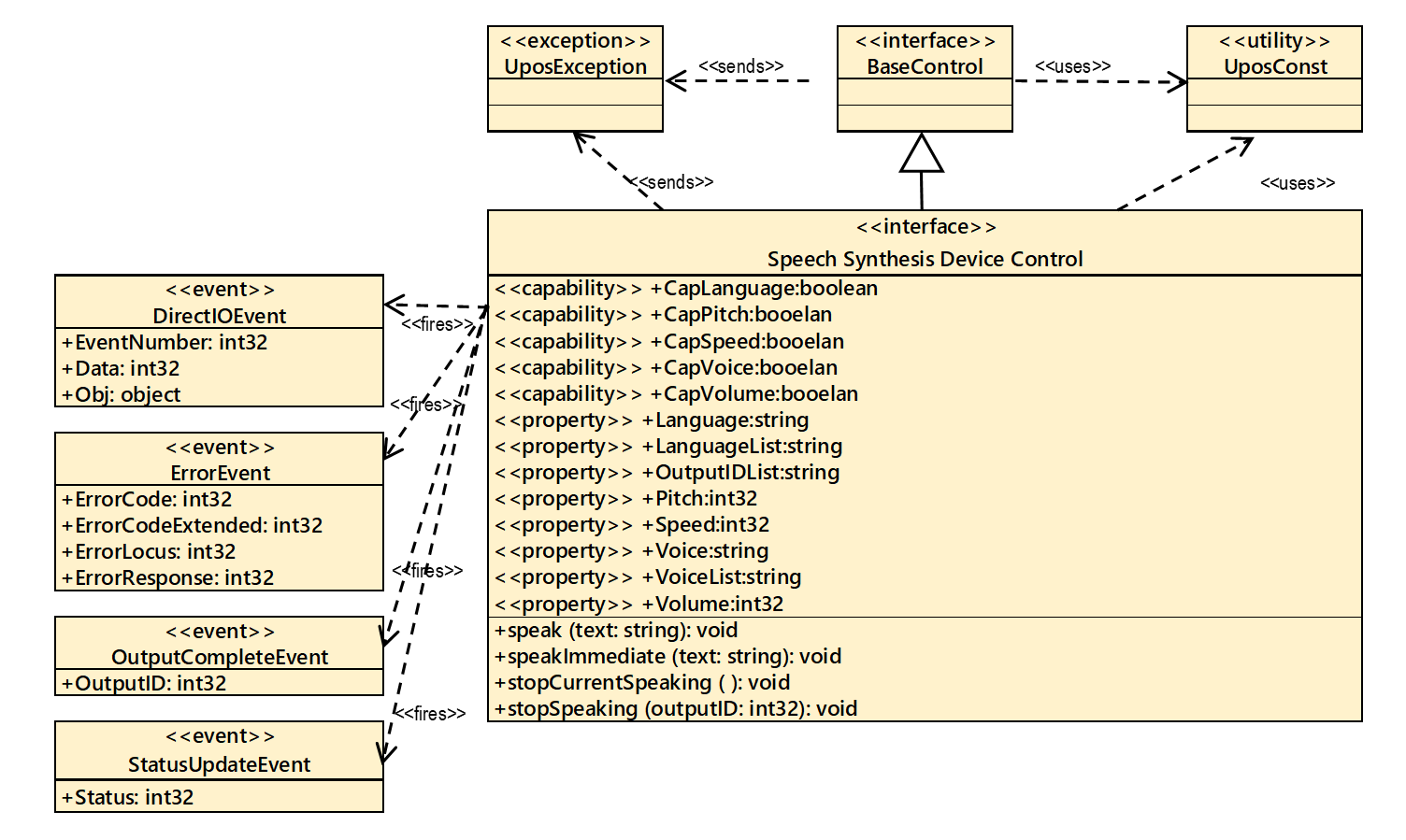


Fig. Chap. 44-1 Speech Synthesis Class Diagram

# Model

The Speech Synthesis follows the general device behavior model for output devices with some enhancements.

The application calls a **speak** method or **speakImmediate** method to speech.

The **speak** method acts to start speaking from the words specified by text, while the **speakImmediate** method ends immediately previous **speak** method, and starts speaking the word specified by text asynchronously and immediately.

When speak or **speakImmediate** method is called device start the speaking based on the setting value of **Language**, **Volume**, **Pitch** and **Speed** properties. And requested utterance written by text data placed in a queue and corresponding OutputID is stored at **OutputID** property and added to the **OutputIDList** property as listed value. And sets the **OutputID** property to a unique integer identifier for this request.

When an utterance of **speak** method or **speakImmediate** method starts, **StatusUpdateEvent** is evoked as the value of SPSY\_SUE\_START\_SPEAK. When the utterance is finished an **OutputCompleteEvent** is enqueued for the delivery to the application and corresponding **OutputID** is stored in **OutputID** property. At the same time **StatusUpDateEvent** is evoked as the value of SPSY\_SUE\_STOP\_SPEAK. The application should compare the returned **OutputCompleteEvent** property **OutputID** value with OutputID value set by the asynchronous process method call used to send the data in order to track what data has been successfully sent to the device

When **speakImmediate** method is called during the utterance of **speak** method or **speakImmediate** method call, utterance will be stopped immediately. And **StatusUpdateEvent** is evoked as the value of SPSY\_SUE\_STOP\_SPEAK. However, **OutputCompleteEvent** is not fired. And current **speak** method or **speakImmediate** method corresponding **OutputID** property and **OutputIDList** property values are not changed.

When s**topCurrentSpeaking** method is called, current utterance generated by **speak** method or **speakImmediate** method will be stopped and **StatusUpdateEvent** is evoked as the value of SPSY\_SUE\_STOP\_SPEAK. And no **OutputCompleteEven**t is fired. And current **speak** method or **speakImmediate** method corresponding **OutputID** property and **OutputIDList** property values are not changed.

When **stopSpeaking** method is called, specified **OutputID** valued utterance is stopped and deleted. And **OutputID** property value in the **OutputIDList** property is eliminated.

When utterance is stopped **StatusUpdateEvent** is evoked as the value of SPSY\_SUE\_STOP\_SPEAK. And no **OutputCompleteEvent** is fired.

If an error occurs while processing a request, an **ErrorEvent** is enqueued which will be delivered to the application after the events already enqueued, including **OutputCompleteEvent**. No further asynchronous output will occur until the event has been delivered to the application. If the response is ER\_CLEAR, then outstanding asynchronous output is cleared. If the response is ER\_RETRY, then output is retried; note that if several outputs were simultaneously in progress at the time that the error was detected, then the service may need to retry all of these outputs.

Asynchronous output is always performed on a first-in first-out basis.

If the request is terminated before completion, due to reasons such as the application calling the **clearOutput** method, then no **OutputCompleteEvent** is delivered.

The application will be informed about any status change with a **StatusUpdateEvent**, also all corresponding status properties will be updated before event delivery.

## **Device Sharing**

The Speech Synthesis 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 accessing some properties or calling methods that update the device.
* See the “Summary” table for precise usage prerequisites.

# Properties (UML attributes)

## **CapLanguage Property**

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

**Remarks** If true, the application can change the language. If false, the application cannot change the language.

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 Intro-20.

**See Also Language** Property

## **CapPitch Property**

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

**Remarks** If true, the application can change the pitch. If false, the application cannot change the pitch.

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 Intro-20.

**See Also Pitch** Property

## **CapSpeed Property**

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

**Remarks** If true, the application can change the speed. If false, the application cannot change the speed.

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 Intro-20.

**See Also Speed** Property

## **CapVoice Property**

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

**Remarks** If true, the application can change the voice. If false, the application cannot change the voice.

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 Intro-20.

**See Also Voice** Property

## **CapVolume Property**

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

**Remarks** If true, the application can change the volume. If false, the application cannot change the volume.

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 Intro-20.

**See Also Volume** Property

## **Language Property**

**Syntax Language: *string* {read-write, access after open-claim-enable}**

**Remarks** Indicates the language to speak. Valid values are one of the values listed in the **LanguageList** property.

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 Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported

language was specified.

**See Also speak** Method, **speakImmediate** Method

## **LanguageList Property**

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

**Remarks** Contains the comma-delimited list of language that are supported by the device. The value representing the language is a value consisting of the language and country code defined in RFC 4664. For example, when the device supports US / English, Japan / Japanese, it will be as follows.  
"en-US, ja-JP"

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 Intro-20.

**See Also Language** Property

## **OutputIDList Property**

**Syntax OutputIDList : *string* {read-write, access after open-claim-enable}**

**Remarks** Comma-separated list of **OutputID** property values of audio being played by **Speak** method or **SpeakImmediate** method. This list indicates the capability how many and what kinds of utterance can be done by the targeted Speech Synthesis device

This property is initialized by the **open** method. It will also be updated as the speech request increases or decreases.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also speak** Method, **speakImmediate** Method

## **Pitch Property**

**Syntax Pitch: *int32* {read-write, access after open-claim-enable}**

**Remarks** Holds the pitch at speech. Legal values range from 50% through 200%.

This property is initialized to 100% by the **open** method.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also speak** Method, **speakImmediate** Method

## **Speed Property**

**Syntax Speed: *int32* {read-write, access after open-claim-enable}**

**Remarks** Holds the speed at speech. Legal values range from 50% through 200%.

This property is initialized to 100% by the **open** method.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also speak** Method, **speakImmediate** Method

## **Voice Property**

**Syntax Voice : *string* {read-write, access after open-claim-enable }**

**Remarks** Indicates the voice tone to speak. Valid values are one of the values listed in the **VoiceList** property.

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 Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported

voice was specified.

**See Also speak** Method, **speakImmediate** Method

## **VoiceList Property**

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

**Remarks** A list of speech able voices are shown in a comma-separated list. For example, when the device supports male and female voice tones, it looks like the following.  
"MALE\_VOICE, FEMALE\_VOICE"  
(The content of the value depends on the 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 Intro-20.

**See Also Voice** Property

## **Volume Property**

**Syntax Volume :** ***int32*****{read-write, access after open-claim-enable}**

**Remarks** Holds the volume at speech. Legal values range from zero through 100.

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 Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also speak** Method, **speakImmediate** Method

# Methods (UML operations)

## **speak Method**

**Syntax** **speak (text: *string*):**

**void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*Text* Specify the text to speak.

**Remarks** Device utters after converting the specified string into speech.

The utterance is executed according to the setting contents of **Language** property, **Volume** property, **Pitch** property, **Speed** property, but by inserting the following tag in the text, it is possible to change the utterance after the tag.

Content written in text is uttered with the following parameter settings.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Description** | **Value**  **(decimal integer)** | **Default Value**  **(decimal integer)** |
| *volume* | Specify the volume of the uttered voice. | 1 to 100 | 50 |
| *pitch* | Specify the high or low of the uttered voice. | 50 to 200 | 100 |
| *speed* | Specify the speed of the uttered voice. | 50 to 200 | 100 |
| *pause* | Specify the time to pause in milliseconds. | 1 to 50000 | 1 |
| *reset* | Rest the effect of volume, pitch, speed to the default value. | - | - |

If dialogue is “ Hello. Today, it’s nice weather.”

Then if you would like to use the default setting of speed, volume, pitch for the “Hello”. And would like to put a pose between “Hello” and “Today” 1000 milliseconds and would like to change the speaking pith of “Today” to 150 and increase the volume to 80. Then for the “It’s nice weather” would like return to the default value by using the reset. It is described as follows

Hello.{pause=1000,pitch=150,volue=80}Today,{reset}It's nice weather.

Those utterance defined as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data | Remarks | |
| Utterance written by text with the **speak** method parameter. Text will be spoken under the assigned parameter condition. | {#=f}XXXX{#=f}YYYY | #:Tag names It is volume, pitch, speed, pause and reset. | f:Tag values It is described in the Tag Value Table. |

When this method is called by the application, device validate the method parameters, and if validation is successful buffer the request in program memory and deliver it to the device and process it. And device sets the unique integer identifier into the **OutputID** property. When device successfully complete a request an **OutputCompleteEvent** is enqueued for delivery to the application. If the device does not support volume change etc., that tag will be ignored. This method is executed asynchronously. To end an utterance halfway, call the **stopCurrentSpeaking** method or the **stopSpeaking** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20. Some possible values of the exception’s **ErrorCode** property are:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. The language set in the **Language** property and the language specified by Text do not match.

**See Also** **Language** Property**, Volume** Property, **Pitch** Property, **OutputID** Property, **Speed** Property, **stopCurrentSpeaking** Method, **stopSpeaking** Method

## **speakImmediate Method**

**Syntax** **speakImmediate (text: *string*):**

**void {raises-exception, use after open-claim-enable}  
Parameter Description**

*text* Specify the text to speak.

**Remarks** The **speak** method acts to start speaking the words specified by text, while the **speakImmediate** method ends immediately previous **speak** method, and starts speaking the word specified by text asynchronously and immediately.

After executing the same processing as the **clearOutput** method, speak the wording specified by text.

Like this **speak** method, this method can also change a specific wording by inserting a tag. For details, refer to the description of **speak** method.

This method is executed asynchronously. To end an utterance halfway, call the **stopCurrentSpeaking** method or the **stopSpeaking** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20. Some possible values of the exception’s **ErrorCode** property are:

**Value Meaning**

E\_ILLEGAL An invalid value was specified. The language set in the **Language** property and the language specified by Text do not match.

**See Also** **Language** Property**, Volume** Property, **Pitch** Property, **Speed** Property, **speak** Method, **stopCurrentSpeaking** Method, **stopSpeaking** Method

## **stopCurrentSpeaking Method**

**Syntax** **stopCurrentSpeaking ( ):**

**void {raises-exception, use after open-claim-enable}**

**Remarks** The **speak** method and **speakImmediate** method start the speaking words specified by text and ends when **stopCurrentSpeaking** method is called. This method handles asynchronously.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL Speech is not running.

**See Also** **speak** Method, **speakImmediate** Method

## **stopSpeaking Method**

**Syntax** **stopSpeaking (outputID : *int32*):**

**void {raises-exception, use after open, claim, enable}**

**Parameter Description**

*outputID* Specify the value of the **OutputID** property you wish to terminate.

**Remarks** Stop and delete the utterance specified in OutputID.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **OutputID** Property, **speak** Method, **speakImmediate** Method

# Events (UML interfaces)

## **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 Sound Player Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** method

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent  
ErrorCode : *int32*{read-write}  
ErrorCodeExtended : *int32*{read-write}  
ErrorLocus : *int32*{read-write}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that a Speech Synthesis Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is re-tried, and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is re-tried, and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period of time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **OutputCompleteEvent**

**<<event>> upos::events::OutputCompleteEvent  
OutputID : *int32*{read-only}**

**Description** Notify the application that the queued output request associated with the *outputID* property has completed successfully.

**Attributes** This event contains the following attributes:

**Attribute 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 Intro-25

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
Status : *int32* {read-only}**

**Description** *Notifies the application that there is an operation status change or a status of the Speech Synthesis device.*

**Attributes**  This event contains the following attribute:

**Attribute Type Description**

*Status* *int32* Indicates a change of operation status of sound player device

.

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process. See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

SPSY\_SUE\_START\_SPEAK It will be notified when speech synthesis starts.

SPSY\_SUE\_STOP\_SPEAK It will be notified when speech synthesis stops.

**Remarks** Enqueued when the Speech Synthesis Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# Chapter 45

# Gesture Control

This Chapter defines the Gesture Control device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | --- | Not supported |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | --- | Not supported |
| **DataEventEnabled:** | *boolean* | {read-write} | --- | Not supported |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | open |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | -- |
|  |  |  |  |  |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **CapAssociatedHardTotalsDevice:** | *string* | {read-only} | 1.16 | open |
| **CapMotion:** | *boolean* | {read-only} | 1.16 | open |
| **CapMotionCreation:** | *boolean* | {read-only} | 1.16 | open |
| **CapPose:** | *boolean* | {read-only} | 1.16 | open |
| **CapPoseCreation:** | *boolean* | {read-only} | 1.16 | open |
| **CapStorageTarget:** | *int32* | {read-only} | 1.16 | open |
| **AutoMode:** | *string* | {read-write} | 1.16 | open, claim & enable |
| **AutoModeList:** | *string* | {read-only} | 1.16 | open |
| **MotionList:** | *string* | {read-only} | 1.16 | open |
| **PoseCreationMode:** | *boolean* | {read-write} | 1.16 | open, claim & enable |
| **PoseList:** | *string* | {read-only} | 1.16 | open |
| **StorageTarget:** | *int32* | {read-write} | 1.16 | open, claim & enable |
| **JointList:** | *string* | {read-only} | 1.16 | open |

|  |  |
| --- | --- |
| **Methods (UML operations)** |  |
| ***Common*** |  |
| ***Name*** | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | 1.16 |
| **close ( ): void {raises-exception, use after open}** | 1.16 |
| **claim (timeout:** *int32***): void {raises-exception, use after open}** | 1.16 |
| **release ( ): void {raises-exception, use after open, claim}** | 1.16 |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | 1.16 |
| **clearInput ( ): void { }** | Not  supported |
| **clearInputProperties ( ): void { }** | Not  supported |
| **clearOutput ( ): void { }** | 1.16 |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, enable}** | 1.16 |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | 1.16 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | 1.16 |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | 1.16 |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, enable}** | 1.16 |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, enable}** | 1.16 |

|  |  |
| --- | --- |
| **Me Methods (UML operations)(continued)** |  |
| ***Specific*** |  |
| ***Name*** | ***Version*** |
| **createMotion (fileName:***string***, poseList:***string***):  void { raises-exception, use after open, claim, enable }** | 1.16 |
| **createPose (fileName:***string***, time:***int32***): void { raises-exception, use after open, claim, enable }** | 1.16 |
| **getPosition (jointID:***string***, position: *out*** *int32 by reference***): void { raises-exception, use after open, claim, enable }** | 1.16 |
| **setPosition (positionList:***string,* **time:***int32 ,* **absolute:***boolean***):  void { raises-exception, use after open, claim, enable }** | 1.16 |
| **setSpeed (speedList:***string***, time:***int32***):  void { raises-exception, use after open, claim, enable }** | 1.16 |
| **startMotion (fileName:***string***):  void { raises-exception, use after open, claim, enable }** | 1.16 |
| **startPose (fileName:***string***):  void { raises-exception, use after open, claim, enable }** | 1.16 |
| **stopControl (outputID:***int32***):  void { raises-exception, use after open, claim, enable }** | 1.16 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Events (UML interfaces)** |  |  |  |
| ***Name*** | ***Type*** | ***Mutability*** | ***Version*** |
|  |  |  |  |
| **upos::events::DataEvent** |  | ***Not Supported*** |  |
|  |  |  |  |
| **upos::events::DirectIOEvent** |  |  | 1.16 |
| **EventNumber:** | *int32* | {read-only} |  |
| **Data:** | *int32* | {read-write} |  |
| **Obj:** | *object* | {read-write} |  |
|  |  |  |  |
| **upos::events::ErrorEvent** |  |  | 1.16 |
| **ErrorCode:** | *int32* | {read-only} |  |
| **ErrorCodeExtended:** | *int32* | {read-only} |  |
| **ErrorLocus:** | *int32* | {read-only} |  |
| **\*pErrorResponse:** | *int32* | {read-write} |  |
|  |  |  |  |
| **upos::events::OutputCompleteEvent** |  |  | 1.16 |
| **OutputID:** | *int32* | {read-only} |  |
|  |  |  |  |
| **upos::events::StatusUpdateEvent** |  |  | 1.16 |
| **Status:** | *int32* | {read-only} |  |

# General Information

The Gesture Control device programmatic name is "Gesture Control".

## **Capabilities**

The Gesture Control device has the following capability:

* It controls the behavior of various joint components and parts.
* The operation is automatically controlled by interlocking various joints and other devices.
* Register and play the defined pose and motion.

## **Gesture Control Class Diagram**

The following diagram shows the relationships between the Gesture Control classes.



Fig. Chap. 45-1 Gesture Control Class Diagram

# Model

The Gesture Control follows the general device behavior model for asynchronous output devices:

* The application calls a **setPosition**, **setSpeed**, **startPose**, **startMotion** method to start output. The Device validates the method parameters and produces an error condition immediately if necessary. If the validation is successful, the Device does the following:

･Buffers the request in program memory, for delivery to the Physical Device as soon as the Physical Device can receive and process it.

･Sets the **OutputID** property to a unique integer identifier for this request.

･Returns as soon as possible.

* When the Device successfully completes a request, an **OutputCompleteEvent** is enqueued for delivery to the application. A property of this event contains the outputID of the completed request. The application should compare the returned **OutputCompleteEvent** property OutputID value with theOutputID value set by the asynchronous process method call used to send the data, in order to track what data has been successfully sent to the device.
* If an error occurs while processing a request, an **ErrorEvent** is enqueued which will be delivered to the application after the events already enqueued, including **OutputCompleteEvent**. No further asynchronous output will occur until the event has been delivered to the application. If the response is ER\_CLEAR, then outstanding asynchronous output is cleared. If the response is ER\_RETRY, then output is retried; note that if several outputs were simultaneously in progress at the time that the error was detected, then the Service may need to retry all of these outputs.
* Asynchronous output is always performed on a first-in first-out basis.
* If the request is terminated before completion, due to reasons such as the application calling the **clearOutput** method, then no **OutputCompleteEvent** is delivered.
* Application can also delete the output individually by calling the **stopControl** method. Also, in this case **OutputCompleteEvent** will not be notified.
* The application will be informed about any status change with a **StatusUpdateEvent**, also all corresponding status properties will be updated before event delivery.

## **Automatic control**

Automatic control of a joint means to automatically control a joint on the device side, such as tracking according to the movement of a person's face, in cooperation with a camera or the like connected to the device.

The automatic control function is device dependent. For possible automatic control, it is enabled by confirming with the **AutoModeList** property and setting a value in the **AutoMode** property.

## **Pose / Motion**

Pose refers to setting the position of one or more defined joints.

For example, it is an action that lifts a hand.

To execute a pose, specify the pose file name by the **startPose** method or the pose name defined in the device.

Create the pose file with the **createPose** method described later. Pose defined in the device will be checked in the value of **PoseList** property.

To execute motion, specify the motion file name or the motion name defined in the device with the **startMotion** method.

Motion files are created by the **createMotion** method to be described later. Motion defined in the device can be checked with the value of **MotionList** property.

To create a pose file, first set the **PoseCreationMode** property to TRUE and enable the pose registration function. When pose registration function is enabled, each joint is set to the default position. At this time, if the automatic control mode is enabled, the automatic control mode is temporarily invalidated.

Then, an application can create a pose file by setting the value defined as a pose with the **setPosition** method and calling the **createPose** method.

A motion file can be created and recorded by specifying the pose defined in the created pose file or the pose defined in the device and creating it as a series of continuously changing actions and calling the **createMotion** method.

Since the created pose and motion files are recorded in the area may store in either the HardTotals devices or the host file system, or both, and the **CapStorageTarget** property will show the device’s data file storage location capability.

If device supports either of both HardTotals devices and the host file system, the application should set the **StorageTarget** property accordingly to tell where to write the data file.

If device needs to be able to write the pose and motion files to a HardTotals device, the **CapAssociatedHardTotalsDevice** property holds the open name of the associated HardTotals device.

## **Device Sharing**

The Gesture Control Device 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 accessing some properties or calling methods that update the device.
* See the “Summary” table for precise usage prerequisites.

# Properties (UML attributes)

## **AutoMode Property**

**Syntax AutoMode: *string* {read-write, access after open-claim-enable}**

**Remarks** Indicates automatic control mode ID. Valid values are the empty string "" or one of the **AutoModeList** properties listed.

If you set one of the properties described in the **AutoModeList** property for this property, the automatic control mode will be enabled in the set mode.

Setting the empty character "" disables the automatic control mode.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **AutoModeList** Property

## **AutoModeList Property**

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

**Remarks** Comma-separated list of joint automatic control IDs supported by the device.

For example, in conjunction with the camera, if the mode of tracking the face of a person by moving only the joint of Joint01, this is “FaceTrack\_Joint01”.

Another example, in conjunction with the camera, if the mode of tracking the face of a person by moving all joints are supported, this is “FaceTrack\_ALL".

(Content and order are dependent on the device.)

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **AutoMode** Property.

## **CapAssociatedHardTotalsDevice Property**

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

**Remarks** Holds the open name of the associated HardTotals device if the device is able to write to such devices which is the case if **CapStorageTarget** is either CST\_ALL or CST\_HARDTOTALS\_ONLY. If **CapStorageTarget** is CST\_HOST\_ONLY this property value must be the empty string.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapStorageTarget** Property

## **CapMotion Property**

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

**Remarks** If true, the device supports making the motion function. Otherwise, it is false.

When this property is false, **startMotion** method, **createMotion** method is not available.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **startMotion** Method, **createMotion** Method.

## **CapMotionCreation Property**

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

**Remarks** If true, the device supports motion registration function.

If false, the device does not support motion registration function.

If this property is FALSE, the **createMotion** method is not available.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **createMotion** Method.

## **CapPose Property**

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

**Remarks** If true, the device supports pose function. Otherwise, it is false.

When this property is FALSE, value **PoseCreationMode** property cannot be changed, and **startPose** method, **createPose** method are not available.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **PoseCreationMode** Property, **startPose** Method, **createPose** Method.

## **CapPoseCreation Property**

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

**Remarks** If true, the device supports pose registration function.

If false, the device does not support pose registration function.

If this property is FALSE, you cannot use the **createPose** method to change the **PoseCreationMode** property.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **PoseCreationMode** Property, **createPose** Method.

## **CapStorageTarget Property**

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

**Remarks** This is an enumeration and announces where the device is able to write the recorded motion and/or pose data file to.  
It holds one of the following values.

**Value Meaning**

CST\_HARDTOTAL\_ONLY Only an associate **HardTotals** device is supported.

CST\_HOST\_ONLY Only the host’s file system is supported.

CST\_ALL Both, the associated **HardTotals** device and the host’s file system is supported.

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

If a HardTotals device is supported the **StorageTarget** the property value should be CST\_HARDTOTAL\_ONLY or CST\_ALL, and the property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **StorageTarget** Property, **CapAssociatedHardTotalsDevice** Property

## **MotionList Property**

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

**Remarks** Comma-separated list of motion IDs defined on the device.

For example, “bowing, welcoming, clapping,…”

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

## **PoseCreationMode Property**

**Syntax PoseCreationMode: *boolean* {read-write, access after open-claim-enable}**

**Remarks** If true, pose registration function is enabled.

If false, pose registration function is invalid.

When this property is set to true, pose registration function is enabled. When false is set, the pose registration function is disabled.

This property is initialized to false when you first enable the device after calling the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **CapPose** Property, **CapPoseCreation** Property.

## **PoseList Property**

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

**Remarks** A comma-separated list of pose IDs defined on the device.

For example, “surprise, bow, think,….”

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

## **StorageTarget Property**

**Syntax StorageTarget: *int32* {read-write, access after open-claim-enable}**

**Remarks** This is an enumeration and defines where the device writes the recorded motion and/or pose data file to. Should be set before an appropriate method call.  
It holds one of the following values.

**Value Meaning**

ST\_HARDTOTAL The motion and/or pose data file is written to the associated HardTotals device. The property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

ST\_HOST The motion and/or pose data file is written to the host’s file system.

ST\_HOST&HARDTOTAL

The motion and/or pose data file is written to the associated **HardTotals** device and host’s file system. The property **CapAssociatedHardTotalsDevice** holds the open name of the associated HardTotals device.

This property is initialized by the **open** method according to the value hold by **CapStorageTarget**. If **CapStorageTarget** has the value CST\_ALL it is initialized to ST\_HOST.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**Value Meaning**

E\_ILLEGAL An invalid value was specified, or recording is ongoing.

**See Also CapStorageTarget** Property, **CapAssociatedHardTotalsDevice** Property

## **JointList Property**

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

**Remarks** Comma-separated list of joint information supported by the device.

Each piece of joint information consists of the following information and is shown in the following order, separated by a colon (":").

**Parameter Description**

*JointID* Indicates a unique ID in the service that identifies the

joint.

Position range availability. If position range is 0, the

Joint does not have the position range. If position range

is 1, the joint holds the position range.

For example, arm joint has a range of rotation width

but wheel for movement does not have the range of

movement amount.

If there is a device with joints that supports pitch, roll,

yaw and wheels that supports rotating and moving back

and forth. In this case they are indicated as follows.

"Joint01\_Pitch:1, Joint01\_Roll:1, Joint01\_Yaw:1,

Wheel\_Turn:0, Wheel\_Move:0"

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

## **Table of Gesture Control Device Listed Items in Property**

|  |  |  |  |
| --- | --- | --- | --- |
| Property Name | Item ID, File Name, Name | Parameter | Remarks |
| AutoModeList | Face Track | Joint01 Joint\_ALL |  |
| Chase | Joint01, Wheel01, Wheel02 Joint\_All, Wheel\_ALL, |
| MotionList | Bowing, Welcoming, Clapping, Farewelling01, Farewelling02, Greeting01, Greeting02 , |  |  |
| PoseList | Surprise, Bow01, Bow02, Think01, Think02 Doubt01, Doubt02 |  |  |
| JointList | Joint | Pitch Roll Yaw |  |
| Wheel | Turn Move Back Move Forth |

# Methods (UML operations)

## **createMotion Method**

**Syntax** **createMotion (fileName: *string*, poseList: *string*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*fileName* Specify the motion file name recorded as motion.

*poseList* Specify the comma-separated list of pose information to be registered.

**Remarks**  A motion file can be created and recorded by specifying the pose defined in the created pose file or the pose defined in the device and creating it as a series of continuously changing actions. The place where the motion file is recorded is the area value in the  **StorageTarget** property.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL fileName is too long or contains unusable characters.

E\_EXISTS fileName already exists.

## **createPose Method**

**Syntax** **createPose (fileName: *string*, time: *int32*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*fileName* Specify the pose file name to record the pose.

*time* Specify the time to reach the pose position.

**Remarks** Record the position of each joint in the pose file.

Before calling this method, you need to set the **PoseCreationMode** property to TRUE and enable pose registration mode.

The place where the pose file is recorded is the area value in the StorageTarget property.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL FileName is too long or contains unusable characters. Or PoseCreationMode is FALSE.

E\_EXISTS FileName already exists.

**See Also** **PoseCreationMode** Property.

## **getPosition Method**

**Syntax** **getPosition (jointID: *string*, position:out *int32 by reference*):   
void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*jointID* Specify the one of the joint ID values that are listed in the JointList property. And specified JointList property should be the position range present one.

*position* Store the specified value as the position associated with jointID.

**Remarks** Acquires the value of position specified by jointID and stores it as position value.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **JointList** Property.

## **setPosition Method**

**Syntax** **setPosition (positionList: *string,* time: *int32,* absolute: *boolean*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*positionList* Specify the position information in a comma-separated list.

*time* Specify the time to device control completion in seconds. If this value is too small, it will be changed to an appropriate value depending on the service.

*absolute* If true, the specified position indicates the absolute value. If false, the specified position indicates relative value.

Each position information specified in the positionList consists of the following information and is shown in the following order separated by a colon (":").

**Parameter Description**

*jointID* Specify the joint ID. Specify one of the values listed in the **JointList** property. However, it must be an ID whose position range is present.

*position* Specify the position to be set. Valid values range from -100 to 100.

100 represents the limit value in the positive direction of the target joint, and -100 represents the limit value in the negative direction.

If absolute is a relative value (false) and the value specified here exceeds the limit value, it will be changed to an appropriate value by the service

For example, to move Yow of Joint01 up to the limit of the positive direction and move Pitch of Joint01to the middle, specify as follows.

"Joint01\_Yaw:100,Joint01:Pitch:0"

**Remarks** The joint position is set with the contents specified in PositionList and device control is started so that device control is completed at the time specified by Time.

Joints that can be specified with this method are only those that have a position range.

Check the **JointList** property for the presence or absence of the position range.

This method is executed asynchronously. To terminate the operation prematurely, call the **stopControl** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **JointList** Property, **stopControl** Method.

## **setSpeed Method**

**Syntax** **setSpeed (speedList: *string*, time: *int32*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*speedList* Specify speed information in a comma-separated list.

*time* Specify the time to device control in seconds. If you specify FOREVER(-1), it will continue to operate until you call the **stopControl** method.

Each speed information specified in the SpeedList consists of the following

information, and it is shown in the following order separated by a colon (":").

**Parameter Description**

*jointID* Specify the joint ID. Specify one of the values listed in the **JointList** property.

*speed* Specify the speed to set. Valid values range from -100 to 100.100 represents the maximum speed in the positive direction of the target joint, and -100 represents the maximum speed in the negative direction.

For example, to move Wheel's X at the maximum speed in the positive direction and Y at the Wheel at half the speed in the negative direction, specify as follows.

"Wheel\_X:100, Wheel\_Y:-50"

**Remarks** It sets the speed of the joint with the contents specified by speedList and performs device control for the time specified by time.

This method is executed asynchronously. To terminate the operation prematurely, call the **stopControl** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **JointList** Property, **stopControl** Method.

## **startMotion Method**

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

**Parameter Description**

*fileName* Prior to start this method, need to specify the name of the motion file or the motion ID value that is listed in the **MotionList** property.

**Remarks** Start the motion defined by fileName or motion defined by the device. This method is executed asynchronously and when the device successfully completes a request, an **OutputCompleteEvent** is enqueued and a property of corresponding event’s OutputID is placed into the **OutputID** property. The application should compare the returned **OutputCompleteEvent** property outputID value set by this method to track what data has been sent to device.

Motion files are placed in the area as the value of **StorageTarget** property.

To terminate motion control prematurely, call the **stopControl** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

E\_NOEXIST File does not exist.

**See Also** **MotionList** Property.

## **startPose Method**

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

**Parameter Description**

*fileName* Specify the name of the pose file to start. Or one of the pose ID lists listed in the **PoseList** property.

**Remarks** Start the pose defined by the pose file or device specified by fileName. This method is executed asynchronously and when the device successfully completes a request, an **OutputCompleteEvent** is enqueued and a property of corresponding event’s OutputID is placed into the **OutputID** property. The application should compare the returned **OutputCompleteEvent** property **OutputID** value set by this method to track what data has been sent to device.

Pose files are placed in the area as the values of **StorageTarget** property.

To terminate pause control prematurely, call the **stopControl** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

E\_NOEXISTS File does not exist.

**See Also** **PoseList** Property, **stopControl** Method.

## **stopControl Method**

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

**Parameter Description**

*outputID* Specify the value of the **OutputID** property you wish to terminate.

**Remarks** Stop the control specified for outputID. When device successfully complete the request, and **OutputCompleteEvent** is enqueued. A property of this event contains the outputID of the completed request. The application should compare the returned **OutputCompleteEven**t property OutputID value with OutputID value set by this method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **setPosition** Method, **setSpeed** Method, **startPose** Method, **startMotion** Method.

# Events (UML interfaces)

## **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 Sound Player Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** method

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent  
ErrorCode : *int32*{read-write}  
ErrorCodeExtended : *int32*{read-write}  
ErrorLocus : *int32*{read-write}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that a Gesture Control Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is re-tried, and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is re-tried, and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **OutputCompleteEvent**

**<<event>> upos::events::OutputCompleteEvent  
 OutputID: int32{read-only}**

**Description** Notify the application that the queued output request associated with the *outputID* property has completed successfully.

**Attributes** This event contains the following attributes:

**Attribute 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 Intro-25

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
 Status : *int32* {read-only}**

**Description** *Notifies the application that there is an operation status change or a status of the Gesture Control device.*

**Attributes**  This event contains the following attribute:

**Attributes Type Description**

*Status* *int32* Indicates a change of operation status of sound player device

.***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process. See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

GCTL\_SUE\_START\_MOTION It will be notified when Gesture Motion start.

GCTL\_SUE\_STOP\_MOTION It will be notified when Gesture Motion stop.

**Remarks** Enqueued when the Gesture Control Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# Chapter 46

# Device Monitor

This Chapter defines the Device Monitor device category.

## **Summary**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pro Properties (UML attributes)** | |  | |  | |  | |  | |
| ***Common*** | | **Type** | | **Mutability** | | ***Version*** | | ***May Use After*** | |
| **AutoDisable:** | | *boolean* | | {read-write} | | 1.16 | | open | |
| **CapCompareFirmwareVersion:** | | *boolean* | | {read-only} | | 1.16 | | open | |
| **CapPowerReporting:** | | *int32* | | {read-only} | | 1.16 | | open | |
| **CapStatisticsReporting:** | | *boolean* | | {read-only} | | 1.16 | | open | |
| **CapUpdateFirmware:** | | *boolean* | | {read-only} | | 1.16 | | open | |
| **CapUpdateStatistics:** | | *boolean* | | {read-only} | | 1.16 | | open | |
| **CheckHealthText:** | | *string* | | {read-only} | | 1.16 | | open | |
| **Claimed:** | | *boolean* | | {read-only} | | 1.16 | | open | |
| **DataCount:** | | *int32* | | {read-only} | | 1.16 | | open | |
| **DataEventEnabled:** | | *boolean* | | {read-write} | | 1.16 | | open | |
| **DeviceEnabled:** | | *boolean* | | {read-write} | | 1.16 | | open & claim | |
| **FreezeEvents:** | | *boolean* | | {read-write} | | 1.16 | | open | |
| **OutputID:** | | *int32* | | {read-only} | | 1.16 | | Not Supported | |
| **PowerNotify:** | | *int32* | | {read-write} | | 1.16 | | open | |
| **PowerState:** | | *int32* | | {read-only} | | 1.16 | | open | |
| **State:** | | *int32* | | {read-only} | | 1.16 | | -- | |
| **DeviceControlDescription:** | | *string* | | {read-only} | | 1.16 | | -- | |
| **DeviceControlVersion:** | | *int32* | | {read-only} | | 1.16 | | -- | |
| **DeviceServiceDescription:** | | *string* | | {read-only} | | 1.16 | | open | |
| **DeviceServiceVersion:** | | *int32* | | {read-only} | | 1.16 | | open | |
| **PhysicalDeviceDescription:** | | *string* | | {read-only} | | 1.16 | | open | |
| **PhysicalDeviceName:** | | *string* | | {read-only} | | 1.16 | | open | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  | |  | |  | |
| ***Specific*** | ***Type*** | ***Mutability*** | | ***Version*** | | ***May Use After*** | |
| **DeviceData:** | *string* | {read-only} | | 1.16 | | open, claim & enable | |
| **DeviceList:** | *string* | {read-only} | | 1.16 | | open | |
| **MonitoringDeviceList:** | *string* | {read-only | | 1.16 | | open, claim & enable | |
|  |  |  | |  | |  | |
| **Me Methods (UML operations)** | | | | | | |  |
| ***Common*** | | | | | | |  |
| ***Name*** | | | | | | | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | | | | | | | 1.16 |
| **close ( ): void {raises-exception, use after open}** | | | | | | | 1.16 |
| **claim (timeout:** *int32***): void {raises-exception, use after open}** | | | | | | | 1.16 |
| **release ( ): void {raises-exception, use after open, claim}** | | | | | | | 1.16 |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | | | | | | | 1.16 |
| **clearInput ( ): void { }** | | | | | | | 1.16 |
| **clearInputProperties ( ): void { }** | | | | | | | 1.16 |
| **clearOutput ( ): void { }** | | | | | | | Not  supported |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, enable}** | | | | | | | 1.16 |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | | | | | | | 1.16 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | | | | | | | 1.16 |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | | | | | | | 1.16 |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, enable}** | | | | | | | 1.16 |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, enable}** | | | | | | | 1.16 |
| ***Specific*** | | | | | | |  |
| **addMonitoringDevice (deviceID:** *string***, monitoringMode:** *int32,* **boundary:** *int32,* **subBoundary:** *int32,* **intervalTime:** *int32* **):  void {raises-exception, use after open, claim, enable}** | | | | | | | 1.16 |
| **deleteMonitoringDevice (deviceID:** *string* **): void {raises-exception, use after open, claim, enable}** | | | | | | | 1.16 |
| **clearMonitoringDevices ( ): void {raises-exception, use after open, claim, enable}** | | | | | | | 1.16 |
| **getDeviceValue (deviceID:** *string***, inout value:** *int32***) void {raises-exception, use after open}** | | | | | | | 1.16 |
| **Events (UML interfaces)** |  | |  | |  | | |
| ***Name*** | ***Type*** | | ***Mutability*** | | ***Version*** | | |
|  |  | |  | |  | | |
| **upos::events::DataEvent** |  | |  | | 1.16 | | |
| **Status:** | *int32* | | {read-only} | |  | | |
|  |  | |  | |  | | |
| **upos::events::DirectIOEvent** |  | |  | | 1.16 | | |
| **EventNumber:** | *int32* | | {read-only} | |  | | |
| **Data:** | *int32* | | {read-write} | |  | | |
| **Obj:** | *object* | | {read-write} | |  | | |
|  |  | |  | |  | | |
| **upos::events::ErrorEvent** |  | |  | | 1.16 | | |
| **ErrorCode:** | *int32* | | {read-only} | |  | | |
| **ErrorCodeExtended:** | *int32* | | {read-only} | |  | | |
| **ErrorLocus:** | *int32* | | {read-only} | |  | | |
| **ErrorResponse:** | *int32* | | {read-write} | |  | | |
|  |  | |  | |  | | |
| **upos::events::OutputCompleteEvent** |  | | ***Not Supported*** | |  | | |
|  |  | |  | |  | | |
| **upos::events::StatusUpdateEvent** |  | |  | | 1.16 | | |
| **Status:** | *int32* | | {read-only} | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |

# General Information

The Device Monitor programmatic name is "Device Monitor".

## **Capabilities**

The Device Monitor Device has the following capability:

* Get values measured by various devices.
* Notify the application of changes in values measured by various devices.

## **Device Monitor Class Diagram**

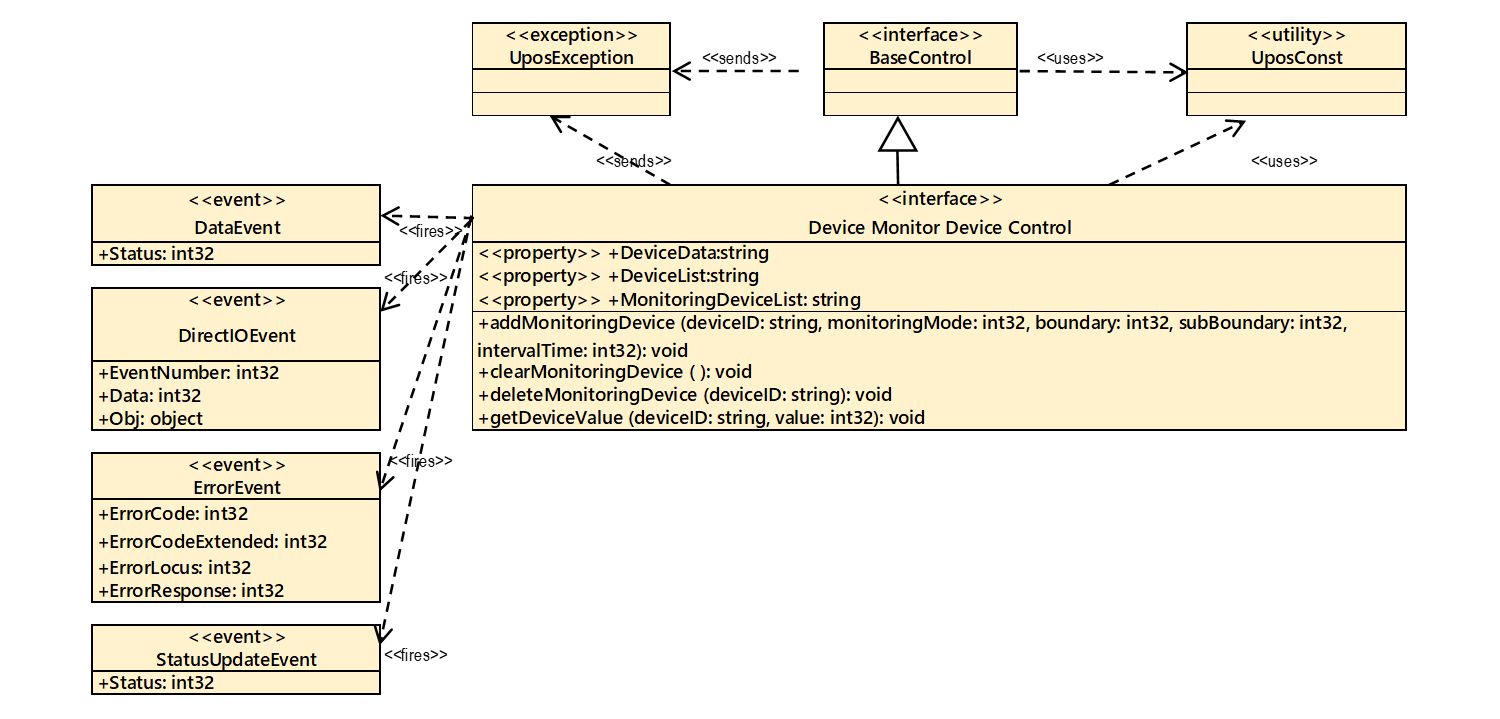
The following diagram shows the relationships between the Device Monitor classes.

Fig. Chap. 46-1 Device Monitor Class Diagram

# Model

The Device Monitor follows the general “Device Input Model” for event-driven input:

* The Device Monitor supports monitoring of values measured by multiple devices connected to the device. A device that can be monitored and its type / value unit is listed in the **DeviceList** property.
* Device Monitor receives a change in the value measured by the device set as the monitoring target, and generates a **DataEvent** when it matches the specified condition.
* To add a device to be monitored, specify the monitoring mode with the **addMonitoringDevice** method and add it. For details on monitoring mode, see the description of **addMonitoringDevice** method.
* If the **AutoDisable** property is true, the device will automatically disable itself when a **DataEvent** is enqueued.
* An enqueued **DataEvent** can be delivered to the application when the **DataEventEnabled** property is true and other event delivery requirements are met. Just before delivering this event, data is copied into corresponding properties, and further data events are disabled by setting **DataEventEnabled** to false. This causes subsequent input data to be enqueued while the application processes the current input and associated properties. When the application has finished processing the current input and is ready for more data, it reenables events by setting **DataEventEnabled** to true.
* An **ErrorEvent** (or events) is enqueued if an error occurs while gathering or processing input, and is delivered to the application when **DataEventEnabled** is true and other event delivery requirements are met.
* The **DataCount** property can be read to obtain the total number of enqueued **DataEvents**.
* All enqueued input may be deleted by calling **ClearInput** method. See the **ClearInput** method description for more details.
* All data properties that are populated as a result of firing a **DataEvent** or **ErrorEvent** can be set back to their default values by calling the **clearInputProperties** method.
* The notified data is stored in the **DeviceData** property.
* In the Device Monitor device control, the measured values of the devices are managed most of cases with the int32 type integers, but some are decimals. In that case, the decimals are implicit, and the actual value can be calculated by dividing the measured value by the coefficient of each device that can be obtained in the **DeviceList** property.
* The application will be informed about any status change with a **StatusUpdateEvent**,

also, all corresponding status properties will be updated before event delivery.

## **Device Sharing**

The Device Monitor 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 the device begins reading input, or before calling methods that manipulate the device.

See the “Summary” table for precise usage prerequisites.

# Properties (UML attributes)

## **DeviceData Property**

**Syntax** **DeviceData: *string* {read-only, access after open-claim-enable}**

**Remarks** Measurement information of the device that matches the condition registered by **addMonitoringDevice** method is set.

Each measurement information consists of the following information and is shown in the following order, separated by a colon (":").

**Parameter Description**

DeviceID The target device ID.

Measured value Measurement value of the device. The measured value is represented by an integer type. To convert it to an actual value, divide the measured value by the coefficient acquired by the **DeviceList** property.

For example,"Device01:365"

Its value is set prior to a **DataEvent** being delivered to the application.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

## **DeviceList Property**

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

**Remarks** Contains the comma-delimited list of device information that are supported by the device.

Each object information consists of the following information and is shown in the following order, separated by a colon (":").

**Parameter Description**

DeviceID Indicates a unique ID in the service that identifies the device.

Type Indicates the device type. For example, if it is a touch sensor it is expressed as "Touch Sensor" and so on. However, this value depends on the service.

Unit Indicates the unit of value held by various devices. For example, it is expressed as "on / off" for a touch sensor, "rad / s" for a gyroscope. However, this value depends on the service.

Coefficient Indicates the coefficient for calculating the actual measured value held by various devices. The **DeviceData** property and the measured value of the device that can be obtained with the **GetDeviceValue** method are expressed as integers, but by dividing this value by the coefficient it is the actual value. Example: Device value = 365, coefficient = 10, actual value = 36.5 For example, if one device supports one touch sensor and one gyroscope, it will be as follows. "Touch 01: Touch Sensor: ON/OFF: 1, GyroX: Gyroscope: rad/s: 100000, GyroY: Gyroscope: rad/s: 100000, GyroZ: Gyroscope: rad/s: 100000"

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 Intro-20.

**See Also** **DeviceData** Property, **addMonitoringDevice** Method, **getDeviceValue** Method.

## **MonitoringDeviceList Property**

**Syntax** **MonitoringDeviceList: *string* {read-only, access after open-claim-enable}**

**Remarks** Contains the comma-delimited list of monitoring information on registered devices that are supported by the device.

Each monitoring information consists of the following information and is shown in the following order, separated by a colon (":").

**Parameter Description**

DeviceID Registered devices ID.

Monitoring mode Registered monitoring mode.

Boundary Registered boundary value. This value is set to 0 when the monitoring mode does not require a boundary value.

Sub boundary Registered sub boundary value. This value is set to 0 when the monitoring mode does not require a sub boundary value.

Interval Registered interval. (millisecond)

For example, if you set monitoring targets as follows,

[Monitor target 1]

Device ID = Device 01, monitoring mode = DMON\_MM\_UPDATE,

boundary line = 0, sub boundary line = 0, interval time = 0

[Monitor target 2]

Device ID = Device 02, monitoring mode = DMON\_MM\_STRADDLED,

boundary line = 365, sub boundary line = 0, interval time = 500

The values shown are as follows.

"Device01:0:0:0:0, Device02:1:365:0:500"

This property is initialized by the **open** method. It is also updated by calling

**addMonitoringDevice** method, **deleteMonitoringDevice** method,

**clearMonitoringDevice** method.

**Errors** A UposException may be thrown when this property is accessed. For further information, see **“Errors"** on page Intro-20.

**See Also** **addMonitoringDevice** Method, **deleteMonitoringDevice** Method, **clearMonitoringDevice** Method.

# Methods (UML operations)

## **addMonitoringDevice Method**

**Syntax** **addMonitoringDevice (deviceID:*string*, monitoringMode:*string,* boundary:*int32*, subBoundary:*int32*, intervalTime:*int32*):  
 void{raises-exception, use after open-claim-enable}**

**Parameter Description**

deviceID The deviceID of the monitored device. Valid values are one of the device ID lists listed in the **DeviceList** property.

monitoringMode Specify the monitoring mode for monitoring.

boundary Specify the boundary value to be monitored.

subBoundary Specify the sub boundary value to be monitored. This value must be less than Boundary. intervalTime Specify the interval in milliseconds between the occurrence of the event and the start of the next monitoring.  
 The monitoring modes specified for MonitoringMode are as follows.

**Value Description**

DMON\_MMODE\_UPDATE  
 Every time the measured value of the target device is updated, an event is notified. When set to this mode, the values of the argument boundary and subBoundary are ignored.

DMON\_MMODE\_STRADDLED  
 When the measured value of the target device crosses the value of the argument boundary, it notifies the event. In addition, when the measured value matches the value of boundary, it notifies the event even when it changes from the matched state. When set to this mode, the value of the argument subBoundary is ignored.

DMON\_MMODE\_HIGH  
 When the measured value of the target device becomes equal to or larger than the value of the argument Boundary, it notifies the event. Even if the measured value is updated and it was again equal to or greater than the value of boundary, we will notify the event each time. When set to this mode, the value of the argument subBoundary is ignored.

DMON\_MMODE\_LOW  
 Notifies the event when the measured value of the target device becomes less than or equal to the value of the argument boundary. Even when the measured value is updated and it was again less than the value of boundary, we will notify the event each time. When set to this mode, the value of the argument subBoundary is ignored.

DMON\_MMODE\_WITHIN  
 It notifies the event while the measured value of the target device is within the range specified by the argument boundary and subBoundary.  
 Even if the measured value is updated and its value is within the range again, the event is notified each time.

DMON\_MMODE\_OUTSIDE  
 It notifies the event while the measured value of the target device is outside the range specified by the argument boundary and subBoundary.  
 Even if the measured value is updated and its value was out of range again, we will notify the event each time.

DMON\_MMODE\_POLLING  
 It notifies the measured value of the target device at the interval specified by intervalTime.  
 When set to this mode, the values of the argument boundary and subBoundary are ignored.

**Remarks** Add the device specified by deviceID to the monitoring target.  
The monitoring mode is specified for monitoringMode, but there are monitoring modes not supported by some devices. In that case, E\_ILLEGAL is raised as the UPOS exception.  
Devices added by this method will be added to the list of **MonitoringDeviceList** properties. If a device to be monitored is specified, it will be changed to a new condition. To exclude the added device from the monitoring target, call **deleteMonitoringDevice** method or **clearMonitoringDevice** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20. Some possible values of the exception’s ErrorCode property are:

**Value Description**

E\_ILLEGAL An invalid value was specified, or unsupported operation with the Device

**See Also** **DeviceList** Property, **MonitoringDeviceList** Property, **deleteMonitoringDevice** Method, **clearMonitoringDevice** Method, **DataEvent**.

## **deleteMonitoringDevice Method**

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

**Parameter Description**

deviceID Specify the device ID of the device to be excluded from monitoring targets.

**Remarks** Exclude the device specified by deviceID from monitoring targets.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.  
Some possible values of the exception’s ErrorCode property are:

**Value Description**

E\_ILLEGAL An invalid value was specified, or unsupported operation

with the Device.

**See Also** **AddMonitoringDevice** Method.

## **clearMonitoringDevices Method**

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

**Remarks** Exclude all devices to be monitored.

**Errors** A UposException may be thrown when this method is invoked.  
For further information, see “Errors" on page Intro-20.

**See Also addMonitoringDevice** Method.

## **getDeviceValue method**

**Syntax getDeviceValue (deviceID: *string****,* **pValue: \**int32*):  
 void{raises-exception, use after open}**

**Parameter Description**

*deviceID* Specify the device ID of the device from which the measurement value is to be acquired. Specify one of the device ID lists listed in the **DeviceList** property.

*pValue*  Pointer that stores measurement values obtained from the device.

**Remarks** Get the measured value of the device specified by deviceID. The retrieved value is stored in pValue.

**Errors** A UposException may be thrown when this method is invoked. For further information, see **“Errors"** on page Intro-20.  
Some possible values of the exception’s ErrorCode property are:

**Value Description**

E\_ILLEGAL An invalid value was specified, or unsupported operation with the Device.

**See Also** **DeviceList** Property.

# Events (UML interfaces)

## **DataEvent**

**<<event>> upos::events::DataEvent**

**Status : *int32*{read-only}**

**Description** Notifies the application when data from the Device Monitor device is available to be read.

**Attributes** This event contains the following attributes:

**Attribute Type Description**

*Status int32 Set to 0.*

**Remarks** Before this event is delivered, the individual recognition information is enqueued into the area that is indicated by the **addMonitoringDevice** method.

**See Also** **addMonitoringDevice** method.

## **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 Device Monitor Device Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** method

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent  
ErrorCode : *int32*{read-write}  
ErrorCodeExtended : *int32*{read-write}  
ErrorLocus : *int32*{read-write}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that a Device Monitor Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is re-tried, and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is re-tried, and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period of time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
 Status : *int32* {read-only}**

**Description** *Notifies the application that there is an operation status change or a status of the Device Monitor device.*

**Attributes**  This event contains the following attribute:

**Attributes Type Description**

*Status* *int32* Indicates a change in the Device Monitor status of the unit.

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process. See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

DMON\_SUE\_START\_MONITERING  
 It will be notified when Device Monitoring start.

DMON\_SUE\_STOP\_MONITORING  
 It will be notified when Device Monitoring stop.

**Remarks** Enqueued when the Device Monitor Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# Chapter 47

# Graphic Display

This Chapter defines the Graphic Display device category.

# Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties (UML attributes)** |  |  |  |  |
| ***Common*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** |
| **AutoDisable:** | *boolean* | {read-write} | 1.16 | open |
| **CapCompareFirmwareVersion:** | *boolean* | {read-only} | 1.16 | open |
| **CapPowerReporting:** | *int32* | {read-only} | 1.16 | open |
| **CapStatisticsReporting:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateFirmware:** | *boolean* | {read-only} | 1.16 | open |
| **CapUpdateStatistics:** | *boolean* | {read-only} | 1.16 | open |
| **CheckHealthText:** | *string* | {read-only} | 1.16 | open |
| **Claimed:** | *boolean* | {read-only} | 1.16 | open |
| **DataCount:** | *int32* | {read-only} | 1.16 | open |
| **DataEventEnabled:** | *boolean* | {read-write} | 1.16 | open |
| **DeviceEnabled:** | *boolean* | {read-write} | 1.16 | open, & claim |
| **FreezeEvents:** | *boolean* | {read-write} | 1.16 | open |
| **OutputID:** | *int32* | {read-only} | 1.16 | open |
| **PowerNotify:** | *int32* | {read-write} | 1.16 | open |
| **PowerState:** | *int32* | {read-only} | 1.16 | open |
| **State:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceControlDescription:** | *string* | {read-only} | 1.16 | -- |
| **DeviceControlVersion:** | *int32* | {read-only} | 1.16 | -- |
| **DeviceServiceDescription:** | *string* | {read-only} | 1.16 | open |
| **DeviceServiceVersion:** | *int32* | {read-only} | 1.16 | open |
| **PhysicalDeviceDescription:** | *string* | {read-only} | 1.16 | open |
| **PhysicalDeviceName:** | *string* | {read-only} | 1.16 | open |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Properties (Continued)** |  |  |  |  | | |
| ***Specific*** | ***Type*** | ***Mutability*** | ***Version*** | ***May Use After*** | | |
| **CapAssociatedHardTotalsDevice:** | *string* | {read-only} | 1.16 | open | | |
| **CapBrightness:** | *boolean* | {read-only} | 1.16 | open | | |
| **CapImageType:** | *boolean* | {read-only} | 1.16 | open | | |
| **CapStorageTarget:** | *int32* | {read-only} | 1.16 | open | | |
| **CapURLBack:** | *boolean* | {read-only} | 1.16 | open | | |
| **CapURLForward:** | *boolean* | {read-only} | 1.16 | open | | |
| **CapVideoType:** | *boolean* | {read-only} | 1.16 | open | | |
| **CapVolume:** | *boolean* | {read-only} | 1.16 | open | | |
| **Brightness:** | *int32* | {read-write} | 1.16 | open, claim & enable | | |
| **DisplayMode:** | *int32* | {read-write} | 1.16 | open, claim & enable | | |
| **ImageType:** | *int32* | {read-write} | 1.16 | open, claim & enable | | |
| **ImageTypeList:** | *string* | {read-only} | 1.16 | open | | |
| **LoadStatus:** | *int32* | {read-only} | 1.16 | open | | |
| **StorageTarget:** | *int32* | {read-write} | 1.16 | open, claim & enable | | |
| **URL:** | *string* | {read-only} | 1.16 | open | | |
| **VideoType:** | *int32* | {read-write} | 1.16 | open, claim & enable | | |
| **VideoTypeList:** | *string* | {read-only} | 1.16 | open | | |
| **Volume:** | *int32* | {read-write} | 1.16 | open, claim & enable | | |
| **Methods (UML operations)** | | | | |  |
| ***Common*** | | | | |  |
| ***Name*** | | | | | ***Version*** |
| **open (logicalDeviceName:** *string***): void {raises-exception}** | | | | | 1.16 |
| **close ( ):**  **void {raises-exception, use after open}** | | | | | 1.16 |
| **claim (timeout:** *int32***): void {raises-exception, use after open}** | | | | | 1.16 |
| **release ( ): void {raises-exception, use after open, claim}** | | | | | 1.16 |
| **checkHealth (level:** *int32***): void {raises-exception, use after open, enable}** | | | | | 1.16 |
| **clearInput ( ): void { }** | | | | | 1.16 |
| **clearInputProperties ( ): void { }** | | | | | 1.16 |
| **Methods (UML operations)(Continued)** | | | | |  |
| **clearOutput ( ): void { }** | | | | | 1.16 |
| **compareFirmwareVersion (firmwareFileName:** *string*, **out** **result:** *int32***): void {raises-exception, use after open, enable}** | | | | | 1.16 |
| **directIO (command:** *int32****,* inout data:** *int32****,* inout obj:** *object***): void {raises-exception, use after open}** | | | | | 1.16 |
| **resetStatistics (statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | | | | | 1.16 |
| **retrieveStatistics (inout statisticsBuffer:** *string***): void {raises-exception, use after open, enable}** | | | | | 1.16 |
| **updateFirmware (firmwareFileName:** *string***): void {raises-exception, use after open, enable}** | | | | | 1.16 |
| **updateStatistics (statisticsBuffer:** *string***):** **void {raises-exception, use after open, enable}** | | | | | 1.16 |
| ***Specific*** | | | | |  |
| ***Name*** | | | | | ***Version*** |
| **cancelLoading ( ):  void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **goURLBack ( ):  void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **goURLForward ( ):  void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **loadImage (fileName:***string***):  void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **loadURL (uRL: *string* ):  void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **playVideo (fileName:***string***, loop:***boolean***):  void { raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **stopVideo ( ): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |
| **updateURLPage (): void {raises-exception, use after open, claim, enable}** | | | | | 1.16 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Events (UML interfaces)** | |  |  |  |
| ***Name*** | | ***Type*** | ***Mutability*** | ***Version*** |
|  | |  |  |  |
| **upos::events::DataEvent** | |  |  |  |
| **Status:** | |  | ***Not Supported*** |  |
|  | |  |  |  |
| **upos::events::DirectIOEvent** | |  |  | 1.16 |
| **EventNumber:** | | *int32* | {read-only} |  |
| **Data:** | | *int32* | {read-write} |  |
| **Obj:** | | *object* | {read-write} |  |
|  | |  |  |  |
| **upos::events::ErrorEvent** | |  |  | 1.16 |
| **ErrorCode:** | | *int32* | {read-only} |  |
| **ErrorCodeExtended:** | | *int32* | {read-only} |  |
| **ErrorLocus:** | | *int32* | {read-only} |  |
| **ErrorResponse** | | *int32* | {read-write} |  |
|  | |  |  |  |
| **upos::events::OutputCompleteEvent** | |  |  | 1.16 |
| **OutputID:** | *int32* | {read-only} |  | |
|  | |  |  |  |
| **upos::events::StatusUpdateEvent** | |  |  | 1.16 |
| **Status:** | | *int32* | {read-only} |  |

# General Information

The Graphic Display programmatic name is “Graphic Display”.

## **Capabilities**

The Graphic Display has the following capability:

Displays the specified image files.

Play the specified video.

Display the specified web page.

Notify the application of changes in the load status of the web page.

## **Graphics Display Class Diagram**

The following diagram shows the relationships between the Graphic Display classes.



Fig. Chap. 47-1 Graphic Display Class Diagram

# Model

The following display modes exist in the graphics control, and the model differs depending on the display mode:

* Image display mode
* Video display mode.
* Web display mode.

The application can change the display mode by changing the value of the **DisplayMode** property.

### Image Display Mode

The image display mode of the graphics control is as follows. Prior to start this mode, need to set the appropriate image type file value in the **ImageType** property from the listed values in the **ImageTypeList** property, if  **CapImageType** property is true. Then the application can call the **loadImage** method to display the image.

Raises **StatusUpdateEvent** at the status change timing of image load start with status GDSP\_SUE\_START\_IMAGE\_LOAD, and image load end with status GDSP\_SUE\_END\_IMAGE\_LOAD. The **ImageTypeList** property lists image files that the device can display.

Applications may need to support “**Hard Totals**” services as image files displaying with **loadImage** method might be placed in the area managed by the associated “**Hard Totals**" service device. If the **CapStorageTarget** is either CST\_ALL or CST\_HARDTOTALS\_ONLY, it is possible to store it in the Associated Hard Totals device and storage device’s open name is held in the **CapAssociatedHardTotalsDevice** property.

If device supports both Hard Totals device and the host file system, the application should set the **StorageTarget** property accordingly to tell where to write the image data file.

### Video Display Mode

The video display mode of Graphic Display follows the general device behavior model for asynchronous output devices. The graphics control of video display modes are as follows.

Prior to start this mode, need to set the appropriate video type file value in the **VideoType** property from the listed values in the **VideoTypeList** property, if **CapVideoType** property is true. Then the application can call the **playVideo** method to display the video. Also, the video being displayed is stopped by calling the **stopVideo** method.

Raises **StatusUpdateEvent** at the status change timing of start play video with status GDSP\_SUE\_START PLAY\_VIDEO, and stop play video with status GDSP\_SUE\_STOP\_PLAY\_VIDEO.

The Device validates the method parameters an error condition immediately if necessary. If the validation is successful, the Device does the following:

• Buffers the request in program memory, for delivery to the Physical Device as soon as the Physical Device can receive and process it.

• Sets the **OutputID** property to a unique integer identifier for this request.

• Returns as soon as possible.

When the Device successfully completes a request, an **OutputCompleteEvent** is enqueued for delivery to the application.

A property of this event contains the output ID of the completed request.

The application should compare the returned **OutputCompleteEvent** property OutputID value with the **OutputID** value set by the asynchronous process method call used to send the data in order to track what data has been successfully sent to the device.

If an error occurs while processing a request, an **ErrorEvent** is enqueued which will be delivered to the application after the events already enqueued, including **OutputCompleteEvents**. No further asynchronous output will occur until the event has been delivered to the application. If the response is ER\_CLEAR, then outstanding asynchronous output is cleared.

If the response is ER\_RETRY, then output is retried; note that if several outputs were simultaneously in progress at the time that the error was detected, then the Service may need to retry all of these outputs.

Asynchronous output is always performed on a first-in first-out basis. If the device supports concurrent playback, the request will be executed simultaneously. To check if the device supports simultaneous playback, check the **CapMultiPlay** property.

If the request is terminated before completion, due to reasons such as the application calling the **clearOutput** method, then no **OutputCompleteEven**t is delivered. You can also delete the output individually by calling the **stopVideo** method. Also, in this case **OutputCompleteEvent** will not be notified.

The video files that the device can display are listed in the **VideoTypeList** property.

Since video files to be displayed using the **playVideo** method must be placed in an area managed by the associated “**Hard Tota**ls" service device. If the **CapStorageTarget** is either CST\_ALL or CST\_HARDTOTALS\_ONLY, it is possible to store it in the Associated Hard Totals device and storage device’s open name is held in the **CapAssociatedHardTotalsDevice** property.

If device supports either or both Hard Totals device and the host file system, the application should set the **StorageTarget** property accordingly to tell where to write the image data file.

The video display mode of graphics control follows an asynchronous output model.

Raises **StatusUpdateEvent** if Graphic Display device power status or a device status changes are occurred during the video displaying.

### Web Display Mode

The web display mode of graphics control is as follows.

The application calls the **loadURL** method to display the web page.

Raises **StatusUpdateEvent** at the timing of Web page load start with status GDSP\_SUE\_START\_LOAD\_WEBPAGE, load finish with status GDSP\_SUE\_FINISH\_LOAD\_WEBPAGE, and load cancel with status GDSP\_SUE\_CANCEL\_LOAD\_WEBPAGE. And application can detect the web page loading status.

The latest loading status of the web page is stored in the **LoadStatus** property when **loadURL** method is called, and its URL information is stored in the **URL** property.

In case when **cancelLoading** method is called during the loading process, current accessed URL information will be stored in the **URL** property.

The graphics control web display mode follows an asynchronous output model.

## **Device Sharing**

The Graphic Display Device 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 accessing some properties or calling methods that update the device.

See the “Summary” table for precise usage prerequisites.

# Properties (UML attributes)

## **Brightness Property**

**Syntax Brightness: *int32* {read-write, access after open-claim-enable}**

**Remarks** Holds the brightness of screen. Legal values range from zero through 100.  
This property is initialized by the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **CapBrightness** Property.

## **CapAssociatedHardTotalsDevice Property**

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

**Remarks** Holds the open name of the associated HardTotals device if the device is able to write to such devices which is the case if **CapStorageTarget** is either CST\_ALL or CST\_HARDTOTALS\_ONLY. If **CapStorageTarget** is CST\_HOST\_ONLY this property value must be the empty string.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **CapStorageTarget** Property

## **CapBrightness Property**

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

**Remarks** If true, the application can change the screen brightness.  
If false, the application cannot change the screen brightness.  
This property is initialized by the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **Brightness** Property.

## **CapImageType Property**

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

**Remarks** If true, indicate the image type file to be used in this target device as the value of the **ImageType** property. Otherwise it is false. This property is initialized by the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **ImageType** Property**, ImageTypeList** Property

## **CapStorageTarget Property**

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

**Remarks** This is an enumeration and announces where the device is able to write the image data file to.  
It holds one of the following values.

**Value Meaning**

CST\_HARDTOTAL\_ONLY Only an associate **HardTotals** device is supported.

CST\_HOST\_ONLY Only the host’s file system is supported.

CST\_ALL Both, the associated **HardTotals** device and the host’s file system is supported.

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

If a HardTotals device is supported the storage target the property value should be CST\_HARDTOTAL\_ONLY or CST\_ALL, and the property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**See Also** **StorageTarget** Property, **CapAssociatedHardTotalsDevice** Property

## **CapURLBack Property**

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

**Remarks** If true, the previous page exists in the browsing history. Application can return to the previous page with **goURLBack** method.

If false, there is no previous page in the browsing history.

This property is initialized to false by the open method. Also, as the web page loading state changes, it is set by the device control.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **goURLBack** Method.

## **CapURLForward Property**

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

**Remarks** If true, the next page exists in the browsing history. Application can go to the next page with the **goURLForward** method.  
If false, there is no next page in the browsing history.  
This property is initialized to false by the open method. Also, as the web page loading state changes, it is set by the device control.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **goURLForward** Method.

## **CapVideoType Property**

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

**Remarks** If true, indicate the vide type value that can be used in this targeted graphics display device as the value of VideoType Property. Otherwise, it is false. This property is initialized by the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **VideoType** Property, **VideoTypeList** Property .

## **CapVolume Property**

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

**Remarks** If true, the application can change the volume of video.  
If false, the application cannot change the volume of video.  
This property is initialized by the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **Volume** Property.

## **DisplayMode Property**

**Syntax DisplayMode: *int32* {read-write, access after open-claim-enable}**

**Remarks** Holds the image and/or video displaying mode.

**Value Meaning**

GDSP\_DMODE\_HIDDEN

It is a mode to hide images and/or video

GDSP\_DMODE\_IMAGE\_FIT

It is a mode to display images. The displayed image is enlarged / reduced to the size that maintains the aspect and just enters the screen.

GDSP\_DMODE\_IMAGE\_FILL

It is a mode to display images.

The displayed image is scaled to the size that

maintains the aspect and covers the entire screen.

GDSP\_DMODE\_IMAGE\_CENTER

It is a mode to display images.

The displayed image is displayed in the center of the screen without changing the size.

GDSP\_DMODE\_VIDEO\_NORMAL

It is a mode to display video. The displayed video

will be displayed in the center of the screen without resizing.

GDSP\_DMODE\_VIDEO\_FULL

It is a mode to display video.

The displayed video will be displayed in full screen.

GDSP\_DMODE\_WEB

Display the web screen.

If application hide other modes and screens while displaying images, movies, or web, all displayed contents will be cleared. The movie will be stopped while the movie is playing.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **CapCaptureColorSpaceList** Property, **VideoCaptureMode** Property

## **ImageType Property**

**Syntax ImageType: *string* {read-write, access after open-claim-enable}**

**Remarks** Contains the image file type that are support by the device, if **CapImageType** property is true. For example, if the device supports BMP, then this property should be set to “BMP”. This property value should be set prior to execute the loadImage method. All of the capable image file types are listed in the ImageTypeList property. \*Notation contents may be different depending on the device. This property is initialized by the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **CapImageType** Property, **ImageTypeList** Property, **loadImage** Method.

## **ImageTypeList Property**

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

**Remarks** Contains the comma-delimited list of image file type that are support by the device. For example, if the device only supports BMP and JPEG, then this property should be set to “BMP,JPEG” One of value in the property should be set in the **ImageType** property, if CapImageType property is true, prior to execute the **loadImage** method.

\*Notation contents may be different depending on the device.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **CapImageType** Property, **ImageType** Property, **loadImage** Method.

## **LoadStatus Property**

**Syntax LoadStatus: *int32* {read-only, access after open-claim-enable}**

**Remarks** Holds loading state of web page.

The parameters to be set are as follows.

**Value Meaning**

GDSP\_LSTATUS\_START Start loading the web page.

GDSP\_LSTATUS\_FINISH It have finished loading the web page.

GDSP\_LSTATUS\_CANCEL It have canceled loading the web page

Its value is set prior to a **StatusUpdateEvent** being delivered to the application.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

## **StorageTarget Property**

**Syntax StorageTarget: *int32* {read-write, access after open-claim-enable}**

**Remarks** This is an enumeration and defines where the device writes the recorded image data file to. Should be set before an appropriate method call.  
It holds one of the following values.

**Value Meaning**

ST\_HARDTOTAL The image data file is written to the associated HardTotals device. The property **CapAssociatedHardTotalsDevice** holds the open name of the associated **HardTotals** device.

ST\_HOST The image data file is written to the host’s file system.

ST\_HOST&HARDTOTAL

The encoded data file is written to the associated **HardTotals** device and host’s file system. The property **CapAssociatedHardTotalsDevice** holds the open name of the associated HardTotals device.

This property is initialized by the **open** method according to the value hold by **CapStorageTarget**. If **CapStorageTarget** has the value CST\_ALL it is initialized to ST\_HOST.

**Errors** UposException may be thrown when this property is accessed.  
 For further information, see “Errors" on page Intro-20.

**Value Meaning**

E\_ILLEGAL An invalid value was specified, or recording is ongoing.

**See Also CapStorageTarget** Property, **CapAssociatedHardTotalsDevice** Property

## **URL Property**

**Syntax URL: *string* {read-only, access after open-claim-enable}**

**Remarks** When the **LoadStatus** property is GDSP\_LSTATUS\_START, the URL of the Web page that starts loading is set.

When the **LoadStatus** property is GDSP\_LSTATUS\_FINISH, the URL of the loaded Web page is set.

When the **LoadStatus** property is GDSP\_STATUS\_CALCEL, the URL of the canceled Web page is set.

Its value is set prior to a **StatusUpdateEvent** being delivered to the application.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **loadStatus** Property.

## **VideoType Property**

**Syntax VideoType: *int32* {read-write, access after open-claim-enable}**

**Remarks** Contains the video file type that are support by the device, if **CapVideoType** property is true. For example, if the device supports AVI MJPG, then this property should be set to “AVI MJPG”. This property value should be set prior to execute the **playVideo** method. All of the capable video file types are listed in the **VideoTypeList** property.  **\***Notation contents may be different depending on the device.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **CapVideoType** Property, **VideoType** Property, **playVideo** Method.

## **VideoTypeList Property**

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

**Remarks** Contains the comma-delimited list of video file type that are support by the device. if the device only supports AVI\_IYUV and AVI\_MJPG, then this property should be set to “AVI\_IYUV,AVI\_MJPG”.　One of value in the property should be set in the **VideoType** property, if CapImageType property is true, prior to execute the **playVideo** method.

\*Notation contents may be different depending on the device.

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

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

**See Also** **CapVideoType** Property, **VideoType** Property, **playVideo** Method.

## **Volume Property**

**Syntax Volume: *int32* {read-write, access after open-claim-enable}**

**Remarks** Holds the volume at playing video. Legal values range from zero through 100.  
This property is initialized by the **open** method.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

**See Also** **CapVolume** Property**, playVideo** Method.

# Methods (UML operations)

## **cancelURLLoading Method**

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

**Remarks** Cancel loading web page.

This method is executed asynchronously. The load status is reported by **StatusUpdateEvent** and **OutputCompleteEven**t or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL It is not loading.

## **goURLBack Method**

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

**Remarks** It returns to the previous page of browsing history.

This method is executed asynchronously. The load status is reported by **StatusUpdateEvent** and **OutputCompleteEvent** or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL There is no previous page in the browsing history.

**See Also** **CapURLBack** Property.

## **goURLForward Method**

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

**Remarks** Go to the next page of browsing history.

This method is executed asynchronously. The load status is reported by **StatusUpdateEvent** and **OutputCompleteEvent** or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL There is no next page in the browsing history.

**See Also** **CapURLForward** Property.

## **loadImage Method**

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

**Parameter Description**

*fileName* Specify the file name of the image to be loaded.

**Remarks** Load the specified image.

This method fails if the value of the **DisplayMode** Property is not set to GDSP\_DMODE\_IMAGE\_FIT, GDSP\_DMODE\_IMAGE\_FILL, or GDSP\_DMODE\_IMAGE\_CENTER.

Image files are located in the area as the stored values of the **StorageTarget** property.

This method is executed asynchronously. Image file loading status is reported by **StatusUpDateEvent** and **OutputCompleteEvent** or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported image file was specified.

E\_NOEXIST File does not exist.

**See Also** **DisplayMode** Property.

## **loadURL Method**

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

**Parameter Description**

*uRL* Specify the uRL of the web page to load.

**Remarks** Load the web page with the specified URL.

This method is executed asynchronously. The load status is reported by **StatusUpdateEvent** and **OutputCompleteEvent** or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified.

## **playVideo Method**

**Syntax** **playVideo (fileName: *string*, loop: *boolean*):   
 void {raises-exception, use after open-claim-enable}**

**Parameter Description**

*fileName* Specify the file name of the video to be played.

*loop* If true, loop playback is performed, and if false, loop playback is not performed.

**Remarks** Play the specified video that are loaded in the storage area by the **loadImage** method. All of loaded file images are listed in the **ImageTypeList** property.

If the value of the **DisplayMode** property is not set to GDSP\_DMODE\_VIDEO\_NORMAL, GDSP\_DMODE\_VIDEO\_FULL, this method will fail.

This method is executed asynchronously. To stop video displaying in the middle, call the **stopVideo** method.

Video files are located in the area as the stored values of the **StorageTarget** property

The video file playing status will be informed by the **StatusUpdateEvent.**

This method is executed asynchronously. Image file loading status and video file playing status are reported by **StatusUpdateEvent** and **OutputCompleteEvent** or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL An invalid value was specified. Or an unsupported video file was specified.

E\_NOEXIST File does not exist.

**See Also** **DisplayMode** Property.

## **stopVideo Method**

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

**Remarks** Stop the video being displayed.

This method is executed asynchronously. Inage file loading status is reported by **StatusUpdateEven**t and **OutputCompleteEvent** or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL The Video is not playing.

**See Also** **startVideo** Method.

## **updateURLPage Method**

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

**Remarks** Reload the current web page.

This method is executed asynchronously. The load status is reported by **StatusUpdateEvent** and **OutputCompleteEvent** or **ErrorEvent**.

**Errors** A UposException may be thrown when this method is invoked. For further information, see “**Errors**" on page Intro-20.

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

**Value Meaning**

E\_ILLEGAL Web page loading.

# Events (UML interfaces)

## **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 Sound Player Service to provide events to the application that are not otherwise supported by the device 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 attribute is settable.

*Obj object* Additional data whose usage varies by the *EventNumber* and the Service. This attribute 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 programform being used with other vendor’s devices which may not have any knowledge of the Service’s need for this event.

**See Also** **“Events"** on page Intro-19, **directIO** method

## **ErrorEvent**

**<<event>> upos::events:: ErrorEvent  
ErrorCode : *int32*{read-write}  
ErrorCodeExtended : *int32*{read-write}**

**ErrorLocus : *int32*{read-write}  
\*pErrorResponse : *int32*{read-write}**

**Description** Notifies the application that a Graphic Display Device error has been detected and 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. See a list of Error Codes on page 20.

*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. If EL\_OUTPUT is specified. An error occurred during asynchronous action.

*ErrorResponse* *int32* Pointer to the error event response. See *ErrorResponse* below for values.

The *ErrorLocus* attribute has one of the following values:

**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 application’s error event handler can set the *ErrorResponse* attribute to one of the following values:

**Value Meaning**

ER\_RETRY Retry sending the data. The error state is exited.  
May be valid for some input devices when the locus is EL\_INPUT, in which case the input is retried and the error state is exited. Typically, valid for asynchronous output devices when the locus is EL\_OUTPUT, in which case the asynchronous output is retried and the error state is exited. This is the default response when the locus is EL\_OUTPUT.

ER\_CLEAR Valid for all loci: EL\_INPUT, EL\_INPUT\_DATA, and EL\_OUTPUT. Clear all buffered input or output data (including all asynchronous output). The error state is exited. This is the default response when the locus is EL\_INPUT.

ER\_CONTINUEINPUT   
Only valid when the locus is EL\_INPUT\_DATA.  
Acknowledges that a data error has occurred and directs the Device to continue input processing. The Device remains in the error state and will deliver additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been delivered and **DataEventEnabled** is again set to true, then another **ErrorEvent** is delivered with locus EL\_INPUT.  
This is the default response when the locus is EL\_INPUT\_DATA.

**Remarks** This event is enqueued when an error is detected and the Device’s **State** transitions into the error state. Input error events are not delivered until **DataEventEnabled** is true, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Device does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at true. Note that the application may set **DataEventEnabled** to false within its event handler if subsequent input events need to be disabled for a period of time.

**See Also** **“Device Input Model"** on page Intro-22, **“Error Handling"** on page Intro-23, **“Device Output Models"** on page Intro-25.

## **OutputCompleteEvent**

**<<event>> upos::events::OutputCompleteEvent   
 OutputID : int32{read-only}**

**Description** Notify the application that the queued output request associated with the *outputID* property has completed successfully.

**Attributes** This event contains the following attributes:

**Attribute 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 Intro-25

## **StatusUpdateEvent**

**<<event>> upos::events:: StatusUpdateEvent  
 Status : *int32* {read-only}**

**Description** *Notifies the application that there is an operation status change or a status of the Graphic Display device.*

**Attributes**  This event contains the following attribute:

**Attributes Type Description**

*Status* *int32* Indicates a change of operation status of graphic display device

***Note that Release 1.3*** added Power State Reporting with additional *Power reporting* **StatusUpdateEvent** *values.*

The Update Firmware capability added additional *Status* values for communicating the status/progress of an asynchronous update firmware process. See “**StatusUpdateEvent**” description on page 1-34.

**Value Meaning**

GDSP\_SUE\_START\_IMAGE\_LOAD  
 It will be notified when image loading start.

GDSP\_SUE\_END\_IMAGE\_LOAD  
 It will be notified when image loading end.

GDSP\_SUE\_START\_LOAD\_WEBPAGE  
 Start loading the web page.

GDSP\_SUE\_FINISH\_LOAD\_WEBPAGE  
 It has finished loading the web page.

GDSP\_SUE\_CANCEL\_LOAD\_WEBPAGE  
 It has canceled loading the web page.

GDSP\_SUE\_START\_PLAY\_VIDEO

Start playing video.

GDSP\_SUE\_STOP\_PLAY\_VIDEO

Stop playing video.

**Remarks** Enqueued when the Graphic Display Device detects a power state change or a status change.

**See Also** **“Events"** on page Intro-19.

# Appendix K

# Relationship to other OMG specification and activities

## **Robotics Domain Task Force**

### Activities in Robotics Domain Task Force

The OMG Robotics Domain Task Force (Robotics DTF) fosters the integration of robotics systems from modular components through the adoption of OMG standards. It recommends the adoption and extends OMG technologies that apply to the specific domain of robotics systems where no current baseline specifications exist, such as MDA for Robotics. The object technology is not solely limited to software but is extended to real objects. It also collaborates with other organizations for standardization, such as the one for home information appliances, and makes an open effort to increase interoperability in the field of robotics.

(<https://www.omg.org/robotics/>)

### RoIS Specification

Robotic Interaction Service Framework [RoIS] defines several functional components for robotic interaction services.

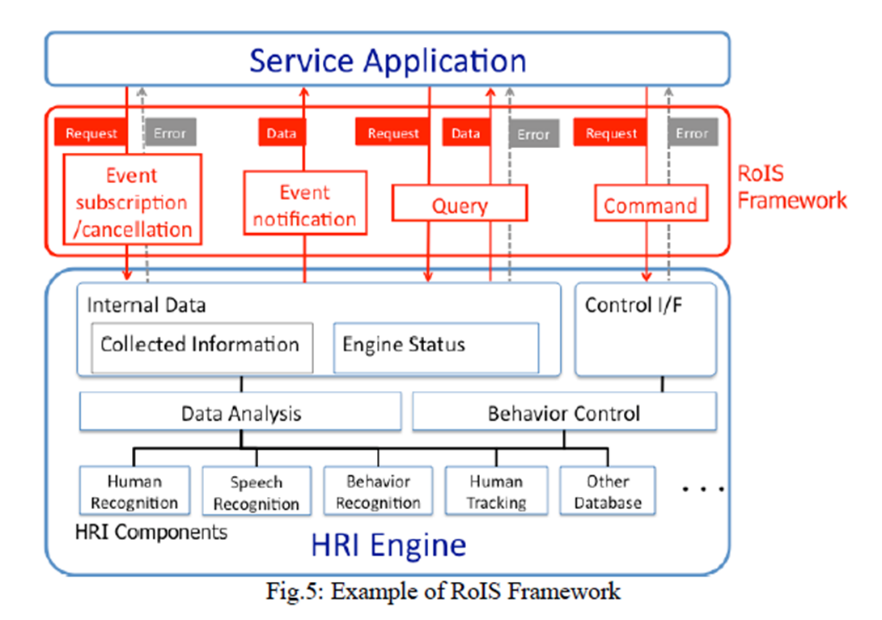
Definitions related to locations of entities in robotic services will be described with Robotic Localization Service[RLS]. Definitions of status of components in services will be described in conjunction with Robotic Technology Component [RTC], Finite State Machine Component for RTC [FSM4RTC] and Unified Component Model for Distributed Real-Time and Embedded Systems [UCM].

RoIS specification seeks that specify a RoIS framework, on top of which various service robot applications are developed.

### Scope of RoIS specification

They are summarized in the following items.

* Interface between service application and Human Robot Interaction (HRI) engine
* Interface to obtain information from HRI Engine according to the timing of the service application’s needs (Query)
* Interface to receive information from HRI Engine triggered by real time events (Event notification / subscription / cancellation)
* Interface for instructions to device control HRI Engine functions (Command)
* Definition of common messages for all HRI Engines



### Robot Service Ontology [RoSO] RFP

A new RFP of Robot Service Ontology[RoSO] currently being discussed in Robotics DTF are based on the concept of RoIS.

RoSO is aiming to define the specification (ontology) that clarifies the concept of a common vocabulary and / or a robot service in order to describe a service provided by a robot or exchange a description of a service provided by a service robot

Below is an example of HRI main component examples from this point of view.

**Table K-1 – (From RoIS 1.2) Basic HRI Components**

|  |  |
| --- | --- |
| **HRI Component Name** | **Description** |
| system information | Provides the information of the system such as status of the system and position of the physical unit. |
| person detection | Detects number of people |
| person localization | Detects position of people |
| person identification | Identifies ID (name) of people |
| face detection | Detects number of human faces |
| face localization | Detects position of human faces |
| sound detection | Detects number of sound sources |
| sound localization | Detects position of sound sources |
| speech recognition | Recognizes person’s speech |
| gesture recognition | Recognizes person’s gesture |
| speech synthesis | Generates robot speech |
| reaction | Performs specified reaction |
| navigation | Moves to specified target location |
| follow | Follows a specified target object |
| move | Moves to specified distance or curve |

## **Interoperability between UPOS RCSD and RoIs**

### Rleationsihp between UPOS RCSD and RoIS

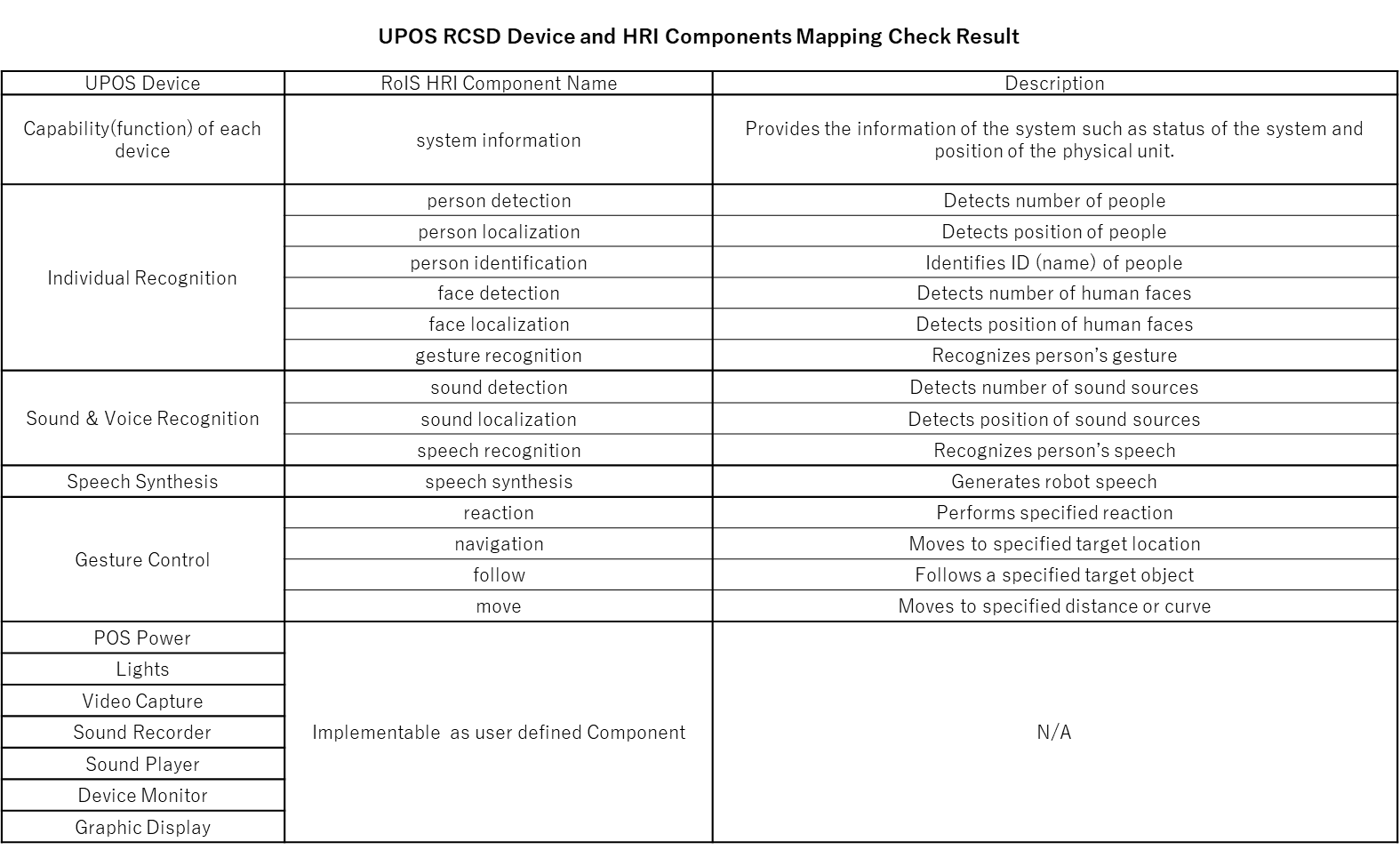
OMG's Robotics standard provides a lower level control layer to manage Robot Device with finer granularity and higher accuracy to accommodate a wide range of industry applications.

On the other hand, the UPOS RCSD specification focuses on the functioning of robotic equipment within the retail store environment. In the UPOS RCSD specification robots are treated as peripheral equipment of the latest POS system. Therefore, the UPOS RCSD specification focuses on the definition of the interface between the POS and the robotic device.

RoIS is already existing as OMG standard and it defined a component frame service that was intended for robotic communication services with people.

Therefore, ROIS developed a general robot service framework, which is different from UPOS RCSD, but it is possible to describe the function of UPOS RCSD.

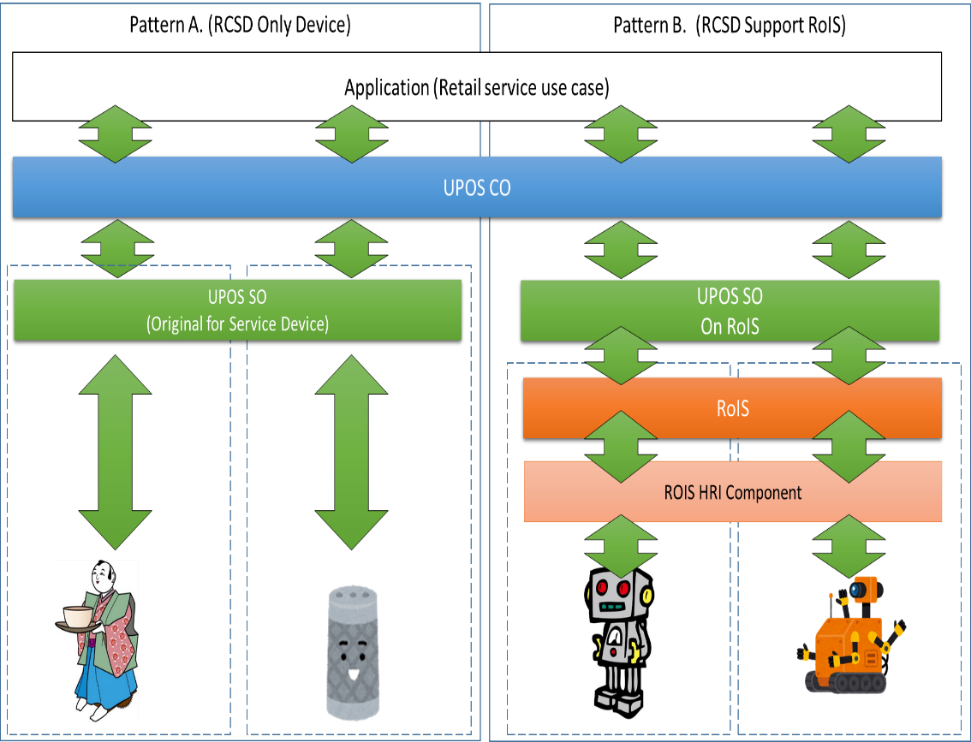
To confirm the compatibility and interoperability of the RCSD functions of RoIS and UPOS, both DTFs created and confirmed the function mapping table.

For this purpose, we use the general RoIS HRI component defined in the RoIS 1.2 specification.

The two teams continue to collaborate between the part of their separate RFP’s and standards that will be established.

For that purpose, it is very necessary to understand the common vocabulary of the robot service and the needs of the ontology.

If each team’s specification satisfies the above mapping table, it is confirmed that the standard can be maintained independently.

In addition, the figure below shows a typical scenario where RCSD and RoIS work independently or in conjunction.

# 

# Document History

## **Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Ver | Date | Sections | Description of Change |
| 1.0 | 2019-2-18 |  | Initial Version – additions and updates to UPOS v1.15 |
| 1.1 | 2019-7-09 |  | Revised for the issus and additions from the Review |
| 1.2 | 2020-2-15 |  | Issues, Upates are added version from the Review |

# Glossary

|  |  |
| --- | --- |
|  | |
| Term | Definition |
| **EVRW** | Electronic Value Reader Writer |
| **CAT** | Credit Authorization Terminal |
|  |  |