

Raritan PX2 JSON-RPC API

Generated by Doxygen 1.8.3.1

Fri Jun 20 2014 19:01:32

Contents

1	Rules and Mechanism for Mapping PX2/EMX-IDL to JSON-RPC	1
1.1	A) Scope	1
1.2	B) Mapping Rules	1
1.2.1	B)1 Type Mapping	1
1.2.2	B)2 Interface version Mapping	2
1.2.3	B)3 Method Mapping	2
1.3	C) Call Execution	2
1.4	D) Language Bindings and Examples	2
2	Curl JSON-RPC Example	5
2.1	Mechanism	5
2.2	Examples	5
3	Perl JSON-RPC Client Bindings	7
3.1	Synopsis	7
3.2	IDL-to-Perl Mapping	7
3.2.1	IDL Modules and Perl Package Names	7
3.2.2	Pre-Requirements	7
3.2.2.1	Linux	7
3.2.2.2	Windows	8
3.2.3	Interfaces and Methods	8
3.2.4	Structures	8
3.2.5	Enumerations	8
3.2.6	Vectors	8
3.2.7	Maps	9
3.3	Exceptions	9
3.4	Raritan::RPC::Agent Method Reference	9
3.4.1	Constructor: new Raritan::RPC::Agent(\$url, \$user, \$password)	9
3.4.2	set_auth_basic(\$user, \$password)	9
3.4.3	set_auth_basic(\$token)	10
3.4.4	timeout(\$seconds)	10
3.4.5	createProxy(\$rid, \$basetype)	10

3.4.6	<code>set_verbose(\$verbose)</code>	10
4	Python JSON-RPC Client Binding	11
4.1	Examples	11
4.1.1	Getting device information of EMX:	11
4.1.2	Switch first outlet of PX:	11
4.2	IDL-to-Python Mapping	12
4.2.1	Modules and Python Packages	12
4.2.2	Interfaces and Methods	12
4.2.3	Structures	12
4.2.4	Enumerations	12
4.2.5	Vectors	12
4.2.6	Maps	13
4.2.7	Exceptions	13
5	Namespace Index	15
5.1	Namespace List	15
6	Hierarchical Index	17
6.1	Class Hierarchy	17
7	Class Index	23
7.1	Class List	23
8	Namespace Documentation	35
8.1	assetmgrmodel Namespace Reference	35
8.1.1	Detailed Description	35
8.2	auth Namespace Reference	35
8.2.1	Detailed Description	36
8.2.2	Enumeration Type Documentation	36
8.2.2.1	Type	36
8.3	auth::ldapsrv Namespace Reference	36
8.3.1	Detailed Description	36
8.3.2	Enumeration Type Documentation	37
8.3.2.1	ServerType	37
8.4	bulkcfg Namespace Reference	37
8.4.1	Detailed Description	37
8.5	cascading Namespace Reference	37
8.5.1	Detailed Description	37
8.6	cert Namespace Reference	37
8.6.1	Detailed Description	38
8.7	cew Namespace Reference	38

8.7.1 Detailed Description	38
8.8 datetime Namespace Reference	38
8.8.1 Detailed Description	38
8.9 devsettings Namespace Reference	38
8.9.1 Detailed Description	39
8.10 diag Namespace Reference	39
8.10.1 Detailed Description	39
8.11 event Namespace Reference	39
8.11.1 Detailed Description	40
8.11.2 Variable Documentation	40
8.11.2.1 UserEvent	40
8.12 firmware Namespace Reference	40
8.12.1 Detailed Description	40
8.12.2 Enumeration Type Documentation	41
8.12.2.1 ImageState	41
8.12.2.2 UpdateFlags	41
8.12.2.3 UpdateHistoryStatus	41
8.13 fitness Namespace Reference	41
8.13.1 Detailed Description	41
8.14 hmi Namespace Reference	42
8.14.1 Detailed Description	42
8.15 idl Namespace Reference	42
8.15.1 Detailed Description	42
8.15.2 Variable Documentation	42
8.15.2.1 Event	42
8.16 jsonrpc Namespace Reference	42
8.16.1 Detailed Description	43
8.17 lhx Namespace Reference	43
8.17.1 Detailed Description	43
8.18 lhxmodel Namespace Reference	43
8.18.1 Detailed Description	43
8.19 logging Namespace Reference	43
8.19.1 Detailed Description	44
8.19.2 Enumeration Type Documentation	44
8.19.2.1 RangeDirection	44
8.20 modelpush Namespace Reference	44
8.20.1 Detailed Description	44
8.21 net Namespace Reference	44
8.21.1 Detailed Description	45
8.21.2 Enumeration Type Documentation	46

8.21.2.1	AuthenticationMode	46
8.21.2.2	AutoConfigs	46
8.21.2.3	EapInnerMethod	46
8.21.2.4	EapOuterMethod	46
8.21.2.5	InterfaceMode_2_0_0	46
8.21.2.6	LanDuplex	47
8.21.2.7	LanSpeed	47
8.22	pdumodel Namespace Reference	47
8.22.1	Detailed Description	48
8.22.2	Enumeration Type Documentation	48
8.22.2.1	PowerLine	48
8.23	peripheral Namespace Reference	48
8.23.1	Detailed Description	49
8.23.2	Enumeration Type Documentation	49
8.23.2.1	PortType	49
8.23.3	Variable Documentation	50
8.23.3.1	Device_2_0_0	50
8.24	portsmodel Namespace Reference	50
8.24.1	Detailed Description	50
8.25	production Namespace Reference	50
8.25.1	Detailed Description	50
8.26	radius Namespace Reference	50
8.26.1	Detailed Description	51
8.26.2	Enumeration Type Documentation	51
8.26.2.1	AuthType	51
8.27	res_mon Namespace Reference	51
8.27.1	Detailed Description	51
8.28	security Namespace Reference	51
8.28.1	Detailed Description	52
8.28.2	Enumeration Type Documentation	52
8.28.2.1	IpfwPolicy	52
8.28.2.2	RoleAccessPolicy	53
8.29	sensors Namespace Reference	53
8.29.1	Detailed Description	53
8.30	serial Namespace Reference	53
8.30.1	Detailed Description	54
8.31	servermon Namespace Reference	54
8.31.1	Detailed Description	54
8.32	session Namespace Reference	54
8.32.1	Detailed Description	54

8.33 smartcard Namespace Reference	54
8.33.1 Detailed Description	55
8.34 sys Namespace Reference	55
8.34.1 Detailed Description	55
8.35 test Namespace Reference	55
8.35.1 Detailed Description	55
8.36 um Namespace Reference	56
8.36.1 Detailed Description	56
8.37 usb Namespace Reference	56
8.37.1 Detailed Description	56
8.38 usermgmt Namespace Reference	56
8.38.1 Detailed Description	57
8.38.2 Enumeration Type Documentation	57
8.38.2.1 LengthEnum	57
8.38.2.2 PressureEnum	58
8.38.2.3 TemperatureEnum	58
8.38.3 Variable Documentation	58
8.38.3.1 AccountEvent	58
8.39 webcam Namespace Reference	58
8.39.1 Detailed Description	59
8.39.2 Enumeration Type Documentation	59
8.39.2.1 PixelFormat	59
8.39.2.2 Priority	59
8.39.3 Variable Documentation	60
8.39.3.1 WebcamEvent	60
8.39.3.2 WebcamSettingsChangedEvent	60
9 Class Documentation	61
9.1 usermgmt::Account Struct Reference	61
9.1.1 Detailed Description	61
9.2 sensors::AccumulatingNumericSensor_2_0_0 Interface Reference	61
9.2.1 Detailed Description	62
9.2.2 Member Data Documentation	62
9.2.2.1 ResetEvent	62
9.3 event::Engine::Action Struct Reference	62
9.3.1 Detailed Description	63
9.4 webcam::StorageManager_1_0_1::Activity Struct Reference	63
9.4.1 Detailed Description	63
9.5 peripheral::Address_2_0_0 Struct Reference	63
9.5.1 Detailed Description	64

9.6	pdumodel::Ade Interface Reference	64
9.6.1	Detailed Description	64
9.6.2	Member Function Documentation	64
9.6.2.1	getCalibrationData	64
9.6.2.2	getLatestSample	65
9.6.2.3	getMetaData	65
9.6.2.4	setCalibrationData	65
9.7	event::AlarmManager::Alarm Struct Reference	65
9.7.1	Detailed Description	66
9.8	event::AlarmManager Interface Reference	66
9.8.1	Detailed Description	66
9.8.2	Member Function Documentation	67
9.8.2.1	acknowledgeAlarm	67
9.8.3	Member Data Documentation	67
9.8.3.1	AlarmAcknowledgedEvent	67
9.8.3.2	AlarmAddedEvent	67
9.8.3.3	AlarmUpdatedEvent	67
9.8.3.4	NO_ERROR	67
9.9	event::AlarmManager::Alert Struct Reference	67
9.9.1	Detailed Description	68
9.10	lhxmodel::Lhx_3_2_1::AlertStatus Struct Reference	68
9.10.1	Detailed Description	69
9.11	serial::AnalogModem Interface Reference	69
9.11.1	Detailed Description	70
9.11.2	Member Function Documentation	70
9.11.2.1	getSettings	70
9.11.2.2	setSettings	70
9.11.3	Member Data Documentation	70
9.11.3.1	CallEndedEvent	70
9.11.3.2	DialInEvent	70
9.11.3.3	SUCCESS	70
9.12	usermgmt::RoleManager::ArgumentDesc Struct Reference	70
9.12.1	Detailed Description	71
9.13	assetmgrmodel::AssetStrip_2_0_1 Interface Reference	71
9.13.1	Detailed Description	73
9.13.2	Member Enumeration Documentation	73
9.13.2.1	CascadeState	73
9.13.2.2	FirmwareUpdateState	73
9.13.2.3	State	73
9.13.2.4	StripType	74

9.13.2.5	TagType	74
9.13.3	Member Function Documentation	74
9.13.3.1	getAllRackUnitInfos	74
9.13.3.2	getAllTags	74
9.13.3.3	getDeviceInfo	74
9.13.3.4	getExtensionTags	75
9.13.3.5	getMainTags	75
9.13.3.6	getRackUnitInfo	75
9.13.3.7	getState	75
9.13.3.8	getStripInfo	75
9.13.3.9	getTag	76
9.13.3.10	triggerPowercycle	76
9.13.4	Member Data Documentation	76
9.13.4.1	BladeOverflowChangedEvent	76
9.13.4.2	CompositionChangedEvent	76
9.13.4.3	FirmwareUpdateStateChangedEvent	76
9.13.4.4	NO_ERROR	76
9.13.4.5	OrientationChangedEvent	76
9.13.4.6	RackUnitChangedEvent	77
9.13.4.7	StateChangedEvent	77
9.13.4.8	StripInfoChangedEvent	77
9.13.4.9	TagEvent	77
9.14	assetmgrmodel::AssetStripConfig_1_0_1 Interface Reference	77
9.14.1	Detailed Description	78
9.14.2	Member Enumeration Documentation	78
9.14.2.1	LEDMode	78
9.14.2.2	LEDOperationMode	79
9.14.2.3	NumberingMode	79
9.14.2.4	Orientation	79
9.14.2.5	ScanMode	79
9.14.3	Member Function Documentation	79
9.14.3.1	getAllRackUnitSettings	79
9.14.3.2	getRackUnitSettings	80
9.14.3.3	getStripSettings	80
9.14.3.4	setMultipleRackUnitSettings	80
9.14.3.5	setRackUnitSettings	80
9.14.3.6	setStripSettings	81
9.14.4	Member Data Documentation	81
9.14.4.1	RackUnitSettingsChangedEvent	81
9.14.4.2	StripSettingsChangedEvent	81

9.15	assetmgrmodel::AssetStripLogger_1_0_2 Interface Reference	81
9.15.1	Detailed Description	82
9.15.2	Member Enumeration Documentation	82
9.15.2.1	RecordType	82
9.15.3	Member Function Documentation	82
9.15.3.1	getInfo	82
9.15.3.2	getRecords	82
9.15.4	Member Data Documentation	83
9.15.4.1	NO_ERROR	83
9.16	auth::AuthManager Interface Reference	83
9.16.1	Detailed Description	83
9.16.2	Member Function Documentation	83
9.16.2.1	getPolicy	83
9.16.2.2	setPolicy	84
9.17	usermgmt::AuxInfo Struct Reference	84
9.17.1	Detailed Description	84
9.18	test::AuxSerial Interface Reference	84
9.18.1	Detailed Description	85
9.18.2	Member Function Documentation	85
9.18.2.1	getNumberOfPorts	85
9.18.2.2	testLoop	85
9.19	pdumodel::Bcm Interface Reference	85
9.19.1	Detailed Description	86
9.19.2	Member Enumeration Documentation	86
9.19.2.1	LineConfig	86
9.19.2.2	TransformerType	86
9.19.3	Member Function Documentation	86
9.19.3.1	getChannelConfigs	86
9.19.3.2	getChannelCount	87
9.19.3.3	setChannelConfig	87
9.20	bulkcfg::BulkConfiguration Interface Reference	87
9.20.1	Detailed Description	87
9.20.2	Member Enumeration Documentation	88
9.20.2.1	Status	88
9.20.3	Member Function Documentation	88
9.20.3.1	getStatus	88
9.21	lhxmodel::Lhx_3_2_1::Capabilities Struct Reference	88
9.21.1	Detailed Description	88
9.22	smartcard::CardReader::CardInformation Struct Reference	88
9.22.1	Detailed Description	89

9.23	smartcard::CardReader Interface Reference	89
9.23.1	Detailed Description	89
9.23.2	Member Function Documentation	90
9.23.2.1	getCardInformation	90
9.23.2.2	getMetaData	90
9.23.3	Member Data Documentation	90
9.23.3.1	CardEvent	90
9.23.3.2	NO_ERROR	90
9.24	smartcard::CardReaderManager Interface Reference	90
9.24.1	Detailed Description	91
9.24.2	Member Function Documentation	91
9.24.2.1	getCardReaderById	91
9.24.2.2	getCardReaders	91
9.24.3	Member Data Documentation	91
9.24.3.1	CardReaderEvent	91
9.25	cascading::Cascading_1_0_1 Interface Reference	91
9.25.1	Detailed Description	92
9.25.2	Member Enumeration Documentation	92
9.25.2.1	Type	92
9.25.3	Member Function Documentation	93
9.25.3.1	getIndex	93
9.25.3.2	getMasterIpAddress	93
9.25.3.3	getMasterIpV6Address	93
9.25.3.4	getProtocolMappings	93
9.25.3.5	getType	93
9.25.3.6	setType	94
9.26	cert::ServerSSLCert::CertInfo Struct Reference	94
9.26.1	Detailed Description	94
9.27	datetime::DateTime_2_0_0::Cfg Struct Reference	95
9.27.1	Detailed Description	95
9.28	webcam::Channel Interface Reference	95
9.28.1	Detailed Description	96
9.28.2	Member Function Documentation	96
9.28.2.1	captureImage	96
9.28.2.2	getClientType	96
9.28.2.3	getWebcam	96
9.28.2.4	isAvailable	96
9.28.2.5	triggerCapture	97
9.28.3	Member Data Documentation	97
9.28.3.1	NO_ERROR	97

9.29	event::Channel_1_0_1 Interface Reference	97
9.29.1	Detailed Description	98
9.29.2	Member Function Documentation	98
9.29.2.1	cancelEvent	98
9.29.2.2	cancelEvents	98
9.29.2.3	cancelEventType	98
9.29.2.4	cancelEventTypes	98
9.29.2.5	demandEvent	99
9.29.2.6	demandEvents	99
9.29.2.7	demandEventType	99
9.29.2.8	demandEventTypes	99
9.29.2.9	pollEvents	99
9.29.2.10	pollEventsNb	100
9.29.2.11	subscribe	100
9.29.2.12	unsubscribe	100
9.30	pdumodel::Bcm::ChannelConfig Struct Reference	100
9.30.1	Detailed Description	101
9.31	pdumodel::CircuitBreakerStatistic Struct Reference	101
9.31.1	Detailed Description	101
9.32	cert::ServerSSLCert::CommonAttributes Struct Reference	101
9.32.1	Detailed Description	102
9.33	lhxmodel::Config_1_0_1::ComSettings Struct Reference	102
9.33.1	Detailed Description	102
9.34	event::Engine::Condition Struct Reference	102
9.34.1	Detailed Description	103
9.34.2	Member Enumeration Documentation	103
9.34.2.1	MatchType	103
9.34.2.2	Op	103
9.35	lhxmodel::Config_1_0_1 Interface Reference	103
9.35.1	Detailed Description	104
9.35.2	Member Function Documentation	104
9.35.2.1	getComSettings	104
9.35.2.2	getName	104
9.35.2.3	setComSettings	105
9.35.2.4	setName	105
9.35.3	Member Data Documentation	105
9.35.3.1	ComSettingsChangedEvent	105
9.35.3.2	PortNameChangedEvent	105
9.36	modelpush::ModelPush::Configuration Struct Reference	105
9.36.1	Detailed Description	106

9.37 devsettings::Sntp_1_0_1::Configuration Struct Reference	106
9.37.1 Detailed Description	107
9.38 devsettings::Snmp_1_0_2::Configuration Struct Reference	107
9.38.1 Detailed Description	107
9.39 event::Consumer Interface Reference	107
9.39.1 Detailed Description	108
9.39.2 Member Function Documentation	108
9.39.2.1 pushEvents	108
9.40 test::Control Interface Reference	108
9.40.1 Detailed Description	108
9.41 pdumodel::Controller_3_0_0 Interface Reference	108
9.41.1 Detailed Description	109
9.41.2 Member Enumeration Documentation	110
9.41.2.1 Status	110
9.41.2.2 Type	110
9.41.3 Member Function Documentation	110
9.41.3.1 getCommunicationStatus	110
9.41.3.2 getMetaData	110
9.41.3.3 getStatistics	110
9.41.4 Member Data Documentation	111
9.41.4.1 MetaDataChangedEvent	111
9.41.4.2 StatusChangedEvent	111
9.42 webcam::Controls Struct Reference	111
9.42.1 Detailed Description	111
9.43 pdumodel::CtrlStatistic Struct Reference	111
9.43.1 Detailed Description	112
9.44 fitness::Fitness::DataEntry Struct Reference	112
9.44.1 Detailed Description	112
9.45 datetime::DateTime_2_0_0 Interface Reference	113
9.45.1 Detailed Description	113
9.45.2 Member Enumeration Documentation	113
9.45.2.1 Protocol	113
9.45.3 Member Function Documentation	114
9.45.3.1 checkNtpServer	114
9.45.3.2 getCfg	114
9.45.3.3 getTime	114
9.45.3.4 getZoneInfos	114
9.45.3.5 setCfg	114
9.46 logging::DebugLog Interface Reference	115
9.46.1 Detailed Description	115

9.46.2	Member Function Documentation	115
9.46.2.1	getEntries	115
9.46.2.2	getFirstId	115
9.46.2.3	getLastId	116
9.47	portsmodel::Port_2_0_1::DetectionMode Struct Reference	116
9.47.1	Detailed Description	116
9.48	peripheral::DeviceID_2_0_0 Struct Reference	116
9.48.1	Detailed Description	117
9.49	assetmgrmodel::AssetStrip_2_0_1::DeviceInfo Struct Reference	117
9.49.1	Detailed Description	117
9.50	peripheral::DeviceManager_2_0_0 Interface Reference	117
9.50.1	Detailed Description	119
9.50.2	Member Enumeration Documentation	119
9.50.2.1	DeviceFirmwareUpdateState	119
9.50.2.2	ZCoordMode	119
9.50.3	Member Function Documentation	120
9.50.3.1	getDeviceSlot	120
9.50.3.2	getDeviceSlots	120
9.50.3.3	getDeviceTypeInfo	120
9.50.3.4	getDiscoveredDevices	120
9.50.3.5	getDiscoveredPackageInfos	120
9.50.3.6	getMetaData	120
9.50.3.7	getSettings	121
9.50.3.8	getStatistics	121
9.50.3.9	setSettings	121
9.50.4	Member Data Documentation	121
9.50.4.1	DeviceEvent	121
9.50.4.2	DeviceFirmwareUpdateStateChangedEvent	121
9.50.4.3	PackageEvent	121
9.50.4.4	SettingsChangedEvent	121
9.50.4.5	UnknownDeviceAttachedEvent	122
9.51	peripheral::DeviceSlot_2_0_0 Interface Reference	122
9.51.1	Detailed Description	123
9.51.2	Member Function Documentation	123
9.51.2.1	assign	123
9.51.2.2	assignAddress	123
9.51.2.3	getDevice	123
9.51.2.4	getSettings	123
9.51.2.5	setSettings	123
9.51.2.6	unassign	124

9.51.3	Member Data Documentation	124
9.51.3.1	DeviceChangedEvent	124
9.51.3.2	SettingsChangedEvent	124
9.52	peripheral::DeviceManager_2_0_0::DeviceTypeInfo Struct Reference	124
9.52.1	Detailed Description	125
9.53	diag::DiagLogSettings Interface Reference	125
9.53.1	Detailed Description	125
9.53.2	Member Enumeration Documentation	126
9.53.2.1	LogLevel	126
9.53.3	Member Function Documentation	126
9.53.3.1	getLogLevelByCtxName	126
9.53.3.2	getLogLevelsForAllCtxNames	126
9.53.3.3	setLogLevelByCtxName	126
9.53.3.4	setLogLevelForAllCtxNames	127
9.54	net::Diagnostics Interface Reference	127
9.54.1	Detailed Description	127
9.54.2	Member Function Documentation	128
9.54.2.1	listTcpConnections	128
9.54.2.2	ping	128
9.54.2.3	traceRoute	128
9.55	test::Display_1_0_1 Interface Reference	128
9.55.1	Detailed Description	129
9.55.2	Member Enumeration Documentation	129
9.55.2.1	Orientation	129
9.55.2.2	TestStatus	129
9.55.3	Member Function Documentation	130
9.55.3.1	getInfo	130
9.55.3.2	getTestStatus	130
9.55.3.3	testSequence	130
9.56	pdumodel::DoublePole_2_0_0 Struct Reference	130
9.56.1	Detailed Description	131
9.57	net::EapSettings Struct Reference	131
9.57.1	Detailed Description	131
9.58	pdumodel::EDevice Interface Reference	132
9.58.1	Detailed Description	132
9.58.2	Member Function Documentation	132
9.58.2.1	getChildren	132
9.58.2.2	getParents	132
9.59	cew::EnergyWiseManager Interface Reference	133
9.59.1	Detailed Description	133

9.59.2	Member Function Documentation	133
9.59.2.1	getSettings	133
9.59.2.2	setSettings	133
9.60	cew::EnergyWiseSettings Struct Reference	133
9.60.1	Detailed Description	134
9.61	event::Engine Interface Reference	134
9.61.1	Detailed Description	135
9.61.2	Member Function Documentation	135
9.61.2.1	addAction	135
9.61.2.2	addRule	135
9.61.2.3	deleteAction	136
9.61.2.4	deleteRule	136
9.61.2.5	deliverEvent	136
9.61.2.6	disableRule	136
9.61.2.7	enableRule	136
9.61.2.8	listActionTypes	136
9.61.2.9	listEventDescs	137
9.61.2.10	listRules	137
9.61.2.11	modifyAction	137
9.61.2.12	modifyRule	137
9.61.2.13	rearmRule	137
9.61.2.14	triggerAction	138
9.62	res_mon::Entry Struct Reference	138
9.62.1	Detailed Description	138
9.62.2	Member Enumeration Documentation	138
9.62.2.1	Type	138
9.63	fitness::Fitness::ErrorLogEntry Struct Reference	139
9.63.1	Detailed Description	139
9.64	test::Ethernet Interface Reference	139
9.64.1	Detailed Description	140
9.64.2	Member Enumeration Documentation	140
9.64.2.1	Duplex	140
9.64.2.2	Speed	140
9.64.3	Member Function Documentation	140
9.64.3.1	setParameters	140
9.65	event::Event Struct Reference	141
9.65.1	Detailed Description	141
9.65.2	Member Enumeration Documentation	141
9.65.2.1	Type	141
9.66	event::Engine::EventDesc Struct Reference	142

9.66.1 Detailed Description	142
9.66.2 Member Enumeration Documentation	142
9.66.2.1 Type	142
9.67 logging::EventLog_1_0_1 Interface Reference	143
9.67.1 Detailed Description	143
9.67.2 Member Function Documentation	143
9.67.2.1 getEntries	143
9.67.2.2 getFilteredEntries	143
9.67.2.3 getFirstId	144
9.67.2.4 getLastId	144
9.68 event::Channel_1_0_1::EventSelect Struct Reference	144
9.68.1 Detailed Description	144
9.69 hmi::ExternalBeeper_1_0_1 Interface Reference	144
9.69.1 Detailed Description	145
9.69.2 Member Function Documentation	145
9.69.2.1 alarm	145
9.69.2.2 off	145
9.69.2.3 on	145
9.70 test::FeatSerial Interface Reference	145
9.70.1 Detailed Description	146
9.70.2 Member Function Documentation	146
9.70.2.1 getNumberOfPorts	146
9.70.2.2 setPower	146
9.70.2.3 testLoopDtrDcd	147
9.70.2.4 testLoopTxRx	147
9.71 firmware::Firmware_1_0_1 Interface Reference	147
9.71.1 Detailed Description	148
9.71.2 Member Function Documentation	148
9.71.2.1 downloadImage	148
9.71.2.2 enableOnlineCheck	148
9.71.2.3 getImageInfo	148
9.71.2.4 getImageStatus	148
9.71.2.5 getUpdateHistory	149
9.71.2.6 getVersion	149
9.71.2.7 onlineCheckEnabled	149
9.71.2.8 performOnlineCheck	149
9.71.2.9 reboot	149
9.71.2.10 startUpdate	149
9.71.2.11 updateAvailable	150
9.72 peripheral::PackageInfo_2_0_0::FirmwareInfo Struct Reference	150

9.73	peripheral::G2Production_2_0_0::FirmwareInfo Struct Reference	150
9.74	firmware::FirmwareUpdateStatus Interface Reference	151
9.74.1	Detailed Description	151
9.74.2	Member Function Documentation	151
9.74.2.1	getStatus	151
9.75	fitness::Fitness Interface Reference	151
9.75.1	Detailed Description	152
9.75.2	Member Function Documentation	152
9.75.2.1	getDataEntries	152
9.75.2.2	getErrorLogEntries	152
9.75.2.3	getErrorLogIndexRange	153
9.75.3	Member Data Documentation	153
9.75.3.1	FLAG_ENTRY_CRITICAL	153
9.75.3.2	FLAG_VALUE_INVALID	153
9.75.3.3	FLAG_VALUE_OLD	153
9.76	webcam::Format_2_0_0 Struct Reference	153
9.76.1	Detailed Description	153
9.77	peripheral::G2Production_2_0_0 Interface Reference	154
9.77.1	Member Enumeration Documentation	155
9.77.1.1	ConfigurationSpace	155
9.77.1.2	ResetMethod	155
9.77.2	Member Function Documentation	155
9.77.2.1	eraseConfigurationSpace	155
9.77.2.2	getFirmwareInfo	155
9.77.2.3	readConfigurationSpace	156
9.77.2.4	readRegisters	156
9.77.2.5	reset	156
9.77.2.6	updateFirmware	157
9.77.2.7	writeConfigurationSpace	157
9.77.2.8	writeRegisterBits	157
9.77.2.9	writeRegisters	158
9.78	serial::GsmModem_1_0_1 Interface Reference	158
9.78.1	Detailed Description	159
9.78.2	Member Enumeration Documentation	159
9.78.2.1	SimSecurityStatus	159
9.78.3	Member Function Documentation	160
9.78.3.1	getInformation	160
9.78.3.2	getInformationWithPin	160
9.78.3.3	getSettings	160
9.78.3.4	getSimSecurityStatus	160

9.78.3.5	sendSms	161
9.78.3.6	sendTestSms	161
9.78.3.7	setSettings	161
9.78.3.8	unlockSimCard	162
9.78.4	Member Data Documentation	162
9.78.4.1	SimPinUpdatedEvent	162
9.78.4.2	SimSecurityStatusChangedEvent	162
9.78.4.3	SUCCESS	162
9.79	peripheral::PackageInfo_2_0_0::HardwareInfo Struct Reference	162
9.80	session::HistoryEntry Struct Reference	163
9.80.1	Detailed Description	163
9.81	webcam::Image_2_0_0 Struct Reference	163
9.81.1	Detailed Description	163
9.82	firmware::ImageInfo_1_0_1 Struct Reference	164
9.82.1	Detailed Description	164
9.83	webcam::ImageMetaData Struct Reference	165
9.83.1	Detailed Description	165
9.84	firmware::ImageStatus Struct Reference	165
9.84.1	Detailed Description	165
9.85	webcam::StorageManager_1_0_1::ImageStorageMetaData Struct Reference	166
9.85.1	Detailed Description	166
9.86	assetmgrmodel::AssetStripLogger_1_0_2::Info Struct Reference	166
9.86.1	Detailed Description	166
9.87	usrmgmt::Role::Info Struct Reference	166
9.87.1	Detailed Description	167
9.88	usrmgmt::RoleManager::Info Struct Reference	167
9.88.1	Detailed Description	167
9.89	cert::ServerSSLCert::Info Struct Reference	167
9.89.1	Detailed Description	168
9.89.2	Member Data Documentation	168
9.89.2.1	maxSignDays	168
9.90	test::Display_1_0_1::Info Struct Reference	168
9.90.1	Detailed Description	168
9.91	serial::GsmModem_1_0_1::Information Struct Reference	169
9.91.1	Detailed Description	169
9.92	webcam::Information_2_0_0 Struct Reference	169
9.92.1	Detailed Description	170
9.93	pdumodel::Inlet_1_2_6 Interface Reference	170
9.93.1	Detailed Description	171
9.93.2	Member Function Documentation	171

9.93.2.1	getMetaData	171
9.93.2.2	getPoles	171
9.93.2.3	getSensors	171
9.93.2.4	getSettings	171
9.93.2.5	isEnabled	171
9.93.2.6	setEnabled	172
9.93.2.7	setSettings	172
9.93.3	Member Data Documentation	172
9.93.3.1	EnableStateChangedEvent	172
9.93.3.2	SettingsChangedEvent	172
9.94	net::InterfaceState_2_0_0 Struct Reference	172
9.94.1	Detailed Description	173
9.95	hmi::InternalBeeper Interface Reference	173
9.95.1	Detailed Description	173
9.95.2	Member Enumeration Documentation	174
9.95.2.1	State	174
9.95.3	Member Function Documentation	174
9.95.3.1	activate	174
9.95.3.2	getState	174
9.95.3.3	isMuted	174
9.95.3.4	mute	174
9.95.4	Member Data Documentation	175
9.95.4.1	StateChangedEvent	175
9.96	security::IpFw_2_0_0 Struct Reference	175
9.96.1	Detailed Description	175
9.97	security::IpfwRule Struct Reference	175
9.97.1	Detailed Description	176
9.98	net::IPv4RoutingEntry Struct Reference	176
9.98.1	Detailed Description	176
9.99	net::IPv6RoutingEntry Struct Reference	176
9.99.1	Detailed Description	176
9.100	event::KeyValue Struct Reference	177
9.100.1	Detailed Description	177
9.101	net::LanInterfaceParameters_2_0_0 Struct Reference	177
9.101.1	Detailed Description	177
9.102	net::LanInterfaceSettings Struct Reference	178
9.102.1	Detailed Description	178
9.103	net::LanLinkMode Struct Reference	178
9.103.1	Detailed Description	178
9.104	auth::LdapManager_1_0_1 Interface Reference	178

9.104.1 Detailed Description	179
9.104.2 Member Function Documentation	179
9.104.2.1 getLdapServers	179
9.104.2.2 setLdapServers	179
9.104.2.3 testLdapServer	180
9.105assetmgrmodel::AssetStripConfig_1_0_1::LEDColor Struct Reference	180
9.105.1 Detailed Description	180
9.106pdumodel::Outlet_1_5_6::LedState Struct Reference	180
9.106.1 Detailed Description	181
9.107lhxmodel::Lhx_3_2_1 Interface Reference	181
9.107.1 Detailed Description	182
9.107.2 Member Function Documentation	182
9.107.2.1 acknowledgeAlertStatus	182
9.107.2.2 getMetaData	182
9.107.2.3 getSettings	183
9.107.2.4 setMaximumCoolingRequest	183
9.107.2.5 setPowerState	183
9.107.2.6 setSettings	183
9.107.3 Member Data Documentation	183
9.107.3.1 OpStateChangedEvent	183
9.107.3.2 SettingsChangedEvent	184
9.108webcam::Location Struct Reference	184
9.108.1 Detailed Description	184
9.109peripheral::DeviceSlot_2_0_0::Location Struct Reference	184
9.109.1 Detailed Description	185
9.110logging::LogEntry Struct Reference	185
9.110.1 Detailed Description	185
9.111sensors::Logger_2_1_2 Interface Reference	185
9.111.1 Detailed Description	188
9.111.2 Member Function Documentation	188
9.111.2.1 getLoggedSensors	188
9.111.2.2 getLogRow	188
9.111.2.3 getPeripheralDeviceRecords	188
9.111.2.4 getPeripheralDeviceTimedRecords	188
9.111.2.5 getSensorRecords	189
9.111.2.6 getSensorSetTimestamp	189
9.111.2.7 getSensorTimedRecords	189
9.111.2.8 getSettings	190
9.111.2.9 getTimeStamps	190
9.111.2.10setLoggedSensors	190

9.111.2.1 setSettings	190
9.111.3 Member Data Documentation	191
9.111.3.1 LoggedSensorsChangedEvent	191
9.111.3.2 SettingsChangedEvent	191
9.111.3.3 STATE_UNAVAILABLE	191
9.112 diag::DiagLogSettings::LogLevelEntry Struct Reference	191
9.112.1 Detailed Description	191
9.113 sensors::Logger_2_1_2::LogRow Struct Reference	191
9.113.1 Detailed Description	192
9.114 pdumodel::MemoryMapController_3_0_0 Interface Reference	192
9.114.1 Detailed Description	192
9.114.2 Member Function Documentation	192
9.114.2.1 readMemory	192
9.114.2.2 writeMemory	193
9.115 sensors::NumericSensor_4_0_0::MetaData Struct Reference	193
9.115.1 Detailed Description	194
9.115.2 Member Data Documentation	194
9.115.2.1 accuracy	194
9.115.2.2 decdigits	194
9.115.2.3 noiseThreshold	194
9.115.2.4 range	194
9.115.2.5 resolution	194
9.115.2.6 tolerance	194
9.116 pdumodel::Unit_2_0_1::MetaData Struct Reference	195
9.116.1 Detailed Description	195
9.117 pdumodel::Outlet_1_5_6::MetaData Struct Reference	195
9.117.1 Detailed Description	195
9.118 pdumodel::OverCurrentProtector_2_1_2::MetaData Struct Reference	196
9.118.1 Detailed Description	196
9.119 pdumodel::Controller_3_0_0::MetaData Struct Reference	196
9.119.1 Detailed Description	197
9.120 pdumodel::Pdu_3_0_0::MetaData Struct Reference	197
9.120.1 Detailed Description	197
9.121 pdumodel::Inlet_1_2_6::MetaData Struct Reference	197
9.121.1 Detailed Description	198
9.122 peripheral::DeviceManager_2_0_0::MetaData Struct Reference	198
9.122.1 Detailed Description	198
9.123 lhmodel::Lhx_3_2_1::MetaData Struct Reference	198
9.123.1 Detailed Description	199
9.124 lhmodel::Parameter_2_0_1::MetaData Struct Reference	199

9.124.1 Detailed Description	199
9.125lhxmodel::Sensor_4_0_0::MetaData Struct Reference	199
9.125.1 Detailed Description	200
9.125.2 Member Data Documentation	200
9.125.2.1 numDecDigits	200
9.126pdumodel::Ade::MetaData Struct Reference	200
9.126.1 Detailed Description	201
9.127smartcard::CardReader::MetaData Struct Reference	201
9.127.1 Detailed Description	201
9.128pdumodel::TransferSwitch_3_1_1::MetaData Struct Reference	201
9.128.1 Detailed Description	202
9.129devsettings::Modbus Interface Reference	202
9.129.1 Detailed Description	202
9.129.2 Member Function Documentation	202
9.129.2.1 getSettings	202
9.129.2.2 setSettings	202
9.130modelpush::ModelPush Interface Reference	203
9.130.1 Detailed Description	203
9.130.2 Member Function Documentation	203
9.130.2.1 getConfiguration	203
9.130.2.2 setConfiguration	203
9.131pdumodel::Nameplate Struct Reference	204
9.131.1 Detailed Description	204
9.132jsonrpc::NameService Interface Reference	204
9.132.1 Detailed Description	205
9.132.2 Member Function Documentation	205
9.132.2.1 lookup	205
9.133net::Net_2_0_2 Interface Reference	205
9.133.1 Detailed Description	206
9.133.2 Member Function Documentation	206
9.133.2.1 getBridgeSlaveCount	206
9.133.2.2 getMACs	206
9.133.2.3 getNetworkConfigInterface	206
9.133.2.4 getNetworkConfigIP	207
9.133.2.5 getNetworkConfigIPv4	207
9.133.2.6 getNetworkConfigIPv6	207
9.133.2.7 getNetworkConfigServices	207
9.133.2.8 setNetworkConfigIP	207
9.133.2.9 setNetworkConfigIPv4	207
9.133.2.10setNetworkConfigIPv6	208

9.133.2.1	setNetworkConfigLan	208
9.133.2.2	setNetworkConfigServices	208
9.133.2.3	setNetworkConfigWlan	208
9.134	net::NetworkActiveValuesIPv6 Struct Reference	209
9.134.1	Detailed Description	209
9.135	net::NetworkConfigIP Struct Reference	209
9.135.1	Detailed Description	210
9.136	net::NetworkConfigIPv4 Struct Reference	210
9.136.1	Detailed Description	210
9.137	net::NetworkConfigIPv6 Struct Reference	211
9.137.1	Detailed Description	211
9.138	datetime::DateTime_2_0_0::NtpCfg Struct Reference	211
9.138.1	Detailed Description	212
9.139	sensors::NumericSensor_4_0_0 Interface Reference	212
9.139.1	Detailed Description	213
9.139.2	Member Function Documentation	213
9.139.2.1	getDefaultThresholds	213
9.139.2.2	getMetaData	213
9.139.2.3	getReading	213
9.139.2.4	getThresholds	214
9.139.2.5	setThresholds	214
9.139.3	Member Data Documentation	214
9.139.3.1	MetaDataChangedEvent	214
9.139.3.2	ReadingChangedEvent	214
9.139.3.3	StateChangedEvent	214
9.139.3.4	ThresholdsChangedEvent	214
9.140	lhxmodel::Sensor_4_0_0::NumThresholds Struct Reference	214
9.140.1	Detailed Description	215
9.141	lhxmodel::Lhx_3_2_1::OpState Struct Reference	215
9.141.1	Detailed Description	216
9.142	pdumodel::Outlet_1_5_6 Interface Reference	216
9.142.1	Detailed Description	217
9.142.2	Member Enumeration Documentation	217
9.142.2.1	PowerState	217
9.142.2.2	StartupState	218
9.142.3	Member Function Documentation	218
9.142.3.1	cyclePowerState	218
9.142.3.2	getController	218
9.142.3.3	getIOP	218
9.142.3.4	getMetaData	218

9.142.3.5 getSensors	219
9.142.3.6 getSettings	219
9.142.3.7 getState	219
9.142.3.8 setPowerState	219
9.142.3.9 setSettings	219
9.142.3.10Unstick	220
9.142.4 Member Data Documentation	220
9.142.4.1 PowerControlEvent	220
9.142.4.2 SettingsChangedEvent	220
9.142.4.3 StateChangedEvent	220
9.143pdumodel::Pdu_3_0_0::OutletSequenceState Struct Reference	220
9.143.1 Detailed Description	221
9.144pdumodel::OutletStatistic Struct Reference	221
9.144.1 Detailed Description	221
9.145pdumodel::OverCurrentProtector_2_1_2 Interface Reference	221
9.145.1 Detailed Description	222
9.145.2 Member Enumeration Documentation	222
9.145.2.1 Type	222
9.145.3 Member Function Documentation	223
9.145.3.1 getInlet	223
9.145.3.2 getMetaData	223
9.145.3.3 getOCP	223
9.145.3.4 getPoles	223
9.145.3.5 getSensors	223
9.145.3.6 getSettings	223
9.145.3.7 setSettings	224
9.145.4 Member Data Documentation	224
9.145.4.1 SettingsChangedEvent	224
9.146peripheral::PackageInfo_2_0_0 Struct Reference	224
9.146.1 Detailed Description	224
9.146.2 Member Enumeration Documentation	225
9.146.2.1 State	225
9.147lhxmodel::Lhx_3_2_1::ParamCfg Struct Reference	225
9.147.1 Detailed Description	225
9.148lhxmodel::Parameter_2_0_1 Interface Reference	225
9.148.1 Detailed Description	226
9.148.2 Member Enumeration Documentation	226
9.148.2.1 Unit	226
9.148.3 Member Function Documentation	227
9.148.3.1 getMetaData	227

9.148.3.2 <code>getRawValue</code>	227
9.148.3.3 <code>getValue</code>	227
9.148.3.4 <code>setRawValue</code>	228
9.148.4 Member Data Documentation	228
9.148.4.1 <code>MetaDataChangedEvent</code>	228
9.148.4.2 <code>ValueChangedEvent</code>	228
9.149 <code>security::PasswordSettings</code> Struct Reference	228
9.149.1 Detailed Description	229
9.150 <code>pduModel::Pdu_3_0_0</code> Interface Reference	229
9.150.1 Detailed Description	231
9.150.2 Member Enumeration Documentation	231
9.150.2.1 <code>StartupState</code>	231
9.150.3 Member Function Documentation	231
9.150.3.1 <code>cycleAllOutletPowerStates</code>	231
9.150.3.2 <code>cycleMultipleOutletPowerStates</code>	231
9.150.3.3 <code>enterRS485ConfigModeAndAssignCtrlBoardAddress</code>	232
9.150.3.4 <code>enterRS485ConfigModeAndAssignSCBoardAddress</code>	232
9.150.3.5 <code>getBeeper</code>	232
9.150.3.6 <code>getControllers</code>	232
9.150.3.7 <code>getFeaturePorts</code>	233
9.150.3.8 <code>getInlets</code>	233
9.150.3.9 <code>getMetaData</code>	233
9.150.3.10 <code>getNameplate</code>	233
9.150.3.11 <code>getOutlets</code>	233
9.150.3.12 <code>getOutletSequenceState</code>	233
9.150.3.13 <code>getOverCurrentProtectors</code>	234
9.150.3.14 <code>getPeripheralDeviceManager</code>	234
9.150.3.15 <code>getSensorLogger</code>	234
9.150.3.16 <code>getSensors</code>	234
9.150.3.17 <code>getSettings</code>	234
9.150.3.18 <code>getStatistic</code>	234
9.150.3.19 <code>getTransferSwitches</code>	234
9.150.3.20 <code>isLoadSheddingActive</code>	235
9.150.3.21 <code>leaveRS485ConfigMode</code>	235
9.150.3.22 <code>setAllOutletPowerStates</code>	235
9.150.3.23 <code>setLoadSheddingActive</code>	235
9.150.3.24 <code>setMultipleOutletPowerStates</code>	235
9.150.3.25 <code>setSettings</code>	236
9.150.4 Member Data Documentation	236
9.150.4.1 <code>LoadSheddingModeChangedEvent</code>	236

9.150.4.2 OutletSequenceStateChangedEvent	236
9.150.4.3 SettingsChangedEvent	236
9.151pdumodel::Bcm::PhaseConfig Struct Reference	236
9.151.1 Detailed Description	237
9.152pdumodel::Pole_2_0_0 Struct Reference	237
9.152.1 Detailed Description	237
9.153auth::Policy Struct Reference	238
9.153.1 Detailed Description	238
9.154portsmodel::Port_2_0_1 Interface Reference	238
9.154.1 Detailed Description	239
9.154.2 Member Enumeration Documentation	239
9.154.2.1 DetectionType	239
9.154.3 Member Function Documentation	239
9.154.3.1 getDetectableDevices	239
9.154.3.2 getDevice	239
9.154.3.3 getDeviceConfig	240
9.154.3.4 getProperties	240
9.154.3.5 setDetectionMode	240
9.154.3.6 setName	240
9.154.4 Member Data Documentation	240
9.154.4.1 DeviceChangedEvent	240
9.154.4.2 NO_ERROR	241
9.154.4.3 PropertiesChangedEvent	241
9.155serial::PortDispatcher_1_1_1 Interface Reference	241
9.155.1 Detailed Description	241
9.155.2 Member Function Documentation	241
9.155.2.1 getPorts	241
9.156peripheral::PosElement Struct Reference	242
9.156.1 Detailed Description	242
9.157pdumodel::PowerQualitySensor_2_0_0 Interface Reference	242
9.157.1 Detailed Description	242
9.158usermgmt::Preferences Struct Reference	243
9.158.1 Detailed Description	243
9.159usermgmt::Role::Privilege Struct Reference	243
9.159.1 Detailed Description	243
9.160usermgmt::RoleManager::PrivilegeDesc Struct Reference	243
9.160.1 Detailed Description	244
9.161production::Production Interface Reference	244
9.161.1 Detailed Description	244
9.161.2 Member Function Documentation	244

9.161.2.1 enterFactoryConfigMode	244
9.161.2.2 isFactoryConfigModeEnabled	245
9.162portsmodel::Port_2_0_1::Properties Struct Reference	245
9.162.1 Detailed Description	245
9.163cascading::Cascading_1_0_1::ProtocolMapping Struct Reference	245
9.163.1 Detailed Description	246
9.164assetmgrmodel::AssetStrip_2_0_1::RackUnitInfo Struct Reference	246
9.164.1 Detailed Description	246
9.165assetmgrmodel::AssetStripConfig_1_0_1::RackUnitSettings Struct Reference	246
9.165.1 Detailed Description	247
9.166auth::RadiusManager Interface Reference	247
9.166.1 Detailed Description	248
9.166.2 Member Function Documentation	248
9.166.2.1 getRadiusServers	248
9.166.2.2 setRadiusServers	248
9.166.2.3 testRadiusServer	248
9.167event::TimerEventManager_2_0_0::Range Struct Reference	248
9.167.1 Detailed Description	249
9.168sensors::NumericSensor_4_0_0::Range Struct Reference	249
9.168.1 Detailed Description	249
9.169pdumodel::Rating Struct Reference	249
9.169.1 Detailed Description	250
9.170pdumodel::Nameplate::Rating Struct Reference	250
9.170.1 Detailed Description	250
9.171sensors::NumericSensor_4_0_0::Reading Struct Reference	250
9.171.1 Detailed Description	251
9.172lhxmodel::Sensor_4_0_0::Reading Struct Reference	251
9.172.1 Detailed Description	251
9.173sensors::Logger_2_1_2::Record Struct Reference	251
9.173.1 Detailed Description	252
9.174assetmgrmodel::AssetStripLogger_1_0_2::Record Struct Reference	252
9.174.1 Detailed Description	253
9.175cert::ServerSSLCert::ReqInfo Struct Reference	253
9.175.1 Detailed Description	253
9.176pdumodel::ResidualCurrentStateSensor_2_0_0 Interface Reference	253
9.176.1 Detailed Description	254
9.176.2 Member Function Documentation	254
9.176.2.1 startSelfTest	254
9.176.3 Member Data Documentation	254
9.176.3.1 STATE_NORMAL	254

9.177res_mon::ResMon Interface Reference	254
9.177.1 Detailed Description	255
9.177.2 Member Function Documentation	255
9.177.2.1 getDataEntries	255
9.178security::RestrictedServiceAgreement Struct Reference	255
9.178.1 Detailed Description	255
9.179test::Result Struct Reference	255
9.179.1 Detailed Description	256
9.180usermgmt::Role Interface Reference	256
9.180.1 Detailed Description	256
9.180.2 Member Function Documentation	256
9.180.2.1 getInfo	256
9.180.2.2 updateFull	257
9.181security::RoleAccessControl Struct Reference	257
9.181.1 Detailed Description	257
9.182security::RoleAccessRule Struct Reference	257
9.182.1 Detailed Description	258
9.183usermgmt::RoleManager::RoleAccount Struct Reference	258
9.183.1 Detailed Description	258
9.184usermgmt::RoleManager Interface Reference	258
9.184.1 Detailed Description	259
9.184.2 Member Function Documentation	259
9.184.2.1 createRoleFull	259
9.184.2.2 deleteRole	260
9.184.2.3 getAllPrivileges	260
9.184.2.4 getAllRoleNames	260
9.184.2.5 getAllRoles	260
9.184.2.6 getInfo	260
9.185test::RS232Serial Interface Reference	260
9.185.1 Detailed Description	261
9.186event::Engine::Rule Struct Reference	261
9.186.1 Detailed Description	261
9.187pdumodel::Ade::Sample Struct Reference	262
9.187.1 Detailed Description	262
9.188event::TimerEventManager_2_0_0::Schedule Struct Reference	262
9.188.1 Detailed Description	263
9.189security::Security_3_0_0 Interface Reference	263
9.189.1 Detailed Description	264
9.189.2 Member Function Documentation	264
9.189.2.1 getBlockSettings	264

9.189.2.2	getFrontPanelPrivileges	264
9.189.2.3	getHttpRedirSettings	265
9.189.2.4	getIdleTimeoutSettings	265
9.189.2.5	getRestrictedServiceAgreement	265
9.189.2.6	getSettings	265
9.189.2.7	getSSHSettings	265
9.189.2.8	getSupportedFrontPanelPrivileges	265
9.189.2.9	setBlockSettings	265
9.189.2.10	setFrontPanelPrivileges	266
9.189.2.11	setHttpRedirSettings	266
9.189.2.12	setIdleTimeoutSettings	266
9.189.2.13	setIpFwSettings	266
9.189.2.14	setIpV6FwSettings	267
9.189.2.15	setPwSettings	267
9.189.2.16	setRestrictedServiceAgreement	267
9.189.2.17	setRoleAccessControlSettings	267
9.189.2.18	setRoleAccessControlSettingsV6	268
9.189.2.19	setSettings	268
9.189.2.20	setSingleLoginLimitation	268
9.189.2.21	setSSHSettings	268
9.190	lhxmodel::Sensor_4_0_0 Interface Reference	268
9.190.1	Detailed Description	270
9.190.2	Member Function Documentation	270
9.190.2.1	getMetaData	270
9.190.2.2	getReading	270
9.190.2.3	getThresholds	270
9.190.2.4	setThresholds	270
9.190.3	Member Data Documentation	271
9.190.3.1	ReadingChangedEvent	271
9.190.3.2	StateChangedEvent	271
9.190.3.3	ThresholdsChangedEvent	271
9.191	sensors::Sensor_4_0_0 Interface Reference	271
9.191.1	Detailed Description	274
9.191.2	Member Enumeration Documentation	274
9.191.2.1	OnOffState	274
9.191.3	Member Function Documentation	274
9.191.3.1	getTypeSpec	274
9.191.3.2	setType	274
9.191.4	Member Data Documentation	274
9.191.4.1	TypeSpecChangedEvent	274

9.192pdumodel::OverCurrentProtector_2_1_2::Sensors Struct Reference	274
9.192.1 Detailed Description	275
9.193pdumodel::Pdu_3_0_0::Sensors Struct Reference	275
9.193.1 Detailed Description	275
9.194pdumodel::Inlet_1_2_6::Sensors Struct Reference	275
9.194.1 Detailed Description	276
9.195pdumodel::TransferSwitch_3_1_1::Sensors Struct Reference	276
9.195.1 Detailed Description	277
9.196pdumodel::Outlet_1_5_6::Sensors Struct Reference	277
9.196.1 Detailed Description	278
9.197sensors::Logger_2_1_2::SensorSet Struct Reference	278
9.197.1 Detailed Description	278
9.198serial::SerialPort_2_0_0 Interface Reference	278
9.198.1 Detailed Description	279
9.198.2 Member Enumeration Documentation	279
9.198.2.1 BaudRate	279
9.198.2.2 PortState	280
9.198.3 Member Function Documentation	280
9.198.3.1 getModem	280
9.198.3.2 getSettings	280
9.198.3.3 getState	280
9.198.3.4 setSettings	280
9.198.4 Member Data Documentation	280
9.198.4.1 ModemEvent	281
9.198.4.2 SUCCESS	281
9.199servermon::ServerMonitor_2_0_0::Server Struct Reference	281
9.199.1 Detailed Description	281
9.200servermon::ServerMonitor_2_0_0 Interface Reference	281
9.200.1 Detailed Description	282
9.200.2 Member Enumeration Documentation	282
9.200.2.1 ServerReachability	282
9.200.3 Member Function Documentation	283
9.200.3.1 addServer	283
9.200.3.2 deleteServer	283
9.200.3.3 getServer	283
9.200.3.4 listServers	283
9.200.3.5 modifyServer	284
9.201auth::ldapsrv::ServerSettings Struct Reference	284
9.201.1 Detailed Description	285
9.202radius::ServerSettings Struct Reference	285

9.202.1 Detailed Description	285
9.203servermon::ServerMonitor_2_0_0::ServerSettings Struct Reference	285
9.203.1 Detailed Description	286
9.204cert::ServerSSLCert Interface Reference	286
9.204.1 Detailed Description	287
9.204.2 Member Function Documentation	287
9.204.2.1 generateSelfSignedKeyPair	287
9.204.2.2 generateUnsignedKeyPair	287
9.204.2.3 getInfo	287
9.204.2.4 installPendingKeyPair	288
9.205servermon::ServerMonitor_2_0_0::ServerStatus Struct Reference	288
9.205.1 Detailed Description	288
9.206event::Service_1_0_1 Interface Reference	288
9.206.1 Detailed Description	289
9.206.2 Member Function Documentation	289
9.206.2.1 createChannel	289
9.206.2.2 destroyChannel	289
9.207security::ServiceAuthorization Interface Reference	289
9.207.1 Detailed Description	290
9.207.2 Member Function Documentation	290
9.207.2.1 setPassword	290
9.208net::ServiceConfig Struct Reference	290
9.208.1 Detailed Description	291
9.209session::Session Struct Reference	291
9.209.1 Detailed Description	291
9.210session::SessionManager Interface Reference	291
9.210.1 Detailed Description	292
9.210.2 Member Enumeration Documentation	292
9.210.2.1 CloseReason	292
9.210.3 Member Function Documentation	293
9.210.3.1 closeCurrentSession	293
9.210.3.2 closeSession	293
9.210.3.3 getCurrentSession	293
9.210.3.4 getSession	293
9.210.3.5 getSessionHistory	293
9.210.3.6 getSessions	293
9.210.3.7 newSession	294
9.210.3.8 touchCurrentSession	294
9.210.3.9 touchSession	294
9.211security::Security_3_0_0::Settings Struct Reference	294

9.211.1 Detailed Description	295
9.212devsettings::Zeroconf::Settings Struct Reference	295
9.212.1 Detailed Description	295
9.213lhxmodel::Lhx_3_2_1::Settings Struct Reference	296
9.213.1 Detailed Description	296
9.214serial::GsmModem_1_0_1::Settings Struct Reference	296
9.214.1 Detailed Description	296
9.215pdumodel::Pdu_3_0_0::Settings Struct Reference	296
9.215.1 Detailed Description	297
9.215.2 Member Data Documentation	297
9.215.2.1 outletPowerStateSequence	297
9.216pdumodel::OverCurrentProtector_2_1_2::Settings Struct Reference	297
9.216.1 Detailed Description	298
9.217serial::AnalogModem::Settings Struct Reference	298
9.218devsettings::Modbus::Settings Struct Reference	298
9.218.1 Detailed Description	298
9.219pdumodel::Outlet_1_5_6::Settings Struct Reference	298
9.219.1 Detailed Description	299
9.220peripheral::DeviceSlot_2_0_0::Settings Struct Reference	299
9.220.1 Detailed Description	299
9.221sensors::Logger_2_1_2::Settings Struct Reference	299
9.221.1 Detailed Description	300
9.222peripheral::DeviceManager_2_0_0::Settings Struct Reference	300
9.222.1 Detailed Description	300
9.223pdumodel::Unit_2_0_1::Settings Struct Reference	301
9.223.1 Detailed Description	301
9.224pdumodel::TransferSwitch_3_1_1::Settings Struct Reference	301
9.224.1 Detailed Description	301
9.225pdumodel::Inlet_1_2_6::Settings Struct Reference	302
9.225.1 Detailed Description	302
9.226serial::SerialPort_2_0_0::Settings Struct Reference	302
9.226.1 Detailed Description	302
9.227webcam::Settings_2_0_0 Struct Reference	302
9.227.1 Detailed Description	303
9.228devsettings::Sntp_1_0_1 Interface Reference	303
9.228.1 Detailed Description	303
9.228.2 Member Function Documentation	304
9.228.2.1 getConfiguration	304
9.228.2.2 setConfiguration	304
9.228.2.3 testConfiguration	304

9.229devsettings::Snmp_1_0_2 Interface Reference	304
9.229.1 Detailed Description	305
9.229.2 Member Function Documentation	305
9.229.2.1 getConfiguration	305
9.229.2.2 getV3EngineId	305
9.229.2.3 setConfiguration	305
9.230um::SnmpV3 Interface Reference	306
9.230.1 Detailed Description	306
9.230.2 Member Enumeration Documentation	306
9.230.2.1 AuthProtocol	306
9.230.2.2 PrivProtocol	306
9.230.2.3 SecurityLevel	307
9.231usermgmt::SnmpV3Settings Struct Reference	307
9.231.1 Detailed Description	307
9.232security::SSHSettings Struct Reference	308
9.232.1 Detailed Description	308
9.233serial::SerialPort_2_0_0::State Struct Reference	308
9.233.1 Detailed Description	308
9.234pdumodel::Outlet_1_5_6::State Struct Reference	308
9.234.1 Detailed Description	309
9.234.2 Member Data Documentation	309
9.234.2.1 cycleInProgress	309
9.234.2.2 switchOnInProgress	309
9.235sensors::StateSensor_4_0_0::State Struct Reference	309
9.235.1 Detailed Description	310
9.236sensors::StateSensor_4_0_0 Interface Reference	310
9.236.1 Detailed Description	310
9.236.2 Member Function Documentation	310
9.236.2.1 getState	310
9.237pdumodel::Pdu_3_0_0::Statistic Struct Reference	311
9.237.1 Detailed Description	311
9.238pdumodel::TransferSwitch_3_1_1::Statistics Struct Reference	311
9.238.1 Detailed Description	312
9.239peripheral::DeviceManager_2_0_0::Statistics Struct Reference	312
9.239.1 Detailed Description	312
9.240sensors::NumericSensor_4_0_0::Reading::Status Struct Reference	312
9.240.1 Detailed Description	312
9.241lhxmodel::Parameter_2_0_1::Status Struct Reference	313
9.241.1 Detailed Description	313
9.242webcam::StorageManager_1_0_1::StorageImage Struct Reference	313

9.242.1 Detailed Description	313
9.243webcam::StorageManager_1_0_1::StorageInformation Struct Reference	313
9.243.1 Detailed Description	314
9.244webcam::StorageManager_1_0_1 Interface Reference	314
9.244.1 Detailed Description	315
9.244.2 Member Enumeration Documentation	315
9.244.2.1 Direction	315
9.244.2.2 StorageStatus	316
9.244.2.3 StorageType	316
9.244.3 Member Function Documentation	316
9.244.3.1 addImage	316
9.244.3.2 getActivities	316
9.244.3.3 getImages	317
9.244.3.4 getInformation	317
9.244.3.5 getMetaData	317
9.244.3.6 getSettings	317
9.244.3.7 getSupportedStorageTypes	318
9.244.3.8 removeImages	318
9.244.3.9 setSettings	318
9.244.3.10startActivity	318
9.244.3.11stopActivity	319
9.244.4 Member Data Documentation	319
9.244.4.1 NO_ERROR	319
9.245webcam::StorageManager_1_0_1::StorageMetaData Struct Reference	319
9.245.1 Detailed Description	319
9.246webcam::StorageManager_1_0_1::StorageSettings Struct Reference	319
9.246.1 Detailed Description	320
9.247assetmgrmodel::AssetStrip_2_0_1::StripInfo Struct Reference	320
9.247.1 Detailed Description	320
9.248assetmgrmodel::AssetStripConfig_1_0_1::StripSettings Struct Reference	321
9.248.1 Detailed Description	321
9.249lhx::Support Interface Reference	321
9.249.1 Detailed Description	321
9.249.2 Member Function Documentation	322
9.249.2.1 isEnabled	322
9.249.2.2 setEnabled	322
9.250sensors::Switch_2_0_0 Interface Reference	322
9.250.1 Detailed Description	323
9.250.2 Member Function Documentation	323
9.250.2.1 setState	323

9.251 sys::System Interface Reference	323
9.251.1 Detailed Description	323
9.251.2 Member Function Documentation	323
9.251.2.1 isDaemonRunning	323
9.251.2.2 restartDaemon	324
9.252 assetmgrmodel::AssetStrip_2_0_1::TagChangeInfo Struct Reference	324
9.252.1 Detailed Description	324
9.253 assetmgrmodel::AssetStrip_2_0_1::TagInfo Struct Reference	324
9.253.1 Detailed Description	325
9.254 devsettings::Modbus::TcpSettings Struct Reference	325
9.254.1 Detailed Description	325
9.255 devsettings::Sntp_1_0_1::TestResult Struct Reference	325
9.255.1 Detailed Description	326
9.256 sensors::NumericSensor_4_0_0::ThresholdCapabilities Struct Reference	326
9.256.1 Detailed Description	326
9.257 sensors::NumericSensor_4_0_0::Thresholds Struct Reference	326
9.257.1 Detailed Description	327
9.258 pdumodel::ThrowPole Struct Reference	327
9.258.1 Detailed Description	327
9.259 sensors::Logger_2_1_2::TimedRecord Struct Reference	327
9.259.1 Detailed Description	328
9.260 event::TimerEventManager_2_0_0::TimerEvent Struct Reference	328
9.260.1 Detailed Description	328
9.261 event::TimerEventManager_2_0_0 Interface Reference	328
9.261.1 Detailed Description	330
9.261.2 Member Function Documentation	330
9.261.2.1 addTimerEvent	330
9.261.2.2 deleteTimerEvent	330
9.261.2.3 modifyTimerEvent	330
9.261.3 Member Data Documentation	331
9.261.3.1 JAN	331
9.261.3.2 NO_ERROR	331
9.261.3.3 SUN	331
9.262 pdumodel::TransferSwitch_3_1_1 Interface Reference	331
9.262.1 Detailed Description	333
9.262.2 Member Enumeration Documentation	333
9.262.2.1 TransferReason	333
9.262.2.2 Type	333
9.262.3 Member Function Documentation	334
9.262.3.1 getLastTransferReason	334

9.262.3.2 getLastTransferWaveform	334
9.262.3.3 getMetaData	334
9.262.3.4 getPoles	334
9.262.3.5 getSensors	334
9.262.3.6 getSettings	334
9.262.3.7 getStatistics	335
9.262.3.8 setSettings	335
9.262.3.9 transferToSource	335
9.262.4 Member Data Documentation	335
9.262.4.1 SettingsChangedEvent	335
9.263sensors::Sensor_4_0_0::TypeSpec Struct Reference	335
9.263.1 Detailed Description	336
9.264test::Unit_1_0_2 Interface Reference	336
9.264.1 Detailed Description	336
9.264.2 Member Function Documentation	336
9.264.2.1 getButtonStates	336
9.264.2.2 getDisplays	337
9.264.2.3 setBuzzer	337
9.264.2.4 triggerSlaveControllerWatchdog	337
9.265pdumodel::Unit_2_0_1 Interface Reference	337
9.265.1 Detailed Description	338
9.265.2 Member Enumeration Documentation	338
9.265.2.1 Orientation	338
9.265.3 Member Function Documentation	338
9.265.3.1 getDisplayOrientation	338
9.265.3.2 getMetaData	338
9.265.3.3 getSettings	339
9.265.3.4 identify	339
9.265.3.5 muteBuzzer	339
9.265.3.6 setSettings	339
9.265.4 Member Data Documentation	339
9.265.4.1 IdentificationStartedEvent	339
9.266firmware::UpdateHistoryEntry Struct Reference	339
9.266.1 Detailed Description	340
9.267firmware::UpdateStatus Struct Reference	340
9.267.1 Detailed Description	340
9.268usb::Usb Interface Reference	340
9.268.1 Detailed Description	341
9.268.2 Member Function Documentation	341
9.268.2.1 getDevices	341

9.269usb::UsbDevice Struct Reference	341
9.269.1 Detailed Description	341
9.270usermgmt::User_1_0_1 Interface Reference	341
9.270.1 Detailed Description	342
9.270.2 Member Function Documentation	343
9.270.2.1 getCapabilities	343
9.270.2.2 getInfo	343
9.270.2.3 getInfoAndPrivileges	343
9.270.2.4 setAccountPassword	343
9.270.2.5 setPreferences	343
9.270.2.6 updateAccountFull	344
9.271usermgmt::UserCapabilities Struct Reference	344
9.271.1 Detailed Description	344
9.272usermgmt::UserInfo Struct Reference	345
9.272.1 Detailed Description	345
9.273usermgmt::UserManager_1_0_2 Interface Reference	345
9.273.1 Detailed Description	347
9.273.2 Member Function Documentation	347
9.273.2.1 createAccount	347
9.273.2.2 createAccountFull	347
9.273.2.3 deleteAccount	348
9.273.2.4 getAccountNames	348
9.273.2.5 getAccountsByRole	348
9.273.2.6 getAllAccounts	348
9.273.2.7 getDefaultPreferences	348
9.273.2.8 setDefaultPreferences	349
9.274lhxmodel::Parameter_2_0_1::Value Struct Reference	349
9.274.1 Detailed Description	349
9.275peripheral::PackageInfo_2_0_0::FirmwareInfo::Version Struct Reference	349
9.276pdumodel::TransferSwitch_3_1_1::WaveformSample Struct Reference	350
9.276.1 Detailed Description	350
9.277webcam::Webcam_2_0_0 Interface Reference	350
9.277.1 Detailed Description	350
9.277.2 Member Function Documentation	350
9.277.2.1 getControlDefaults	350
9.277.2.2 getInformation	351
9.277.2.3 getSettings	351
9.277.2.4 setControls	351
9.277.2.5 setSettings	351
9.278webcam::WebcamManager_2_0_0 Interface Reference	351

9.278.1 Detailed Description	352
9.278.2 Member Function Documentation	352
9.278.2.1 getChannel	352
9.278.2.2 getChannels	353
9.278.2.3 getClientTypePriorities	353
9.278.2.4 getClientTypes	353
9.278.2.5 getWebcamPriorities	353
9.278.2.6 getWebcams	353
9.278.2.7 removeClientType	353
9.278.2.8 setClientTypePriorities	354
9.278.2.9 setWebcamPriorities	354
9.278.3 Member Data Documentation	354
9.278.3.1 NO_ERROR	354
9.279webcam::StorageManager_1_0_1::WebcamStorageInfo Struct Reference	354
9.279.1 Detailed Description	355
9.280net::WirelessInterfaceSettings Struct Reference	355
9.280.1 Detailed Description	355
9.281devsettings::Zeroconf Interface Reference	355
9.281.1 Detailed Description	356
9.281.2 Member Function Documentation	356
9.281.2.1 getSettings	356
9.281.2.2 setSettings	356
9.282datetime::DateTime_2_0_0::ZoneCfg Struct Reference	356
9.282.1 Detailed Description	357
9.283datetime::DateTime_2_0_0::ZoneInfo Struct Reference	357
9.283.1 Detailed Description	357

Chapter 1

Rules and Mechanism for Mapping PX2/EMX-IDL to JSON-RPC

1.1 A) Scope

PX2/EMX devices provide a remote interface. The interface is a **REST** architecture and utilizes **JSON-RPC version 2.0** for formatting messages sent to and received from the device. This document defines how formal PX2/EMX-IDL is mapped to JSON-RPC.

1.2 B) Mapping Rules

1.2.1 B)1 Type Mapping

IDL types are mapped to **JSON** value types as follows:

- boolean: true or false
- int: number
- long: number
- float: number
- double: number
- string: string
- time: int (containing number of seconds since the start of the Unix epoch: midnight UTC of January 1, 1970)
- vector: array
- map: object (array of objects, each containing two elements: 'key' contains the entry's key, 'value' contains the entry's value)
- structure: object (containing elements of pairs, structure's element name is mapped to pair's name, value to value respectively)
- object-reference: object (containing two elements: 'rid' element contains object identifier, 'type' element contains most derived interface type of object)

1.2.2 B)2 Interface version Mapping

Interface versions as specified in IDL-File (for example NumericSensor_1_0_1) are not mapped to the protocol as such as interface identifiers are not directly part of any protocol message. However interface version is part of the type information which is sent along with object references (see section B)1). Clients may use type information and hence version information to select a compatible proxy for a received object reference.

The interface changes between firmware releases are documented in the file [Changelog.txt](#).

1.2.3 B)3 Method Mapping

IDL method are mapped to JSON-RPC calls as follows:

in JSON-RPC Request:

- method name: element 'method' of request object
- method in-parameters: element 'params' of request object, containing an object with elements where elements names correspond to in-parameter names and element values corresponds to in-parameter values
- method out-parameters: not mapped
- method return-parameter: not mapped

in JSON-RPC Response:

- method name: not mapped
- method in-parameters: not mapped
- method out-parameters: element 'result' of response object, containing an object with elements where element names correspond to out-parameter name and element values corresponds to out-parameter value
- method return-parameter: special element named '*ref*' of element 'result' of response object
- system errors / exceptions: element 'error' of response object

1.3 C) Call Execution

PX2/EMX-JSON-RPC uses HTTP-POST or HTTPS-POST requests. JSON-RPC request message is put into POST data, JSON-RPC response message is delivered with HTTP/HTTPS response.

URL of HTTP/HTTPS serves as object reference which uniquely identifies a particular resource that implements a particular IDL interface. There are well known references that must be used in initial request. All well known references are defined in file [Well-Known-URIs.txt](#).

JSON-RPC responses may contain other object references (see B)1). Those references are never well known references but opaque references. The 'rid' element of the opaque references must be used as URI in further call directed to this resource.

1.4 D) Language Bindings and Examples

IDL definitions may be translated to so called language bindings, i.e. translations of the IDL and the JSON-RPC protocol to concrete programming languages. Language bindings for Perl and Python are delivered along with this documentation. They are described in the sections hereafter. Additionally examples are given using command line tool curl.

Please note that the examples given are not specific to the product this API documentation was released for. Examples use APIs of PX2 or EMX. Nevertheless the principles of the API usage is identical for either product. For the very concrete API refer to the according pages of this document.

- [Curl JSON-RPC Example](#)
- [Perl JSON-RPC Client Bindings](#)
- [Python JSON-RPC Client Binding](#)

Chapter 2

Curl JSON-RPC Example

2.1 Mechanism

Curl is a cross-platform command line tool for getting or sending files using URL syntax. Using the tool a HTTP--POST request can be formulated that contains the JSON-RPC request. The response document is printed on the console of Curl. This is a direct way to check out particular objects and their methods.

Authentication data must be provided using options of curl or as part of the URL given to curl.

In case HTTPS is used as protocol and the certificate installed on the device cannot be validated option '-k' must be provided. This will allow for insecure connections.

In order to direct the request to the desired object implementing a particular interface a well known resource identifier must be used as part of the URL. All well known URIs are defined in file [Well-Known-URIs.txt](#). Note that the link points to a file which is specific for one product, EMX or PX.

2.2 Examples

- get device information of an EMX:

```
curl -skd '{ "jsonrpc": "2.0", "method": "getFullInfo", "id": 23 }' https://  
admin:raritan@10.0.42.2/model/pdu/0 | indent  
  
{  
  "jsonrpc": "2.0",  
  "result": {  
    "_ret_": {  
      "nameplate": {  
        "manufacturer": "Raritan",  
        "model": "EMX2-888-v0",  
        "partNumber": "emx2-888",  
        "serialNumber": "PH81234567",  
        "rating": {  
          "voltage": "100-240V",  
          "current": "0.7A",  
          "frequency": "50/60Hz",  
          "power": "70VA"  
        },  
        "imageFileURL": ""  
      },  
      "ctrlBoardSerial": "PNW1691120",  
      "hwRevision": "0x01",  
      "fwRevision": "2.2.0.5-0",  
      "macAddress": "fe:00:10:42:00:67"  
    },  
    "id": 23  
  }  
}
```

- switch first outlet of a PX:

```
curl -skd '{ "jsonrpc": "2.0", "method": "setPowerState", "params": { "pstate": 1}, "id": 23 }\'
```

```
https://admin:raritan@192.168.7.67/model/pdu/0/outlet/0 | indent  
  
{  
  "jsonrpc": "2.0",  
  "result": {  
    "_ret_": 0  
  },  
  "id": 23  
}
```

Chapter 3

Perl JSON-RPC Client Bindings

3.1 Synopsis

```
use Raritan::RPC;

my $agent = new Raritan::RPC::Agent('https://my-px2', 'admin', 'raritan');
my $pdu = $agent->createProxy('/model/pdu/0');
my $outlets = $pdu->getOutlets();
$outlets->[0]->cyclePowerState();
```

3.2 IDL-to-Perl Mapping

3.2.1 IDL Modules and Perl Package Names

All classes dealing with Raritan JSON-RPC communication are located in Perl packages starting with `Raritan::RPC`. All named sections in IDL files (`module`, `interface` and `enumeration`) provide a scope for their included identifiers and are added as further components to the Perl package name.

3.2.2 Pre-Requirements

Raritan's JSON-RPC Perl bindings require the following perl packages:

- `LWP::Protocol::https`
- `LWP::UserAgent`
- `HTTP::Request::Common`
- `JSON`
- `JSON::RPC::Common::Marshal::HTTP`
- `Error`
- `Data::Dumper`
- `parent`

3.2.2.1 Linux

In yum-based Linux distributions the following command will install the required packages:

```
yum install "perl(LWP::Protocol::https)" "perl(LWP::UserAgent)" \
"perl(HTTP::Request::Common)" "perl(JSON)" \
"perl(JSON::RPC::Common::Marshal::HTTP)" "perl(Error)" \
"perl(Data::Dumper)" "perl(parent)"
```

3.2.2.2 Windows

There are various Perl distributions available for Windows (see <http://perl.org/get.html>). The bindings were tested with ActiveState Perl (<http://www.activestate.com/activeperl/downloads>). After installing ActiveState Perl, the required modules can be installed with the following command:

```
ppm install LWP::Protocol::https LWP::UserAgent HTTP::Request::Common \
JSON JSON::RPC::Common::Marshal::HTTP Error Data::Dumper parent
```

3.2.3 Interfaces and Methods

IDL interfaces are mapped to Perl proxy classes which provide all methods defined by the IDL interface and its bases. Perl methods require the same number of scalar arguments as defined in the IDL signature. Simple types like integers, strings or enumerations are passed by scalar value; structures, vectors and maps are passed by reference. Output parameters are passed as references to a scalar variable which will be modified to point to the returned value. If the method is declared to have a non-void return type the Perl method will return a single scalar value.

Examples:

```
# no parameters, no return value
$firmware->reboot();

# two input parameters, integer return value
$ret = $user_manager->createAccount('newuser', 'newpassword');

# one output parameter, integer return value
$ret = $diagnostics->listTcpConnections(\$results);
```

3.2.4 Structures

IDL structures are mapped as scalar references to Perl hashes. Each element of the structure results in a hash entry whose key is the element identifier and whose value is a Perl scalar. When passing a structure to a method call additional hash entries are ignored.

Example:

```
$ssh_settings = {
    'allowPasswordAuth' => 1,
    'allowPublicKeyAuth' => 0
};
$security->setSSHSettings($ssh_settings);
```

3.2.5 Enumerations

Enumerated values are referenced by their fully-qualified name:

```
$ret = $outlet->setPowerState(Raritan::RPC::pdumodel::Outlet::PowerState::PS_ON);
```

3.2.6 Vectors

Vectors are mapped as scalar references to Perl lists. Each list entry must be a scalar.

```
$recipients = [ 'somebody@invalid.com', 'somebody.else@invalid.com' ];
$smtp->testConfiguration($cfg, $recipients);
```


3.2.7 Maps

Maps are mapped as scalar references to Perl hashes:

```
$priorities = {  
  'important_client' => Raritan::RPC::webcam::Priority::VERY_HIGH,  
  'other_client'     => Raritan::RPC::webcam::Priority::LOW  
};  
$ret = $webcam->setClientTypePriorities($priorities);
```

3.3 Exceptions

In case of error conditions during a proxy method call or the `createProxy` function exceptions will be thrown. They can be caught using the `try` statement from the `Perl::Error` module.

The following exceptions are defined:

- **Raritan::RPC::HttpException**

An error occurred during HTTP communication, e.g. in case the connection was refused or the authentication failed. Please check the agent URL, the resource ID and your authentication credentials.

- **Raritan::RPC::JsonRpcSyntaxException**

The response from the server was not a valid JSON object. Please check the agent URL and the resource ID.

- **Raritan::RPC::JsonRpcErrorException**

The method call failed. This should not happen under normal circumstances and indicates a problem on the device.

3.4 Raritan::RPC::Agent Method Reference

3.4.1 Constructor: `new Raritan::RPC::Agent($url, $user, $password)`

Creates a new agent.

Parameters:

- **\$url** Base URL of the device
- **\$user** User Name (optional)
- **\$password** Password (optional)

3.4.2 `set_auth_basic($user, $password)`

This method enables HTTP basic authentication using username and password.

Parameters:

- **\$user** User Name (optional)
- **\$password** Password (optional)

3.4.3 `set_auth_basic($token)`

This method enables HTTP authentication using a session token. A session token can be obtained by calling the `newSession` method of the [session.SessionManager](#) interface.

Parameters:

- **\$token** Session Token

3.4.4 `timeout($seconds)`

Returns and optionally sets the request timeout.

Parameters:

- **\$seconds** New timeout in seconds (optional)

Returns:

- Previous timeout in seconds

3.4.5 `createProxy($rid, $basetype)`

Creates a new proxy object for the given resource id.

This method will query the type information for the specified resource ID and create a proxy object which provides all methods defined in the IDL definition.

When connecting to old firmware version (PX2 2.2 and earlier, EMX 2.0 and earlier) the type information of a resource ID cannot be queried. In this case the **\$basetype** argument is required.

Parameters:

- **\$rid** The resource ID, e.g. `"/auth/user/admin"`
- **\$basetype** The base type, e.g. `"usermgmt.User"` (optional)

Returns:

- A new proxy object for the specified resource ID

3.4.6 `set_verbose($verbose)`

Enables request tracing. When verbose mode is enabled all JSON-RPC requests and responses will be echoed to the console.

Parameters:

- **\$verbose** `true` to enable verbose mode

Chapter 4

Python JSON-RPC Client Binding

4.1 Examples

4.1.1 Getting device information of EMX:

- Code Snippet:

```
from raritan import rpc
from raritan.rpc import emdmodel

agent = rpc.Agent("https", "MyEMX", "admin", "raritan")
emx = emdmodel.Emd("/model/emd", agent)
mdata = e.getMetaData()
print mdata
```

- Output produced if there is no error:

```
emdmodel.Emd.MetaData:
  nameplate      = pdumodel.Nameplate:
    manufacturer = Raritan
    model        = EMX2-888-v0
    partNumber   = emx2-888
    serialNumber = PH81234567
  rating         = pdumodel.Nameplate.Rating:
    voltage      = 100-240V
    current      = 0.7A
    frequency    = 50/60Hz
    power        = 70VA
  imageUrlURL    =
  ctrlBoardSerial = PNW1691120
  hwRevision      = 0x01
  fwRevision      = 2.2.0.5-0
  macAddress      = fe:00:10:42:00:67
```

4.1.2 Switch first outlet of PX:

- Code Snippet:

```
from raritan import rpc
from raritan.rpc import pdumodel

agent = rpc.Agent("https", "MyPX", "admin", "raritan")
pdu = pdumodel.Pdu("/model/pdu/0", agent)
outlets = pdu.getOutlets()
outlets[0].setPowerState(pdumodel.Outlet.PowerState.PS_OFF)
```

Note that the URIs given to the object proxies (such as `"/model/pdu/0"`), are well known references to static (i.e. they are always there) object instances implemented in the device. All well known references along with their interface are documented in file `Well-Known-URIs.txt` which comes along with the IDL files.

4.2 IDL-to-Python Mapping

4.2.1 Modules and Python Packages

All supporting and generated symbols are placed underneath common package `raritan.rpc`. IDL Modules are mapped to corresponding Python package within this common package. Import of all definitions within a module can be done via (for instance):

```
from raritan.rpc.pdumodel import *
```

4.2.2 Interfaces and Methods

IDL interfaces are mapped to Python classes which are defined in the `init.py` file of their according package. The python class implements a proxy that delegates calls to its IDL defined methods to the actual object implementation across the network.

The signature of the methods is slightly altered compared to the IDL signature. All `in` arguments are expected as arguments to the Python method in the same order as they appear in IDL. The return argument, if any, and all `out` arguments are returned as a Python tuple in the same order they appear in IDL. That means if there is a return argument defined it will appear as first element in the tuple, otherwise the first `out` argument is the first element in the tuple. If the IDL method has either only a return value or a single out argument, a single value is returned by the Python method without an embracing tuple.

4.2.3 Structures

IDL structures are mapped to Python classes which are, like all other definitions, defined in the `init.py` file of their according package. The class has all elements as defined in IDL. In addition there is a constructor with a signature that initializes all elements.

The following is an example for enabling a threshold in a sensor:

```
from raritan import rpc
from raritan.rpc import pdumodel, sensors

agent = rpc.Agent("https", "MyPX", "admin", "raritan")
outlet = emdmodel.Emd("/model/pdu/0/outlet/0", agent)
currentsens = outlet.getSensors().current
thresh = currentsens.getThresholds()
thresh.lowerWarning = 1.0
thresh.lowerWarningActive = True
currentsens.setThresholds(thresh)
```

4.2.4 Enumerations

An IDL enumeration is mapped to a concrete class which has an element for each enumerated value. That means access to a particular enumerated value is accomplished by naming the enumeration type and the enumerated value. Consider the following example for switching an outlet:

```
# pdumodel has been imported and outlet has been initialized before, see above
outlet.setPowerState(pdumodel.Outlet.PowerState.PS_OFF)
```

`PowerState` is an enumerated type defined in the `Outlet` interface. `PS_OFF` is a specific enumerated value defined in the `PowerState` enumeration.

4.2.5 Vectors

IDL vectors are mapped to Python lists. Consider the following example for looping over the list of all outlets of a Pdu and printing out informative data:

```

from raritan import rpc
from raritan.rpc import pdumodel

agent = rpc.Agent("https", "MyPX", "admin", "raritan")
pdu = pdumodel.Pdu("/model/pdu/0", agent)
outlets = pdu.getOutlets()
for outlet in outlets:
    print outlet.getMetaData()

```

4.2.6 Maps

IDL maps are mapped to Python dictionaries. Consider the following example for looping over all observed servers of the `servermon.ServerMonitor` and retrieving their entry id mapped to server entry structure which contains information and status:

```

from raritan import rpc
from raritan.rpc import servermon

agent = rpc.Agent("https", "MyEMX", "admin", "raritan")
srvmonitor = servermon.ServerMonitor("/servermon", agent);
servers = srvmonitor.listServers()
for srid, srventry in servers.iteritems():
    print srid, srventry.settings.host, srventry.status.reachable

```

4.2.7 Exceptions

Exceptional errors conditions may occur because of communication problems or encoding/decoding problems or because of system errors that occur on the server while processing a request. Reporting of such error conditions is not part a regular IDL signature but part of the JSON-RPC protocol.

The Python implementation of the JSON-RPC protocol maps these error conditions to exception:

- **`raritan.rpc.HttpException`:** This exception is raised in case a communication error occurred. Typical examples include an unreachable server or a failed authentication. The exception contains a descriptive text that gives more details on the error condition.
- **`raritan.rpc.JsonRpcErrorException`:** This exception is raised in case an error happened on the server side while processing the request. The device might be in an unexpected state. Also there might be a version mismatch between client and server, what means the communication contract between client and server as specified by IDL is broken. The exception contains a descriptive text that gives more details on the error condition.
- **`raritan.rpc.JsonRpcSyntaxException`:** This exception is raised in case demarshaling of a JSON message went wrong. This may indicate a version mismatch between client and server. The exception contains a descriptive text that gives more details on the error condition.

Consider the following example for catching a communication error:

```

from raritan import rpc
from raritan.rpc import emdmodel

try:
    agent = rpc.Agent("https", "MyEMX", "admin", "raritan")
    emx = emdmodel.Emd("/model/emd", agent)
    print "Getting EMD MetaData...",
    mdata = e.getMetaData()
    print "OK ->"
    print mdata
except rpc.HttpException, e:
    print "ERROR:", str(e)

```


Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

assetmgrmodel	Asset Management Model	35
auth	Local and Remote Authentication Management	35
auth::ldapsrv	LDAP server interface	36
bulkcfg	Bulk Configuration	37
cascading	Cascading	37
cert	SSL Certificate Management	37
cew	Cisco EnergyWise	38
datetime	Device Date and Time Configuration	38
devsettings	Device Settings	38
diag	Diagnostics	39
event	Event interface	39
firmware	Firmware Management	40
fitness	Fitness Daemon	41
hmi	Human Machine Interface	42
idl	Basic IDL definitions	42
jsonrpc	Raritan JSON-RPC	42
lhx	LHX	43
lhxmodel	LHX Model	43
logging	Device Logging	43

modelpush	Model data push service	44
net	Network Configuration	44
pdumodel	PDU Model	47
peripheral	Peripheral Devices	48
portsmodel	Ports	50
production	Methods used during production	50
radius	RADIUS server interface	50
res_mon	Resource Monitor interface	51
security	Security Configuration	51
sensors	Sensors Model	53
serial	Serial Ports	53
servermon	Server Monitor	54
session	Session Management	54
smartcard	Card Reader	54
sys	Low-Level system access methods	55
test	Test Interfaces	55
um	User Management	56
usb	USB Ports	56
usermgmt	User Management	56
webcam	Webcam Management	58

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

usermgmt::Account	61
event::Engine::Action	62
webcam::StorageManager_1_0_1::Activity	63
peripheral::Address_2_0_0	63
pdumodel::Ade	64
event::AlarmManager::Alarm	65
event::AlarmManager	66
event::AlarmManager::Alert	67
lhxmodel::Lhx_3_2_1::AlertStatus	68
serial::AnalogModem	69
usermgmt::RoleManager::ArgumentDesc	70
assetmgrmodel::AssetStrip_2_0_1	71
assetmgrmodel::AssetStripConfig_1_0_1	77
assetmgrmodel::AssetStripLogger_1_0_2	81
auth::AuthManager	83
usermgmt::AuxInfo	84
test::AuxSerial	84
pdumodel::Bcm	85
bulkcfg::BulkConfiguration	87
lhxmodel::Lhx_3_2_1::Capabilities	88
smartcard::CardReader::CardInformation	88
smartcard::CardReader	89
smartcard::CardReaderManager	90
cascading::Cascading_1_0_1	91
cert::ServerSSLCert::CertInfo	94
datetime::DateTime_2_0_0::Cfg	95
webcam::Channel	95
event::Channel_1_0_1	97
pdumodel::Bcm::ChannelConfig	100
pdumodel::CircuitBreakerStatistic	101
cert::ServerSSLCert::CommonAttributes	101
lhxmodel::Config_1_0_1::ComSettings	102
event::Engine::Condition	102
lhxmodel::Config_1_0_1	103
modelpush::ModelPush::Configuration	105
devsettings::Sntp_1_0_1::Configuration	106
devsettings::Snmp_1_0_2::Configuration	107
event::Consumer	107

test::Control	108
pdumodel::Controller_3_0_0	108
pdumodel::MemoryMapController_3_0_0	192
webcam::Controls	111
pdumodel::CtrlStatistic	111
fitness::Fitness::DataEntry	112
datetime::DateTime_2_0_0	113
logging::DebugLog	115
portsmodel::Port_2_0_1::DetectionMode	116
peripheral::DeviceID_2_0_0	116
assetmgrmodel::AssetStrip_2_0_1::DeviceInfo	117
peripheral::DeviceManager_2_0_0	117
peripheral::DeviceSlot_2_0_0	122
peripheral::DeviceManager_2_0_0::DeviceTypeInfo	124
diag::DiagLogSettings	125
net::Diagnostics	127
test::Display_1_0_1	128
pdumodel::DoublePole_2_0_0	130
net::EapSettings	131
pdumodel::EDevice	132
pdumodel::Inlet_1_2_6	170
pdumodel::Outlet_1_5_6	216
pdumodel::OverCurrentProtector_2_1_2	221
pdumodel::TransferSwitch_3_1_1	331
cew::EnergyWiseManager	133
cew::EnergyWiseSettings	133
event::Engine	134
res_mon::Entry	138
fitness::Fitness::ErrorLogEntry	139
test::Ethernet	139
event::Event	141
event::Engine::EventDesc	142
logging::EventLog_1_0_1	143
event::Channel_1_0_1::EventSelect	144
hmi::ExternalBeeper_1_0_1	144
test::FeatSerial	145
firmware::Firmware_1_0_1	147
peripheral::PackageInfo_2_0_0::FirmwareInfo	150
peripheral::G2Production_2_0_0::FirmwareInfo	150
firmware::FirmwareUpdateStatus	151
fitness::Fitness	151
webcam::Format_2_0_0	153
peripheral::G2Production_2_0_0	154
serial::GsmModem_1_0_1	158
peripheral::PackageInfo_2_0_0::HardwareInfo	162
session::HistoryEntry	163
webcam::Image_2_0_0	163
firmware::ImageInfo_1_0_1	164
webcam::ImageMetaData	165
firmware::ImageStatus	165
webcam::StorageManager_1_0_1::ImageStorageMetaData	166
assetmgrmodel::AssetStripLogger_1_0_2::Info	166
usermgmt::Role::Info	166
usermgmt::RoleManager::Info	167
cert::ServerSSLCert::Info	167
test::Display_1_0_1::Info	168
serial::GsmModem_1_0_1::Information	169
webcam::Information_2_0_0	169

net::InterfaceState_2_0_0	172
hmi::InternalBeeper	173
security::IpFw_2_0_0	175
security::IpfwRule	175
net::IPv4RoutingEntry	176
net::IPv6RoutingEntry	176
event::KeyValue	177
net::LanInterfaceParameters_2_0_0	177
net::LanInterfaceSettings	178
net::LanLinkMode	178
auth::LdapManager_1_0_1	178
assetmgrmodel::AssetStripConfig_1_0_1::LEDColor	180
pdumodel::Outlet_1_5_6::LedState	180
lhxmodel::Lhx_3_2_1	181
webcam::Location	184
peripheral::DeviceSlot_2_0_0::Location	184
logging::LogEntry	185
sensors::Logger_2_1_2	185
diag::DiagLogSettings::LogLevelEntry	191
sensors::Logger_2_1_2::LogRow	191
sensors::NumericSensor_4_0_0::MetaData	193
pdumodel::Unit_2_0_1::MetaData	195
pdumodel::Outlet_1_5_6::MetaData	195
pdumodel::OverCurrentProtector_2_1_2::MetaData	196
pdumodel::Controller_3_0_0::MetaData	196
pdumodel::Pdu_3_0_0::MetaData	197
pdumodel::Inlet_1_2_6::MetaData	197
peripheral::DeviceManager_2_0_0::MetaData	198
lhxmodel::Lhx_3_2_1::MetaData	198
lhxmodel::Parameter_2_0_1::MetaData	199
lhxmodel::Sensor_4_0_0::MetaData	199
pdumodel::Ade::MetaData	200
smartcard::CardReader::MetaData	201
pdumodel::TransferSwitch_3_1_1::MetaData	201
devsettings::Modbus	202
modelpush::ModelPush	203
pdumodel::Nameplate	204
jsonrpc::NameService	204
net::Net_2_0_2	205
net::NetworkActiveValuesIPv6	209
net::NetworkConfigIP	209
net::NetworkConfigIPv4	210
net::NetworkConfigIPv6	211
datetime::DateTime_2_0_0::NtpCfg	211
lhxmodel::Sensor_4_0_0::NumThresholds	214
lhxmodel::Lhx_3_2_1::OpState	215
pdumodel::Pdu_3_0_0::OutletSequenceState	220
pdumodel::OutletStatistic	221
peripheral::PackageInfo_2_0_0	224
lhxmodel::Lhx_3_2_1::ParamCfg	225
lhxmodel::Parameter_2_0_1	225
security::PasswordSettings	228
pdumodel::Pdu_3_0_0	229
pdumodel::Bcm::PhaseConfig	236
pdumodel::Pole_2_0_0	237
auth::Policy	238
portsmodel::Port_2_0_1	238
serial::PortDispatcher_1_1_1	241

peripheral::PosElement	242
usermgmt::Preferences	243
usermgmt::Role::Privilege	243
usermgmt::RoleManager::PrivilegeDesc	243
production::Production	244
portsmodel::Port_2_0_1::Properties	245
cascading::Cascading_1_0_1::ProtocolMapping	245
assetmgrmodel::AssetStrip_2_0_1::RackUnitInfo	246
assetmgrmodel::AssetStripConfig_1_0_1::RackUnitSettings	246
auth::RadiusManager	247
event::TimerEventManager_2_0_0::Range	248
sensors::NumericSensor_4_0_0::Range	249
pdumodel::Rating	249
pdumodel::Nameplate::Rating	250
sensors::NumericSensor_4_0_0::Reading	250
lhmodel::Sensor_4_0_0::Reading	251
sensors::Logger_2_1_2::Record	251
assetmgrmodel::AssetStripLogger_1_0_2::Record	252
cert::ServerSSLCert::ReqInfo	253
res_mon::ResMon	254
security::RestrictedServiceAgreement	255
test::Result	255
usermgmt::Role	256
security::RoleAccessControl	257
security::RoleAccessRule	257
usermgmt::RoleManager::RoleAccount	258
usermgmt::RoleManager	258
test::RS232Serial	260
event::Engine::Rule	261
pdumodel::Ade::Sample	262
event::TimerEventManager_2_0_0::Schedule	262
security::Security_3_0_0	263
sensors::Sensor_4_0_0	271
lhmodel::Sensor_4_0_0	268
sensors::NumericSensor_4_0_0	212
sensors::AccumulatingNumericSensor_2_0_0	61
sensors::StateSensor_4_0_0	310
pdumodel::PowerQualitySensor_2_0_0	242
pdumodel::ResidualCurrentStateSensor_2_0_0	253
sensors::Switch_2_0_0	322
pdumodel::OverCurrentProtector_2_1_2::Sensors	274
pdumodel::Pdu_3_0_0::Sensors	275
pdumodel::Inlet_1_2_6::Sensors	275
pdumodel::TransferSwitch_3_1_1::Sensors	276
pdumodel::Outlet_1_5_6::Sensors	277
sensors::Logger_2_1_2::SensorSet	278
serial::SerialPort_2_0_0	278
servermon::ServerMonitor_2_0_0::Server	281
servermon::ServerMonitor_2_0_0	281
auth::ldapsrv::ServerSettings	284
radius::ServerSettings	285
servermon::ServerMonitor_2_0_0::ServerSettings	285
cert::ServerSSLCert	286
servermon::ServerMonitor_2_0_0::ServerStatus	288
event::Service_1_0_1	288
security::ServiceAuthorization	289
net::ServiceConfig	290
session::Session	291

session::SessionManager	291
security::Security_3_0_0::Settings	294
devsettings::Zeroconf::Settings	295
lhxmodel::Lhx_3_2_1::Settings	296
serial::GsmModem_1_0_1::Settings	296
pdumodel::Pdu_3_0_0::Settings	296
pdumodel::OverCurrentProtector_2_1_2::Settings	297
serial::AnalogModem::Settings	298
devsettings::Modbus::Settings	298
pdumodel::Outlet_1_5_6::Settings	298
peripheral::DeviceSlot_2_0_0::Settings	299
sensors::Logger_2_1_2::Settings	299
peripheral::DeviceManager_2_0_0::Settings	300
pdumodel::Unit_2_0_1::Settings	301
pdumodel::TransferSwitch_3_1_1::Settings	301
pdumodel::Inlet_1_2_6::Settings	302
serial::SerialPort_2_0_0::Settings	302
webcam::Settings_2_0_0	302
devsettings::Sntp_1_0_1	303
devsettings::Snmp_1_0_2	304
um::SnmpV3	306
usermgmt::SnmpV3Settings	307
security::SSHSettings	308
serial::SerialPort_2_0_0::State	308
pdumodel::Outlet_1_5_6::State	308
sensors::StateSensor_4_0_0::State	309
pdumodel::Pdu_3_0_0::Statistic	311
pdumodel::TransferSwitch_3_1_1::Statistics	311
peripheral::DeviceManager_2_0_0::Statistics	312
sensors::NumericSensor_4_0_0::Reading::Status	312
lhxmodel::Parameter_2_0_1::Status	313
webcam::StorageManager_1_0_1::StorageImage	313
webcam::StorageManager_1_0_1::StorageInformation	313
webcam::StorageManager_1_0_1	314
webcam::StorageManager_1_0_1::StorageMetaData	319
webcam::StorageManager_1_0_1::StorageSettings	319
assetmgrmodel::AssetStrip_2_0_1::StripInfo	320
assetmgrmodel::AssetStripConfig_1_0_1::StripSettings	321
lhx::Support	321
sys::System	323
assetmgrmodel::AssetStrip_2_0_1::TagChangeInfo	324
assetmgrmodel::AssetStrip_2_0_1::TagInfo	324
devsettings::Modbus::TcpSettings	325
devsettings::Sntp_1_0_1::TestResult	325
sensors::NumericSensor_4_0_0::ThresholdCapabilities	326
sensors::NumericSensor_4_0_0::Thresholds	326
pdumodel::ThrowPole	327
sensors::Logger_2_1_2::TimedRecord	327
event::TimerEventManager_2_0_0::TimerEvent	328
event::TimerEventManager_2_0_0	328
sensors::Sensor_4_0_0::TypeSpec	335
test::Unit_1_0_2	336
pdumodel::Unit_2_0_1	337
firmware::UpdateHistoryEntry	339
firmware::UpdateStatus	340
usb::Usb	340
usb::UsbDevice	341
usermgmt::User_1_0_1	341

usermgmt::UserCapabilities	344
usermgmt::UserInfo	345
usermgmt::UserManager_1_0_2	345
lhxmodel::Parameter_2_0_1::Value	349
peripheral::PackageInfo_2_0_0::FirmwareInfo::Version	349
pdumodel::TransferSwitch_3_1_1::WaveformSample	350
webcam::Webcam_2_0_0	350
webcam::WebcamManager_2_0_0	351
webcam::StorageManager_1_0_1::WebcamStorageInfo	354
net::WirelessInterfaceSettings	355
devsettings::Zeroconf	355
datetime::DateTime_2_0_0::ZoneCfg	356
datetime::DateTime_2_0_0::ZoneInfo	357

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

usermgmt::Account	
Account information	61
sensors::AccumulatingNumericSensor_2_0_0	
A sensor which accumulates numeric readings (e.g	61
event::Engine::Action	
An action is a tuple of 'id' (unique within the scope of this event engine), 'name' which is unique as well and used by the GUI as user readable identifier, 'isSystem' which denotes the action as system action, 'type' which defines what it is, and an argument vector that vary depending on type and which is passed to the action	62
webcam::StorageManager_1_0_1::Activity	
Activity	63
peripheral::Address_2_0_0	
Peripheral device position based address	63
pdumodel::Ade	
Interface for ADE chips directly connected to main controller	64
event::AlarmManager::Alarm	
Alarm structure	65
event::AlarmManager	
AlarmManager interface	66
event::AlarmManager::Alert	
Alert structure	67
lhxmodel::Lhx_3_2_1::AlertStatus	
LHX alert status	68
serial::AnalogModem	
Interface for communication with an analog modem attached to a serial port	69
usermgmt::RoleManager::ArgumentDesc	
Privilege Argument Description	70
assetmgrmodel::AssetStrip_2_0_1	
Asset Management Strip interface	71
assetmgrmodel::AssetStripConfig_1_0_1	
Asset Strip Config interface	77
assetmgrmodel::AssetStripLogger_1_0_2	
Asset Strip Logger interface	81
auth::AuthManager	
Authentication manager interface	83
usermgmt::AuxInfo	
Auxiliary user information	84

test::AuxSerial	
Test routines for Raritan Aux Serial interface (RS485 on pins 3 and 6 of RJ45) Require TestMode to be ON	84
pdumodel::Bcm	
Branch Circuit Monitor	85
bulkcfg::BulkConfiguration	
Bulk Configuration Interface	87
lhxmodel::Lhx_3_2_1::Capabilities	
LHX capabilities	88
smartcard::CardReader::CardInformation	
Card Information	88
smartcard::CardReader	
Card Reader Interface	89
smartcard::CardReaderManager	
Card Reader Manager Interface	90
cascading::Cascading_1_0_1	
Cascading Interface	91
cert::ServerSSLCert::CertInfo	
Certificate information	94
datetime::DateTime_2_0_0::Cfg	
Device date and time configuration	95
webcam::Channel	
The channel interface	95
event::Channel_1_0_1	
Event Channel	97
pdumodel::Bcm::ChannelConfig	
Channel Configuration	100
pdumodel::CircuitBreakerStatistic	
Overcurrent protector statistics	101
cert::ServerSSLCert::CommonAttributes	
Certificate issuer or subject attributes	101
lhxmodel::Config_1_0_1::ComSettings	
LHX port communication settings	102
event::Engine::Condition	
Condition is a logical combination of multiple events	102
lhxmodel::Config_1_0_1	
LHX Configuration Interface	103
modelpush::ModelPush::Configuration	
Model push service configuration	105
devsettings::Smtplib_1_0_1::Configuration	
SMTP server configuration	106
devsettings::Snmp_1_0_2::Configuration	
SNMP agent configuration	107
event::Consumer	
Consumer interface is for event consumers that want to be called back in case new events have occurred	107
test::Control	
Interface to enter and exit special test modes	108
pdumodel::Controller_3_0_0	
Slave controller interface	108
webcam::Controls	
Controls	111
pdumodel::CtrlStatistic	
Slave controller statistics	111
fitness::Fitness::DataEntry	
An entry in the reliability database	112
datetime::DateTime_2_0_0	
Date and time configuration methods	113

logging::DebugLog	
Device debug log interface	115
portsmodel::Port_2_0_1::DetectionMode	
Port detection mode	116
peripheral::DeviceID_2_0_0	
Peripheral device identification	116
assetmgrmodel::AssetStrip_2_0_1::DeviceInfo	
Static (type, version) information for an AssetStrip	117
peripheral::DeviceManager_2_0_0	
Peripheral Device Manager	117
peripheral::DeviceSlot_2_0_0	
Peripheral Device Slot	122
peripheral::DeviceManager_2_0_0::DeviceTypeInfo	
Peripheral device type info	124
diag::DiagLogSettings	
Diagnostic log settings	125
net::Diagnostics	
Diagnostics interface	127
test::Display_1_0_1	
Type-independent display test interface	128
pdumodel::DoublePole_2_0_0	
For OCP	130
net::EapSettings	
EAP authentication settings	131
pdumodel::EDevice	
Common base interface for any kind of electrical device that is used in the PDU model, such as inlets, OCPs and outlets	132
cew::EnergyWiseManager	
EnergyWise manager	133
cew::EnergyWiseSettings	
EnergyWise settings	133
event::Engine	
There is a single event engine instance reachable by a well known reference	134
res_mon::Entry	
ResMon Entry	138
fitness::Fitness::ErrorLogEntry	
An entry in the reliability error log	139
test::Ethernet	
Test routines for RJ45 Ethernet port This is low level interface using ethtool that does not persist any of the settings made (TODO: this interface may be combined with a 'decent' network interface	139
event::Event	
Event has a type: a STATE event indicates that a boolean state has been changed, i.e	141
event::Engine::EventDesc	
An event descriptor	142
logging::EventLog_1_0_1	
Device event log interface	143
event::Channel_1_0_1::EventSelect	
Structure to select an Event *	144
hmi::ExternalBeeper_1_0_1	
External Beeper interface	144
test::FeatSerial	
Test routines for Raritan Feature Serial interface (RS232 with some control lines and switched power) Require TestMode to be ON	145
firmware::Firmware_1_0_1	
Firmware management methods	147
peripheral::PackageInfo_2_0_0::FirmwareInfo	150
peripheral::G2Production_2_0_0::FirmwareInfo	150

firmware::FirmwareUpdateStatus	
Firmware update status interface	151
fitness::Fitness	
Fitness Daemon interface	151
webcam::Format_2_0_0	
Format	153
peripheral::G2Production_2_0_0	154
serial::GsmModem_1_0_1	
Interface for communication with a GSM modem attached to a serial port	158
peripheral::PackageInfo_2_0_0::HardwareInfo	162
session::HistoryEntry	
Session history entry	163
webcam::Image_2_0_0	
Image	163
firmware::ImageInfo_1_0_1	
Firmware image information	164
webcam::ImageMetaData	
Image meta data	165
firmware::ImageStatus	
Image upload/download status	165
webcam::StorageManager_1_0_1::ImageStorageMetaData	
StorageMetaData	166
assetmgrmodel::AssetStripLogger_1_0_2::Info	
Log information structure	166
usermgmt::Role::Info	
Role information	166
usermgmt::RoleManager::Info	
Full role manager information	167
cert::ServerSSLCert::Info	
Certificate manager information	167
test::Display_1_0_1::Info	
Collected display meta information	168
serial::GsmModem_1_0_1::Information	
Structure holding information about the modem and the SIM card	169
webcam::Information_2_0_0	
Webcam information	169
pdumodel::Inlet_1_2_6	
Inlet interface	170
net::InterfaceState_2_0_0	
LAN interface state	172
hmi::InternalBeeper	
Internal beeper interface	173
security::IpFw_2_0_0	
IP packet filter configuration	175
security::IpfwRule	
IP packet filter rule	175
net::IPv4RoutingEntry	
IPv4 Routing entry	176
net::IPv6RoutingEntry	
IPv6 Routing entry	176
event::KeyValue	
Helper that is used wherever key/value pairs are required	177
net::LanInterfaceParameters_2_0_0	
Current LAN interface parameters	177
net::LanInterfaceSettings	
LAN interface settings	178
net::LanLinkMode	
LAN interface link mode	178

auth::LdapManager_1_0_1	
LDAP server configuration interface	178
assetmgrmodel::AssetStripConfig_1_0_1::LEDColor	
The LED color in RGB format, 8 bit per channel	180
pdumodel::Outlet_1_5_6::LedState	
Outlet LED state	180
lhxmodel::Lhx_3_2_1	
LHX Interface	181
webcam::Location	
Location	184
peripheral::DeviceSlot_2_0_0::Location	
User writeable location	184
logging::LogEntry	
Device event	185
sensors::Logger_2_1_2	
Sensor logger interface	185
diag::DiagLogSettings::LogLevelEntry	
An entry containing a context name and its associated context	191
sensors::Logger_2_1_2::LogRow	
One full log row	191
pdumodel::MemoryMapController_3_0_0	
Memory map controller	192
sensors::NumericSensor_4_0_0::MetaData	
Numeric sensor metadata	193
pdumodel::Unit_2_0_1::MetaData	
Unit metadata	195
pdumodel::Outlet_1_5_6::MetaData	
Outlet metadata	195
pdumodel::OverCurrentProtector_2_1_2::MetaData	
Overcurrent protector metadata	196
pdumodel::Controller_3_0_0::MetaData	
Slave controller metadata	196
pdumodel::Pdu_3_0_0::MetaData	
PDU metadata	197
pdumodel::Inlet_1_2_6::MetaData	
Inlet metadata	197
peripheral::DeviceManager_2_0_0::MetaData	
Peripheral DeviceManager's metadata	198
lhxmodel::Lhx_3_2_1::MetaData	
LHX metadata	198
lhxmodel::Parameter_2_0_1::MetaData	
Parameter Metadata	199
lhxmodel::Sensor_4_0_0::MetaData	
Sensor's self describing data	199
pdumodel::Ade::MetaData	
ADE metadata	200
smartcard::CardReader::MetaData	
Reader Metadata	201
pdumodel::TransferSwitch_3_1_1::MetaData	
Transfer switch metadata	201
devsettings::Modbus	
Modbus service settings interface	202
modelpush::ModelPush	
Model push service settings interface	203
pdumodel::Nameplate	
Component nameplate information	204
jsonrpc::NameService	
RPC Object Name Service	204

net::Net_2_0_2	
Network configuration interface	205
net::NetworkActiveValuesIPv6	
Device IPv6 active values	209
net::NetworkConfigIP	
Device IP configuration	209
net::NetworkConfigIPv4	
Device IPv4 configuration	210
net::NetworkConfigIPv6	
Device IPv6 configuration	211
datetime::DateTime_2_0_0::NtpCfg	
NTP server configuration	211
sensors::NumericSensor_4_0_0	
A sensor with numeric readings	212
lhxmodel::Sensor_4_0_0::NumThresholds	
Numerical sensor thresholds	214
lhxmodel::Lhx_3_2_1::OpState	
LHX operational state	215
pdumodel::Outlet_1_5_6	
Outlet interface	216
pdumodel::Pdu_3_0_0::OutletSequenceState	
Outlet sequencing status	220
pdumodel::OutletStatistic	
Outlet statistics	221
pdumodel::OverCurrentProtector_2_1_2	
Overcurrent protector interface	221
peripheral::PackageInfo_2_0_0	
Peripheral device package information	224
lhxmodel::Lhx_3_2_1::ParamCfg	
Configuration parameter characteristics	225
lhxmodel::Parameter_2_0_1	
LHX Parameter Interface	225
security::PasswordSettings	
Password settings	228
pdumodel::Pdu_3_0_0	
Main PDU interface	229
pdumodel::Bcm::PhaseConfig	
Phase Configuration	236
pdumodel::Pole_2_0_0	
An inlet or outlet pole	237
auth::Policy	
Authentication policy	238
portsmodel::Port_2_0_1	
Port interface	238
serial::PortDispatcher_1_1_1	
Top-level interface of the serial port manager	241
peripheral::PosElement	
Peripheral device position element, list forms position	242
pdumodel::PowerQualitySensor_2_0_0	
Power quality sensor interface	242
usermgmt::Preferences	
User preferences	243
usermgmt::Role::Privilege	
A granted privilege	243
usermgmt::RoleManager::PrivilegeDesc	
Privilege Description	243
production::Production	
Methods used during production	244

portsmodel::Port_2_0_1::Properties	
Port properties	245
cascading::Cascading_1_0_1::ProtocolMapping	
Mapping from appl protocol id to name and transport protocol	245
assetmgrmodel::AssetStrip_2_0_1::RackUnitInfo	
Infos for a single rack unit	246
assetmgrmodel::AssetStripConfig_1_0_1::RackUnitSettings	
Settings for a single rack unit (LED state)	246
auth::RadiusManager	
RADIUS server configuration interface	247
event::TimerEventManager_2_0_0::Range	
Range structure	248
sensors::NumericSensor_4_0_0::Range	
Range of possible sensor readings	249
pdumodel::Rating	
Numerical usage ratings	249
pdumodel::Nameplate::Rating	
Component ratings	250
sensors::NumericSensor_4_0_0::Reading	
Numeric sensor reading	250
lhxmodel::Sensor_4_0_0::Reading	
Sensor reading	251
sensors::Logger_2_1_2::Record	
Sensor log record	251
assetmgrmodel::AssetStripLogger_1_0_2::Record	
Log record structure	252
cert::ServerSSLCert::ReqInfo	
Certificate signing request information	253
pdumodel::ResidualCurrentStateSensor_2_0_0	
Residual current state sensor interface	253
res_mon::ResMon	
ResMon interface	254
security::RestrictedServiceAgreement	
Restricted Service Agreement settings	255
test::Result	
Convenience structure to return test or operation results	255
usermgmt::Role	
Role management interface	256
security::RoleAccessControl	
Role-based access control settings	257
security::RoleAccessRule	
Role-based access rule	257
usermgmt::RoleManager::RoleAccount	
Role information	258
usermgmt::RoleManager	
Role manager interface	258
test::RS232Serial	
Test routines for full RS232 Serial Interface	260
event::Engine::Rule	
A Rule binds an action to a condition	261
pdumodel::Ade::Sample	
Raw sample data for a single channel	262
event::TimerEventManager_2_0_0::Schedule	
Schedule structure	262
security::Security_3_0_0	
Security configuration interface	263
lhxmodel::Sensor_4_0_0	
LHX Sensor Interface	268

sensors::Sensor_4_0_0	
Sensor interface	271
pdumodel::OverCurrentProtector_2_1_2::Sensors	
Overcurrent protector sensors	274
pdumodel::Pdu_3_0_0::Sensors	
PDU sensors	275
pdumodel::Inlet_1_2_6::Sensors	
Inlet sensors	275
pdumodel::TransferSwitch_3_1_1::Sensors	
Transfer switch sensors	276
pdumodel::Outlet_1_5_6::Sensors	
Outlet sensors	277
sensors::Logger_2_1_2::SensorSet	
Set of logged sensors	278
serial::SerialPort_2_0_0	
Interface describing a physical serial port and the devices which can be attached to it	278
servermon::ServerMonitor_2_0_0::Server	
Server Entry	281
servermon::ServerMonitor_2_0_0	
Server Monitor Interface	281
auth::ldapsrv::ServerSettings	
Server settings	284
radius::ServerSettings	
Server settings	285
servermon::ServerMonitor_2_0_0::ServerSettings	
Server Reachability Settings	285
cert::ServerSSLCert	
SSL certificate management interface	286
servermon::ServerMonitor_2_0_0::ServerStatus	
Server Reachability Status	288
event::Service_1_0_1	
Event Service	288
security::ServiceAuthorization	
Service Authorization Configuration	289
net::ServiceConfig	
Network service configuration	290
session::Session	
Session information	291
session::SessionManager	
Session manager interface	291
security::Security_3_0_0::Settings	
Security configuration This structure is deprecated and will be removed in V3.0, use concrete getters and setters instead!	294
devsettings::Zeroconf::Settings	
Zero-configservice settings	295
lhxmodel::Lhx_3_2_1::Settings	
LHX settings	296
serial::GsmModem_1_0_1::Settings	
Structure for holding settings of the GSM modem and its SIM card	296
pdumodel::Pdu_3_0_0::Settings	
PDU settings	296
pdumodel::OverCurrentProtector_2_1_2::Settings	
Overcurrent protector settings	297
serial::AnalogModem::Settings	298
devsettings::Modbus::Settings	
Modbus service settings	298
pdumodel::Outlet_1_5_6::Settings	
Outlet settings	298

peripheral::DeviceSlot_2_0_0::Settings	
User configurable slot attributes	299
sensors::Logger_2_1_2::Settings	
Sensor logger settings	299
peripheral::DeviceManager_2_0_0::Settings	
Peripheral DeviceManager's s settings	300
pdumodel::Unit_2_0_1::Settings	
Unit settings	301
pdumodel::TransferSwitch_3_1_1::Settings	
Transfer switch settings	301
pdumodel::Inlet_1_2_6::Settings	
Inlet settings	302
serial::SerialPort_2_0_0::Settings	
Port settings	302
webcam::Settings_2_0_0	
Webcam settings	302
devsettings::Sntp_1_0_1	
SMTP settings interface	303
devsettings::Snmp_1_0_2	
SNMP agent settings interface	304
um::SnmpV3	
SNMPv3 interface	306
usermgmt::SnmpV3Settings	
SNMPv3 settings	307
security::SSHSettings	
SSH authentication settings	308
serial::SerialPort_2_0_0::State	
Structure holding information about the current state of the port	308
pdumodel::Outlet_1_5_6::State	
Outlet state	308
sensors::StateSensor_4_0_0::State	
Sensor state	309
sensors::StateSensor_4_0_0	
Sensor with discrete readings	310
pdumodel::Pdu_3_0_0::Statistic	
PDU statistics	311
pdumodel::TransferSwitch_3_1_1::Statistics	
Transfer switch statistics	311
peripheral::DeviceManager_2_0_0::Statistics	
Peripheral device statistics	312
sensors::NumericSensor_4_0_0::Reading::Status	
Numeric sensor status	312
lhxmodel::Parameter_2_0_1::Status	
Parameter Status	313
webcam::StorageManager_1_0_1::StorageImage	
StorageImage	313
webcam::StorageManager_1_0_1::StorageInformation	
Information	313
webcam::StorageManager_1_0_1	
The storage manager interface	314
webcam::StorageManager_1_0_1::StorageMetaData	
StorageMetaData	319
webcam::StorageManager_1_0_1::StorageSettings	
Settings	319
assetmgrmodel::AssetStrip_2_0_1::StripInfo	
Dynamic (may change with a connected strip) information for an AssetStrip	320
assetmgrmodel::AssetStripConfig_1_0_1::StripSettings	
Settings for this Asset Strip	321

lhx::Support	
LHX Support Interface	321
sensors::Switch_2_0_0	
Switch is an actuator and an extension to StateSensor	322
sys::System	
System access methods	323
assetmgrmodel::AssetStrip_2_0_1::TagChangeInfo	
Information describing a tag change	324
assetmgrmodel::AssetStrip_2_0_1::TagInfo	
Information for a single tag	324
devsettings::Modbus::TcpSettings	
Modbus/TCP settings	325
devsettings::Smt_1_0_1::TestResult	
Result of SMTP configuration test	325
sensors::NumericSensor_4_0_0::ThresholdCapabilities	
Threshold capabilities	326
sensors::NumericSensor_4_0_0::Thresholds	
Numeric sensor thresholds	326
pdumodel::ThrowPole	
A pole that can select one of multiple inputs	327
sensors::Logger_2_1_2::TimedRecord	
Sensor log record with timestamp	327
event::TimerEventManager_2_0_0::TimerEvent	
TimerEvent structure	328
event::TimerEventManager_2_0_0	
TimerEventManager interface	328
pdumodel::TransferSwitch_3_1_1	
Transfer switch interface	331
sensors::Sensor_4_0_0::TypeSpec	
Complete sensor type specification	335
test::Unit_1_0_2	
Test interface for PDU components controlled by topofw	336
pdumodel::Unit_2_0_1	
Unit interface	337
firmware::UpdateHistoryEntry	
Firmware update history entry TODO: implement CR# 45668 on next interface change add comment field based on firmware tag "char tag[64];" to improve firmware update history entries without rootfs images	339
firmware::UpdateStatus	
Firmware update status	340
usb::Usb	
USB interface	340
usb::UsbDevice	
USB device information	341
usermgmt::User_1_0_1	
User interface	341
usermgmt::UserCapabilities	
User Capabilities Describe if certain operations can be performed for user	344
usermgmt::UserInfo	
User information	345
usermgmt::UserManager_1_0_2	
User manager interface	345
lhxmodel::Parameter_2_0_1::Value	
Parameter Value	349
peripheral::PackageInfo_2_0_0::FirmwareInfo::Version	349
pdumodel::TransferSwitch_3_1_1::WaveformSample	
Sample of voltage and current waveform	350

webcam::Webcam_2_0_0	
The webcam interface	350
webcam::WebcamManager_2_0_0	
The webcam manager interface	351
webcam::StorageManager_1_0_1::WebcamStorageInfo	
Webcam Storage Info	354
net::WirelessInterfaceSettings	
Wireless interface settings	355
devsettings::Zeroconf	
Zero-config service settings interface	355
datetime::DateTime_2_0_0::ZoneCfg	
Time zone configuration	356
datetime::DateTime_2_0_0::ZoneInfo	
Time zone information	357

Chapter 8

Namespace Documentation

8.1 assetmgrmodel Namespace Reference

Asset Management Model.

Classes

- interface [AssetStrip_2_0_1](#)
Asset Management Strip interface.
- interface [AssetStripConfig_1_0_1](#)
Asset Strip Config interface.
- interface [AssetStripLogger_1_0_2](#)
Asset Strip Logger interface.

8.1.1 Detailed Description

Asset Management Model.

8.2 auth Namespace Reference

Local and Remote Authentication Management.

Namespaces

- namespace [ldapsrv](#)
LDAP server interface.

Classes

- struct [Policy](#)
Authentication policy.
- interface [AuthManager](#)
Authentication manager interface.
- interface [LdapManager_1_0_1](#)
LDAP server configuration interface.

- interface [RadiusManager](#)

RADIUS server configuration interface.

Enumerations

- enum [Type](#) {
 [LOCAL](#), [RADIUS](#), [KERBEROS](#), [TACACS_PLUS](#),
 [LDAP](#) }

Authentication type.

8.2.1 Detailed Description

Local and Remote Authentication Management.

8.2.2 Enumeration Type Documentation

8.2.2.1 enum auth::Type

Authentication type.

Enumerator

LOCAL local authentication

RADIUS authentication via radius server

KERBEROS authentication with kerberos tickets

TACACS_PLUS authentication via TACACS+

LDAP authentication via LDAP server

8.3 auth::ldapsrv Namespace Reference

LDAP server interface.

Classes

- struct [ServerSettings](#)

Server settings.

Enumerations

- enum [ServerType](#) { [ACTIVE_DIRECTORY](#), [OPEN_LDAP](#) }

LDAP server type.

8.3.1 Detailed Description

LDAP server interface.

8.3.2 Enumeration Type Documentation

8.3.2.1 enum auth::ldapsrv::ServerType

LDAP server type.

Enumerator

ACTIVE_DIRECTORY Active directory.

OPEN_LDAP OpenLDAP.

8.4 bulkcfg Namespace Reference

Bulk Configuration.

Classes

- interface [BulkConfiguration](#)
Bulk Configuration Interface.

8.4.1 Detailed Description

Bulk Configuration.

8.5 cascading Namespace Reference

Cascading.

Classes

- interface [Cascading_1_0_1](#)
Cascading Interface.

8.5.1 Detailed Description

Cascading.

8.6 cert Namespace Reference

SSL Certificate Management.

Classes

- interface [ServerSSLCert](#)
SSL certificate management interface.

8.6.1 Detailed Description

SSL Certificate Management.

8.7 cew Namespace Reference

Cisco EnergyWise.

Classes

- interface [EnergyWiseManager](#)
EnergyWise manager.
- struct [EnergyWiseSettings](#)
EnergyWise settings.

8.7.1 Detailed Description

Cisco EnergyWise.

8.8 datetime Namespace Reference

Device Date and Time Configuration.

Classes

- interface [DateTime_2_0_0](#)
Date and time configuration methods.

8.8.1 Detailed Description

Device Date and Time Configuration.

8.9 devsettings Namespace Reference

Device Settings.

Classes

- interface [Modbus](#)
Modbus service settings interface
- interface [Sntp_1_0_1](#)
SMTP settings interface.
- interface [Snmp_1_0_2](#)
SNMP agent settings interface.
- interface [Zeroconf](#)
Zero-config service settings interface.

8.9.1 Detailed Description

Device Settings.

8.10 diag Namespace Reference

Diagnostics.

Classes

- interface [DiagLogSettings](#)
Diagnostic log settings.

8.10.1 Detailed Description

Diagnostics.

8.11 event Namespace Reference

[Event](#) interface.

Classes

- interface [AlarmManager](#)
AlarmManager interface.
- struct [KeyValue](#)
Helper that is used wherever key/value pairs are required.
- struct [Event](#)
Event has a type: a STATE event indicates that a boolean state has been changed, i.e.
- interface [Engine](#)
There is a single event engine instance reachable by a well known reference.
- interface [Consumer](#)
Consumer interface is for event consumers that want to be called back in case new events have occurred.
- interface [Channel_1_0_1](#)
Event Channel.
- interface [Service_1_0_1](#)
Event Service.
- interface [TimerEventManager_2_0_0](#)
TimerEventManager interface.

Variables

- valueobject [UserEvent](#)
This UserEvent may be used as base valueobject for all concrete events that are triggered because of user interaction.
- string [actIpAddr](#)
ip or device on which user is logged in

8.11.1 Detailed Description

[Event](#) interface.

8.11.2 Variable Documentation

8.11.2.1 valueobject event::UserEvent

This UserEvent may be used as base valueobject for all concrete events that are triggered because of user interaction.

user who triggered event

8.12 firmware Namespace Reference

Firmware Management

Classes

- struct [UpdateHistoryEntry](#)
Firmware update history entry TODO: implement CR# 45668 on next interface change add comment field based on firmware tag "char tag[64];" to improve firmware update history entries without rootfs images
- struct [ImageStatus](#)
Image upload/download status.
- struct [ImageInfo_1_0_1](#)
Firmware image information
- interface [Firmware_1_0_1](#)
Firmware management methods
- struct [UpdateStatus](#)
Firmware update status
- interface [FirmwareUpdateStatus](#)
Firmware update status interface.

Enumerations

- enum [UpdateHistoryStatus](#) { [SUCCESSFUL](#), [FAILED](#), [INCOMPLETE](#) }
Firmware update history status
- enum [ImageState](#) { [NONE](#), [UPLOADING](#), [UPLOAD_FAILED](#), [DOWNLOADING](#), [DOWNLOAD_FAILED](#), [COMPLETE](#) }
Image upload/download state.
- enum [UpdateFlags](#) { [CROSS_OEM](#), [CROSS_HW](#), [ALLOW_UNTRUSTED](#) }
Flags for startUpdate() method.

8.12.1 Detailed Description

Firmware Management

8.12.2 Enumeration Type Documentation

8.12.2.1 enum firmware::ImageState

Image upload/download state.

Enumerator

NONE No firmware image has been uploaded/downloaded.

UPLOADING A firmware image is currently being uploaded.

UPLOAD_FAILED There was a problem uploading an image to the device.

DOWNLOADING The device is downloading a firmware image from a URL.

DOWNLOAD_FAILED There was a problem downloading the image from a URL.

COMPLETE A complete image has been successfully uploaded/downloaded.

8.12.2.2 enum firmware::UpdateFlags

Flags for startUpdate() method.

Enumerator

CROSS_OEM Ignore version, product and OEM constraints.

CROSS_HW Ignore hardware constraints.

ALLOW_UNTRUSTED Allow untrusted firmwares.

8.12.2.3 enum firmware::UpdateHistoryStatus

Firmware update history status

Enumerator

SUCCESSFUL The update was successfully completed.

FAILED The update failed.

INCOMPLETE The update was not completed.

8.13 fitness Namespace Reference

Fitness Daemon

Classes

- interface [Fitness](#)

Fitness Daemon interface

8.13.1 Detailed Description

Fitness Daemon

8.14 hmi Namespace Reference

Human Machine Interface.

Classes

- interface [ExternalBeeper_1_0_1](#)
External Beeper interface.
- interface [InternalBeeper](#)
Internal beeper interface.

8.14.1 Detailed Description

Human Machine Interface.

8.15 idl Namespace Reference

Basic IDL definitions.

Variables

- valueobject [Event](#)
Common base for all events.

8.15.1 Detailed Description

Basic IDL definitions.

8.15.2 Variable Documentation

8.15.2.1 valueobject idl::Event

Initial value:

```
{  
    Object source
```

Common base for all events.

IDL object that originated the event

8.16 jsonrpc Namespace Reference

Raritan JSON-RPC.

Classes

- interface [NameService](#)
RPC Object Name Service.

8.16.1 Detailed Description

Raritan JSON-RPC.

8.17 Ihx Namespace Reference

LHX.

Classes

- interface [Support](#)
LHX [Support](#) Interface.

8.17.1 Detailed Description

LHX.

8.18 Ihxmodel Namespace Reference

LHX Model.

Classes

- interface [Lhx_3_2_1](#)
LHX Interface.
- interface [Config_1_0_1](#)
LHX Configuration Interface.
- interface [Parameter_2_0_1](#)
LHX Parameter Interface.
- interface [Sensor_4_0_0](#)
LHX Sensor Interface.

8.18.1 Detailed Description

LHX Model.

8.19 logging Namespace Reference

Device Logging.

Classes

- interface [DebugLog](#)
Device debug log interface.
- interface [EventLog_1_0_1](#)
Device event log interface.
- struct [LogEntry](#)
Device event.

Enumerations

- enum [RangeDirection](#) { [FORWARD](#), [BACKWARD](#) }
Range direction when fetching events.

Variables

- valueobject [EventLogClearedEvent](#)
Event log cleared event.

8.19.1 Detailed Description

Device Logging.

8.19.2 Enumeration Type Documentation

8.19.2.1 enum logging::RangeDirection

Range direction when fetching events.

Enumerator

- FORWARD*** Ascending serial numbers.
- BACKWARD*** Descending serial numbers.

8.20 modelpush Namespace Reference

Model data push service.

Classes

- interface [ModelPush](#)
Model push service settings interface.

8.20.1 Detailed Description

Model data push service.

8.21 net Namespace Reference

Network Configuration.

Classes

- interface [Diagnostics](#)
Diagnostics interface.
- struct [NetworkConfigIP](#)
Device IP configuration.

- struct [IPv4RoutingEntry](#)
IPv4 Routing entry.
- struct [NetworkConfigIPv4](#)
Device IPv4 configuration.
- struct [NetworkConfigIPv6](#)
Device IPv6 configuration.
- struct [IPv6RoutingEntry](#)
IPv6 Routing entry.
- struct [NetworkActiveValuesIPv6](#)
Device IPv6 active values.
- struct [ServiceConfig](#)
Network service configuration.
- struct [LanLinkMode](#)
LAN interface link mode.
- struct [InterfaceState_2_0_0](#)
LAN interface state.
- struct [LanInterfaceSettings](#)
LAN interface settings.
- struct [LanInterfaceParameters_2_0_0](#)
Current LAN interface parameters.
- struct [EapSettings](#)
EAP authentication settings.
- struct [WirelessInterfaceSettings](#)
Wireless interface settings.
- interface [Net_2_0_2](#)
Network configuration interface.

Enumerations

- enum [AutoConfigs](#) { [STATIC](#), [DHCP](#), [AUTO](#) }
Automatic network configuration protocols.
- enum [LanSpeed](#) { [LAN_SPEED_AUTO](#), [LAN_SPEED_10MBIT](#), [LAN_SPEED_100MBIT](#), [LAN_SPEED_1000MBIT](#), [LAN_SPEED_UNKNOWN](#) }
LAN interface speed.
- enum [LanDuplex](#) { [LAN_DUPLEX_AUTO](#), [LAN_DUPLEX_HALF](#), [LAN_DUPLEX_FULL](#), [LAN_DUPLEX_UNKNOWN](#) }
LAN interface duplex mode.
- enum [InterfaceMode_2_0_0](#) { [IF_MODE_WIRED](#), [IF_MODE_WIRELESS](#), [IF_MODE_USB_DEVICE](#) }
LAN interface mode.
- enum [AuthenticationMode](#) { [AUTH_NONE](#), [AUTH_PSK](#), [AUTH_EAP](#) }
WLAN authentication mode.
- enum [EapOuterMethod](#) { [EAP_PEAP](#) }
EAP outer authentication method.
- enum [EapInnerMethod](#) { [EAP_MSCHAPv2](#) }
EAP inner authentication method.

8.21.1 Detailed Description

Network Configuration.

8.21.2 Enumeration Type Documentation

8.21.2.1 enum net::AuthenticationMode

WLAN authentication mode.

Enumerator

AUTH_NONE No authentication.

AUTH_PSK Pre-shared key authentication.

AUTH_EAP EAP authentication.

8.21.2.2 enum net::AutoConfigs

Automatic network configuration protocols.

Enumerator

STATIC No automatic configuration.

DHCP Use DHCP for automatic configuration (used for IPv4)

AUTO Use automatic configuration (used for IPv6)

8.21.2.3 enum net::EapInnerMethod

EAP inner authentication method.

Enumerator

EAP_MSCHAPv2 MSCHAPv2 authentication.

8.21.2.4 enum net::EapOuterMethod

EAP outer authentication method.

Enumerator

EAP_PEAP PEAP authentication.

8.21.2.5 enum net::InterfaceMode_2_0_0

LAN interface mode.

Enumerator

IF_MODE_WIRED Use wired network interface.

IF_MODE_WIRELESS Use wireless network interface.

IF_MODE_USB_DEVICE Use ethernet gadget on USB device port.

8.21.2.6 enum net::LanDuplex

LAN interface duplex mode.

Enumerator

- LAN_DUPLEX_AUTO** Use auto-negotiation to set duplex mode.
- LAN_DUPLEX_HALF** Half duplex.
- LAN_DUPLEX_FULL** Full duplex.
- LAN_DUPLEX_UNKNOWN** Unknown duplex mode.

8.21.2.7 enum net::LanSpeed

LAN interface speed.

Enumerator

- LAN_SPEED_AUTO** Use auto-negotiation to set speed.
- LAN_SPEED_10MBIT** 10 MBit/s
- LAN_SPEED_100MBIT** 100 MBit/s
- LAN_SPEED_1000MBIT** 1 GBit/s
- LAN_SPEED_UNKNOWN** Unknown speed.

8.22 pdumodel Namespace Reference

PDU Model.

Classes

- interface [Ade](#)
Interface for ADE chips directly connected to main controller.
- interface [Bcm](#)
Branch Circuit Monitor.
- struct [CtrlStatistic](#)
Slave controller statistics.
- interface [Controller_3_0_0](#)
Slave controller interface.
- interface [EDevice](#)
Common base interface for any kind of electrical device that is used in the PDU model, such as inlets, OCPs and outlets.
- interface [Inlet_1_2_6](#)
Inlet interface
- interface [MemoryMapController_3_0_0](#)
Memory map controller.
- struct [Rating](#)
Numerical usage ratings.
- struct [Nameplate](#)
Component nameplate information.
- struct [OutletStatistic](#)
Outlet statistics

- interface [Outlet_1_5_6](#)
Outlet interface
- struct [CircuitBreakerStatistic](#)
Overcurrent protector statistics.
- interface [OverCurrentProtector_2_1_2](#)
Overcurrent protector interface.
- interface [Pdu_3_0_0](#)
Main PDU interface.
- struct [Pole_2_0_0](#)
An inlet or outlet pole.
- struct [DoublePole_2_0_0](#)
for OCP
- struct [ThrowPole](#)
A pole that can select one of multiple inputs.
- interface [PowerQualitySensor_2_0_0](#)
Power quality sensor interface.
- interface [ResidualCurrentStateSensor_2_0_0](#)
Residual current state sensor interface.
- interface [TransferSwitch_3_1_1](#)
Transfer switch interface.
- interface [Unit_2_0_1](#)
Unit interface.

Enumerations

- enum [PowerLine](#) { [L1](#), [L2](#), [L3](#), [NEUTRAL](#) }
Power line.

8.22.1 Detailed Description

PDU Model.

8.22.2 Enumeration Type Documentation

8.22.2.1 enum pdumodel::PowerLine

Power line.

Enumerator

- L1** Line 1.
- L2** Line 2.
- L3** Line 3.
- NEUTRAL** Neutral.

8.23 peripheral Namespace Reference

Peripheral Devices.

Classes

- interface [DeviceManager_2_0_0](#)
Peripheral Device Manager.
- struct [PackageInfo_2_0_0](#)
Peripheral device package information.
- struct [PosElement](#)
peripheral device position element, list forms position
- struct [DeviceID_2_0_0](#)
peripheral device identification
- struct [Address_2_0_0](#)
peripheral device position based address
- interface [DeviceSlot_2_0_0](#)
Peripheral Device Slot.
- interface [G2Production_2_0_0](#)

Enumerations

- enum [PortType](#) { [ONEWIRE_ONBOARD](#), [ONEWIRE_DEV_PORT](#), [ONEWIRE_HUB_PORT](#), [ONEWIRE_CHAIN_POS](#) }
peripheral device port types

Variables

- valueobject [Device_2_0_0](#)
A peripheral device is the collection of.
- vector< [PosElement](#) > [position](#)
Position within 1-wire topo.
- string [packageClass](#)
physical package identifier
- sensors Sensor_4_0_0 [device](#)
device reference

8.23.1 Detailed Description

Peripheral Devices.

8.23.2 Enumeration Type Documentation

8.23.2.1 enum peripheral::PortType

peripheral device port types

Enumerator

- [ONEWIRE_ONBOARD](#)** a built in, inaccessible port
- [ONEWIRE_DEV_PORT](#)** a 1-wire port on the device
- [ONEWIRE_HUB_PORT](#)** a port on a Hub
- [ONEWIRE_CHAIN_POS](#)** a chain position

8.23.3 Variable Documentation

8.23.3.1 valueobject peripheral::Device_2_0_0

Initial value:

```
{  
    DeviceID_2_0_0    deviceID
```

A peripheral device is the collection of.

- device identification
- device position
- a flag indicating actuator type
- device referencedevice identification

8.24 portsmodel Namespace Reference

Ports.

Classes

- interface [Port_2_0_1](#)
Port interface.

8.24.1 Detailed Description

Ports.

8.25 production Namespace Reference

Methods used during production.

Classes

- interface [Production](#)
Methods used during production.

8.25.1 Detailed Description

Methods used during production.

8.26 radius Namespace Reference

RADIUS server interface.

Classes

- struct [ServerSettings](#)

Server settings.

Enumerations

- enum [AuthType](#) { [PAP](#), [CHAP](#) }

RADIUS auth type.

8.26.1 Detailed Description

RADIUS server interface.

8.26.2 Enumeration Type Documentation

8.26.2.1 enum radius::AuthType

RADIUS auth type.

Enumerator

PAP PAP.

CHAP CHAP.

8.27 res_mon Namespace Reference

Resource Monitor interface.

Classes

- struct [Entry](#)

ResMon Entry.

- interface [ResMon](#)

ResMon interface.

8.27.1 Detailed Description

Resource Monitor interface.

8.28 security Namespace Reference

Security Configuration

Classes

- struct [IpfwRule](#)
IP packet filter rule.
- struct [IpFw_2_0_0](#)
IP packet filter configuration.
- struct [RoleAccessRule](#)
Role-based access rule.
- struct [RoleAccessControl](#)
Role-based access control settings.
- struct [PasswordSettings](#)
Password settings.
- struct [SSHSettings](#)
SSH authentication settings.
- struct [RestrictedServiceAgreement](#)
Restricted Service Agreement settings.
- interface [Security_3_0_0](#)
Security configuration interface
- interface [ServiceAuthorization](#)
Service Authorization Configuration.

Enumerations

- enum [IpfwPolicy](#) { [ACCEPT](#), [DROP](#), [REJECT](#) }
IP packet filter policy.
- enum [RoleAccessPolicy](#) { [ALLOW](#), [DENY](#) }
Role-based access policy.

Variables

- valueobject [PasswordSettingsChanged](#)
This Event is emitted after any of the password-settings has been changed.
- [PasswordSettings](#) **newSettings**

8.28.1 Detailed Description

Security Configuration

8.28.2 Enumeration Type Documentation

8.28.2.1 enum security::IpfwPolicy

IP packet filter policy.

Enumerator

- ACCEPT** Accept the packet.
- DROP** Silently discard the packet.
- REJECT** Discard packet, send error response.

8.28.2.2 enum security::RoleAccessPolicy

Role-based access policy.

Enumerator

ALLOW Access granted.

DENY Access denied.

8.29 sensors Namespace Reference

Sensors Model.

Classes

- interface [AccumulatingNumericSensor_2_0_0](#)
A sensor which accumulates numeric readings (e.g.
- interface [NumericSensor_4_0_0](#)
A sensor with numeric readings.
- interface [Sensor_4_0_0](#)
Sensor interface
- interface [Logger_2_1_2](#)
Sensor logger interface.
- interface [StateSensor_4_0_0](#)
Sensor with discrete readings.
- interface [Switch_2_0_0](#)
Switch is an actuator and an extension to StateSensor.

8.29.1 Detailed Description

Sensors Model.

8.30 serial Namespace Reference

Serial Ports.

Classes

- interface [AnalogModem](#)
Interface for communication with an analog modem attached to a serial port.
- interface [GsmModem_1_0_1](#)
Interface for communication with a GSM modem attached to a serial port.
- interface [PortDispatcher_1_1_1](#)
Top-level interface of the serial port manager.
- interface [SerialPort_2_0_0](#)
Interface describing a physical serial port and the devices which can be attached to it.

8.30.1 Detailed Description

Serial Ports.

8.31 servermon Namespace Reference

Server Monitor.

Classes

- interface [ServerMonitor_2_0_0](#)
Server Monitor Interface.

8.31.1 Detailed Description

Server Monitor.

8.32 session Namespace Reference

Session Management

Classes

- struct [Session](#)
Session information
- struct [HistoryEntry](#)
Session history entry
- interface [SessionManager](#)
Session manager interface

8.32.1 Detailed Description

Session Management

8.33 smartcard Namespace Reference

Card Reader.

Classes

- interface [CardReader](#)
Card Reader Interface.
- interface [CardReaderManager](#)
Card Reader Manager Interface.

8.33.1 Detailed Description

Card Reader. Card Reader Manager.

8.34 sys Namespace Reference

Low-Level system access methods.

Classes

- interface [System](#)
System access methods.

8.34.1 Detailed Description

Low-Level system access methods.

8.35 test Namespace Reference

Test Interfaces.

Classes

- interface [Display_1_0_1](#)
Type-independent display test interface.
- struct [Result](#)
Convenience structure to return test or operation results.
- interface [Control](#)
Interface to enter and exit special test modes.
- interface [RS232Serial](#)
Test routines for full RS232 Serial Interface.
- interface [FeatSerial](#)
test routines for Raritan Feature Serial interface (RS232 with some control lines and switched power) Require Test-Mode to be ON.
- interface [AuxSerial](#)
test routines for Raritan Aux Serial interface (RS485 on pins 3 and 6 of RJ45) Require TestMode to be ON.
- interface [Ethernet](#)
test routines for RJ45 Ethernet port This is low level interface using ethtool that does not persist any of the settings made (TODO: this interface may be combined with a 'decent' network interface.
- interface [Unit_1_0_2](#)
Test interface for PDU components controlled by topofw.

8.35.1 Detailed Description

Test Interfaces. system test interfaces

8.36 um Namespace Reference

User Management.

Classes

- interface [SnmpV3](#)
SNMPv3 interface.

8.36.1 Detailed Description

User Management.

8.37 usb Namespace Reference

USB Ports.

Classes

- struct [UsbDevice](#)
USB device information.
- interface [Usb](#)
USB interface.

8.37.1 Detailed Description

USB Ports.

8.38 usermgmt Namespace Reference

User Management

Classes

- interface [Role](#)
Role management interface
- interface [RoleManager](#)
Role manager interface.
- struct [SnmpV3Settings](#)
SNMPv3 settings.
- struct [AuxInfo](#)
Auxiliary user information.
- struct [Preferences](#)
User preferences
- struct [UserInfo](#)
User information
- struct [UserCapabilities](#)

User Capabilities Describe if certain operations can be performed for user.

- interface [User_1_0_1](#)
User interface
- struct [Account](#)
Account information
- interface [UserManager_1_0_2](#)
User manager interface

Enumerations

- enum [TemperatureEnum](#) { [DEG_C](#), [DEG_F](#) }
Preferred display unit for temperature sensors.
- enum [LengthEnum](#) { [METER](#), [FEET](#) }
Preferred display unit for length measurements, e.g.
- enum [PressureEnum](#) { [PASCAL](#), [PSI](#) }
Preferred display unit for (air) pressure sensors.

Variables

- valueobject [RoleEvent](#)
Base type of all account event.
- valueobject **RoleAdded**
- valueobject **RoleRemoved**
- valueobject **RoleChanged**
- [Role](#) Info **newSettings**
- valueobject [AccountEvent](#)
Base type of all account event.
- valueobject [AccountAdded](#)
This event is emitted after a new account with the provided username was added.
- valueobject [AccountRemoved](#)
This event is emitted after the account with the provided username has been removed.
- valueobject [PasswordChanged](#)
This event is emitted after the password for an account was changed.
- valueobject [AccountChanged](#)
This event is emitted if the settings of an account as defined in [usermgmt.UserInfo](#) have changed (Note: we may add an indication what in the structure has changed or even split the event, if handling is difficult)

8.38.1 Detailed Description

User Management

8.38.2 Enumeration Type Documentation

8.38.2.1 enum usermgmt::LengthEnum

Preferred display unit for length measurements, e.g.

device altitude

Enumerator

METER Meters.

FEET Feet.

8.38.2.2 enum usermgmt::PressureEnum

Preferred display unit for (air) pressure sensors.

Enumerator

PASCAL Pascal.

PSI pound-force per square inch

8.38.2.3 enum usermgmt::TemperatureEnum

Preferred display unit for temperature sensors.

Enumerator

DEG_C Degrees Celsius.

DEG_F Degrees Fahrenheit.

8.38.3 Variable Documentation

8.38.3.1 valueobject usermgmt::AccountEvent

Base type of all account event.

id of user which was affected

8.39 webcam Namespace Reference

Webcam Management.

Classes

- interface [StorageManager_1_0_1](#)
The storage manager interface.
- struct [Format_2_0_0](#)
Format.
- struct [Controls](#)
Controls.
- struct [Location](#)
Location.
- struct [ImageMetaData](#)
Image meta data.
- struct [Image_2_0_0](#)
Image.
- struct [Settings_2_0_0](#)
Webcam settings.
- struct [Information_2_0_0](#)
Webcam information.
- interface [Webcam_2_0_0](#)
The webcam interface.
- interface [Channel](#)
The channel interface.
- interface [WebcamManager_2_0_0](#)
The webcam manager interface.

Enumerations

- enum [PixelFormat](#) { MJPEG, JPEG, RGB, YUV }
PixelFormat.
- enum [PowerLineFrequency](#) { NOT_SUPPORTED, HZ50, HZ60, DISABLED }
PowerLineFrequency.
- enum [Priority](#) {
VERY_LOW, LOW, NORMAL, HIGH,
VERY_HIGH }
Priority.

Variables

- valueobject [WebcamEvent](#)
Base type of all webcam event.
- valueobject [WebcamAddedEvent](#)
This event is emitted after a webcam was added.
- valueobject [WebcamRemovedEvent](#)
This event is emitted after a webcam has been removed.
- valueobject [WebcamSettingsChangedEvent](#)
This event is emitted after the settings of a webcam were changed.
- webcam [Settings_2_0_0 oldSettings](#)
the old settings
- webcam [Settings_2_0_0 newSettings](#)
the new settings

8.39.1 Detailed Description

Webcam Management.

8.39.2 Enumeration Type Documentation

8.39.2.1 enum webcam::PixelFormat

PixelFormat.

Enumerator

MJPEG Motion JPEG.
JPEG JPEG.
RGB RGB encoded.
YUV YUV encoded.

8.39.2.2 enum webcam::Priority

Priority.

Enumerator

VERY_LOW very low
LOW low
NORMAL normal
HIGH high
VERY_HIGH very high

8.39.3 Variable Documentation

8.39.3.1 valueobject webcam::WebcamEvent

Base type of all webcam event.

the webcam which was affected

8.39.3.2 valueobject webcam::WebcamSettingsChangedEvent

This event is emitted after the settings of a webcam were changed.

the user that caused the change

Chapter 9

Class Documentation

9.1 usermgmt::Account Struct Reference

Account information

```
import "UserManager.idl";
```

Public Attributes

- string [name](#)
Account name
- [UserInfo](#) [info](#)
User information

9.1.1 Detailed Description

Account information

The documentation for this struct was generated from the following file:

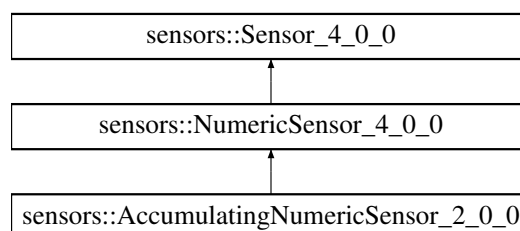
- pdu-json-rpc-api/idl/UserManager.idl

9.2 sensors::AccumulatingNumericSensor_2_0_0 Interface Reference

A sensor which accumulates numeric readings (e.g.

```
import "AccumulatingNumericSensor.idl";
```

Inheritance diagram for sensors::AccumulatingNumericSensor_2_0_0:



Public Member Functions

- void `resetValue` ()
Resets the accumulated value of the sensor.

Public Attributes

- valueobject `ResetEvent`: `event.UserEvent` { `NumericSensor_4_0_0.Reading` oldReading
Event: Accumulated value has been reset.
- `NumericSensor_4_0_0` Reading newReading
Value after reset.

Additional Inherited Members

9.2.1 Detailed Description

A sensor which accumulates numeric readings (e.g. energy counter)

9.2.2 Member Data Documentation

9.2.2.1 valueobject sensors::AccumulatingNumericSensor_2_0_0::ResetEvent

Event: Accumulated value has been reset.

Value before reset

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/AccumulatingNumericSensor.idl

9.3 event::Engine::Action Struct Reference

An action is a tuple of 'id' (unique within the scope of this event engine), 'name' which is unique as well and used by the GUI as user readable identifier, 'isSystem' which denotes the action as system action, 'type' which defines what it is, and an argument vector that vary depending on type and which is passed to the action.

```
import "EventEngine.idl";
```

Public Attributes

- string `id`
Action ID.
- string `name`
User-defined name.
- boolean `isSystem`
true for system-defined actions
- string `type`
Action type.
- vector< `KeyValue` > `arguments`
Action argument map.

9.3.1 Detailed Description

An action is a tuple of 'id' (unique within the scope of this event engine), 'name' which is unique as well and used by the GUI as user readable identifier, 'isSystem' which denotes the action as system action, 'type' which defines what it is, and an argument vector that vary depending on type and which is passed to the action.

The 'isSystem' flag is readonly and if set marks the action as not deletable.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EventEngine.idl

9.4 webcam::StorageManager_1_0_1::Activity Struct Reference

[Activity.](#)

```
import "StorageManager.idl";
```

Public Attributes

- [Webcam_2_0_0 webcam](#)
webcam object
- int [interval](#)
capture interval
- int [count](#)
nr of images to take
- int [done](#)
nr of images taken

9.4.1 Detailed Description

[Activity.](#)

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.5 peripheral::Address_2_0_0 Struct Reference

peripheral device position based address

```
import "PeripheralDeviceSlot.idl";
```

Public Attributes

- vector< [PosElement](#) > [position](#)
Position within 1-wire topo.
- sensors Sensor_4_0_0 TypeSpec [type](#)
device's type spec
- boolean [isActuator](#)
true if device is an actuator
- int [channel](#)
Channel number.

9.5.1 Detailed Description

peripheral device position based address

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceSlot.idl

9.6 pdumodel::Ade Interface Reference

Interface for ADE chips directly connected to main controller.

```
import "Ade.idl";
```

Classes

- struct [MetaData](#)
ADE metadata.
- struct [Sample](#)
Raw sample data for a single channel.

Public Types

- typedef map< string, long > [RegisterMap](#)
Map of ADE register values.

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the ADE metadata.
- vector< [Sample](#) > [getLatestSample](#) ()
Retrieve the latest raw samples.
- [RegisterMap](#) [getCalibrationData](#) ()
Retrieve the values of all supported calibration registers.
- int [setCalibrationData](#) (in [RegisterMap](#) regs)
Set new values for some or all calibration registers.

9.6.1 Detailed Description

Interface for ADE chips directly connected to main controller.

9.6.2 Member Function Documentation

9.6.2.1 RegisterMap pdumodel::Ade::getCalibrationData ()

Retrieve the values of all supported calibration registers.

Returns

Map of calibration register values

9.6.2.2 `vector<Sample> pdumodel::Ade::getLatestSample ( )`

Retrieve the latest raw samples.

Returns

Vector of samples, one for each channel

9.6.2.3 `MetaData pdumodel::Ade::getMetaData ( )`

Retrieve the ADE metadata.

Returns

ADE metadata

9.6.2.4 `int pdumodel::Ade::setCalibrationData ( in RegisterMap regs )`

Set new values for some or all calibration registers.

Note

This command is only available during manufacturing!

Parameters

<i>regs</i>	Map of new calibration register values
-------------	--

Returns

- 0 if OK
- 1 if any parameters are invalid
- 2 if the device is not in factory configuration mode

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Ade.idl

9.7 event::AlarmManager::Alarm Struct Reference

[Alarm](#) structure.

```
import "AlarmManager.idl";
```

Public Attributes

- string [id](#)
Alarm id.
- string [name](#)
Alarm name.
- string [actionId](#)
Corresponding action id.
- vector< [Alert](#) > [alerts](#)
List of alerts.

9.7.1 Detailed Description

[Alarm](#) structure.

An alarm has a name, a reference to its action source and a list of all alerts which created the alarm.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AlarmManager.idl

9.8 event::AlarmManager Interface Reference

[AlarmManager](#) interface.

```
import "AlarmManager.idl";
```

Classes

- struct [Alarm](#)
Alarm structure.
- struct [Alert](#)
Alert structure.

Public Member Functions

- int [acknowledgeAlarm](#) (in string alarmId)
Acknowledges an alarm.
- vector< [Alarm](#) > [listAlarms](#) ()
List alarms that need to be acknowledged.

Public Attributes

- constant int [NO_ERROR](#) = 0
Error codes.
- constant int [ERR_UNKNOWN_ALARM_ID](#) = 1
unknown alarmId
- constant int [ERR_EXECUTING_ACTIONS](#) = 2
failure during executing actions
- valueobject [AlarmAddedEvent](#): idl.Event { [Alarm](#) alarm
New alarm added event.
- valueobject [AlarmUpdatedEvent](#): idl.Event { [Alarm](#) alarm
Alarm updated event.
- valueobject [AlarmAcknowledgedEvent](#): idl.Event { string alarmId
Existing alarm acknowledgement event.

9.8.1 Detailed Description

[AlarmManager](#) interface.

9.8.2 Member Function Documentation

9.8.2.1 int event::AlarmManager::acknowledgeAlarm (in string *alarmId*)

Acknowledges an alarm.

This stops notification sending and will remove the specified alarm from the alarm list.

Parameters

<i>alarmId</i>	alarm id
----------------	----------

Returns

NO_ERROR if OK

ERR_UNKNOWN_ALARM_ID if alarmId is unknown

ERR_EXECUTING_ACTIONS if failure during executing acknowledgment actions

9.8.3 Member Data Documentation

9.8.3.1 valueobject event::AlarmManager::AlarmAcknowledgedEvent

Existing alarm acknowledgement event.

[Alarm](#) id of acknowledged alarm

9.8.3.2 valueobject event::AlarmManager::AlarmAddedEvent

New alarm added event.

Newly added alarm

9.8.3.3 valueobject event::AlarmManager::AlarmUpdatedEvent

[Alarm](#) updated event.

Updated alarm

9.8.3.4 constant int event::AlarmManager::NO_ERROR = 0

Error codes.

operation successful, no error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/AlarmManager.idl

9.9 event::AlarmManager::Alert Struct Reference

[Alert](#) structure.

```
import "AlarmManager.idl";
```

Public Attributes

- string [eventCondition](#)
Event condition.
- string [message](#)
Log message.
- time [firstAppearance](#)
Date & time of first appearance.
- time [lastAppearance](#)
Date & time of last appearance.
- int [numberAlerts](#)
Number of alerts.

9.9.1 Detailed Description

[Alert](#) structure.

An alert contains the event id, the log message of the triggered alarm condition plus time and counter fields to express when and how often the alarm condition was triggered.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AlarmManager.idl

9.10 Ihxmodel::Lhx_3_2_1::AlertStatus Struct Reference

LHX alert status.

```
import "Lhx.idl";
```

Public Attributes

- vector< boolean > [sensorFailure](#)
Sensor failure (broken or short circuit)
- vector< boolean > [fanFailure](#)
Fan motor failure.
- vector< boolean > [powerSupplyFailure](#)
Power supply failure.
- boolean [thresholdAirOutlet](#)
The air outlet temperature threshold was crossed.
- boolean [thresholdAirInlet](#)
The air inlet temperature threshold was crossed.
- boolean [thresholdWaterInlet](#)
The water inlet temperature threshold was crossed.
- boolean [doorOpened](#)
The door was opened.
- boolean [maximumCoolingRequest](#)
Maximum cooling was requested.
- boolean [emergencyCooling](#)
LHX is in emergency cooling mode.
- boolean [waterLeak](#)
Water leakage was detected.

- boolean [thresholdHumidity](#)
The humidity threshold was crossed.
- boolean [externalWaterCoolingFailure](#)
An external water cooling failure occurred.
- boolean [thresholdWaterOutlet](#)
The water outlet temperature threshold was crossed.
- boolean [stBusError](#)
ST-Bus communication error.
- boolean [condenserPumpFailure](#)
Condenser pump failure occurred.
- boolean [baseElectronicsFailure](#)
Base electronics failure occurred.
- boolean [voltageLow](#)
The battery voltage is low.

9.10.1 Detailed Description

LHX alert status.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Lhx.idl

9.11 serial::AnalogModem Interface Reference

Interface for communication with an analog modem attached to a serial port.

```
import "AnalogModem.idl";
```

Classes

- struct [Settings](#)

Public Member Functions

- [Settings](#) [getSettings](#) ()
Get modem settings.
- int [setSettings](#) (in [Settings](#) settings)
Set modem settings.

Public Attributes

- constant int [SUCCESS](#) = 0
Error codes.
- constant int [ERR_INVALID_VALUE](#) = 1
Invalid argument.
- valueobject [DialInEvent](#): [idl.Event](#) { string number
Dial-in event.
- valueobject [CallReceivedEvent](#): [DialInEvent](#) { }
An event that's emitted when a dial-in call was answered.
- valueobject [CallEndedEvent](#): [DialInEvent](#) { boolean disconnectedRemotely
An event that's emitted when a dial-in call was hung up.

9.11.1 Detailed Description

Interface for communication with an analog modem attached to a serial port.

9.11.2 Member Function Documentation

9.11.2.1 Settings serial::AnalogModem::getSettings ()

Get modem settings.

Returns

– Current modem settings

9.11.2.2 int serial::AnalogModem::setSettings (in Settings settings)

Set modem settings.

Parameters

<i>settings</i>	– New settings
-----------------	----------------

Returns

SUCCESS – on success

ERR_INVALID_VALUE – if any passed value was invalid

9.11.3 Member Data Documentation

9.11.3.1 valueobject serial::AnalogModem::CallEndedEvent

An event that's emitted when a dial-in call was hung up.

If true, the call was disconnected by the caller If false, the call was disconnected locally, e.g. due to a connection failure

9.11.3.2 valueobject serial::AnalogModem::DialInEvent

Dial-in event.

The caller's phone number

9.11.3.3 constant int serial::AnalogModem::SUCCESS = 0

Error codes.

No error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/AnalogModem.idl

9.12 usermgmt::RoleManager::ArgumentDesc Struct Reference

Privilege Argument Description.

```
import "RoleManager.idl";
```

Public Attributes

- string [name](#)
Argument name.
- string [desc](#)
Argument description.

9.12.1 Detailed Description

Privilege Argument Description.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/RoleManager.idl

9.13 assetmgrmodel::AssetStrip_2_0_1 Interface Reference

Asset Management Strip interface.

```
import "AssetStrip.idl";
```

Classes

- struct [DeviceInfo](#)
Static (type, version) information for an AssetStrip.
- struct [RackUnitInfo](#)
Infos for a single rack unit.
- struct [StripInfo](#)
Dynamic (may change with a connected strip) information for an AssetStrip.
- struct [TagChangeInfo](#)
Information describing a tag change.
- struct [TagInfo](#)
Information for a single tag.

Public Types

- enum [State](#) { [DISCONNECTED](#), [FIRMWARE_UPDATE](#), [UNSUPPORTED](#), [AVAILABLE](#) }
AssetStrip state
- enum [StripType](#) { [SIMPLE](#), [COMPOSITE](#) }
Type of the connected asset strip.
- enum [TagType](#) { [SINGLE](#), [EXTENSION](#), [NONE](#) }
Type of an asset tag connected to a rack unit.
- enum [CascadeState](#) { [CASCADE_ACTIVE](#), [CASCADE_FIRMWARE_UPDATE](#) }
For composite asset strips CascadeState shows additional information about the state of the complete cascade.
- enum [FirmwareUpdateState](#) { [UPDATE_STARTED](#), [UPDATE_SUCCESSFUL](#), [UPDATE_FAILED](#) }
Enumeration: State of firmware update.

Public Member Functions

- [State](#) [getState](#) ()
Get the current state of the AssetStrip.
- [DeviceInfo](#) [getDeviceInfo](#) ()
Get static (hardware and firmware) information.
- [StripInfo](#) [getStripInfo](#) ()
Get dynamic (number of tags) information.
- int [getRackUnitInfo](#) (in int rackUnitNumber, out [RackUnitInfo](#) info)
Get info with all settings of a rack unit at once.
- vector< [RackUnitInfo](#) > [getAllRackUnitInfos](#) ()
Get infos with settings for all rack units.
- int [getTag](#) (in int rackUnitNumber, in int slotNumber, out [TagInfo](#) tagInfo)
Get the asset tag for a rack unit.
- vector< [TagInfo](#) > [getAllTags](#) ()
Get all asset tags of the strip.
- vector< [TagInfo](#) > [getMainTags](#) ()
Get all asset tags on the main strip.
- int [getExtensionTags](#) (in int rackUnitNumber, out vector< [TagInfo](#) > tags)
Get all asset tags on an extension.
- void [triggerPowercycle](#) (in boolean hard)
Trigger a powercycle of either the whole asset strip port or the LED part power supply on the asset strip.

Public Attributes

- constant int [NO_ERROR](#) = 0
Error codes.
- constant int [ERR_INVALID_PARAM](#) = 1
Invalid parameter for an operation.
- constant int [ERR_NO_SUCH_OBJECT](#) = 2
Requested object does not exist.
- constant int [MAIN_STRIP_COLUMN](#) = 0
Constants.
- valueobject [StripInfoChangedEvent](#): idl.Event { [StripInfo](#) oldInfo
Event: Asset strip dynamic information has changed.
- [StripInfo](#) [newInfo](#)
Information after change.
- valueobject [StateChangedEvent](#): idl.Event { [State](#) oldState
Event: Asset strip state has changed.
- [State](#) [newState](#)
State after change.
- [DeviceInfo](#) [deviceInfo](#)

Information about connected strip, only valid if newState is AVAILABLE
- valueobject [RackUnitChangedEvent](#): event.UserEvent { int rackUnitNumber
Event: A rack unit has changed.
- [RackUnitInfo](#) [rackUnit](#)
New rack unit information.
- valueobject [TagEvent](#): idl.Event { vector<[TagChangeInfo](#)> tags
Event: A tag was added or removed.
- vector< [TagInfo](#) > [allTags](#)

- New list of detected tags after change.*
- valueobject [TagAddedEvent](#): [TagEvent](#) { }
Event: A tag was added.
- valueobject [TagRemovedEvent](#): [TagEvent](#) { }
Event: A tag was removed.
- valueobject [FirmwareUpdateStateChangedEvent](#): [idl.Event](#) { [FirmwareUpdateState](#) state
Event: Firmware update was started or has finished.
- valueobject [BladeOverflowChangedEvent](#): [idl.Event](#) { boolean overflow
Event: Tag overflow.
- valueobject [OrientationChangedEvent](#): [idl.Event](#) { [AssetStripConfig_1_0_1.Orientation](#) oldOrientation
Event: Detected strip orientation has changed.
- [AssetStripConfig_1_0_1](#) Orientation [newOrientation](#)
Strip orientation after change.
- valueobject [CompositionChangedEvent](#): [idl.Event](#) { int oldComponentCount
Event: Strip composition has changed.
- int [newComponentCount](#)
Component count after change.

9.13.1 Detailed Description

Asset Management Strip interface.

9.13.2 Member Enumeration Documentation

9.13.2.1 enum assetmgrmodel::AssetStrip_2_0_1::CascadeState

For composite asset strips CascadeState shows additional information about the state of the complete cascade.

Enumerator

- CASCADE_ACTIVE** cascade is up and running
- CASCADE_FIRMWARE_UPDATE** a device in the cascade receives a firmware update

9.13.2.2 enum assetmgrmodel::AssetStrip_2_0_1::FirmwareUpdateState

Enumeration: State of firmware update.

Enumerator

- UPDATE_STARTED** Update is running.
- UPDATE_SUCCESSFUL** Update was completed successfully.
- UPDATE_FAILED** Update has failed.

9.13.2.3 enum assetmgrmodel::AssetStrip_2_0_1::State

AssetStrip state

Enumerator

- DISCONNECTED** No strip connected.
- FIRMWARE_UPDATE** Firmware update in progress.
- UNSUPPORTED** Connected asset strip is unsupported.
- AVAILABLE** Asset strip is up and running normally.

9.13.2.4 enum assetmgrmodel::AssetStrip_2_0_1::StripType

Type of the connected asset strip.

Enumerator

SIMPLE single, monolithic strip

COMPOSITE strip consisting of multiple cascaded strips

9.13.2.5 enum assetmgrmodel::AssetStrip_2_0_1::TagType

Type of an asset tag connected to a rack unit.

Enumerator

SINGLE single asset tag connected to main strip or an extension

EXTENSION blade server extension, only possible on the main strip

NONE no asset tag connected to main strip or an extension

9.13.3 Member Function Documentation

9.13.3.1 vector<RackUnitInfo> assetmgrmodel::AssetStrip_2_0_1::getAllRackUnitInfos ()

Get infos with settings for all rack units.

Returns

Result: the rack unit infos with settings

9.13.3.2 vector<TagInfo> assetmgrmodel::AssetStrip_2_0_1::getAllTags ()

Get all asset tags of the strip.

Please note that in case there is not a single tag connected to the strip the resulting list will be empty, only connected tag info structures are returned It is guaranteed that extensions on the main strip are returned before any tag on an extension itself.

Returns

Result: list asset tag infos

9.13.3.3 DeviceInfo assetmgrmodel::AssetStrip_2_0_1::getDeviceInfo ()

Get static (hardware and firmware) information.

Returns

Result: hardware and firmware information

9.13.3.4 `int assetmgrmodel::AssetStrip_2_0_1::getExtensionTags ( in int rackUnitNumber, out vector< TagInfo > tags )`

Get all asset tags on an extension.

Gets all tags on a single extension for a certain rack unit. List will be empty if there are no tags connected

Parameters

<i>rackUnitNumber</i>	rack unit to get the extension tags for, range 0..rackUnitCount-1
<i>tags</i>	Result: list asset tag infos

Returns

NO_ERROR on success

ERR_INVALID_PARAM rack unit count exceeded or rack unit does not contain an extension

9.13.3.5 `vector<TagInfo> assetmgrmodel::AssetStrip_2_0_1::getMainTags ( )`

Get all asset tags on the main strip.

Same as getAllTags, but only consider tags connected to the main asset strip and not on any connected extension. Extensions on the main strip themselves are returned.

Returns

Result: list asset tag infos

9.13.3.6 `int assetmgrmodel::AssetStrip_2_0_1::getRackUnitInfo ( in int rackUnitNumber, out RackUnitInfo info )`

Get info with all settings of a rack unit at once.

Parameters

<i>rackUnitNumber</i>	rack unit to get the info for, range 0..rackUnitCount-1
<i>info</i>	Result: info for this rack unit

Returns

NO_ERROR on success

ERR_INVALID_PARAM rack unit count exceeded

9.13.3.7 `State assetmgrmodel::AssetStrip_2_0_1::getState ( )`

Get the current state of the AssetStrip.

Returns

State of the Asset Strip

9.13.3.8 `StripInfo assetmgrmodel::AssetStrip_2_0_1::getStripInfo ( )`

Get dynamic (number of tags) information.

Returns

Result: tag related information

9.13.3.9 `int assetmgrmodel::AssetStrip_2_0_1::getTag ( in int rackUnitNumber, in int slotNumber, out TagInfo tagInfo )`

Get the asset tag for a rack unit.

Parameters

<i>rackUnitNumber</i>	rack unit to read the asset tag for, range 0..rackUnitCount-1
<i>slotNumber</i>	slot to read the asset tag for, 0 for the main strip, >0 for blades
<i>tagInfo</i>	Result: asset tag information

Returns

NO_ERROR on success
 ERR_INVALID_PARAM rack unit count exceeded or colum not existing
 ERR_NO_SUCH_OBJECT no tag connected to this rack unit

9.13.3.10 `void assetmgrmodel::AssetStrip_2_0_1::triggerPowercycle ( in boolean hard )`

Trigger a powercycle of either the whole asset strip port or the LED part power supply on the asset strip.

Parameters

<i>hard</i>	true=whole port, false=LEDs only
-------------	----------------------------------

9.13.4 Member Data Documentation

9.13.4.1 `valueobject assetmgrmodel::AssetStrip_2_0_1::BladeOverflowChangedEvent`

Event: Tag overflow.

`Whether the strip is out of space`

for new blade extension tags

9.13.4.2 `valueobject assetmgrmodel::AssetStrip_2_0_1::CompositionChangedEvent`

Event: Strip composition has changed.

Component count before change

9.13.4.3 `valueobject assetmgrmodel::AssetStrip_2_0_1::FirmwareUpdateStateChangedEvent`

Event: Firmware update was started or has finished.

Update state

9.13.4.4 `constant int assetmgrmodel::AssetStrip_2_0_1::NO_ERROR = 0`

Error codes.

Operation successful, no error

9.13.4.5 `valueobject assetmgrmodel::AssetStrip_2_0_1::OrientationChangedEvent`

Event: Detected strip orientation has changed.

Strip orientation before change

9.13.4.6 valueobject assetmgrmodel::AssetStrip_2_0_1::RackUnitChangedEvent

Event: A rack unit has changed.

Affected rack unit position

9.13.4.7 valueobject assetmgrmodel::AssetStrip_2_0_1::StateChangedEvent

Event: Asset strip state has changed.

State before change

9.13.4.8 valueobject assetmgrmodel::AssetStrip_2_0_1::StripInfoChangedEvent

Event: Asset strip dynamic information has changed.

Information before change

9.13.4.9 valueobject assetmgrmodel::AssetStrip_2_0_1::TagEvent

Event: A tag was added or removed.

Affected tags

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/AssetStrip.idl

9.14 assetmgrmodel::AssetStripConfig_1_0_1 Interface Reference

Asset Strip Config interface.

```
import "AssetStripConfig.idl";
```

Classes

- struct [LEDColor](#)
The LED color in RGB format, 8 bit per channel.
- struct [RackUnitSettings](#)
Settings for a single rack unit (LED state)
- struct [StripSettings](#)
Settings for this Asset Strip.

Public Types

- enum [ScanMode](#) { [SCANMODE_DISABLED](#), [SCANMODE_BOTH](#) }
AssetStripConfig scan mode is active
- enum [NumberingMode](#) { [TOP_DOWN](#), [BOTTOM_UP](#) }
AssetStripConfig rack unit numbering mode
- enum [Orientation](#) { [TOP_CONNECTOR](#), [BOTTOM_CONNECTOR](#) }
AssetStripConfig orientation

- enum [LEDOperationMode](#) { [LED_OPERATION_MANUAL](#), [LED_OPERATION_AUTO](#) }
Operation mode for the LED of a single rack unit.
- enum [LEDMode](#) { [LED_MODE_ON](#), [LED_MODE_OFF](#), [LED_MODE_BLINK_FAST](#), [LED_MODE_BLINK_SLOW](#) }
Mode for the LED of a single rack unit.

Public Member Functions

- [StripSettings](#) [getStripSettings](#) ()
Get the asset strip settings which are not rack unit specific.
- int [setStripSettings](#) (in [StripSettings](#) settings)
Set the asset strip settings which are not rack unit specific.
- int [getRackUnitSettings](#) (in int rackUnitNumber, out [RackUnitSettings](#) settings)
Get settings of a rack unit at once.
- vector< [RackUnitSettings](#) > [getAllRackUnitSettings](#) ()
Get settings for all rack units.
- int [setRackUnitSettings](#) (in int rackUnitNumber, in [RackUnitSettings](#) settings)
Set all settings of the specified rack unit at once.
- int [setMultipleRackUnitSettings](#) (in map< int, [RackUnitSettings](#) > settings)
Set all settings of multiple rack units at once.

Public Attributes

- valueobject [StripSettingsChangedEvent](#): event.UserEvent { [StripSettings](#) oldSettings
Event: Asset strip settings were changed.
- [StripSettings](#) newSettings
Settings after change.
- valueobject [RackUnitSettingsChangedEvent](#): event.UserEvent { int rackUnitNumber
Event: A rack unit's settings were changed.
- [RackUnitSettings](#) oldSettings
Settings before change.
- [RackUnitSettings](#) newSettings
Settings after change.

9.14.1 Detailed Description

Asset Strip Config interface.

9.14.2 Member Enumeration Documentation

9.14.2.1 enum assetmgrmodel::AssetStripConfig_1_0_1::LEDMode

Mode for the LED of a single rack unit.

Enumerator

- [LED_MODE_ON](#)** LED on.
- [LED_MODE_OFF](#)** LED off.
- [LED_MODE_BLINK_FAST](#)** LED is blinking (fast)
- [LED_MODE_BLINK_SLOW](#)** LED is blinking (slow)

9.14.2.2 enum assetmgrmodel::AssetStripConfig_1_0_1::LEDOperationMode

Operation mode for the LED of a single rack unit.

The LED may either be manually controlled or its color may be chosen automatically depending on whether an asset tag is connected or not. In automatic mode the LED is always on.

Enumerator

LED_OPERATION_MANUAL LED color and mode is manually controlled.

LED_OPERATION_AUTO LED color controlled automatically, LED always on.

9.14.2.3 enum assetmgrmodel::AssetStripConfig_1_0_1::NumberingMode

AssetStripConfig rack unit numbering mode

Indicates the way the rack unit numbers are defined. Basically this determines what number are 'written/printed' at a specific rack unit on the rack.

An additional numberingOffset may be specified in the settings if the numbering does not start at the default 1.

Enumerator

TOP_DOWN numbering goes from top to bottom, top is the smallest number

BOTTOM_UP numbering goes from bottom to top, bottom is the smallest number

9.14.2.4 enum assetmgrmodel::AssetStripConfig_1_0_1::Orientation

AssetStripConfig orientation

The asset strip may be mounted in a rack a) with its cable connector on top, growing top->bottom b) with its cable connector on bottom, growing bottom->top

This enumeration indicates the mounting option. It is detected automatically and writes to the setting are ignored.

Enumerator

TOP_CONNECTOR cable connector on top, strip growing top->bottom

BOTTOM_CONNECTOR cable connector on top, strip growing bottom->top

9.14.2.5 enum assetmgrmodel::AssetStripConfig_1_0_1::ScanMode

AssetStripConfig scan mode is active

This is a demonstration feature providing a 'running lights' kind of effect where not all LEDs are lit statically (depending on their mode) but only a few LED which change all the time.

Enumerator

SCANMODE_DISABLED LED scanmode is disabled, all LEDs are lit up statically.

SCANMODE_BOTH LED scanmode is enabled providing a 'running light' effect.

9.14.3 Member Function Documentation

9.14.3.1 vector<RackUnitSettings> assetmgrmodel::AssetStripConfig_1_0_1::getAllRackUnitSettings ()

Get settings for all rack units.

Returns

Result: the rack unit settings

9.14.3.2 `int assetmgrmodel::AssetStripConfig_1_0_1::getRackUnitSettings ( in int rackUnitNumber, out RackUnitSettings settings )`

Get settings of a rack unit at once.

Parameters

<i>rackUnitNumber</i>	rack unit to get the settings for, range 0..rackUnitCount-1
<i>settings</i>	settings for this rack unit

Returns

NO_ERROR on success

ERR_INVALID_PARAM rack unit count exceeded

9.14.3.3 `StripSettings assetmgrmodel::AssetStripConfig_1_0_1::getStripSettings ( )`

Get the asset strip settings which are not rack unit specific.

Returns

settings settings for this asset strip

9.14.3.4 `int assetmgrmodel::AssetStripConfig_1_0_1::setMultipleRackUnitSettings ( in map< int, RackUnitSettings > settings )`

Set all settings of multiple rack units at once.

Parameters

<i>settings</i>	map of rack units and their settings
-----------------	--------------------------------------

Returns

NO_ERROR on success

ERR_INVALID_PARAM one or more of the settings is invalid (rack unit count exceeded or color/mode not supported, rack unit count and setting count differ...)

9.14.3.5 `int assetmgrmodel::AssetStripConfig_1_0_1::setRackUnitSettings ( in int rackUnitNumber, in RackUnitSettings settings )`

Set all settings of the specified rack unit at once.

The addressed rack unit is part of the parameter

Parameters

<i>rackUnitNumber</i>	rack unit to set the settings for, range 0..rackUnitCount-1
<i>settings</i>	settings for this rack unit

Returns

NO_ERROR on success
 ERR_INVALID_PARAM one or more of the settings is invalid

9.14.3.6 int assetmgrmodel::AssetStripConfig_1_0_1::setStripSettings (in StripSettings settings)

Set the asset strip settings which are not rack unit specific.

Parameters

<i>settings</i>	Settings for this asset strip
-----------------	-------------------------------

Returns

NO_ERROR on success
 ERR_INVALID_PARAM one or more of the settings is invalid

9.14.4 Member Data Documentation**9.14.4.1 valueobject assetmgrmodel::AssetStripConfig_1_0_1::RackUnitSettingsChangedEvent**

Event: A rack unit's settings were changed.

Affected rack unit position

9.14.4.2 valueobject assetmgrmodel::AssetStripConfig_1_0_1::StripSettingsChangedEvent

Event: Asset strip settings were changed.

Settings before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/AssetStripConfig.idl

9.15 assetmgrmodel::AssetStripLogger_1_0_2 Interface Reference

Asset Strip Logger interface.

```
import "AssetStripLogger.idl";
```

Classes

- struct [Info](#)
Log information structure.
- struct [Record](#)
Log record structure.

Public Types

- enum [RecordType](#) { [EMPTY](#), [ASSET_TAG_CONNECTED](#), [ASSET_TAG_DISCONNECTED](#), [ASSET_STRIP_STATE_CHANGED](#) }
Log record type.

Public Member Functions

- [Info](#) `getInfo ()`
Retrieve the log information.
- `int` [getRecords](#) (out vector< [Record](#) > records, in int id, in int count)
Retrieve records from the log ring buffer.

Public Attributes

- constant int [NO_ERROR](#) = 0
Error codes.
- constant int [ERR_INVALID_PARAM](#) = 1
Invalid parameter.

9.15.1 Detailed Description

Asset Strip Logger interface.

9.15.2 Member Enumeration Documentation

9.15.2.1 enum assetmgrmodel::AssetStripLogger_1_0_2::RecordType

Log record type.

Enumerator

- EMPTY** The log record is empty.
- ASSET_TAG_CONNECTED** An asset tag has been connected.
- ASSET_TAG_DISCONNECTED** An asset tag has been disconnected.
- ASSET_STRIP_STATE_CHANGED** The state of the asset strip changed.

9.15.3 Member Function Documentation

9.15.3.1 Info assetmgrmodel::AssetStripLogger_1_0_2::getInfo ()

Retrieve the log information.

Returns

Log information

9.15.3.2 int assetmgrmodel::AssetStripLogger_1_0_2::getRecords (out vector< [Record](#) > records, in int id, in int count)

Retrieve records from the log ring buffer.

This method is used to read one or more records from the log ring buffer. It is allowed to read unused entries. The record index will wrap to 0 when reading beyond the end of the log.

Parameters

<i>records</i>	Result: The requested log entries
<i>id</i>	Index of the first log index to read (0..capacity)
<i>count</i>	Number of records to read (1..capacity)

Returns

NO_ERROR on success
 ERR_INVALID_PARAM if any parameter is out of range

9.15.4 Member Data Documentation

9.15.4.1 constant int assetmgrmodel::AssetStripLogger_1_0_2::NO_ERROR = 0

Error codes.

Operation successful, no error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/AssetStripLogger.idl

9.16 auth::AuthManager Interface Reference

Authentication manager interface.

```
import "AuthManager.idl";
```

Public Member Functions

- [Policy](#) [getPolicy](#) ()
Gets a policy.
- int [setPolicy](#) (in [Policy](#) p)
Sets a policy.

Public Attributes

- constant int [ERR_UNSUPPORTED_TYPE](#) = 1
Unsupported authentication type.

9.16.1 Detailed Description

Authentication manager interface.

9.16.2 Member Function Documentation

9.16.2.1 [Policy](#) [auth::AuthManager::getPolicy](#) ()

Gets a policy.

Returns

a [Policy](#) object

9.16.2.2 int auth::AuthManager::setPolicy (in Policy p)

Sets a policy.

Returns

- 0 on success
- 1 in case authentication type is not supported

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/AuthManager.idl

9.17 usermgmt::AuxInfo Struct Reference

Auxiliary user information.

```
import "User.idl";
```

Public Attributes

- string [fullname](#)
Full name.
- string [telephone](#)
Telephone number.
- string [eMail](#)
Email address.

9.17.1 Detailed Description

Auxiliary user information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/User.idl

9.18 test::AuxSerial Interface Reference

test routines for Raritan Aux Serial interface (RS485 on pins 3 and 6 of RJ45) Require TestMode to be ON.

```
import "testrpc.idl";
```

Public Member Functions

- int [getNumberOfPorts](#) (out int numPorts)
returns number of ports
- int [testLoop](#) (in int portNum, out string errstr)
Performs a loop test with special test adapter.

Public Attributes

- constant int `OK` = 0
No error.
- constant int `ERR_NO_TEST_MODE` = 1
Not in test mode.
- constant int `ERR_INVALID_PORT_NUM` = 2
Invalid port number.
- constant int `ERR_TEST_FAILED` = 3
Test failed.

9.18.1 Detailed Description

test routines for Raritan Aux Serial interface (RS485 on pins 3 and 6 of RJ45) Require TestMode to be ON.

9.18.2 Member Function Documentation

9.18.2.1 int test::AuxSerial::getNumberOfPorts (out int *numPorts*)

returns number of ports

Returns

- 0 if OK
- 1 if not in test mode

9.18.2.2 int test::AuxSerial::testLoop (in int *portNum*, out string *errstr*)

Performs a loop test with special test adapter.

Returns

- OK if OK
- ERR_NO_TEST_MODE if not in test mode
- ERR_INVALID_PORT_NUM if invalid port number
- ERR_TEST_FAILED if test failed, *errstr* may be set

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/testrpc.idl

9.19 pdumodel::Bcm Interface Reference

Branch Circuit Monitor.

```
import "Bcm.idl";
```

Classes

- struct `ChannelConfig`
Channel Configuration.
- struct `PhaseConfig`
Phase Configuration.

Public Types

- enum [LineConfig](#) { [UNCONNECTED](#), [L1_N](#), [L2_N](#), [L3_N](#) }
Power Line Configuration.
- enum [TransformerType](#) { [VOLTAGE](#), [TURNSRATIO](#) }
Current Transformer Type.

Public Member Functions

- int [getChannelCount](#) ()
Retrieve the BMC's channel count.
- vector< [ChannelConfig](#) > [getChannelConfigs](#) ()
Retrieve the configuration for all channels.
- int [setChannelConfig](#) (in int channel, in [ChannelConfig](#) config)
Set the configuration for one channel.

9.19.1 Detailed Description

Branch Circuit Monitor.

9.19.2 Member Enumeration Documentation

9.19.2.1 enum `pdumodel::Bcm::LineConfig`

Power Line Configuration.

Enumerator

UNCONNECTED Not connected.
L1_N L1-Neutral.
L2_N L2-Neutral.
L3_N L3-Neutral.

9.19.2.2 enum `pdumodel::Bcm::TransformerType`

Current Transformer Type.

Enumerator

VOLTAGE Voltage-type transformer.
TURNSRATIO Turns ratio transformer.

9.19.3 Member Function Documentation

9.19.3.1 vector<[ChannelConfig](#)> `pdumodel::Bcm::getChannelConfigs` ()

Retrieve the configuration for all channels.

Returns

List of channel configurations

9.19.3.2 `int pdumodel::Bcm::getChannelCount ( )`

Retrieve the BMC's channel count.

Returns

Number of BCM channels

9.19.3.3 `int pdumodel::Bcm::setChannelConfig ( in int channel, in ChannelConfig config )`

Set the configuration for one channel.

Parameters

<i>channel</i>	Channel number
<i>config</i>	New channel configuration

Returns

0 if OK
1 for invalid phase configuration
2 for invalid arguments

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Bcm.idl

9.20 bulkcfg::BulkConfiguration Interface Reference

Bulk Configuration Interface.

```
import "BulkConfiguration.idl";
```

Public Types

- enum [Status](#) {
UNKNOWN, UPLOAD_FAILED, RESTORE_PENDING, RESTORE_OK,
RESTORE_FAILED }

Status of the last bulk configuration restore operation.

Public Member Functions

- void [getStatus](#) (out [Status](#) status, out time timeStamp)
Retrieve the status of the last bulk configuration restore operation.

9.20.1 Detailed Description

Bulk Configuration Interface.

9.20.2 Member Enumeration Documentation

9.20.2.1 enum bulkcfg::BulkConfiguration::Status

Status of the last bulk configuration restore operation.

Enumerator

- UNKNOWN** No bulk configuration was done yet.
- UPLOAD_FAILED** Uploading a bulk configuration failed.
- RESTORE_PENDING** Restore is pending.
- RESTORE_OK** Restoring bulk configuration successful.
- RESTORE_FAILED** Restoring bulk configuration failed.

9.20.3 Member Function Documentation

9.20.3.1 void bulkcfg::BulkConfiguration::getStatus (out Status *status*, out time *timeStamp*)

Retrieve the status of the last bulk configuration restore operation.

Parameters

<i>status</i>	Result: Bulk configuration restore status
<i>timeStamp</i>	Result: Time of last restore operation

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/BulkConfiguration.idl

9.21 Ihxmodel::Lhx_3_2_1::Capabilities Struct Reference

LHX capabilities.

```
import "Lhx.idl";
```

Public Attributes

- [AlertStatus](#) **alerts**
- map< string, boolean > **features**

9.21.1 Detailed Description

LHX capabilities.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Lhx.idl

9.22 smartcard::CardReader::CardInformation Struct Reference

Card Information.

```
import "CardReader.idl";
```


Public Attributes

- string [type](#)
card type
- string [uid](#)
card id

9.22.1 Detailed Description

Card Information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/CardReader.idl

9.23 smartcard::CardReader Interface Reference

Card Reader Interface.

```
import "CardReader.idl";
```

Classes

- struct [CardInformation](#)
Card Information.
- struct [MetaData](#)
Reader Metadata.

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve reader metadata.
- int [getCardInformation](#) (out [CardInformation](#) cardInfo)
Retrieve card information from reader.

Public Attributes

- constant int [NO_ERROR](#) = 0
Error codes.
- constant int [ERR_SLOT_EMPTY](#) = 1
no card present in reader
- valueobject [CardEvent](#): [idl.Event](#) { [CardInformation](#) cardInfo
Card base event.
- valueobject [CardInsertedEvent](#): [CardEvent](#) {}
Card inserted event.
- valueobject [CardRemovedEvent](#): [CardEvent](#) {}
Card removed event.

9.23.1 Detailed Description

Card Reader Interface.

9.23.2 Member Function Documentation

9.23.2.1 `int smartcard::CardReader::getCardInformation ( out CardInformation cardInfo )`

Retrieve card information from reader.

Parameters

<i>cardInfo</i>	Card Information
-----------------	------------------

Returns

NO_ERROR if OK
ERR_SLOT_EMPTY if reader sees no card

9.23.2.2 `MetaData smartcard::CardReader::getMetaData ( )`

Retrieve reader metadata.

Returns

metadata

9.23.3 Member Data Documentation

9.23.3.1 `valueobject smartcard::CardReader::CardEvent`

Card base event.

card info of affected card

9.23.3.2 `constant int smartcard::CardReader::NO_ERROR = 0`

Error codes.

operation successful, no error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/CardReader.idl

9.24 `smartcard::CardReaderManager` Interface Reference

Card Reader Manager Interface.

```
import "CardReaderManager.idl";
```

Public Member Functions

- `vector< CardReader > getCardReaders ()`
Retrieve the list of connected card readers.
- `CardReader getCardReaderById (in string readerId)`
Get card reader for a specific id.

Public Attributes

- valueobject [CardReaderEvent](#): [idl.Event](#) { [CardReader](#) cardReader
Card Reader base event.
- valueobject [CardReaderAttachedEvent](#): [CardReaderEvent](#) {}
Card Reader attached event.
- valueobject [CardReaderDetachedEvent](#): [CardReaderEvent](#) {}
Card Reader detached event.

9.24.1 Detailed Description

Card Reader Manager Interface.

9.24.2 Member Function Documentation

9.24.2.1 [CardReader](#) smartcard::CardReaderManager::getCardReaderById (in string *readerId*)

Get card reader for a specific id.

Parameters

<i>readerId</i>	card reader id
-----------------	----------------

Returns

Card Reader with given id or null

9.24.2.2 [vector](#)<[CardReader](#)> smartcard::CardReaderManager::getCardReaders ()

Retrieve the list of connected card readers.

Returns

Card Readers list

9.24.3 Member Data Documentation

9.24.3.1 valueobject smartcard::CardReaderManager::CardReaderEvent

Card Reader base event.

affected card reader

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/CardReaderManager.idl

9.25 cascading::Cascading_1_0_1 Interface Reference

Cascading Interface.

```
import "Cascading.idl";
```

Classes

- struct [ProtocolMapping](#)
Mapping from appl protocol id to name and transport protocol.

Public Types

- enum [Type](#) { [USB_MULTI_IP](#), [USB_SINGLE_IP_NAT](#) }
Cascading type.

Public Member Functions

- int [setType](#) (in [Type](#) type)
Set the cascading type.
- int [getType](#) (out [Type](#) type)
Get the cascading type.
- void [getIndex](#) (out int idx)
Get the index of this unit in the cascade.
- void [getMasterIpAddress](#) (out string masterIpAddress)
Get the IPv4 address of the master.
- void [getMasterIpV6Address](#) (out string masterIpV6Address)
Get the IPv6 address of the master.
- void [getProtocolMappings](#) (out vector< [ProtocolMapping](#) > mappings)
Get the application protocol id -> name/transport-proto mappings.

Public Attributes

- constant int [ERR_NOT_AVAILABLE](#) = 1
Error codes.
- constant int [ERR_NOT_SUPPORTED_ON_SLAVE](#) = 2

9.25.1 Detailed Description

Cascading Interface.

9.25.2 Member Enumeration Documentation

9.25.2.1 enum cascading::Cascading_1_0_1::Type

Cascading type.

Enumerator

[USB_MULTI_IP](#) USB chain; one IP per device; transparent.

[USB_SINGLE_IP_NAT](#) USB chain; single IP; NAT(port-forwarding)

9.25.3 Member Function Documentation

9.25.3.1 void cascading::Cascading_1_0_1::getIndex (out int *idx*)

Get the index of this unit in the cascade.

Index 0 means this is the master unit. A cascading master is a unit with an active ethernet uplink and no active uplink on the USB-B connector.

Parameters

<i>idx</i>	- the index of this unit in the cascade
------------	---

9.25.3.2 void cascading::Cascading_1_0_1::getMasterIpAddress (out string *masterIpAddress*)

Get the IPv4 address of the master.

If the address is not available an empty string is returned.

Parameters

<i>masterIpAddress</i>	- the IPv4 address of the master
------------------------	----------------------------------

9.25.3.3 void cascading::Cascading_1_0_1::getMasterIpV6Address (out string *masterIpV6Address*)

Get the IPv6 address of the master.

If the address is not available an empty string is returned.

Parameters

<i>masterIpV6-Address</i>	- the IPv6 address of the master
---------------------------	----------------------------------

9.25.3.4 void cascading::Cascading_1_0_1::getProtocolMappings (out vector< ProtocolMapping > *mappings*)

Get the application protocol id -> name/transport-proto mappings.

Parameters

<i>mappings</i>	- the protocol id -> name/transport-proto mappings
-----------------	--

9.25.3.5 int cascading::Cascading_1_0_1::getType (out Type *type*)

Get the cascading type.

A slave unit will return ERR_NOT_AVAILABLE when the master unit has not yet propagated the cascading type to the slaves. This behaviour is typically seen while the device is booting and network setup did not finish yet.

Parameters

<i>type</i>	- the cascading type
-------------	----------------------

Returns

0 on success
 ERR_NOT_AVAILABLE if not yet known

9.25.3.6 int cascading::Cascading_1.0.1::setType (in Type type)

Set the cascading type.

This operation is only supported on the cascading master. On slave units it will return ERR_NOT_SUPPORTED_ON_SLAVE. A cascading master is a unit with an active ethernet uplink and no active uplink on the USB-B connector.

Parameters

<i>type</i>	- the cascading type
-------------	----------------------

Returns

0 on success
 ERR_NOT_SUPPORTED_ON_SLAVE if this is a slave unit

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Cascading.idl

9.26 cert::ServerSSLCert::CertInfo Struct Reference

Certificate information.

```
import "ServerSSLCert.idl";
```

Public Attributes

- [CommonAttributes subject](#)
Subject attributes.
- [CommonAttributes issuer](#)
Issuer attributes.
- string [invalidBefore](#)
Begin of validity period.
- string [invalidAfter](#)
End of validity period.
- string [serialNumber](#)
Serial number.
- int [keyLength](#)
Key length in bits.

9.26.1 Detailed Description

Certificate information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ServerSSLCert.idl

9.27 datetime::DateTime_2_0_0::Cfg Struct Reference

Device date and time configuration.

```
import "DateTime.idl";
```

Public Attributes

- [ZoneCfg zoneCfg](#)
Time zone configuration.
- [Protocol protocol](#)
Time synchronization protocol.
- time [deviceTime](#)
Device date and time (local time)
- [NtpCfg ntpCfg](#)
NTP server configuration.

9.27.1 Detailed Description

Device date and time configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/DateTime.idl

9.28 webcam::Channel Interface Reference

The channel interface.

```
import "WebcamChannel.idl";
```

Public Member Functions

- string [getClientType](#) ()
Retrieve the channel type.
- [Webcam_2_0_0 getWebcam](#) ()
Retrieve the webcam.
- boolean [isAvailable](#) ()
Determine whether the channel is available.
- void [release](#) ()
Releases the channel to allow lower proritized clients to view an image otherwise the channel expires when it is unused (neither a image related method nor "isAvailable" is called) for 20 seconds.
- int [captureImage](#) (out [Image_2_0_0](#) image)
Retrieve a image.
- int [triggerCapture](#) (out string captureToken)
Triggers the capture process to the temporary folder and returns a capture token to view them.

Public Attributes

- constant int `NO_ERROR` = 0
Error codes.
- constant int `ERR_NOT_AVAILABLE` = 2
Channel is not available.
- constant int `ERR_NO_SUCH_OBJECT` = 3
Requested object does not exist.

9.28.1 Detailed Description

The channel interface.

9.28.2 Member Function Documentation

9.28.2.1 `int webcam::Channel::captureImage ( out Image_2_0_0 image )`

Retrieve a image.

Parameters

<i>image</i>	Result: the image
--------------	-------------------

Returns

`NO_ERROR` on success
`ERR_NOT_AVAILABLE` this channel is temporarily paused, because all view ports are currently used by higher prioritized events
`ERR_NO_SUCH_OBJECT` the viewed webcam was detached

9.28.2.2 `string webcam::Channel::getClientType ( )`

Retrieve the channel type.

Returns

the client type

9.28.2.3 `Webcam_2_0_0 webcam::Channel::getWebcam ( )`

Retrieve the webcam.

Returns

the webcam

9.28.2.4 `boolean webcam::Channel::isAvailable ( )`

Determine whether the channel is available.

A channel may become unavailable when higher prioritized viewers are active.

Returns

true if the channel is available, false otherwise

9.28.2.5 int webcam::Channel::triggerCapture (out string *captureToken*)

Triggers the capture process to the temporary folder and returns a capture token to view them.

The capture token is valid for 9 seconds and needs to be refreshed when, or even better before the old token expires.

Parameters

<i>captureToken</i>	Result: a new captureToken
---------------------	----------------------------

Returns

NO_ERROR on success

ERR_NOT_AVAILABLE this channel is temporarily paused, because all view ports are currently used by higher prioritized events

ERR_NO_SUCH_OBJECT the viewed webcam was detached

9.28.3 Member Data Documentation

9.28.3.1 constant int webcam::Channel::NO_ERROR = 0

Error codes.

Operation successful, no error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/WebcamChannel.idl

9.29 event::Channel_1_0_1 Interface Reference

[Event](#) Channel.

```
import "EventService.idl";
```

Classes

- struct [EventSelect](#)
*Structure to select an [Event](#) *.*

Public Member Functions

- void [demandEventType](#) (in typecode type)
Subscribe for events of a given type.
- void [cancelEventType](#) (in typecode type)
Cancel the subscription for events of a given type.
- void [demandEventTypes](#) (in vector< typecode > types)
Subscribe for multiple event types at once.
- void [cancelEventTypes](#) (in vector< typecode > types)
Cancel subscription for events that are of any of given types.
- void [demandEvent](#) (in typecode type, in Object src)
Subscribe for events that are of given type and emitted by a specific object instance.
- void [cancelEvent](#) (in typecode type, in Object src)

Cancel the subscription for events that are of given type and emitted by a specific object instance.

- void [demandEvents](#) (in vector< [EventSelect](#) > events)

Subscribe for multiple specific events at once.

- void [cancelEvents](#) (in vector< [EventSelect](#) > events)

Cancel the subscription for multiple specific events.

- void [subscribe](#) (in [Consumer](#) consumer)

Subscribe for push notifications from EventService.

- int [unsubscribe](#) (in [Consumer](#) consumer)

Unsubscribe from push notifications from EventService.

- boolean [pollEvents](#) (out vector< [idl.Event](#) > events)

Poll for new events blockingly.

- boolean [pollEventsNb](#) (out vector< [idl.Event](#) > events)

Poll for new events non-blockingly.

9.29.1 Detailed Description

[Event](#) Channel.

9.29.2 Member Function Documentation

9.29.2.1 void event::Channel_1_0_1::cancelEvent (in typecode *type*, in Object *src*)

Cancel the subscription for events that are of given type and emitted by a specific object instance.

Parameters

<i>evttype</i>	Event typecode
<i>src</i>	Event source instance

9.29.2.2 void event::Channel_1_0_1::cancelEvents (in vector< [EventSelect](#) > *events*)

Cancel the subscription for multiple specific events.

Parameters

<i>events</i>	List of typecodes to unsubscribe from
---------------	---------------------------------------

9.29.2.3 void event::Channel_1_0_1::cancelEventType (in typecode *type*)

Cancel the subscription for events of a given type.

Parameters

<i>evttype</i>	typecode of valueobject of demanded event
----------------	---

9.29.2.4 void event::Channel_1_0_1::cancelEventTypes (in vector< typecode > *types*)

Cancel subscription for events that are of any of given types.

Parameters

<i>evtypes</i>	List of event typecodes
----------------	-------------------------

9.29.2.5 void event::Channel_1_0_1::demandEvent (in typecode *type*, in Object *src*)

Subscribe for events that are of given type and emitted by a specific object instance.

Parameters

<i>evtype</i>	Event typecode
<i>src</i>	Event source instance

9.29.2.6 void event::Channel_1_0_1::demandEvents (in vector< EventSelect > *events*)

Subscribe for multiple specific events at once.

Parameters

<i>events</i>	List of typecodes to subscribe for
---------------	------------------------------------

9.29.2.7 void event::Channel_1_0_1::demandEventType (in typecode *type*)

Subscribe for events of a given type.

Parameters

<i>evtype</i>	typecode of valueobject of demanded event
---------------	---

9.29.2.8 void event::Channel_1_0_1::demandEventTypes (in vector< typecode > *types*)

Subscribe for multiple event types at once.

Parameters

<i>evtypes</i>	List of event typecodes
----------------	-------------------------

9.29.2.9 boolean event::Channel_1_0_1::pollEvents (out vector< idl.Event > *events*)

Poll for new events blockingly.

This method will block in case the queue is empty. It will return as soon as at least one event is available, or after a maximum wait time of 30 seconds.

The method will not return more than an implementation-defined maximum number of events. The boolean return value indicates whether there are more events in the queue.

Parameters

<i>events</i>	List of new events
---------------	--------------------

Returns

`true` if there are more events in the queue `false` if the queue is empty

9.29.2.10 boolean event::Channel_1_0_1::pollEventsNb (out vector< idl.Event > events)

Poll for new events non-blockingly.

The method will not return more than an implementation-defined maximum number of events. The boolean return value indicates whether there are more events in the queue.

Parameters

<i>events</i>	List of new events
---------------	--------------------

Returns

true if there are more events in the queue *false* if the queue is empty

9.29.2.11 void event::Channel_1_0_1::subscribe (in Consumer consumer)

Subscribe for push notifications from EventService.

Parameters

<i>consumer</i>	interface on which subscriber is called back.
-----------------	---

9.29.2.12 int event::Channel_1_0_1::unsubscribe (in Consumer consumer)

Unsubscribe from push notifications from EventService.

Parameters

<i>consumer</i>	interface which was used for the subscription.
-----------------	--

Returns

0 if OK 1 if consumer is unknown to channel

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/EventService.idl

9.30 pdumodel::Bcm::ChannelConfig Struct Reference

Channel Configuration.

```
import "Bcm.idl";
```

Public Attributes

- int [channel](#)
Channel number.
- string [label](#)
Channel label.
- boolean [is3Phase](#)
true for three-phase channels

- [PhaseConfig lineA](#)
Line A phase config.
- [PhaseConfig lineB](#)
Line B phase config (three-phase only)
- [PhaseConfig lineC](#)
Line C phase config (three-phase only)

9.30.1 Detailed Description

Channel Configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Bcm.idl

9.31 pdumodel::CircuitBreakerStatistic Struct Reference

Overcurrent protector statistics.

```
import "OverCurrentProtector.idl";
```

Public Attributes

- int [tripCnt](#)
Trip count.

9.31.1 Detailed Description

Overcurrent protector statistics.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/OverCurrentProtector.idl

9.32 cert::ServerSSLCert::CommonAttributes Struct Reference

Certificate issuer or subject attributes.

```
import "ServerSSLCert.idl";
```

Public Attributes

- string [country](#)
Country code.
- string [stateOrProvince](#)
State or province.
- string [locality](#)
Locality or city.
- string [organization](#)
Organization.
- string [organizationalUnit](#)

Organizational Unit.

- string [commonName](#)

Common Name.

- string [emailAddress](#)

Email Address.

9.32.1 Detailed Description

Certificate issuer or subject attributes.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ServerSSLCert.idl

9.33 Ihxmodel::Config_1_0_1::ComSettings Struct Reference

LHX port communication settings.

```
import "LhxConfig.idl";
```

Public Attributes

- byte [devAddr](#)

EMX/PX STBus address.

- byte [lhxAddr](#)

LHX STBus address.

9.33.1 Detailed Description

LHX port communication settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LhxConfig.idl

9.34 event::Engine::Condition Struct Reference

[Condition](#) is a logical combination of multiple events.

```
import "EventEngine.idl";
```

Public Types

- enum [Op](#) { [AND](#), [OR](#), [XOR](#) }

logical operation to be applied over all conditions and event

- enum [MatchType](#) { [ASSERTED](#), [DEASSERTED](#), [BOTH](#) }

the match type how to match the event assertion state

Public Attributes

- boolean [negate](#)
Negate the result.
- [Op](#) operation
Logical operation to be applied.
- [MatchType](#) matchType
Match type.
- vector< string > [eventId](#)
Event ID.
- vector< [Condition](#) > [conditions](#)
List of subordinate conditions.

9.34.1 Detailed Description

[Condition](#) is a logical combination of multiple events.

Normally you should use STATE events in logical conditions. In case a TRIGGER event is part of a condition or logical operation it will per default be interpreted as deasserted. The TRIGGER will evaluate as asserted if that exact event was raised and is matched against the event rules. In some sense the 'assertion state' of the TRIGGER is true only at the moment the [Event](#) exists and set to false again once it passed.

9.34.2 Member Enumeration Documentation

9.34.2.1 enum event::Engine::Condition::MatchType

the match type how to match the event assertion state

Enumerator

- ASSERTED** Match if the event is asserted.
- DEASSERTED** Match if the event is deasserted.
- BOTH** Match both (for trigger events)

9.34.2.2 enum event::Engine::Condition::Op

logical operation to be applied over all conditions and event

Enumerator

- AND** Logical And.
- OR** Logical Or.
- XOR** Exclusive Or.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EventEngine.idl

9.35 Ihxmodel::Config_1_0_1 Interface Reference

LHX Configuration Interface.

```
import "LhxConfig.idl";
```

Classes

- struct [ComSettings](#)
LHX port communication settings.

Public Member Functions

- [ComSettings](#) [getComSettings](#) ()
Retrieve the LHX port communication settings.
- int [setComSettings](#) (in [ComSettings](#) settings)
Change the LHX port communication settings.
- string [getName](#) ()
Retrieve the LHX port name.
- int [setName](#) (in string name)
Change the LHX port name.

Public Attributes

- constant int [NO_ERROR](#) = 0
No error.
- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.
- valueobject [ComSettingsChangedEvent](#): [event.UserEvent](#) { [ComSettings](#) oldSettings
Event: LHX port communication settings have changed.
- [ComSettings](#) [newSettings](#)
Settings after change.
- valueobject [PortNameChangedEvent](#): [event.UserEvent](#) { string oldName
Event: LHX port name has been changed.
- string [newName](#)
Name after change.

9.35.1 Detailed Description

LHX Configuration Interface.

9.35.2 Member Function Documentation

9.35.2.1 [ComSettings](#) [lhxmodel::Config_1_0_1::getComSettings](#) ()

Retrieve the LHX port communication settings.

Returns

LHX port communication settings

9.35.2.2 [string](#) [lhxmodel::Config_1_0_1::getName](#) ()

Retrieve the LHX port name.

Returns

LHX port name

9.35.2.3 int lhxmodel::Config_1_0_1::setComSettings (in ComSettings *settings*)

Change the LHX port communication settings.

Parameters

<i>settings</i>	New LHX port communication settings
-----------------	-------------------------------------

Returns

0 if OK
ERR_INVALID_PARAMS if any parameters are invalid

9.35.2.4 int lhxmodel::Config_1_0_1::setName (in string *name*)

Change the LHX port name.

Parameters

<i>name</i>	New LHX port name
-------------	-------------------

Returns

0 if OK
ERR_INVALID_PARAMS if any parameters are invalid

9.35.3 Member Data Documentation

9.35.3.1 valueobject lhxmodel::Config_1_0_1::ComSettingsChangedEvent

Event: LHX port communication settings have changed.

Settings before change

9.35.3.2 valueobject lhxmodel::Config_1_0_1::PortNameChangedEvent

Event: LHX port name has been changed.

Name before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/LhxConfig.idl

9.36 modelpush::ModelPush::Configuration Struct Reference

Model push service configuration.

```
import "ModelPush.idl";
```

Public Attributes

- boolean [enabled](#)
Enable the sensor log push service.

- int [interval](#)
Push interval in seconds.
- string [host](#)
Destination host.
- int [port](#)
Destination HTTP port.
- string [path](#)
Destination path.
- boolean [useHttps](#)
true to enable HTTPS
- string [caCert](#)
HTTPS CA certificate.
- boolean [useAuth](#)
true to use HTTP basic authentication
- string [username](#)
Authentication user name.
- string [password](#)
Password; write-only, empty to leave unchanged.

9.36.1 Detailed Description

Model push service configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ModelPush.idl

9.37 devsettings::Smtp_1_0_1::Configuration Struct Reference

SMTP server configuration.

```
import "Smtp.idl";
```

Public Attributes

- string [host](#)
SMTP server host name or IP address.
- int [port](#)
SMTP server port.
- string [sender](#)
Sender email address.
- boolean [useAuth](#)
SMTP server requires authentication.
- string [username](#)
Authentication user name.
- string [password](#)
Password; write-only, empty to leave unchanged.
- int [retryCount](#)
Number of attempts at sending the email.
- int [retryInterval](#)
Sending retry interval in minutes.

9.37.1 Detailed Description

SMTP server configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Sntp.idl

9.38 devsettings::Snmp_1_0_2::Configuration Struct Reference

SNMP agent configuration.

```
import "Snmp.idl";
```

Public Attributes

- boolean [v2enable](#)
SNMP v1 / 2c enabled.
- boolean [v3enable](#)
SNMP v3 enabled.
- string [readComm](#)
read community string
- string [writeComm](#)
write community string
- string [sysContact](#)
system contact
- string [sysName](#)
system name
- string [sysLocation](#)
system location

9.38.1 Detailed Description

SNMP agent configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Snmp.idl

9.39 event::Consumer Interface Reference

[Consumer](#) interface is for event consumers that want to be called back in case new events have occurred.

```
import "EventService.idl";
```

Public Member Functions

- void [pushEvents](#) (in vector< [idl.Event](#) > events)
This method is called by EventService to notify [Consumer](#) about new events.

9.39.1 Detailed Description

[Consumer](#) interface is for event consumers that want to be called back in case new events have occurred.

Subscription with this interface happens in Channel interface. Note that this interface cannot be used over transports that do not support callbacks.

9.39.2 Member Function Documentation

9.39.2.1 void event::Consumer::pushEvents (in vector< [idl.Event](#) > *events*)

This method is called by EventService to notify [Consumer](#) about new events.

[Consumer](#) is not supposed to block this function but return immediately. Policy of how often and with how many events this method is called is up to the EventService Implementation.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/EventService.idl

9.40 test::Control Interface Reference

Interface to enter and exit special test modes.

```
import "testrpc.idl";
```

Public Member Functions

- boolean [isTestMode](#) ()
returns actual state of test mode
- void [setTestMode](#) (in boolean isTestModeOn)
sets the unit in test mode which pulls away services dealing with feature ports, aux ports, 1wire ports

9.40.1 Detailed Description

Interface to enter and exit special test modes.

Some of the port tests require the unit to be set in dedicated test modes because special equipment is plugged into the ports which would confuse ordinary services observing those ports.

The documentation for this interface was generated from the following file:

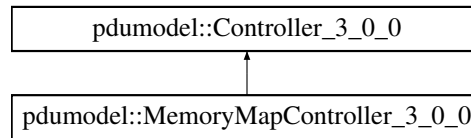
- pdu-json-rpc-api/idl/testrpc.idl

9.41 pdumodel::Controller_3_0_0 Interface Reference

Slave controller interface.

```
import "Controller.idl";
```

Inheritance diagram for pdumodel::Controller_3_0_0:



Classes

- struct [MetaData](#)
Slave controller metadata.

Public Types

- enum [Status](#) {
OK, COMMUNICATION_UNSTABLE, COMMUNICATION_FAILURE, UNKNOWN,
INCOMPATIBLE }
Communication status.
- enum [Type](#) { OUTLET_CTRL, INLET_CTRL, METER_CTRL }
Slave controller type.

Public Member Functions

- [Status](#) [getCommunicationStatus](#) ()
Retrieve the current status of communication with controller.
- [CtrlStatistic](#) [getStatistics](#) ()
Retrieve statistics.
- [MetaData](#) [getMetaData](#) ()
Retrieve the slave controller metadata.

Public Attributes

- valueobject [StatusChangedEvent](#): [idl.Event](#) { [Status](#) oldStatus
Event: Controller communication status has changed.
- [Status](#) [newStatus](#)
Status after change.
- valueobject [MetaDataChangedEvent](#): [idl.Event](#) { [MetaData](#) oldMetaData
Event: Controller metadata has changed.
- [MetaData](#) [newMetaData](#)
Metadata after change.

9.41.1 Detailed Description

Slave controller interface.

9.41.2 Member Enumeration Documentation

9.41.2.1 enum pdumodel::Controller_3_0_0::Status

Communication status.

Enumerator

OK Communication with controller is known to be working.

COMMUNICATION_UNSTABLE Controller can be communicated with sporadically.

COMMUNICATION_FAILURE Controller can't be communicated with.

UNKNOWN Communication status is unknown, e.g. after startup.

INCOMPATIBLE The characteristics of the controller don't meet the expectations.

9.41.2.2 enum pdumodel::Controller_3_0_0::Type

Slave controller type.

Enumerator

OUTLET_CTRL Outlet controller

INLET_CTRL Inlet controller

METER_CTRL General metering controller

9.41.3 Member Function Documentation

9.41.3.1 Status pdumodel::Controller_3_0_0::getCommunicationStatus ()

Retrieve the current status of communication with controller.

Returns

communication status

9.41.3.2 MetaData pdumodel::Controller_3_0_0::getMetaData ()

Retrieve the slave controller metadata.

Returns

Controller metadata

9.41.3.3 CtrlStatistic pdumodel::Controller_3_0_0::getStatistics ()

Retrieve statistics.

Returns

statistics of Controller

9.41.4 Member Data Documentation

9.41.4.1 valueobject pdumodel::Controller_3_0_0::MetaDataChangedEvent

Event: Controller metadata has changed.

Metadata before change

9.41.4.2 valueobject pdumodel::Controller_3_0_0::StatusChangedEvent

Event: Controller communication status has changed.

Status before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Controller.idl

9.42 webcam::Controls Struct Reference

[Controls.](#)

```
import "Webcam.idl";
```

Public Attributes

- int [brightness](#)
brightness
- int [contrast](#)
contrast
- int [saturation](#)
saturation
- int [gain](#)
gain
- int [gamma](#)
gamma
- [PowerLineFrequency](#) [powerLineFrequency](#)
power line frequency (50Hz, 60Hz or disabled)

9.42.1 Detailed Description

[Controls.](#)

All int values are normed to a range of 1 to 1000 0 means "auto" if available, -1 means not supported by the webcam

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Webcam.idl

9.43 pdumodel::CtrlStatistic Struct Reference

Slave controller statistics.

```
import "Controller.idl";
```

Public Attributes

- int [masterCSumErrCnt](#)
Master CRC error counter.
- int [slaveCSumErrCnt](#)
Slave CRC error counter.
- int [timeoutCnt](#)
Master timeout counter.
- int [resetCnt](#)
Controller reset counter.
- int [emResetCnt](#)
Energy meter reset counter.

9.43.1 Detailed Description

Slave controller statistics.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Controller.idl

9.44 fitness::Fitness::DataEntry Struct Reference

An entry in the reliability database.

```
import "Fitness.idl";
```

Public Attributes

- string [id](#)
unique id of the entry
- int [value](#)
normalized value
- int [maxValue](#)
maximum possible normalized value
- int [worstValue](#)
worst normalized value seen yet
- int [thresholdValue](#)
normalized threshold value
- long [rawValue](#)
raw value
- int [flags](#)
flags (see above)

9.44.1 Detailed Description

An entry in the reliability database.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Fitness.idl

9.45 `datetime::DateTime_2_0_0` Interface Reference

Date and time configuration methods.

```
import "DateTime.idl";
```

Classes

- struct [Cfg](#)
Device date and time configuration.
- struct [NtpCfg](#)
NTP server configuration.
- struct [ZoneCfg](#)
Time zone configuration.
- struct [ZoneInfo](#)
Time zone information.

Public Types

- enum [Protocol](#) { [STATIC](#), [NTP](#) }
Time synchronization protocol.

Public Member Functions

- void [getZoneInfos](#) (out vector< [ZoneInfo](#) > zoneInfos, in boolean useOlson)
List all supported time zones.
- boolean [checkNtpServer](#) (in string ntpServer)
Check if a specified NTP server is usable.
- void [getCfg](#) (out [Cfg](#) cfg)
Retrieve the device date and time configuration.
- int [setCfg](#) (in [Cfg](#) cfg)
Set the device date and time configuration.
- void [getTime](#) (in boolean useOlson, out [ZoneInfo](#) zone, out boolean dstEnabled, out int utcOffset, out time currentTime)
Retrieve the current device date and time.

9.45.1 Detailed Description

Date and time configuration methods.

9.45.2 Member Enumeration Documentation

9.45.2.1 `enum datetime::DateTime_2_0_0::Protocol`

Time synchronization protocol.

Enumerator

- STATIC** Device time is configured locally.
- NTP** Device time is synchronized via NTP.

9.45.3 Member Function Documentation

9.45.3.1 `boolean datetime::DateTime_2_0_0::checkNtpServer ( in string ntpServer )`

Check if a specified NTP server is usable.

Parameters

<i>ntpServer</i>	NTP server to be checked
------------------	--------------------------

Returns

`true` if the NTP server is usable

9.45.3.2 `void datetime::DateTime_2_0_0::getCfg ( out Cfg cfg )`

Retrieve the device date and time configuration.

Parameters

<i>cfg</i>	Result: Current date and time configuration
------------	---

9.45.3.3 `void datetime::DateTime_2_0_0::getTime ( in boolean useOlson, out ZoneInfo zone, out boolean dstEnabled, out int utcOffset, out time currentTime )`

Retrieve the current device date and time.

Parameters

<i>useOlson</i>	Use Olson zoneinfo name
<i>zone</i>	Result: Active time zone
<i>dstEnabled</i>	if false, the time zone DST flag is not used
<i>utcOffset</i>	Result: Offset (in minutes) between local time and UTC
<i>currentTime</i>	Result: Device date and time

9.45.3.4 `void datetime::DateTime_2_0_0::getZoneInfos ( out vector< ZoneInfo > zoneInfos, in boolean useOlson )`

List all supported time zones.

Parameters

<i>zoneInfos</i>	Result: List of time zones
<i>useOlson</i>	Use Olson zoneinfo names

9.45.3.5 `int datetime::DateTime_2_0_0::setCfg ( in Cfg cfg )`

Set the device date and time configuration.

Depending on the value of the *protocol* field either *deviceTime* or *ntpCfg* will be used from the *cfg* parameter.

Parameters

<i>cfg</i>	New date and time configuration.
------------	----------------------------------

Returns

- 0 if OK
- 1 if the configuration is invalid

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/DateTime.idl

9.46 logging::DebugLog Interface Reference

Device debug log interface.

```
import "DebugLog.idl";
```

Public Member Functions

- void [clear](#) ()
Clear the debug log.
- int [getFirstId](#) ()
Get the id of the first debug log entry.
- int [getLastId](#) ()
Get the id of the last debug log entry.
- void [getEntries](#) (out vector< [LogEntry](#) > *entries*, in int *refId*, in int *count*, in [RangeDirection](#) *direction*)
Fetch entries from the debug log.

9.46.1 Detailed Description

Device debug log interface.

9.46.2 Member Function Documentation

9.46.2.1 void logging::DebugLog::getEntries (out vector< [LogEntry](#) > *entries*, in int *refId*, in int *count*, in [RangeDirection](#) *direction*)

Fetch entries from the debug log.

Parameters

<i>entries</i>	Result: List of log entries
<i>refId</i>	First log id to fetch
<i>count</i>	Number of entries to fetch
<i>direction</i>	Range direction

9.46.2.2 int logging::DebugLog::getFirstId ()

Get the id of the first debug log entry.

Returns

Id of first log entry

9.46.2.3 int logging::DebugLog::getLastId ()

Get the id of the last debug log entry.

Returns

Id of last log entry.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/DebugLog.idl

9.47 portsmodel::Port_2_0_1::DetectionMode Struct Reference

Port detection mode.

```
import "Port.idl";
```

Public Attributes

- [DetectionType](#) type
detection type: auto or pinned
- string [pinnedDeviceType](#)
contains specific device type in pinned mode, not used for auto

9.47.1 Detailed Description

Port detection mode.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Port.idl

9.48 peripheral::DeviceID_2_0_0 Struct Reference

peripheral device identification

```
import "PeripheralDeviceSlot.idl";
```

Public Attributes

- string [serial](#)
Serial number.
- sensors Sensor_4_0_0 TypeSpec [type](#)
device's type spec
- boolean [isActuator](#)
true if device is an actuator
- int [channel](#)
Channel number.

9.48.1 Detailed Description

peripheral device identification

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceSlot.idl

9.49 assetmgrmodel::AssetStrip_2_0_1::DeviceInfo Struct Reference

Static (type, version) information for an AssetStrip.

```
import "AssetStrip.idl";
```

Public Attributes

- int [deviceId](#)
Device type (indicated a certain hardware)
- int [hardwareId](#)
Hardware ID, revision.
- int [protocolVersion](#)
Protocol version the strip is supporting.
- int [bootVersion](#)
Bootcode software version.
- int [appVersion](#)
Application code software version.
- boolean [orientationSensAvailable](#)
Indicates whether the strip has an orientation sensor.
- boolean [isCascadable](#)
The asset strip type.
- boolean [rackUnitCountConfigurable](#)
Rack unit count has to be configured, i.e. is not auto detected.

9.49.1 Detailed Description

Static (type, version) information for an AssetStrip.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStrip.idl

9.50 peripheral::DeviceManager_2_0_0 Interface Reference

Peripheral Device Manager.

```
import "PeripheralDeviceManager.idl";
```

Classes

- struct [DeviceTypeInfo](#)
Peripheral device type info.
- struct [MetaData](#)
Peripheral DeviceManager's metadata.
- struct [Settings](#)
peripheral DeviceManager's s settings
- struct [Statistics](#)
Peripheral device statistics.

Public Types

- enum [ZCoordMode](#) { [RACKUNITS](#), [FREEFORM](#) }
Z Coordinate Mode identifier.
- enum [DeviceFirmwareUpdateState](#) { [UPDATE_STARTED](#), [UPDATE_SUCCESSFUL](#), [UPDATE_FAILED](#) }
Enumeration: State of device firmware update.

Public Member Functions

- vector< [DeviceSlot_2_0_0](#) > [getDeviceSlots](#) ()
Get the list of peripheral device slots.
- [DeviceSlot_2_0_0](#) [getDeviceSlot](#) (in int idx)
Get a DeviceSlot by its index.
- vector< [Device_2_0_0](#) > [getDiscoveredDevices](#) ()
Get the list of currently attached peripheral devices.
- vector< [PackageInfo_2_0_0](#) > [getDiscoveredPackageInfos](#) ()
Get the list of currently attached peripheral device packages.
- [Settings](#) [getSettings](#) ()
Retrieve the peripheral DeviceManager's settings.
- int [setSettings](#) (in [Settings](#) settings)
Change the peripheral DeviceManager's settings.
- [MetaData](#) [getMetaData](#) ()
Retreive the Peripheral DeviceManager's metadata.
- vector< [DeviceTypeInfo](#) > [getDeviceTypeInfos](#) ()
Get the list of all peripheral device type infos.
- [Statistics](#) [getStatistics](#) ()
Retrieve statistics.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.
- valueobject [SettingsChangedEvent](#): [event.UserEvent](#) { [Settings](#) oldSettings
Event: Peripheral device manager's settings have been changed.
- [Settings](#) [newSettings](#)
[Settings](#) after change.
- valueobject [DeviceEvent](#): [idl.Event](#) { [Device_2_0_0](#) device
Event: A peripheral device was added or removed.
- vector< [Device_2_0_0](#) > [allDevices](#)

- New list of discovered devices after change.*

 - valueobject [DeviceAddedEvent](#): [DeviceEvent](#) { }
 - Event: A peripheral device was added.*
 - valueobject [DeviceRemovedEvent](#): [DeviceEvent](#) { }
 - Event: A peripheral device was removed.*
 - valueobject [UnknownDeviceAttachedEvent](#): [idl.Event](#) { string romCode
 - Event: An unknown device was attached.*
 - vector< [PosElement](#) > [position](#)
 - Device position in the chain.*
 - valueobject [DeviceFirmwareUpdateStateChangedEvent](#): [idl.Event](#) { string oldVersion
 - Event: Firmware update on a device was started or has finished.*
 - string [newVersion](#)
 - Firmware version to be updated to.*
 - string [serial](#)
 - Serial number of device.*
 - [DeviceFirmwareUpdateState](#) state
 - Update state.*
 - valueobject [PackageEvent](#): [idl.Event](#) { [PackageInfo_2_0_0](#) packageInfo
 - Event: A peripheral device package was added or removed.*
 - vector< [PackageInfo_2_0_0](#) > [allPackages](#)
 - New list of discovered packages after change.*
 - valueobject [PackageAddedEvent](#): [PackageEvent](#) { }
 - Event: A peripheral device package was added.*
 - valueobject [PackageRemovedEvent](#): [PackageEvent](#) { }
 - Event: A peripheral device package was removed.*

9.50.1 Detailed Description

Peripheral Device Manager.

9.50.2 Member Enumeration Documentation

9.50.2.1 enum peripheral::DeviceManager_2_0_0::DeviceFirmwareUpdateState

Enumeration: State of device firmware update.

Enumerator

- UPDATE_STARTED** Update is running.
- UPDATE_SUCCESSFUL** Update has finished successfully.
- UPDATE_FAILED** Update has failed.

9.50.2.2 enum peripheral::DeviceManager_2_0_0::ZCoordMode

Z Coordinate Mode identifier.

Enumerator

- RACKUNITS** Z coordinate of slot settings is in rack units.
- FREEFORM** Z coordinate of slot settings is free form text.

9.50.3 Member Function Documentation

9.50.3.1 DeviceSlot_2_0_0 peripheral::DeviceManager_2_0_0::getDeviceSlot (in int *idx*)

Get a DeviceSlot by its index.

Parameters

<i>idx</i>	index of the slot to get
------------	--------------------------

Returns

the requested slot

9.50.3.2 vector<DeviceSlot_2_0_0> peripheral::DeviceManager_2_0_0::getDeviceSlots ()

Get the list of peripheral device slots.

Returns

List of peripheral device slots

9.50.3.3 vector<DeviceTypeInfo> peripheral::DeviceManager_2_0_0::getDeviceTypeInfos ()

Get the list of all peripheral device type infos.

Returns

List of all peripheral device type infos

9.50.3.4 vector<Device_2_0_0> peripheral::DeviceManager_2_0_0::getDiscoveredDevices ()

Get the list of currently attached peripheral devices.

Returns

List of all discovered peripheral devices

9.50.3.5 vector<PackageInfo_2_0_0> peripheral::DeviceManager_2_0_0::getDiscoveredPackageInfos ()

Get the list of currently attached peripheral device packages.

Returns

List of all discovered peripheral device packages

9.50.3.6 MetaData peripheral::DeviceManager_2_0_0::getMetaData ()

Retreive the Peripheral DeviceManager's metadata.

Returns

Peripheral DeviceManager's metadata

9.50.3.7 Settings peripheral::DeviceManager_2_0_0::getSettings ()

Retrieve the peripheral DeviceManager's settings.

Returns

peripheral DeviceManager's settings

9.50.3.8 Statistics peripheral::DeviceManager_2_0_0::getStatistics ()

Retrieve statistics.

Returns

peripheral device statistics

9.50.3.9 int peripheral::DeviceManager_2_0_0::setSettings (in Settings settings)

Change the peripheral DeviceManager's settings.

Parameters

<i>settings</i>	New peripheral DeviceManager's settings
-----------------	---

Returns

0 if OK
ERR_INVALID_PARAMS if any parameters are invalid

9.50.4 Member Data Documentation

9.50.4.1 valueobject peripheral::DeviceManager_2_0_0::DeviceEvent

Event: A peripheral device was added or removed.

Affected device

9.50.4.2 valueobject peripheral::DeviceManager_2_0_0::DeviceFirmwareUpdateStateChangedEvent

Event: Firmware update on a device was started or has finished.

Firmware version before update

9.50.4.3 valueobject peripheral::DeviceManager_2_0_0::PackageEvent

Event: A peripheral device package was added or removed.

Information about affected package

9.50.4.4 valueobject peripheral::DeviceManager_2_0_0::SettingsChangedEvent

Event: Peripheral device manager's settings have been changed.

[Settings](#) before change

9.50.4.5 valueobject peripheral::DeviceManager_2_0_0::UnknownDeviceAttachedEvent

Event: An unknown device was attached.

Device ROM code

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceManager.idl

9.51 peripheral::DeviceSlot_2_0_0 Interface Reference

Peripheral Device Slot.

```
import "PeripheralDeviceSlot.idl";
```

Classes

- struct [Location](#)
user writeable location
- struct [Settings](#)
user configurable slot attributes

Public Member Functions

- [Device_2_0_0](#) [getDevice](#) ()
Returns the actual device reference.
- int [assign](#) (in [DeviceID_2_0_0](#) devid)
Associate this slot with a given (old or detected new) peripheral device.
- int [assignAddress](#) (in string [packageClass](#), in [Address_2_0_0](#) address)
Associate this slot with an addressable (new) peripheral device.
- int [unassign](#) ()
Break the association for this slot.
- [Settings](#) [getSettings](#) ()
Retrieve the user-defined settings.
- int [setSettings](#) (in [Settings](#) settings)
Change the slot settings.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.
- constant int [ERR_NOT_SUPPORTED](#) = 2
Operation not supported.
- constant int [CHANNEL_INVALID](#) = -1
Device has no channels.
- valueobject [DeviceChangedEvent](#): [idl.Event](#) { [Device_2_0_0](#) oldDevice
Event: The device attached to this slot has changed.
- [Device_2_0_0](#) [newDevice](#)
Device after change.
- valueobject [SettingsChangedEvent](#): [event.UserEvent](#) { [Settings](#) oldSettings
Event: The slot settings have been changed.
- [Settings](#) [newSettings](#)
Settings after change.

9.51.1 Detailed Description

Peripheral Device Slot.

9.51.2 Member Function Documentation

9.51.2.1 `int peripheral::DeviceSlot_2_0_0::assign ( in DeviceID_2_0_0 devid )`

Associate this slot with a given (old or detected new) peripheral device.

Parameters

<i>devid</i>	peripheral device identification
--------------	----------------------------------

Returns

0 if OK
ERR_INVALID_PARAMS if *devid* is unknown or invalid

9.51.2.2 `int peripheral::DeviceSlot_2_0_0::assignAddress ( in string packageClass, in Address_2_0_0 address )`

Associate this slot with an addressable (new) peripheral device.

Parameters

<i>address</i>	peripheral device address
----------------	---------------------------

Returns

0 if OK
ERR_INVALID_PARAMS if *address* is invalid

9.51.2.3 `Device_2_0_0 peripheral::DeviceSlot_2_0_0::getDevice ( )`

Returns the actual device reference.

The reference becomes invalid due to assign/unassign method call. This conditions is also flagged by EVT_KEY_-DEVICE_CHANGED event

9.51.2.4 `Settings peripheral::DeviceSlot_2_0_0::getSettings ( )`

Retrieve the user-defined settings.

Returns

Slot settings

9.51.2.5 `int peripheral::DeviceSlot_2_0_0::setSettings ( in Settings settings )`

Change the slot settings.

Parameters

<i>settings</i>	New slot settings
-----------------	-------------------

Returns

0 if OK
 ERR_INVALID_PARAMS if any parameters are invalid

9.51.2.6 `int peripheral::DeviceSlot_2_0_0::unassign ( )`

Break the association for this slot.

Returns

0 if OK
 ERR_NOT_SUPPORTED if operation is not supported this is the case for sensors with complete position information

9.51.3 Member Data Documentation**9.51.3.1** `valueobject peripheral::DeviceSlot_2_0_0::DeviceChangedEvent`

Event: The device attached to this slot has changed.

Device before change

9.51.3.2 `valueobject peripheral::DeviceSlot_2_0_0::SettingsChangedEvent`

Event: The slot settings have been changed.

[Settings](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceSlot.idl

9.52 `peripheral::DeviceManager_2_0_0::DeviceTypeInfo` Struct Reference

Peripheral device type info.

```
import "PeripheralDeviceManager.idl";
```

Public Attributes

- sensors Sensor_4_0_0 TypeSpec [type](#)
Device (sensor) type.
- boolean [isActuator](#)
Is actuator or not.
- string [identifier](#)
Device type identifier.
- string [name](#)
Device type display name.
- sensors NumericSensor_4_0_0 Range [defaultRange](#)
Default sensor range (numeric sensors only)
- int [defaultDecDigits](#)
Default sensor precision (numeric sensors only)

9.52.1 Detailed Description

Peripheral device type info.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceManager.idl

9.53 diag::DiagLogSettings Interface Reference

Diagnostic log settings.

```
import "DiagLogSettings.idl";
```

Classes

- struct [LogLevelEntry](#)
An entry containing a context name and its associated context.

Public Types

- enum [LogLevel](#) {
LOG_LEVEL_NONE, LOG_LEVEL_ERR, LOG_LEVEL_WARN, LOG_LEVEL_INFO,
LOG_LEVEL_DEBUG, LOG_LEVEL_TRACE }
Log levels.

Public Member Functions

- void [resetLogLevelsForAllCtxNames](#) ()
Reset log levels of all contexts to default.
- vector< [LogLevelEntry](#) > [getLogLevelsForAllCtxNames](#) ()
Get log levels of all contexts.
- int [setLogLevelForAllCtxNames](#) (in [LogLevel](#) logLevel)
Set the log level for all contexts at once.
- int [getLogLevelByCtxName](#) (in string ctxName, out [LogLevel](#) logLevel)
Get the log level for a specific context.
- int [setLogLevelByCtxName](#) (in string ctxName, in [LogLevel](#) logLevel)
Set the log level for a specific context.

Public Attributes

- constant int [ERR_NONE](#) = 0
no error
- constant int [ERR_UNKNOWN_LOG_CONTEXT_NAME](#) = 1
unknown log context
- constant int [ERR_UNKNOWN_LOG_LEVEL](#) = 2
unknown log level

9.53.1 Detailed Description

Diagnostic log settings.

9.53.2 Member Enumeration Documentation

9.53.2.1 enum diag::DiagLogSettings::LogLevel

Log levels.

Enumerator

LOG_LEVEL_NONE no log messages
LOG_LEVEL_ERR errors
LOG_LEVEL_WARN warnings + errors
LOG_LEVEL_INFO info + warnings + errors
LOG_LEVEL_DEBUG debug + info + warnings + errors
LOG_LEVEL_TRACE trace + debug + info + warnings + errors

9.53.3 Member Function Documentation

9.53.3.1 int diag::DiagLogSettings::getLogLevelByCtxName (in string *ctxName*, out LogLevel *logLevel*)

Get the log level for a specific context.

Parameters

<i>ctxName</i>	- context name
<i>logLevel</i>	- result: log level

Returns

ERR_NONE on success
 ERR_UNKNOWN_LOG_CONTEXT_NAME if the context is unknown
 ERR_UNKNOWN_LOG_LEVEL if log level cannot be mapped to JSON

9.53.3.2 vector<LogLevelEntry> diag::DiagLogSettings::getLogLevelsForAllCtxNames ()

Get log levels of all contexts.

Returns

vector of [LogLevelEntry](#) structures

9.53.3.3 int diag::DiagLogSettings::setLogLevelByCtxName (in string *ctxName*, in LogLevel *logLevel*)

Set the log level for a specific context.

Parameters

<i>ctxName</i>	- context name
<i>logLevel</i>	- log level

Returns

ERR_NONE on success
 ERR_UNKNOWN_LOG_CONTEXT_NAME if the context is unknown
 ERR_UNKNOWN_LOG_LEVEL if log level is unknown

9.53.3.4 `int diag::DiagLogSettings::setLogLevelForAllCtxNames ( in LogLevel logLevel )`

Set the log level for all contexts at once.

Parameters

<i>logLevel</i>	- log level
-----------------	-------------

Returns

ERR_NONE on success
ERR_UNKNOWN_LOG_LEVEL if log level is unknown

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/DiagLogSettings.idl

9.54 net::Diagnostics Interface Reference

[Diagnostics](#) interface.

```
import "Diagnostics.idl";
```

Public Member Functions

- `int ping` (in string *hostName*, in int *count*, out vector< string > *results*)
Ping a network host (send out ICMP echo requests)
- `int traceRoute` (in string *hostName*, in int *timeout*, in boolean *useIcmp*, out vector< string > *results*)
Get the route packet trace to a network host.
- `int listTcpConnections` (out vector< string > *results*)
List the currently active TCP connections (netstat -ta) (lists both listening as well as established connections)

Public Attributes

- constant int `NO_ERROR` = 0
No error.
- constant int `ERR_INVALID_PARAM` = 1
Invalid parameters.
- constant int `ERR_EXEC_FAIL` = 2
Error during execution.
- constant int `ERR_TIMEOUT` = 3
Timeout.
- constant int `ERR_RESOLVE_FAIL` = 4
Name resolution failure.

9.54.1 Detailed Description

[Diagnostics](#) interface.

9.54.2 Member Function Documentation

9.54.2.1 `int net::Diagnostics::listTcpConnections ( out vector< string > results )`

List the currently active TCP connections (netstat -ta) (lists both listening as well as established connections)

Returns

NO_ERROR if netstat was successful
ERR_EXEC_FAIL if there was an error during netstat execution

9.54.2.2 `int net::Diagnostics::ping ( in string hostName, in int count, out vector< string > results )`

Ping a network host (send out ICMP echo requests)

Parameters

<i>hostName</i>	host that should be pinged
<i>count</i>	number of echo requests that should be sent (up to 20)
<i>results</i>	output of the ping command

Returns

NO_ERROR if ping command was successful
ERR_INVALID_PARAM if any parameters were invalid
ERR_EXEC_FAIL if there was an error during ping execution
ERR_RESOLVE_FAIL if the host name could not be resolved

9.54.2.3 `int net::Diagnostics::traceRoute ( in string hostName, in int timeout, in boolean useIcmp, out vector< string > results )`

Get the route packet trace to a network host.

Parameters

<i>hostName</i>	destination host to track
<i>timeout</i>	Timeout (in seconds) to wait for traceroute results (up to 900)
<i>useIcmp</i>	use ICMP packets instead of UDP packets
<i>results</i>	trace output

Returns

NO_ERROR if traceroute was successful
ERR_INVALID_PARAM if any parameters were invalid
ERR_EXEC_FAIL if there was an error during traceroute execution
ERR_TIMEOUT if traceroute didn't finish before timeout elapsed
ERR_RESOLVE_FAIL if the host name could not be resolved

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Diagnostics.idl

9.55 test::Display_1_0_1 Interface Reference

Type-independent display test interface.


```
import "TestDisplay.idl";
```

Classes

- struct [Info](#)
Collected display meta information.

Public Types

- enum [Orientation](#) { [NORMAL](#), [FLIPPED](#), [LEFT](#), [RIGHT](#) }
Display orientation.
- enum [TestStatus](#) { [TEST_IDLE](#), [TEST_BUSY](#), [TEST_PASSED](#), [TEST_FAILED](#) }
Status of interactive test.

Public Member Functions

- [Info](#) [getInfo](#) ()
Retrieve display's meta information.
- void [testSequence](#) (in int cycleTime_ms)
Execute a test sequence that is a bit more elaborate than just on and off all.
- void [enterTestMode](#) ()
Start an interactive test on units which support it.
- [TestStatus](#) [getTestStatus](#) ()
Get the current status of the interactive test.

9.55.1 Detailed Description

Type-independent display test interface.

9.55.2 Member Enumeration Documentation

9.55.2.1 enum test::Display_1_0_1::Orientation

Display orientation.

Enumerator

NORMAL Normal orientation.
FLIPPED Upside-down.
LEFT Left side down.
RIGHT Right side down.

9.55.2.2 enum test::Display_1_0_1::TestStatus

Status of interactive test.

Enumerator

TEST_IDLE Test has not been started.
TEST_BUSY Test is in progress.
TEST_PASSED The test finished successfully.
TEST_FAILED The test failed.

9.55.3 Member Function Documentation

9.55.3.1 Info test::Display_1_0_1::getInfo ()

Retrieve display's meta information.

Returns

Display information

9.55.3.2 TestStatus test::Display_1_0_1::getTestStatus ()

Get the current status of the interactive test.

Returns

Test status

9.55.3.3 void test::Display_1_0_1::testSequence (in int *cycleTime_ms*)

Execute a test sequence that is a bit more elaborate than just on and off all.

Parameters

<i>cycleTime_ms</i>	Delay between state changes in ms
---------------------	-----------------------------------

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/TestDisplay.idl

9.56 pdumodel::DoublePole_2_0_0 Struct Reference

for OCP

```
import "Pole.idl";
```

Public Attributes

- string [label](#)
Pole label
- [PowerLine](#) [line](#)
Power line.
- int [inNodeId](#)
Upstream (inlet-side) circuit node id.
- int [outNodeId](#)
Downstream (outlet-side) circuit node id.
- sensors NumericSensor_4_0_0 [voltage](#)
RMS voltage sensor, L-L.
- sensors NumericSensor_4_0_0 [voltageLN](#)
RMS voltage sensor, L-N.
- sensors NumericSensor_4_0_0 [current](#)
RMS current sensor.

- sensors NumericSensor_4_0_0 [peakCurrent](#)
Peak current sensor.
- sensors NumericSensor_4_0_0 [activePower](#)
Active power sensor.
- sensors NumericSensor_4_0_0 [apparentPower](#)
Apparent power sensor.
- sensors NumericSensor_4_0_0 [powerFactor](#)
Power factor sensor.
- sensors NumericSensor_4_0_0 [activeEnergy](#)
Active energy sensor.
- sensors NumericSensor_4_0_0 [apparentEnergy](#)
Apparent energy sensor.

9.56.1 Detailed Description

for OCP

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Pole.idl

9.57 net::EapSettings Struct Reference

EAP authentication settings.

```
import "Net.idl";
```

Public Attributes

- string [identity](#)
EAP identity.
- string [password](#)
EAP password.
- [EapOuterMethod](#) [outerMethod](#)
Outer authentication method.
- [EapInnerMethod](#) [innerMethod](#)
Inner authentication method.
- string [caCertificate](#)
CA certificate.

9.57.1 Detailed Description

EAP authentication settings.

The documentation for this struct was generated from the following file:

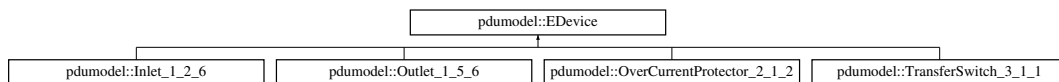
- pdu-json-rpc-api/idl/Net.idl

9.58 pdumodel::EDevice Interface Reference

Common base interface for any kind of electrical device that is used in the PDU model, such as inlets, OCPs and outlets.

```
import "EDevice.idl";
```

Inheritance diagram for pdumodel::EDevice:



Public Member Functions

- `vector< EDevice > getParents ()`
Get the list of devices that provide energy to this device.
- `vector< EDevice > getChildren ()`
Get the list of devices that are directly fed by this device.

9.58.1 Detailed Description

Common base interface for any kind of electrical device that is used in the PDU model, such as inlets, OCPs and outlets.

EDevices form a hierarchy of parent-child relationships. An [EDevice](#) is defined to be the parent of another if it "provides energy" to the latter. E.g. an inlet could be the parent of a number of OCPs, wires or outlets.

An [EDevice](#) can have multiple parents, e.g. in case of transfer switches which select power from multiple sources.

9.58.2 Member Function Documentation

9.58.2.1 `vector<EDevice> pdumodel::EDevice::getChildren ( )`

Get the list of devices that are directly fed by this device.

Returns

List of child devices

9.58.2.2 `vector<EDevice> pdumodel::EDevice::getParents ( )`

Get the list of devices that provide energy to this device.

Returns

List of parent devices

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/EDevice.idl

9.59 cew::EnergyWiseManager Interface Reference

EnergyWise manager.

```
import "EnergyWiseManager.idl";
```

Public Member Functions

- [EnergyWiseSettings](#) `getSettings ()`
Retrieve [EnergyWiseSettings](#).
- int `setSettings` (in [EnergyWiseSettings](#) settings)
Set settings (domain level)

9.59.1 Detailed Description

EnergyWise manager.

9.59.2 Member Function Documentation

9.59.2.1 [EnergyWiseSettings](#) `cew::EnergyWiseManager::getSettings ( )`

Retrieve [EnergyWiseSettings](#).

Returns

[EnergyWiseSettings](#)

9.59.2.2 int `cew::EnergyWiseManager::setSettings ( in EnergyWiseSettings settings )`

Set settings (domain level)

Returns

- 0 if OK
- 1 if invalid param

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/EnergyWiseManager.idl

9.60 cew::EnergyWiseSettings Struct Reference

EnergyWise settings.

```
import "EnergyWiseSettings.idl";
```

Public Attributes

- boolean `enabled`
Enabled.
- string `domainName`
Domain name.

- string [secret](#)
Secret.
- int [port](#)
Port.
- int [pollingInterval](#)
Polling interval.

9.60.1 Detailed Description

EnergyWise settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EnergyWiseSettings.idl

9.61 event::Engine Interface Reference

There is a single event engine instance reachable by a well known reference.

```
import "EventEngine.idl";
```

Classes

- struct [Action](#)
An action is a tuple of 'id' (unique within the scope of this event engine), 'name' which is unique as well and used by the GUI as user readable identifier, 'isSystem' which denotes the action as system action, 'type' which defines what it is, and an argument vector that vary depending on type and which is passed to the action.
- struct [Condition](#)
[Condition](#) is a logical combination of multiple events.
- struct [EventDesc](#)
An event descriptor.
- struct [Rule](#)
A [Rule](#) binds an action to a condition.

Public Member Functions

- int [listEventDescs](#) (in vector< string > eventIdPrefix, out vector< [EventDesc](#) > eventDescs)
Query existing event descriptors.
- vector< string > [listActionTypes](#) ()
List all available action types.
- int [addAction](#) (in [Action](#) action, out string actionId)
Add a new action.
- int [modifyAction](#) (in [Action](#) action)
Modify an action.
- int [deleteAction](#) (in string actionId)
Remove an action.
- vector< [Action](#) > [listActions](#) ()
List all actions currently know to this event engine.
- int [triggerAction](#) (in string actionId, out string errMsg, in vector< [KeyValue](#) > context)
Trigger an action.
- int [addRule](#) (in [Rule](#) rule, out string ruleId)

- Add a new rule.*
 - int `modifyRule` (in `Rule` rule)
- Modify a rule.*
 - int `enableRule` (in string ruleId)
- Enable a rule.*
 - int `disableRule` (in string ruleId)
- Disable a rule.*
 - int `deleteRule` (in string ruleId)
- Delete a rule.*
 - vector< `Rule` > `listRules` ()
- List all rules.*
 - int `deliverEvent` (in `Event` event)
- Deliver an event.*
 - int `rearmRule` (in string ruleId)
- Rearm an event rule that is active.*

9.61.1 Detailed Description

There is a single event engine instance reachable by a well known reference.

9.61.2 Member Function Documentation

9.61.2.1 int event::Engine::addAction (in Action action, out string actionId)

Add a new action.

The id and isSystem fields are ignored. The actions's id field is allocated automatically and returned in the actionId parameter.

Returns

- 0 if OK
- 1 if the name of the new action already exists
- 2 if generating the action id failed
- 3 if the maximum number of actions have been created

9.61.2.2 int event::Engine::addRule (in Rule rule, out string ruleId)

Add a new rule.

The id, hasMatched and isSystem fields are ignored. The rule's id field is allocated automatically and returned in the the ruleId parameter.

Returns

- 0 if OK
- 1 if the name of the new rule already exists
- 2 if generating the rule id failed
- 3 if the rule condition contains an invalid event id
- 4 if the maximum number of rules have been created
- 5 if the maximum number of actions per rule is exceeded

9.61.2.3 `int event::Engine::deleteAction ( in string actionId )`

Remove an action.

Returns

- 0 if OK
- 1 if id is unknown
- 2 if the action is not deletable

9.61.2.4 `int event::Engine::deleteRule ( in string ruleId )`

Delete a rule.

Returns

- 0 if OK
- 1 if ruleId is unknown
- 2 if the rule is not deletable

9.61.2.5 `int event::Engine::deliverEvent ( in Event event )`

Deliver an event.

Returns

- 0 if OK
- 1 if event is unknown

9.61.2.6 `int event::Engine::disableRule ( in string ruleId )`

Disable a rule.

A disabled rule will be ignored when processing events.

Returns

- 0 if OK
- 1 if ruleId is unknown

9.61.2.7 `int event::Engine::enableRule ( in string ruleId )`

Enable a rule.

An enabled rule will be evaluated when processing events.

Returns

- 0 if OK
- 1 if id is unknown

9.61.2.8 `vector<string> event::Engine::listActionTypes ( )`

List all available action types.

The action type is intentionally defined generically so that implementation can be extensible without changing interface.

9.61.2.9 `int event::Engine::listEventDescs ( in vector< string > eventIdPrefix, out vector< EventDesc > eventDescs )`

Query existing event descriptors.

Returns

- 0 if OK
- 1 if id was not found

9.61.2.10 `vector<Rule> event::Engine::listRules ( )`

List all rules.

Question: will number of rules be small enough to list them all in a single operation. A single [Rule](#) may be quite large... See below for alternative interface using iterator.

9.61.2.11 `int event::Engine::modifyAction ( in Action action )`

Modify an action.

Returns

- 0 if OK
- 1 if the name of the new action already exists
- 2 if the action id does not exist

9.61.2.12 `int event::Engine::modifyRule ( in Rule rule )`

Modify a rule.

The `hasMatched` and `isSystem` fields are ignored.

Returns

- 0 if OK
- 1 if the name of the new rule already exists
- 2 if the rule id does not exist
- 3 if the rule condition contains an invalid event id
- 4 if the maximum number of actions per rule is exceeded

9.61.2.13 `int event::Engine::rearmRule ( in string ruleId )`

Rearm an event rule that is active.

Once a [Rule](#) triggers its actions it is done once. If the condition of the rules becomes false and back again true, no action will be performed unless the rule is rearmed. Rules may be auto-rearmed what means whenever the rule's conditions changes from false to true actions are triggered over and again.

Returns

- 0 if OK
- 1 if the rule is unknown
- 2 if the rule is not enabled

9.61.2.14 `int event::Engine::triggerAction ( in string actionId, out string errMsg, in vector< KeyValue > context )`

Trigger an action.

This is mainly used for test and debugging. The operation will block until the action has been performed or an error occurred. In the latter case `errMsg` will contain an error message. The context is passed to the actions.

NOTE: Currently this function doesn't block!

Returns

- 0 if OK
- 1 if id is unknown
- 2 if no actor was found for the action
- 3 if performing the action failed, `errMsg` will be set

The documentation for this interface was generated from the following file:

- `pdu-json-rpc-api/idl/EventEngine.idl`

9.62 `res_mon::Entry` Struct Reference

[ResMon Entry](#).

```
import "ResMon.idl";
```

Public Types

- enum [Type](#) {
[GLOBAL_CPU_USAGE](#), [GLOBAL_FREE_MEM](#), [GLOBAL_PROC_COUNT](#), [FS_FREE_SPACE](#),
[FS_FREE_INODES](#), [PROC_CPU_USAGE](#), [PROC_VM_SIZE](#), [PROC_FREE_FILE_DESC](#),
[PROC_LIFE_TIME](#), [PROC_COUNT](#) }

Type of this [ResMon Entry](#).

Public Attributes

- [Type](#) `type`
Type of this [ResMon Entry](#).
- string `name`
[ResMon Entry](#) name.
- long `value`
[ResMon Entry](#) value.

9.62.1 Detailed Description

[ResMon Entry](#).

9.62.2 Member Enumeration Documentation

9.62.2.1 `enum res_mon::Entry::Type`

Type of this [ResMon Entry](#).

Enumerator

GLOBAL_CPU_USAGE global cpu usage
GLOBAL_FREE_MEM global free memory
GLOBAL_PROC_COUNT global process count
FS_FREE_SPACE free filesystem space
FS_FREE_INODES free filesystem inodes
PROC_CPU_USAGE process cpu usage
PROC_VM_SIZE process virtual mem size
PROC_FREE_FILE_DESC process free file descriptors
PROC_LIFE_TIME process life time
PROC_COUNT process count

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ResMon.idl

9.63 fitness::Fitness::ErrorLogEntry Struct Reference

An entry in the reliability error log.

```
import "Fitness.idl";
```

Public Attributes

- string [id](#)
id of the associated data entry
- int [value](#)
normalized value
- int [thresholdValue](#)
normalized threshold value
- long [rawValue](#)
raw value
- int [powerOnHours](#)
power on hours when error occurred
- time [timeStampUTC](#)
UTC time stamp when error occurred.

9.63.1 Detailed Description

An entry in the reliability error log.

An error log entry is made when the normalized value of a data entry drops below the normalized threshold value.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Fitness.idl

9.64 test::Ethernet Interface Reference

test routines for RJ45 [Ethernet](#) port This is low level interface using ethtool that does not persist any of the settings made (TODO: this interface may be combined with a 'decent' network interface.

```
import "testrpc.idl";
```

Public Types

- enum [Speed](#) { [SPD_10](#), [SPD_100](#), [SPD_1000](#) }
Ethernet Speed.
- enum [Duplex](#) { [DPX_HALF](#), [DPX_FULL](#) }
Ethernet Duplex Mode.

Public Member Functions

- `vector< string > getDevices ()`
returns a a list of valid network interface devices
- `int setParameters (in string device, in Speed speed, in Duplex duplex, in boolean isAutoNeg)`
transiently sets the interface to the desired parameters

9.64.1 Detailed Description

test routines for RJ45 [Ethernet](#) port This is low level interface using ethtool that does not persist any of the settings made (TODO: this interface may be combined with a 'decent' network interface.

Currently the network interface is dedicated to special use cases)

9.64.2 Member Enumeration Documentation

9.64.2.1 enum test::Ethernet::Duplex

[Ethernet](#) Duplex Mode.

Enumerator

[DPX_HALF](#) Half Duplex.
[DPX_FULL](#) Full Duplex.

9.64.2.2 enum test::Ethernet::Speed

[Ethernet](#) Speed.

Enumerator

[SPD_10](#) 10 Mbit/s
[SPD_100](#) 100 Mbit/s
[SPD_1000](#) 1 Gbit/s

9.64.3 Member Function Documentation

9.64.3.1 int test::Ethernet::setParameters (in string *device*, in [Speed](#) *speed*, in [Duplex](#) *duplex*, in boolean *isAutoNeg*)

transiently sets the interface to the desired parameters

Returns

- 0 if OK
- 1 if invalid param
- 2 if system error executing operation

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/testrpc.idl

9.65 event::Event Struct Reference

[Event](#) has a type: a STATE event indicates that a boolean state has been changed, i.e.

```
import "EventEngine.idl";
```

Public Types

- enum [Type](#) { [STATE](#), [TRIGGER](#) }
- Event type.*

Public Attributes

- [Type](#) type
- Event type.*
- vector< string > [id](#)
- Event id vector.*
- boolean [asserted](#)
- Assertion value.*
- time [timeStamp](#)
- Timestamp.*
- vector< [KeyValue](#) > [context](#)
- Context map.*

9.65.1 Detailed Description

[Event](#) has a type: a STATE event indicates that a boolean state has been changed, i.e.

asserted or deasserted a TRIGGER event is one that has no state assigned. conceptually it is asserted and deasserted at once. The id has multiple components that form a path into a hierarchy. The value ("asserted") indicates whether the state has become true (assertion) or false (deassertion). For events of type TRIGGER this will be true always.

9.65.2 Member Enumeration Documentation

9.65.2.1 enum event::Event::Type

[Event](#) type.

Enumerator

- STATE** State event.
- TRIGGER** Trigger event.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EventEngine.idl

9.66 event::Engine::EventDesc Struct Reference

An event descriptor.

```
import "EventEngine.idl";
```

Public Types

- enum [Type](#) { [NODE](#), [DYN_NODE](#), [LEAF](#) }
[Event](#) descriptor type.

Public Attributes

- [Type](#) eventDescType
[Event](#) descriptor type.
- [Event](#) Type eventType
[Event](#) type.
- string dynNodeContext
Dynamic node context.
- string idComp
[Event](#) ID component.
- string name
User-defined name.
- vector< [EventDesc](#) > entries
Child nodes.

9.66.1 Detailed Description

An event descriptor.

In case eventDescType is LEAF then the descriptor refers to a 'real' event. In this case eventType is set and the entries vector is empty. Otherwise eventType is a don't care and the entries vector contains sub-entries. In case eventDescType is DYN_NODE then the dynNodeContext contains a key which is used to generate a dynamic node.

9.66.2 Member Enumeration Documentation

9.66.2.1 enum event::Engine::EventDesc::Type

[Event](#) descriptor type.

Enumerator

- NODE** Intermediate node.
- DYN_NODE** Dynamic node.
- LEAF** Leaf node.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EventEngine.idl

9.67 logging::EventLog_1_0_1 Interface Reference

Device event log interface.

```
import "EventLog.idl";
```

Public Member Functions

- void [clear](#) ()
Clear the event log.
- int [getFirstId](#) ()
Get the first available event id.
- int [getLastId](#) ()
Get the last available event id.
- void [getEntries](#) (out vector< [LogEntry](#) > *entries*, in int *refId*, in int *count*, in [RangeDirection](#) *direction*)
Fetch entries from the event log.
- void [getFilteredEntries](#) (out vector< [LogEntry](#) > *entries*, in int *refId*, in int *count*, in [RangeDirection](#) *direction*, in vector< string > *eventClasses*)
Fetch entries of certain event classes from the event log.

9.67.1 Detailed Description

Device event log interface.

9.67.2 Member Function Documentation

9.67.2.1 void logging::EventLog_1_0_1::getEntries (out vector< [LogEntry](#) > *entries*, in int *refId*, in int *count*, in [RangeDirection](#) *direction*)

Fetch entries from the event log.

Parameters

<i>entries</i>	Result: List of events
<i>refId</i>	First event id to fetch
<i>count</i>	Number of events to fetch
<i>direction</i>	Range direction

9.67.2.2 void logging::EventLog_1_0_1::getFilteredEntries (out vector< [LogEntry](#) > *entries*, in int *refId*, in int *count*, in [RangeDirection](#) *direction*, in vector< string > *eventClasses*)

Fetch entries of certain event classes from the event log.

Parameters

<i>entries</i>	Result: List of events
<i>refId</i>	First event id to fetch
<i>count</i>	Number of events to fetch
<i>direction</i>	Range direction
<i>eventClasses</i>	Event classes to filter for If empty, the method behaves the same way as getEntries ()).

9.67.2.3 `int logging::EventLog_1_0_1::getFirstId ( )`

Get the first available event id.

Returns

Event serial number

9.67.2.4 `int logging::EventLog_1_0_1::getLastId ( )`

Get the last available event id.

Returns

Event serial number

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/EventLog.idl

9.68 `event::Channel_1_0_1::EventSelect` Struct Reference

Structure to select an [Event](#) *.

```
import "EventService.idl";
```

Public Attributes

- typecode **type**
- Object **src**

9.68.1 Detailed Description

Structure to select an [Event](#) *.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EventService.idl

9.69 `hmi::ExternalBeeper_1_0_1` Interface Reference

External Beeper interface.

```
import "ExternalBeeper.idl";
```

Public Types

- enum [State](#) { **OFF**, **ON**, **ALARMING** }
Beeper state.

Public Member Functions

- [State](#) `getState ()`
Get current beeper state.
- void [alarm \(\)](#)
Start beeper alarm.
- void [on \(\)](#)
Turn beeper on.
- void [off \(\)](#)
Turn beeper off.

Public Attributes

- valueobject [StateChangedEvent](#): [idl.Event](#) { [State](#) oldState
Event: beeper state has changed.
- [State](#) **newState**

9.69.1 Detailed Description

External Beeper interface.

9.69.2 Member Function Documentation

9.69.2.1 void hmi::ExternalBeeper_1_0_1::alarm ()

Start beeper alarm.

Starts and repeats an alarm sequence of beeper on and beeper off. The sequence will be infinitely repeated until "off" is called.

9.69.2.2 void hmi::ExternalBeeper_1_0_1::off ()

Turn beeper off.

Disables beeper sound and also stops beeper alarm.

9.69.2.3 void hmi::ExternalBeeper_1_0_1::on ()

Turn beeper on.

Enables the beeper sound.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/ExternalBeeper.idl

9.70 test::FeatSerial Interface Reference

test routines for Raritan Feature Serial interface (RS232 with some control lines and switched power) Require Test-Mode to be ON.

```
import "testrpc.idl";
```

Public Member Functions

- int [getNumberOfPorts](#) (out int numPorts)
returns number of ports
- int [setPower](#) (in int portNum, in boolean hasPower)
Switches Power supplied to the port Observable with special test adapter Power->LED.
- int [testLoopTxRx](#) (in int portNum, out string errstr)
Performs a loop test with special test adapter TX->RX.
- int [testLoopDtrDcd](#) (in int portNum, out string errstr)
Performs a loop test with special test adapter DTR->DCD.

Public Attributes

- constant int [OK](#) = 0
No error.
- constant int [ERR_NO_TEST_MODE](#) = 1
Not in test mode.
- constant int [ERR_INVALID_PORT_NUM](#) = 2
Invalid port number.
- constant int [ERR_TEST_FAILED](#) = 3
Test failed.

9.70.1 Detailed Description

test routines for Raritan Feature Serial interface (RS232 with some control lines and switched power) Require Test-Mode to be ON.

9.70.2 Member Function Documentation

9.70.2.1 int test::FeatSerial::getNumberOfPorts (out int numPorts)

returns number of ports

Returns

- 0 if OK
- 1 if not in test mode

9.70.2.2 int test::FeatSerial::setPower (in int portNum, in boolean hasPower)

Switches Power supplied to the port Observable with special test adapter Power->LED.

Returns

- OK if OK
- ERR_NO_TEST_MODE if not in test mode
- ERR_INVALID_PORT_NUM if invalid port number

9.70.2.3 int test::FeatSerial::testLoopDtrDcd (in int *portNum*, out string *errstr*)

Performs a loop test with special test adapter DTR->DCD.

Returns

OK if OK
 ERR_NO_TEST_MODE if not in test mode
 ERR_INVALID_PORT_NUM if invalid port number
 ERR_TEST_FAILED if test failed, *errstr* may be set

9.70.2.4 int test::FeatSerial::testLoopTxRx (in int *portNum*, out string *errstr*)

Performs a loop test with special test adapter TX->RX.

Returns

OK if OK
 ERR_NO_TEST_MODE if not in test mode
 ERR_INVALID_PORT_NUM if invalid port number
 ERR_TEST_FAILED if test failed, *errstr* may be set

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/testrpc.idl

9.71 firmware::Firmware_1_0_1 Interface Reference

Firmware management methods

```
import "Firmware.idl";
```

Public Member Functions

- void [reboot](#) ()
Reboot the device.
- void [factoryReset](#) ()
Reset the device configuration to factory defaults.
- string [getVersion](#) ()
Returns the currently installed firmware version.
- vector< [UpdateHistoryEntry](#) > [getUpdateHistory](#) ()
Fetch the firmware update history.
- boolean [updateAvailable](#) (out boolean failedCheck, out time lastChecked, out string version, out string url)
Check whether a newer firmware version is available.
- void [enableOnlineCheck](#) (in boolean enable)
Enable the nightly online update checks.
- boolean [onlineCheckEnabled](#) ()
Check whether the nightly online update checks are enabled.
- void [performOnlineCheck](#) ()
Perform an online update check.
- boolean [downloadImage](#) (in string url)
Download a firmware update from a given URL.
- void [cancelDownload](#) ()

- Cancel the running download of a firmware image.*
- [ImageStatus getImageStatus](#) ()
Get the current firmware image upload/download status.
- void [discardImage](#) ()
Discard the currently uploaded firmware image, cancel the update.
- boolean [getImageInfo](#) (out [ImageInfo_1_0_1](#) info)
Return information about a currently uploaded firmware image.
- void [startUpdate](#) (in vector< [UpdateFlags](#) > flags)
Launch the firmware update process.

9.71.1 Detailed Description

Firmware management methods

9.71.2 Member Function Documentation

9.71.2.1 boolean firmware::Firmware_1_0_1::downloadImage (in string *url*)

Download a firmware update from a given URL.

This function instructs the device to download a firmware update image from the specified URL.

Returns

`true` if the download was successful, `false` otherwise.

9.71.2.2 void firmware::Firmware_1_0_1::enableOnlineCheck (in boolean *enable*)

Enable the nightly online update checks.

If this feature is enabled the device will automatically check whether a firmware update is available every night between 2:00 and 3:00 am.

Parameters

<i>enable</i>	Enable the automatic online update check
---------------	--

9.71.2.3 boolean firmware::Firmware_1_0_1::getImageInfo (out [ImageInfo_1_0_1](#) *info*)

Return information about a currently uploaded firmware image.

Parameters

<i>info</i>	Firmware image information
-------------	----------------------------

Returns

`true` if a firmware image is uploaded, `false` otherwise.

9.71.2.4 [ImageStatus](#) firmware::Firmware_1_0_1::getImageStatus ()

Get the current firmware image upload/download status.

Returns

Image status structure.

9.71.2.5 `vector<UpdateHistoryEntry> firmware::Firmware_1_0_1::getUpdateHistory ( )`

Fetch the firmware update history.

Returns

Vector of firmware update history entries

9.71.2.6 `string firmware::Firmware_1_0_1::getVersion ( )`

Returns the currently installed firmware version.

Returns

Firmware version

9.71.2.7 `boolean firmware::Firmware_1_0_1::onlineCheckEnabled ( )`

Check whether the nightly online update checks are enabled.

Returns

`true` if online update checks are enabled, `false` otherwise.

9.71.2.8 `void firmware::Firmware_1_0_1::performOnlineCheck ( )`

Perform an online update check.

This function instructs the device to contact the Raritan web site to check whether a newer firmware version is available. The result of the check (if successful) can be queried using [updateAvailable\(\)](#).

9.71.2.9 `void firmware::Firmware_1_0_1::reboot ( )`

Reboot the device.

This function will fail if a firmware update is in progress.

9.71.2.10 `void firmware::Firmware_1_0_1::startUpdate ( in vector< UpdateFlags > flags )`

Launch the firmware update process.

The device will stop handling RPC requests shortly after this method has been successfully called. The client should poll the `fwupdate_progress.cgi` page to monitor the update progress.

Parameters

<i>flags</i>	List of firmware update flags; may be empty
--------------	---

9.71.2.11 `boolean firmware::Firmware_1_0_1::updateAvailable ( out boolean failedCheck, out time lastChecked, out string version, out string url )`

Check whether a newer firmware version is available.

This function does not perform the actual online check, but just returns the result of the last successful check. Use [performOnlineCheck\(\)](#) or enable the automatic nightly checks to perform an online check.

Parameters

<i>failedCheck</i>	<code>true</code> if an online check attempt failed since the last successful one
<i>lastChecked</i>	Timestamp of last successful update check, negative means "never"
<i>version</i>	The new firmware version (if available)
<i>url</i>	The new firmware download URL (if available)

Returns

`true` if a newer firmware version is available, `false` otherwise.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Firmware.idl

9.72 peripheral::PackageInfo_2_0_0::FirmwareInfo Struct Reference

Classes

- struct [Version](#)

Public Attributes

- time [compileDate](#)
Date of firmware compilation.
- [Version](#) [version](#)
Firmware version (0.0 if not applicable)
- time [updateDate](#)
Date of device firmware update.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDevicePackage.idl

9.73 peripheral::G2Production_2_0_0::FirmwareInfo Struct Reference

Public Attributes

- int **crc**
- string **compiler**
- int **compilerVersion**
- string **compileDate**
- int **version**
- int **subVersion**
- int **configurationId**

- string **updateDate**

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralG2Production.idl

9.74 firmware::FirmwareUpdateStatus Interface Reference

Firmware update status interface.

```
import "FirmwareUpdateStatus.idl";
```

Public Member Functions

- [UpdateStatus getStatus](#) ()
Returns the device's firmware update status.

9.74.1 Detailed Description

Firmware update status interface.

Unlike all other sysrpc methods this function is implemented by a CGI script which is available even during a firmware update. The URL for this interface is /cgi-bin/fwupdate_progress.cgi.

9.74.2 Member Function Documentation

9.74.2.1 UpdateStatus firmware::FirmwareUpdateStatus::getStatus ()

Returns the device's firmware update status.

Returns

Update status structure

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/FirmwareUpdateStatus.idl

9.75 fitness::Fitness Interface Reference

Fitness Daemon interface

```
import "Fitness.idl";
```

Classes

- struct [DataEntry](#)
An entry in the reliability database.
- struct [ErrorLogEntry](#)
An entry in the reliability error log.

Public Member Functions

- `vector< DataEntry > getDataEntries ()`
Returns the fitness data entries.
- `void getErrorLogIndexRange (out int firstIndex, out int entryCount)`
Returns the error log index range.
- `vector< ErrorLogEntry > getErrorLogEntries (in int startIndex, in int count)`
Returns the error log.

Public Attributes

- constant int [FLAG_VALUE_INVALID](#) = 0x1
The value/worstValue/rawValue is invalid (e.g.
- constant int [FLAG_VALUE_OLD](#) = 0x2
The value/rawValue is out-dated.
- constant int [FLAG_ENTRY_CRITICAL](#) = 0x4
Violating the threshold is a critical event.

9.75.1 Detailed Description

Fitness Daemon interface

9.75.2 Member Function Documentation

9.75.2.1 `vector<DataEntry> fitness::Fitness::getDataEntries ( )`

Returns the fitness data entries.

The count of entries depends on the device. For example a PDU each slave board currently has 3 entries + 2 entries per relay. The data is updated only once or twice a minute.

Returns

– the vector with the fitness data entries

9.75.2.2 `vector<ErrorLogEntry> fitness::Fitness::getErrorLogEntries ( in int startIndex, in int count )`

Returns the error log.

If the startIndex is smaller than the first index in the log than the count of returned entries is smaller (or even 0) as well.

Parameters

<i>startIndex</i>	– the index of the first entry to return
<i>count</i>	– the count of entries starting from the startIndex

Returns

- the vector with the error log entries

9.75.2.3 void fitness::Fitness::getErrorLogIndexRange (out int *firstIndex*, out int *entryCount*)

Returns the error log index range.

Parameters

<i>firstIndex</i>	– the first valid index
<i>entryCount</i>	– the count of entries in the error log

9.75.3 Member Data Documentation

9.75.3.1 constant int fitness::Fitness::FLAG_ENTRY_CRITICAL = 0x4

Violating the threshold is a critical event.

9.75.3.2 constant int fitness::Fitness::FLAG_VALUE_INVALID = 0x1

The value/worstValue/rawValue is invalid (e.g. not initialized).

9.75.3.3 constant int fitness::Fitness::FLAG_VALUE_OLD = 0x2

The value/rawValue is out-dated.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Fitness.idl

9.76 webcam::Format_2_0_0 Struct Reference

Format.

```
import "Webcam.idl";
```

Public Attributes

- int [width](#)
image width
- int [height](#)
image height
- [PixelFormat](#) [pixelFormat](#)
pixel format

9.76.1 Detailed Description

Format.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Webcam.idl

9.77 peripheral::G2Production_2_0_0 Interface Reference

Classes

- struct [FirmwareInfo](#)

Public Types

- enum [ConfigurationSpace](#) { [HARDWARE](#), [FUNCTION](#), [FIRMWARE](#), [RESERVED](#) }
- enum [ResetMethod](#) { [BROWNOUT](#), [WATCHDOG](#) }

Public Member Functions

- int [updateFirmware](#) (in string romcode)
Update G2 peripheral firmware with binary previously uploaded using fwupload_g2pdev.cgi script.
- int [updateFirmwarePos](#) (in vector< [PosElement](#) > [position](#))
- int [getFirmwareInfo](#) (in string romcode, out [FirmwareInfo](#) info)
Read G2 peripheral device firmware information.
- int [getFirmwareInfoPos](#) (in vector< [PosElement](#) > [position](#), out [FirmwareInfo](#) info)
- int [readConfigurationSpace](#) (in string romcode, in [ConfigurationSpace](#) cs, out vector< byte > cfg)
Read the configuration space of a G2 peripheral device.
- int [readConfigurationSpacePos](#) (in vector< [PosElement](#) > [position](#), in [ConfigurationSpace](#) cs, out vector< byte > cfg)
- int [eraseConfigurationSpace](#) (in string romcode, in [ConfigurationSpace](#) cs)
Erase the configuration space of a G2 peripheral device.
- int [eraseConfigurationSpacePos](#) (in vector< [PosElement](#) > [position](#), in [ConfigurationSpace](#) cs)
- int [writeConfigurationSpace](#) (in string romcode, in [ConfigurationSpace](#) cs, in vector< byte > cfg)
Write a complete configuration space of a G2 peripheral device.
- int [writeConfigurationSpacePos](#) (in vector< [PosElement](#) > [position](#), in [ConfigurationSpace](#) cs, in vector< byte > cfg)
- int [readRegisters](#) (in string romcode, in int address, in int count, out vector< byte > data)
Read the registers of a G2 peripheral device.
- int [readRegistersPos](#) (in vector< [PosElement](#) > [position](#), in int address, in int count, out vector< byte > data)
- int [writeRegisters](#) (in string romcode, in int address, in vector< byte > data)
Write the registers of a G2 peripheral device.
- int [writeRegistersPos](#) (in vector< [PosElement](#) > [position](#), in int address, in vector< byte > data)
- int [writeRegisterBits](#) (in string romcode, in int address, in byte mask, in byte bits)
Set single (masked) bits of a G2 peripheral device register.
- int [writeRegisterBitsPos](#) (in vector< [PosElement](#) > [position](#), in int address, in byte mask, in byte bits)
- int [reset](#) (in string romcode, in [ResetMethod](#) method)
Reset a G2 peripheral device.
- int [resetPos](#) (in vector< [PosElement](#) > [position](#), in [ResetMethod](#) method)

Public Attributes

- constant int **ERR_INVALID_PARAMS** = 1
- constant int **ERR_NO_CONFIG_MODE** = 2
- constant int **ERR_NO_DEVICE** = 3
- constant int **ERR_NO_FIRMWARE_FILE** = 4
- constant int **ERR_FIRMWARE_INVALID** = 5
- constant int **ERR_PROTECTED** = 6
- constant int **ERR_UPDATE_IN_PROGRESS** = 7

9.77.1 Member Enumeration Documentation

9.77.1.1 enum peripheral::G2Production_2_0_0::ConfigurationSpace

Enumerator

- HARDWARE** HW-specific configuration data.
- FUNCTION** Function-specific configuration data.
- FIRMWARE** Firmware-specific configuration data.
- RESERVED** Reserved, Development only.

9.77.1.2 enum peripheral::G2Production_2_0_0::ResetMethod

Enumerator

- BROWNOUT** reset triggered instantly by firmware
- WATCHDOG** sensor is halted until watchdog triggers reset

9.77.2 Member Function Documentation

9.77.2.1 int peripheral::G2Production_2_0_0::eraseConfigurationSpace (in string *romcode*, in ConfigurationSpace *cs*)

Erase the configuration space of a G2 peripheral device.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>cs</i>	the configuration space identifier

Returns

- 0 if OK
- ERR_NO_CONFIG_MODE if the device is not in factory configuration mode
- ERR_NO_DEVICE no device present or wrong romcode

9.77.2.2 int peripheral::G2Production_2_0_0::getFirmwareInfo (in string *romcode*, out FirmwareInfo *info*)

Read G2 peripheral device firmware information.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>info</i>	the firmware information

Returns

0 if OK
ERR_NO_DEVICE no device present or wrong romcode
ERR_FIRMWARE_INVALID if firmware information invalid
ERR_UPDATE_IN_PROGRESS if firmware update is in progress

9.77.2.3 `int peripheral::G2Production_2.0.0::readConfigurationSpace ( in string romcode, in ConfigurationSpace cs, out vector< byte > cfg )`

Read the configuration space of a G2 peripheral device.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>cs</i>	the configuration space identifier

Returns

0 if OK
ERR_NO_CONFIG_MODE if the device is not in factory configuration mode
ERR_NO_DEVICE no device present or wrong romcode

9.77.2.4 `int peripheral::G2Production_2.0.0::readRegisters ( in string romcode, in int address, in int count, out vector< byte > data )`

Read the registers of a G2 peripheral device.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>address</i>	the address of (first) register to read from
<i>count</i>	the number of registers to read
<i>data</i>	the register data being read out

Returns

0 if OK
ERR_NO_CONFIG_MODE if the device is not in factory configuration mode
ERR_NO_DEVICE no device present or wrong romcode
ERR_INVALID_PARAMS if address or count out of bounds

9.77.2.5 `int peripheral::G2Production_2.0.0::reset ( in string romcode, in ResetMethod method )`

Reset a G2 peripheral device.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>method</i>	the reset method to be executed

Returns

0 if OK
ERR_NO_DEVICE no device present or wrong romcode

9.77.2.6 int peripheral::G2Production_2_0_0::updateFirmware (in string *romcode*)

Update G2 peripheral firmware with binary pereumously uploaded using fwupload_g2pdev.cgi script.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
----------------	---

Returns

0 if OK
 ERR_NO_CONFIG_MODE if the device is not in factory configuration mode
 ERR_NO_DEVICE no device present or wrong romcode
 ERR_NO_FIRMWARE_FILE if no previously uploaded firmware file is present
 ERR_FIRMWARE_INVALID if previously uploaded firmware file is invalid
 ERR_UPDATE_IN_PROGRESS if firmware update is already in progress

9.77.2.7 int peripheral::G2Production_2_0_0::writeConfigurationSpace (in string *romcode*, in ConfigurationSpace *cs*, in vector< byte > *cfg*)

Write a complete configuration space of a G2 peripheral device.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>cs</i>	the configuration space identifier
<i>cfg</i>	the configuration bytes

Returns

0 if OK
 ERR_NO_CONFIG_MODE if the device is not in factory configuration mode
 ERR_NO_DEVICE no device present or wrong romcode
 ERR_PROTECTED if configuration space is not writeable

9.77.2.8 int peripheral::G2Production_2_0_0::writeRegisterBits (in string *romcode*, in int *address*, in byte *mask*, in byte *bits*)

Set single (masked) bits of a G2 peripheral device register.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>address</i>	the address of register to set bits of
<i>mask</i>	the bitmask to apply (set bit: 1, don't set bit 0)
<i>bits</i>	the bit vailues to set

Returns

0 if OK
 ERR_NO_CONFIG_MODE if the device is not in factory configuration mode
 ERR_NO_DEVICE no device present or wrong romcode
 ERR_INVALID_PARAMS if address or count ouf of bounds
 ERR_PROTECTED if registers are not writeable

9.77.2.9 `int peripheral::G2Production_2_0_0::writeRegisters ( in string romcode, in int address, in vector< byte > data )`

Write the registers of a G2 peripheral device.

Parameters

<i>romcode</i>	1-wire rom code for device identification, can be left empty if only one device connected
<i>address</i>	the address of (first) register to write to
<i>data</i>	the register data to be written

Returns

0 if OK
 ERR_NO_CONFIG_MODE if the device is not in factory configuration mode
 ERR_NO_DEVICE no device present or wrong romcode
 ERR_INVALID_PARAMS if address or count out of bounds
 ERR_PROTECTED if registers are not writeable

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralG2Production.idl

9.78 serial::GsmModem_1_0_1 Interface Reference

Interface for communication with a GSM modem attached to a serial port.

```
import "GsmModem.idl";
```

Classes

- struct [Information](#)
Structure holding information about the modem and the SIM card.
- struct [Settings](#)
Structure for holding settings of the GSM modem and its SIM card.

Public Types

- enum [SimSecurityStatus](#) { [UNLOCKED](#), [WAITFORPIN](#), [WAITFORPUK](#), [UNKNOWN](#) }
Possible security states the sim card can be in at a given time.

Public Member Functions

- [Settings](#) [getSettings](#) ()
Get modem settings.
- int [setSettings](#) (in [Settings](#) settings)
Set modem settings.
- int [sendSms](#) (in string recipient, in string text)
Send out a SMS message.
- int [sendTestSms](#) (in string recipient, in [Settings](#) testSettings)
Send out a test SMS message.
- int [getInformation](#) (out [Information](#) info)
Retrieve low-level information about the modem and the SIM card.

- int [getInformationWithPin](#) (in string pin, out [Information](#) info)
Retrieve low-level information about the modem and the SIM card.
- int [getSimSecurityStatus](#) (out [SimSecurityStatus](#) simStatus)
Retrieve security status of the SIM card.
- int [unlockSimCard](#) (in string puk, in string newPin)
Unlock SIM card with PUK and set new PIN if the SIM card is in security status WAITFORPUK.

Public Attributes

- constant int [SUCCESS](#) = 0
Error codes.
- constant int [ERR_INVALID_VALUE](#) = 1
Invalid argument.
- constant int [ERR_WRONG_PIN](#) = 2
The PIN is incorrect or missing.
- constant int [ERR_SMS_SEND_FAILED](#) = 3
SMS delivery failed.
- constant int [ERR_COMMUNICATION_FAILURE](#) = 4
Communication with the modem failed.
- constant int [ERR_SIM_LOCKED](#) = 5
The SIM card is locked and waits for the PUK.
- constant int [ERR_WRONG_SIM_STATUS](#) = 6
The SIM card doesn't wait for the PUK.
- constant int [ERR_WRONG_PUK](#) = 7
The PUK is incorrect or missing.
- valueobject [SimSecurityStatusChangedEvent](#): [idl.Event](#) { [SimSecurityStatus](#) newSimStatus
Sim card security status changed event.
- valueobject [SimPinUpdatedEvent](#): [idl.Event](#) { string newPin
Sim pin updated event.

9.78.1 Detailed Description

Interface for communication with a GSM modem attached to a serial port.

9.78.2 Member Enumeration Documentation

9.78.2.1 enum serial::GsmModem_1_0_1::SimSecurityStatus

Possible security states the sim card can be in at a given time.

Enumerator

- UNLOCKED** SIM card is unlocked.
- WAITFORPIN** PIN must be entered to unlock the SIM card.
- WAITFORPUK** PUK and new PIN must be entered to unlock the SIM card.
- UNKNOWN** Unkown security status.

9.78.3 Member Function Documentation

9.78.3.1 `int serial::GsmModem_1_0_1::getInformation ( out Information info )`

Retrieve low-level information about the modem and the SIM card.

Parameters

<i>info</i>	– structure holding the returned information
-------------	--

Returns

SUCCESS – on success
ERR_WRONG_PIN – if the used PIN is incorrect or missing
ERR_SIM_LOCKED – if the SIM card is locked and waits for the PUK
ERR_COMMUNICATION_FAILURE – if communication with the modem failed

9.78.3.2 `int serial::GsmModem_1_0_1::getInformationWithPin ( in string pin, out Information info )`

Retrieve low-level information about the modem and the SIM card.

Like `#getInformation`, but allows providing a PIN not stored in the settings

Parameters

<i>pin</i>	– PIN to use for authentication
<i>info</i>	– structure holding the returned information

Returns

SUCCESS – on success
ERR_WRONG_PIN – if the used PIN is incorrect or missing
ERR_SIM_LOCKED – if the SIM card is locked and waits for the PUK
ERR_COMMUNICATION_FAILURE – if communication with the modem failed

9.78.3.3 `Settings serial::GsmModem_1_0_1::getSettings ( )`

Get modem settings.

Returns

– Current modem settings

9.78.3.4 `int serial::GsmModem_1_0_1::getSimSecurityStatus ( out SimSecurityStatus simStatus )`

Retrieve security status of the SIM card.

Parameters

<i>simStatus</i>	– SIM card security status
------------------	----------------------------

Returns

SUCCESS – on success
ERR_COMMUNICATION_FAILURE – if communication with the modem failed

9.78.3.5 int serial::GsmModem_1_0_1::sendSms (in string *recipient*, in string *text*)

Send out a SMS message.

Parameters

<i>recipient</i>	– Phone number of the message recipient in ITU-T E.164 format
<i>text</i>	– Message text (will be sent in multiple messages if longer than 160 characters)

Returns

SUCCESS – on success
ERR_WRONG_PIN – if the PIN currently stored in the settings is incorrect or missing
ERR_SMS_SEND_FAILED – if the delivery of the SMS to the network failed
ERR_SIM_LOCKED – if the SIM card is locked and waits for the PUK
ERR_COMMUNICATION_FAILURE – if communication with the modem failed

9.78.3.6 int serial::GsmModem_1_0_1::sendTestSms (in string *recipient*, in Settings *testSettings*)

Send out a test SMS message.

The message will be sent to the selected recipient with the text 'SMS Test'.

Parameters

<i>recipient</i>	– Phone number of the message recipient in ITU-T E.164 format
<i>testSettings</i>	– Modem settings to be used temporarily during testing

Returns

SUCCESS – on success
ERR_WRONG_PIN – if the PIN currently stored in the settings is incorrect or missing
ERR_SMS_SEND_FAILED – if the delivery of the SMS to the network failed
ERR_SIM_LOCKED – if the SIM card is locked and waits for the PUK
ERR_COMMUNICATION_FAILURE – if communication with the modem failed

9.78.3.7 int serial::GsmModem_1_0_1::setSettings (in Settings *settings*)

Set modem settings.

Parameters

<i>settings</i>	– New settings
-----------------	----------------

Returns

SUCCESS – on success
ERR_INVALID_VALUE – if any passed value was invalid

9.78.3.8 int serial::GsmModem_1_0_1::unlockSimCard (in string puk, in string newPin)

Unlock SIM card with PUK and set new PIN if the SIM card is in security status WAITFORPUK.

The new PIN is automatically saved in the settings.

Parameters

<i>puk</i>	– PUK to use for authentication
<i>newPin</i>	– new PIN to use for future authentication

Returns

SUCCESS – on success
 ERR_WRONG_SIM_STATUS – if the SIM card doesn't wait for the PUK
 ERR_WRONG_PUK – if the used PUK is incorrect or missing
 ERR_COMMUNICATION_FAILURE – if communication with the modem failed

9.78.4 Member Data Documentation

9.78.4.1 valueobject serial::GsmModem_1_0_1::SimPinUpdatedEvent

Sim pin updated event.

new PIN for SIM card after Unlock

9.78.4.2 valueobject serial::GsmModem_1_0_1::SimSecurityStatusChangedEvent

Sim card security status changed event.

new SIM card security status

9.78.4.3 constant int serial::GsmModem_1_0_1::SUCCESS = 0

Error codes.

No error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/GsmModem.idl

9.79 peripheral::PackageInfo_2_0_0::HardwareInfo Struct Reference

Public Attributes

- string [serial](#)
serial number
- string [packageClass](#)
serial number prefix for current packages
- string [model](#)
like 'DPX-CC2' or 'DX-D2C6'
- int [minDowngradeVersion](#)
minimum downgrade version (or -1)

- string [revision](#)
hardware revision

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDevicePackage.idl

9.80 session::HistoryEntry Struct Reference

Session history entry

```
import "SessionManager.idl";
```

Public Attributes

- time [creationTime](#)
Session creation timestamp.
- string [remotelp](#)
Session IP address.
- string [clientType](#)
Session client type.

9.80.1 Detailed Description

Session history entry

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SessionManager.idl

9.81 webcam::Image_2_0_0 Struct Reference

Image.

```
import "Webcam.idl";
```

Public Attributes

- [ImageMetaData](#) [meta](#)
image meta data
- string [data](#)
base64 encoded image data

9.81.1 Detailed Description

Image.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Webcam.idl

9.82 firmware::ImageInfo_1_0_1 Struct Reference

Firmware image information

```
import "Firmware.idl";
```

Public Attributes

- boolean [valid](#)
The file is a valid firmware image.
- string [version](#)
Firmware image version
- string [min_required_version](#)
Minimum running firmware version for image.
- string [min_downgrade_version](#)
Minimum image version for running firmware.
- string [product](#)
Product name.
- string [platform](#)
Platform name.
- string [oem](#)
OEM name.
- string [hwid_whitelist](#)
Hardware ID whitelist.
- string [hwid_blacklist](#)
Hardware ID blacklist.
- boolean [compatible](#)
true if the image is compatible with this device
- boolean [signature_present](#)
true if the image is signed
- string [signed_by](#)
Signature issuer.
- boolean [signature_good](#)
true if the signature is valid
- string [certified_by](#)
Key certificate issuer.
- boolean [certificate_good](#)
true if the key certificate is valid
- boolean [model_list_present](#)
true if the image includes a supported models list
- boolean [model_supported](#)
true if the model is found on the support list

9.82.1 Detailed Description

Firmware image information

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Firmware.idl

9.83 webcam::ImageMetaData Struct Reference

Image meta data.

```
import "Webcam.idl";
```

Public Attributes

- [Format_2_0_0 format](#)
image format information
- long [timestamp](#)
image timestamp
- [Location location](#)
source webcam location

9.83.1 Detailed Description

Image meta data.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Webcam.idl

9.84 firmware::ImageStatus Struct Reference

Image upload/download status.

```
import "Firmware.idl";
```

Public Attributes

- [ImageState state](#)
Image upload/download state.
- string [error_message](#)
Error message; empty if there was no error.
- time [time_started](#)
Timestamp of the last state change (if available)
- int [size_total](#)
Total size of the image (if available)
- int [size_done](#)
Progress of the running upload or download (if available)

9.84.1 Detailed Description

Image upload/download status.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Firmware.idl

9.85 webcam::StorageManager_1_0_1::ImageStorageMetaData Struct Reference

[StorageMetaData](#).

```
import "StorageManager.idl";
```

Public Attributes

- [ImageMetaData](#) `imageMeta`
image related meta data
- int [fileSize](#)
image file size in bytes
- [StorageMetaData](#) `storageMeta`
store related meta data

9.85.1 Detailed Description

[StorageMetaData](#).

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.86 assetmgrmodel::AssetStripLogger_1_0_2::Info Struct Reference

Log information structure.

```
import "AssetStripLogger.idl";
```

Public Attributes

- int [capacity](#)
Maximum number of entries in the record ring buffer.
- int [oldestRecord](#)
Pointer to the oldest log entry; -1 if the log is empty.
- int [newestRecord](#)
Pointer to the newest log entry; -1 if the log is empty.
- int [totalEventCount](#)
Total number of events logged.

9.86.1 Detailed Description

Log information structure.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStripLogger.idl

9.87 usermgmt::Role::Info Struct Reference

Role information

```
import "Role.idl";
```

Public Attributes

- string [description](#)
Free-form description.
- boolean [locked](#)
true if the role cannot be deleted
- vector< [Privilege](#) > [privileges](#)
List of privileges for this role.

9.87.1 Detailed Description

Role information

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Role.idl

9.88 usermgmt::RoleManager::Info Struct Reference

Full role manager information.

```
import "RoleManager.idl";
```

Public Attributes

- vector< [PrivilegeDesc](#) > [privileges](#)
List of supported privileges.
- vector< [RoleAccount](#) > [roles](#)
List of active roles.

9.88.1 Detailed Description

Full role manager information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/RoleManager.idl

9.89 cert::ServerSSLCert::Info Struct Reference

Certificate manager information.

```
import "ServerSSLCert.idl";
```

Public Attributes

- boolean [havePendingReq](#)
true if a CSR is pending
- boolean [havePendingCert](#)
true if an uploaded certificate is pending activation
- [ReqInfo](#) [pendingReqInfo](#)

Information about pending CSR.

- [CertInfo pendingCertInfo](#)

Information about pending certificate.

- [CertInfo activeCertInfo](#)

Information about active certificate.

- `int` [maxSignDays](#)

Maximum number of days a self signed certificate will be valid.

9.89.1 Detailed Description

Certificate manager information.

9.89.2 Member Data Documentation

9.89.2.1 `int cert::ServerSSLCert::Info::maxSignDays`

Maximum number of days a self signed certificate will be valid.

Necessary because openssl < 1.0 does not handle time overflows.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ServerSSLCert.idl

9.90 `test::Display_1_0_1::Info` Struct Reference

Collected display meta information.

```
import "TestDisplay.idl";
```

Public Attributes

- `string` [type](#)

Display type.

- `string` [address](#)

Display address.

- `map< string, string >` [options](#)

Display options.

- [Orientation](#) [orientation](#)

Display orientation.

9.90.1 Detailed Description

Collected display meta information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TestDisplay.idl

9.91 serial::GsmModem_1_0_1::Information Struct Reference

Structure holding information about the modem and the SIM card.

```
import "GsmModem.idl";
```

Public Attributes

- string [imei](#)
IMEI of the modem.
- string [imsi](#)
IMSI of the SIM card.
- string [manufacturer](#)
modem manufacturer string
- string [model](#)
modem model string
- string [revision](#)
modem revision string
- string [ownNumber](#)
own phone number in ITU-T E.164 format
- string [simSmsc](#)
SMS center number stored on SIM card.
- string [networkName](#)
Name of the currently used network (PLMN)
- string [serviceProviderName](#)
Name of the service provider (SPN)
- int [receptionLevel](#)
*reception level in dBm
0 means unknown, -1 means no reception*

9.91.1 Detailed Description

Structure holding information about the modem and the SIM card.

Any of the fields may be empty if the information can not be retrieved.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/GsmModem.idl

9.92 webcam::Information_2_0_0 Struct Reference

Webcam information.

```
import "Webcam.idl";
```

Public Attributes

- string [id](#)
port and camera specific ID string
- vector< [Format_2_0_0](#) > [supportedFormats](#)
supported image formats

9.92.1 Detailed Description

Webcam information.

The documentation for this struct was generated from the following file:

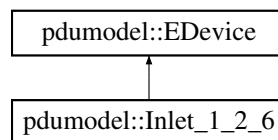
- pdu-json-rpc-api/idl/Webcam.idl

9.93 pdumodel::Inlet_1_2_6 Interface Reference

Inlet interface

```
import "Inlet.idl";
```

Inheritance diagram for pdumodel::Inlet_1_2_6:



Classes

- struct [MetaData](#)
Inlet metadata
- struct [Sensors](#)
Inlet sensors
- struct [Settings](#)
Inlet settings

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the inlet metadata.
- [Sensors](#) [getSensors](#) ()
Get the inlet sensors.
- vector< [Pole_2_0_0](#) > [getPoles](#) ()
Get the list of inlet poles.
- [Settings](#) [getSettings](#) ()
Retrieve the inlet settings.
- int [setSettings](#) (in [Settings](#) settings)
Change the inlet settings.
- void [setEnabled](#) (in boolean enabled)
Enable/disable PDU operation for this inlet.
- boolean [isEnabled](#) ()
Test whether PDU operation is enabled for this inlet.

Public Attributes

- valueobject [SettingsChangedEvent](#): [event.UserEvent](#) { [Settings](#) oldSettings
Event: Inlet settings have been changed.
- [Settings](#) newSettings
Settings after change.
- valueobject [EnableStateChangedEvent](#): [event.UserEvent](#) { boolean enabled
Event: Inlet has been enabled or disabled.

9.93.1 Detailed Description

Inlet interface

9.93.2 Member Function Documentation

9.93.2.1 [MetaData](#) pdumodel::Inlet_1_2_6::getMetaData ()

Retrieve the inlet metadata.

Returns

Inlet metadata

9.93.2.2 [vector<Pole_2_0_0>](#) pdumodel::Inlet_1_2_6::getPoles ()

Get the list of inlet poles.

Returns

List of inlet poles

9.93.2.3 [Sensors](#) pdumodel::Inlet_1_2_6::getSensors ()

Get the inlet sensors.

Returns

Inlet sensors

9.93.2.4 [Settings](#) pdumodel::Inlet_1_2_6::getSettings ()

Retrieve the inlet settings.

Returns

Inlet settings

9.93.2.5 [boolean](#) pdumodel::Inlet_1_2_6::isEnabled ()

Test whether PDU operation is enabled for this inlet.

Returns

`true` if PDU operation is enabled

9.93.2.6 void pdumodel::Inlet_1_2_6::setEnabled (in boolean *enabled*)

Enable/disable PDU operation for this inlet.

When PDU operation is disabled the sensors for this inlet and all children will no longer be updated, and outlet switching is no longer allowed. This can be useful for multi-inlet units if one inlet is temporarily expected to be powered down.

Parameters

<i>enabled</i>	true to enable PDU operation
----------------	------------------------------

9.93.2.7 int pdumodel::Inlet_1_2_6::setSettings (in Settings *settings*)

Change the inlet settings.

Parameters

<i>settings</i>	New inlet settings
-----------------	--------------------

Returns

- 0 if OK
- 1 if any parameters are invalid

9.93.3 Member Data Documentation

9.93.3.1 valueobject pdumodel::Inlet_1_2_6::EnableStateChangedEvent

Event: Inlet has been enabled or disabled.

New enable state

9.93.3.2 valueobject pdumodel::Inlet_1_2_6::SettingsChangedEvent

Event: Inlet settings have been changed.

[Settings](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Inlet.idl

9.94 net::InterfaceState_2_0_0 Struct Reference

LAN interface state.

```
import "Net.idl";
```

Public Attributes

- [InterfaceMode_2_0_0](#) mode
Wired or wireless interface configured?
- [InterfaceMode_2_0_0](#) activeMode
Currently used interface type.

- boolean [wirelessSupported](#)
if a wireless interface is available

9.94.1 Detailed Description

LAN interface state.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.95 hmi::InternalBeeper Interface Reference

Internal beeper interface.

```
import "InternalBeeper.idl";
```

Public Types

- enum [State](#) { [OFF](#), [ON_NOTIFICATION](#), [ON_ACTIVATION](#) }
Activation state.

Public Member Functions

- void [mute](#) (in boolean muted)
Mute beeper, turn of all internal alarm notifications.
- boolean [isMuted](#) ()
Check whether beeper is currently muted.
- void [activate](#) (in boolean on, in string [reason](#), in int timeout)
Activate the beeper for a given time.
- [State](#) [getState](#) (out string [reason](#))
Retrieve the current beeper activation state.

Public Attributes

- valueobject [MuteChangedEvent](#): [event.UserEvent](#) { boolean muted
Event: The beeper has been muted or unmuted.
- valueobject [StateChangedEvent](#): [idl.Event](#) { [State](#) state
Event: The beeper activation status has changed.
- string [reason](#)
Activation reason.

9.95.1 Detailed Description

Internal beeper interface.

9.95.2 Member Enumeration Documentation

9.95.2.1 enum hmi::InternalBeeper::State

Activation state.

Enumerator

OFF Beeper is currently off.

ON_NOTIFICATION Beeper is currently active due to an internal alarm notification.

ON_ACTIVATION Beeper is currently active due to an external activation.

9.95.3 Member Function Documentation

9.95.3.1 void hmi::InternalBeeper::activate (in boolean *on*, in string *reason*, in int *timeout*)

Activate the beeper for a given time.

Parameters

<i>activate</i>	Whether to turn on or off the beeper
<i>reason</i>	Description of the reason to turn on the beeper (only valid whtn turning on the beeper)
<i>timeout</i>	Activation timeout in seconds (only valid when turning on the beeper)

9.95.3.2 State hmi::InternalBeeper::getState (out string *reason*)

Retrieve the current beeper activation state.

Parameters

<i>reason</i>	Return value for activation reason if the beeper is currently active
---------------	--

Returns

The current beeper state

9.95.3.3 boolean hmi::InternalBeeper::isMuted ()

Check whether beeper is currently muted.

Returns

`true` if muted, `false` if not

9.95.3.4 void hmi::InternalBeeper::mute (in boolean *muted*)

Mute beeper, turn of all internal alarm notifications.

Parameters

<i>mute</i>	<code>true</code> to mute beeper, <code>false</code> for normal mode
-------------	--

9.95.4 Member Data Documentation

9.95.4.1 valueobject hmi::InternalBeeper::StateChangedEvent

Event: The beeper activation status has changed.

The current beeper state

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/InternalBeeper.idl

9.96 security::IpFw_2_0_0 Struct Reference

IP packet filter configuration.

```
import "Security.idl";
```

Public Attributes

- boolean [enabled](#)
true to enable packet filtering
- [IpfwPolicy](#) [defaultPolicyIn](#)
The default policy for inbound traffic in case no rule matches.
- [IpfwPolicy](#) [defaultPolicyOut](#)
The default policy for outbound traffic in case no rule matches.
- vector< [IpfwRule](#) > [ruleSetIn](#)
Ordered list of inbound firewall rules.
- vector< [IpfwRule](#) > [ruleSetOut](#)
Ordered list of outbound firewall rules.

9.96.1 Detailed Description

IP packet filter configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.97 security::IpfwRule Struct Reference

IP packet filter rule.

```
import "Security.idl";
```

Public Attributes

- string [ipMask](#)
Remote IP and network mask.
- [IpfwPolicy](#) [policy](#)
Filter policy.

9.97.1 Detailed Description

IP packet filter rule.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.98 net::IPv4RoutingEntry Struct Reference

IPv4 Routing entry.

```
import "Net.idl";
```

Public Attributes

- string [dest](#)
Destination address.
- string [nexthop](#)
Next hop address / Router.
- string [intf](#)
Network interface.

9.98.1 Detailed Description

IPv4 Routing entry.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.99 net::IPv6RoutingEntry Struct Reference

IPv6 Routing entry.

```
import "Net.idl";
```

Public Attributes

- string [dest](#)
Destination address.
- string [nexthop](#)
Next hop address.
- string [intf](#)
Network interface.

9.99.1 Detailed Description

IPv6 Routing entry.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.100 event::KeyValue Struct Reference

Helper that is used wherever key/value pairs are required.

```
import "EventEngine.idl";
```

Public Attributes

- string [key](#)
Key.
- string [value](#)
Value.

9.100.1 Detailed Description

Helper that is used wherever key/value pairs are required.

Note

This interface was designed before IDL maps were supported.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EventEngine.idl

9.101 net::LanInterfaceParameters_2_0_0 Struct Reference

Current LAN interface parameters.

```
import "Net.idl";
```

Public Attributes

- [LanSpeed](#) [speed](#)
Current speed.
- [LanDuplex](#) [duplex](#)
Current duplex mode.
- boolean [autonegotiation](#)
true if auto-negotiation is enabled
- boolean [link](#)
true if a link is detected
- vector< [LanLinkMode](#) > [supportedModes](#)
Supported link modes.

9.101.1 Detailed Description

Current LAN interface parameters.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.102 net::LanInterfaceSettings Struct Reference

LAN interface settings.

```
import "Net.idl";
```

Public Attributes

- [LanSpeed speed](#)
Speed.
- [LanDuplex duplex](#)
Duplex mode.

9.102.1 Detailed Description

LAN interface settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.103 net::LanLinkMode Struct Reference

LAN interface link mode.

```
import "Net.idl";
```

Public Attributes

- [LanSpeed speed](#)
Interface speed.
- [LanDuplex duplex](#)
Interface duplex mode.

9.103.1 Detailed Description

LAN interface link mode.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.104 auth::LdapManager_1_0_1 Interface Reference

LDAP server configuration interface.

```
import "LdapManager.idl";
```

Public Member Functions

- `vector< Idapsrv.ServerSettings > getLdapServers ( )`
Get a list of LDAP server settings.
- `int setLdapServers (in vector< Idapsrv.ServerSettings > serverList)`
Sets a list of LDAP servers.
- `int testLdapServer (in string username, in string password, in Idapsrv.ServerSettings settings)`
Tests an LDAP server configuration.

Public Attributes

- constant int [ERR_CYCLIC_DEP](#) = 1
Cyclic dependency in server list.
- constant int [ERR_INVALID_CFG](#) = 2
The server configuration is invalid.
- constant int [ERR_SERVER_UNSPECIFIED](#) = 1
Unspecified error.
- constant int [ERR_SERVER_UNREACHABLE](#) = 3
LDAP server could not be contacted.
- constant int [ERR_AUTHENTICATION_FAILED](#) = 4
User could not be authenticated.
- constant int [ERR_NO_ROLES](#) = 5
No roles are defined for the user.
- constant int [ERR_NO_KNOWN_ROLES](#) = 6
No known rules are defined for the user.

9.104.1 Detailed Description

LDAP server configuration interface.

9.104.2 Member Function Documentation

9.104.2.1 `vector<Idapsrv.ServerSettings> auth::LdapManager_1_0_1::getLdapServers ( )`

Get a list of LDAP server settings.

Returns

list of [Idapsrv.ServerSettings](#)

9.104.2.2 `int auth::LdapManager_1_0_1::setLdapServers ( in vector< Idapsrv.ServerSettings > serverList )`

Sets a list of LDAP servers.

Any existing LDAP Server configuration will be cleared / overwritten.

Returns

0 on success
[ERR_CYCLIC_DEP](#) in case of cyclic dependency
[ERR_INVALID_CFG](#) in case of invalid configuration

9.104.2.3 `int auth::LdapManager_1_0_1::testLdapServer ( in string username, in string password, in Idapsrv.ServerSettings settings )`

Tests an LDAP server configuration.

Returns

0 on success
 ERR_SERVER_UNSPECIFIED an unspecified error occurred
 ERR_INVALID_CFG LDAP server configuration is invalid (reused from setLdapServers)
 ERR_SERVER_UNREACHABLE LDAP server could not be contacted
 ERR_AUTHENTICATION_FAILED user could not be authenticated
 ERR_NO_ROLES no roles are defined for the user
 ERR_NO_KNOWN_ROLES no known roles are defined for the user

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/LdapManager.idl

9.105 assetmgrmodel::AssetStripConfig_1_0_1::LEDColor Struct Reference

The LED color in RGB format, 8 bit per channel.

```
import "AssetStripConfig.idl";
```

Public Attributes

- int `r`
red channel of the LED
- int `g`
green channel of the LED
- int `b`
blue channel of the LED

9.105.1 Detailed Description

The LED color in RGB format, 8 bit per channel.

Supported range is 0-255

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStripConfig.idl

9.106 pdumodel::Outlet_1_5_6::LedState Struct Reference

Outlet LED state

```
import "Outlet.idl";
```

Public Attributes

- boolean `red`

- true if the red LED is enabled*
- boolean [green](#)
 - true if the green LED is enabled*
- boolean [blinking](#)
 - true if the LED is blinking*

9.106.1 Detailed Description

Outlet LED state

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Outlet.idl

9.107 Ihxmodel::Lhx_3_2_1 Interface Reference

LHX Interface.

```
import "Lhx.idl";
```

Classes

- struct [AlertStatus](#)
 - LHX alert status.*
- struct [Capabilities](#)
 - LHX capabilities.*
- struct [MetaData](#)
 - LHX metadata.*
- struct [OpState](#)
 - LHX operational state.*
- struct [ParamCfg](#)
 - Configuration parameter characteristics.*
- struct [Settings](#)
 - LHX settings.*

Public Member Functions

- [Capabilities](#) [getCapabilities](#) ()
 - Retrieve the LHX capabilities (static).*
- [MetaData](#) [getMetaData](#) ()
 - Retrieve the LHX metadata.*
- [Settings](#) [getSettings](#) ()
 - Retrieve the LHX settings.*
- int [setSettings](#) (in [Settings](#) settings)
 - Change the LHX settings.*
- vector< [Sensor_4_0_0](#) > [getSensors](#) ()
 - Get LHX sensors.*
- [OpState](#) [getOpState](#) ()
 - Get LHX operational state.*
- int [setPowerState](#) (in [sensors.Sensor_4_0_0.OnOffState](#) state)

- *Switch powerstate of LHX.*
- vector< [Parameter_2_0_1](#) > [getParameters](#) ()
Get parameter list.
- vector< [Parameter_2_0_1](#) > [getActualValues](#) ()
Get actual value list.
- int [setMaximumCoolingRequest](#) (in boolean requested)
Request maximum cooling.
- int [acknowledgeAlertStatus](#) ()
Acknowledge alert status.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.
- constant int [ERR_NOT_SUPPORTED](#) = 2
Not supported.
- valueobject [OpStateChangedEvent](#): [idl.Event](#) { [OpState](#) oldState
Event: LHX operational status has changed.
- [OpState](#) newState
Status after change.
- valueobject [SettingsChangedEvent](#): [event.UserEvent](#) { [Settings](#) oldSettings
Event: LHX settings have changed.
- [Settings](#) newSettings
Settings after change.

9.107.1 Detailed Description

LHX Interface.

9.107.2 Member Function Documentation

9.107.2.1 int [Ihxmodel::Lhx_3_2_1::acknowledgeAlertStatus](#) ()

Acknowledge alert status.

Returns

0 if OK
ERR_NOT_SUPPORTED if operation is not supported

9.107.2.2 [MetaData](#) [Ihxmodel::Lhx_3_2_1::getMetaData](#) ()

Retrieve the LHX metadata.

Returns

LHX metadata

9.107.2.3 Settings Ihxmodel::Lhx_3_2_1::getSettings ()

Retrieve the LHX settings.

Returns

LHX settings

9.107.2.4 int Ihxmodel::Lhx_3_2_1::setMaximumCoolingRequest (in boolean *requested*)

Request maximum cooling.

Parameters

<i>requested</i>	true if request maximum cooling false if request normal operation
------------------	---

Returns

0 if OK
ERR_NOT_SUPPORTED if operation is not supported

9.107.2.5 int Ihxmodel::Lhx_3_2_1::setPowerState (in sensors.Sensor_4_0_0.OnOffState *state*)

Switch powerstate of LHX.

Parameters

<i>state</i>	on or off
--------------	-----------

Returns

0 if OK
ERR_NOT_SUPPORTED if operation is not supported

9.107.2.6 int Ihxmodel::Lhx_3_2_1::setSettings (in Settings *settings*)

Change the LHX settings.

Parameters

<i>settings</i>	New LHX settings
-----------------	------------------

Returns

0 if OK
ERR_INVALID_PARAMS if any parameters are invalid

9.107.3 Member Data Documentation

9.107.3.1 valueobject Ihxmodel::Lhx_3_2_1::OpStateChangedEvent

Event: LHX operational status has changed.

Status before change

9.107.3.2 valueobject lhxmodel::Lhx_3_2_1::SettingsChangedEvent

Event: LHX settings have changed.

[Settings](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Lhx.idl

9.108 webcam::Location Struct Reference

[Location](#).

```
import "Webcam.idl";
```

Public Attributes

- string [name](#)
location name
- string [x](#)
x
- string [y](#)
y
- string [z](#)
z

9.108.1 Detailed Description

[Location](#).

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Webcam.idl

9.109 peripheral::DeviceSlot_2_0_0::Location Struct Reference

user writeable location

```
import "PeripheralDeviceSlot.idl";
```

Public Attributes

- string [x](#)
X coordinate.
- string [y](#)
Y coordinate.
- string [z](#)
Z coordinate (semantics depends on ZCoordMode)

9.109.1 Detailed Description

user writeable location

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceSlot.idl

9.110 logging::LogEntry Struct Reference

Device event.

```
import "Log.idl";
```

Public Attributes

- int [id](#)
Serial number.
- time [timestamp](#)
Event time stamp.
- string [eventClass](#)
Event class code.
- string [message](#)
Event message.

9.110.1 Detailed Description

Device event.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Log.idl

9.111 sensors::Logger_2_1_2 Interface Reference

Sensor logger interface.

```
import "SensorLogger.idl";
```

Classes

- struct [LogRow](#)
One full log row.
- struct [Record](#)
Sensor log record.
- struct [SensorSet](#)
Set of logged sensors.
- struct [Settings](#)
Sensor logger settings.
- struct [TimedRecord](#)
Sensor log record with timestamp.

Public Member Functions

- [Settings](#) [getSettings](#) ()
Retrieve the sensor logger settings.
- int [setSettings](#) (in boolean isEnabled, in int samplesPerRecord)
Change the sensor logger settings.
- int [getTimeStamps](#) (out vector< time > timestamps, in int recid, in int count)
Retrieve a set of log record timestamps.
- int [getSensorRecords](#) (out vector< [Record](#) > recs, in [sensors.Sensor_4_0_0](#) sensor, in int recid, in int count)
Retrieve log records for a given sensor.
- int [getPeripheralDeviceRecords](#) (out vector< [Record](#) > recs, in [peripheral.DeviceSlot_2_0_0](#) slot, in int recid, in int count)
Retrieve log records for an peripheral device slot.
- int [getSensorTimedRecords](#) (out vector< [TimedRecord](#) > recs, in [sensors.Sensor_4_0_0](#) sensor, in int recid, in int count)
Retrieve log records with timestamps for a given sensor.
- int [getPeripheralDeviceTimedRecords](#) (out vector< [TimedRecord](#) > recs, in [peripheral.DeviceSlot_2_0_0](#) slot, in int recid, in int count)
Retrieve log records with timestamps for an peripheral device slot.
- [SensorSet](#) [getLoggedSensors](#) ()
Retrieve the set of logged sensors.
- int [setLoggedSensors](#) (in [SensorSet](#) sensors)
Change the set of logged sensors.
- void [enableAllSensors](#) ()
Enable logging for all PDU sensors.
- void [disableAllSensors](#) ()
Disable logging for all PDU sensors.
- time [getSensorSetTimestamp](#) ()
Get the time of the last sensor set modification.
- int [getLogRow](#) (out [LogRow](#) row, in int recid)
Get one full log row.

Public Attributes

- valueobject [SettingsChangedEvent](#): [event.UserEvent](#) { [Settings](#) oldSettings
Event: Sensor logger settings have been changed.
- [Settings](#) [newSettings](#)
Settings after change.
- valueobject [LoggedSensorsChangedEvent](#): [event.UserEvent](#) { [SensorSet](#) oldSensors
Event: Set of logged sensors has been changed.
- [SensorSet](#) [newSensors](#)
Sensor set after change.
- constant int [STATE_UNAVAILABLE](#) = 0
Sensor state in log record.
- constant int [STATE_OPEN](#) = 1
Circuit breaker open.
- constant int [STATE_CLOSE](#) = 2
Circuit breaker closed.
- constant int [STATE_BELOW_LOWER_CRITICAL](#) = 3
Numeric sensor below lower critical threshold.

- constant int `STATE_BELOW_LOWER_WARNING` = 4
Numeric sensor below lower warning threshold.
- constant int `STATE_NORMAL` = 5
Numeric sensor in normal range; normal operation.
- constant int `STATE_ABOVE_UPPER_WARNING` = 6
Numeric sensor above upper warning threshold.
- constant int `STATE_ABOVE_UPPER_CRITICAL` = 7
Numeric sensor above upper critical threshold.
- constant int `STATE_ON` = 8
Power state on.
- constant int `STATE_OFF` = 9
Power state off.
- constant int `STATE_ALARMED` = 10
Alarmed.
- constant int `STATE_OK` = 11
OK.
- constant int `STATE_MARGINAL` = 12
Marginal.
- constant int `STATE_FAIL` = 13
Fail.
- constant int `STATE_YES` = 14
Yes.
- constant int `STATE_NO` = 15
No.
- constant int `STATE_STANDBY` = 16
Standby operation.
- constant int `STATE_ONE` = 17
First source active.
- constant int `STATE_TWO` = 18
Second source active.
- constant int `STATE_IN_SYNC` = 19
Phases are in sync.
- constant int `STATE_OUT_OF_SYNC` = 20
Phases are out of sync.
- constant int `STATE_FAULT` = 21
Fault.
- constant int `STATE_SELF_TEST` = 22
Sensor is currently testing itself.
- constant int `STATE_I1_OPEN_FAULT` = 23
Inlet 1 switch open fault.
- constant int `STATE_I1_SHORT_FAULT` = 24
Inlet 1 switch short fault.
- constant int `STATE_I2_OPEN_FAULT` = 25
Inlet 2 switch open fault.
- constant int `STATE_I2_SHORT_FAULT` = 26
Inlet 2 switch short fault.
- constant int `STATE_WARNING` = 27
Warning.
- constant int `STATE_CRITICAL` = 28
Critical.

9.111.1 Detailed Description

Sensor logger interface.

This is a very specific interface to fulfill the the sensor logging requirements as specified by SNMP-MIB. That is reason why sensor logging is not specified along with a sensor but with this special service.

9.111.2 Member Function Documentation

9.111.2.1 `SensorSet sensors::Logger_2_1_2::getLoggedSensors ( )`

Retrieve the set of logged sensors.

Returns

Set of logged sensors

9.111.2.2 `int sensors::Logger_2_1_2::getLogRow ( out LogRow row, in int recid )`

Get one full log row.

Parameters

<i>row</i>	Result: Log row
<i>recid</i>	Record id

Returns

0 if OK
1 if the record id is invalid

9.111.2.3 `int sensors::Logger_2_1_2::getPeripheralDeviceRecords ( out vector< Record > recs, in peripheral.DeviceSlot_2_0_0 slot, in int recid, in int count )`

Retrieve log records for an peripheral device slot.

Parameters

<i>recs</i>	Result: Sensor log records
<i>slot</i>	Peripheral device slot reference
<i>recid</i>	First record id
<i>count</i>	Number of records

Returns

0 if OK
1 if any record id is invalid

9.111.2.4 `int sensors::Logger_2_1_2::getPeripheralDeviceTimedRecords ( out vector< TimedRecord > recs, in peripheral.DeviceSlot_2_0_0 slot, in int recid, in int count )`

Retrieve log records with timestamps for an peripheral device slot.

Parameters

<i>recs</i>	Result: Sensor log records
<i>slot</i>	Peripheral device slot reference
<i>recid</i>	First record id
<i>count</i>	Number of records

Returns

- 0 if OK
- 1 if any record id is invalid

9.111.2.5 `int sensors::Logger_2_1_2::getSensorRecords ( out vector< Record > recs, in sensors.Sensor_4_0_0 sensor, in int recid, in int count )`

Retrieve log records for a given sensor.

Parameters

<i>recs</i>	Result: Sensor log records
<i>sensor</i>	Sensor reference
<i>recid</i>	First record id
<i>count</i>	Number of records

Returns

- 0 if OK
- 1 if any record id is invalid

9.111.2.6 `time sensors::Logger_2_1_2::getSensorSetTimestamp ( )`

Get the time of the last sensor set modification.

This can be used by clients which keep a cached copy of the sensor set to determine whether that copy is still up-to-date.

Returns

Sensor set time stamp

9.111.2.7 `int sensors::Logger_2_1_2::getSensorTimedRecords ( out vector< TimedRecord > recs, in sensors.Sensor_4_0_0 sensor, in int recid, in int count )`

Retrieve log records with timestamps for a given sensor.

Parameters

<i>recs</i>	Result: Sensor log records
<i>sensor</i>	Sensor reference
<i>recid</i>	First record id
<i>count</i>	Number of records

Returns

0 if OK
1 if any record id is invalid

9.111.2.8 Settings sensors::Logger_2_1_2::getSettings ()

Retrieve the sensor logger settings.

Returns

Sensor logger settings

9.111.2.9 int sensors::Logger_2_1_2::getTimeStamps (out vector< time > *timestamps*, in int *recid*, in int *count*)

Retrieve a set of log record timestamps.

Parameters

<i>timestamps</i>	Result: Log record timestamps
<i>recid</i>	First record id
<i>count</i>	Number of records

Returns

0 if OK
1 if any record id is invalid

9.111.2.10 int sensors::Logger_2_1_2::setLoggedSensors (in SensorSet *sensors*)

Change the set of logged sensors.

Parameters

<i>sensors</i>	New set of sensors
----------------	--------------------

Returns

0 if OK
1 if any sensor in the list is unknown

9.111.2.11 int sensors::Logger_2_1_2::setSettings (in boolean *isEnabled*, in int *samplesPerRecord*)

Change the sensor logger settings.

Parameters

<i>isEnabled</i>	<code>true</code> to enable sensor logging
<i>samplesPer-Record</i>	Number of samples per log record

Returns

- 0 if OK
- 1 if any parameters are invalid

9.111.3 Member Data Documentation**9.111.3.1 valueobject sensors::Logger_2_1_2::LoggedSensorsChangedEvent**

Event: Set of logged sensors has been changed.

Sensor set before change

9.111.3.2 valueobject sensors::Logger_2_1_2::SettingsChangedEvent

Event: Sensor logger settings have been changed.

[Settings](#) before change

9.111.3.3 constant int sensors::Logger_2_1_2::STATE_UNAVAILABLE = 0

Sensor state in log record.

Unavailable

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/SensorLogger.idl

9.112 diag::DiagLogSettings::LogLevelEntry Struct Reference

An entry containing a context name and its associated context.

```
import "DiagLogSettings.idl";
```

Public Attributes

- string [ctxName](#)
log context name
- [LogLevel](#) [logLevel](#)
log level

9.112.1 Detailed Description

An entry containing a context name and its associated context.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/DiagLogSettings.idl

9.113 sensors::Logger_2_1_2::LogRow Struct Reference

One full log row.

```
import "SensorLogger.idl";
```

Public Attributes

- time [sensorSetTimestamp](#)
Time of last sensor set modification.
- time [timestamp](#)
Log row time stamp.
- vector< [Record](#) > [sensorRecords](#)
Sensor records; same order as in [SensorSet::sensors](#).
- vector< [Record](#) > [peripheralDeviceRecords](#)
Peripheral device records; same order as in [SensorSet::slots](#).

9.113.1 Detailed Description

One full log row.

The documentation for this struct was generated from the following file:

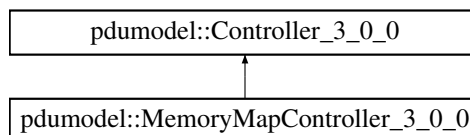
- pdu-json-rpc-api/idl/SensorLogger.idl

9.114 pdumodel::MemoryMapController_3_0_0 Interface Reference

Memory map controller.

```
import "MemoryMapController.idl";
```

Inheritance diagram for pdumodel::MemoryMapController_3_0_0:



Public Member Functions

- int [readMemory](#) (in int address, in int size, out vector< byte > memory)
Read a given number of bytes from an address.
- int [writeMemory](#) (in int address, in vector< byte > memory)
Write bytes to an address.

Additional Inherited Members

9.114.1 Detailed Description

Memory map controller.

9.114.2 Member Function Documentation

9.114.2.1 int pdumodel::MemoryMapController_3_0_0::readMemory (in int *address*, in int *size*, out vector< byte > *memory*)

Read a given number of bytes from an address.

Parameters

<i>address</i>	The address to read
<i>size</i>	The number of bytes to read
<i>memory</i>	The vector with the read bytes

Returns

0 if OK
an error code otherwise

9.114.2.2 `int pdumodel::MemoryMapController_3_0_0::writeMemory ( in int address, in vector< byte > memory )`

Write bytes to an address.

Parameters

<i>address</i>	The address to write
<i>memory</i>	The vector with the bytes to write

Returns

0 if OK
an error code otherwise

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/MemoryMapController.idl

9.115 sensors::NumericSensor_4_0_0::MetaData Struct Reference

Numeric sensor metadata.

```
import "NumericSensor.idl";
```

Public Attributes

- [Sensor_4_0_0 TypeSpec type](#)
Sensor type, reading type and unit.
- int [decdigits](#)
Number of significant decimal digits.
- float [accuracy](#)
Sensor accuracy in percent.
- float [resolution](#)
Sensor resolution.
- float [tolerance](#)
Sensor tolerance.
- float [noiseThreshold](#)
Sensor noise threshold.
- [Range range](#)
Range of possible sensor readings.
- [ThresholdCapabilities thresholdCaps](#)
Threshold capabilities.

9.115.1 Detailed Description

Numeric sensor metadata.

9.115.2 Member Data Documentation

9.115.2.1 `float sensors::NumericSensor_4_0_0::MetaData::accuracy`

Sensor accuracy in percent.

How close in percent measurement is to actual value. This value has an implicit precision of 2, i.e. the double value must be rounded for 2 decimal digits before use. For example a reading of 10.0 and an accuracy of 0.2 means the actual reading value is 10.0 +/- 0.2%.

A value of 0 means unused.

9.115.2.2 `int sensors::NumericSensor_4_0_0::MetaData::decdigits`

Number of significant decimal digits.

Indicates how many digits should be displayed to the right of the decimal point. I.e. double values must be rounded with this precision.

9.115.2.3 `float sensors::NumericSensor_4_0_0::MetaData::noiseThreshold`

Sensor noise threshold.

Threshold under which sensor measurements will be ignored. Sensor measurements below that value will be reported at the lower bound of the sensor range.

9.115.2.4 `Range sensors::NumericSensor_4_0_0::MetaData::range`

[Range](#) of possible sensor readings.

[Range](#) values are rounded with with decimal digits.

9.115.2.5 `float sensors::NumericSensor_4_0_0::MetaData::resolution`

Sensor resolution.

Minimum difference between any two measured values. Must be rounded with decimal digits.

9.115.2.6 `float sensors::NumericSensor_4_0_0::MetaData::tolerance`

Sensor tolerance.

Tolerance is given in +/- counts of the reading value. It indicates a constant magnitude possible error in the quantization of an analog input to the sensor. Rounded with decimal digits + 1.

A value of 0 means unused.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/NumericSensor.idl

9.116 pdumodel::Unit_2_0_1::MetaData Struct Reference

Unit metadata

```
import "Unit.idl";
```

Public Attributes

- boolean [hasOrientationSensor](#)
true if a tilt sensor is present
- vector< [Orientation](#) > [supportedDisplayOrientations](#)
Supported display orientations.

9.116.1 Detailed Description

Unit metadata

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Unit.idl

9.117 pdumodel::Outlet_1_5_6::MetaData Struct Reference

Outlet metadata

```
import "Outlet.idl";
```

Public Attributes

- string [label](#)
Outlet label
- string [receptacleType](#)
Receptacle type.
- [Nameplate](#) [namePlate](#)
Nameplate information
- [Rating](#) [rating](#)
Numerical usage ratings.
- boolean [isSwitchable](#)
true if the outlet is switchable
- boolean [isLatching](#)
true if the outlet is able to keep its state after power loss
- int [maxRelayCycleCnt](#)
Maximum relay cycle count.

9.117.1 Detailed Description

Outlet metadata

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Outlet.idl

9.118 pdumodel::OverCurrentProtector_2_1_2::MetaData Struct Reference

Overcurrent protector metadata.

```
import "OverCurrentProtector.idl";
```

Public Attributes

- string [label](#)
OCP label.
- Nameplate [namePlate](#)
Nameplate information
- Rating [rating](#)
Numerical usage ratings.
- Type [type](#)
OCP type.
- int [maxTripCnt](#)
Maximum trip count.

9.118.1 Detailed Description

Overcurrent protector metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/OverCurrentProtector.idl

9.119 pdumodel::Controller_3_0_0::MetaData Struct Reference

Slave controller metadata.

```
import "Controller.idl";
```

Public Attributes

- Type [type](#)
Controller type
- string [address](#)
Slave address.
- int [magic](#)
Magic code.
- boolean [versionAvailable](#)
true if version information and serial number is available
- int [fwAppVersion](#)
Firmware application version; 0 if unavailable.
- int [fwBootVersion](#)
Firmware bootloader version; 0 if unavailable.
- int [hwVersion](#)
Hardware version; 0 if unavailable.
- string [serial](#)
Serial number; empty if unavailable.

- boolean [haveResetCnt](#)
true if controller reset counter is available
- boolean [haveEmResetCnt](#)
true if energy meter reset counter is available

9.119.1 Detailed Description

Slave controller metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Controller.idl

9.120 pdumodel::Pdu_3_0_0::MetaData Struct Reference

PDU metadata.

```
import "Pdu.idl";
```

Public Attributes

- [Nameplate nameplate](#)
Nameplate information
- string [ctrlBoardSerial](#)
Main controller serial number.
- string [hwRevision](#)
Hardware revision.
- string [fwRevision](#)
Firmware revision.
- string [macAddress](#)
MAC address.
- boolean [hasSwitchableOutlets](#)
true if at least one outlet is switchable
- boolean [hasMeteredOutlets](#)
true if at least one outlet is metered
- boolean [hasLatchingOutletRelays](#)
true if at least one outlet has a latching relay
- boolean [isInlineMeter](#)
true if all inlets have exactly one outlet

9.120.1 Detailed Description

PDU metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Pdu.idl

9.121 pdumodel::Inlet_1_2_6::MetaData Struct Reference

Inlet metadata

```
import "Inlet.idl";
```

Public Attributes

- string [label](#)
Inlet label
- string [plugType](#)
Plug type.
- [Nameplate](#) [namePlate](#)
Nameplate information
- [Rating](#) [rating](#)
Numerical usage ratings.

9.121.1 Detailed Description

Inlet metadata

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Inlet.idl

9.122 peripheral::DeviceManager_2_0_0::MetaData Struct Reference

Peripheral DeviceManager's metadata.

```
import "PeripheralDeviceManager.idl";
```

Public Attributes

- int [oneWirePortCount](#)
Number of 1-wire ports.
- int [onboardDeviceCount](#)
Number of onboard peripheral devices.

9.122.1 Detailed Description

Peripheral DeviceManager's metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceManager.idl

9.123 Ihxmodel::Lhx_3_2_1::MetaData Struct Reference

LHX metadata.

```
import "Lhx.idl";
```

Public Attributes

- string [model](#)
The LHX model (e.g. "LHX 20")
- string [version](#)

The LHX firmware version, empty if not available.

- [ParamCfg setpointWaterValveCfg](#)

Water valve configuration characteristics.

- [ParamCfg setpointVentilatorsCfg](#)

Ventilators configuration characteristics.

- [ParamCfg defaultFanSpeedCfg](#)

Default fan speed configuration characteristics.

9.123.1 Detailed Description

LHX metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Lhx.idl

9.124 Ihxmodel::Parameter_2_0_1::MetaData Struct Reference

Parameter Metadata.

```
import "LhxParameter.idl";
```

Public Attributes

- [Unit unit](#)

Parameter unit.

- [string id](#)

Parameter ID.

- [double defaultValue](#)

Default value.

- [double min](#)

Minimum value.

- [double max](#)

Maximum value.

- [boolean read_only](#)

true for read-only parameters

- [int decDigits](#)

Number of decimal digits.

9.124.1 Detailed Description

Parameter Metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LhxParameter.idl

9.125 Ihxmodel::Sensor_4_0_0::MetaData Struct Reference

Sensor's self describing data.

```
import "LhxSensor.idl";
```

Public Attributes

- sensors [Sensor_4_0_0 TypeSpec type](#)
Sensor type, reading type and unit.
- int [numDecDigits](#)
Number of significant decimal digits.
- double [numRangeMin](#)
Smallest possible Numeric [Reading](#) Value.
- double [numRangeMax](#)
Largest possible Numeric [Reading](#) Value.
- double [numThresholdMin](#)
Smallest possible Numeric [Reading](#) Threshold Value.
- double [numThresholdMax](#)
Largest possible Numeric [Reading](#) Threshold Value.
- string [label](#)
The sensor label.
- string [id](#)
Descriptive ID of sensor containing label.

9.125.1 Detailed Description

Sensor's self describing data.

9.125.2 Member Data Documentation

9.125.2.1 int `lhxmodel::Sensor_4_0_0::MetaData::numDecDigits`

Number of significant decimal digits.

Indicates how many digits should be displayed to the right of the decimal point. I.e. double values must be rounded with this precision.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LhxSensor.idl

9.126 pdumodel::Ade::MetaData Struct Reference

ADE metadata.

```
import "Ade.idl";
```

Public Attributes

- string [adeType](#)
ADE chip model.
- int [channels](#)
Number of channels.
- double [currentDivider](#)
Divider for converting raw readings to Amperes.
- double [voltageDivider](#)
Divider for converting raw readings to Volts.
- double [energyDivider](#)
Divider for converting raw readings to Wh/VAh.

9.126.1 Detailed Description

ADE metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Ade.idl

9.127 smartcard::CardReader::MetaData Struct Reference

Reader Metadata.

```
import "CardReader.idl";
```

Public Attributes

- string [manufacturer](#)
manufacturer
- string [product](#)
product
- string [serialNumber](#)
serial number

9.127.1 Detailed Description

Reader Metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/CardReader.idl

9.128 pdumodel::TransferSwitch_3_1_1::MetaData Struct Reference

Transfer switch metadata.

```
import "TransferSwitch.idl";
```

Public Attributes

- string [label](#)
Transfer switch label.
- [Nameplate](#) [namePlate](#)
Nameplate information
- [Rating](#) [rating](#)
Numerical usage ratings.
- [Type](#) [type](#)
Transfer switch type.
- int [sourceCount](#)
Number of sources.

9.128.1 Detailed Description

Transfer switch metadata.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TransferSwitch.idl

9.129 devsettings::Modbus Interface Reference

Modbus service settings interface

```
import "Modbus.idl";
```

Classes

- struct [Settings](#)
Modbus service settings
- struct [TcpSettings](#)
Modbus/TCP settings.

Public Member Functions

- [Settings](#) [getSettings](#) ()
Retrieve the Modbus service settings.
- void [setSettings](#) (in [Settings](#) settings)
Set the Modbus service settings.

9.129.1 Detailed Description

Modbus service settings interface

9.129.2 Member Function Documentation

9.129.2.1 [Settings](#) devsettings::Modbus::getSettings ()

Retrieve the Modbus service settings.

Returns

[Modbus](#) service settings

9.129.2.2 void devsettings::Modbus::setSettings (in [Settings](#) settings)

Set the Modbus service settings.

Parameters

<i>settings</i>	New settings
-----------------	--------------

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Modbus.idl

9.130 modelpush::ModelPush Interface Reference

Model push service settings interface.

```
import "ModelPush.idl";
```

Classes

- struct [Configuration](#)
Model push service configuration.

Public Member Functions

- [Configuration](#) [getConfiguration](#) ()
Retrieve the model push service configuration.
- int [setConfiguration](#) (in [Configuration](#) cfg)
Set the model push service configuration.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.

9.130.1 Detailed Description

Model push service settings interface.

9.130.2 Member Function Documentation

9.130.2.1 [Configuration](#) modelpush::ModelPush::getConfiguration ()

Retrieve the model push service configuration.

Returns

Model push service configuration

9.130.2.2 int modelpush::ModelPush::setConfiguration (in [Configuration](#) cfg)

Set the model push service configuration.

Parameters

cfg	New model push service configuration
---------------------	--------------------------------------

Returns

0 if OK
1 if any parameters are invalid

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/ModelPush.idl

9.131 pdumodel::Nameplate Struct Reference

Component nameplate information.

```
import "Nameplate.idl";
```

Classes

- struct [Rating](#)
Component ratings.

Public Attributes

- string [manufacturer](#)
Component manufacturer.
- string [model](#)
Component model.
- string [partNumber](#)
Part number.
- string [serialNumber](#)
Serial number.
- [Rating](#) [rating](#)
Ratings.
- string [imageFileURL](#)
URL to component image.

9.131.1 Detailed Description

Component nameplate information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Nameplate.idl

9.132 jsonrpc::NameService Interface Reference

RPC Object Name Service.

```
import "NameService.idl";
```

Public Member Functions

- Object [lookup](#) (in string name)
Find an RPC object by its resource ID or well-known name.

9.132.1 Detailed Description

RPC Object Name Service.

9.132.2 Member Function Documentation

9.132.2.1 Object jsonrpc::NameService::lookup (in string *name*)

Find an RPC object by its resource ID or well-known name.

Parameters

<i>name</i>	Resource id or well-known name
-------------	--------------------------------

Returns

Object reference; `nil` if the name can't be resolved

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/NameService.idl

9.133 net::Net_2_0_2 Interface Reference

Network configuration interface.

```
import "Net.idl";
```

Public Member Functions

- int [setNetworkConfigIP](#) (in [NetworkConfigIP](#) cfg)
Set common IP parameters.
- void [getNetworkConfigIP](#) (out [NetworkConfigIP](#) cfg)
Retrieve the common IP parameters.
- int [setNetworkConfigIPv4](#) (in [NetworkConfigIPv4](#) cfg4)
Set the IPv4 configuration.
- void [getNetworkConfigIPv4](#) (out [NetworkConfigIPv4](#) cfg4, out [NetworkConfigIPv4](#) cfg4current)
Retrieve the IPv4 configuration.
- void [getNetworkConfigRoutesIPv4](#) (out vector< [IPv4RoutingEntry](#) > static_routes, out vector< [IPv4RoutingEntry](#) > active_routes)
- int [setNetworkConfigRoutesIPv4](#) (in vector< [IPv4RoutingEntry](#) > static_routes)
- void [getNetworkConfigRoutesIPv6](#) (out vector< [IPv6RoutingEntry](#) > static_routes, out vector< [IPv6RoutingEntry](#) > active_routes)
- int [setNetworkConfigRoutesIPv6](#) (in vector< [IPv6RoutingEntry](#) > static_routes)
- int [setNetworkConfigIPv6](#) (in [NetworkConfigIPv6](#) cfg6)
Set the IPv6 configuration.
- void [getNetworkConfigIPv6](#) (out [NetworkConfigIPv6](#) cfg6, out [NetworkActiveValuesIPv6](#) ipv6current)
Retrieve the IPv6 configuration.
- int [setNetworkConfigServices](#) (in vector< [ServiceConfig](#) > services)
Change the network service configuration.
- void [getNetworkConfigServices](#) (out vector< [ServiceConfig](#) > services)
Retrieve the network service configuration.

- void [getNetworkConfigInterface](#) (out [InterfaceState_2_0_0](#) state, out [LanInterfaceSettings](#) lan, out [LanInterfaceParameters_2_0_0](#) lancurrent, out [WirelessInterfaceSettings](#) wlan)
Retrieve the current LAN interface configuration.
- void [getMACs](#) (out [InterfaceState_2_0_0](#) state, out string ethmac, out string wlanmac)
Get MAC-Addresses of lan interfaces.
- int [setNetworkConfigLan](#) (in [LanInterfaceSettings](#) lancfg)
Set the LAN interface configuration, enable wired networking.
- int [setNetworkConfigWlan](#) (in [WirelessInterfaceSettings](#) wlancfg)
Set the wireless configuration, enable wireless networking.
- int [getBridgeSlaveCount](#) ()
Get the number of slave units that are directly sharing this unit's network connection.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.

9.133.1 Detailed Description

Network configuration interface.

9.133.2 Member Function Documentation

9.133.2.1 int net::Net_2_0_2::getBridgeSlaveCount ()

Get the number of slave units that are directly sharing this unit's network connection.

Returns

Number of slaves directly connected to this unit.

9.133.2.2 void net::Net_2_0_2::getMACs (out [InterfaceState_2_0_0](#) state, out string ethmac, out string wlanmac)

Get MAC-Addresses of lan interfaces.

If wlan is not enabled or present, wlanmac may be set to empty string

9.133.2.3 void net::Net_2_0_2::getNetworkConfigInterface (out [InterfaceState_2_0_0](#) state, out [LanInterfaceSettings](#) lan, out [LanInterfaceParameters_2_0_0](#) lancurrent, out [WirelessInterfaceSettings](#) wlan)

Retrieve the current LAN interface configuration.

Parameters

<i>state</i>	Result: Wired or wireless interface?
<i>lan</i>	Result: Wired interface settings
<i>lancurrent</i>	Result: Current wired interface parameters
<i>wlan</i>	Result: Wireless interface settings

9.133.2.4 void net::Net_2_0_2::getNetworkConfigIP (out NetworkConfigIP *cfg*)

Retrieve the common IP parameters.

Parameters

<i>cfg</i>	Result: Configured IP settings
------------	--------------------------------

9.133.2.5 void net::Net_2_0_2::getNetworkConfigIPv4 (out NetworkConfigIPv4 *cfg4*, out NetworkConfigIPv4 *cfg4current*)

Retrieve the IPv4 configuration.

Parameters

<i>cfg4</i>	Result: Configured IPv4 settings
<i>cfg4current</i>	Result: Active values (e.g. in case of DHCP)

9.133.2.6 void net::Net_2_0_2::getNetworkConfigIPv6 (out NetworkConfigIPv6 *cfg6*, out NetworkActiveValuesIPv6 *ipv6current*)

Retrieve the IPv6 configuration.

Parameters

<i>cfg6</i>	Result: Configured IPv6 settings
<i>ipv6current</i>	Result: Active values (e.g. in case of DHCP, stateless config)

9.133.2.7 void net::Net_2_0_2::getNetworkConfigServices (out vector< ServiceConfig > *services*)

Retrieve the network service configuration.

Parameters

<i>services</i>	List of all supported network services
-----------------	--

9.133.2.8 int net::Net_2_0_2::setNetworkConfigIP (in NetworkConfigIP *cfg*)

Set common IP parameters.

Parameters

<i>cfg</i>	New IP settings
------------	-----------------

Returns

- 0 if OK
- 1 if any parameters were invalid

9.133.2.9 int net::Net_2_0_2::setNetworkConfigIPv4 (in NetworkConfigIPv4 *cfg4*)

Set the IPv4 configuration.

Parameters

<i>cfg4</i>	New IPv4 settings
-------------	-------------------

Returns

- 0 if OK
- 1 if any parameters were invalid

9.133.2.10 `int net::Net_2_0_2::setNetworkConfigIPv6 ( in NetworkConfigIPv6 cfg6 )`

Set the IPv6 configuration.

Parameters

<i>cfg6</i>	New IPv6 settings
-------------	-------------------

Returns

- 0 if OK
- 1 if any parameters were invalid

9.133.2.11 `int net::Net_2_0_2::setNetworkConfigLan ( in LanInterfaceSettings lancfg )`

Set the LAN interface configuration, enable wired networking.

Parameters

<i>lancfg</i>	New LAN interface settings
---------------	----------------------------

Returns

- 0 if OK
- 1 if any parameters were invalid

9.133.2.12 `int net::Net_2_0_2::setNetworkConfigServices ( in vector< ServiceConfig > services )`

Change the network service configuration.

This call changes the configuration of one or more network services identified by name. Other services are not affected. The resulting configuration of all enabled services must be consistent, i.e. there must be not port collisions.

Parameters

<i>services</i>	List of network services to be changed
-----------------	--

Returns

- 0 if OK
- 1 if any parameters were invalid

9.133.2.13 `int net::Net_2_0_2::setNetworkConfigWLAN ( in WirelessInterfaceSettings wlancfg )`

Set the wireless configuration, enable wireless networking.

Parameters

<i>wlancfg</i>	New wireless interface settings
----------------	---------------------------------

Returns

- 0 if OK
- 1 if any parameters were invalid

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.134 net::NetworkActiveValuesIPv6 Struct Reference

Device IPv6 active values.

```
import "Net.idl";
```

Public Attributes

- boolean [enabled](#)
IPv6 enabled.
- [AutoConfigs autocfg](#)
Automatic configuration protocol.
- vector< string > [ipaddrs](#)
List of active IPv6 addresses / Prefix Length.
- vector< [IPv6RoutingEntry](#) > [routes](#)
List of active IPv6 routes.
- boolean [ra_managed](#)
Managed flag set in RAs.
- boolean [ra_otherconf](#)
Otherconf flag set in RAs.
- vector< string > [dns_suffixes](#)
List of DNS domain suffixes.
- string [dns_ip_1](#)
Primary nameserver IP.
- string [dns_ip_2](#)
Secondary nameserver IP.

9.134.1 Detailed Description

Device IPv6 active values.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.135 net::NetworkConfigIP Struct Reference

Device IP configuration.

```
import "Net.idl";
```

Public Attributes

- boolean [gai_prefer_ipv6](#)
getaddrinfo prefers IPv6 addresses

9.135.1 Detailed Description

Device IP configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.136 net::NetworkConfigIPv4 Struct Reference

Device IPv4 configuration.

```
import "Net.idl";
```

Public Attributes

- boolean [enabled](#)
IPv4 enabled.
- [AutoConfigs](#) [autocfg](#)
Automatic configuration protocol.
- string [ipaddr](#)
Device IP address.
- string [netmask](#)
Network mask.
- string [gateway](#)
Gateway IP address.
- string [hostname](#)
Device hostname.
- vector< string > [dns_suffixes](#)
List of DNS domain suffixes.
- boolean [override_dns](#)
Override nameserver information from DHCP.
- string [dns_ip_1](#)
Primary nameserver IP.
- string [dns_ip_2](#)
Secondary nameserver IP.
- string [domain_name](#)
Domain name.

9.136.1 Detailed Description

Device IPv4 configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.137 net::NetworkConfigIPv6 Struct Reference

Device IPv6 configuration.

```
import "Net.idl";
```

Public Attributes

- boolean [enabled](#)
IPv6 enabled.
- [AutoConfigs autocfg](#)
Automatic configuration protocol.
- string [ipaddr](#)
Device IPv6 address / Prefix Length.
- string [gateway](#)
Gateway IP address.
- string [hostname](#)
Device hostname.
- vector< string > [dns_suffixes](#)
List of DNS domain suffixes.
- boolean [override_dns](#)
Override nameserver information from DHCP.
- string [dns_ip_1](#)
Primary nameserver IP.
- string [dns_ip_2](#)
Secondary nameserver IP.
- string [domain_name](#)
Domain name.

9.137.1 Detailed Description

Device IPv6 configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.138 datetime::DateTime_2_0_0::NtpCfg Struct Reference

NTP server configuration.

```
import "DateTime.idl";
```

Public Attributes

- boolean [forceStatic](#)
Enforce use of static NTP servers.
- string [server1](#)
Primary NTP server.
- string [server2](#)
Secondary NTP server.

9.138.1 Detailed Description

NTP server configuration.

The documentation for this struct was generated from the following file:

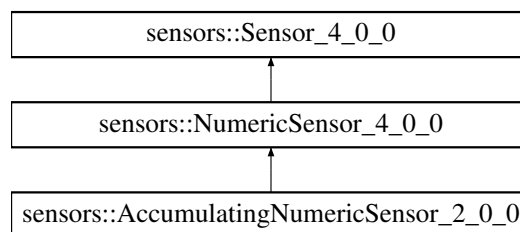
- pdu-json-rpc-api/idl/DateTime.idl

9.139 sensors::NumericSensor_4_0_0 Interface Reference

A sensor with numeric readings.

```
import "NumericSensor.idl";
```

Inheritance diagram for sensors::NumericSensor_4_0_0:



Classes

- struct [MetaData](#)
Numeric sensor metadata.
- struct [Range](#)
Range of possible sensor readings.
- struct [Reading](#)
Numeric sensor reading.
- struct [ThresholdCapabilities](#)
Threshold capabilities.
- struct [Thresholds](#)
Numeric sensor thresholds.

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the sensor metadata.
- [Thresholds](#) [getDefaultThresholds](#) ()
Retrieve the sensor default thresholds.
- [Thresholds](#) [getThresholds](#) ()
Retrieve the active thresholds.
- int [setThresholds](#) (in [Thresholds](#) thresh)
Change the active thresholds.
- [Reading](#) [getReading](#) ()
Get the sensor reading.

Public Attributes

- constant int [THRESHOLD_OUT_OF_RANGE](#) = 1
The threshold exceeds the sensor range.
- constant int [THRESHOLD_INVALID](#) = 2
The threshold constraints are not met.
- constant int [THRESHOLD_NOT_SUPPORTED](#) = 3
The sensor does not support setting this threshold.
- valueobject [ReadingChangedEvent](#): [idl.Event](#) { [Reading](#) newReading
Event: Numeric sensor reading has changed.
- valueobject [StateChangedEvent](#): [idl.Event](#) { [Reading](#) oldReading
Event: Sensor state has changed.
- [Reading](#) newReading
Reading after state change.
- valueobject [MetaDataChangedEvent](#): [idl.Event](#) { [MetaData](#) oldMetaData
Event: Sensor metadata has changed.
- [MetaData](#) newMetaData
Metadata after change.
- valueobject [ThresholdsChangedEvent](#): [event.UserEvent](#) { [Thresholds](#) oldThresholds
Event: Sensor thresholds have changed.
- [Thresholds](#) newThresholds
Threshold set after change.

Additional Inherited Members

9.139.1 Detailed Description

A sensor with numeric readings.

9.139.2 Member Function Documentation

9.139.2.1 Thresholds sensors::NumericSensor_4_0_0::getDefaultThresholds ()

Retrieve the sensor default thresholds.

Returns

Set of default thresholds

9.139.2.2 MetaData sensors::NumericSensor_4_0_0::getMetaData ()

Retrieve the sensor metadata.

Returns

Sensor metadata

9.139.2.3 Reading sensors::NumericSensor_4_0_0::getReading ()

Get the sensor reading.

Returns

Sensor reading

9.139.2.4 Thresholds sensors::NumericSensor_4_0_0::getThresholds ()

Retrieve the active thresholds.

Returns

Set of active thresholds

9.139.2.5 int sensors::NumericSensor_4_0_0::setThresholds (in Thresholds *thresh*)

Change the active thresholds.

Parameters

<i>thresh</i>	New set of thresholds
---------------	-----------------------

Returns

0 if OK
 THRESHOLD_OUT_OF_RANGE if any threshold is out of range
 THRESHOLD_INVALID if thresholds don't meet the requirements
 THRESHOLD_NOT_SUPPORTED if threshold is not supported

9.139.3 Member Data Documentation

9.139.3.1 valueobject sensors::NumericSensor_4_0_0::MetaDataChangedEvent

Event: Sensor metadata has changed.

Metadata before change

9.139.3.2 valueobject sensors::NumericSensor_4_0_0::ReadingChangedEvent

Event: Numeric sensor reading has changed.

New numeric sensor reading

9.139.3.3 valueobject sensors::NumericSensor_4_0_0::StateChangedEvent

Event: Sensor state has changed.

[Reading](#) before state change

9.139.3.4 valueobject sensors::NumericSensor_4_0_0::ThresholdsChangedEvent

Event: Sensor thresholds have changed.

Threshold set before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/NumericSensor.idl

9.140 Ihxmodel::Sensor_4_0_0::NumThresholds Struct Reference

Numerical sensor thresholds.

```
import "LhxSensor.idl";
```

Public Attributes

- boolean [lowerCriticalIsEnabled](#)
Lower critical threshold enabled.
- double [lowerCritical](#)
Lower critical threshold value.
- boolean [lowerWarningIsEnabled](#)
Lower warning threshold enabled.
- double [lowerWarning](#)
Lower warning threshold value.
- boolean [upperWarningIsEnabled](#)
Upper warning threshold enabled.
- double [upperWarning](#)
Upper warning threshold value.
- boolean [upperCriticalIsEnabled](#)
Upper critical threshold enabled.
- double [upperCritical](#)
Upper critical threshold value.
- double [hysteresis](#)
Deassertion hysteresis.

9.140.1 Detailed Description

Numerical sensor thresholds.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LhxSensor.idl

9.141 Ihxmodel::Lhx_3_2_1::OpState Struct Reference

LHX operational state.

```
import "Lhx.idl";
```

Public Attributes

- boolean [on](#)
LHX is switched on.
- [AlertStatus](#) [alertStatus](#)
Alert status of LHX controller.
- int [operatingHoursLhx](#)
Operating hours of Varistar LHX.
- vector< int > [operatingHoursFan](#)
Operating hours of Fans.

9.141.1 Detailed Description

LHX operational state.

The documentation for this struct was generated from the following file:

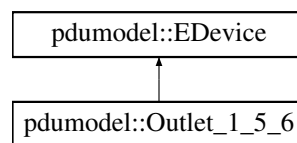
- pdu-json-rpc-api/idl/Lhx.idl

9.142 pdumodel::Outlet_1_5_6 Interface Reference

Outlet interface

```
import "Outlet.idl";
```

Inheritance diagram for pdumodel::Outlet_1_5_6:



Classes

- struct [LedState](#)
Outlet LED state
- struct [MetaData](#)
Outlet metadata
- struct [Sensors](#)
Outlet sensors
- struct [Settings](#)
Outlet settings
- struct [State](#)
Outlet state

Public Types

- enum [PowerState](#) { [PS_OFF](#), [PS_ON](#) }
Outlet power state.
- enum [StartupState](#) { [SS_ON](#), [SS_OFF](#), [SS_LASTKNOWN](#), [SS_PDUDEF](#) }
Outlet power state on device startup

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the outlet metadata.
- [Sensors](#) [getSensors](#) ()
Get the outlet sensors.
- [State](#) [getState](#) ()
Retrieve the outlet state.
- int [setPowerState](#) (in [PowerState](#) pstate)
Switch the outlet.

- int `cyclePowerState` ()
Power-cycle the outlet.
- Settings `getSettings` ()
Retrieve the outlet settings.
- int `setSettings` (in Settings settings)
Change the outlet settings.
- void `getIOP` (out Inlet_1_2_6 i, out OverCurrentProtector_2_1_2 o, out vector< Pole_2_0_0 > p)
Get inlet, overcurrent protector and poles.
- Controller_3_0_0 `getController` ()
Get the controller for this outlet.
- int `unstick` ()
Trigger an attempt to un-stick sticking relay contacts.

Public Attributes

- constant int `ERR_OUTLET_NOT_SWITCHABLE` = 1
Outlet is not switchable.
- constant int `ERR_LOAD_SHEDDING_ACTIVE` = 2
Load-shedding is enabled.
- constant int `ERR_OUTLET_DISABLED` = 3
Outlet is disabled.
- constant int `ERR_OUTLET_NOT_OFF` = 4
Outlet is on or in power-cycle; unstick not possible.
- constant int `ERR_INVALID_PARAM` = 1
Invalid parameters.
- valueobject `PowerControlEvent`: event.UserEvent { `PowerState` state
Event: Power control was initiated.
- boolean `cycle`
Whether the outlet was cycled.
- valueobject `StateChangedEvent`: idl.Event { `State` oldState
Event: Outlet state has changed.
- `State` `newState`
State after change.
- valueobject `SettingsChangedEvent`: event.UserEvent { `Settings` oldSettings
Event: Outlet settings have been changed.
- `Settings` `newSettings`
Settings after change.

9.142.1 Detailed Description

Outlet interface

9.142.2 Member Enumeration Documentation

9.142.2.1 enum pdumodel::Outlet_1_5_6::PowerState

Outlet power state.

Used both for switching and representing the current state

Enumerator

PS_OFF Switch off / Power is off.

PS_ON Switch on / Power is on.

9.142.2.2 enum pdumodel::Outlet_1_5_6::StartupState

Outlet power state on device startup

Enumerator

- SS_ON** Outlet will be switched on
- SS_OFF** Outlet will be switched off
- SS_LASTKNOWN** Last known power state will be restored.
- SS_PDUEF** Use default state as defined in PDU settings.

9.142.3 Member Function Documentation

9.142.3.1 int pdumodel::Outlet_1_5_6::cyclePowerState ()

Power-cycle the outlet.

Returns

- 0 if OK
- 1 if the outlet is not switchable
- 2 if switching is not possible due to load shedding
- 3 if the outlet is disabled

9.142.3.2 Controller_3_0_0 pdumodel::Outlet_1_5_6::getController ()

Get the controller for this outlet.

Returns

Slave controller reference

9.142.3.3 void pdumodel::Outlet_1_5_6::getIOP (out Inlet_1_2_6 *i*, out OverCurrentProtector_2_1_2 *o*, out vector< Pole_2_0_0 > *p*)

Get inlet, overcurrent protector and poles.

Parameters

<i>i</i>	Result: Inlet reference
<i>o</i>	Result: Overcurrent protector reference
<i>p</i>	Result: List of poles

9.142.3.4 MetaData pdumodel::Outlet_1_5_6::getMetaData ()

Retrieve the outlet metadata.

Returns

Outlet metadata

9.142.3.5 Sensors pdumodel::Outlet_1_5_6::getSensors ()

Get the outlet sensors.

Returns

Outlet sensors

9.142.3.6 Settings pdumodel::Outlet_1_5_6::getSettings ()

Retrieve the outlet settings.

Returns

Outlet settings

9.142.3.7 State pdumodel::Outlet_1_5_6::getState ()

Retrieve the outlet state.

Returns

Outlet state

9.142.3.8 int pdumodel::Outlet_1_5_6::setPowerState (in PowerState *pstate*)

Switch the outlet.

Parameters

<i>pstate</i>	New power state
---------------	-----------------

Returns

- 0 if OK
- 1 if the outlet is not switchable
- 2 if switching is not possible due to load shedding
- 3 if the outlet is disabled

9.142.3.9 int pdumodel::Outlet_1_5_6::setSettings (in Settings *settings*)

Change the outlet settings.

Parameters

<i>settings</i>	New outlet settings
-----------------	---------------------

Returns

- 0 if OK
- 1 if any parameters are invalid

9.142.3.10 int pdumodel::Outlet_1_5_6::unstick ()

Trigger an attempt to un-stick sticking relay contacts.

Tries repairing relay contacts that are stuck together due to wear by switching the relay in a certain pattern. Prior to running this method, the outlet must be in 'off' state to acknowledge that loads were disconnected.

Returns

- 0 if unsticking was triggered successfully
- 1 if outlet is not switchable
- 3 if the outlet is disabled
- 4 if relay is in a power cycle or on

9.142.4 Member Data Documentation

9.142.4.1 valueobject pdumodel::Outlet_1_5_6::PowerControlEvent

Event: Power control was initiated.

[State](#) the outlet was switched to (if cycle is false)

9.142.4.2 valueobject pdumodel::Outlet_1_5_6::SettingsChangedEvent

Event: Outlet settings have been changed.

[Settings](#) before change

9.142.4.3 valueobject pdumodel::Outlet_1_5_6::StateChangedEvent

Event: Outlet state has changed.

[State](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Outlet.idl

9.143 pdumodel::Pdu_3_0_0::OutletSequenceState Struct Reference

Outlet sequencing status

```
import "Pdu.idl";
```

Public Attributes

- boolean [sequenceRunning](#)
true if an outlet sequence is currently running
- int [nextOutletToSwitch](#)
Number (zero-based) of the next outlet in the sequence.
- int [timeUntilNextSwitch](#)
Time in milliseconds before the next outlet is switched.
- int [outletsRemaining](#)
Number of outlets remaining in the sequence.

9.143.1 Detailed Description

Outlet sequencing status

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Pdu.idl

9.144 pdumodel::OutletStatistic Struct Reference

Outlet statistics

```
import "Outlet.idl";
```

Public Attributes

- int [relayCycleCnt](#)
Relay switch count.
- int [relayFailCnt](#)
Relay failure count.

9.144.1 Detailed Description

Outlet statistics

The documentation for this struct was generated from the following file:

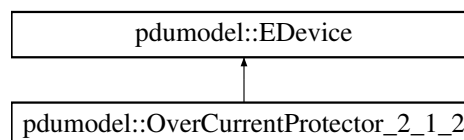
- pdu-json-rpc-api/idl/Outlet.idl

9.145 pdumodel::OverCurrentProtector_2_1_2 Interface Reference

Overcurrent protector interface.

```
import "OverCurrentProtector.idl";
```

Inheritance diagram for pdumodel::OverCurrentProtector_2_1_2:



Classes

- struct [MetaData](#)
Overcurrent protector metadata.
- struct [Sensors](#)
Overcurrent protector sensors.
- struct [Settings](#)
Overcurrent protector settings.

Public Types

- enum [Type](#) {
[BREAKER_1POLE](#), [BREAKER_2POLE](#), [BREAKER_3POLE](#), [FUSE](#),
[FUSE_PAIR](#), [RCBO_2POLE](#), [RCBO_3POLE](#), [RCBO_4POLE](#) }
Overcurrent protector type.

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the OCP metadata.
- [Sensors](#) [getSensors](#) ()
Get the OCP sensors.
- vector< [DoublePole_2_0_0](#) > [getPoles](#) ()
Get the list of OCP poles.
- [Inlet_1_2_6](#) [getInlet](#) ()
Get the inlet this OCP is connected to.
- [OverCurrentProtector_2_1_2](#) [getOCP](#) ()
Get parent OCP - next OCP going towards inlet (for cascaded OCPs).
- [Settings](#) [getSettings](#) ()
Retrieve the OCP settings.
- int [setSettings](#) (in [Settings](#) settings)
Change the OCP settings.

Public Attributes

- valueobject [SettingsChangedEvent](#): [event.UserEvent](#) { [Settings](#) oldSettings
Event: Overcurrent protector settings have been changed.
- [Settings](#) [newSettings](#)
[Settings](#) after change.

9.145.1 Detailed Description

Overcurrent protector interface.

9.145.2 Member Enumeration Documentation

9.145.2.1 enum pdumodel::OverCurrentProtector_2_1_2::Type

Overcurrent protector type.

Enumerator

[BREAKER_1POLE](#) Single-pole circuit breaker.

[BREAKER_2POLE](#) Two-pole circuit breaker.

[BREAKER_3POLE](#) Three-pole circuit breaker.

[FUSE](#) Fuse.

[FUSE_PAIR](#) Fuse Pair.

[RCBO_2POLE](#) Two-pole residual-current device including overcurrent protection.

[RCBO_3POLE](#) Three-pole residual-current device including overcurrent protection.

[RCBO_4POLE](#) Four-pole residual-current device including overcurrent protection.

9.145.3 Member Function Documentation

9.145.3.1 Inlet_1_2_6 pdumodel::OverCurrentProtector_2_1_2::getInlet ()

Get the inlet this OCP is connected to.

Returns

inlet

9.145.3.2 MetaData pdumodel::OverCurrentProtector_2_1_2::getMetaData ()

Retrieve the OCP metadata.

Returns

OCP metadata

9.145.3.3 OverCurrentProtector_2_1_2 pdumodel::OverCurrentProtector_2_1_2::getOCP ()

Get parent OCP - next OCP going towards inlet (for cascaded OCPs).

Returns

OCP or null

9.145.3.4 vector<DoublePole_2_0_0> pdumodel::OverCurrentProtector_2_1_2::getPoles ()

Get the list of OCP poles.

Returns

List of OCP poles

9.145.3.5 Sensors pdumodel::OverCurrentProtector_2_1_2::getSensors ()

Get the OCP sensors.

Returns

OCP sensors

9.145.3.6 Settings pdumodel::OverCurrentProtector_2_1_2::getSettings ()

Retrieve the OCP settings.

Returns

OCP settings

9.145.3.7 int pdumodel::OverCurrentProtector_2_1_2::setSettings (in Settings settings)

Change the OCP settings.

Parameters

<i>settings</i>	New OCP settings
-----------------	------------------

Returns

- 0 if OK
- 1 if any parameters are invalid

9.145.4 Member Data Documentation

9.145.4.1 valueobject pdumodel::OverCurrentProtector_2_1_2::SettingsChangedEvent

Event: Overcurrent protector settings have been changed.

[Settings](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/OverCurrentProtector.idl

9.146 peripheral::PackageInfo_2_0_0 Struct Reference

Peripheral device package information.

```
import "PeripheralDevicePackage.idl";
```

Classes

- struct [FirmwareInfo](#)
- struct [HardwareInfo](#)

Public Types

- enum [State](#) { [NORMAL](#), [FW_UPDATE](#), [INTERNAL_ERROR](#), [CONFIG_ERROR](#) }

Public Attributes

- [State](#) state
The peripheral device package operational state.
- vector< [PosElement](#) > [position](#)
Position within 1-wire topo.
- [HardwareInfo](#) hwInfo
Device package hardware specific information.
- [FirmwareInfo](#) fwInfo
Device package firmware specific information.

9.146.1 Detailed Description

Peripheral device package information.

9.146.2 Member Enumeration Documentation

9.146.2.1 enum peripheral::PackageInfo_2_0_0::State

Enumerator

- NORMAL** Device package is in normal operation.
- FW_UPDATE** Device package's firmware is being updated.
- INTERNAL_ERROR** Device package's internal error flag is set.
- CONFIG_ERROR** Device package's internal configuration is invalid.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDevicePackage.idl

9.147 Ihxmodel::Lhx_3_2_1::ParamCfg Struct Reference

Configuration parameter characteristics.

```
import "Lhx.idl";
```

Public Attributes

- double [min](#)
Minimum value.
- double [max](#)
Maximum value.
- int [decdigits](#)
Number of significant decimal digits.

9.147.1 Detailed Description

Configuration parameter characteristics.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Lhx.idl

9.148 Ihxmodel::Parameter_2_0_1 Interface Reference

LHX Parameter Interface.

```
import "LhxParameter.idl";
```

Classes

- struct [MetaData](#)
Parameter Metadata.
- struct [Status](#)
Parameter Status.
- struct [Value](#)
Parameter Value.

Public Types

- enum [Unit](#) {
[NONE](#), [NUMBER](#), [BINARY](#), [TEMP_ABS](#),
[TEMP_REL](#), [BAR](#), [PASCAL](#), [SIEMENS](#),
[METER](#), [VOLT](#), [AMPERE](#), [HOURS](#),
[MINUTES](#), [SECONDS](#), [TIME](#), [METERS_PER_SECOND](#),
[NEWTON](#), [GRAMMS](#), [HUMIDITY_REL](#), [HERTZ](#),
[OHM](#), [PERCENT](#), [LITERS_PER_MINUTE](#), [LITERS_PER_HOUR](#) }

Parameter Unit.

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the parameter metadata.
- [Value](#) [getValue](#) ()
Retrieve the parameter value.
- int [getRawValue](#) ()
Retrieve the parameter raw value.
- int [setRawValue](#) (in int rawValue, in boolean validateRange)
Change the parameter value.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.
- constant int [ERR_READ_ONLY](#) = 2
Attempt to write a read-only parameter.
- valueobject [MetaDataChangedEvent](#): [idl.Event](#) { [MetaData](#) oldMetaData
Event: Parameter metadata has been changed.
- [MetaData](#) [newMetaData](#)
Metadata after change.
- valueobject [ValueChangedEvent](#): [idl.Event](#) { [Value](#) newValue
Event: Parameter value has changed.

9.148.1 Detailed Description

LHX Parameter Interface.

9.148.2 Member Enumeration Documentation

9.148.2.1 enum [Ihxmodel::Parameter_2_0_1::Unit](#)

Parameter Unit.

Enumerator

- NONE*** No unit.
- NUMBER*** Number.
- BINARY*** Binary.
- TEMP_ABS*** Absolute temperature.

TEMP_REL Relative temperature.
BAR Bar.
PASCAL Pascal.
SIEMENS Siemens.
METER Meter.
VOLT Volt.
AMPERE Ampere.
HOURS Hours.
MINUTES Minutes.
SECONDS Seconds.
TIME Time.
METERS_PER_SECOND Meters/second.
NEWTON Newton.
GRAMMS Gramms.
HUMIDITY_REL Relative humidity.
HERTZ Hertz.
OHM Ohm.
PERCENT Percent.
LITERS_PER_MINUTE Liters/minute.
LITERS_PER_HOUR Liters/hour.

9.148.3 Member Function Documentation

9.148.3.1 `MetaData Ihxmodel::Parameter_2_0_1::getMetaData ( )`

Retrieve the parameter metadata.

Returns

Parameter metadata

9.148.3.2 `int Ihxmodel::Parameter_2_0_1::getRawValue ( )`

Retrieve the parameter raw value.

Returns

Raw value

9.148.3.3 `Value Ihxmodel::Parameter_2_0_1::getValue ( )`

Retrieve the parameter value.

Returns

Parameter value

9.148.3.4 int lhxmodel::Parameter_2_0_1::setRawValue (in int *rawValue*, in boolean *validateRange*)

Change the parameter value.

Parameters

<i>rawValue</i>	New value
<i>validateRange</i>	validate value against parameter range

Returns

0 if OK
 ERR_INVALID_PARAMS if any parameter value is invalid
 ERR_READ_ONLY if attempting to write read only parameter

9.148.4 Member Data Documentation

9.148.4.1 valueobject lhxmodel::Parameter_2_0_1::MetaDataChangedEvent

Event: Parameter metadata has been changed.

Metadata before change

9.148.4.2 valueobject lhxmodel::Parameter_2_0_1::ValueChangedEvent

Event: Parameter value has changed.

New parameter value

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/LhxParameter.idl

9.149 security::PasswordSettings Struct Reference

Password settings.

```
import "Security.idl";
```

Public Attributes

- boolean [enableAging](#)
true to enable password aging
- int [agingInterval](#)
Aging interval.
- boolean [enableStrongReq](#)
true to enable strong password requirements
- int [minPwLength](#)
Minimum password length.
- int [maxPwLength](#)
Maximum password length.
- boolean [enforceLower](#)
Passwords must contain at least one lower case character.
- boolean [enforceUpper](#)

- Passwords must contain at least one upper case character.*
- boolean [enforceNumeric](#)

Passwords must contain at least one numeric character.
- boolean [enforceSpecial](#)

Passwords must contain at least one special character.
- int [pwHistoryDepth](#)

Number of entries in password history.

9.149.1 Detailed Description

Password settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.150 pdumodel::Pdu_3_0_0 Interface Reference

Main PDU interface.

```
import "Pdu.idl";
```

Classes

- struct [MetaData](#)

PDU metadata.
- struct [OutletSequenceState](#)

Outlet sequencing status
- struct [Sensors](#)

PDU sensors.
- struct [Settings](#)

PDU settings.
- struct [Statistic](#)

PDU statistics.

Public Types

- enum [StartupState](#) { [SS_ON](#), [SS_OFF](#), [SS_LASTKNOWN](#) }

Outlet power state on device startup

Public Member Functions

- [Nameplate](#) [getNameplate](#) ()

Retrieve the PDU nameplate information.
- [MetaData](#) [getMetaData](#) ()

Retrieve the PDU metadata.
- [Sensors](#) [getSensors](#) ()

Retrieve the PDU sensors.
- sensors [Logger_2_1_2](#) [getSensorLogger](#) ()

Get the sensor logger.

- `vector< Controller\_3\_0\_0 > getControllers ()`
Get the list of slave controllers.
- `vector< Outlet\_1\_5\_6 > getOutlets ()`
Get the list of outlets.
- `vector`
`< OverCurrentProtector\_2\_1\_2 > getOverCurrentProtectors ()`
Get the list of overcurrent protectors.
- `vector< Inlet\_1\_2\_6 > getInlets ()`
Get the list of inlets.
- `vector< TransferSwitch\_3\_1\_1 > getTransferSwitches ()`
Returns list of Transfer Switches.
- `peripheral DeviceManager_2_0_0 getPeripheralDeviceManager ()`
Get the peripheral device manager.
- `hmi InternalBeeper getBeeper ()`
Get the built-in beeper, if there is any.
- `Settings getSettings ()`
Retrieve the PDU settings.
- `boolean isLoadSheddingActive ()`
Retrieve the current load shedding state.
- `int setSettings (in Settings settings)`
Change the PDU settings.
- `void setLoadSheddingActive (in boolean active)`
Enable or disable load shedding.
- `vector< portsmodel.Port\_2\_0\_1 > getFeaturePorts ()`
Get all feature ports of this device.
- `int enterRS485ConfigModeAndAssignCtrlBoardAddress (in int addr)`
Enter RS485 config mode and assign an address to a relay board.
- `int enterRS485ConfigModeAndAssignSCBoardAddress (in int deviceId, in int addr)`
Enter RS485 config mode and assign an address to a slave controller device with a given device ID.
- `int leaveRS485ConfigMode ()`
Leave RS485 config mode.
- `int setAllOutletPowerStates (in Outlet\_1\_5\_6.PowerState pstate)`
Switch all outlets.
- `int setMultipleOutletPowerStates (in vector< int > outletNumbers, in Outlet\_1\_5\_6.PowerState state, in boolean respectSequence)`
Switch multiple outlets.
- `int cycleAllOutletPowerStates ()`
Power-cycle all outlets.
- `int cycleMultipleOutletPowerStates (in vector< int > outletNumbers, in boolean respectSequence)`
Power-cycle multiple outlets.
- `Statistic getStatistic ()`
Retrieve PDU statistics.
- `OutletSequenceState getOutletSequenceState ()`
Retrieve the current outlet sequencing status.
- `void cancelOutletSequence ()`
Stop a currently running outlet sequence.

Public Attributes

- constant int [ERR_INVALID_PARAM](#) = 1
Invalid parameters.
- valueobject [SettingsChangedEvent](#): [event.UserEvent](#) { [Settings](#) oldSettings
Event: PDU settings have been changed.
- [Settings](#) newSettings
Settings after change.
- valueobject [LoadSheddingModeChangedEvent](#): [event.UserEvent](#) { boolean enabled
Event: Load shedding mode was enabled or disabled.
- valueobject [OutletSequenceStateChangedEvent](#): [idl.Event](#) { [OutletSequenceState](#) newState
Event: Outlet sequencing state has changed.

9.150.1 Detailed Description

Main PDU interface.

9.150.2 Member Enumeration Documentation

9.150.2.1 enum pdumodel::Pdu_3_0_0::StartupState

Outlet power state on device startup

Enumerator

- SS_ON** Outlet will be switched on
- SS_OFF** Outlet will be switched off
- SS_LASTKNOWN** Restore last known power state.

9.150.3 Member Function Documentation

9.150.3.1 int pdumodel::Pdu_3_0_0::cycleAllOutletPowerStates ()

Power-cycle all outlets.

Returns

- 0 if OK
- 1 if an outlet sequence is currently running

9.150.3.2 int pdumodel::Pdu_3_0_0::cycleMultipleOutletPowerStates (in vector< int > *outletNumbers*, in boolean *respectSequence*)

Power-cycle multiple outlets.

Parameters

<i>outletNumbers</i>	List of outlet numbers (zero-based)
<i>respect-Sequence</i>	true to switch in defined sequence order

Returns

- 0 if OK
- 1 if outlet sequence is currently running

9.150.3.3 `int pdumodel::Pdu_3_0_0::enterRS485ConfigModeAndAssignCtrlBoardAddress ( in int addr )`

Enter RS485 config mode and assign an address to a relay board.

Warning

This is dangerous! Do not use except for manufacturing.

Parameters

<i>addr</i>	New relay board address
-------------	-------------------------

Returns

- 0 if OK
- 1 if any parameters are invalid

9.150.3.4 `int pdumodel::Pdu_3_0_0::enterRS485ConfigModeAndAssignSCBoardAddress ( in int deviceld, in int addr )`

Enter RS485 config mode and assign an address to a slave controller device with a given device ID.

Warning

This is dangerous! Do not use except for manufacturing.

Parameters

<i>deviceld</i>	Device id of the slave controller board which is supposed to get the address
<i>addr</i>	New relay board address

Returns

- 0 if OK
- 1 if any parameters are invalid

9.150.3.5 `hmi InternalBeeper pdumodel::Pdu_3_0_0::getBeeper ( )`

Get the built-in beeper, if there is any.

Returns

Beeper interface

9.150.3.6 `vector<Controller_3_0_0> pdumodel::Pdu_3_0_0::getControllers ( )`

Get the list of slave controllers.

Returns

List of slave controllers

9.150.3.7 `vector<portsmodel.Port_2_0_1> pdumodel::Pdu_3_0_0::getFeaturePorts ( )`

Get all feature ports of this device.

This returns an entry for all feature ports, no matter whether something is connected or not. A device with n feature ports will return n entries here.

Returns

List of all Feature Ports

9.150.3.8 `vector<Inlet_1_2_6> pdumodel::Pdu_3_0_0::getInlets ( )`

Get the list of inlets.

Returns

List of inlets, indexed by their number (zero-based)

9.150.3.9 `MetaData pdumodel::Pdu_3_0_0::getMetaData ( )`

Retrieve the PDU metadata.

Returns

PDU metadata

9.150.3.10 `Nameplate pdumodel::Pdu_3_0_0::getNameplate ( )`

Retrieve the PDU nameplate information.

Returns

[Nameplate](#) information

9.150.3.11 `vector<Outlet_1_5_6> pdumodel::Pdu_3_0_0::getOutlets ( )`

Get the list of outlets.

Returns

List of outlets, indexed by their number (zero-based)

9.150.3.12 `OutletSequenceState pdumodel::Pdu_3_0_0::getOutletSequenceState ( )`

Retrieve the current outlet sequencing status.

Returns

Sequencing status

9.150.3.13 **vector<OverCurrentProtector_2_1_2>** pdumodel::Pdu_3_0_0::getOverCurrentProtectors ()

Get the list of overcurrent protectors.

Returns

List of OCPs, indexed by their number (zero-based)

9.150.3.14 **peripheral DeviceManager_2_0_0** pdumodel::Pdu_3_0_0::getPeripheralDeviceManager ()

Get the peripheral device manager.

Returns

Peripheral device manager

9.150.3.15 **sensors Logger_2_1_2** pdumodel::Pdu_3_0_0::getSensorLogger ()

Get the sensor logger.

Returns

Sensor logger reference

9.150.3.16 **Sensors** pdumodel::Pdu_3_0_0::getSensors ()

Retrieve the PDU sensors.

Returns

PDU sensors

9.150.3.17 **Settings** pdumodel::Pdu_3_0_0::getSettings ()

Retrieve the PDU settings.

Returns

PDU settings

9.150.3.18 **Statistic** pdumodel::Pdu_3_0_0::getStatistic ()

Retrieve PDU statistics.

Returns

PDU statistics

9.150.3.19 **vector<TransferSwitch_3_1_1>** pdumodel::Pdu_3_0_0::getTransferSwitches ()

Returns list of Transfer Switches.

This list may be empty.

9.150.3.20 boolean pdumodel::Pdu_3_0_0::isLoadSheddingActive ()

Retrieve the current load shedding state.

Returns

`true` if load shedding is currently enabled

9.150.3.21 int pdumodel::Pdu_3_0_0::leaveRS485ConfigMode ()

Leave RS485 config mode.

Returns

0 if OK

9.150.3.22 int pdumodel::Pdu_3_0_0::setAllOutletPowerStates (in Outlet_1_5_6.PowerState *pstate*)

Switch all outlets.

Parameters

<i>pstate</i>	New power state for all outlets
---------------	---------------------------------

Returns

0 if OK
1 if an outlet sequence is currently running

9.150.3.23 void pdumodel::Pdu_3_0_0::setLoadSheddingActive (in boolean *active*)

Enable or disable load shedding.

Parameters

<i>active</i>	<code>true</code> to enable, <code>false</code> to disable load shedding
---------------	--

9.150.3.24 int pdumodel::Pdu_3_0_0::setMultipleOutletPowerStates (in vector< int > *outletNumbers*, in Outlet_1_5_6.PowerState *state*, in boolean *respectSequence*)

Switch multiple outlets.

Parameters

<i>outletNumbers</i>	List of outlet numbers (zero-based)
<i>state</i>	New power state for all outlets in list
<i>respect-Sequence</i>	<code>true</code> to switch in defined sequence order

Returns

0 if OK
1 if an outlet sequence is currently running

9.150.3.25 int pdumodel::Pdu_3_0_0::setSettings (in Settings settings)

Change the PDU settings.

Parameters

<i>settings</i>	New PDU settings
-----------------	------------------

Returns

- 0 if OK
- 1 if any parameters are invalid

9.150.4 Member Data Documentation

9.150.4.1 valueobject pdumodel::Pdu_3_0_0::LoadSheddingModeChangedEvent

Event: Load shedding mode was enabled or disabled.

Whether load shedding mode is enabled after the change

9.150.4.2 valueobject pdumodel::Pdu_3_0_0::OutletSequenceStateChangedEvent

Event: Outlet sequencing state has changed.

New sequencing state

9.150.4.3 valueobject pdumodel::Pdu_3_0_0::SettingsChangedEvent

Event: PDU settings have been changed.

[Settings](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Pdu.idl

9.151 pdumodel::Bcm::PhaseConfig Struct Reference

Phase Configuration.

```
import "Bcm.idl";
```

Public Attributes

- [LineConfig](#) lineConfig
Power line configuration.
- [TransformerType](#) ctType
Current transformer type.
- int fsCurrent
Full-scale current (voltage-type transformer)
- int fsVoltage
Full-scale voltage (voltage-type transformer)
- int turnsRatio

Turns ratio (turns ratio transformer)

- int [burdenResistor](#)

Burden resistor (turns ratio transformer)

9.151.1 Detailed Description

Phase Configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Bcm.idl

9.152 pdumodel::Pole_2_0_0 Struct Reference

An inlet or outlet pole.

```
import "Pole.idl";
```

Public Attributes

- string [label](#)
Pole label
- [PowerLine](#) [line](#)
Power line.
- int [nodeId](#)
Circuit node id.
- sensors NumericSensor_4_0_0 [voltage](#)
RMS voltage sensor, L-L.
- sensors NumericSensor_4_0_0 [voltageLN](#)
RMS voltage sensor, L-N.
- sensors NumericSensor_4_0_0 [current](#)
RMS current sensor.
- sensors NumericSensor_4_0_0 [peakCurrent](#)
Peak current sensor.
- sensors NumericSensor_4_0_0 [activePower](#)
Active power sensor.
- sensors NumericSensor_4_0_0 [apparentPower](#)
Apparent power sensor.
- sensors NumericSensor_4_0_0 [powerFactor](#)
Power factor sensor.
- sensors NumericSensor_4_0_0 [activeEnergy](#)
Active energy sensor.
- sensors NumericSensor_4_0_0 [apparentEnergy](#)
Apparent energy sensor.

9.152.1 Detailed Description

An inlet or outlet pole.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Pole.idl

9.153 auth::Policy Struct Reference

Authentication policy.

```
import "AuthManager.idl";
```

Public Attributes

- auth [Type](#) type
Authentication type.
- boolean [useLocalIfRemoteFailed](#)
Fall back to local authentication if remote authentication fails.

9.153.1 Detailed Description

Authentication policy.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AuthManager.idl

9.154 portsmodel::Port_2_0_1 Interface Reference

Port interface.

```
import "Port.idl";
```

Classes

- struct [DetectionMode](#)
Port detection mode.
- struct [Properties](#)
Port properties.

Public Types

- enum [DetectionType](#) { [AUTO](#), [PINNED](#), [DISABLED](#) }
Port detection type.

Public Member Functions

- [Properties](#) [getProperties](#) ()
Get the current properties of the port.
- void [setName](#) (in string name)
NOT USED RIGHT NOW!
- int [setDetectionMode](#) (in [DetectionMode](#) mode)
Set the detection mode for the port.
- vector< string > [getDetectableDevices](#) ()
Get all detectable devices of this port.
- Object [getDevice](#) ()
Get the connected device of the port.

- Object [getDeviceConfig](#) (in string deviceType)
Get device type specific configuration interface.

Public Attributes

- constant int [NO_ERROR](#) = 0
Error codes.
- constant int [ERR_INVALID_PARAM](#) = 1
invalid parameter for an operation
- constant int [ERR_DEVICE_BUSY](#) = 2
operation fails because connected device is busy
- valueobject [PropertiesChangedEvent](#): [idl.Event](#) { [Properties](#) oldProperties
Event: The port properties have changed.
- [Properties](#) newProperties
Properties after change.
- valueobject [DeviceChangedEvent](#): [idl.Event](#) { Object oldDevice
Event: The device connected to the port has changed.
- Object [newDevice](#)
Connected device after change.

9.154.1 Detailed Description

Port interface.

9.154.2 Member Enumeration Documentation

9.154.2.1 enum portsmode::Port_2_0_1::DetectionType

Port detection type.

Enumerator

AUTO auto detection of connected devices
PINNED port is pinned to a specific device type
DISABLED port is disabled and will not detect any device connected

9.154.3 Member Function Documentation

9.154.3.1 vector<string> portsmode::Port_2_0_1::getDetectableDevices ()

Get all detectable devices of this port.

Returns

List of all registered detectable Devices

9.154.3.2 Object portsmode::Port_2_0_1::getDevice ()

Get the connected device of the port.

Returns

Device connected to Port

9.154.3.3 Object `portsmodel::Port_2_0_1::getDeviceConfig ( in string deviceType )`

Get device type specific configuration interface.

Parameters

<i>deviceType</i>	Device type to get configuration interface for
-------------------	--

Returns

Device configuration interface

9.154.3.4 Properties `portsmodel::Port_2_0_1::getProperties ( )`

Get the current properties of the port.

Returns

[Properties](#) of the Port

9.154.3.5 `int portsmodel::Port_2_0_1::setDetectionMode ( in DetectionMode mode )`

Set the detection mode for the port.

Parameters

<i>mode</i>	new detection mode
-------------	--------------------

Returns

NO_ERROR on success
ERR_INVALID_PARAM invalid parameter
ERR_DEVICE_BUSY device busy (e.g. Asset Strip Firmware Update)

9.154.3.6 `void portsmodel::Port_2_0_1::setName ( in string name )`

NOT USED RIGHT NOW!

Set the port name

Parameters

<i>name</i>	new port name
-------------	---------------

9.154.4 Member Data Documentation

9.154.4.1 `valueobject portsmodel::Port_2_0_1::DeviceChangedEvent`

Event: The device connected to the port has changed.

Connected device before change

9.154.4.2 `constant int portsmode::Port_2_0_1::NO_ERROR = 0`

Error codes.

operation successful, no error

9.154.4.3 `valueobject portsmode::Port_2_0_1::PropertiesChangedEvent`

Event: The port properties have changed.

[Properties](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Port.idl

9.155 serial::PortDispatcher_1_1_1 Interface Reference

Top-level interface of the serial port manager.

```
import "PortDispatcher.idl";
```

Public Member Functions

- `map< string, SerialPort\_2\_0\_0 > getPorts ()`
Get serial ports.

9.155.1 Detailed Description

Top-level interface of the serial port manager.

9.155.2 Member Function Documentation

9.155.2.1 `map<string, SerialPort\_2\_0\_0> serial::PortDispatcher_1_1_1::getPorts ( )`

Get serial ports.

Returns a map of serial ports that are both

- available on the device
- monitored by the system

The mapping is 'symbolic port name' -> 'port instance'.

Returns

- A map with available port instances

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/PortDispatcher.idl

9.156 peripheral::PosElement Struct Reference

peripheral device position element, list forms position

```
import "PeripheralDeviceSlot.idl";
```

Public Attributes

- [PortType portType](#)
type of the element
- string [port](#)
value of the element, a label

9.156.1 Detailed Description

peripheral device position element, list forms position

The documentation for this struct was generated from the following file:

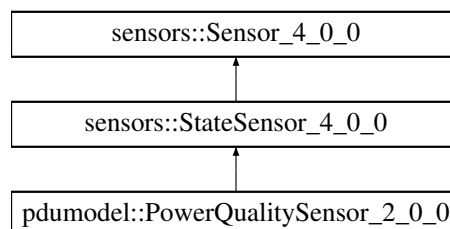
- pdu-json-rpc-api/idl/PeripheralDeviceSlot.idl

9.157 pdumodel::PowerQualitySensor_2_0_0 Interface Reference

Power quality sensor interface.

```
import "PowerQualitySensor.idl";
```

Inheritance diagram for pdumodel::PowerQualitySensor_2_0_0:



Public Attributes

- constant int [STATE_NORMAL](#) = 0
Power quality is normal.
- constant int [STATE_WARNING](#) = 1
Power quality is warning.
- constant int [STATE_CRITICAL](#) = 2
Power quality is critical.

Additional Inherited Members

9.157.1 Detailed Description

Power quality sensor interface.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/PowerQualitySensor.idl

9.158 usermgmt::Preferences Struct Reference

User preferences

```
import "User.idl";
```

Public Attributes

- [TemperatureEnum temperatureUnit](#)
Display unit for temperature sensors.
- [LengthEnum lengthUnit](#)
Display unit for length measurements.
- [PressureEnum pressureUnit](#)
Display unit for pressure sensors.

9.158.1 Detailed Description

User preferences

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/User.idl

9.159 usermgmt::Role::Privilege Struct Reference

A granted privilege.

```
import "Role.idl";
```

Public Attributes

- string [name](#)
Privilege name.
- vector< string > [args](#)
Privilege arguments.

9.159.1 Detailed Description

A granted privilege.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Role.idl

9.160 usermgmt::RoleManager::PrivilegeDesc Struct Reference

Privilege Description.

```
import "RoleManager.idl";
```

Public Attributes

- string [name](#)
Privilege name.
- string [desc](#)
Privilege description.
- vector< [ArgumentDesc](#) > [args](#)
List of supported arguments.

9.160.1 Detailed Description

Privilege Description.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/RoleManager.idl

9.161 production::Production Interface Reference

Methods used during production.

```
import "Production.idl";
```

Public Member Functions

- int [enterFactoryConfigMode](#) (in string password)
Enter factory configuration mode.
- void [leaveFactoryConfigMode](#) ()
Leave factory configuration mode.
- boolean [isFactoryConfigModeEnabled](#) ()
Check whether device is in factory configuration mode.

9.161.1 Detailed Description

Methods used during production.

9.161.2 Member Function Documentation

9.161.2.1 int production::Production::enterFactoryConfigMode (in string *password*)

Enter factory configuration mode.

This mode unlocks several features which should only be available during production, e.g. the RS485 address assignment or sensor calibration. Factory configuration mode will be enabled until the next reboot or until [leaveFactoryConfigMode](#) is called.

Parameters

<i>password</i>	Password required to enter factory config mdoe.
-----------------	---

Returns

- 0 if OK
- 1 if the password is wrong

9.161.2.2 boolean production::Production::isFactoryConfigModeEnabled ()

Check whether device is in factory configuration mode.

Returns

- `true` if factory configuration mode is enabled

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Production.idl

9.162 portsmode::Port_2_0_1::Properties Struct Reference

Port properties.

```
import "Port.idl";
```

Public Attributes

- string [name](#)
user defineable name - NOT USED RIGHT NOW!
- string [label](#)
label on device
- [DetectionMode](#) [mode](#)
detection mode
- string [detectedDeviceType](#)
detected device type or empty if nothing connected
- string [detectedDeviceName](#)
detected device name or empty if nothing connected

9.162.1 Detailed Description

Port properties.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Port.idl

9.163 cascading::Cascading_1_0_1::ProtocolMapping Struct Reference

Mapping from appl protocol id to name and transport protocol.

```
import "Cascading.idl";
```

Public Attributes

- int **appProtold**
- string **appProtoName**
- string **transportProtoName**

9.163.1 Detailed Description

Mapping from appl protocol id to name and transport protocol.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Cascading.idl

9.164 assetmgrmodel::AssetStrip_2_0_1::RackUnitInfo Struct Reference

Infos for a single rack unit.

```
import "AssetStrip.idl";
```

Public Attributes

- int [rackUnitNumber](#)
rack unit for the settings, range 0..rackUnitCount-1
- int [rackUnitPosition](#)
resulting rack unit position (display number)
- [TagType](#) type
type of the asset tag (single, extension, none or unknown)
- int [size](#)
blade extension size (4,8,16), 1 for single tags or 0 if nothing connected
- [AssetStripConfig_1_0_1](#)
RackUnitSettings [settings](#)
settings for a single rack unit

9.164.1 Detailed Description

Infos for a single rack unit.

In case no asset strip is connected, type defaults to single and size defaults to 1

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStrip.idl

9.165 assetmgrmodel::AssetStripConfig_1_0_1::RackUnitSettings Struct Reference

Settings for a single rack unit (LED state)

```
import "AssetStripConfig.idl";
```

Public Attributes

- [LEDOperationMode](#) `opmode`
Operation mode for this rack unit.
- [LEDMode](#) `mode`
LED mode (on,off,blinking)
- [LEDColor](#) `color`
Color of the LED at this rack unit.
- string `name`
user defined name (up to 64 alphanumeric characters)

9.165.1 Detailed Description

Settings for a single rack unit (LED state)

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStripConfig.idl

9.166 auth::RadiusManager Interface Reference

RADIUS server configuration interface.

```
import "RadiusManager.idl";
```

Public Member Functions

- vector< [radius.ServerSettings](#) > [getRadiusServers](#) ()
Get a list of RADIUS server settings.
- int [setRadiusServers](#) (in vector< [radius.ServerSettings](#) > serverList)
Sets a list of RADIUS servers.
- int [testRadiusServer](#) (in string username, in string password, in [radius.ServerSettings](#) settings)
Tests an RADIUS server configuration.

Public Attributes

- constant int [ERR_INVALID_CFG](#) = 1
The server configuration is invalid.
- constant int [ERR_SERVER_UNSPECIFIED](#) = 2
Unspecified error.
- constant int [ERR_INVALID_SHARED_SECRET](#) = 3
The shared secret is invalid.
- constant int [ERR_SERVER_UNREACHABLE](#) = 4
RADIUS server could not be contacted.
- constant int [ERR_AUTHENTICATION_FAILED](#) = 5
User could not be authenticated.
- constant int [ERR_NO_ROLES](#) = 6
No roles are defined for the user.
- constant int [ERR_NO_KNOWN_ROLES](#) = 7
No known rules are defined for the user.

9.166.1 Detailed Description

RADIUS server configuration interface.

9.166.2 Member Function Documentation

9.166.2.1 `vector<radius.ServerSettings> auth::RadiusManager::getRadiusServers ( )`

Get a list of RADIUS server settings.

Returns

list of [radius.ServerSettings](#)

9.166.2.2 `int auth::RadiusManager::setRadiusServers ( in vector< radius.ServerSettings > serverList )`

Sets a list of RADIUS servers.

Any existing RADIUS Server configuration will be cleared / overwritten.

Returns

0 on success
ERR_INVALID_CFG in case of invalid configuration

9.166.2.3 `int auth::RadiusManager::testRadiusServer ( in string username, in string password, in radius.ServerSettings settings )`

Tests an RADIUS server configuration.

Returns

0 on success
ERR_SERVER_UNSPECIFIED an unspecified error occurred
ERR_INVALID_CFG RADIUS server configuration is invalid (reused from setRadiusServers)
ERR_INVALID_SHARED_SECRET the shared secret is invalid
ERR_SERVER_UNREACHABLE RADIUS server could not be contacted
ERR_AUTHENTICATION_FAILED user could not be authenticated
ERR_NO_ROLES no roles are defined for the user
ERR_NO_KNOWN_ROLES no known roles are defined for the user

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/RadiusManager.idl

9.167 event::TimerEventManager_2_0_0::Range Struct Reference

[Range](#) structure.

```
import "TimerEventManager.idl";
```


Public Attributes

- int [start](#)
Start time.
- int [end](#)
End time.
- int [step](#)
Step.

9.167.1 Detailed Description

[Range](#) structure.

A range defines a time between start and end point in time. If start and end point are the same it is a specific point in time and no range. Step defines the repeating interval in time. Step of 1 is default and matches all values in range.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TimerEventManager.idl

9.168 sensors::NumericSensor_4_0_0::Range Struct Reference

[Range](#) of possible sensor readings.

```
import "NumericSensor.idl";
```

Public Attributes

- double [lower](#)
Minimum reading.
- double [upper](#)
Maximum reading.

9.168.1 Detailed Description

[Range](#) of possible sensor readings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/NumericSensor.idl

9.169 pdumodel::Rating Struct Reference

Numerical usage ratings.

```
import "Nameplate.idl";
```

Public Attributes

- int [current](#)
Maximum current in Amperes.

- int [minVoltage](#)
Minimum voltage in Volts.
- int [maxVoltage](#)
Maximum voltage in Volts.

9.169.1 Detailed Description

Numerical usage ratings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Nameplate.idl

9.170 pdumodel::Nameplate::Rating Struct Reference

Component ratings.

```
import "Nameplate.idl";
```

Public Attributes

- string [voltage](#)
Voltage rating.
- string [current](#)
Current rating.
- string [frequency](#)
Frequency rating.
- string [power](#)
Power rating.

9.170.1 Detailed Description

Component ratings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Nameplate.idl

9.171 sensors::NumericSensor_4_0_0::Reading Struct Reference

Numeric sensor reading.

```
import "NumericSensor.idl";
```

Classes

- struct [Status](#)
Numeric sensor status.

Public Attributes

- time [timestamp](#)
Timestamp of last sample.
- boolean [available](#)
true if the sensor is available
- [Status](#) [status](#)
Numeric sensor status.
- boolean [valid](#)
true if the sensor reading is valid
- double [value](#)
Numeric sensor reading.

9.171.1 Detailed Description

Numeric sensor reading.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/NumericSensor.idl

9.172 Ihxmodel::Sensor_4_0_0::Reading Struct Reference

Sensor reading.

```
import "LhxSensor.idl";
```

Public Attributes

- time [timestamp](#)
Time of sample.
- int [state](#)
discrete reading or state
- double [value](#)
numeric reading value
- boolean [isValid](#)
numeric value is valid or NAN

9.172.1 Detailed Description

Sensor reading.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LhxSensor.idl

9.173 sensors::Logger_2_1_2::Record Struct Reference

Sensor log record.

```
import "SensorLogger.idl";
```

Public Attributes

- boolean [available](#)
Sensor was available for at least one sample.
- int [takenValidSamples](#)
Number of samples with a valid reading/state.
- int [state](#)
Sensor state.
- double [minValue](#)
Minimum sensor reading.
- double [avgValue](#)
Average sensor reading.
- double [maxValue](#)
Maximum sensor reading.

9.173.1 Detailed Description

Sensor log record.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SensorLogger.idl

9.174 assetmgrmodel::AssetStripLogger_1_0_2::Record Struct Reference

Log record structure.

```
import "AssetStripLogger.idl";
```

Public Attributes

- time [timestamp](#)
Time of log entry creation.
- [RecordType](#) type
Entry type.
- int [assetStripNumber](#)
Asset strip number (0-based), -1 if unknown.
- int [rackUnitNumber](#)
Rack unit number (0-based), -1 if unknown.
- int [rackUnitPosition](#)
Rack unit position, -1 if unknown.
- int [slotNumber](#)
Blade extension slot number, -1 if unknown, 0 is main strip, >0 is blade.
- string [tagId](#)
The ID of the asset management tag, empty if unknown.
- string [parentBladeId](#)
ID of the parent blade extension tag in case slotNumber>0, empty otherwise.
- [AssetStrip_2_0_1](#) State [state](#)
Asset strip state.

9.174.1 Detailed Description

Log record structure.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStripLogger.idl

9.175 cert::ServerSSLCert::ReqInfo Struct Reference

Certificate signing request information.

```
import "ServerSSLCert.idl";
```

Public Attributes

- [CommonAttributes](#) `subject`
Certificate subject attributes.
- `int` [keyLength](#)
Key length in bits.

9.175.1 Detailed Description

Certificate signing request information.

The documentation for this struct was generated from the following file:

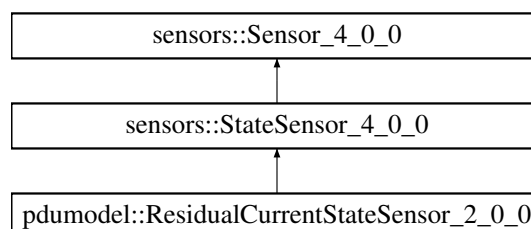
- pdu-json-rpc-api/idl/ServerSSLCert.idl

9.176 pdumodel::ResidualCurrentStateSensor_2_0_0 Interface Reference

Residual current state sensor interface.

```
import "ResidualCurrentStateSensor.idl";
```

Inheritance diagram for pdumodel::ResidualCurrentStateSensor_2_0_0:



Public Member Functions

- `int` [startSelfTest](#) ()
Start a self test of the residual current monitor.

Public Attributes

- constant int [STATE_NORMAL](#) = 0
Possible sensor state values.
- constant int [STATE_WARNING](#) = 1
Residual current sensor detected overcurrent.
- constant int [STATE_CRITICAL](#) = 2
Residual current sensor detected overcurrent.
- constant int [STATE_SELFTEST](#) = 3
Residual current sensor currently conducts a self test.
- constant int [STATE_FAILURE](#) = 4
Residual current sensor is unavailable or self test failed.

Additional Inherited Members

9.176.1 Detailed Description

Residual current state sensor interface.

9.176.2 Member Function Documentation

9.176.2.1 int pdumodel::ResidualCurrentStateSensor_2_0_0::startSelfTest ()

Start a self test of the residual current monitor.

Returns

- 0 if OK
- 1 if no residual current monitor is present
- 2 if a self test is already running

9.176.3 Member Data Documentation

9.176.3.1 constant int pdumodel::ResidualCurrentStateSensor_2_0_0::STATE_NORMAL = 0

Possible sensor state values.

Residual current sensor is operating normally

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/ResidualCurrentStateSensor.idl

9.177 res_mon::ResMon Interface Reference

[ResMon](#) interface.

```
import "ResMon.idl";
```

Public Member Functions

- void [getDataEntries](#) (out vector< [Entry](#) > entries)
Retrieve [ResMon](#) data [Entry](#) array.

9.177.1 Detailed Description

[ResMon](#) interface.

9.177.2 Member Function Documentation

9.177.2.1 void res_mon::ResMon::getDataEntries (out vector< [Entry](#) > *entries*)

Retrieve [ResMon](#) data [Entry](#) array.

Parameters

<i>entries</i>	Result: List of data events
----------------	-----------------------------

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/ResMon.idl

9.178 security::RestrictedServiceAgreement Struct Reference

Restricted Service Agreement settings.

```
import "Security.idl";
```

Public Attributes

- boolean [enabled](#)
Enforce Restricted Service Agreement.
- string [banner](#)
Restricted Service Agreement Banner.

9.178.1 Detailed Description

Restricted Service Agreement settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.179 test::Result Struct Reference

Convenience structure to return test or operation results.

```
import "testrpc.idl";
```

Public Attributes

- int [code](#)
Result code.
- string [errtext](#)
Descriptive error string.

9.179.1 Detailed Description

Convenience structure to return test or operation results.

- There is a result code Typically 0 means OK 1 means Test Error 2 means Invalid Parameter However individual test may redefine or extend this definition
- There is a descriptive error string, which is empty in case test went OK

This structure is not necessarily used for all operations but only for those where it comes in handy

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/testrpc.idl

9.180 usermgmt::Role Interface Reference

Role management interface

```
import "Role.idl";
```

Classes

- struct [Info](#)
Role information
- struct [Privilege](#)
A granted privilege.

Public Member Functions

- [Info](#) [getInfo](#) ()
Retrieve role information.
- int [updateFull](#) (in [Info](#) info)
Change role settings.

Public Attributes

- constant int [ERR_INVALID_VALUE](#) = 1
Invalid arguments.

9.180.1 Detailed Description

Role management interface

9.180.2 Member Function Documentation

9.180.2.1 Info usermgmt::Role::getInfo ()

Retrieve role information.

Returns

role info

9.180.2.2 int usermgmt::Role::updateFull (in Info *info*)

Change role settings.

Parameters

<i>info</i>	New role information
-------------	----------------------

Returns

- 0 if OK
- 1 if the role information is invalid

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Role.idl

9.181 security::RoleAccessControl Struct Reference

Role-based access control settings.

```
import "Security.idl";
```

Public Attributes

- boolean [enabled](#)
true to enable role-based access control
- [RoleAccessPolicy](#) [defaultPolicy](#)
Default policy.
- vector< [RoleAccessRule](#) > [rules](#)
List of access rules.

9.181.1 Detailed Description

Role-based access control settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.182 security::RoleAccessRule Struct Reference

Role-based access rule.

```
import "Security.idl";
```

Public Attributes

- string [startIp](#)
Start of IP range.
- string [endIp](#)
End of IP range.

- int [roleid](#)
Role id.
- [RoleAccessPolicy](#) policy
Access policy.

9.182.1 Detailed Description

Role-based access rule.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.183 usermgmt::RoleManager::RoleAccount Struct Reference

[Role](#) information.

```
import "RoleManager.idl";
```

Public Attributes

- int [id](#)
Unique role id.
- string [name](#)
[Role](#) name.
- [Role Info](#) info
[Role](#) information.

9.183.1 Detailed Description

[Role](#) information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/RoleManager.idl

9.184 usermgmt::RoleManager Interface Reference

[Role](#) manager interface.

```
import "RoleManager.idl";
```

Classes

- struct [ArgumentDesc](#)
Privilege Argument Description.
- struct [Info](#)
Full role manager information.
- struct [PrivilegeDesc](#)
Privilege Description.
- struct [RoleAccount](#)
[Role](#) information.

Public Member Functions

- int [createRoleFull](#) (in string name, in [Role.Info](#) info)
Create new role with full information.
- int [deleteRole](#) (in string name)
Delete a role.
- vector< string > [getAllRoleNames](#) ()
Retrieve a list of role names.
- vector< [RoleAccount](#) > [getAllRoles](#) ()
Retrieve a list of active roles.
- vector< [PrivilegeDesc](#) > [getAllPrivileges](#) ()
Retrieve a list of supported privileges.
- [Info](#) [getInfo](#) ()
Retrieve full role manager information.

Public Attributes

- constant int [ERR_ROLE_ALREADY_EXISTS](#) = 1
A role with that name already exists.
- constant int [ERR_MAX_ROLES_REACHED](#) = 2
Maximum number of roles reached.
- constant int [ERR_INVALID_VALUE](#) = 3
Invalid arguments.
- constant int [ERR_ROLE_DOESNT_EXIST](#) = 1
The role does not exist.
- constant int [ERR_ROLE_NOT_DELETABLE](#) = 2
The role cannot be deleted.

9.184.1 Detailed Description

[Role](#) manager interface.

9.184.2 Member Function Documentation

9.184.2.1 int usermgmt::RoleManager::createRoleFull (in string name, in [Role.Info](#) info)

Create new role with full information.

Parameters

<i>name</i>	New role name
<i>info</i>	New role information

Returns

- 0 if OK
- 1 if a role with that name already exists
- 2 if the maximum number of roles is reached
- 3 if the role information is invalid

9.184.2.2 `int usermgmt::RoleManager::deleteRole ( in string name )`

Delete a role.

Parameters

<i>name</i>	Name of the role to delete
-------------	----------------------------

Returns

- 0 if OK
- 1 if a role with the given name does not exist
- 2 if the role cannot be deleted

9.184.2.3 `vector<PrivilegeDesc> usermgmt::RoleManager::getAllPrivileges ( )`

Retrieve a list of supported privileges.

Returns

List of privilege names

9.184.2.4 `vector<string> usermgmt::RoleManager::getAllRoleNames ( )`

Retrieve a list of role names.

Returns

List of role names

9.184.2.5 `vector<RoleAccount> usermgmt::RoleManager::getAllRoles ( )`

Retrieve a list of active roles.

Returns

List of active roles

9.184.2.6 `Info usermgmt::RoleManager::getInfo ( )`

Retrieve full role manager information.

Returns

[Role](#) manager information

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/RoleManager.idl

9.185 `test::RS232Serial` Interface Reference

Test routines for full RS232 Serial Interface.

```
import "testrpc.idl";
```

Public Member Functions

- vector< string > [getDeviceFiles](#) ()
returns list of valid device files for the test
- [Result testLoop1](#) (in string devfile)
loop test with adapter 1 DTR->DSR, RTS->CTS
- [Result testLoop2](#) (in string devfile)
loop test with adapter 2 DTR->DCD, RTS->RI

9.185.1 Detailed Description

Test routines for full RS232 Serial Interface.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/testrpc.idl

9.186 event::Engine::Rule Struct Reference

A [Rule](#) binds an action to a condition.

```
import "EventEngine.idl";
```

Public Attributes

- string [id](#)
Rule ID.
- string [name](#)
User-defined name.
- boolean [isSystem](#)
true for system-defined rules
- boolean [isEnabled](#)
true if the rule is enabled
- boolean [isAutoRearm](#)
true for auto-rearming rules
- boolean [hasMatched](#)
true if the rule has matched since being armed
- [Condition](#) [condition](#)
Trigger condition.
- vector< string > [actionIds](#)
List of action IDs.
- vector< [KeyValue](#) > [arguments](#)
Argument map.

9.186.1 Detailed Description

A [Rule](#) binds an action to a condition.

The 'id' is an unique identification within the scope of this event engine. The 'name' is unique as well and used by the GUI as user readable identifier. The flag 'isAutoRearm' determines whether actions are triggered again after condition reoccures. The 'hasMatched' flag is true if the condition was true once and the rule is not rearmed

yet. The 'isSystem' flag is readonly and if set marks the action as not deletable. The 'actionIds' are the actions to perform if the rule evaluates to true. The arguments are passed to the actions.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/EventEngine.idl

9.187 pdumodel::Ade::Sample Struct Reference

Raw sample data for a single channel.

```
import "Ade.idl";
```

Public Attributes

- long [vrms](#)
RMS voltage.
- long [irms](#)
RMS current.
- long [watt](#)
Active power.
- long [va](#)
Apparent power.
- long [wh](#)
Active energy (for this sample)
- long [vah](#)
Apparent energy (for this sample)

9.187.1 Detailed Description

Raw sample data for a single channel.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Ade.idl

9.188 event::TimerEventManager_2_0_0::Schedule Struct Reference

[Schedule](#) structure.

```
import "TimerEventManager.idl";
```

Public Attributes

- boolean [enabled](#)
Whether the timer event is enabled.
- vector< [Range](#) > [minute](#)
Ranges for minute.
- vector< [Range](#) > [hour](#)
Ranges for hour.
- vector< [Range](#) > [dayOfMonth](#)

Ranges for day of month.

- vector< [Range](#) > month

Ranges for month.

- vector< [Range](#) > dayOfWeek

Ranges for day of week.

9.188.1 Detailed Description

[Schedule](#) structure.

An empty vector means a match for all values. The values range of every field depends on the meaning of the field. For minutes 0-59 is possible, for hour 0-24 and for day of month 0-31. For month and day of weeks the definitions above should be used.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TimerEventManager.idl

9.189 security::Security_3_0_0 Interface Reference

Security configuration interface

```
import "Security.idl";
```

Classes

- struct [Settings](#)

Security configuration This structure is deprecated and will be removed in V3.0, use concrete getters and setters instead!

Public Member Functions

- [Settings](#) [getSettings](#) ()
Retrieve the security configuration.
- int [setSettings](#) (in [Settings](#) settings)
Set the security configuration.
- void [setHttpRedirSettings](#) (in boolean http2httpsRedir)
Enable or disable HTTP-to-HTTPS redirection.
- int [setIpFwSettings](#) (in [IpFw_2_0_0](#) ipFw)
Set the IP packet filter configuration.
- int [setIpV6FwSettings](#) (in [IpFw_2_0_0](#) ipV6Fw)
Set the IPv6 packet filter configuration.
- int [setRoleAccessControlSettings](#) (in [RoleAccessControl](#) settings)
Change the role-based access control settings.
- int [setRoleAccessControlSettingsV6](#) (in [RoleAccessControl](#) settings)
Change the role-based access control settings for IPv6.
- int [setBlockSettings](#) (in int blockTimeout, in int maxFailedLogins)
Change the user blocking settings.
- int [setPwSettings](#) (in [PasswordSettings](#) pwSettings)
Change the password settings.
- int [setIdleTimeoutSettings](#) (in int idleTimeout)
Change the session idle timeout.

- void [setSingleLoginLimitation](#) (in boolean singleLogin)
Enable or disable single login limitation.
- int [getIdleTimeoutSettings](#) ()
Retrieve the current idle timeout.
- boolean [getHttpRedirSettings](#) ()
Retrieve the current state of the HTTP-to-HTTPS redirection.
- void [getBlockSettings](#) (out int blockTimeout, out int maxFailedLogins)
Retrieve the current user blocking settings.
- [SSHSettings](#) [getSSHSettings](#) ()
Retrieve the current SSH settings.
- void [setSSHSettings](#) (in [SSHSettings](#) settings)
Change the SSH settings.
- [RestrictedServiceAgreement](#) [getRestrictedServiceAgreement](#) ()
Retrieve the current Restricted Service Agreement settings.
- int [setRestrictedServiceAgreement](#) (in [RestrictedServiceAgreement](#) settings)
Change the Restricted Service Agreement settings.
- vector< string > [getSupportedFrontPanelPrivileges](#) ()
Retrieve a list of supported privileges for the front panel.
- int [setFrontPanelPrivileges](#) (in vector< string > privileges)
Set the privileges for the front panel.
- vector< string > [getFrontPanelPrivileges](#) ()
Retrieve the list of active front panel privileges.

Public Attributes

- constant int [ERR_INVALID_VALUE](#) = 1
Invalid arguments.

9.189.1 Detailed Description

Security configuration interface

9.189.2 Member Function Documentation

9.189.2.1 void security::Security_3_0_0::getBlockSettings (out int *blockTimeout*, out int *maxFailedLogins*)

Retrieve the current user blocking settings.

Returns

blockTimeout The block timeout in minutes
maxFailedLogins The maximum failure count

9.189.2.2 vector<string> security::Security_3_0_0::getFrontPanelPrivileges ()

Retrieve the list of active front panel privileges.

Returns

List of privilege names

9.189.2.3 `boolean security::Security_3_0_0::getHttpRedirSettings ( )`

Retrieve the current state of the HTTP-to-HTTPS redirection.

Returns

`true` if the HTTP-to-HTTPS redirection is enabled

9.189.2.4 `int security::Security_3_0_0::getIdleTimeoutSettings ( )`

Retrieve the current idle timeout.

Returns

Idle timeout in minutes

9.189.2.5 `RestrictedServiceAgreement security::Security_3_0_0::getRestrictedServiceAgreement ( )`

Retrieve the current Restricted Service Agreement settings.

Returns

Restricted Service Agreement settings

9.189.2.6 `Settings security::Security_3_0_0::getSettings ( )`

Retrieve the security configuration.

This method is deprecated and will be removed in V3.0, use concrete getter instead!

Returns

Security configuration

9.189.2.7 `SSHSettings security::Security_3_0_0::getSSHSettings ( )`

Retrieve the current SSH settings.

Returns

SSH settings

9.189.2.8 `vector<string> security::Security_3_0_0::getSupportedFrontPanelPrivileges ( )`

Retrieve a list of supported privileges for the front panel.

Returns

List of privilege names

9.189.2.9 `int security::Security_3_0_0::setBlockSettings ( in int blockTimeout, in int maxFailedLogins )`

Change the user blocking settings.

Parameters

<i>blockTimeout</i>	User blocking timeout in minutes
<i>maxFailedLogins</i>	Maximum number of failed logins

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.10 `int security::Security_3.0.0::setFrontPanelPrivileges ( in vector< string > privileges )`

Set the privileges for the front panel.

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.11 `void security::Security_3.0.0::setHttpRedirSettings ( in boolean http2httpsRedir )`

Enable or disable HTTP-to-HTTPS redirection.

Parameters

<i>http2httpsRedir</i>	true to enable the redirection
------------------------	--------------------------------

9.189.2.12 `int security::Security_3.0.0::setIdleTimeoutSettings ( in int idleTimeout )`

Change the session idle timeout.

Parameters

<i>idleTimeout</i>	New idle timeout in minutes
--------------------	-----------------------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.13 `int security::Security_3.0.0::setIpFwSettings ( in IpFw_2.0.0 ipFw )`

Set the IP packet filter configuration.

Parameters

<i>ipFw</i>	New packet filter settings
-------------	----------------------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.14 int security::Security_3_0_0::setIPv6FwSettings (in IpFw_2_0_0 *ipV6Fw*)

Set the IPv6 packet filter configuration.

Parameters

<i>ipV6Fw</i>	New packet filter settings
---------------	----------------------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.15 int security::Security_3_0_0::setPwSettings (in PasswordSettings *pwSettings*)

Change the password settings.

Parameters

<i>pwSettings</i>	New settings
-------------------	--------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.16 int security::Security_3_0_0::setRestrictedServiceAgreement (in RestrictedServiceAgreement *settings*)

Change the Restricted Service Agreement settings.

Parameters

<i>settings</i>	New settings
-----------------	--------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.17 int security::Security_3_0_0::setRoleAccessControlSettings (in RoleAccessControl *settings*)

Change the role-based access control settings.

Parameters

<i>settings</i>	New settings
-----------------	--------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.18 `int security::Security_3_0_0::setRoleAccessControlSettingsV6 ( in RoleAccessControl settings )`

Change the role-based access control settings for IPv6.

Parameters

<i>settings</i>	New settings
-----------------	--------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.19 `int security::Security_3_0_0::setSettings ( in Settings settings )`

Set the security configuration.

This method is deprecated and will be removed in V3.0, use concrete setter instead!

Parameters

<i>settings</i>	New security settings
-----------------	-----------------------

Returns

0 on success
ERR_INVALID_VALUE if any argument was invalid

9.189.2.20 `void security::Security_3_0_0::setSingleLoginLimitation ( in boolean singleLogin )`

Enable or disable single login limitation.

Parameters

<i>singleLogin</i>	true to enable single login limitation
--------------------	--

9.189.2.21 `void security::Security_3_0_0::setSSHSettings ( in SSHSettings settings )`

Change the SSH settings.

Parameters

<i>settings</i>	New settings
-----------------	--------------

The documentation for this interface was generated from the following file:

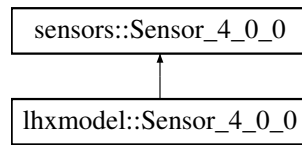
- pdu-json-rpc-api/idl/Security.idl

9.190 Ihxmodel::Sensor_4_0_0 Interface Reference

LHX Sensor Interface.

```
import "LhxSensor.idl";
```

Inheritance diagram for lhxmodel::Sensor_4_0_0:



Classes

- struct [MetaData](#)
Sensor's self describing data.
- struct [NumThresholds](#)
Numerical sensor thresholds.
- struct [Reading](#)
Sensor reading.

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the LHX metadata.
- [NumThresholds](#) [getThresholds](#) ()
Retrieve Numeric Thresholds.
- int [setThresholds](#) (in [NumThresholds](#) thresholds)
Set Numeric Thresholds.
- [Reading](#) [getReading](#) ()
Retrieve sensors reading.

Public Attributes

- constant int [STATE_NOT_AVAILABLE](#) = -1
Communication to sensor lost.
- constant int [STATE_CLOSED](#) = 0
Closed.
- constant int [STATE_OPEN](#) = 1
Open.
- constant int [STATE_NUM_NORMAL](#) = 0
Numerical sensor in normal range.
- constant int [STATE_NUM_ABOVE_UPPER_CRITICAL](#) = 1
Above upper critical threshold.
- constant int [STATE_NUM_ABOVE_UPPER_WARNING](#) = 2
Above upper warning threshold.
- constant int [STATE_NUM_BELOW_LOWER_WARNING](#) = 3
Below lower warning threshold.
- constant int [STATE_NUM_BELOW_LOWER_CRITICAL](#) = 4
Below lower critical threshold.
- valueobject [ThresholdsChangedEvent](#): [event.UserEvent](#) { [NumThresholds](#) oldThresholds
Event: Sensor thresholds have been changed.
- [NumThresholds](#) [newThresholds](#)
Thresholds after change.

- valueobject [StateChangedEvent](#): [idl.Event](#) { [Reading](#) oldReading
Event: Sensor state has changed.
- [Reading](#) newReading
Reading after change.
- valueobject [ReadingChangedEvent](#): [idl.Event](#) { [Reading](#) newReading
Event: Sensor numeric reading has changed.
- constant int [ERR_INVALID_PARAM](#) = 1
Invalid parameters.
- constant int [ERR_NOT_SUPPORTED](#) = 2
Not supported.

Additional Inherited Members

9.190.1 Detailed Description

LHX Sensor Interface.

9.190.2 Member Function Documentation

9.190.2.1 [MetaData](#) [Ihxmodel::Sensor_4_0_0::getMetaData](#) ()

Retrieve the LHX metadata.

Returns

metadata

9.190.2.2 [Reading](#) [Ihxmodel::Sensor_4_0_0::getReading](#) ()

Retrieve sensors reading.

Returns

reading

9.190.2.3 [NumThresholds](#) [Ihxmodel::Sensor_4_0_0::getThresholds](#) ()

Retrieve Numeric Thresholds.

Returns

sensor thresholds

9.190.2.4 [int](#) [Ihxmodel::Sensor_4_0_0::setThresholds](#) (in [NumThresholds](#) *thresholds*)

Set Numeric Thresholds.

Returns

0 if OK
ERR_INVALID_PARAM if any parameters are invalid

9.190.3 Member Data Documentation

9.190.3.1 valueobject Ihxmodel::Sensor_4_0_0::ReadingChangedEvent

Event: Sensor numeric reading has changed.

New reading

9.190.3.2 valueobject Ihxmodel::Sensor_4_0_0::StateChangedEvent

Event: Sensor state has changed.

[Reading](#) before change

9.190.3.3 valueobject Ihxmodel::Sensor_4_0_0::ThresholdsChangedEvent

Event: Sensor thresholds have been changed.

Thresholds before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/LhxSensor.idl

9.191 sensors::Sensor_4_0_0 Interface Reference

Sensor interface

```
import "Sensor.idl";
```

Inheritance diagram for sensors::Sensor_4_0_0:



Classes

- struct [TypeSpec](#)
Complete sensor type specification.

Public Types

- enum [OnOffState](#) { [OFF](#), [ON](#) }
DISCRETE_ON_OFF Sensor State.

Public Member Functions

- [TypeSpec](#) [getTypeSpec](#) ()
Retrieve the sensor type specification.
- int [setType](#) (in int type, in int unit)
Set sensor type and unit.

Public Attributes

- constant int **ERR_NOT_SUPPORTED** = 1
The operation is not supported.
- constant int **NUMERIC** = 0
Sensor reading type
- constant int **DISCRETE_ON_OFF** = 1
- constant int **DISCRETE_MULTI** = 2
- constant int **UNSPECIFIED** = 0
Sensor type
- constant int **VOLTAGE** = 1
- constant int **CURRENT** = 2
- constant int **UNBALANCE_CURRENT** = 3
- constant int **POWER** = 4
- constant int **POWER_FACTOR** = 5
- constant int **ENERGY** = 6
- constant int **FREQUENCY** = 7
- constant int **TEMPERATURE** = 8
- constant int **HUMIDITY** = 9
- constant int **AIR_FLOW** = 10
- constant int **AIR_PRESSURE** = 11
- constant int **CONTACT_CLOSURE** = 12
- constant int **ON_OFF_SENSOR** = 13
- constant int **TRIP_SENSOR** = 14
- constant int **VIBRATION** = 15
- constant int **WATER_LEAK** = 16
- constant int **SMOKE_DETECTOR** = 17
- constant int **TOTAL_HARMONIC_DISTORTION** = 18
- constant int **MASS** = 19
- constant int **ELECTRICAL_RESISTANCE** = 20
- constant int **FLUX** = 21
- constant int **LUMINOUS_INTENSITY** = 22
- constant int **ACCELERATION** = 23
- constant int **MAGNETIC_FLUX_DENSITY** = 24
- constant int **ELECTRIC_FIELD_STRENGTH** = 25
- constant int **MAGNETIC_FIELD_STRENGTH** = 26
- constant int **ANGLE** = 27
- constant int **SELECTION** = 28
- constant int **FAULT_STATE** = 29
- constant int **POWER_QUALITY** = 30
- constant int **ROTATIONAL_SPEED** = 31
- constant int **LUMINOUS_ENERGY** = 32
- constant int **LUMINOUS_FLUX** = 33
- constant int **ILLUMINANCE** = 34
- constant int **LUMINOUS_EMITTANCE** = 35
- constant int **MOTION** = 36
- constant int **OCCUPANCY** = 37
- constant int **TAMPER** = 38
- constant int **DRY_CONTACT** = 39
- constant int **POWERED_DRY_CONTACT** = 40
- constant int **NONE** = 0
Sensor unit
- constant int **VOLT** = 1
- constant int **AMPERE** = 2

- constant int **WATT** = 3
- constant int **VOLT_AMP** = 4
- constant int **WATT_HOUR** = 5
- constant int **VOLT_AMP_HOUR** = 6
- constant int **DEGREE_CELSIUS** = 7
- constant int **HZ** = 8
- constant int **PERCENT** = 9
- constant int **METER_PER_SEC** = 10
- constant int **PASCAL** = 11
- constant int **G** = 12
- constant int **RPM** = 13
- constant int **METER** = 14
- constant int **HOUR** = 15
- constant int **MINUTE** = 16
- constant int **SECOND** = 17
- constant int **VOLT_AMP_REACTIVE** = 18
- constant int **VOLT_AMP_REACTIVE_HOUR** = 19
- constant int **GRAM** = 20
- constant int **OHM** = 21
- constant int **LITERS_PER_HOUR** = 22
- constant int **CANDELA** = 23
- constant int **METER_PER_SQARE_SEC** = 24
- constant int **TESLA** = 25
- constant int **VOLT_PER_METER** = 26
- constant int **VOLT_PER_AMPERE** = 27
- constant int **DEGREE** = 28
- constant int **DEGREE_FAHRENHEIT** = 29
- constant int **KELVIN** = 30
- constant int **JOULE** = 31
- constant int **COULOMB** = 32
- constant int **NIT** = 33
- constant int **LUMEN** = 34
- constant int **LUMEN_SECOND** = 35
- constant int **LUX** = 36
- constant int **PSI** = 37
- constant int **NEWTON** = 38
- constant int **FOOT** = 39
- constant int **FOOT_PER_SEC** = 40
- constant int **CUBIC_METER** = 41
- constant int **RADIANT** = 42
- constant int **STERADIANT** = 43
- constant int **HENRY** = 44
- constant int **FARAD** = 45
- constant int **MOL** = 46
- constant int **BECQUEREL** = 47
- constant int **GRAY** = 48
- constant int **SIEVERT** = 49
- constant int **G_PER_CUBIC_METER** = 50
- valueobject [TypeSpecChangedEvent](#): [idl.Event](#) { [TypeSpec](#) oldTypeSpec
Event: The type specification of the sensor changed.
- [TypeSpec](#) newTypeSpec
Type specification after change.

9.191.1 Detailed Description

Sensor interface

9.191.2 Member Enumeration Documentation

9.191.2.1 enum sensors::Sensor_4_0_0::OnOffState

DISCRETE_ON_OFF Sensor State.

Enumerator

OFF off

ON on

9.191.3 Member Function Documentation

9.191.3.1 TypeSpec sensors::Sensor_4_0_0::getTypeSpec ()

Retrieve the sensor type specification.

Returns

Type specification

9.191.3.2 int sensors::Sensor_4_0_0::setType (in int *type*, in int *unit*)

Set sensor type and unit.

Parameters

<i>type</i>	the sensor type to set
<i>unit</i>	the sensor unit to set

Returns

ERR_NOT_SUPPORTED or 0

9.191.4 Member Data Documentation

9.191.4.1 valueobject sensors::Sensor_4_0_0::TypeSpecChangedEvent

Event: The type specification of the sensor changed.

Type specification before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Sensor.idl

9.192 pdumodel::OverCurrentProtector_2_1_2::Sensors Struct Reference

Overcurrent protector sensors.

```
import "OverCurrentProtector.idl";
```

Public Attributes

- sensors StateSensor_4_0_0 [trip](#)
Trip sensor.
- sensors NumericSensor_4_0_0 [current](#)
RMS current sensor.
- sensors NumericSensor_4_0_0 [peakCurrent](#)
Peak current sensor.

9.192.1 Detailed Description

Overcurrent protector sensors.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/OverCurrentProtector.idl

9.193 pdumodel::Pdu_3_0_0::Sensors Struct Reference

PDU sensors.

```
import "Pdu.idl";
```

Public Attributes

- vector< [sensors.StateSensor_4_0_0](#) > [powerSupplyStatus](#)
Power supply fault status.
- sensors NumericSensor_4_0_0 [activePower](#)
Active power sensor.
- sensors NumericSensor_4_0_0 [activeEnergy](#)
Active energy sensor.

9.193.1 Detailed Description

PDU sensors.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Pdu.idl

9.194 pdumodel::Inlet_1_2_6::Sensors Struct Reference

Inlet sensors

```
import "Inlet.idl";
```

Public Attributes

- sensors NumericSensor_4_0_0 [voltage](#)
RMS voltage sensor.
- sensors NumericSensor_4_0_0 [current](#)

- RMS current sensor.*
- sensors NumericSensor_4_0_0 [peakCurrent](#)
- Peak current sensor.*
- sensors NumericSensor_4_0_0 [residualCurrent](#)
- Residual current sensor.*
- sensors NumericSensor_4_0_0 [activePower](#)
- Active power sensor.*
- sensors NumericSensor_4_0_0 [apparentPower](#)
- Apparent power sensor.*
- sensors NumericSensor_4_0_0 [powerFactor](#)
- Power factor sensor.*
- sensors NumericSensor_4_0_0 [activeEnergy](#)
- Active energy sensor.*
- sensors NumericSensor_4_0_0 [apparentEnergy](#)
- Apparent energy sensor.*
- sensors NumericSensor_4_0_0 [unbalancedCurrent](#)
- Current unbalance sensor.*
- sensors NumericSensor_4_0_0 [lineFrequency](#)
- Line AC frequency sensor.*
- sensors NumericSensor_4_0_0 [phaseAngle](#)
- Phase angle sensor.*
- sensors StateSensor_4_0_0 [powerQuality](#)
- Power quality sensor.*
- sensors StateSensor_4_0_0 [surgeProtectorStatus](#)
- Surge protector status sensor.*
- [ResidualCurrentStateSensor_2_0_0](#) [residualCurrentStatus](#)
- Residual current monitor state sensor.*

9.194.1 Detailed Description

Inlet sensors

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Inlet.idl

9.195 pdumodel::TransferSwitch_3_1_1::Sensors Struct Reference

Transfer switch sensors.

```
import "TransferSwitch.idl";
```

Public Attributes

- sensors StateSensor_4_0_0 [selectedSource](#)
- Selected source sensor.*
- sensors StateSensor_4_0_0 [operationalState](#)
- Operational state sensor (off, normal, standby)*
- sensors NumericSensor_4_0_0 [sourceVoltagePhaseSyncAngle](#)
- Maximum phase difference between two sources.*
- sensors StateSensor_4_0_0 [overloadAlarm](#)

Overload alarm.

- sensors StateSensor_4_0_0 [phaseSyncAlarm](#)

Source phases out of sync.

- sensors StateSensor_4_0_0 [switchFault](#)

Switch fault (ok, open, short)

9.195.1 Detailed Description

Transfer switch sensors.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TransferSwitch.idl

9.196 pdumodel::Outlet_1_5_6::Sensors Struct Reference

Outlet sensors

```
import "Outlet.idl";
```

Public Attributes

- sensors NumericSensor_4_0_0 [voltage](#)
RMS voltage sensor.
- sensors NumericSensor_4_0_0 [current](#)
RMS current sensor.
- sensors NumericSensor_4_0_0 [peakCurrent](#)
Peak current sensor.
- sensors NumericSensor_4_0_0 [maximumCurrent](#)
Maximum current sensor.
- sensors NumericSensor_4_0_0 [unbalancedCurrent](#)
Current unbalance sensor.
- sensors NumericSensor_4_0_0 [activePower](#)
Active power sensor.
- sensors NumericSensor_4_0_0 [apparentPower](#)
Apparent power sensor.
- sensors NumericSensor_4_0_0 [powerFactor](#)
Power factor sensor.
- sensors NumericSensor_4_0_0 [activeEnergy](#)
Active energy sensor.
- sensors NumericSensor_4_0_0 [apparentEnergy](#)
Apparent energy sensor.
- sensors NumericSensor_4_0_0 [phaseAngle](#)
Phase angle sensor.
- sensors NumericSensor_4_0_0 [lineFrequency](#)
AC line frequency sensor.
- sensors StateSensor_4_0_0 [outletState](#)
Outlet power state sensor

9.196.1 Detailed Description

Outlet sensors

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Outlet.idl

9.197 sensors::Logger_2_1_2::SensorSet Struct Reference

Set of logged sensors.

```
import "SensorLogger.idl";
```

Public Attributes

- vector< [sensors.Sensor_4_0_0](#) > [sensors](#)
List of numeric or state sensors.
- vector
< [peripheral.DeviceSlot_2_0_0](#) > [slots](#)
List of peripheral device slots.

9.197.1 Detailed Description

Set of logged sensors.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SensorLogger.idl

9.198 serial::SerialPort_2_0_0 Interface Reference

Interface describing a physical serial port and the devices which can be attached to it.

```
import "SerialPort.idl";
```

Classes

- struct [Settings](#)
Port settings.
- struct [State](#)
Structure holding information about the current state of the port.

Public Types

- enum [PortState](#) { [CONSOLE](#), [ANALOGMODEM](#), [GSMMODEM](#) }
Possible states the port can be in at a given time.
- enum [BaudRate](#) { [BR1200](#), [BR2400](#), [BR4800](#), [BR9600](#), [BR19200](#), [BR38400](#), [BR57600](#), [BR115200](#) }
Possible baud rates.

Public Member Functions

- [Settings](#) [getSettings](#) ()
Get current settings.
- int [setSettings](#) (in [Settings](#) settings)
Set settings.
- [State](#) [getState](#) ()
Get current port state.
- Object [getModem](#) ()
Get modem connected to port.

Public Attributes

- constant int [SUCCESS](#) = 0
Error codes.
- constant int [ERR_INVALID_VALUE](#) = 1
Invalid arguments.
- valueobject [ModemEvent](#): [idl.Event](#) { Object modem
Event emitted when the modem connection state changes.
- valueobject [ModemAddedEvent](#): [ModemEvent](#) { }
Event emitted when a modem is connected.
- valueobject [ModemRemovedEvent](#): [ModemEvent](#) { }
Event emitted when a modem is disconnected.

9.198.1 Detailed Description

Interface describing a physical serial port and the devices which can be attached to it.

9.198.2 Member Enumeration Documentation

9.198.2.1 enum serial::SerialPort_2_0_0::BaudRate

Possible baud rates.

Enumerator

- BR1200** 1.200 kbit/s
- BR2400** 2.400 kbit/s
- BR4800** 4.800 kbit/s
- BR9600** 9.600 kbit/s
- BR19200** 19.200 kbit/s
- BR38400** 38.400 kbit/s
- BR57600** 57.600 kbit/s
- BR115200** 115.200 kbit/s

9.198.2.2 enum serial::SerialPort_2_0_0::PortState

Possible states the port can be in at a given time.

Enumerator

CONSOLE No modem is attached, the console application is running on the port.

ANALOGMODEM An analog modem is attached to the port.

GSMMODEM A GSM modem is attached to the port.

9.198.3 Member Function Documentation

9.198.3.1 Object serial::SerialPort_2_0_0::getModem ()

Get modem connected to port.

Returns

– an instance of the connected modem (either [AnalogModem](#) or [GsmModem](#))

9.198.3.2 Settings serial::SerialPort_2_0_0::getSettings ()

Get current settings.

Returns

– [Settings](#)

9.198.3.3 State serial::SerialPort_2_0_0::getState ()

Get current port state.

Returns

– Port state

9.198.3.4 int serial::SerialPort_2_0_0::setSettings (in Settings settings)

Set settings.

Parameters

<i>settings</i>	– new settings
-----------------	----------------

Returns

SUCCESS – on success

ERR_INVALID_VALUE – if any passed value was invalid

9.198.4 Member Data Documentation

9.198.4.1 valueobject serial::SerialPort_2_0_0::ModemEvent

Event emitted when the modem connection state changes.

Either a [AnalogModem](#) or a [GsmModem](#)

9.198.4.2 constant int serial::SerialPort_2_0_0::SUCCESS = 0

Error codes.

No error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/SerialPort.idl

9.199 servermon::ServerMonitor_2_0_0::Server Struct Reference

[Server](#) Entry.

```
import "ServerMonitor.idl";
```

Public Attributes

- [ServerSettings settings](#)
Server settings.
- [ServerStatus status](#)
Server status.

9.199.1 Detailed Description

[Server](#) Entry.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ServerMonitor.idl

9.200 servermon::ServerMonitor_2_0_0 Interface Reference

[Server](#) Monitor Interface.

```
import "ServerMonitor.idl";
```

Classes

- struct [Server](#)
Server Entry.
- struct [ServerSettings](#)
Server Reachability Settings.
- struct [ServerStatus](#)
Server Reachability Status.

Public Types

- enum [ServerReachability](#) { [WAITING](#), [REACHABLE](#), [UNREACHABLE](#), [ERROR](#) }
Server Reachability State.

Public Member Functions

- int [addServer](#) (out int id, in [ServerSettings](#) settings)
Add a new server entry.
- int [modifyServer](#) (in int id, in [ServerSettings](#) settings)
Modify an existing server entry.
- int [deleteServer](#) (in int id)
Delete a server entry.
- int [getServer](#) (out [Server](#) server, in int id)
Retrieve a server entry (settings and status).
- map< int, [Server](#) > [listServers](#) ()
Retrieve a list of server entries (settings and status).

Public Attributes

- constant int [ERR_NO_SUCH_ID](#) = 1
No such ID.
- constant int [ERR_INVALID_SETTINGS](#) = 2
Invalid settings.
- constant int [ERR_DUPLICATE_HOSTNAME](#) = 3
Duplicate hostname.
- constant int [ERR_MAX_SERVERS_REACHED](#) = 4
Maximum number of server entries.

9.200.1 Detailed Description

[Server](#) Monitor Interface.

9.200.2 Member Enumeration Documentation

9.200.2.1 enum servermon::ServerMonitor_2_0_0::ServerReachability

[Server](#) Reachability State.

Enumerator

- [WAITING](#)** Waiting for reliable connection.
- [REACHABLE](#)** [Server](#) is up and running.
- [UNREACHABLE](#)** No response from server.
- [ERROR](#)** Error pinging server (e.g. DNS lookup failure)

9.200.3 Member Function Documentation

9.200.3.1 int servermon::ServerMonitor_2_0_0::addServer (out int *id*, in ServerSettings *settings*)

Add a new server entry.

Parameters

<i>id</i>	New entry id, automatically assigned
<i>settings</i>	New server settings

Returns

- 0 if OK
- 2 if the settings are invalid
- 3 if an entry for the given hostname exists
- 4 if the maximum number of servers is reached

9.200.3.2 int servermon::ServerMonitor_2_0_0::deleteServer (in int *id*)

Delete a server entry.

Parameters

<i>id</i>	Entry id
-----------	----------

Returns

- 0 if OK
- 1 if the entry does not exist

9.200.3.3 int servermon::ServerMonitor_2_0_0::getServer (out Server *server*, in int *id*)

Retrieve a server entry (settings and status).

Parameters

<i>server</i>	Server settings and status
<i>id</i>	Entry id

Returns

- 0 if OK
- 1 if the entry does not exist

9.200.3.4 map<int, Server> servermon::ServerMonitor_2_0_0::listServers ()

Retrieve a list of server entries (settings and status).

Returns

[Server](#) list

9.200.3.5 int servermon::ServerMonitor_2.0.0::modifyServer (in int *id*, in ServerSettings *settings*)

Modify an existing server entry.

Parameters

<i>id</i>	Entry id
<i>settings</i>	New settings

Returns

- 0 if OK
- 1 if the entry does not exist
- 2 if the settings are invalid
- 3 if an entry for the given hostname exists

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/ServerMonitor.idl

9.201 auth::ldapsrv::ServerSettings Struct Reference

Server settings.

```
import "LdapServerSettings.idl";
```

Public Attributes

- string [id](#)
Entry ID.
- string [primaryServer](#)
IP or name of primary server.
- string [secondaryServer](#)
IP or name of secondary server.
- string [adoptSettingsId](#)
Use settings from LDAP server with <ID>
- [ServerType](#) type
Type of LDAP server.
- int [port](#)
Server port.
- int [sslPort](#)
SSL port.
- boolean [useSSL](#)
Use SSL.
- boolean [forceTrustedCert](#)
Enforce trusted certificates.
- string [certificate](#)
Certificates.
- string [adsDomain](#)
ADS domain.
- boolean [useAnonymousBind](#)
use anonymous bind

- string [bindDN](#)
Bind DN.
- string [bindPwd](#)
Bind password.
- string [searchBaseDN](#)
Base DN for search.
- string [loginNameAttr](#)
Login name attribute.
- string [userEntryObjClass](#)
User entry object class.
- string [userSearchFilter](#)
User search subfilter.

9.201.1 Detailed Description

Server settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LdapServerSettings.idl

9.202 radius::ServerSettings Struct Reference

Server settings.

```
import "RadiusServerSettings.idl";
```

Public Attributes

- string [id](#)
Entry ID.
- string [server](#)
IP or name of the radius servers.
- string **sharedSecret**
- int **udpAuthPort**
- int **udpAccountPort**
- int **timeout**
- int **retries**
- [AuthType](#) **authType**

9.202.1 Detailed Description

Server settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/RadiusServerSettings.idl

9.203 servermon::ServerMonitor_2_0_0::ServerSettings Struct Reference

[Server](#) Reachability Settings.

```
import "ServerMonitor.idl";
```

Public Attributes

- string [host](#)
Server hostname/IP address.
- boolean [enabled](#)
Pinging enabled.
- int [pingInterval](#)
Wait time after successful ping.
- int [retryInterval](#)
Wait time after unsuccessful ping.
- int [activationCount](#)
Minimum number of successful pings to enable feature.
- int [failureCount](#)
Number of unsuccessful pings to consider server down.
- int [resumeDelay](#)
Wait time before resuming pinging.
- int [resumeCount](#)
Number of resumes before going back to WAITING state.

9.203.1 Detailed Description

[Server](#) Reachability Settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ServerMonitor.idl

9.204 cert::ServerSSLCert Interface Reference

SSL certificate management interface.

```
import "ServerSSLCert.idl";
```

Classes

- struct [CertInfo](#)
Certificate information.
- struct [CommonAttributes](#)
Certificate issuer or subject attributes.
- struct [Info](#)
Certificate manager information.
- struct [ReqInfo](#)
Certificate signing request information.

Public Member Functions

- int [generateUnsignedKeyPair](#) (in [ReqInfo](#) reqInfo, in string challenge)
Generate an unsigned key pair.
- int [generateSelfSignedKeyPair](#) (in [ReqInfo](#) reqInfo, in int days)
Generate a self-signed key pair.
- void [deletePending](#) ()

Remove a pending certificate signing request or certificate.

- void [getInfo](#) (out [Info](#) info)

Retrieve certificate manager information.

- int [installPendingKeyPair](#) ()

Activate a pending key pair.

9.204.1 Detailed Description

SSL certificate management interface.

9.204.2 Member Function Documentation

9.204.2.1 int cert::ServerSSLCert::generateSelfSignedKeyPair (in ReqInfo reqInfo, in int days)

Generate a self-signed key pair.

Parameters

<i>reqInfo</i>	Certificate signing request information
<i>days</i>	Number of days the certificate will be valid

Returns

- 0 if OK
- 1 if the requested key length is invalid
- 2 if there is already a pending CSR
- 3 if the key generation failed

9.204.2.2 int cert::ServerSSLCert::generateUnsignedKeyPair (in ReqInfo reqInfo, in string challenge)

Generate an unsigned key pair.

Parameters

<i>reqInfo</i>	Certificate signing request information
<i>challenge</i>	Challenge password

Returns

- 0 if OK
- 1 if the requested key length is invalid
- 2 if there is already a pending CSR
- 3 if the key generation failed

9.204.2.3 void cert::ServerSSLCert::getInfo (out Info info)

Retrieve certificate manager information.

Parameters

<i>info</i>	Result: Certificate manager information
-------------	---

9.204.2.4 int cert::ServerSSLCert::installPendingKeyPair ()

Activate a pending key pair.

Returns

- 0 if OK
- 1 if no key is pending
- 2 if no certificate is pending
- 3 if the certificate format is invalid
- 4 if the certificate does not match the key

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/ServerSSLCert.idl

9.205 servermon::ServerMonitor_2_0_0::ServerStatus Struct Reference

[Server](#) Reachability Status.

```
import "ServerMonitor.idl";
```

Public Attributes

- [ServerReachability](#) *reachable*
Reachability state.
- time [lastRequest](#)
Timestamp of last request sent.
- time [lastResponse](#)
Timestamp of last response received.
- int [requests](#)
Number of requests sent.
- int [responses](#)
Number of responses received.
- int [failures](#)
Number of consecutive failed pings.
- int [resumes](#)
Number of resumes.

9.205.1 Detailed Description

[Server](#) Reachability Status.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/ServerMonitor.idl

9.206 event::Service_1_0_1 Interface Reference

[Event](#) Service.

```
import "EventService.idl";
```


Public Member Functions

- [Channel_1_0_1 createChannel](#) ()
Create a new event channel.
- int [destroyChannel](#) (in [Channel_1_0_1](#) channel)
Destroy an event channel.
- void [pushEvent](#) (in [idl.Event](#) event)
Push an [Event](#) into the service and to all existing receiver channels.
- void [pushEvents](#) (in vector< [idl.Event](#) > events)
Push a vector of Events into the service and to all existing receiver channels.

Public Attributes

- constant int **INVALID_CHANNEL** = 1

9.206.1 Detailed Description

[Event](#) Service.

9.206.2 Member Function Documentation

9.206.2.1 [Channel_1_0_1](#) event::Service_1_0_1::createChannel ()

Create a new event channel.

Returns

New event channel reference or nil if all channels have been used

9.206.2.2 int event::Service_1_0_1::destroyChannel (in [Channel_1_0_1](#) channel)

Destroy an event channel.

Parameters

<i>channel</i>	Event channel reference
----------------	-------------------------

Returns

0 if OK
INVALID_CHANNEL if channel is not implemented by this Service

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/EventService.idl

9.207 security::ServiceAuthorization Interface Reference

Service Authorization Configuration.

```
import "ServiceAuthorization.idl";
```

Public Member Functions

- int [setPassword](#) (in string *service*, in string *password*)
Change a certain service password.

Public Attributes

- constant int [ERR_PASSWORD_INVALID](#) = 1
Invalid password.

9.207.1 Detailed Description

Service Authorization Configuration.

9.207.2 Member Function Documentation

9.207.2.1 int security::ServiceAuthorization::setPassword (in string *service*, in string *password*)

Change a certain service password.

Parameters

<i>service</i>	The service to set the password for
<i>password</i>	The new password

Returns

- 0 if ok
- ERR_PASSWORD_INVALID if new password cannot be applied

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/ServiceAuthorization.idl

9.208 net::ServiceConfig Struct Reference

Network service configuration.

```
import "Net.idl";
```

Public Attributes

- string [service](#)
Service name.
- boolean [enable](#)
true if the service is enabled
- int [port](#)
Service TCP port.

9.208.1 Detailed Description

Network service configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.209 session::Session Struct Reference

Session information

```
import "SessionManager.idl";
```

Public Attributes

- string [token](#)
Session token to be used for authentication
- string [username](#)
Name of user owning the session.
- string [remotelp](#)
Session IP address.
- string [clientType](#)
Client type.
- time [creationTime](#)
Session creation timestamp.
- int [timeout](#)
Session timeout in seconds.
- int [idle](#)
Session idle time in seconds.
- int [userIdle](#)
User idle time in seconds.

9.209.1 Detailed Description

Session information

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SessionManager.idl

9.210 session::SessionManager Interface Reference

Session manager interface

```
import "SessionManager.idl";
```

Public Types

- enum [CloseReason](#) { [CLOSE_REASON_LOGOUT](#), [CLOSE_REASON_TIMEOUT](#), [CLOSE_REASON_BROWSER_CLOSED](#), [CLOSE_REASON_FORCED_DISCONNECT](#) }
Session close reasons

Public Member Functions

- int [newSession](#) (out [Session](#) session)
Open a new session.
- [Session](#) [getSession](#) (in string token)
Retrieve information about session identified by token.
- [Session](#) [getCurrentSession](#) ()
Retrieve current session information.
- vector< [Session](#) > [getSessions](#) ()
Retrieve all open sessions.
- void [closeSession](#) (in string token, in [CloseReason](#) reason)
Close a session identified by its token.
- void [closeCurrentSession](#) (in [CloseReason](#) reason)
Close the current session.
- void [touchSession](#) (in string token)
Reset a session's idle timer.
- void [touchCurrentSession](#) (in boolean userActivity)
Reset the current session's idle timer.
- vector< [HistoryEntry](#) > [getSessionHistory](#) ()
Get previous session data for the current user.

Public Attributes

- constant int [ERR_ACTIVE_SESSION_EXCLUSIVE_FOR_USER](#) = 1
[Session](#) creation denied due to single login limitation.

9.210.1 Detailed Description

Session manager interface

[Session](#) manager allows clients to announce a user session, i.e. consecutive activity that is related to each other, and make use of session services. Depending on transport protocol an established session allows simplified authentication using the session token. For instance, for HTTP transport implementation session token can be written into HTTP Header while other authentication schemes may be omitted (for details, see transport mapping documentation). Each session has a defined timeout and an idle timer. A session is deleted once idle time is equal or greater than session timeout. Idle timer is implicitly touched by transport implementation whenever a call arrives that can be mapped to a particular session. In addition a client may decide to call [touchCurrentSession](#) with `userActivity` flag set to true. In this case `userIdle` attribute of session is reset to 0. This has purely informational character and will not cause any further action of session manager. It may be used to determine user activity under the assumptions that clients may do frequent background calls without actual user activity.

9.210.2 Member Enumeration Documentation

9.210.2.1 enum `session::SessionManager::CloseReason`

Session close reasons

Enumerator

- `CLOSE_REASON_LOGOUT`** Regular logout.
- `CLOSE_REASON_TIMEOUT`** [Session](#) timed out.
- `CLOSE_REASON_BROWSER_CLOSED`** Browser window was closed.
- `CLOSE_REASON_FORCED_DISCONNECT`** [Session](#) was forcibly closed.

9.210.3 Member Function Documentation

9.210.3.1 void session::SessionManager::closeCurrentSession (in CloseReason *reason*)

Close the current session.

This call must be authenticated using a session token.

Parameters

<i>reason</i>	close reason
---------------	--------------

9.210.3.2 void session::SessionManager::closeSession (in string *token*, in CloseReason *reason*)

Close a session identified by its token.

Parameters

<i>token</i>	Session token
<i>reason</i>	close reason

9.210.3.3 Session session::SessionManager::getCurrentSession ()

Retrieve current session information.

This call must be authenticated using a session token.

Returns

Session information

9.210.3.4 Session session::SessionManager::getSession (in string *token*)

Retrieve information about session identified by token.

Parameters

<i>token</i>	Session token
--------------	---------------

Returns

Session information

9.210.3.5 vector<HistoryEntry> session::SessionManager::getSessionHistory ()

Get previous session data for the current user.

Returns

History data, sorted from newer to older sessions

9.210.3.6 vector<Session> session::SessionManager::getSessions ()

Retrieve all open sessions.

Returns

List of sessions

9.210.3.7 int session::SessionManager::newSession (out Session *session*)

Open a new session.

This function will create a new session for the authenticated user. Upon success it will return a session token which can be used to authenticate future requests.

Parameters

<i>session</i>	Session information
----------------	---------------------

Returns

0 if OK
1 if session creation was denied due to single login limitation

9.210.3.8 void session::SessionManager::touchCurrentSession (in boolean *userActivity*)

Reset the current session's idle timer.

Parameters

<i>userActivity</i>	Indicates that the session is touched due to user activity.
---------------------	---

If *userActivity* is not set, this is internally a NOP since any RPC call will implicitly touch the session. This call must be authenticated using a session token.

9.210.3.9 void session::SessionManager::touchSession (in string *token*)

Reset a session's idle timer.

Parameters

<i>token</i>	Session token
--------------	---------------

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/SessionManager.idl

9.211 security::Security_3_0_0::Settings Struct Reference

Security configuration This structure is deprecated and will be removed in V3.0, use concrete getters and setters instead!

```
import "Security.idl";
```

Public Attributes

- boolean [http2httpsRedir](#)
true to enable HTTP-to-HTTPS redirection

- int [userBlockTimeout](#)
User blocking timeout in minutes.
- int [userMaxFailedLogins](#)
Maximum number of failed logins before blocking a user.
- [IpFw_2_0_0](#) [ipFw](#)
IP packet filter configuration.
- [IpFw_2_0_0](#) [ipV6Fw](#)
IPv6 packet filter configuration.
- [RoleAccessControl](#) [roleAccessControl](#)
Role-based access control settings.
- [RoleAccessControl](#) [roleAccessControlV6](#)
Role-based access control settings for IPv6.
- [PasswordSettings](#) [pwSettings](#)
Password settings.
- int [idleTimeout](#)
Session idle timeout in minutes.
- boolean [singleLogin](#)
true to enable single login limitation
- [SSHSettings](#) [sshSettings](#)
SSH authentication settings.

9.211.1 Detailed Description

Security configuration This structure is deprecated and will be removed in V3.0, use concrete getters and setters instead!

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.212 devsettings::Zeroconf::Settings Struct Reference

Zero-configservice settings.

```
import "Zeroconf.idl";
```

Public Attributes

- boolean [mdnsEnabled](#)
Enable zero-config advertising via mDNS for device.

9.212.1 Detailed Description

Zero-configservice settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Zeroconf.idl

9.213 Ihxmodel::Lhx_3_2_1::Settings Struct Reference

LHX settings.

```
import "Lhx.idl";
```

Public Attributes

- double [setpointWaterValve](#)
setpoint temperature, default 20deg C
- double [setpointVentilators](#)
setpoint temperature, default 3deg C / Pa
- double [defaultFanSpeed](#)
fanspeed for 'normal' operation, default 80%

9.213.1 Detailed Description

LHX settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Lhx.idl

9.214 serial::GsmModem_1_0_1::Settings Struct Reference

Structure for holding settings of the GSM modem and its SIM card.

```
import "GsmModem.idl";
```

Public Attributes

- string [pin](#)
PIN of the SIM card.
- string [smcsc](#)
*Custom SMS center number (in ITU-T E.164 format),
leave empty to use number stored on SIM card*

9.214.1 Detailed Description

Structure for holding settings of the GSM modem and its SIM card.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/GsmModem.idl

9.215 pdumodel::Pdu_3_0_0::Settings Struct Reference

PDU settings.

```
import "Pdu.idl";
```


Public Attributes

- string [name](#)
User-defined name.
- [StartupState](#) [startupState](#)
Default outlet state on device startup; can be overridden per outlet.
- int [cycleDelay](#)
Default power-cycle interval in seconds; can be overridden per outlet.
- int [inRushGuardDelay](#)
Minimum delay in milliseconds between switching two outlets on.
- vector< int > [outletPowerStateSequence](#)
The order in which multiple outlets should be switched.
- int [powerOnDelay](#)
Delay in seconds before restoring outlet states after device startup.
- boolean [latchingRelays](#)
If true, relays keep their state during power-cycling.

9.215.1 Detailed Description

PDU settings.

9.215.2 Member Data Documentation

9.215.2.1 vector<int> pdumodel::Pdu_3_0_0::Settings::outletPowerStateSequence

The order in which multiple outlets should be switched.

Format: List of outlet numbers (zero-based), empty for default.

Affects the following functions:

- `setAllOutletPowerStates`
- `cycleAllOutletPowerStates`
- `setMultipleOutletPowerStates`
- `cycleMultipleOutletPowerStates`

The documentation for this struct was generated from the following file:

- `pdu-json-rpc-api/idl/Pdu.idl`

9.216 pdumodel::OverCurrentProtector_2_1_2::Settings Struct Reference

Overcurrent protector settings.

```
import "OverCurrentProtector.idl";
```

Public Attributes

- string [name](#)
User-defined name.

9.216.1 Detailed Description

Overcurrent protector settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/OverCurrentProtector.idl

9.217 serial::AnalogModem::Settings Struct Reference

Public Attributes

- boolean [dialInEnabled](#)
Whether dial-in to device is enabled.
- int [ringsUntilAnswer](#)
Number of rings until incoming call is answered.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AnalogModem.idl

9.218 devsettings::Modbus::Settings Struct Reference

Modbus service settings

```
import "Modbus.idl";
```

Public Attributes

- [TcpSettings](#) tcp
Modbus/TCP settings

9.218.1 Detailed Description

Modbus service settings

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Modbus.idl

9.219 pdumodel::Outlet_1_5_6::Settings Struct Reference

Outlet settings

```
import "Outlet.idl";
```

Public Attributes

- string [name](#)
User-defined name.
- [StartupState](#) startupState

- *Power state on device startup.*
boolean [usePduCycleDelay](#)
true to use power-cycle delay as defined in PDU settings
- int [cycleDelay](#)
Outlet-specific power-cycle delay
- boolean [nonCritical](#)
true if outlet is non-critical (for load shedding)
- int [sequenceDelay](#)
Delay in ms after this outlet when switching multiple outlets on.

9.219.1 Detailed Description

Outlet settings

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Outlet.idl

9.220 peripheral::DeviceSlot_2_0_0::Settings Struct Reference

user configurable slot attributes

```
import "PeripheralDeviceSlot.idl";
```

Public Attributes

- string [name](#)
User-defined name.
- string [description](#)
User-defined description.
- Location [location](#)
user-defined device location
- boolean [useDefaultThresholds](#)
use default thresholds
- map< string, string > [properties](#)
sensor specific settings

9.220.1 Detailed Description

user configurable slot attributes

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceSlot.idl

9.221 sensors::Logger_2_1_2::Settings Struct Reference

Sensor logger settings.

```
import "SensorLogger.idl";
```

Public Attributes

- boolean [isEnabled](#)
true if sensor logging is enabled
- int [samplePeriod](#)
Sensor scan interval in milliseconds.
- int [samplesPerRecord](#)
Number of samples per log record.
- int [oldestRecId](#)
Id of oldest record in ring buffer.
- int [newestRecId](#)
Id of newest record in ring buffer.
- int [logCapacity](#)
Number of log records in ring buffer.

9.221.1 Detailed Description

Sensor logger settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SensorLogger.idl

9.222 peripheral::DeviceManager_2_0_0::Settings Struct Reference

peripheral DeviceManager's s settings

```
import "PeripheralDeviceManager.idl";
```

Public Attributes

- [ZCoordMode](#) [zCoordMode](#)
Z coordinate semantics.
- boolean [autoManageNewDevices](#)
Automatically manage newly detected devices.
- float [deviceAltitude](#)
Altitude of device in meters.
- int [presenceDetectionTimeout](#)
Timeout for presence detection (sec)
- map< string,
[sensors.NumericSensor_4_0_0.Thresholds](#) > [defaultThresholdsMap](#)
Default thresholds by peripheral device type.

9.222.1 Detailed Description

peripheral DeviceManager's s settings

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceManager.idl

9.223 pdumodel::Unit_2_0_1::Settings Struct Reference

Unit settings

```
import "Unit.idl";
```

Public Attributes

- boolean [buzzerMuted](#)
true if the buzzer is muted
- boolean [autoDisplayOrientation](#)
true for automatic orientation control
- [Orientation displayOrientation](#)
Manually configured orientation

9.223.1 Detailed Description

Unit settings

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Unit.idl

9.224 pdumodel::TransferSwitch_3_1_1::Settings Struct Reference

Transfer switch settings.

```
import "TransferSwitch.idl";
```

Public Attributes

- string [name](#)
User-defined name.
- int [preferredSource](#)
Preferred inlet.
- boolean [autoRetransfer](#)
Enable automatic retransfer if power on active inlet is restored.
- boolean [noAutoRetransferIfPhaseFault](#)
Don't automatically retransfer if inlet phases are out of sync.
- int [autoRetransferWaitTime](#)
Time (in s) to delay retransfer after power restoration.
- boolean [manualTransferEnabled](#)
Enable state of 'manual transfer' front panel button.

9.224.1 Detailed Description

Transfer switch settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TransferSwitch.idl

9.225 pdumodel::Inlet_1_2_6::Settings Struct Reference

Inlet settings

```
import "Inlet.idl";
```

Public Attributes

- string [name](#)
User-defined name.

9.225.1 Detailed Description

Inlet settings

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Inlet.idl

9.226 serial::SerialPort_2_0_0::Settings Struct Reference

Port settings.

```
import "SerialPort.idl";
```

Public Attributes

- [BaudRate consoleBaudRate](#)
Baud rate to be used for running the console application.
- [BaudRate modemBaudRate](#)
Baud rate to be used for communicating with an attached modem.

9.226.1 Detailed Description

Port settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SerialPort.idl

9.227 webcam::Settings_2_0_0 Struct Reference

Webcam settings.

```
import "Webcam.idl";
```

Public Attributes

- [Format_2_0_0 format](#)
currently selected image format
- [Controls controls](#)

- *image settings like brightness, contrast, gain, ...*
- string [name](#)
webcam name
- Location [location](#)
webcam location
- int [refreshInterval](#)
in ms, toggle "video" and "static image" mode

9.227.1 Detailed Description

Webcam settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Webcam.idl

9.228 devsettings::Sntp_1_0_1 Interface Reference

SMTP settings interface.

```
import "Sntp.idl";
```

Classes

- struct [Configuration](#)
SMTP server configuration.
- struct [TestResult](#)
Result of SMTP configuration test.

Public Member Functions

- [Configuration](#) [getConfiguration](#) ()
Retrieve the SMTP server configuration.
- int [setConfiguration](#) (in [Configuration](#) cfg)
Set the SMTP server configuration.
- [TestResult](#) [testConfiguration](#) (in [Configuration](#) cfg, in vector< string > recipients)
Test an SMTP server configuration.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.

9.228.1 Detailed Description

SMTP settings interface.

9.228.2 Member Function Documentation

9.228.2.1 Configuration devsettings::Sntp_1_0_1::getConfiguration ()

Retrieve the SMTP server configuration.

Returns

SMTP server configuration

9.228.2.2 int devsettings::Sntp_1_0_1::setConfiguration (in Configuration *cfg*)

Set the SMTP server configuration.

Parameters

<i>cfg</i>	New SMTP server settings
------------	--------------------------

Returns

0 if OK
1 if any parameters are invalid

9.228.2.3 TestResult devsettings::Sntp_1_0_1::testConfiguration (in Configuration *cfg*, in vector< string > *recipients*)

Test an SMTP server configuration.

The active server configuration is not changed.

Parameters

<i>cfg</i>	SMTP server settings to test
<i>recipients</i>	Recipient email addresses

Returns

Result of configuration test

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Sntp.idl

9.229 devsettings::Snmp_1_0_2 Interface Reference

SNMP agent settings interface.

```
import "Snmp.idl";
```

Classes

- struct [Configuration](#)
SNMP agent configuration.

Public Member Functions

- [Configuration](#) [getConfiguration](#) ()
Retrieve the SNMP agent configuration.
- int [setConfiguration](#) (in [Configuration](#) cfg)
Set the SNMP agent configuration.
- string [getV3EngineId](#) ()
Retrieve the SNMP V3 Engine ID.

Public Attributes

- constant int [ERR_INVALID_PARAMS](#) = 1
Invalid parameters.
- valueobject [ConfigurationChangedEvent](#): [idl.Event](#) { string userName
user who triggered event
- string [ipAddr](#)
ip or device on which user is logged in
- [Configuration](#) [oldConfig](#)
old configuration
- [Configuration](#) [newConfig](#)
new configuration

9.229.1 Detailed Description

SNMP agent settings interface.

9.229.2 Member Function Documentation

9.229.2.1 [Configuration](#) devsettings::Snmp_1_0_2::getConfiguration ()

Retrieve the SNMP agent configuration.

Returns

SNMP agent configuration

9.229.2.2 string devsettings::Snmp_1_0_2::getV3EngineId ()

Retrieve the SNMP V3 Engine ID.

Returns

SNMP V3 Engine ID

9.229.2.3 int devsettings::Snmp_1_0_2::setConfiguration (in [Configuration](#) cfg)

Set the SNMP agent configuration.

Parameters

cfg	New SNMP agent settings
---------------------	-------------------------

Returns

- 0 if OK
- 1 if any parameters are invalid

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Snmp.idl

9.230 um::SnmpV3 Interface Reference

SNMPv3 interface.

```
import "SnmpV3.idl";
```

Public Types

- enum [SecurityLevel](#) { [NO_AUTH_NO_PRIV](#), [AUTH_NO_PRIV](#), [AUTH_PRIV](#) }
SNMP v3 security level.
- enum [AuthProtocol](#) { [MD5](#), [SHA1](#) }
SNMP v3 authentication protocol.
- enum [PrivProtocol](#) { [DES](#), [AES128](#) }
SNMP v3 privacy protocol.

9.230.1 Detailed Description

SNMPv3 interface.

9.230.2 Member Enumeration Documentation

9.230.2.1 enum um::SnmpV3::AuthProtocol

SNMP v3 authentication protocol.

Enumerator

- MD5** Use MD5 for authentication.
- SHA1** Use SHA1 for authentication.

9.230.2.2 enum um::SnmpV3::PrivProtocol

SNMP v3 privacy protocol.

Enumerator

- DES** Use DES encryption for privacy.
- AES128** Use AES encryption for privacy.

9.230.2.3 enum um::SnmpV3::SecurityLevel

SNMP v3 security level.

Enumerator

NO_AUTH_NO_PRIV No authentication and no privacy protocol.

AUTH_NO_PRIV Use authentication but no privacy protocol.

AUTH_PRIV Use both, authentication and privacy protocol.

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/SnmpV3.idl

9.231 usermgmt::SnmpV3Settings Struct Reference

SNMPv3 settings.

```
import "User.idl";
```

Public Attributes

- boolean [enabled](#)
SNMPv3 enabled.
- um SnmpV3 SecurityLevel [secLevel](#)
Security level.
- um SnmpV3 AuthProtocol [authProtocol](#)
Authentication protocol.
- boolean [usePasswordAsAuthPassphrase](#)
Use account password for SNMPv3 authentication.
- boolean [haveAuthPassphrase](#)
Authentication passphrase present.
- string [authPassphrase](#)
Authentication passphrase; cannot be read back.
- um SnmpV3 PrivProtocol [privProtocol](#)
Privacy protocol.
- boolean [useAuthPassphraseAsPrivPassphrase](#)
Use authentication passphrase as privacy passphrase.
- boolean [havePrivPassphrase](#)
Privacy passphrase present.
- string [privPassphrase](#)
Privacy passphrase; cannot be read back.

9.231.1 Detailed Description

SNMPv3 settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/User.idl

9.232 security::SSHSettings Struct Reference

SSH authentication settings.

```
import "Security.idl";
```

Public Attributes

- boolean [allowPasswordAuth](#)
Allow password authentication.
- boolean [allowPublicKeyAuth](#)
Allow public key authentication.

9.232.1 Detailed Description

SSH authentication settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Security.idl

9.233 serial::SerialPort_2_0_0::State Struct Reference

Structure holding information about the current state of the port.

```
import "SerialPort.idl";
```

Public Attributes

- [PortState](#) [state](#)
Current connection state.
- string [deviceName](#)
Name of the device currently connected.

9.233.1 Detailed Description

Structure holding information about the current state of the port.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SerialPort.idl

9.234 pdumodel::Outlet_1_5_6::State Struct Reference

Outlet state

```
import "Outlet.idl";
```

Public Attributes

- boolean [available](#)
powerState is available
- [PowerState](#) [powerState](#)
*Current power state of outlet
(represented by the control state of the relay, which was set by the last command sent to it)*
- boolean [switchOnInProgress](#)
*\c true if the outlet is pending to be switched on
after the sequencing delay has passed.*
- boolean [cycleInProgress](#)
if a power-cycle is in progress.
- [LedState](#) [ledState](#)
LED state.
- time [lastPowerStateChange](#)
Time of last power state change.

9.234.1 Detailed Description

Outlet state

9.234.2 Member Data Documentation

9.234.2.1 boolean pdumodel::Outlet_1_5_6::State::cycleInProgress

if a power-cycle is in progress.

The outlet will be switched on after the cycle delay has passed.

9.234.2.2 boolean pdumodel::Outlet_1_5_6::State::switchOnInProgress

\c true if the outlet is pending to be switched on

after the sequencing delay has passed.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Outlet.idl

9.235 sensors::StateSensor_4_0_0::State Struct Reference

Sensor state.

```
import "StateSensor.idl";
```

Public Attributes

- time [timestamp](#)
Timestamp of last sample.
- boolean [available](#)
true if the sensor is available
- int [value](#)
Discrete sensor value; interpretation depends on the type of sensor.

9.235.1 Detailed Description

Sensor state.

The documentation for this struct was generated from the following file:

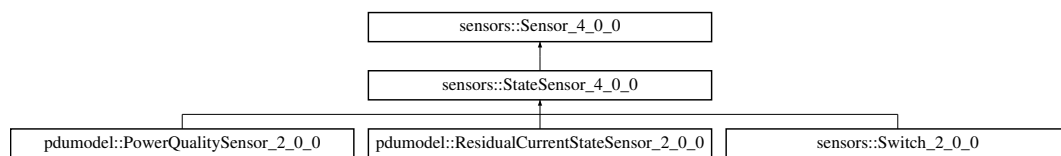
- pdu-json-rpc-api/idl/StateSensor.idl

9.236 sensors::StateSensor_4_0_0 Interface Reference

Sensor with discrete readings.

```
import "StateSensor.idl";
```

Inheritance diagram for sensors::StateSensor_4_0_0:



Classes

- struct [State](#)
Sensor state.

Public Member Functions

- [State](#) [getState](#) ()
Get the sensor state.

Public Attributes

- valueobject **StateChangedEvent**: [idl.Event](#) { [State](#) oldState
- [State](#) **newState**

Additional Inherited Members

9.236.1 Detailed Description

Sensor with discrete readings.

9.236.2 Member Function Documentation

9.236.2.1 State sensors::StateSensor_4_0_0::getState ()

Get the sensor state.

Returns

Sensor state

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/StateSensor.idl

9.237 pdumodel::Pdu_3_0_0::Statistic Struct Reference

PDU statistics.

```
import "Pdu.idl";
```

Public Attributes

- vector< [CircuitBreakerStatistic](#) > [cbStats](#)
Circuit breaker statistics.
- vector< [CtrlStatistic](#) > [ctrlStats](#)
Slave controller statistics.
- vector< [OutletStatistic](#) > [outletStats](#)
Outlet statistics
- peripheral DeviceManager_2_0_0
Statistics [peripheralStats](#)
Peripheral device statistics.

9.237.1 Detailed Description

PDU statistics.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Pdu.idl

9.238 pdumodel::TransferSwitch_3_1_1::Statistics Struct Reference

Transfer switch statistics.

```
import "TransferSwitch.idl";
```

Public Attributes

- int [transferCount](#)
Number of transfers since device startup.
- int [powerFailDetectTime](#)
Detection time in us for the last inlet power failure.
- int [relayOpenTime](#)
Time in us until all relays have opened during the last transfer.
- int [totalTransferTime](#)
Total time in us for last transfer.

9.238.1 Detailed Description

Transfer switch statistics.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TransferSwitch.idl

9.239 peripheral::DeviceManager_2_0_0::Statistics Struct Reference

Peripheral device statistics.

```
import "PeripheralDeviceManager.idl";
```

Public Attributes

- int [cSumErrCnt](#)
CRC / checksum error counter.

9.239.1 Detailed Description

Peripheral device statistics.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDeviceManager.idl

9.240 sensors::NumericSensor_4_0_0::Reading::Status Struct Reference

Numeric sensor status.

```
import "NumericSensor.idl";
```

Public Attributes

- boolean [aboveUpperCritical](#)
Reading is above upper critical threshold.
- boolean [aboveUpperWarning](#)
Reading is above upper warning threshold.
- boolean [belowLowerWarning](#)
Reading is below lower warning threshold.
- boolean [belowLowerCritical](#)
Reading is below lower critical threshold.

9.240.1 Detailed Description

Numeric sensor status.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/NumericSensor.idl

9.241 Ihxmodel::Parameter_2_0_1::Status Struct Reference

Parameter [Status](#).

```
import "LhxParameter.idl";
```

Public Attributes

- boolean [switchedOn](#)
LHX On / Off.
- boolean [active](#)
Active.
- boolean [overflow](#)
Overflow.
- boolean [underflow](#)
Underflow.
- boolean [valid](#)
Valid.

9.241.1 Detailed Description

Parameter [Status](#).

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LhxParameter.idl

9.242 webcam::StorageManager_1_0_1::StorageImage Struct Reference

[StorageImage](#).

```
import "StorageManager.idl";
```

Public Attributes

- [Image_2_0_0](#) *image*
image object
- [StorageMetaData](#) *metaData*
meta data

9.242.1 Detailed Description

[StorageImage](#).

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.243 webcam::StorageManager_1_0_1::StorageInformation Struct Reference

Information.

```
import "StorageManager.idl";
```

Public Attributes

- [StorageStatus status](#)
storage status
- int [capacity](#)
over-all nr of storable images
- int [used](#)
nr of stored images
- vector< [WebcamStorageInfo](#) > [webcamStorageInfo](#)
List of storage information for each webcam.

9.243.1 Detailed Description

Information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.244 webcam::StorageManager_1_0_1 Interface Reference

The storage manager interface.

```
import "StorageManager.idl";
```

Classes

- struct [Activity](#)
Activity.
- struct [ImageStorageMetaData](#)
StorageMetaData.
- struct [StorageImage](#)
StorageImage.
- struct [StorageInformation](#)
Information.
- struct [StorageMetaData](#)
StorageMetaData.
- struct [StorageSettings](#)
Settings.
- struct [WebcamStorageInfo](#)
Webcam Storage Info.

Public Types

- enum [StorageType](#) { [LOCAL](#), [FTP](#), [CIFS](#), [NFS](#) }
StorageType.
- enum [Direction](#) { [ASCENDING](#), [DESCENDING](#) }
Direction.
- enum [StorageStatus](#) { [INITIALIZING](#), [READY](#) }
StorageStatus.

Public Member Functions

- `vector< StorageType > getSupportedStorageTypes ()`
Get supported storage types.
- `StorageInformation getInformation ()`
get storage information
- `StorageSettings getSettings ()`
get storage settings
- `int setSettings (in StorageSettings settings)`
set storage settings
- `int addImage (in Webcam\_2\_0\_0 webcam, in Image\_2\_0\_0 image, out long index)`
add an image to the storage
- `int removeImages (in Webcam\_2\_0\_0 webcam, in long start, in int count, in Direction direction)`
remove an image of the storage
- `int getMetaData (in Webcam\_2\_0\_0 webcam, in long start, in int count, in Direction direction, out vector< ImageStorageMetaData > meta)`
get meta data of images from storage
- `int getImages (in Webcam\_2\_0\_0 webcam, in long start, in int count, in Direction direction, out vector< StorageImage > image)`
retrieve images from the storage
- `vector< Activity > getActivities ()`
get all running activities
- `int startActivity (in Webcam\_2\_0\_0 webcam, in int count, in int interval)`
start a capture activity
- `int stopActivity (in Webcam\_2\_0\_0 webcam)`
stop a capture activity

Public Attributes

- constant int `NO_ERROR` = 0
Error codes.
- constant int `ERR_INVALID_PARAM` = 1
Invalid parameter for an operation.
- constant int `ERR_INIT_IN_PROGRESS` = 2
Storage information is going to be initialized.
- constant int `ERR_ALREADY_RUNNING` = 3
The activity is already running.
- constant int `ERR_TOO_LARGE` = 4
The requested result is too large.

9.244.1 Detailed Description

The storage manager interface.

9.244.2 Member Enumeration Documentation

9.244.2.1 enum webcam::StorageManager_1_0_1::Direction

Direction.

Enumerator

ASCENDING ascending**DESCENDING** descending

9.244.2.2 enum webcam::StorageManager_1_0_1::StorageStatus

StorageStatus.

Enumerator

INITIALIZING Initializing is in progress,.**READY** Storage is ready for usage.

9.244.2.3 enum webcam::StorageManager_1_0_1::StorageType

StorageType.

Enumerator

LOCAL Local.**FTP** FTP.**CIFS** CIFS.**NFS** NFS.

9.244.3 Member Function Documentation

9.244.3.1 int webcam::StorageManager_1_0_1::addImage (in Webcam_2_0_0 *webcam*, in Image_2_0_0 *image*, out long *index*)

add an image to the storage

Parameters

<i>webcam</i>	image source webcam
<i>image</i>	image
<i>index</i>	index of the added image

Returns

NO_ERROR on success

9.244.3.2 vector<Activity> webcam::StorageManager_1_0_1::getActivities ()

get all running activities

Returns

list of running activities

9.244.3.3 `int webcam::StorageManager_1_0_1::getImages ( in Webcam_2_0_0 webcam, in long start, in int count, in Direction direction, out vector< StorageImage > image )`

retrieve images from the storage

Parameters

<i>webcam</i>	image source webcam
<i>start</i>	start index
<i>count</i>	number of images
<i>direction</i>	index counting direction
<i>image</i>	result: list of storage images

Returns

NO_ERROR on success
ERR_TOO_LARGE too many images requested

9.244.3.4 **StorageInformation** `webcam::StorageManager_1_0_1::getInformation ( )`

get storage information

Returns

[StorageInformation](#)

9.244.3.5 `int webcam::StorageManager_1_0_1::getMetaData ( in Webcam_2_0_0 webcam, in long start, in int count, in Direction direction, out vector< ImageStorageMetaData > meta )`

get meta data of images from storage

Parameters

<i>webcam</i>	image source webcam
<i>start</i>	start index
<i>count</i>	number of images
<i>direction</i>	index counting direction
<i>meta</i>	result: list of storage meta data

Returns

NO_ERROR on success
ERR_TOO_LARGE too many information requested

9.244.3.6 **StorageSettings** `webcam::StorageManager_1_0_1::getSettings ( )`

get storage settings

Returns

[StorageSettings](#)

9.244.3.7 `vector<StorageType> webcam::StorageManager_1_0_1::getSupportedStorageTypes ( )`

Get supported storage types.

Returns

a list of supported storage types

9.244.3.8 `int webcam::StorageManager_1_0_1::removeImages ( in Webcam_2_0_0 webcam, in long start, in int count, in Direction direction )`

remove an image of the storage

Parameters

<i>webcam</i>	image source webcam
<i>start</i>	start index
<i>count</i>	number of images
<i>direction</i>	index counting direction

Returns

NO_ERROR on success

9.244.3.9 `int webcam::StorageManager_1_0_1::setSettings ( in StorageSettings settings )`

set storage settings

Parameters

<i>settings</i>	settings structure
-----------------	--------------------

Returns

NO_ERROR on success

ERR_INVALID_PARAM invalid settings

9.244.3.10 `int webcam::StorageManager_1_0_1::startActivity ( in Webcam_2_0_0 webcam, in int count, in int interval )`

start a capture activity

Parameters

<i>webcam</i>	webcam
<i>count</i>	number of images to store, zero is interpreted as infinite
<i>interval</i>	interval in ms

Returns

NO_ERROR on success

ERR_INVALID_PARAM webcam not found

9.244.3.11 int webcam::StorageManager_1_0_1::stopActivity (in Webcam_2_0_0 webcam)

stop a capture activity

Parameters

<i>webcam</i>	webcam
---------------	--------

Returns

NO_ERROR on success

ERR_INVALID_PARAM no matching activity found

9.244.4 Member Data Documentation

9.244.4.1 constant int webcam::StorageManager_1_0_1::NO_ERROR = 0

Error codes.

Operation successful, no error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.245 webcam::StorageManager_1_0_1::StorageMetaData Struct Reference

[StorageMetaData](#).

```
import "StorageManager.idl";
```

Public Attributes

- long [index](#)
current image index
- [Webcam_2_0_0 webcam](#)
source webcam

9.245.1 Detailed Description

[StorageMetaData](#).

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.246 webcam::StorageManager_1_0_1::StorageSettings Struct Reference

Settings.

```
import "StorageManager.idl";
```

Public Attributes

- [StorageType](#) `type`
storage type
- `int` [capacity](#)
nr of storable images, must ont be greater than the over-all totalCapacity
- `string` [server](#)
server ip (empty/ignored for LOCAL storage type)
- `string` [username](#)
username (empty/ignored for LOCAL storage type)
- `string` [password](#)
password (empty/ignored for LOCAL storage type)

9.246.1 Detailed Description

Settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.247 assetmgrmodel::AssetStrip_2_0_1::StripInfo Struct Reference

Dynamic (may change with a connected strip) information for an AssetStrip.

```
import "AssetStrip.idl";
```

Public Attributes

- `int` [maxMainTagCount](#)
Maximum number of tags supported on the main strip.
- `int` [maxBladeTagCount](#)
Maximum number of tags supported on blade extensions.
- `int` [mainTagCount](#)
Current number of tags on the main asset strip.
- `int` [bladeTagCount](#)
Current number of tags on all blade extensions.
- `boolean` [bladeOverflow](#)
Out of space for new blade extension tags, read-only.
- `int` [rackUnitCount](#)
Rack unit count, i.e. number of tags connectable.
- `int` [componentCount](#)
Number of components.
- [CascadeState](#) `cascadeState`
State of the cascade (only for composite strips)

9.247.1 Detailed Description

Dynamic (may change with a connected strip) information for an AssetStrip.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStrip.idl

9.248 assetmgrmodel::AssetStripConfig_1_0_1::StripSettings Struct Reference

Settings for this Asset Strip.

```
import "AssetStripConfig.idl";
```

Public Attributes

- int [rackUnitCount](#)
rack unit count, number of rack units (range: 8..64)
- string [name](#)
user defined name (up to 64 alphanumeric characters)
- [ScanMode](#) [scanMode](#)
LED scan (demo) mode.
- [LEDColor](#) [defaultColorConnected](#)
auto color for rack units with an asset tag connected
- [LEDColor](#) [defaultColorDisconnected](#)
auto color for rack units without an asset tag connected
- [NumberingMode](#) [numberingMode](#)
rack unit numbering mode (top down, bottom up)
- int [numberingOffset](#)
rack unit numbering starting offset (default is 1)
- [Orientation](#) [orientation](#)
orientation. If orientationSensAvailable, writes are ignored

9.248.1 Detailed Description

Settings for this Asset Strip.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStripConfig.idl

9.249 Lhx::Support Interface Reference

LHX [Support](#) Interface.

```
import "LhxSupport.idl";
```

Public Member Functions

- void [setEnabled](#) (in boolean enabled)
Set the enabled state for LHX [Support](#) feature.
- boolean [isEnabled](#) ()
Determine the enabled state of LHX [Support](#) feature.

9.249.1 Detailed Description

LHX [Support](#) Interface.

9.249.2 Member Function Documentation

9.249.2.1 boolean `lhx::Support::isEnabled` ()

Determine the enabled state of LHX [Support](#) feature.

Returns

true feature enabled
false feature disabled or unknown

9.249.2.2 void `lhx::Support::setEnabled` (in boolean *enabled*)

Set the enabled state for LHX [Support](#) feature.

Parameters

<i>enabled</i>	the feature state
----------------	-------------------

The documentation for this interface was generated from the following file:

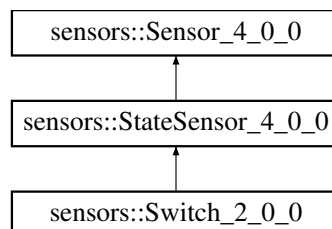
- pdu-json-rpc-api/idl/LhxSupport.idl

9.250 `sensors::Switch_2_0_0` Interface Reference

Switch is an actuator and an extension to StateSensor.

```
import "Switch.idl";
```

Inheritance diagram for `sensors::Switch_2_0_0`:



Public Member Functions

- int [setState](#) (in int newState)
This method outputs the given value.

Public Attributes

- constant int **ERR_INVALID_PARAMETER** = 1
- constant int **ERR_NOT_AVAILABLE** = 2

Additional Inherited Members

9.250.1 Detailed Description

Switch is an actuator and an extension to StateSensor.

An actuator actively outputs its state. In case of a Dry-Contact, for instance, the switch may output ON or OFF. Because Switch is a StateSensor it is also possible to query the actual state of the Switch. The type of switch is determined by the sensor's TypeSpec

9.250.2 Member Function Documentation

9.250.2.1 int sensors::Switch_2_0_0::setState (in int *newState*)

This method outputs the given value.

The int input value is the counterpart of the value returned by StateSensor's getState method. Valid values are determined by the StateSensor's TypeSpec

Parameters

<i>newState</i>	the new state the switch shall switch to
-----------------	--

Returns

- 0 if OK
- ERR_INVALID_PARAMETER if invalid parameter
- ERR_NOT_AVAILABLE if unable to set new state

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Switch.idl

9.251 sys::System Interface Reference

[System](#) access methods.

```
import "System.idl";
```

Public Member Functions

- boolean [isDaemonRunning](#) (in string name)
Check whether a daemon process is running.
- void [restartDaemon](#) (in string name)
Restart a daemon process.

9.251.1 Detailed Description

[System](#) access methods.

9.251.2 Member Function Documentation

9.251.2.1 boolean sys::System::isDaemonRunning (in string *name*)

Check whether a daemon process is running.

Parameters

<i>name</i>	Daemon name
-------------	-------------

Returns

`true` if the daemon process is running

9.251.2.2 void sys::System::restartDaemon (in string *name*)

Restart a daemon process.

Parameters

<i>name</i>	Daemon name
-------------	-------------

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/System.idl

9.252 assetmgrmodel::AssetStrip_2_0_1::TagChangeInfo Struct Reference

Information describing a tag change.

```
import "AssetStrip.idl";
```

Public Attributes

- [TagInfo tag](#)
Tag which was attached or detached.
- [RackUnitInfo info](#)
Rack unit the tag was/is connected to.
- string [parentBladeTagId](#)
*Asset tag ID of the parent blade tag,
empty if the tag is not an extension tag*
- int [slotPosition](#)
*Blade slot position of the tag,
0 if the tag is not an extension tag*

9.252.1 Detailed Description

Information describing a tag change.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStrip.idl

9.253 assetmgrmodel::AssetStrip_2_0_1::TagInfo Struct Reference

Information for a single tag.

```
import "AssetStrip.idl";
```

Public Attributes

- int [rackUnitNumber](#)
The rack unit this tag is connected to, range 0..rackUnitCount-1.
- int [slotNumber](#)
Blade slot this tag is connected to, 0 is the main strip, >0 for blades.
- string [familyDesc](#)
Tag family description, indicating different tag hardware.
- string [rawId](#)
The asset tag ID (6 byte hexadecimal string 'AABBCCDDEEFF')

9.253.1 Detailed Description

Information for a single tag.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/AssetStrip.idl

9.254 devsettings::Modbus::TcpSettings Struct Reference

Modbus/TCP settings.

```
import "Modbus.idl";
```

Public Attributes

- boolean [readonly](#)
Disallow write requests.

9.254.1 Detailed Description

Modbus/TCP settings.

Note

Port number and enabled flag are configured using `net::Net::setNetworkConfigServices`.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Modbus.idl

9.255 devsettings::Sntp_1_0_1::TestResult Struct Reference

Result of SMTP configuration test.

```
import "Sntp.idl";
```

Public Attributes

- int [status](#)
Status code; 0 if OK.
- string [message](#)
Status message.

9.255.1 Detailed Description

Result of SMTP configuration test.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Smtp.idl

9.256 sensors::NumericSensor_4_0_0::ThresholdCapabilities Struct Reference

Threshold capabilities.

```
import "NumericSensor.idl";
```

Public Attributes

- boolean [hasUpperCritical](#)
Sensor has upper critical threshold.
- boolean [hasUpperWarning](#)
Sensor has upper warning threshold.
- boolean [hasLowerWarning](#)
Sensor has lower warning threshold.
- boolean [hasLowerCritical](#)
Sensor has lower critical threshold.

9.256.1 Detailed Description

Threshold capabilities.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/NumericSensor.idl

9.257 sensors::NumericSensor_4_0_0::Thresholds Struct Reference

Numeric sensor thresholds.

```
import "NumericSensor.idl";
```

Public Attributes

- boolean [upperCriticalActive](#)
true if the upper critical threshold is enabled
- double [upperCritical](#)
Upper critical threshold.
- boolean [upperWarningActive](#)
true if the upper warning threshold is enabled
- double [upperWarning](#)
Upper warning threshold.
- boolean [lowerWarningActive](#)
true if the lower warning threshold is enabled

- double [lowerWarning](#)
Lower warning threshold.
- boolean [lowerCriticalActive](#)
true if the lower critical threshold is enabled
- double [lowerCritical](#)
Lower critical threshold.
- int [assertionTimeout](#)
Assertion timeout in samples.
- float [deassertionHysteresis](#)
Deassertion hysteresis.

9.257.1 Detailed Description

Numeric sensor thresholds.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/NumericSensor.idl

9.258 pdumodel::ThrowPole Struct Reference

A pole that can select one of multiple inputs.

```
import "Pole.idl";
```

Public Attributes

- string [label](#)
Pole label
- [PowerLine](#) [line](#)
Power line.
- vector< int > [inNodeIds](#)
Upstream node ids.
- int [outNodeId](#)
Downstream node id.

9.258.1 Detailed Description

A pole that can select one of multiple inputs.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Pole.idl

9.259 sensors::Logger_2_1_2::TimedRecord Struct Reference

Sensor log record with timestamp.

```
import "SensorLogger.idl";
```

Public Attributes

- time [timestamp](#)
Timestamp.
- [Record](#) [record](#)
Log record.

9.259.1 Detailed Description

Sensor log record with timestamp.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/SensorLogger.idl

9.260 [event::TimerEventManager_2_0_0::TimerEvent](#) Struct Reference

[TimerEvent](#) structure.

```
import "TimerEventManager.idl";
```

Public Attributes

- vector< string > [eventId](#)
Event ID.
- [Schedule](#) [executionTime](#)
Schedule for execution time.

9.260.1 Detailed Description

[TimerEvent](#) structure.

A [TimerEvent](#) can be used in the event engine to run actions at a specified time. The 'eventId' is an unique identification of every timer event. The 'executionTime' contains the schedule for the execution time.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TimerEventManager.idl

9.261 [event::TimerEventManager_2_0_0](#) Interface Reference

[TimerEventManager](#) interface.

```
import "TimerEventManager.idl";
```

Classes

- struct [Range](#)
Range structure.
- struct [Schedule](#)
Schedule structure.
- struct [TimerEvent](#)
TimerEvent structure.

Public Member Functions

- int [addTimerEvent](#) (in [Schedule](#) schedule, out vector< string > eventId)
Add a new timer event.
- int [modifyTimerEvent](#) (in vector< string > eventId, in [Schedule](#) schedule)
Modify a timer event.
- int [deleteTimerEvent](#) (in vector< string > eventId)
Delete a timer event.
- vector< [TimerEvent](#) > [listTimerEvents](#) ()
List all timer events.

Public Attributes

- constant int [NO_ERROR](#) = 0
Error codes.
- constant int [ERR_INVALID_SCHEDULE](#) = 1
failure in schedule
- constant int [ERR_UNKNOWN_EVENT_ID](#) = 2
unknown eventId
- constant int [ERR_CREATE_EVENT_ID_FAILED](#) = 3
creating eventId failed
- constant int [ERR_MAX_TIMERS_CREATED](#) = 4
max number of timers have been created
- constant int [JAN](#) = 1
[Schedule](#) defines and structures.
- constant int [FEB](#) = 2
February.
- constant int [MAR](#) = 3
March.
- constant int [APR](#) = 4
April.
- constant int [MAY](#) = 5
May.
- constant int [JUN](#) = 6
June.
- constant int [JUL](#) = 7
July.
- constant int [AUG](#) = 8
August.
- constant int [SEP](#) = 9
September.
- constant int [OCT](#) = 10
October.
- constant int [NOV](#) = 11
November.
- constant int [DEC](#) = 12
December.
- constant int [SUN](#) = 0
Days of week.
- constant int [MON](#) = 1
Monday.

- constant int **TUE** = 2
Tuesday.
- constant int **WED** = 3
Wednesday.
- constant int **THU** = 4
Thursday.
- constant int **FRI** = 5
Friday.
- constant int **SAT** = 6
Saturday.

9.261.1 Detailed Description

TimerEventManager interface.

9.261.2 Member Function Documentation

9.261.2.1 int event::TimerEventManager_2_0_0::addTimerEvent (in Schedule *schedule*, out vector< string > *eventId*)

Add a new timer event.

The timer event id field is allocated automatically and returned in the timerEventId parameter.

Parameters

<i>schedule</i>	schedule for timer
<i>eventId</i>	created event id

Returns

NO_ERROR if OK
 ERR_INVALID_SCHEDULE if schedule is invalid
 ERR_CREATE_EVENT_ID_FAILED if generating the eventId failed
 ERR_MAX_TIMERS_CREATED if max timer count already created

9.261.2.2 int event::TimerEventManager_2_0_0::deleteTimerEvent (in vector< string > *eventId*)

Delete a timer event.

Parameters

<i>eventId</i>	event id
----------------	----------

Returns

NO_ERROR if OK
 ERR_UNKNOWN_EVENT_ID if eventId is unknown

9.261.2.3 int event::TimerEventManager_2_0_0::modifyTimerEvent (in vector< string > *eventId*, in Schedule *schedule*)

Modify a timer event.

Parameters

<i>eventId</i>	event id
<i>schedule</i>	new schedule for timer

Returns

NO_ERROR if OK
 ERR_INVALID_SCHEDULE if schedule is invalid
 ERR_UNKNOWN_EVENT_ID if eventId is unknown

9.261.3 Member Data Documentation

9.261.3.1 constant int event::TimerEventManager_2_0_0::JAN = 1

[Schedule](#) defines and structures.

The goal of the following declarations is to express crontab entries in a structured way. Months January

9.261.3.2 constant int event::TimerEventManager_2_0_0::NO_ERROR = 0

Error codes.

operation successful, no error

9.261.3.3 constant int event::TimerEventManager_2_0_0::SUN = 0

Days of week.

Sunday

The documentation for this interface was generated from the following file:

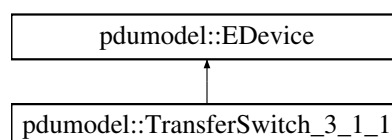
- pdu-json-rpc-api/idl/TimerEventManager.idl

9.262 pdumodel::TransferSwitch_3_1_1 Interface Reference

Transfer switch interface.

```
import "TransferSwitch.idl";
```

Inheritance diagram for pdumodel::TransferSwitch_3_1_1:



Classes

- struct [MetaData](#)
Transfer switch metadata.
- struct [Sensors](#)
Transfer switch sensors.

- struct [Settings](#)
Transfer switch settings.
- struct [Statistics](#)
Transfer switch statistics.
- struct [WaveformSample](#)
Sample of voltage and current waveform.

Public Types

- enum [Type](#) { STS, ATS, HTS }
Transfer switch type.
- enum [TransferReason](#) {
REASON_UNKNOWN, REASON_STARTUP, REASON_MANUAL_TRANSFER, REASON_AUTO_RETRANSFER,
REASON_POWER_FAILURE, REASON_POWER_QUALITY, REASON_OVERLOAD, REASON_OVERHEAT,
REASON_INTERNAL_FAILURE }

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the transfer switch metadata.
- [Sensors](#) [getSensors](#) ()
Get the transfer switch sensors.
- vector< [ThrowPole](#) > [getPoles](#) ()
Get the list of transfer switch poles.
- [Settings](#) [getSettings](#) ()
Retrieve the transfer switch settings.
- int [setSettings](#) (in [Settings](#) settings)
Change the transfer switch settings.
- [Statistics](#) [getStatistics](#) ()
Retrieve the transfer switch statistics.
- int [transferToSource](#) (in int source, in boolean faultOverride)
Select the active inlet.
- [TransferReason](#) [getLastTransferReason](#) ()
Get the reason for the last transfer.
- vector< [WaveformSample](#) > [getLastTransferWaveform](#) ()
Get the voltage and current waveforms during the last transfer.

Public Attributes

- constant int [ERR_INVALID_PARAM](#) = 1
Invalid parameters.
- constant int [ERR_SWITCH_PREVENTED](#) = 2
Switching failed due to an alarm that may be overridden.
- constant int [ERR_SWITCH_FAILED](#) = 3
Switching failed, no override possible.
- constant int [OPERATIONAL_STATE_OFF](#) = 0
Both inlets are off.
- constant int [OPERATIONAL_STATE_NORMAL](#) = 1
Active inlet equals preferred inlet.

- constant int `OPERATIONAL_STATE_STANDBY` = 2
Active inlet and preferred inlet are different.
- constant int `SWITCH_FAULT_I1_SHORT` = 1
Inlet 1 switch is permanently closed.
- constant int `SWITCH_FAULT_I1_OPEN` = 2
Inlet 1 switch is permanently open.
- constant int `SWITCH_FAULT_I2_SHORT` = 4
Inlet 2 switch is permanently closed.
- constant int `SWITCH_FAULT_I2_OPEN` = 8
Inlet 2 switch is permanently open.
- valueobject `SettingsChangedEvent`: `event.UserEvent { Settings oldSettings`
Event: Transfer switch settings have been changed.
- `Settings newSettings`
Settings after change.

9.262.1 Detailed Description

Transfer switch interface.

9.262.2 Member Enumeration Documentation

9.262.2.1 enum pdumodel::TransferSwitch_3_1_1::TransferReason

Enumerator

- REASON_UNKNOWN** Transfer reason unknown.
- REASON_STARTUP** Startup or return to normal conditions.
- REASON_MANUAL_TRANSFER** Manual transfer.
- REASON_AUTO_RETRANSFER** Automatic retransfer.
- REASON_POWER_FAILURE** Previous inlet power failed.
- REASON_POWER_QUALITY** New inlet provided better power quality.
- REASON_OVERLOAD** Switched off due to overload alarm.
- REASON_OVERHEAT** Switched off due to overheat alarm.
- REASON_INTERNAL_FAILURE** Transferred because of hardware failure (e.g. switch fault)

9.262.2.2 enum pdumodel::TransferSwitch_3_1_1::Type

Transfer switch type.

Enumerator

- STS** Static transfer switch (using SCRs as switch technology)
- ATS** Asynchronous transfer switch (using relays)
- HTS** Hybrid transfer switch (relays plus SCRs)

9.262.3 Member Function Documentation

9.262.3.1 **TransferReason** pdumodel::TransferSwitch_3_1_1::getLastTransferReason ()

Get the reason for the last transfer.

Returns

Last transfer reason

9.262.3.2 **vector<WaveformSample>** pdumodel::TransferSwitch_3_1_1::getLastTransferWaveform ()

Get the voltage and current waveforms during the last transfer.

Note

The interval between two samples is 1/4800 s. This is subject to change in future updates.

Returns

Waveform samples

9.262.3.3 **MetaData** pdumodel::TransferSwitch_3_1_1::getMetaData ()

Retrieve the transfer switch metadata.

Returns

Transfer switch metadata

9.262.3.4 **vector<ThrowPole>** pdumodel::TransferSwitch_3_1_1::getPoles ()

Get the list of transfer switch poles.

Returns

List of poles

9.262.3.5 **Sensors** pdumodel::TransferSwitch_3_1_1::getSensors ()

Get the transfer switch sensors.

Returns

Transfer switch sensors

9.262.3.6 **Settings** pdumodel::TransferSwitch_3_1_1::getSettings ()

Retrieve the transfer switch settings.

Returns

Transfer switch settings

9.262.3.7 Statistics pdumodel::TransferSwitch_3_1_1::getStatistics ()

Retrieve the transfer switch statistics.

Returns

Transfer switch statistics

9.262.3.8 int pdumodel::TransferSwitch_3_1_1::setSettings (in Settings settings)

Change the transfer switch settings.

Returns

0 if OK
1 if any parameters are invalid

9.262.3.9 int pdumodel::TransferSwitch_3_1_1::transferToSource (in int source, in boolean faultOverride)

Select the active inlet.

If the new inlet is available, it will become both active and preferred.

Parameters

<i>source</i>	New active inlet index
<i>faultOverride</i>	Force switch even if the phase sync angle between the inputs is too large

Returns

0 if OK
ERR_INVALID_PARAM if the selected source is invalid
ERR_SWITCH_PREVENTED if switching failed due to an alarm that may be overridden
ERR_SWITCH_FAILED if switching failed and no override is possible

9.262.4 Member Data Documentation**9.262.4.1 valueobject pdumodel::TransferSwitch_3_1_1::SettingsChangedEvent**

Event: Transfer switch settings have been changed.

[Settings](#) before change

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/TransferSwitch.idl

9.263 sensors::Sensor_4_0_0::TypeSpec Struct Reference

Complete sensor type specification.

```
import "Sensor.idl";
```

Public Attributes

- int [readingtype](#)

- Sensor reading type*
- int [type](#)
- Sensor type*
- int [unit](#)
- Sensor unit*

9.263.1 Detailed Description

Complete sensor type specification.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Sensor.idl

9.264 test::Unit_1_0_2 Interface Reference

Test interface for PDU components controlled by topofw.

```
import "TestUnit.idl";
```

Public Member Functions

- vector< [Display_1_0_1](#) > [getDisplayStates](#) ()
Retrieve all Displays attached to the unit.
- vector< boolean > [getButtonStates](#) ()
Retrieve state of all buttons attached to the unit.
- void [setBuzzer](#) (in boolean isOn)
Force on the buzzer.
- void [resetAllSlaveControllers](#) ()
Reset all slave controllers via RS485 break condition.
- void [triggerSlaveControllerWatchdog](#) (in int rs485Addr)
Trigger watchdog of a selected slave controller.

9.264.1 Detailed Description

Test interface for PDU components controlled by topofw.

9.264.2 Member Function Documentation

9.264.2.1 vector<boolean> test::Unit_1_0_2::getButtonStates ()

Retrieve state of all buttons attached to the unit.

Note that semantics of the buttons is not defined by this interface, although it is guaranteed that the order in which states are returned will not change.

Returns

List of button states (`true` if pressed)

9.264.2.2 `vector<Display_1_0_1> test::Unit_1_0_2::getDisplays ( )`

Retrieve all Displays attached to the unit.

Returns

List of displays

9.264.2.3 `void test::Unit_1_0_2::setBuzzer ( in boolean isOn )`

Force on the buzzer.

Parameters

<i>isOn</i>	true to force buzzer on, false for normal mode
-------------	--

9.264.2.4 `void test::Unit_1_0_2::triggerSlaveControllerWatchdog ( in int rs485Addr )`

Trigger watchdog of a selected slave controller.

Parameters

<i>rs485Addr</i>	RS485 address of the slave to trigger the watchdog for.
------------------	---

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/TestUnit.idl

9.265 pdumodel::Unit_2_0_1 Interface Reference

Unit interface.

```
import "Unit.idl";
```

Classes

- struct [MetaData](#)
Unit metadata
- struct [Settings](#)
Unit settings

Public Types

- enum [Orientation](#) { [NORMAL](#), [FLIPPED](#) }
Display orientation.

Public Member Functions

- [MetaData](#) [getMetaData](#) ()
Retrieve the unit metadata.
- [Settings](#) [getSettings](#) ()

- Retrieve the unit settings.*
 - int [setSettings](#) (in [Settings](#) settings)
- Change the unit settings.*
 - void [identify](#) (in int seconds)
- Display something distinctive to identify the unit.*
 - void [muteBuzzer](#) (in boolean mute)
- Mute buzzer, turn of all audible alarms.*
 - [Orientation](#) [getDisplayOrientation](#) ()
- Retrieve the current orientation of the display.*

Public Attributes

- constant int [ERR_INVALID_PARAM](#) = 1
- Invalid parameters.*
- valueobject [IdentificationStartedEvent](#): [event.UserEvent](#) { int duration
- Event: Unit identification requested.*

9.265.1 Detailed Description

Unit interface.

9.265.2 Member Enumeration Documentation

9.265.2.1 enum [pdumodel::Unit_2_0_1::Orientation](#)

Display orientation.

Enumerator

- NORMAL*** Normal orientation.
- FLIPPED*** Upside-down.

9.265.3 Member Function Documentation

9.265.3.1 [Orientation](#) [pdumodel::Unit_2_0_1::getDisplayOrientation](#) ()

Retrieve the current orientation of the display.

Returns

Display orientation.

9.265.3.2 [MetaData](#) [pdumodel::Unit_2_0_1::getMetaData](#) ()

Retrieve the unit metadata.

Returns

Unit metadata

9.265.3.3 Settings pduModel::Unit_2_0_1::getSettings ()

Retrieve the unit settings.

Returns

Unit settings

9.265.3.4 void pduModel::Unit_2_0_1::identify (in int *seconds*)

Display something distinctive to identify the unit.

Parameters

<i>seconds</i>	Number of seconds to display the identify string
----------------	--

9.265.3.5 void pduModel::Unit_2_0_1::muteBuzzer (in boolean *mute*)

Mute buzzer, turn of all audible alarms.

Parameters

<i>mute</i>	true to mute buzzer, false for normal mode
-------------	--

9.265.3.6 int pduModel::Unit_2_0_1::setSettings (in Settings *settings*)

Change the unit settings.

Parameters

<i>settings</i>	New unit settings
-----------------	-------------------

Returns

0 if OK
ERR_INVALID_PARAM if any parameters are invalid

9.265.4 Member Data Documentation

9.265.4.1 valueobject pduModel::Unit_2_0_1::IdentificationStartedEvent

Event: Unit identification requested.

Number of seconds the identification should be displayed

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Unit.idl

9.266 firmware::UpdateHistoryEntry Struct Reference

Firmware update history entry TODO: implement CR# 45668 on next interface change add comment field based on firmware tag "char tag[64];" to improve firmware update history entries without rootfs images

```
import "Firmware.idl";
```

Public Attributes

- time [timestamp](#)
Timestamp when the update was started.
- string [oldVersion](#)
Previous firmware version.
- string [imageVersion](#)
Firmware version of update image.
- string [imageMD5](#)
MD5 hash of update image.
- [UpdateHistoryStatus](#) [status](#)
Update status.

9.266.1 Detailed Description

Firmware update history entry TODO: implement CR# 45668 on next interface change add comment field based on firmware tag "char tag[64];" to improve firmware update history entries without rootfs images

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Firmware.idl

9.267 firmware::UpdateStatus Struct Reference

Firmware update status

```
import "FirmwareUpdateStatus.idl";
```

Public Attributes

- string [state](#)
Current state of firmware update: NONE, INIT, PREP, SIMULATE, UPDATE, SUCCESS or FAIL.
- int [elapsed](#)
Seconds since start of update, 0 if not available.
- int [estimated](#)
Estimated total time for update, 0 if not available.
- string [error_message](#)
Error message; empty if there was no error.

9.267.1 Detailed Description

Firmware update status

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/FirmwareUpdateStatus.idl

9.268 usb::Usb Interface Reference

USB interface.

```
import "Usb.idl";
```

Public Member Functions

- void [getDevices](#) (out vector< [UsbDevice](#) > usbDevices)
Get a list of USB devices connected to the host port.

9.268.1 Detailed Description

USB interface.

9.268.2 Member Function Documentation

9.268.2.1 void usb::Usb::getDevices (out vector< [UsbDevice](#) > *usbDevices*)

Get a list of USB devices connected to the host port.

Parameters

<i>usbDevices</i>	Result: List of discovered devices
-------------------	------------------------------------

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Usb.idl

9.269 usb::UsbDevice Struct Reference

USB device information.

```
import "Usb.idl";
```

Public Attributes

- int [bus](#)
Bus number.
- int [device](#)
Device address.
- int [vendorId](#)
Vendor ID.
- int [productId](#)
Product ID.

9.269.1 Detailed Description

USB device information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Usb.idl

9.270 usermgmt::User_1_0_1 Interface Reference

User interface

```
import "User.idl";
```

Public Member Functions

- [UserInfo](#) [getInfo](#) ()
Get user information.
- int [setAccountPassword](#) (in string password)
Set the account password.
- int [updateAccountFull](#) (in string password, in [UserInfo](#) info)
Update user information.
- void [getInfoAndPrivileges](#) (out [UserInfo](#) info, out vector< [Role.Privilege](#) > privileges)
Get information and a list of granted privileges for a user.
- int [setPreferences](#) (in [Preferences](#) prefs)
Sets the user preferences.
- [UserCapabilities](#) [getCapabilities](#) ()
Gets the user capabilities.

Public Attributes

- constant int [ERR_PASSWORD_UNCHANGED](#) = 1
The new password must differ from the old password.
- constant int [ERR_PASSWORD_EMPTY](#) = 2
The password must not be empty.
- constant int [ERR_PASSWORD_TOO_SHORT](#) = 3
The password is too short.
- constant int [ERR_PASSWORD_TOO_LONG](#) = 4
The password is too long.
- constant int [ERR_PASSWORD_CTRL_CHARS](#) = 5
The password must not contain control characters.
- constant int [ERR_PASSWORD_NEED_LOWER](#) = 6
The password must contain at least one lower-case character.
- constant int [ERR_PASSWORD_NEED_UPPER](#) = 7
The password must contain at least one upper-case character.
- constant int [ERR_PASSWORD_NEED_NUMERIC](#) = 8
The password must contain at least one numeric character.
- constant int [ERR_PASSWORD_NEED_SPECIAL](#) = 9
The password must contain at least one special character.
- constant int [ERR_PASSWORD_IN_HISTORY](#) = 10
The password is already in the password history.
- constant int [ERR_PASSWORD_TOO_SHORT_FOR_SNMP](#) = 11
The password is too short to be used as SNMPv3 passphrase.
- constant int [ERR_INVALID_ARGUMENT](#) = 12
Invalid arguments.

9.270.1 Detailed Description

User interface

9.270.2 Member Function Documentation

9.270.2.1 UserCapabilities usermgmt::User_1_0_1::getCapabilities ()

Gets the user capabilities.

Returns

capabilities

9.270.2.2 UserInfo usermgmt::User_1_0_1::getInfo ()

Get user information.

Returns

User information

9.270.2.3 void usermgmt::User_1_0_1::getInfoAndPrivileges (out UserInfo *info*, out vector< Role.Privilege > *privileges*)

Get information and a list of granted privileges for a user.

Parameters

<i>info</i>	User information
<i>privileges</i>	List of granted privileges

9.270.2.4 int usermgmt::User_1_0_1::setAccountPassword (in string *password*)

Set the account password.

Parameters

<i>password</i>	The new password
-----------------	------------------

Returns

- 0 OK
- 1 The new password has to differ from old password.
- 2 The password must not be empty.
- 3 The password is too short.
- 4 The password is too long.
- 5 The password must not contain control characters.
- 6 The password has to contain at least one lower case character.
- 7 The password has to contain at least one upper case character.
- 8 The password has to contain at least one numeric character.
- 9 The password has to contain at least one printable special character.
- 10 The password already is in history.
- 11 SNMPv3 USM is activated for the user and the password shall be used as auth passphrase. For this case, the password is too short (must be at least 8 characters).

9.270.2.5 int usermgmt::User_1_0_1::setPreferences (in Preferences *prefs*)

Sets the user preferences.

Parameters

<i>prefs</i>	User Preferences
--------------	----------------------------------

Returns

- 0 OK
- ERR_INVALID_ARGUMENT An argument is invalid or out of range

9.270.2.6 int usermgmt::User_1_0_1::updateAccountFull (in string *password*, in UserInfo *info*)

Update user information.

Parameters

<i>password</i>	The new password; empty to leave unchanged
<i>info</i>	The new user information

Returns

- 0 OK
- 1 The new password has to differ from old password.
- 3 The password is too short.
- 4 The password is too long.
- 5 The password must not contain control characters.
- 6 The password has to contain at least one lower case character.
- 7 The password has to contain at least one upper case character.
- 8 The password has to contain at least one numeric character.
- 9 The password has to contain at least one printable special character.
- 10 The password already is in history.
- 11 SNMPv3 USM is activated for the user and the password shall be used as auth passphrase. For this case, the password is too short (must be at least 8 characters).
- 12 An argument is invalid or out of range

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/User.idl

9.271 usermgmt::UserCapabilities Struct Reference

User Capabilities Describe if certain operations can be performed for user.

```
import "User.idl";
```

Public Attributes

- boolean [canSetPassword](#)
User password is modifyable.
- boolean [canSetPreferences](#)
User preferences are modifyable.

9.271.1 Detailed Description

User Capabilities Describe if certain operations can be performed for user.

May require according privileges.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/User.idl

9.272 usermgmt::UserInfo Struct Reference

User information

```
import "User.idl";
```

Public Attributes

- boolean [enabled](#)
true if the account is enabled
- boolean [locked](#)
true if the account cannot be deleted
- boolean [blocked](#)
true if the account is blocked due to failed logins
- boolean [needPasswordChange](#)
true to force a password change on the next login
- [AuxInfo](#) [auxInfo](#)
Auxiliary user information.
- [SnmpV3Settings](#) [snmpV3Settings](#)
SNMPv3 settings.
- string [sshPublicKey](#)
Public key for SSH access.
- [Preferences](#) [preferences](#)
User preferences
- vector< int > [roleIds](#)
List of role ids for this account.

9.272.1 Detailed Description

User information

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/User.idl

9.273 usermgmt::UserManager_1_0_2 Interface Reference

User manager interface

```
import "UserManager.idl";
```

Public Member Functions

- `vector< string > getAccountNames ()`
Get a list of account names available on the system.
- `int createAccount (in string username, in string password)`
Create a new account.
- `int deleteAccount (in string username)`
Deletes an account.
- `vector< Account > getAllAccounts ()`
Get information about all available user accounts.
- `int createAccountFull (in string username, in string password, in UserInfo info)`
Create a new account with defined settings.
- `vector< Account > getAccountsByRole (in string roleName)`
Get a list of accounts that have a given role.
- `Preferences getDefaultPreferences ()`
Get default user preferences.
- `int setDefaultPreferences (in Preferences prefs)`
Set default user preferences.

Public Attributes

- `constant int ERR\_USER\_DOESNT\_EXIST = 1`
A user with the given name does not exist.
- `constant int ERR\_USER\_NOT\_DELETABLE = 2`
The user is not deletable.
- `constant int ERR\_USER\_ALREADY\_EXISTS = 1`
A user with the given name already exists.
- `constant int ERR\_MAX\_USERS\_REACHED = 2`
Maximum number of users reached.
- `constant int ERR\_PASSWORD\_TOO\_SHORT\_FOR\_SNMP = 3`
The password is too short to be used as SNMPv3 passphrase.
- `constant int ERR\_INVALID\_VALUE = 4`
Invalid arguments.
- `constant int ERR\_PASSWORD\_EMPTY = 5`
The password must not be empty.
- `constant int ERR\_PASSWORD\_TOO\_SHORT = 6`
The password is too short.
- `constant int ERR\_PASSWORD\_TOO\_LONG = 7`
The password is too long.
- `constant int ERR\_PASSWORD\_CTRL\_CHARS = 8`
The password must not contain control characters.
- `constant int ERR\_PASSWORD\_NEED\_LOWER = 9`
The password must contain at least one lower-case character.
- `constant int ERR\_PASSWORD\_NEED\_UPPER = 10`
The password must contain at least one upper-case character.
- `constant int ERR\_PASSWORD\_NEED\_NUMERIC = 11`
The password must contain at least one numeric character.
- `constant int ERR\_PASSWORD\_NEED\_SPECIAL = 12`
The password must contain at least one special character.

9.273.1 Detailed Description

User manager interface

9.273.2 Member Function Documentation

9.273.2.1 int usermgmt::UserManager_1_0_2::createAccount (in string *username*, in string *password*)

Create a new account.

Parameters

<i>username</i>	New user name
<i>password</i>	New password

Returns

- 0 if OK
- 1 if a user with the given name already exists
- 2 if the maximum number of users is reached
- 3 SNMPv3 USM is activated for the user and the password shall be used as auth passphrase. For this case, the password is too short (must be at least 8 characters).
- 4 if user name is invalid
- 5 The password must not be empty.
- 6 The password is too short.
- 7 The password is too long.
- 8 The password must not contain control characters.
- 9 The password has to contain at least one lower case character.
- 10 The password has to contain at least one upper case character.
- 11 The password has to contain at least one numeric character.
- 12 The password has to contain at least one printable special character.

9.273.2.2 int usermgmt::UserManager_1_0_2::createAccountFull (in string *username*, in string *password*, in UserInfo *info*)

Create a new account with defined settings.

Parameters

<i>username</i>	New user name
<i>password</i>	New password
<i>info</i>	New user information

Returns

- 0 if OK
- 1 if a user with the given name already exists
- 2 if the maximum number of users is reached
- 3 SNMPv3 USM is activated for the user and the password shall be used as auth passphrase. For this case, the password is too short (must be at least 8 characters).
- 4 if any value except password is invalid
- 5 The password must not be empty.
- 6 The password is too short.
- 7 The password is too long.
- 8 The password must not contain control characters.
- 9 The password has to contain at least one lower case character.
- 10 The password has to contain at least one upper case character.

- 11 The password has to contain at least one numeric character.
- 12 The password has to contain at least one printable special character.

9.273.2.3 `int usermgmt::UserManager_1_0_2::deleteAccount ( in string username )`

Deletes an account.

Parameters

<i>username</i>	Name of user to delete
-----------------	------------------------

Returns

- 0 if OK
- 1 if a user with the given name does not exist
- 2 if the user cannot be deleted

9.273.2.4 `vector<string> usermgmt::UserManager_1_0_2::getAccountNames ( )`

Get a list of account names available on the system.

Returns

List of account names

9.273.2.5 `vector<Account> usermgmt::UserManager_1_0_2::getAccountsByRole ( in string roleName )`

Get a list of accounts that have a given role.

Parameters

<i>roleName</i>	Role name
-----------------	---------------------------

Returns

List of accounts

9.273.2.6 `vector<Account> usermgmt::UserManager_1_0_2::getAllAccounts ( )`

Get information about all available user accounts.

Returns

List of accounts

9.273.2.7 `Preferences usermgmt::UserManager_1_0_2::getDefaultPreferences ( )`

Get default user preferences.

Returns

Default user preferences.

9.273.2.8 int usermgmt::UserManager_1_0_2::setDefaultPreferences (in Preferences *prefs*)

Set default user preferences.

Parameters

<i>prefs</i>	Default user preferences.
--------------	---------------------------

Returns

0 if OK

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/UserManager.idl

9.274 Ihxmodel::Parameter_2_0_1::Value Struct Reference

Parameter [Value](#).

```
import "LhxParameter.idl";
```

Public Attributes

- time [timestamp](#)
Timestamp of last sample.
- [Status](#) [status](#)
The state of the parameter.
- double [value](#)
The value of the parameter.

9.274.1 Detailed Description

Parameter [Value](#).

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/LhxParameter.idl

9.275 peripheral::PackageInfo_2_0_0::FirmwareInfo::Version Struct Reference

Public Attributes

- int **majorNumber**
- int **minorNumber**

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/PeripheralDevicePackage.idl

9.276 pdumodel::TransferSwitch_3_1_1::WaveformSample Struct Reference

Sample of voltage and current waveform.

```
import "TransferSwitch.idl";
```

Public Attributes

- double [voltage](#)
Voltage sample in Volts.
- double [current](#)
Current sample in Amperes.

9.276.1 Detailed Description

Sample of voltage and current waveform.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/TransferSwitch.idl

9.277 webcam::Webcam_2_0_0 Interface Reference

The webcam interface.

```
import "Webcam.idl";
```

Public Member Functions

- [Information_2_0_0](#) [getInformation](#) ()
Retrieve information of a specific webcam.
- [Settings_2_0_0](#) [getSettings](#) ()
Retrieve settings of a specific webcam.
- int [setSettings](#) (in [Settings_2_0_0](#) settings)
Change settings of a specific webcam.
- int [setControls](#) (in [Controls](#) controls)
Apply webcam control settings without storing them.
- [Controls](#) [getControlDefaults](#) ()
Retrieve the default value of the controls.

9.277.1 Detailed Description

The webcam interface.

9.277.2 Member Function Documentation

9.277.2.1 Controls [webcam::Webcam_2_0_0::getControlDefaults](#) ()

Retrieve the default value of the controls.

Returns

[Controls](#)

9.277.2.2 Information_2_0_0 webcam::Webcam_2_0_0::getInformation ()

Retrieve information of a specific webcam.

Returns

Information

9.277.2.3 Settings_2_0_0 webcam::Webcam_2_0_0::getSettings ()

Retrieve settings of a specific webcam.

Returns

Settings

9.277.2.4 int webcam::Webcam_2_0_0::setControls (in Controls *controls*)

Apply webcam control settings without storing them.

Parameters

<i>controls</i>	Control settings
-----------------	------------------

Returns

0 if success

9.277.2.5 int webcam::Webcam_2_0_0::setSettings (in Settings_2_0_0 *settings*)

Change settings of a specific webcam.

Parameters

<i>settings</i>	New webcam settings
-----------------	---------------------

Returns

0 if success

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Webcam.idl

9.278 webcam::WebcamManager_2_0_0 Interface Reference

The webcam manager interface.

```
import "WebcamManager.idl";
```

Public Member Functions

- vector< [Webcam_2_0_0](#) > [getWebcams](#) ()

- Retrieve all connected webcams.*
- int [getChannel](#) (in [Webcam_2_0_0](#) webcam, in string *clientType*, out [Channel](#) channel)
Returns an existing channel that is currently viewing the given webcam or creates one If the client type is unknown it's added to the list of client types.
- vector< [Channel](#) > [getChannels](#) ()
Returns all channels.
- int [removeClientType](#) (in string *clientType*)
Remove a client type.
- vector< string > [getClientTypes](#) ()
Get all known client types.
- map< string, [Priority](#) > [getClientTypePriorities](#) ()
Get the priority of all known client types.
- int [setClientTypePriorities](#) (in map< string, [Priority](#) > priorities)
Set the priority of a client type.
- map< string, [Priority](#) > [getWebcamPriorities](#) ()
Get the priority of a webcam.
- int [setWebcamPriorities](#) (in map< string, [Priority](#) > priorities)
Set the priority of a webcam.

Public Attributes

- constant int [NO_ERROR](#) = 0
Error codes.
- constant int [ERR_INVALID_PARAM](#) = 1
Invalid parameter for an operation.

9.278.1 Detailed Description

The webcam manager interface.

9.278.2 Member Function Documentation

9.278.2.1 int webcam::WebcamManager_2_0_0::getChannel (in Webcam_2_0_0 webcam, in string *clientType*, out [Channel](#) channel)

Returns an existing channel that is currently viewing the given webcam or creates one If the client type is unknown it's added to the list of client types.

Note: When a new channel is created the priority of the given webcam and channel type is summed up. The result is used to determine the used overall priority.

Parameters

<i>webcam</i>	webcam
<i>clientType</i>	the channel client type -> getClientTypes()
<i>channel</i>	Result: The channel

Returns

- NO_ERROR on success
- ERR_INVALID_PARAM one of the parameters is invalid

9.278.2.2 `vector<Channel> webcam::WebcamManager_2_0_0::getChannels ( )`

Returns all channels.

Note: you need to release every unneeded channel

Returns

List of all channels

9.278.2.3 `map<string, Priority> webcam::WebcamManager_2_0_0::getClientTypePriorities ( )`

Get the priority of all known client types.

Returns

a hash map with the key "event type ID" and the value "priority"

9.278.2.4 `vector<string> webcam::WebcamManager_2_0_0::getClientTypes ( )`

Get all known client types.

Returns

List of all client types

9.278.2.5 `map<string, Priority> webcam::WebcamManager_2_0_0::getWebcamPriorities ( )`

Get the priority of a webcam.

Returns

a hash map with the key "webcam ID" and the value "priority"

9.278.2.6 `vector<Webcam_2_0_0> webcam::WebcamManager_2_0_0::getWebcams ( )`

Retrieve all connected webcams.

Returns

List of webcams

9.278.2.7 `int webcam::WebcamManager_2_0_0::removeClientType ( in string clientType )`

Remove a client type.

Parameters

<i>clientType</i>	client type
-------------------	-------------

Returns

NO_ERROR on success
 ERR_INVALID_PARAM one of the parameters is invalid

9.278.2.8 int webcam::WebcamManager_2_0_0::setClientTypePriorities (in map< string, Priority > priorities)

Set the priority of a client type.

Parameters

<i>priorities</i>	a hash map with the key "event type ID" and the value "priority"
-------------------	--

Returns

NO_ERROR on success
 ERR_INVALID_PARAM one of the parameters is invalid

9.278.2.9 int webcam::WebcamManager_2_0_0::setWebcamPriorities (in map< string, Priority > priorities)

Set the priority of a webcam.

Parameters

<i>priorities</i>	a hash map with the key "webcam ID" and the value "priority"
-------------------	--

Returns

NO_ERROR on success
 ERR_INVALID_PARAM one of the parameters is invalid

9.278.3 Member Data Documentation**9.278.3.1 constant int webcam::WebcamManager_2_0_0::NO_ERROR = 0**

Error codes.

Operation successful, no error

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/WebcamManager.idl

9.279 webcam::StorageManager_1_0_1::WebcamStorageInfo Struct Reference

Webcam Storage Info.

```
import "StorageManager.idl";
```

Public Attributes

- [Webcam_2_0_0 webcam](#)
webcam object
- long [newestIndex](#)

- *newest know index*
- long [oldestIndex](#)
oldest know index
- int [count](#)
nr of stored images from this webcam

9.279.1 Detailed Description

Webcam Storage Info.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/StorageManager.idl

9.280 net::WirelessInterfaceSettings Struct Reference

Wireless interface settings.

```
import "Net.idl";
```

Public Attributes

- string [ssid](#)
SSID.
- [AuthenticationMode](#) [authentication](#)
Authentication mode.
- string [psk](#)
Pre-shared key (for PSK authentication)
- [EapSettings](#) [eap](#)
EAP settings (for EAP authentication)
- string [bssid](#)
BSSID, leave empty for automatic access point selection.

9.280.1 Detailed Description

Wireless interface settings.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/Net.idl

9.281 devsettings::Zeroconf Interface Reference

Zero-config service settings interface.

```
import "Zeroconf.idl";
```

Classes

- struct [Settings](#)
Zero-configservice settings.

Public Member Functions

- [Settings](#) `getSettings ()`
Retrieve the service settings.
- void [setSettings](#) (in [Settings](#) settings)
Set the service settings.

Public Attributes

- valueobject [SettingsChangedEvent](#): [idl.Event](#) { [Settings](#) oldSettings
old settings
- [Settings](#) `newSettings`
new settings

9.281.1 Detailed Description

Zero-config service settings interface.

9.281.2 Member Function Documentation

9.281.2.1 [Settings](#) `devsettings::Zeroconf::getSettings ( )`

Retrieve the service settings.

Returns

Zero-config service settings

9.281.2.2 void `devsettings::Zeroconf::setSettings ( in Settings settings )`

Set the service settings.

Parameters

<i>settings</i>	New settings
-----------------	--------------

The documentation for this interface was generated from the following file:

- pdu-json-rpc-api/idl/Zeroconf.idl

9.282 [datetime::DateTime_2_0_0::ZoneCfg](#) Struct Reference

Time zone configuration.

```
import "DateTime.idl";
```

Public Attributes

- int [id](#)
Selected time zone id.
- string [name](#)

Selected time zone name.

- boolean [enableAutoDST](#)

Enable automatic DST adjustment.

9.282.1 Detailed Description

Time zone configuration.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/DateTime.idl

9.283 datetime::DateTime_2_0_0::ZoneInfo Struct Reference

Time zone information.

```
import "DateTime.idl";
```

Public Attributes

- int [id](#)

Time zone id.

- string [name](#)

Time zone name.

- boolean [hasDSTInfo](#)

true if the time zone uses DST

9.283.1 Detailed Description

Time zone information.

The documentation for this struct was generated from the following file:

- pdu-json-rpc-api/idl/DateTime.idl

Index

ACCEPT
 security, [52](#)
ACTIVE_DIRECTORY
 auth::ldapsrv, [37](#)
AES128
 um::SnmpV3, [306](#)
ALLOW
 security, [53](#)
ALLOW_UNTRUSTED
 firmware, [41](#)
AMPERE
 lhxmodel::Parameter_2_0_1, [227](#)
ANALOGMODEM
 serial::SerialPort_2_0_0, [280](#)
AND
 event::Engine::Condition, [103](#)
ASCENDING
 webcam::StorageManager_1_0_1, [316](#)
ASSERTED
 event::Engine::Condition, [103](#)
ASSET_STRIP_STATE_CHANGED
 assetmgrmodel::AssetStripLogger_1_0_2, [82](#)
ASSET_TAG_CONNECTED
 assetmgrmodel::AssetStripLogger_1_0_2, [82](#)
ASSET_TAG_DISCONNECTED
 assetmgrmodel::AssetStripLogger_1_0_2, [82](#)
ATS
 pdumodel::TransferSwitch_3_1_1, [333](#)
AUTH_EAP
 net, [46](#)
AUTH_NO_PRIV
 um::SnmpV3, [307](#)
AUTH_NONE
 net, [46](#)
AUTH_PRIV
 um::SnmpV3, [307](#)
AUTH_PSK
 net, [46](#)
AUTO
 net, [46](#)
 portsmodel::Port_2_0_1, [239](#)
AVAILABLE
 assetmgrmodel::AssetStrip_2_0_1, [73](#)
AccountEvent
 usermgmt, [58](#)
accuracy
 sensors::NumericSensor_4_0_0::MetaData, [194](#)
acknowledgeAlarm
 event::AlarmManager, [67](#)
acknowledgeAlertStatus
 lhxmodel::Lhx_3_2_1, [182](#)
activate
 hmi::InternalBeeper, [174](#)
addAction
 event::Engine, [135](#)
addImage
 webcam::StorageManager_1_0_1, [316](#)
addRule
 event::Engine, [135](#)
addServer
 servermon::ServerMonitor_2_0_0, [283](#)
addTimerEvent
 event::TimerEventManager_2_0_0, [330](#)
alarm
 hmi::ExternalBeeper_1_0_1, [145](#)
AlarmAcknowledgedEvent
 event::AlarmManager, [67](#)
AlarmAddedEvent
 event::AlarmManager, [67](#)
AlarmUpdatedEvent
 event::AlarmManager, [67](#)
assetmgrmodel, [35](#)
assetmgrmodel::AssetStrip_2_0_1
 AVAILABLE, [73](#)
 CASCADE_ACTIVE, [73](#)
 CASCADE_FIRMWARE_UPDATE, [73](#)
 COMPOSITE, [74](#)
 DISCONNECTED, [73](#)
 EXTENSION, [74](#)
 FIRMWARE_UPDATE, [73](#)
 NONE, [74](#)
 SIMPLE, [74](#)
 SINGLE, [74](#)
 UNSUPPORTED, [73](#)
 UPDATE_FAILED, [73](#)
 UPDATE_STARTED, [73](#)
 UPDATE_SUCCESSFUL, [73](#)
assetmgrmodel::AssetStripConfig_1_0_1
 BOTTOM_CONNECTOR, [79](#)
 BOTTOM_UP, [79](#)
 LED_MODE_BLINK_FAST, [78](#)
 LED_MODE_BLINK_SLOW, [78](#)
 LED_MODE_OFF, [78](#)
 LED_MODE_ON, [78](#)
 LED_OPERATION_AUTO, [79](#)
 LED_OPERATION_MANUAL, [79](#)
 SCANMODE_BOTH, [79](#)
 SCANMODE_DISABLED, [79](#)

- TOP_CONNECTOR, [79](#)
- TOP_DOWN, [79](#)
- assetmgrmodel::AssetStripLogger_1_0_2
 - ASSET_STRIP_STATE_CHANGED, [82](#)
 - ASSET_TAG_CONNECTED, [82](#)
 - ASSET_TAG_DISCONNECTED, [82](#)
 - EMPTY, [82](#)
- assetmgrmodel::AssetStrip_2_0_1, [71](#)
 - BladeOverflowChangedEvent, [76](#)
 - CascadeState, [73](#)
 - CompositionChangedEvent, [76](#)
 - FirmwareUpdateState, [73](#)
 - FirmwareUpdateStateChangedEvent, [76](#)
 - getAllRackUnitInfos, [74](#)
 - getAllTags, [74](#)
 - getDeviceInfo, [74](#)
 - getExtensionTags, [74](#)
 - getMainTags, [75](#)
 - getRackUnitInfo, [75](#)
 - getState, [75](#)
 - getStripInfo, [75](#)
 - getTag, [75](#)
 - NO_ERROR, [76](#)
 - OrientationChangedEvent, [76](#)
 - RackUnitChangedEvent, [77](#)
 - State, [73](#)
 - StateChangedEvent, [77](#)
 - StripInfoChangedEvent, [77](#)
 - StripType, [73](#)
 - TagEvent, [77](#)
 - TagType, [74](#)
 - triggerPowercycle, [76](#)
- assetmgrmodel::AssetStrip_2_0_1::DeviceInfo, [117](#)
- assetmgrmodel::AssetStrip_2_0_1::RackUnitInfo, [246](#)
- assetmgrmodel::AssetStrip_2_0_1::StripInfo, [320](#)
- assetmgrmodel::AssetStrip_2_0_1::TagChangeInfo, [324](#)
- assetmgrmodel::AssetStrip_2_0_1::TagInfo, [324](#)
- assetmgrmodel::AssetStripConfig_1_0_1, [77](#)
 - getAllRackUnitSettings, [79](#)
 - getRackUnitSettings, [80](#)
 - getStripSettings, [80](#)
 - LEDMode, [78](#)
 - LEDOperationMode, [78](#)
 - NumberingMode, [79](#)
 - Orientation, [79](#)
 - RackUnitSettingsChangedEvent, [81](#)
 - ScanMode, [79](#)
 - setMultipleRackUnitSettings, [80](#)
 - setRackUnitSettings, [80](#)
 - setStripSettings, [81](#)
 - StripSettingsChangedEvent, [81](#)
- assetmgrmodel::AssetStripConfig_1_0_1::LEDColor, [180](#)
- assetmgrmodel::AssetStripConfig_1_0_1::RackUnitSettings, [246](#)
- assetmgrmodel::AssetStripConfig_1_0_1::StripSettings, [321](#)
- assetmgrmodel::AssetStripLogger_1_0_2, [81](#)
 - getInfo, [82](#)
 - getRecords, [82](#)
 - NO_ERROR, [83](#)
 - RecordType, [82](#)
- assetmgrmodel::AssetStripLogger_1_0_2::Info, [166](#)
- assetmgrmodel::AssetStripLogger_1_0_2::Record, [252](#)
- assign
 - peripheral::DeviceSlot_2_0_0, [123](#)
- assignAddress
 - peripheral::DeviceSlot_2_0_0, [123](#)
- auth, [35](#)
 - KERBEROS, [36](#)
 - LDAP, [36](#)
 - LOCAL, [36](#)
 - RADIUS, [36](#)
 - TACACS_PLUS, [36](#)
 - Type, [36](#)
- auth::ldapsrv
 - ACTIVE_DIRECTORY, [37](#)
 - OPEN_LDAP, [37](#)
- auth::AuthManager, [83](#)
 - getPolicy, [83](#)
 - setPolicy, [83](#)
- auth::LdapManager_1_0_1, [178](#)
 - getLdapServers, [179](#)
 - setLdapServers, [179](#)
 - testLdapServer, [179](#)
- auth::Policy, [238](#)
- auth::RadiusManager, [247](#)
 - getRadiusServers, [248](#)
 - setRadiusServers, [248](#)
 - testRadiusServer, [248](#)
- auth::ldapsrv, [36](#)
 - ServerType, [37](#)
- auth::ldapsrv::ServerSettings, [284](#)
- AuthProtocol
 - um::SnmpV3, [306](#)
- AuthType
 - radius, [51](#)
- AuthenticationMode
 - net, [46](#)
- AutoConfigs
 - net, [46](#)
- BACKWARD
 - logging, [44](#)
- BAR
 - lhxmodel::Parameter_2_0_1, [227](#)
- BINARY
 - lhxmodel::Parameter_2_0_1, [226](#)
- BOTH
 - event::Engine::Condition, [103](#)
- BOTTOM_CONNECTOR
 - assetmgrmodel::AssetStripConfig_1_0_1, [79](#)
- BOTTOM_UP
 - assetmgrmodel::AssetStripConfig_1_0_1, [79](#)
- BR115200
 - serial::SerialPort_2_0_0, [279](#)

- BR1200
 - serial::SerialPort_2_0_0, [279](#)
- BR19200
 - serial::SerialPort_2_0_0, [279](#)
- BR2400
 - serial::SerialPort_2_0_0, [279](#)
- BR38400
 - serial::SerialPort_2_0_0, [279](#)
- BR4800
 - serial::SerialPort_2_0_0, [279](#)
- BR57600
 - serial::SerialPort_2_0_0, [279](#)
- BR9600
 - serial::SerialPort_2_0_0, [279](#)
- BREAKER_1POLE
 - pduModel::OverCurrentProtector_2_1_2, [222](#)
- BREAKER_2POLE
 - pduModel::OverCurrentProtector_2_1_2, [222](#)
- BREAKER_3POLE
 - pduModel::OverCurrentProtector_2_1_2, [222](#)
- BROWNOUT
 - peripheral::G2Production_2_0_0, [155](#)
- BaudRate
 - serial::SerialPort_2_0_0, [279](#)
- BladeOverflowChangedEvent
 - assetmgrmodel::AssetStrip_2_0_1, [76](#)
- bulkcfg, [37](#)
 - bulkcfg::BulkConfiguration
 - RESTORE_FAILED, [88](#)
 - RESTORE_OK, [88](#)
 - RESTORE_PENDING, [88](#)
 - UNKNOWN, [88](#)
 - UPLOAD_FAILED, [88](#)
 - bulkcfg::BulkConfiguration, [87](#)
 - getStatus, [88](#)
 - Status, [88](#)
- CASCADE_ACTIVE
 - assetmgrmodel::AssetStrip_2_0_1, [73](#)
- CASCADE_FIRMWARE_UPDATE
 - assetmgrmodel::AssetStrip_2_0_1, [73](#)
- CHAP
 - radius, [51](#)
- CIFS
 - webcam::StorageManager_1_0_1, [316](#)
- CLOSE_REASON_BROWSER_CLOSED
 - session::SessionManager, [292](#)
- CLOSE_REASON_FORCED_DISCONNECT
 - session::SessionManager, [292](#)
- CLOSE_REASON_LOGOUT
 - session::SessionManager, [292](#)
- CLOSE_REASON_TIMEOUT
 - session::SessionManager, [292](#)
- COMMUNICATION_FAILURE
 - pduModel::Controller_3_0_0, [110](#)
- COMMUNICATION_UNSTABLE
 - pduModel::Controller_3_0_0, [110](#)
- COMPLETE
 - firmware, [41](#)
- COMPOSITE
 - assetmgrmodel::AssetStrip_2_0_1, [74](#)
- CONFIG_ERROR
 - peripheral::PackageInfo_2_0_0, [225](#)
- CONSOLE
 - serial::SerialPort_2_0_0, [280](#)
- CROSS_HW
 - firmware, [41](#)
- CROSS_OEM
 - firmware, [41](#)
- CallEndedEvent
 - serial::AnalogModem, [70](#)
- cancelEvent
 - event::Channel_1_0_1, [98](#)
- cancelEventType
 - event::Channel_1_0_1, [98](#)
- cancelEventTypes
 - event::Channel_1_0_1, [98](#)
- cancelEvents
 - event::Channel_1_0_1, [98](#)
- captureImage
 - webcam::Channel, [96](#)
- CardEvent
 - smartcard::CardReader, [90](#)
- CardReaderEvent
 - smartcard::CardReaderManager, [91](#)
- CascadeState
 - assetmgrmodel::AssetStrip_2_0_1, [73](#)
- cascading, [37](#)
 - cascading::Cascading_1_0_1
 - USB_MULTI_IP, [92](#)
 - USB_SINGLE_IP_NAT, [92](#)
 - cascading::Cascading_1_0_1, [91](#)
 - getIndex, [93](#)
 - getMasterIpAddress, [93](#)
 - getMasterIpV6Address, [93](#)
 - getProtocolMappings, [93](#)
 - getType, [93](#)
 - setType, [94](#)
 - Type, [92](#)
 - cascading::Cascading_1_0_1::ProtocolMapping, [245](#)
- cert, [37](#)
 - cert::ServerSSLCert, [286](#)
 - generateSelfSignedKeyPair, [287](#)
 - generateUnsignedKeyPair, [287](#)
 - getInfo, [287](#)
 - installPendingKeyPair, [287](#)
 - cert::ServerSSLCert::CertInfo, [94](#)
 - cert::ServerSSLCert::CommonAttributes, [101](#)
 - cert::ServerSSLCert::Info, [167](#)
 - maxSignDays, [168](#)
 - cert::ServerSSLCert::ReqInfo, [253](#)
- cew, [38](#)
 - cew::EnergyWiseManager, [133](#)
 - getSettings, [133](#)
 - setSettings, [133](#)
 - cew::EnergyWiseSettings, [133](#)
- checkNtpServer

- datetime::DateTime_2_0_0, 114
- closeCurrentSession
 - session::SessionManager, 293
- CloseReason
 - session::SessionManager, 292
- closeSession
 - session::SessionManager, 293
- ComSettingsChangedEvent
 - lhxmodel::Config_1_0_1, 105
- CompositionChangedEvent
 - assetmgrmodel::AssetStrip_2_0_1, 76
- ConfigurationSpace
 - peripheral::G2Production_2_0_0, 155
- createAccount
 - usermgmt::UserManager_1_0_2, 347
- createAccountFull
 - usermgmt::UserManager_1_0_2, 347
- createChannel
 - event::Service_1_0_1, 289
- createRoleFull
 - usermgmt::RoleManager, 259
- cycleAllOutletPowerStates
 - pdumodel::Pdu_3_0_0, 231
- cycleInProgress
 - pdumodel::Outlet_1_5_6::State, 309
- cycleMultipleOutletPowerStates
 - pdumodel::Pdu_3_0_0, 231
- cyclePowerState
 - pdumodel::Outlet_1_5_6, 218
- DEASSERTED
 - event::Engine::Condition, 103
- DEG_C
 - usermgmt, 58
- DEG_F
 - usermgmt, 58
- DENY
 - security, 53
- DES
 - um::SnmpV3, 306
- DESCENDING
 - webcam::StorageManager_1_0_1, 316
- DHCP
 - net, 46
- DISABLED
 - portsmodel::Port_2_0_1, 239
- DISCONNECTED
 - assetmgrmodel::AssetStrip_2_0_1, 73
- DOWNLOAD_FAILED
 - firmware, 41
- DOWNLOADING
 - firmware, 41
- DPX_FULL
 - test::Ethernet, 140
- DPX_HALF
 - test::Ethernet, 140
- DROP
 - security, 52
- DYN_NODE
 - event::Engine::EventDesc, 142
- datetime, 38
- datetime::DateTime_2_0_0
 - NTP, 113
 - STATIC, 113
- datetime::DateTime_2_0_0, 113
 - checkNtpServer, 114
 - getCfg, 114
 - getTime, 114
 - getZoneInfos, 114
 - Protocol, 113
 - setCfg, 114
- datetime::DateTime_2_0_0::Cfg, 95
- datetime::DateTime_2_0_0::NtpCfg, 211
- datetime::DateTime_2_0_0::ZoneCfg, 356
- datetime::DateTime_2_0_0::ZoneInfo, 357
- decdigits
 - sensors::NumericSensor_4_0_0::MetaData, 194
- deleteAccount
 - usermgmt::UserManager_1_0_2, 348
- deleteAction
 - event::Engine, 135
- deleteRole
 - usermgmt::RoleManager, 259
- deleteRule
 - event::Engine, 136
- deleteServer
 - servermon::ServerMonitor_2_0_0, 283
- deleteTimerEvent
 - event::TimerEventManager_2_0_0, 330
- deliverEvent
 - event::Engine, 136
- demandEvent
 - event::Channel_1_0_1, 99
- demandEventType
 - event::Channel_1_0_1, 99
- demandEventTypes
 - event::Channel_1_0_1, 99
- demandEvents
 - event::Channel_1_0_1, 99
- destroyChannel
 - event::Service_1_0_1, 289
- DetectionType
 - portsmodel::Port_2_0_1, 239
- Device_2_0_0
 - peripheral, 50
- DeviceChangedEvent
 - peripheral::DeviceSlot_2_0_0, 124
 - portsmodel::Port_2_0_1, 240
- DeviceEvent
 - peripheral::DeviceManager_2_0_0, 121
- DeviceFirmwareUpdateState
 - peripheral::DeviceManager_2_0_0, 119
- DeviceFirmwareUpdateStateChangedEvent
 - peripheral::DeviceManager_2_0_0, 121
- devsettings, 38
- devsettings::Modbus, 202
 - getSettings, 202

- setSettings, 202
- devsettings::Modbus::Settings, 298
- devsettings::Modbus::TcpSettings, 325
- devsettings::Sntp_1_0_1, 303
 - getConfiguration, 304
 - setConfiguration, 304
 - testConfiguration, 304
- devsettings::Sntp_1_0_1::Configuration, 106
- devsettings::Sntp_1_0_1::TestResult, 325
- devsettings::Snmp_1_0_2, 304
 - getConfiguration, 305
 - getV3EngineId, 305
 - setConfiguration, 305
- devsettings::Snmp_1_0_2::Configuration, 107
- devsettings::Zeroconf, 355
 - getSettings, 356
 - setSettings, 356
- devsettings::Zeroconf::Settings, 295
- diag, 39
- diag::DiagLogSettings
 - LOG_LEVEL_DEBUG, 126
 - LOG_LEVEL_ERR, 126
 - LOG_LEVEL_INFO, 126
 - LOG_LEVEL_NONE, 126
 - LOG_LEVEL_TRACE, 126
 - LOG_LEVEL_WARN, 126
- diag::DiagLogSettings, 125
 - getLogLevelByCtxName, 126
 - getLogLevelsForAllCtxNames, 126
 - LogLevel, 126
 - setLogLevelByCtxName, 126
 - setLogLevelForAllCtxNames, 127
- diag::DiagLogSettings::LogLevelEntry, 191
- DialInEvent
 - serial::AnalogModem, 70
- Direction
 - webcam::StorageManager_1_0_1, 315
- disableRule
 - event::Engine, 136
- downloadImage
 - firmware::Firmware_1_0_1, 148
- Duplex
 - test::Ethernet, 140
- EAP_MSCHAPv2
 - net, 46
- EAP_PEAP
 - net, 46
- EMPTY
 - assetmgrmodel::AssetStripLogger_1_0_2, 82
- ERROR
 - servermon::ServerMonitor_2_0_0, 282
- EXTENSION
 - assetmgrmodel::AssetStrip_2_0_1, 74
- EapInnerMethod
 - net, 46
- EapOuterMethod
 - net, 46
- enableOnlineCheck
 - firmware::Firmware_1_0_1, 148
- enableRule
 - event::Engine, 136
- EnableStateChangedEvent
 - pdumodel::Inlet_1_2_6, 172
- enterFactoryConfigMode
 - production::Production, 244
- enterRS485ConfigModeAndAssignCtrlBoardAddress
 - pdumodel::Pdu_3_0_0, 232
- enterRS485ConfigModeAndAssignSCBoardAddress
 - pdumodel::Pdu_3_0_0, 232
- eraseConfigurationSpace
 - peripheral::G2Production_2_0_0, 155
- Event
 - idl, 42
- event, 39
 - UserEvent, 40
- event::Engine::Condition
 - AND, 103
 - ASSERTED, 103
 - BOTH, 103
 - DEASSERTED, 103
 - OR, 103
 - XOR, 103
- event::Engine::EventDesc
 - DYN_NODE, 142
 - LEAF, 142
 - NODE, 142
- event::Event
 - STATE, 141
 - TRIGGER, 141
- event::AlarmManager, 66
 - acknowledgeAlarm, 67
 - AlarmAcknowledgedEvent, 67
 - AlarmAddedEvent, 67
 - AlarmUpdatedEvent, 67
 - NO_ERROR, 67
- event::AlarmManager::Alarm, 65
- event::AlarmManager::Alert, 67
- event::Channel_1_0_1, 97
 - cancelEvent, 98
 - cancelEventType, 98
 - cancelEventTypes, 98
 - cancelEvents, 98
 - demandEvent, 99
 - demandEventType, 99
 - demandEventTypes, 99
 - demandEvents, 99
 - pollEvents, 99
 - pollEventsNb, 99
 - subscribe, 100
 - unsubscribe, 100
- event::Channel_1_0_1::EventSelect, 144
- event::Consumer, 107
 - pushEvents, 108
- event::Engine, 134
 - addAction, 135
 - addRule, 135

- deleteAction, [135](#)
- deleteRule, [136](#)
- deliverEvent, [136](#)
- disableRule, [136](#)
- enableRule, [136](#)
- listActionTypes, [136](#)
- listEventDescs, [136](#)
- listRules, [137](#)
- modifyAction, [137](#)
- modifyRule, [137](#)
- rearmRule, [137](#)
- triggerAction, [137](#)
- event::Engine::Action, [62](#)
- event::Engine::Condition, [102](#)
 - MatchType, [103](#)
 - Op, [103](#)
- event::Engine::EventDesc, [142](#)
 - Type, [142](#)
- event::Engine::Rule, [261](#)
- event::Event, [141](#)
 - Type, [141](#)
- event::KeyValue, [177](#)
- event::Service_1_0_1, [288](#)
 - createChannel, [289](#)
 - destroyChannel, [289](#)
- event::TimerEventManager_2_0_0, [328](#)
 - addTimerEvent, [330](#)
 - deleteTimerEvent, [330](#)
 - JAN, [331](#)
 - modifyTimerEvent, [330](#)
 - NO_ERROR, [331](#)
 - SUN, [331](#)
- event::TimerEventManager_2_0_0::Range, [248](#)
- event::TimerEventManager_2_0_0::Schedule, [262](#)
- event::TimerEventManager_2_0_0::TimerEvent, [328](#)
- FAILED
 - firmware, [41](#)
- FEET
 - usermgmt, [57](#)
- FIRMWARE
 - peripheral::G2Production_2_0_0, [155](#)
- FIRMWARE_UPDATE
 - assetmgrmodel::AssetStrip_2_0_1, [73](#)
- FLIPPED
 - pdumodel::Unit_2_0_1, [338](#)
 - test::Display_1_0_1, [129](#)
- FORWARD
 - logging, [44](#)
- FREEFORM
 - peripheral::DeviceManager_2_0_0, [119](#)
- FS_FREE_INODES
 - res_mon::Entry, [139](#)
- FS_FREE_SPACE
 - res_mon::Entry, [139](#)
- FTP
 - webcam::StorageManager_1_0_1, [316](#)
- FUNCTION
 - peripheral::G2Production_2_0_0, [155](#)
- FUSE
 - pdumodel::OverCurrentProtector_2_1_2, [222](#)
- FUSE_PAIR
 - pdumodel::OverCurrentProtector_2_1_2, [222](#)
- FW_UPDATE
 - peripheral::PackageInfo_2_0_0, [225](#)
- FLAG_ENTRY_CRITICAL
 - fitness::Fitness, [153](#)
- FLAG_VALUE_INVALID
 - fitness::Fitness, [153](#)
- FLAG_VALUE_OLD
 - fitness::Fitness, [153](#)
- firmware, [40](#)
 - ALLOW_UNTRUSTED, [41](#)
 - COMPLETE, [41](#)
 - CROSS_HW, [41](#)
 - CROSS_OEM, [41](#)
 - DOWNLOAD_FAILED, [41](#)
 - DOWNLOADING, [41](#)
 - FAILED, [41](#)
 - INCOMPLETE, [41](#)
 - ImageState, [41](#)
 - NONE, [41](#)
 - SUCCESSFUL, [41](#)
 - UPLOAD_FAILED, [41](#)
 - UPLOADING, [41](#)
 - UpdateFlags, [41](#)
 - UpdateHistoryStatus, [41](#)
- firmware::Firmware_1_0_1, [147](#)
 - downloadImage, [148](#)
 - enableOnlineCheck, [148](#)
 - getImageInfo, [148](#)
 - getImageStatus, [148](#)
 - getUpdateHistory, [149](#)
 - getVersion, [149](#)
 - onlineCheckEnabled, [149](#)
 - performOnlineCheck, [149](#)
 - reboot, [149](#)
 - startUpdate, [149](#)
 - updateAvailable, [149](#)
- firmware::FirmwareUpdateStatus, [151](#)
 - getStatus, [151](#)
- firmware::ImageInfo_1_0_1, [164](#)
- firmware::ImageStatus, [165](#)
- firmware::UpdateHistoryEntry, [339](#)
- firmware::UpdateStatus, [340](#)
- FirmwareUpdateState
 - assetmgrmodel::AssetStrip_2_0_1, [73](#)
- FirmwareUpdateStateChangedEvent
 - assetmgrmodel::AssetStrip_2_0_1, [76](#)
- fitness, [41](#)
 - fitness::Fitness, [151](#)
 - FLAG_VALUE_INVALID, [153](#)
 - FLAG_VALUE_OLD, [153](#)
 - getDataEntries, [152](#)
 - getErrorLogEntries, [152](#)
 - getErrorLogIndexRange, [153](#)
 - fitness::Fitness::DataEntry, [112](#)

fitness::Fitness::ErrorLogEntry, 139
 GLOBAL_CPU_USAGE
 res_mon::Entry, 139
 GLOBAL_FREE_MEM
 res_mon::Entry, 139
 GLOBAL_PROC_COUNT
 res_mon::Entry, 139
 GRAMMS
 lhmodel::Parameter_2_0_1, 227
 GSMMODEM
 serial::SerialPort_2_0_0, 280
 generateSelfSignedKeyPair
 cert::ServerSSLCert, 287
 generateUnsignedKeyPair
 cert::ServerSSLCert, 287
 getAccountNames
 usermgmt::UserManager_1_0_2, 348
 getAccountsByRole
 usermgmt::UserManager_1_0_2, 348
 getActivities
 webcam::StorageManager_1_0_1, 316
 getAllAccounts
 usermgmt::UserManager_1_0_2, 348
 getAllPrivileges
 usermgmt::RoleManager, 260
 getAllRackUnitInfos
 assetmgrmodel::AssetStrip_2_0_1, 74
 getAllRackUnitSettings
 assetmgrmodel::AssetStripConfig_1_0_1, 79
 getAllRoleNames
 usermgmt::RoleManager, 260
 getAllRoles
 usermgmt::RoleManager, 260
 getAllTags
 assetmgrmodel::AssetStrip_2_0_1, 74
 getBeeper
 pdumodel::Pdu_3_0_0, 232
 getBlockSettings
 security::Security_3_0_0, 264
 getBridgeSlaveCount
 net::Net_2_0_2, 206
 getButtonStates
 test::Unit_1_0_2, 336
 getCalibrationData
 pdumodel::Ade, 64
 getCapabilities
 usermgmt::User_1_0_1, 343
 getCardInformation
 smartcard::CardReader, 90
 getCardReaderById
 smartcard::CardReaderManager, 91
 getCardReaders
 smartcard::CardReaderManager, 91
 getCfg
 datetime::DateTime_2_0_0, 114
 getChannel
 webcam::WebcamManager_2_0_0, 352
 getChannelConfigs
 pdumodel::Bcm, 86
 getChannelCount
 pdumodel::Bcm, 86
 getChannels
 webcam::WebcamManager_2_0_0, 352
 getChildren
 pdumodel::EDevice, 132
 getClientType
 webcam::Channel, 96
 getClientTypePriorities
 webcam::WebcamManager_2_0_0, 353
 getClientTypes
 webcam::WebcamManager_2_0_0, 353
 getComSettings
 lhmodel::Config_1_0_1, 104
 getCommunicationStatus
 pdumodel::Controller_3_0_0, 110
 getConfiguration
 devsettings::Sntp_1_0_1, 304
 devsettings::Snmp_1_0_2, 305
 modelpush::ModelPush, 203
 getControlDefaults
 webcam::Webcam_2_0_0, 350
 getController
 pdumodel::Outlet_1_5_6, 218
 getControllers
 pdumodel::Pdu_3_0_0, 232
 getCurrentSession
 session::SessionManager, 293
 getDataEntries
 fitness::Fitness, 152
 res_mon::ResMon, 255
 getDefaultPreferences
 usermgmt::UserManager_1_0_2, 348
 getDefaultThresholds
 sensors::NumericSensor_4_0_0, 213
 getDetectableDevices
 portsmodel::Port_2_0_1, 239
 getDevice
 peripheral::DeviceSlot_2_0_0, 123
 portsmodel::Port_2_0_1, 239
 getDeviceConfig
 portsmodel::Port_2_0_1, 239
 getDeviceInfo
 assetmgrmodel::AssetStrip_2_0_1, 74
 getDeviceSlot
 peripheral::DeviceManager_2_0_0, 120
 getDeviceSlots
 peripheral::DeviceManager_2_0_0, 120
 getDeviceTypeInfoInfos
 peripheral::DeviceManager_2_0_0, 120
 getDevices
 usb::Usb, 341
 getDiscoveredDevices
 peripheral::DeviceManager_2_0_0, 120
 getDiscoveredPackageInfos
 peripheral::DeviceManager_2_0_0, 120
 getDisplayOrientation

pdumodel::Unit_2_0_1, 338
 getDisplays
 test::Unit_1_0_2, 336
 getEntries
 logging::DebugLog, 115
 logging::EventLog_1_0_1, 143
 getErrorLogEntries
 fitness::Fitness, 152
 getErrorLogIndexRange
 fitness::Fitness, 153
 getExtensionTags
 assetmgrmodel::AssetStrip_2_0_1, 74
 getFeaturePorts
 pdumodel::Pdu_3_0_0, 233
 getFilteredEntries
 logging::EventLog_1_0_1, 143
 getFirmwareInfo
 peripheral::G2Production_2_0_0, 155
 getFirstId
 logging::DebugLog, 115
 logging::EventLog_1_0_1, 143
 getFrontPanelPrivileges
 security::Security_3_0_0, 264
 getHttpRedirSettings
 security::Security_3_0_0, 264
 getIOP
 pdumodel::Outlet_1_5_6, 218
 getIdleTimeoutSettings
 security::Security_3_0_0, 265
 getImageInfo
 firmware::Firmware_1_0_1, 148
 getImageStatus
 firmware::Firmware_1_0_1, 148
 getImages
 webcam::StorageManager_1_0_1, 316
 getIndex
 cascading::Cascading_1_0_1, 93
 getInfo
 assetmgrmodel::AssetStripLogger_1_0_2, 82
 cert::ServerSSLCert, 287
 test::Display_1_0_1, 130
 usermgmt::Role, 256
 usermgmt::RoleManager, 260
 usermgmt::User_1_0_1, 343
 getInfoAndPrivileges
 usermgmt::User_1_0_1, 343
 getInformation
 serial::GsmModem_1_0_1, 160
 webcam::StorageManager_1_0_1, 317
 webcam::Webcam_2_0_0, 350
 getInformationWithPin
 serial::GsmModem_1_0_1, 160
 getInlet
 pdumodel::OverCurrentProtector_2_1_2, 223
 getInlets
 pdumodel::Pdu_3_0_0, 233
 getLastId
 logging::DebugLog, 115
 logging::EventLog_1_0_1, 144
 getLastTransferReason
 pdumodel::TransferSwitch_3_1_1, 334
 getLastTransferWaveform
 pdumodel::TransferSwitch_3_1_1, 334
 getLatestSample
 pdumodel::Ade, 64
 getLdapServers
 auth::LdapManager_1_0_1, 179
 getLogLevelByCtxName
 diag::DiagLogSettings, 126
 getLogLevelsForAllCtxNames
 diag::DiagLogSettings, 126
 getLogRow
 sensors::Logger_2_1_2, 188
 getLoggedSensors
 sensors::Logger_2_1_2, 188
 getMACs
 net::Net_2_0_2, 206
 getMainTags
 assetmgrmodel::AssetStrip_2_0_1, 75
 getMasterIpAddress
 cascading::Cascading_1_0_1, 93
 getMasterIpV6Address
 cascading::Cascading_1_0_1, 93
 getMetaData
 lhxmodel::Lhx_3_2_1, 182
 lhxmodel::Parameter_2_0_1, 227
 lhxmodel::Sensor_4_0_0, 270
 pdumodel::Ade, 65
 pdumodel::Controller_3_0_0, 110
 pdumodel::Inlet_1_2_6, 171
 pdumodel::Outlet_1_5_6, 218
 pdumodel::OverCurrentProtector_2_1_2, 223
 pdumodel::Pdu_3_0_0, 233
 pdumodel::TransferSwitch_3_1_1, 334
 pdumodel::Unit_2_0_1, 338
 peripheral::DeviceManager_2_0_0, 120
 sensors::NumericSensor_4_0_0, 213
 smartcard::CardReader, 90
 webcam::StorageManager_1_0_1, 317
 getModem
 serial::SerialPort_2_0_0, 280
 getName
 lhxmodel::Config_1_0_1, 104
 getNameplate
 pdumodel::Pdu_3_0_0, 233
 getNetworkConfigIP
 net::Net_2_0_2, 206
 getNetworkConfigIPv4
 net::Net_2_0_2, 207
 getNetworkConfigIPv6
 net::Net_2_0_2, 207
 getNetworkConfigInterface
 net::Net_2_0_2, 206
 getNetworkConfigServices
 net::Net_2_0_2, 207
 getNumberOfPorts

- test::AuxSerial, [85](#)
- test::FeatSerial, [146](#)
- getOCP
 - pdumodel::OverCurrentProtector_2_1_2, [223](#)
- getOutletSequenceState
 - pdumodel::Pdu_3_0_0, [233](#)
- getOutlets
 - pdumodel::Pdu_3_0_0, [233](#)
- getOverCurrentProtectors
 - pdumodel::Pdu_3_0_0, [233](#)
- getParents
 - pdumodel::EDevice, [132](#)
- getPeripheralDeviceManager
 - pdumodel::Pdu_3_0_0, [234](#)
- getPeripheralDeviceRecords
 - sensors::Logger_2_1_2, [188](#)
- getPeripheralDeviceTimedRecords
 - sensors::Logger_2_1_2, [188](#)
- getPoles
 - pdumodel::Inlet_1_2_6, [171](#)
 - pdumodel::OverCurrentProtector_2_1_2, [223](#)
 - pdumodel::TransferSwitch_3_1_1, [334](#)
- getPolicy
 - auth::AuthManager, [83](#)
- getPorts
 - serial::PortDispatcher_1_1_1, [241](#)
- getProperties
 - portsmodel::Port_2_0_1, [240](#)
- getProtocolMappings
 - cascading::Cascading_1_0_1, [93](#)
- getRackUnitInfo
 - assetmgrmodel::AssetStrip_2_0_1, [75](#)
- getRackUnitSettings
 - assetmgrmodel::AssetStripConfig_1_0_1, [80](#)
- getRadiusServers
 - auth::RadiusManager, [248](#)
- getRawValue
 - lhxmodel::Parameter_2_0_1, [227](#)
- getReading
 - lhxmodel::Sensor_4_0_0, [270](#)
 - sensors::NumericSensor_4_0_0, [213](#)
- getRecords
 - assetmgrmodel::AssetStripLogger_1_0_2, [82](#)
- getRestrictedServiceAgreement
 - security::Security_3_0_0, [265](#)
- getSSHSettings
 - security::Security_3_0_0, [265](#)
- getSensorLogger
 - pdumodel::Pdu_3_0_0, [234](#)
- getSensorRecords
 - sensors::Logger_2_1_2, [189](#)
- getSensorSetTimestamp
 - sensors::Logger_2_1_2, [189](#)
- getSensorTimedRecords
 - sensors::Logger_2_1_2, [189](#)
- getSensors
 - pdumodel::Inlet_1_2_6, [171](#)
 - pdumodel::Outlet_1_5_6, [218](#)
 - pdumodel::OverCurrentProtector_2_1_2, [223](#)
 - pdumodel::Pdu_3_0_0, [234](#)
 - pdumodel::TransferSwitch_3_1_1, [334](#)
- getServer
 - servermon::ServerMonitor_2_0_0, [283](#)
- getSession
 - session::SessionManager, [293](#)
- getSessionHistory
 - session::SessionManager, [293](#)
- getSessions
 - session::SessionManager, [293](#)
- getSettings
 - cew::EnergyWiseManager, [133](#)
 - devsettings::Modbus, [202](#)
 - devsettings::Zeroconf, [356](#)
 - lhxmodel::Lhx_3_2_1, [182](#)
 - pdumodel::Inlet_1_2_6, [171](#)
 - pdumodel::Outlet_1_5_6, [219](#)
 - pdumodel::OverCurrentProtector_2_1_2, [223](#)
 - pdumodel::Pdu_3_0_0, [234](#)
 - pdumodel::TransferSwitch_3_1_1, [334](#)
 - pdumodel::Unit_2_0_1, [338](#)
 - peripheral::DeviceManager_2_0_0, [120](#)
 - peripheral::DeviceSlot_2_0_0, [123](#)
 - security::Security_3_0_0, [265](#)
 - sensors::Logger_2_1_2, [190](#)
 - serial::AnalogModem, [70](#)
 - serial::GsmModem_1_0_1, [160](#)
 - serial::SerialPort_2_0_0, [280](#)
 - webcam::StorageManager_1_0_1, [317](#)
 - webcam::Webcam_2_0_0, [351](#)
- getSimSecurityStatus
 - serial::GsmModem_1_0_1, [160](#)
- getState
 - assetmgrmodel::AssetStrip_2_0_1, [75](#)
 - hmi::InternalBeeper, [174](#)
 - pdumodel::Outlet_1_5_6, [219](#)
 - sensors::StateSensor_4_0_0, [310](#)
 - serial::SerialPort_2_0_0, [280](#)
- getStatistic
 - pdumodel::Pdu_3_0_0, [234](#)
- getStatistics
 - pdumodel::Controller_3_0_0, [110](#)
 - pdumodel::TransferSwitch_3_1_1, [334](#)
 - peripheral::DeviceManager_2_0_0, [121](#)
- getStatus
 - bulkcfg::BulkConfiguration, [88](#)
 - firmware::FirmwareUpdateStatus, [151](#)
- getStripInfo
 - assetmgrmodel::AssetStrip_2_0_1, [75](#)
- getStripSettings
 - assetmgrmodel::AssetStripConfig_1_0_1, [80](#)
- getSupportedFrontPanelPrivileges
 - security::Security_3_0_0, [265](#)
- getSupportedStorageTypes
 - webcam::StorageManager_1_0_1, [317](#)
- getTag
 - assetmgrmodel::AssetStrip_2_0_1, [75](#)

- getTestStatus
 - test::Display_1_0_1, [130](#)
- getThresholds
 - lhxmodel::Sensor_4_0_0, [270](#)
 - sensors::NumericSensor_4_0_0, [213](#)
- getTime
 - datetime::DateTime_2_0_0, [114](#)
- getTimeStamps
 - sensors::Logger_2_1_2, [190](#)
- getTransferSwitches
 - pduModel::Pdu_3_0_0, [234](#)
- getType
 - cascading::Cascading_1_0_1, [93](#)
- getTypeSpec
 - sensors::Sensor_4_0_0, [274](#)
- getUpdateHistory
 - firmware::Firmware_1_0_1, [149](#)
- getV3EngineId
 - devsettings::Snmp_1_0_2, [305](#)
- getValue
 - lhxmodel::Parameter_2_0_1, [227](#)
- getVersion
 - firmware::Firmware_1_0_1, [149](#)
- getWebcam
 - webcam::Channel, [96](#)
- getWebcamPriorities
 - webcam::WebcamManager_2_0_0, [353](#)
- getWebcams
 - webcam::WebcamManager_2_0_0, [353](#)
- getZoneInfos
 - datetime::DateTime_2_0_0, [114](#)
- HARDWARE
 - peripheral::G2Production_2_0_0, [155](#)
- HERTZ
 - lhxmodel::Parameter_2_0_1, [227](#)
- HIGH
 - webcam, [59](#)
- HOURS
 - lhxmodel::Parameter_2_0_1, [227](#)
- HTS
 - pduModel::TransferSwitch_3_1_1, [333](#)
- HUMIDITY_REL
 - lhxmodel::Parameter_2_0_1, [227](#)
- hmi, [42](#)
- hmi::InternalBeeper
 - OFF, [174](#)
 - ON_ACTIVATION, [174](#)
 - ON_NOTIFICATION, [174](#)
- hmi::ExternalBeeper_1_0_1, [144](#)
 - alarm, [145](#)
 - off, [145](#)
 - on, [145](#)
- hmi::InternalBeeper, [173](#)
 - activate, [174](#)
 - getState, [174](#)
 - isMuted, [174](#)
 - mute, [174](#)
 - State, [174](#)
- StateChangedEvent, [175](#)
- IF_MODE_USB_DEVICE
 - net, [46](#)
- IF_MODE_WIRED
 - net, [46](#)
- IF_MODE_WIRELESS
 - net, [46](#)
- INCOMPATIBLE
 - pduModel::Controller_3_0_0, [110](#)
- INCOMPLETE
 - firmware, [41](#)
- INITIALIZING
 - webcam::StorageManager_1_0_1, [316](#)
- INLET_CTRL
 - pduModel::Controller_3_0_0, [110](#)
- INTERNAL_ERROR
 - peripheral::PackageInfo_2_0_0, [225](#)
- IdentificationStartedEvent
 - pduModel::Unit_2_0_1, [339](#)
- identify
 - pduModel::Unit_2_0_1, [339](#)
- idl, [42](#)
 - Event, [42](#)
- ImageState
 - firmware, [41](#)
- installPendingKeyPair
 - cert::ServerSSLCert, [287](#)
- InterfaceMode_2_0_0
 - net, [46](#)
- lpfwPolicy
 - security, [52](#)
- isAvailable
 - webcam::Channel, [96](#)
- isDaemonRunning
 - sys::System, [323](#)
- isEnabled
 - lhx::Support, [322](#)
 - pduModel::Inlet_1_2_6, [171](#)
- isFactoryConfigModeEnabled
 - production::Production, [245](#)
- isLoadSheddingActive
 - pduModel::Pdu_3_0_0, [234](#)
- isMuted
 - hmi::InternalBeeper, [174](#)
- JPEG
 - webcam, [59](#)
- JAN
 - event::TimerEventManager_2_0_0, [331](#)
- jsonrpc, [42](#)
- jsonrpc::NameService, [204](#)
 - lookup, [205](#)
- KERBEROS
 - auth, [36](#)
- L1
 - pduModel, [48](#)

- L1_N
 - pdumodel::Bcm, [86](#)
- L2
 - pdumodel, [48](#)
- L2_N
 - pdumodel::Bcm, [86](#)
- L3
 - pdumodel, [48](#)
- L3_N
 - pdumodel::Bcm, [86](#)
- LAN_DUPLEX_AUTO
 - net, [47](#)
- LAN_DUPLEX_FULL
 - net, [47](#)
- LAN_DUPLEX_HALF
 - net, [47](#)
- LAN_DUPLEX_UNKNOWN
 - net, [47](#)
- LAN_SPEED_1000MBIT
 - net, [47](#)
- LAN_SPEED_100MBIT
 - net, [47](#)
- LAN_SPEED_10MBIT
 - net, [47](#)
- LAN_SPEED_AUTO
 - net, [47](#)
- LAN_SPEED_UNKNOWN
 - net, [47](#)
- LDAP
 - auth, [36](#)
- LEAF
 - event::Engine::EventDesc, [142](#)
- LED_MODE_BLINK_FAST
 - assetmgrmodel::AssetStripConfig_1_0_1, [78](#)
- LED_MODE_BLINK_SLOW
 - assetmgrmodel::AssetStripConfig_1_0_1, [78](#)
- LED_MODE_OFF
 - assetmgrmodel::AssetStripConfig_1_0_1, [78](#)
- LED_MODE_ON
 - assetmgrmodel::AssetStripConfig_1_0_1, [78](#)
- LED_OPERATION_AUTO
 - assetmgrmodel::AssetStripConfig_1_0_1, [79](#)
- LED_OPERATION_MANUAL
 - assetmgrmodel::AssetStripConfig_1_0_1, [79](#)
- LEFT
 - test::Display_1_0_1, [129](#)
- LITERS_PER_HOUR
 - lhxmodel::Parameter_2_0_1, [227](#)
- LITERS_PER_MINUTE
 - lhxmodel::Parameter_2_0_1, [227](#)
- LOCAL
 - auth, [36](#)
 - webcam::StorageManager_1_0_1, [316](#)
- LOG_LEVEL_DEBUG
 - diag::DiagLogSettings, [126](#)
- LOG_LEVEL_ERR
 - diag::DiagLogSettings, [126](#)
- LOG_LEVEL_INFO
 - diag::DiagLogSettings, [126](#)
- LOG_LEVEL_NONE
 - diag::DiagLogSettings, [126](#)
- LOG_LEVEL_TRACE
 - diag::DiagLogSettings, [126](#)
- LOG_LEVEL_WARN
 - diag::DiagLogSettings, [126](#)
- LOW
 - webcam, [59](#)
- LEDMode
 - assetmgrmodel::AssetStripConfig_1_0_1, [78](#)
- LEDOperationMode
 - assetmgrmodel::AssetStripConfig_1_0_1, [78](#)
- LanDuplex
 - net, [46](#)
- LanSpeed
 - net, [47](#)
- leaveRS485ConfigMode
 - pdumodel::Pdu_3_0_0, [235](#)
- LengthEnum
 - usermgmt, [57](#)
- lhx, [43](#)
- lhx::Support, [321](#)
 - isEnabled, [322](#)
 - setEnabled, [322](#)
- lhxmodel, [43](#)
- lhxmodel::Parameter_2_0_1
 - AMPERE, [227](#)
 - BAR, [227](#)
 - BINARY, [226](#)
 - GRAMMS, [227](#)
 - HERTZ, [227](#)
 - HOURS, [227](#)
 - HUMIDITY_REL, [227](#)
 - LITERS_PER_HOUR, [227](#)
 - LITERS_PER_MINUTE, [227](#)
 - METER, [227](#)
 - METERS_PER_SECOND, [227](#)
 - MINUTES, [227](#)
 - NEWTON, [227](#)
 - NONE, [226](#)
 - NUMBER, [226](#)
 - OHM, [227](#)
 - PASCAL, [227](#)
 - PERCENT, [227](#)
 - SECONDS, [227](#)
 - SIEMENS, [227](#)
 - TEMP_ABS, [226](#)
 - TEMP_REL, [226](#)
 - TIME, [227](#)
 - VOLT, [227](#)
- lhxmodel::Config_1_0_1, [103](#)
 - ComSettingsChangedEvent, [105](#)
 - getComSettings, [104](#)
 - getName, [104](#)
 - PortNameChangedEvent, [105](#)
 - setComSettings, [104](#)
 - setName, [105](#)

- lhxmodel::Config_1_0_1::ComSettings, 102
- lhxmodel::Lhx_3_2_1, 181
 - acknowledgeAlertStatus, 182
 - getMetaData, 182
 - getSettings, 182
 - OpStateChangedEvent, 183
 - setMaximumCoolingRequest, 183
 - setPowerState, 183
 - setSettings, 183
 - SettingsChangedEvent, 183
- lhxmodel::Lhx_3_2_1::AlertStatus, 68
- lhxmodel::Lhx_3_2_1::Capabilities, 88
- lhxmodel::Lhx_3_2_1::MetaData, 198
- lhxmodel::Lhx_3_2_1::OpState, 215
- lhxmodel::Lhx_3_2_1::ParamCfg, 225
- lhxmodel::Lhx_3_2_1::Settings, 296
- lhxmodel::Parameter_2_0_1, 225
 - getMetaData, 227
 - getRawValue, 227
 - getValue, 227
 - MetaDataChangedEvent, 228
 - setRawValue, 227
 - Unit, 226
 - ValueChangedEvent, 228
- lhxmodel::Parameter_2_0_1::MetaData, 199
- lhxmodel::Parameter_2_0_1::Status, 313
- lhxmodel::Parameter_2_0_1::Value, 349
- lhxmodel::Sensor_4_0_0, 268
 - getMetaData, 270
 - getReading, 270
 - getThresholds, 270
 - ReadingChangedEvent, 271
 - setThresholds, 270
 - StateChangedEvent, 271
 - ThresholdsChangedEvent, 271
- lhxmodel::Sensor_4_0_0::MetaData, 199
 - numDecDigits, 200
- lhxmodel::Sensor_4_0_0::NumThresholds, 214
- lhxmodel::Sensor_4_0_0::Reading, 251
- LineConfig
 - pdumodel::Bcm, 86
- listActionTypes
 - event::Engine, 136
- listEventDescs
 - event::Engine, 136
- listRules
 - event::Engine, 137
- listServers
 - servermon::ServerMonitor_2_0_0, 283
- listTcpConnections
 - net::Diagnostics, 128
- LoadSheddingModeChangedEvent
 - pdumodel::Pdu_3_0_0, 236
- LogLevel
 - diag::DiagLogSettings, 126
- LoggedSensorsChangedEvent
 - sensors::Logger_2_1_2, 191
- logging, 43
 - BACKWARD, 44
 - FORWARD, 44
 - RangeDirection, 44
 - logging::DebugLog, 115
 - getEntries, 115
 - getFirstId, 115
 - getLastId, 115
 - logging::EventLog_1_0_1, 143
 - getEntries, 143
 - getFilteredEntries, 143
 - getFirstId, 143
 - getLastId, 144
 - logging::LogEntry, 185
 - lookup
 - jsonrpc::NameService, 205
- MD5
 - um::SnmpV3, 306
- METER
 - lhxmodel::Parameter_2_0_1, 227
 - usermgmt, 57
- METER_CTRL
 - pdumodel::Controller_3_0_0, 110
- METERS_PER_SECOND
 - lhxmodel::Parameter_2_0_1, 227
- MINUTES
 - lhxmodel::Parameter_2_0_1, 227
- MJPEG
 - webcam, 59
- MatchType
 - event::Engine::Condition, 103
- maxSignDays
 - cert::ServerSSLCert::Info, 168
- MetaDataChangedEvent
 - lhxmodel::Parameter_2_0_1, 228
 - pdumodel::Controller_3_0_0, 111
 - sensors::NumericSensor_4_0_0, 214
- modelpush, 44
- modelpush::ModelPush, 203
 - getConfiguration, 203
 - setConfiguration, 203
- modelpush::ModelPush::Configuration, 105
- ModemEvent
 - serial::SerialPort_2_0_0, 280
- modifyAction
 - event::Engine, 137
- modifyRule
 - event::Engine, 137
- modifyServer
 - servermon::ServerMonitor_2_0_0, 283
- modifyTimerEvent
 - event::TimerEventManager_2_0_0, 330
- mute
 - hmi::InternalBeeper, 174
- muteBuzzer
 - pdumodel::Unit_2_0_1, 339
- NEUTRAL
 - pdumodel, 48

- NEWTON
 - lhxmodel::Parameter_2_0_1, 227
- NFS
 - webcam::StorageManager_1_0_1, 316
- NO_AUTH_NO_PRIV
 - um::SnmpV3, 307
- NODE
 - event::Engine::EventDesc, 142
- NONE
 - assetmgrmodel::AssetStrip_2_0_1, 74
 - firmware, 41
 - lhxmodel::Parameter_2_0_1, 226
- NORMAL
 - pdumodel::Unit_2_0_1, 338
 - peripheral::PackageInfo_2_0_0, 225
 - test::Display_1_0_1, 129
 - webcam, 59
- NTP
 - datetime::DateTime_2_0_0, 113
- NUMBER
 - lhxmodel::Parameter_2_0_1, 226
- NO_ERROR
 - assetmgrmodel::AssetStrip_2_0_1, 76
 - assetmgrmodel::AssetStripLogger_1_0_2, 83
 - event::AlarmManager, 67
 - event::TimerEventManager_2_0_0, 331
 - portsmodel::Port_2_0_1, 240
 - smartcard::CardReader, 90
 - webcam::Channel, 97
 - webcam::StorageManager_1_0_1, 319
 - webcam::WebcamManager_2_0_0, 354
- net, 44
 - AUTH_EAP, 46
 - AUTH_NONE, 46
 - AUTH_PSK, 46
 - AUTO, 46
 - AuthenticationMode, 46
 - AutoConfigs, 46
 - DHCP, 46
 - EAP_MSCHAPv2, 46
 - EAP_PEAP, 46
 - EapInnerMethod, 46
 - EapOuterMethod, 46
 - IF_MODE_USB_DEVICE, 46
 - IF_MODE_WIRED, 46
 - IF_MODE_WIRELESS, 46
 - InterfaceMode_2_0_0, 46
 - LAN_DUPLEX_AUTO, 47
 - LAN_DUPLEX_FULL, 47
 - LAN_DUPLEX_HALF, 47
 - LAN_DUPLEX_UNKNOWN, 47
 - LAN_SPEED_1000MBIT, 47
 - LAN_SPEED_100MBIT, 47
 - LAN_SPEED_10MBIT, 47
 - LAN_SPEED_AUTO, 47
 - LAN_SPEED_UNKNOWN, 47
 - LanDuplex, 46
 - LanSpeed, 47
 - STATIC, 46
 - net::Diagnostics, 127
 - listTcpConnections, 128
 - ping, 128
 - traceRoute, 128
 - net::EapSettings, 131
 - net::IPv4RoutingEntry, 176
 - net::IPv6RoutingEntry, 176
 - net::InterfaceState_2_0_0, 172
 - net::LanInterfaceParameters_2_0_0, 177
 - net::LanInterfaceSettings, 178
 - net::LanLinkMode, 178
 - net::Net_2_0_2, 205
 - getBridgeSlaveCount, 206
 - getMACs, 206
 - getNetworkConfigIP, 206
 - getNetworkConfigIPv4, 207
 - getNetworkConfigIPv6, 207
 - getNetworkConfigInterface, 206
 - getNetworkConfigServices, 207
 - setNetworkConfigIP, 207
 - setNetworkConfigIPv4, 207
 - setNetworkConfigIPv6, 208
 - setNetworkConfigLan, 208
 - setNetworkConfigServices, 208
 - setNetworkConfigWLan, 208
 - net::NetworkActiveValuesIPv6, 209
 - net::NetworkConfigIP, 209
 - net::NetworkConfigIPv4, 210
 - net::NetworkConfigIPv6, 211
 - net::ServiceConfig, 290
 - net::WirelessInterfaceSettings, 355
 - newSession
 - session::SessionManager, 294
 - noiseThreshold
 - sensors::NumericSensor_4_0_0::MetaData, 194
 - numDecDigits
 - lhxmodel::Sensor_4_0_0::MetaData, 200
 - NumberingMode
 - assetmgrmodel::AssetStripConfig_1_0_1, 79
 - OFF
 - hmi::InternalBeeper, 174
 - sensors::Sensor_4_0_0, 274
 - OHM
 - lhxmodel::Parameter_2_0_1, 227
 - OK
 - pdumodel::Controller_3_0_0, 110
 - ON
 - sensors::Sensor_4_0_0, 274
 - ON_ACTIVATION
 - hmi::InternalBeeper, 174
 - ON_NOTIFICATION
 - hmi::InternalBeeper, 174
 - ONEWIRE_CHAIN_POS
 - peripheral, 49
 - ONEWIRE_DEV_PORT
 - peripheral, 49
 - ONEWIRE_HUB_PORT

- peripheral, [49](#)
- ONEWIRE_ONBOARD
 - peripheral, [49](#)
- OPEN_LDAP
 - auth::ldapsrv, [37](#)
- OR
 - event::Engine::Condition, [103](#)
- OUTLET_CTRL
 - pdumodel::Controller_3_0_0, [110](#)
- off
 - hmi::ExternalBeeper_1_0_1, [145](#)
- on
 - hmi::ExternalBeeper_1_0_1, [145](#)
- OnOffState
 - sensors::Sensor_4_0_0, [274](#)
- onlineCheckEnabled
 - firmware::Firmware_1_0_1, [149](#)
- Op
 - event::Engine::Condition, [103](#)
- OpStateChangedEvent
 - lhxmodel::Lhx_3_2_1, [183](#)
- Orientation
 - assetmgrmodel::AssetStripConfig_1_0_1, [79](#)
 - pdumodel::Unit_2_0_1, [338](#)
 - test::Display_1_0_1, [129](#)
- OrientationChangedEvent
 - assetmgrmodel::AssetStrip_2_0_1, [76](#)
- outletPowerStateSequence
 - pdumodel::Pdu_3_0_0::Settings, [297](#)
- OutletSequenceStateChangedEvent
 - pdumodel::Pdu_3_0_0, [236](#)
- PAP
 - radius, [51](#)
- PASCAL
 - lhxmodel::Parameter_2_0_1, [227](#)
 - usermgmt, [58](#)
- PERCENT
 - lhxmodel::Parameter_2_0_1, [227](#)
- PINNED
 - portsmodel::Port_2_0_1, [239](#)
- PROC_COUNT
 - res_mon::Entry, [139](#)
- PROC_CPU_USAGE
 - res_mon::Entry, [139](#)
- PROC_FREE_FILE_DESC
 - res_mon::Entry, [139](#)
- PROC_LIFE_TIME
 - res_mon::Entry, [139](#)
- PROC_VM_SIZE
 - res_mon::Entry, [139](#)
- PS_OFF
 - pdumodel::Outlet_1_5_6, [217](#)
- PS_ON
 - pdumodel::Outlet_1_5_6, [217](#)
- PSI
 - usermgmt, [58](#)
- PackageEvent
 - peripheral::DeviceManager_2_0_0, [121](#)
- pdumodel, [47](#)
 - L1, [48](#)
 - L2, [48](#)
 - L3, [48](#)
 - NEUTRAL, [48](#)
 - PowerLine, [48](#)
- pdumodel::Bcm
 - L1_N, [86](#)
 - L2_N, [86](#)
 - L3_N, [86](#)
 - TURNRATIO, [86](#)
 - UNCONNECTED, [86](#)
 - VOLTAGE, [86](#)
- pdumodel::Controller_3_0_0
 - COMMUNICATION_FAILURE, [110](#)
 - COMMUNICATION_UNSTABLE, [110](#)
 - INCOMPATIBLE, [110](#)
 - INLET_CTRL, [110](#)
 - METER_CTRL, [110](#)
 - OK, [110](#)
 - OUTLET_CTRL, [110](#)
 - UNKNOWN, [110](#)
- pdumodel::Outlet_1_5_6
 - PS_OFF, [217](#)
 - PS_ON, [217](#)
 - SS_LASTKNOWN, [218](#)
 - SS_OFF, [218](#)
 - SS_ON, [218](#)
 - SS_PDUDEF, [218](#)
- pdumodel::OverCurrentProtector_2_1_2
 - BREAKER_1POLE, [222](#)
 - BREAKER_2POLE, [222](#)
 - BREAKER_3POLE, [222](#)
 - FUSE, [222](#)
 - FUSE_PAIR, [222](#)
 - RCBO_2POLE, [222](#)
 - RCBO_3POLE, [222](#)
 - RCBO_4POLE, [222](#)
- pdumodel::Pdu_3_0_0
 - SS_LASTKNOWN, [231](#)
 - SS_OFF, [231](#)
 - SS_ON, [231](#)
- pdumodel::TransferSwitch_3_1_1
 - ATS, [333](#)
 - HTS, [333](#)
 - REASON_AUTO_RETRANSFER, [333](#)
 - REASON_INTERNAL_FAILURE, [333](#)
 - REASON_MANUAL_TRANSFER, [333](#)
 - REASON_OVERHEAT, [333](#)
 - REASON_OVERLOAD, [333](#)
 - REASON_POWER_FAILURE, [333](#)
 - REASON_POWER_QUALITY, [333](#)
 - REASON_STARTUP, [333](#)
 - REASON_UNKNOWN, [333](#)
 - STS, [333](#)
- pdumodel::Unit_2_0_1
 - FLIPPED, [338](#)
 - NORMAL, [338](#)

- pdumodel::Ade, 64
 - getCalibrationData, 64
 - getLatestSample, 64
 - getMetaData, 65
 - setCalibrationData, 65
- pdumodel::Ade::MetaData, 200
- pdumodel::Ade::Sample, 262
- pdumodel::Bcm, 85
 - getChannelConfigs, 86
 - getChannelCount, 86
 - LineConfig, 86
 - setChannelConfig, 87
 - TransformerType, 86
- pdumodel::Bcm::ChannelConfig, 100
- pdumodel::Bcm::PhaseConfig, 236
- pdumodel::CircuitBreakerStatistic, 101
- pdumodel::Controller_3_0_0, 108
 - getCommunicationStatus, 110
 - getMetaData, 110
 - getStatistics, 110
 - MetaDataChangedEvent, 111
 - Status, 110
 - StatusChangedEvent, 111
 - Type, 110
- pdumodel::Controller_3_0_0::MetaData, 196
- pdumodel::CtrlStatistic, 111
- pdumodel::DoublePole_2_0_0, 130
- pdumodel::EDevice, 132
 - getChildren, 132
 - getParents, 132
- pdumodel::Inlet_1_2_6, 170
 - EnableStateChangedEvent, 172
 - getMetaData, 171
 - getPoles, 171
 - getSensors, 171
 - getSettings, 171
 - isEnabled, 171
 - setEnabled, 171
 - setSettings, 172
 - SettingsChangedEvent, 172
- pdumodel::Inlet_1_2_6::MetaData, 197
- pdumodel::Inlet_1_2_6::Sensors, 275
- pdumodel::Inlet_1_2_6::Settings, 302
- pdumodel::MemoryMapController_3_0_0, 192
 - readMemory, 192
 - writeMemory, 193
- pdumodel::Nameplate, 204
- pdumodel::Nameplate::Rating, 250
- pdumodel::Outlet_1_5_6, 216
 - cyclePowerState, 218
 - getController, 218
 - getIOP, 218
 - getMetaData, 218
 - getSensors, 218
 - getSettings, 219
 - getState, 219
 - PowerControlEvent, 220
 - PowerState, 217
 - setPowerState, 219
 - setSettings, 219
 - SettingsChangedEvent, 220
 - StartupState, 217
 - StateChangedEvent, 220
 - unstick, 219
- pdumodel::Outlet_1_5_6::LedState, 180
- pdumodel::Outlet_1_5_6::MetaData, 195
- pdumodel::Outlet_1_5_6::Sensors, 277
- pdumodel::Outlet_1_5_6::Settings, 298
- pdumodel::Outlet_1_5_6::State, 308
 - cycleInProgress, 309
 - switchOnInProgress, 309
- pdumodel::OutletStatistic, 221
- pdumodel::OverCurrentProtector_2_1_2, 221
 - getInlet, 223
 - getMetaData, 223
 - getOCP, 223
 - getPoles, 223
 - getSensors, 223
 - getSettings, 223
 - setSettings, 223
 - SettingsChangedEvent, 224
 - Type, 222
- pdumodel::OverCurrentProtector_2_1_2::MetaData, 196
- pdumodel::OverCurrentProtector_2_1_2::Sensors, 274
- pdumodel::OverCurrentProtector_2_1_2::Settings, 297
- pdumodel::Pdu_3_0_0, 229
 - cycleAllOutletPowerStates, 231
 - cycleMultipleOutletPowerStates, 231
 - enterRS485ConfigModeAndAssignCtrlBoard-Address, 232
 - enterRS485ConfigModeAndAssignSCBoard-Address, 232
 - getBeeper, 232
 - getControllers, 232
 - getFeaturePorts, 233
 - getInlets, 233
 - getMetaData, 233
 - getNameplate, 233
 - getOutletSequenceState, 233
 - getOutlets, 233
 - getOverCurrentProtectors, 233
 - getPeripheralDeviceManager, 234
 - getSensorLogger, 234
 - getSensors, 234
 - getSettings, 234
 - getStatistic, 234
 - getTransferSwitches, 234
 - isLoadSheddingActive, 234
 - leaveRS485ConfigMode, 235
 - LoadSheddingModeChangedEvent, 236
 - OutletSequenceStateChangedEvent, 236
 - setAllOutletPowerStates, 235
 - setLoadSheddingActive, 235
 - setMultipleOutletPowerStates, 235
 - setSettings, 235

- SettingsChangedEvent, 236
- StartupState, 231
- pduModel::Pdu_3_0_0::MetaData, 197
- pduModel::Pdu_3_0_0::OutletSequenceState, 220
- pduModel::Pdu_3_0_0::Sensors, 275
- pduModel::Pdu_3_0_0::Settings, 296
 - outletPowerStateSequence, 297
- pduModel::Pdu_3_0_0::Statistic, 311
- pduModel::Pole_2_0_0, 237
- pduModel::PowerQualitySensor_2_0_0, 242
- pduModel::Rating, 249
- pduModel::ResidualCurrentStateSensor_2_0_0, 253
 - startSelfTest, 254
- pduModel::ThrowPole, 327
- pduModel::TransferSwitch_3_1_1, 331
 - getLastTransferReason, 334
 - getLastTransferWaveform, 334
 - getMetaData, 334
 - getPoles, 334
 - getSensors, 334
 - getSettings, 334
 - getStatistics, 334
 - setSettings, 335
 - SettingsChangedEvent, 335
 - TransferReason, 333
 - transferToSource, 335
 - Type, 333
- pduModel::TransferSwitch_3_1_1::MetaData, 201
- pduModel::TransferSwitch_3_1_1::Sensors, 276
- pduModel::TransferSwitch_3_1_1::Settings, 301
- pduModel::TransferSwitch_3_1_1::Statistics, 311
- pduModel::TransferSwitch_3_1_1::WaveformSample, 350
- pduModel::Unit_2_0_1, 337
 - getDisplayOrientation, 338
 - getMetaData, 338
 - getSettings, 338
 - IdentificationStartedEvent, 339
 - identify, 339
 - muteBuzzer, 339
 - Orientation, 338
 - setSettings, 339
- pduModel::Unit_2_0_1::MetaData, 195
- pduModel::Unit_2_0_1::Settings, 301
- performOnlineCheck
 - firmware::Firmware_1_0_1, 149
- peripheral, 48
 - Device_2_0_0, 50
 - ONEWIRE_CHAIN_POS, 49
 - ONEWIRE_DEV_PORT, 49
 - ONEWIRE_HUB_PORT, 49
 - ONEWIRE_ONBOARD, 49
 - PortType, 49
- peripheral::DeviceManager_2_0_0
 - FREEFORM, 119
 - RACKUNITS, 119
 - UPDATE_FAILED, 119
 - UPDATE_STARTED, 119
 - UPDATE_SUCCESSFUL, 119
- peripheral::G2Production_2_0_0
 - BROWNOUT, 155
 - FIRMWARE, 155
 - FUNCTION, 155
 - HARDWARE, 155
 - RESERVED, 155
 - WATCHDOG, 155
- peripheral::PackageInfo_2_0_0
 - CONFIG_ERROR, 225
 - FW_UPDATE, 225
 - INTERNAL_ERROR, 225
 - NORMAL, 225
- peripheral::Address_2_0_0, 63
- peripheral::DeviceID_2_0_0, 116
- peripheral::DeviceManager_2_0_0, 117
 - DeviceEvent, 121
 - DeviceFirmwareUpdateState, 119
 - DeviceFirmwareUpdateStateChangedEvent, 121
 - getDeviceSlot, 120
 - getDeviceSlots, 120
 - getDeviceTypeInfo, 120
 - getDiscoveredDevices, 120
 - getDiscoveredPackageInfos, 120
 - getMetaData, 120
 - getSettings, 120
 - getStatistics, 121
 - PackageEvent, 121
 - setSettings, 121
 - SettingsChangedEvent, 121
 - UnknownDeviceAttachedEvent, 121
 - ZCoordMode, 119
- peripheral::DeviceManager_2_0_0::DeviceTypeInfo, 124
- peripheral::DeviceManager_2_0_0::MetaData, 198
- peripheral::DeviceManager_2_0_0::Settings, 300
- peripheral::DeviceManager_2_0_0::Statistics, 312
- peripheral::DeviceSlot_2_0_0, 122
 - assign, 123
 - assignAddress, 123
 - DeviceChangedEvent, 124
 - getDevice, 123
 - getSettings, 123
 - setSettings, 123
 - SettingsChangedEvent, 124
 - unassign, 124
- peripheral::DeviceSlot_2_0_0::Location, 184
- peripheral::DeviceSlot_2_0_0::Settings, 299
- peripheral::G2Production_2_0_0, 154
 - ConfigurationSpace, 155
 - eraseConfigurationSpace, 155
 - getFirmwareInfo, 155
 - readConfigurationSpace, 156
 - readRegisters, 156
 - reset, 156
 - ResetMethod, 155
 - updateFirmware, 156
 - writeConfigurationSpace, 157

- writeRegisterBits, 157
- writeRegisters, 157
- peripheral::G2Production_2_0_0::FirmwareInfo, 150
- peripheral::PackageInfo_2_0_0, 224
 - State, 225
- peripheral::PackageInfo_2_0_0::FirmwareInfo, 150
- peripheral::PackageInfo_2_0_0::FirmwareInfo::Version, 349
- peripheral::PackageInfo_2_0_0::HardwareInfo, 162
- peripheral::PosElement, 242
- ping
 - net::Diagnostics, 128
- PixelFormat
 - webcam, 59
- pollEvents
 - event::Channel_1_0_1, 99
- pollEventsNb
 - event::Channel_1_0_1, 99
- PortNameChangedEvent
 - lhxmodel::Config_1_0_1, 105
- PortState
 - serial::SerialPort_2_0_0, 279
- PortType
 - peripheral, 49
- portsmodel, 50
- portsmodel::Port_2_0_1
 - AUTO, 239
 - DISABLED, 239
 - PINNED, 239
- portsmodel::Port_2_0_1, 238
 - DetectionType, 239
 - DeviceChangedEvent, 240
 - getDetectableDevices, 239
 - getDevice, 239
 - getDeviceConfig, 239
 - getProperties, 240
 - NO_ERROR, 240
 - PropertiesChangedEvent, 241
 - setDetectionMode, 240
 - setName, 240
- portsmodel::Port_2_0_1::DetectionMode, 116
- portsmodel::Port_2_0_1::Properties, 245
- PowerControlEvent
 - pdumodel::Outlet_1_5_6, 220
- PowerLine
 - pdumodel, 48
- PowerState
 - pdumodel::Outlet_1_5_6, 217
- PressureEnum
 - usermgmt, 57
- Priority
 - webcam, 59
- PrivProtocol
 - um::SnmpV3, 306
- production, 50
- production::Production, 244
 - enterFactoryConfigMode, 244
 - isFactoryConfigModeEnabled, 245
- PropertiesChangedEvent
 - portsmodel::Port_2_0_1, 241
- Protocol
 - datetime::DateTime_2_0_0, 113
- pushEvents
 - event::Consumer, 108
- RACKUNITS
 - peripheral::DeviceManager_2_0_0, 119
- RADIUS
 - auth, 36
- RCBO_2POLE
 - pdumodel::OverCurrentProtector_2_1_2, 222
- RCBO_3POLE
 - pdumodel::OverCurrentProtector_2_1_2, 222
- RCBO_4POLE
 - pdumodel::OverCurrentProtector_2_1_2, 222
- REACHABLE
 - servermon::ServerMonitor_2_0_0, 282
- READY
 - webcam::StorageManager_1_0_1, 316
- REASON_AUTO_RETRANSFER
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_INTERNAL_FAILURE
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_MANUAL_TRANSFER
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_OVERHEAT
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_OVERLOAD
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_POWER_FAILURE
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_POWER_QUALITY
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_STARTUP
 - pdumodel::TransferSwitch_3_1_1, 333
- REASON_UNKNOWN
 - pdumodel::TransferSwitch_3_1_1, 333
- REJECT
 - security, 52
- RESERVED
 - peripheral::G2Production_2_0_0, 155
- RESTORE_FAILED
 - bulkcfg::BulkConfiguration, 88
- RESTORE_OK
 - bulkcfg::BulkConfiguration, 88
- RESTORE_PENDING
 - bulkcfg::BulkConfiguration, 88
- RGB
 - webcam, 59
- RIGHT
 - test::Display_1_0_1, 129
- RackUnitChangedEvent
 - assetmgrmodel::AssetStrip_2_0_1, 77
- RackUnitSettingsChangedEvent
 - assetmgrmodel::AssetStripConfig_1_0_1, 81
- radius, 50
 - AuthType, 51

- CHAP, 51
- PAP, 51
- radius::ServerSettings, 285
- range
 - sensors::NumericSensor_4_0_0::MetaData, 194
- RangeDirection
 - logging, 44
- readConfigurationSpace
 - peripheral::G2Production_2_0_0, 156
- readMemory
 - pduModel::MemoryMapController_3_0_0, 192
- readRegisters
 - peripheral::G2Production_2_0_0, 156
- ReadingChangedEvent
 - lhXmodel::Sensor_4_0_0, 271
 - sensors::NumericSensor_4_0_0, 214
- rearmRule
 - event::Engine, 137
- reboot
 - firmware::Firmware_1_0_1, 149
- RecordType
 - assetmgrmodel::AssetStripLogger_1_0_2, 82
- removeClientType
 - webcam::WebcamManager_2_0_0, 353
- removeImages
 - webcam::StorageManager_1_0_1, 318
- res_mon::Entry
 - FS_FREE_INODES, 139
 - FS_FREE_SPACE, 139
 - GLOBAL_CPU_USAGE, 139
 - GLOBAL_FREE_MEM, 139
 - GLOBAL_PROC_COUNT, 139
 - PROC_COUNT, 139
 - PROC_CPU_USAGE, 139
 - PROC_FREE_FILE_DESC, 139
 - PROC_LIFE_TIME, 139
 - PROC_VM_SIZE, 139
- res_mon, 51
- res_mon::Entry, 138
 - Type, 138
- res_mon::ResMon, 254
 - getDataEntries, 255
- reset
 - peripheral::G2Production_2_0_0, 156
- ResetEvent
 - sensors::AccumulatingNumericSensor_2_0_0, 62
- ResetMethod
 - peripheral::G2Production_2_0_0, 155
- resolution
 - sensors::NumericSensor_4_0_0::MetaData, 194
- restartDaemon
 - sys::System, 324
- RoleAccessPolicy
 - security, 52
- SCANMODE_BOTH
 - assetmgrmodel::AssetStripConfig_1_0_1, 79
- SCANMODE_DISABLED
 - assetmgrmodel::AssetStripConfig_1_0_1, 79
- SECONDS
 - lhXmodel::Parameter_2_0_1, 227
- SHA1
 - um::SnmpV3, 306
- SIEMENS
 - lhXmodel::Parameter_2_0_1, 227
- SIMPLE
 - assetmgrmodel::AssetStrip_2_0_1, 74
- SINGLE
 - assetmgrmodel::AssetStrip_2_0_1, 74
- SPD_10
 - test::Ethernet, 140
- SPD_100
 - test::Ethernet, 140
- SPD_1000
 - test::Ethernet, 140
- SS_LASTKNOWN
 - pduModel::Outlet_1_5_6, 218
 - pduModel::Pdu_3_0_0, 231
- SS_OFF
 - pduModel::Outlet_1_5_6, 218
 - pduModel::Pdu_3_0_0, 231
- SS_ON
 - pduModel::Outlet_1_5_6, 218
 - pduModel::Pdu_3_0_0, 231
- SS_PDUDEF
 - pduModel::Outlet_1_5_6, 218
- STATE
 - event::Event, 141
- STATIC
 - datetime::DateTime_2_0_0, 113
 - net, 46
- STS
 - pduModel::TransferSwitch_3_1_1, 333
- SUCCESSFUL
 - firmware, 41
- STATE_NORMAL
 - pduModel::ResidualCurrentStateSensor_2_0_0, 254
- STATE_UNAVAILABLE
 - sensors::Logger_2_1_2, 191
- SUCCESS
 - serial::AnalogModem, 70
 - serial::GsmModem_1_0_1, 162
 - serial::SerialPort_2_0_0, 281
- SUN
 - event::TimerEventManager_2_0_0, 331
- ScanMode
 - assetmgrmodel::AssetStripConfig_1_0_1, 79
- security, 51
 - ACCEPT, 52
 - ALLOW, 53
 - DENY, 53
 - DROP, 52
 - lpfwPolicy, 52
 - REJECT, 52
 - RoleAccessPolicy, 52
 - security::lpFw_2_0_0, 175

- security::IpfwRule, 175
- security::PasswordSettings, 228
- security::RestrictedServiceAgreement, 255
- security::RoleAccessControl, 257
- security::RoleAccessRule, 257
- security::SSHSettings, 308
- security::Security_3_0_0, 263
 - getBlockSettings, 264
 - getFrontPanelPrivileges, 264
 - getHttpRedirSettings, 264
 - getIdleTimeoutSettings, 265
 - getRestrictedServiceAgreement, 265
 - getSSHSettings, 265
 - getSettings, 265
 - getSupportedFrontPanelPrivileges, 265
 - setBlockSettings, 265
 - setFrontPanelPrivileges, 266
 - setHttpRedirSettings, 266
 - setIdleTimeoutSettings, 266
 - setIpFwSettings, 266
 - setIpV6FwSettings, 266
 - setPwSettings, 267
 - setRestrictedServiceAgreement, 267
 - setRoleAccessControlSettings, 267
 - setRoleAccessControlSettingsV6, 267
 - setSSHSettings, 268
 - setSettings, 268
 - setSingleLoginLimitation, 268
- security::Security_3_0_0::Settings, 294
- security::ServiceAuthorization, 289
 - setPassword, 290
- SecurityLevel
 - um::SnmpV3, 306
- sendSms
 - serial::GsmModem_1_0_1, 161
- sendTestSms
 - serial::GsmModem_1_0_1, 161
- sensors, 53
- sensors::Sensor_4_0_0
 - OFF, 274
 - ON, 274
- sensors::AccumulatingNumericSensor_2_0_0, 61
 - ResetEvent, 62
- sensors::Logger_2_1_2, 185
 - getLogRow, 188
 - getLoggedSensors, 188
 - getPeripheralDeviceRecords, 188
 - getPeripheralDeviceTimedRecords, 188
 - getSensorRecords, 189
 - getSensorSetTimestamp, 189
 - getSensorTimedRecords, 189
 - getSettings, 190
 - getTimeStamps, 190
 - LoggedSensorsChangedEvent, 191
 - setLoggedSensors, 190
 - setSettings, 190
 - SettingsChangedEvent, 191
- sensors::Logger_2_1_2::LogRow, 191
- sensors::Logger_2_1_2::Record, 251
- sensors::Logger_2_1_2::SensorSet, 278
- sensors::Logger_2_1_2::Settings, 299
- sensors::Logger_2_1_2::TimedRecord, 327
- sensors::NumericSensor_4_0_0, 212
 - getDefaultThresholds, 213
 - getMetaData, 213
 - getReading, 213
 - getThresholds, 213
 - MetaDataChangedEvent, 214
 - ReadingChangedEvent, 214
 - setThresholds, 214
 - StateChangedEvent, 214
 - ThresholdsChangedEvent, 214
- sensors::NumericSensor_4_0_0::MetaData, 193
 - accuracy, 194
 - decdigits, 194
 - noiseThreshold, 194
 - range, 194
 - resolution, 194
 - tolerance, 194
- sensors::NumericSensor_4_0_0::Range, 249
- sensors::NumericSensor_4_0_0::Reading, 250
- sensors::NumericSensor_4_0_0::Reading::Status, 312
- sensors::NumericSensor_4_0_0::ThresholdCapabilities, 326
- sensors::NumericSensor_4_0_0::Thresholds, 326
- sensors::Sensor_4_0_0, 271
 - getTypeSpec, 274
 - OnOffState, 274
 - setType, 274
 - TypeSpecChangedEvent, 274
- sensors::Sensor_4_0_0::TypeSpec, 335
- sensors::StateSensor_4_0_0, 310
 - getState, 310
- sensors::StateSensor_4_0_0::State, 309
- sensors::Switch_2_0_0, 322
 - setState, 323
- serial, 53
- serial::GsmModem_1_0_1
 - UNKNOWN, 159
 - UNLOCKED, 159
 - WAITFORPIN, 159
 - WAITFORPUK, 159
- serial::SerialPort_2_0_0
 - ANALOGMODEM, 280
 - BR115200, 279
 - BR1200, 279
 - BR19200, 279
 - BR2400, 279
 - BR38400, 279
 - BR4800, 279
 - BR57600, 279
 - BR9600, 279
 - CONSOLE, 280
 - GSMMODEM, 280
- serial::AnalogModem, 69
 - CallEndedEvent, 70

- DialInEvent, [70](#)
- getSettings, [70](#)
- SUCCESS, [70](#)
- setSettings, [70](#)
- serial::AnalogModem::Settings, [298](#)
- serial::GsmModem_1_0_1, [158](#)
 - getInformation, [160](#)
 - getInformationWithPin, [160](#)
 - getSettings, [160](#)
 - getSimSecurityStatus, [160](#)
 - SUCCESS, [162](#)
 - sendSms, [161](#)
 - sendTestSms, [161](#)
 - setSettings, [161](#)
 - SimPinUpdatedEvent, [162](#)
 - SimSecurityStatus, [159](#)
 - SimSecurityStatusChangedEvent, [162](#)
 - unlockSimCard, [161](#)
- serial::GsmModem_1_0_1::Information, [169](#)
- serial::GsmModem_1_0_1::Settings, [296](#)
- serial::PortDispatcher_1_1_1, [241](#)
 - getPorts, [241](#)
- serial::SerialPort_2_0_0, [278](#)
 - BaudRate, [279](#)
 - getModem, [280](#)
 - getSettings, [280](#)
 - getState, [280](#)
 - ModemEvent, [280](#)
 - PortState, [279](#)
 - SUCCESS, [281](#)
 - setSettings, [280](#)
- serial::SerialPort_2_0_0::Settings, [302](#)
- serial::SerialPort_2_0_0::State, [308](#)
- ServerReachability
 - servermon::ServerMonitor_2_0_0, [282](#)
- ServerType
 - auth::ldapsrv, [37](#)
- servermon, [54](#)
- servermon::ServerMonitor_2_0_0
 - ERROR, [282](#)
 - REACHABLE, [282](#)
 - UNREACHABLE, [282](#)
 - WAITING, [282](#)
- servermon::ServerMonitor_2_0_0, [281](#)
 - addServer, [283](#)
 - deleteServer, [283](#)
 - getServer, [283](#)
 - listServers, [283](#)
 - modifyServer, [283](#)
 - ServerReachability, [282](#)
- servermon::ServerMonitor_2_0_0::Server, [281](#)
- servermon::ServerMonitor_2_0_0::ServerSettings, [285](#)
- servermon::ServerMonitor_2_0_0::ServerStatus, [288](#)
- session, [54](#)
- session::SessionManager
 - CLOSE_REASON_BROWSER_CLOSED, [292](#)
 - CLOSE_REASON_FORCED_DISCONNECT, [292](#)
 - CLOSE_REASON_LOGOUT, [292](#)
 - CLOSE_REASON_TIMEOUT, [292](#)
 - session::HistoryEntry, [163](#)
 - session::Session, [291](#)
 - session::SessionManager, [291](#)
 - closeCurrentSession, [293](#)
 - CloseReason, [292](#)
 - closeSession, [293](#)
 - getCurrentSession, [293](#)
 - getSession, [293](#)
 - getSessionHistory, [293](#)
 - getSessions, [293](#)
 - newSession, [294](#)
 - touchCurrentSession, [294](#)
 - touchSession, [294](#)
- setAccountPassword
 - usermgmt::User_1_0_1, [343](#)
- setAllOutletPowerStates
 - pduModel::Pdu_3_0_0, [235](#)
- setBlockSettings
 - security::Security_3_0_0, [265](#)
- setBuzzer
 - test::Unit_1_0_2, [337](#)
- setCalibrationData
 - pduModel::Ade, [65](#)
- setCfg
 - datetime::DateTime_2_0_0, [114](#)
- setChannelConfig
 - pduModel::Bcm, [87](#)
- setClientTypePriorities
 - webcam::WebcamManager_2_0_0, [354](#)
- setComSettings
 - lhxmodel::Config_1_0_1, [104](#)
- setConfiguration
 - devsettings::Sntp_1_0_1, [304](#)
 - devsettings::Snmp_1_0_2, [305](#)
 - modelpush::ModelPush, [203](#)
- setControls
 - webcam::Webcam_2_0_0, [351](#)
- setDefaultPreferences
 - usermgmt::UserManager_1_0_2, [348](#)
- setDetectionMode
 - portsmodel::Port_2_0_1, [240](#)
- setEnabled
 - lhx::Support, [322](#)
 - pduModel::Inlet_1_2_6, [171](#)
- setFrontPanelPrivileges
 - security::Security_3_0_0, [266](#)
- setHttpRedirSettings
 - security::Security_3_0_0, [266](#)
- setIdleTimeoutSettings
 - security::Security_3_0_0, [266](#)
- setIpFwSettings
 - security::Security_3_0_0, [266](#)
- setIpV6FwSettings
 - security::Security_3_0_0, [266](#)
- setLdapServers
 - auth::LdapManager_1_0_1, [179](#)
- setLoadSheddingActive

- pdumodel::Pdu_3_0_0, 235
- setLogLevelByCtxName
 - diag::DiagLogSettings, 126
- setLogLevelForAllCtxNames
 - diag::DiagLogSettings, 127
- setLoggedSensors
 - sensors::Logger_2_1_2, 190
- setMaximumCoolingRequest
 - lhxmodel::Lhx_3_2_1, 183
- setMultipleOutletPowerStates
 - pdumodel::Pdu_3_0_0, 235
- setMultipleRackUnitSettings
 - assetmgrmodel::AssetStripConfig_1_0_1, 80
- setName
 - lhxmodel::Config_1_0_1, 105
 - portsmodel::Port_2_0_1, 240
- setNetworkConfigIP
 - net::Net_2_0_2, 207
- setNetworkConfigIPv4
 - net::Net_2_0_2, 207
- setNetworkConfigIPv6
 - net::Net_2_0_2, 208
- setNetworkConfigLan
 - net::Net_2_0_2, 208
- setNetworkConfigServices
 - net::Net_2_0_2, 208
- setNetworkConfigWLAN
 - net::Net_2_0_2, 208
- setParameters
 - test::Ethernet, 140
- setPassword
 - security::ServiceAuthorization, 290
- setPolicy
 - auth::AuthManager, 83
- setPower
 - test::FeatSerial, 146
- setPowerState
 - lhxmodel::Lhx_3_2_1, 183
 - pdumodel::Outlet_1_5_6, 219
- setPreferences
 - usermgmt::User_1_0_1, 343
- setPwSettings
 - security::Security_3_0_0, 267
- setRackUnitSettings
 - assetmgrmodel::AssetStripConfig_1_0_1, 80
- setRadiusServers
 - auth::RadiusManager, 248
- setRawValue
 - lhxmodel::Parameter_2_0_1, 227
- setRestrictedServiceAgreement
 - security::Security_3_0_0, 267
- setRoleAccessControlSettings
 - security::Security_3_0_0, 267
- setRoleAccessControlSettingsV6
 - security::Security_3_0_0, 267
- setSSHSettings
 - security::Security_3_0_0, 268
- setSettings
 - cew::EnergyWiseManager, 133
 - devsettings::Modbus, 202
 - devsettings::Zeroconf, 356
 - lhxmodel::Lhx_3_2_1, 183
 - pdumodel::Inlet_1_2_6, 172
 - pdumodel::Outlet_1_5_6, 219
 - pdumodel::OverCurrentProtector_2_1_2, 223
 - pdumodel::Pdu_3_0_0, 235
 - pdumodel::TransferSwitch_3_1_1, 335
 - pdumodel::Unit_2_0_1, 339
 - peripheral::DeviceManager_2_0_0, 121
 - peripheral::DeviceSlot_2_0_0, 123
 - security::Security_3_0_0, 268
 - sensors::Logger_2_1_2, 190
 - serial::AnalogModem, 70
 - serial::GsmModem_1_0_1, 161
 - serial::SerialPort_2_0_0, 280
 - webcam::StorageManager_1_0_1, 318
 - webcam::Webcam_2_0_0, 351
- setSingleLoginLimitation
 - security::Security_3_0_0, 268
- setState
 - sensors::Switch_2_0_0, 323
- setStripSettings
 - assetmgrmodel::AssetStripConfig_1_0_1, 81
- setThresholds
 - lhxmodel::Sensor_4_0_0, 270
 - sensors::NumericSensor_4_0_0, 214
- setType
 - cascading::Cascading_1_0_1, 94
 - sensors::Sensor_4_0_0, 274
- setWebcamPriorities
 - webcam::WebcamManager_2_0_0, 354
- SettingsChangedEvent
 - lhxmodel::Lhx_3_2_1, 183
 - pdumodel::Inlet_1_2_6, 172
 - pdumodel::Outlet_1_5_6, 220
 - pdumodel::OverCurrentProtector_2_1_2, 224
 - pdumodel::Pdu_3_0_0, 236
 - pdumodel::TransferSwitch_3_1_1, 335
 - peripheral::DeviceManager_2_0_0, 121
 - peripheral::DeviceSlot_2_0_0, 124
 - sensors::Logger_2_1_2, 191
- SimPinUpdatedEvent
 - serial::GsmModem_1_0_1, 162
- SimSecurityStatus
 - serial::GsmModem_1_0_1, 159
- SimSecurityStatusChangedEvent
 - serial::GsmModem_1_0_1, 162
- smartcard, 54
 - smartcard::CardReader, 89
 - CardEvent, 90
 - getCardInformation, 90
 - getMetaData, 90
 - NO_ERROR, 90
 - smartcard::CardReader::CardInformation, 88
 - smartcard::CardReader::MetaData, 201
 - smartcard::CardReaderManager, 90

- CardReaderEvent, [91](#)
- getCardReaderById, [91](#)
- getCardReaders, [91](#)
- Speed
 - test::Ethernet, [140](#)
- startActivity
 - webcam::StorageManager_1_0_1, [318](#)
- startSelfTest
 - pdumodel::ResidualCurrentStateSensor_2_0_0, [254](#)
- startUpdate
 - firmware::Firmware_1_0_1, [149](#)
- StartupState
 - pdumodel::Outlet_1_5_6, [217](#)
 - pdumodel::Pdu_3_0_0, [231](#)
- State
 - assetmgrmodel::AssetStrip_2_0_1, [73](#)
 - hmi::InternalBeeper, [174](#)
 - peripheral::PackageInfo_2_0_0, [225](#)
- StateChangedEvent
 - assetmgrmodel::AssetStrip_2_0_1, [77](#)
 - hmi::InternalBeeper, [175](#)
 - lhxmodel::Sensor_4_0_0, [271](#)
 - pdumodel::Outlet_1_5_6, [220](#)
 - sensors::NumericSensor_4_0_0, [214](#)
- Status
 - bulkcfg::BulkConfiguration, [88](#)
 - pdumodel::Controller_3_0_0, [110](#)
- StatusChangedEvent
 - pdumodel::Controller_3_0_0, [111](#)
- stopActivity
 - webcam::StorageManager_1_0_1, [318](#)
- StorageStatus
 - webcam::StorageManager_1_0_1, [316](#)
- StorageType
 - webcam::StorageManager_1_0_1, [316](#)
- StripInfoChangedEvent
 - assetmgrmodel::AssetStrip_2_0_1, [77](#)
- StripSettingsChangedEvent
 - assetmgrmodel::AssetStripConfig_1_0_1, [81](#)
- StripType
 - assetmgrmodel::AssetStrip_2_0_1, [73](#)
- subscribe
 - event::Channel_1_0_1, [100](#)
- switchOnInProgress
 - pdumodel::Outlet_1_5_6::State, [309](#)
- sys, [55](#)
- sys::System, [323](#)
 - isDaemonRunning, [323](#)
 - restartDaemon, [324](#)
- TACACS_PLUS
 - auth, [36](#)
- TEMP_ABS
 - lhxmodel::Parameter_2_0_1, [226](#)
- TEMP_REL
 - lhxmodel::Parameter_2_0_1, [226](#)
- TEST_BUSY
 - test::Display_1_0_1, [129](#)
- TEST_FAILED
 - test::Display_1_0_1, [129](#)
- TEST_IDLE
 - test::Display_1_0_1, [129](#)
- TEST_PASSED
 - test::Display_1_0_1, [129](#)
- TIME
 - lhxmodel::Parameter_2_0_1, [227](#)
- TOP_CONNECTOR
 - assetmgrmodel::AssetStripConfig_1_0_1, [79](#)
- TOP_DOWN
 - assetmgrmodel::AssetStripConfig_1_0_1, [79](#)
- TRIGGER
 - event::Event, [141](#)
- TURNRATIO
 - pdumodel::Bcm, [86](#)
- TagEvent
 - assetmgrmodel::AssetStrip_2_0_1, [77](#)
- TagType
 - assetmgrmodel::AssetStrip_2_0_1, [74](#)
- TemperatureEnum
 - usermgmt, [58](#)
- test, [55](#)
- test::Display_1_0_1
 - FLIPPED, [129](#)
 - LEFT, [129](#)
 - NORMAL, [129](#)
 - RIGHT, [129](#)
 - TEST_BUSY, [129](#)
 - TEST_FAILED, [129](#)
 - TEST_IDLE, [129](#)
 - TEST_PASSED, [129](#)
- test::Ethernet
 - DPX_FULL, [140](#)
 - DPX_HALF, [140](#)
 - SPD_10, [140](#)
 - SPD_100, [140](#)
 - SPD_1000, [140](#)
- test::AuxSerial, [84](#)
 - getNumberOfPorts, [85](#)
 - testLoop, [85](#)
- test::Control, [108](#)
- test::Display_1_0_1, [128](#)
 - getInfo, [130](#)
 - getTestStatus, [130](#)
 - Orientation, [129](#)
 - testSequence, [130](#)
 - TestStatus, [129](#)
- test::Display_1_0_1::Info, [168](#)
- test::Ethernet, [139](#)
 - Duplex, [140](#)
 - setParameters, [140](#)
 - Speed, [140](#)
- test::FeatSerial, [145](#)
 - getNumberOfPorts, [146](#)
 - setPower, [146](#)
 - testLoopDtrDcd, [146](#)
 - testLoopTxRx, [147](#)

- test::RS232Serial, 260
- test::Result, 255
- test::Unit_1_0_2, 336
 - getButtonStates, 336
 - getDisplays, 336
 - setBuzzer, 337
 - triggerSlaveControllerWatchdog, 337
- testConfiguration
 - devsettings::Sntp_1_0_1, 304
- testLdapServer
 - auth::LdapManager_1_0_1, 179
- testLoop
 - test::AuxSerial, 85
- testLoopDtrDcd
 - test::FeatSerial, 146
- testLoopTxRx
 - test::FeatSerial, 147
- testRadiusServer
 - auth::RadiusManager, 248
- testSequence
 - test::Display_1_0_1, 130
- TestStatus
 - test::Display_1_0_1, 129
- ThresholdsChangedEvent
 - lhxmodel::Sensor_4_0_0, 271
 - sensors::NumericSensor_4_0_0, 214
- tolerance
 - sensors::NumericSensor_4_0_0::MetaData, 194
- touchCurrentSession
 - session::SessionManager, 294
- touchSession
 - session::SessionManager, 294
- traceRoute
 - net::Diagnostics, 128
- TransferReason
 - pdumodel::TransferSwitch_3_1_1, 333
- transferToSource
 - pdumodel::TransferSwitch_3_1_1, 335
- TransformerType
 - pdumodel::Bcm, 86
- triggerAction
 - event::Engine, 137
- triggerCapture
 - webcam::Channel, 96
- triggerPowercycle
 - assetmgrmodel::AssetStrip_2_0_1, 76
- triggerSlaveControllerWatchdog
 - test::Unit_1_0_2, 337
- Type
 - auth, 36
 - cascading::Cascading_1_0_1, 92
 - event::Engine::EventDesc, 142
 - event::Event, 141
 - pdumodel::Controller_3_0_0, 110
 - pdumodel::OverCurrentProtector_2_1_2, 222
 - pdumodel::TransferSwitch_3_1_1, 333
 - res_mon::Entry, 138
- TypeSpecChangedEvent
 - sensors::Sensor_4_0_0, 274
- UNCONNECTED
 - pdumodel::Bcm, 86
- UNKNOWN
 - bulkcfg::BulkConfiguration, 88
 - pdumodel::Controller_3_0_0, 110
 - serial::GsmModem_1_0_1, 159
- UNLOCKED
 - serial::GsmModem_1_0_1, 159
- UNREACHABLE
 - servermon::ServerMonitor_2_0_0, 282
- UNSUPPORTED
 - assetmgrmodel::AssetStrip_2_0_1, 73
- UPDATE_FAILED
 - assetmgrmodel::AssetStrip_2_0_1, 73
 - peripheral::DeviceManager_2_0_0, 119
- UPDATE_STARTED
 - assetmgrmodel::AssetStrip_2_0_1, 73
 - peripheral::DeviceManager_2_0_0, 119
- UPDATE_SUCCESSFUL
 - assetmgrmodel::AssetStrip_2_0_1, 73
 - peripheral::DeviceManager_2_0_0, 119
- UPLOAD_FAILED
 - bulkcfg::BulkConfiguration, 88
 - firmware, 41
- UPLOADING
 - firmware, 41
- USB_MULTI_IP
 - cascading::Cascading_1_0_1, 92
- USB_SINGLE_IP_NAT
 - cascading::Cascading_1_0_1, 92
- um, 56
- um::SnmpV3
 - AES128, 306
 - AUTH_NO_PRIV, 307
 - AUTH_PRIV, 307
 - DES, 306
 - MD5, 306
 - NO_AUTH_NO_PRIV, 307
 - SHA1, 306
- um::SnmpV3, 306
 - AuthProtocol, 306
 - PrivProtocol, 306
 - SecurityLevel, 306
- unassign
 - peripheral::DeviceSlot_2_0_0, 124
- Unit
 - lhxmodel::Parameter_2_0_1, 226
- UnknownDeviceAttachedEvent
 - peripheral::DeviceManager_2_0_0, 121
- unlockSimCard
 - serial::GsmModem_1_0_1, 161
- unstick
 - pdumodel::Outlet_1_5_6, 219
- unsubscribe
 - event::Channel_1_0_1, 100
- updateAccountFull
 - usermgmt::User_1_0_1, 344

- updateAvailable
 - firmware::Firmware_1_0_1, 149
- updateFirmware
 - peripheral::G2Production_2_0_0, 156
- UpdateFlags
 - firmware, 41
- updateFull
 - usermgmt::Role, 256
- UpdateHistoryStatus
 - firmware, 41
- usb, 56
- usb::Usb, 340
 - getDevices, 341
- usb::UsbDevice, 341
- UserEvent
 - event, 40
- usermgmt, 56
 - AccountEvent, 58
 - DEG_C, 58
 - DEG_F, 58
 - FEET, 57
 - LengthEnum, 57
 - METER, 57
 - PASCAL, 58
 - PSI, 58
 - PressureEnum, 57
 - TemperatureEnum, 58
- usermgmt::Account, 61
- usermgmt::AuxInfo, 84
- usermgmt::Preferences, 243
- usermgmt::Role, 256
 - getInfo, 256
 - updateFull, 256
- usermgmt::Role::Info, 166
- usermgmt::Role::Privilege, 243
- usermgmt::RoleManager, 258
 - createRoleFull, 259
 - deleteRole, 259
 - getAllPrivileges, 260
 - getAllRoleNames, 260
 - getAllRoles, 260
 - getInfo, 260
- usermgmt::RoleManager::ArgumentDesc, 70
- usermgmt::RoleManager::Info, 167
- usermgmt::RoleManager::PrivilegeDesc, 243
- usermgmt::RoleManager::RoleAccount, 258
- usermgmt::SnmPV3Settings, 307
- usermgmt::User_1_0_1, 341
 - getCapabilities, 343
 - getInfo, 343
 - getInfoAndPrivileges, 343
 - setAccountPassword, 343
 - setPreferences, 343
 - updateAccountFull, 344
- usermgmt::UserCapabilities, 344
- usermgmt::UserInfo, 345
- usermgmt::UserManager_1_0_2, 345
 - createAccount, 347
 - createAccountFull, 347
 - deleteAccount, 348
 - getAccountNames, 348
 - getAccountsByRole, 348
 - getAllAccounts, 348
 - getDefaultPreferences, 348
 - setDefaultPreferences, 348
- VERY_HIGH
 - webcam, 59
- VERY_LOW
 - webcam, 59
- VOLT
 - lhxmodel::Parameter_2_0_1, 227
- VOLTAGE
 - pdumodel::Bcm, 86
- ValueChangedEvent
 - lhxmodel::Parameter_2_0_1, 228
- WAITFORPIN
 - serial::GsmModem_1_0_1, 159
- WAITFORPUK
 - serial::GsmModem_1_0_1, 159
- WAITING
 - servermon::ServerMonitor_2_0_0, 282
- WATCHDOG
 - peripheral::G2Production_2_0_0, 155
- webcam, 58
 - HIGH, 59
 - JPEG, 59
 - LOW, 59
 - MJPEG, 59
 - NORMAL, 59
 - PixelFormat, 59
 - Priority, 59
 - RGB, 59
 - VERY_HIGH, 59
 - VERY_LOW, 59
 - WebcamEvent, 60
 - WebcamSettingsChangedEvent, 60
 - YUV, 59
- webcam::StorageManager_1_0_1
 - ASCENDING, 316
 - CIFS, 316
 - DESCENDING, 316
 - FTP, 316
 - INITIALIZING, 316
 - LOCAL, 316
 - NFS, 316
 - READY, 316
- webcam::Channel, 95
 - captureImage, 96
 - getClientType, 96
 - getWebcam, 96
 - isAvailable, 96
 - NO_ERROR, 97
 - triggerCapture, 96
- webcam::Controls, 111
- webcam::Format_2_0_0, 153

- webcam::Image_2_0_0, [163](#)
- webcam::ImageMetaData, [165](#)
- webcam::Information_2_0_0, [169](#)
- webcam::Location, [184](#)
- webcam::Settings_2_0_0, [302](#)
- webcam::StorageManager_1_0_1, [314](#)
 - addImage, [316](#)
 - Direction, [315](#)
 - getActivities, [316](#)
 - getImages, [316](#)
 - getInformation, [317](#)
 - getMetaData, [317](#)
 - getSettings, [317](#)
 - getSupportedStorageTypes, [317](#)
 - NO_ERROR, [319](#)
 - removeImages, [318](#)
 - setSettings, [318](#)
 - startActivity, [318](#)
 - stopActivity, [318](#)
 - StorageStatus, [316](#)
 - StorageType, [316](#)
- webcam::StorageManager_1_0_1::Activity, [63](#)
- webcam::StorageManager_1_0_1::ImageStorageMeta-Data, [166](#)
- webcam::StorageManager_1_0_1::StorageImage, [313](#)
- webcam::StorageManager_1_0_1::StorageInformation, [313](#)
- webcam::StorageManager_1_0_1::StorageMetaData, [319](#)
- webcam::StorageManager_1_0_1::StorageSettings, [319](#)
- webcam::StorageManager_1_0_1::WebcamStorage-Info, [354](#)
- webcam::Webcam_2_0_0, [350](#)
 - getControlDefaults, [350](#)
 - getInformation, [350](#)
 - getSettings, [351](#)
 - setControls, [351](#)
 - setSettings, [351](#)
- webcam::WebcamManager_2_0_0, [351](#)
 - getChannel, [352](#)
 - getChannels, [352](#)
 - getClientTypePriorities, [353](#)
 - getClientTypes, [353](#)
 - getWebcamPriorities, [353](#)
 - getWebcams, [353](#)
 - NO_ERROR, [354](#)
 - removeClientType, [353](#)
 - setClientTypePriorities, [354](#)
 - setWebcamPriorities, [354](#)
- WebcamEvent
 - webcam, [60](#)
- WebcamSettingsChangedEvent
 - webcam, [60](#)
- writeConfigurationSpace
 - peripheral::G2Production_2_0_0, [157](#)
- writeMemory
 - pduModel::MemoryMapController_3_0_0, [193](#)
- writeRegisterBits
 - peripheral::G2Production_2_0_0, [157](#)
- writeRegisters
 - peripheral::G2Production_2_0_0, [157](#)
- XOR
 - event::Engine::Condition, [103](#)
- YUV
 - webcam, [59](#)
- ZCoordMode
 - peripheral::DeviceManager_2_0_0, [119](#)