

ArnLib

4.0.x

Generated by Doxygen 1.8.14

Contents

- 1 README** **1**

- 2 ArnLib Changelog / Todo** **5**

- 3 General Description** **9**
 - 3.1 Arn Data Objects 9
 - 3.1.1 ArnItem access 9
 - 3.1.2 Modes 10
 - 3.1.3 Local 10
 - 3.1.4 Naming conventions 11
 - 3.1.5 Bidirectional Arn Data Objects 11
 - 3.1.6 Pipe Arn Data Objects 12
 - 3.1.6.1 Pipe sequence check 12
 - 3.1.6.2 Pipe anti congest 12
 - 3.1.7 Persistent Arn Data Objects 13
 - 3.1.7.1 Saving objects in files 13
 - 3.1.8 Sharing Arn Data Objects 14
 - 3.1.8.1 Dynamic port 14
 - 3.1.9 Sync rules 14
 - 3.1.9.1 Sync rules for Pipe 15
 - 3.1.9.2 ClientSyncMode 15
 - 3.2 RPC and SAPI 16
 - 3.2.1 RPC and SAPI method name overload 16
 - 3.2.2 RPC and SAPI communication format 17
 - 3.3 ZeroConfig 19
 - 3.3.1 ZeroConfig definitions 19
 - 3.3.1.1 Service name 20
 - 3.3.1.2 Sub types 20
 - 3.3.1.3 Text record 20
 - 3.3.2 Discover 20
 - 3.3.3 Discover remote 21
 - 3.4 Application notations 22

4	Installation and usage	23
4.1	Introduction	23
4.2	Documentation	23
4.3	Building ArnLib	24
4.3.1	A) Unix	24
4.3.2	B) Win32/MSVC	25
4.3.3	C) Win32/MinGW	25
4.3.4	D) MacOSX	26
4.3.5	E) Qt Embedded	26
4.4	Using ArnLib	26
5	ArnLib Internals	27
5.1	ScriptJobs	27
5.2	ArnMonitor	28
5.3	Destroy	28
6	Example Collection	31
6.1	Chat Demo	31
6.1.1	Chat Server	31
6.1.1.1	ChatSapi.hpp	31
6.1.1.2	MainWindow.hpp	32
6.1.1.3	MainWindow.cpp	32
6.1.1.4	main.cpp	34
6.1.2	Chat Client	34
6.1.2.1	MainWindow.hpp	34
6.1.2.2	MainWindow.cpp	35
6.1.2.3	main.cpp	36
6.1.3	Pictures	36
7	Help descriptions	37
7.1	Discover	37
7.1.1	Description	37

8	Deprecated List	39
9	Namespace Index	41
9.1	Namespace List	41
10	Hierarchical Index	43
10.1	Class Hierarchy	43
11	Class Index	47
11.1	Class List	47
12	File Index	51
12.1	File List	51
13	Namespace Documentation	53
13.1	Arn Namespace Reference	53
13.1.1	Function Documentation	55
13.1.1.1	_log2_u()	55
13.1.1.2	_log2_ull()	55
13.1.1.3	_mod_i()	56
13.1.1.4	_mod_ll()	56
13.1.1.5	addPath()	56
13.1.1.6	changeBasePath()	57
13.1.1.7	childPath()	57
13.1.1.8	circVal()	58
13.1.1.9	convertName()	58
13.1.1.10	convertPath()	58
13.1.1.11	fullPath()	59
13.1.1.12	hostFromHostWithInfo()	59
13.1.1.13	isFolderPath()	60
13.1.1.14	isNullPtr()	60
13.1.1.15	isPower2()	60
13.1.1.16	isProviderPath()	61

13.1.1.17	itemName()	61
13.1.1.18	log2()	61
13.1.1.19	makeHostWithInfo()	62
13.1.1.20	makePath()	62
13.1.1.21	maxLim()	63
13.1.1.22	minLim()	63
13.1.1.23	mod()	63
13.1.1.24	parentPath()	63
13.1.1.25	providerPathIf()	64
13.1.1.26	rand()	64
13.1.1.27	rangeLim()	65
13.1.1.28	twinPath()	65
13.1.1.29	uuidPath()	65
13.1.2	Variable Documentation	66
13.1.2.1	debugDepend	66
13.1.2.2	debugDiscover	66
13.1.2.3	debugLinkDestroy	66
13.1.2.4	debugLinkRef	66
13.1.2.5	debugMDNS	66
13.1.2.6	debugMonitor	67
13.1.2.7	debugMonitorTest	67
13.1.2.8	debugQmlNetwork	67
13.1.2.9	debugReclnOut	67
13.1.2.10	debugRPC	67
13.1.2.11	debugShareObj	67
13.1.2.12	debugSizes	68
13.1.2.13	debugThreading	68
13.1.2.14	debugZeroConf	68
13.1.2.15	defaultTcpPort	68
13.1.2.16	offHeartbeat	68
13.1.2.17	pathDiscover	68
13.1.2.18	pathDiscoverConnect	69
13.1.2.19	pathDiscoverThis	69
13.1.2.20	pathLocal	69
13.1.2.21	pathLocalSys	69
13.1.2.22	pathServer	69
13.1.2.23	pathServerSessions	69
13.1.2.24	resourceArnLib	70
13.1.2.25	resourceArnRoot	70
13.1.2.26	warningMDNS	70
13.2	ArnDiscover Namespace Reference	70
13.3	ArnZeroConf Namespace Reference	70

14 Class Documentation	71
14.1 Arn::_InitEnumTxt Struct Reference	71
14.1.1 Detailed Description	71
14.1.2 Member Data Documentation	71
14.1.2.1 enumTxt	71
14.1.2.2 enumVal	71
14.1.2.3 ns	72
14.2 Arn::_InitSubEnum Struct Reference	72
14.2.1 Detailed Description	72
14.2.2 Member Data Documentation	72
14.2.2.1 enumTxtClass	73
14.2.2.2 factor	73
14.2.2.3 mask	73
14.3 Arn::Allow Class Reference	73
14.3.1 Detailed Description	73
14.3.2 Member Enumeration Documentation	73
14.3.2.1 E	73
14.4 ArnAdaptItem Class Reference	74
14.4.1 Detailed Description	78
14.4.2 Member Typedef Documentation	79
14.4.2.1 ArnEventCB	79
14.4.2.2 ChangedCB	79
14.4.2.3 LinkDestroyedCB	79
14.4.3 Constructor & Destructor Documentation	79
14.4.3.1 ArnAdaptItem()	80
14.4.3.2 ~ArnAdaptItem()	80
14.4.4 Member Function Documentation	80
14.4.4.1 addMode()	80
14.4.4.2 addValue() [1/2]	80
14.4.4.3 addValue() [2/2]	81

14.4.4.4	arnEventCallback()	81
14.4.4.5	arnExport()	82
14.4.4.6	arnImport()	82
14.4.4.7	ChangedCallback()	82
14.4.4.8	close()	83
14.4.4.9	destroyLink()	83
14.4.4.10	destroyLinkLocal()	83
14.4.4.11	getMode()	84
14.4.4.12	isAutoDestroy()	84
14.4.4.13	isBiDirMode()	84
14.4.4.14	isFolder()	85
14.4.4.15	isIgnoreSameValue()	85
14.4.4.16	isMaster()	85
14.4.4.17	isOpen()	85
14.4.4.18	isPipeMode()	86
14.4.4.19	isProvider()	86
14.4.4.20	isSaveMode()	86
14.4.4.21	isUncrossed()	87
14.4.4.22	itemId()	87
14.4.4.23	linkDestroyedCallback()	88
14.4.4.24	linkId()	88
14.4.4.25	mutex()	88
14.4.4.26	name()	88
14.4.4.27	open()	89
14.4.4.28	operator+=() [1/2]	89
14.4.4.29	operator+=() [2/2]	89
14.4.4.30	operator=() [1/10]	90
14.4.4.31	operator=() [2/10]	90
14.4.4.32	operator=() [3/10]	90
14.4.4.33	operator=() [4/10]	90

14.4.4.34 operator=() [5/10]	90
14.4.4.35 operator=() [6/10]	90
14.4.4.36 operator=() [7/10]	91
14.4.4.37 operator=() [8/10]	91
14.4.4.38 operator=() [9/10]	91
14.4.4.39 operator=() [10/10]	91
14.4.4.40 path()	91
14.4.4.41 refCount()	92
14.4.4.42 reference()	92
14.4.4.43 setArnEventCallback()	92
14.4.4.44 setAutoDestroy()	93
14.4.4.45 setBiDirMode()	93
14.4.4.46 setBits()	93
14.4.4.47 setChangedCallback()	94
14.4.4.48 setIgnoreSameValue()	94
14.4.4.49 setLinkDestroyedCallback()	95
14.4.4.50 setMaster()	95
14.4.4.51 setPipeMode()	96
14.4.4.52 setReference()	96
14.4.4.53 setSaveMode()	96
14.4.4.54 setUncrossed()	97
14.4.4.55 setValue() [1/11]	97
14.4.4.56 setValue() [2/11]	97
14.4.4.57 setValue() [3/11]	98
14.4.4.58 setValue() [4/11]	98
14.4.4.59 setValue() [5/11]	99
14.4.4.60 setValue() [6/11]	99
14.4.4.61 setValue() [7/11]	99
14.4.4.62 setValue() [8/11]	100
14.4.4.63 setValue() [9/11]	100

14.4.4.64 setValue() [10/11]	101
14.4.4.65 setValue() [11/11]	101
14.4.4.66 syncMode()	102
14.4.4.67 thread()	102
14.4.4.68 toBool()	102
14.4.4.69 toByteArray()	103
14.4.4.70 toDouble()	103
14.4.4.71 toInt()	104
14.4.4.72 toInt64()	104
14.4.4.73 toReal()	104
14.4.4.74 toString()	105
14.4.4.75 toUInt()	105
14.4.4.76 toUInt64()	106
14.4.4.77 toVariant()	106
14.4.4.78 type()	106
14.5 ArnAtomicOp Class Reference	107
14.5.1 Detailed Description	107
14.5.2 Member Enumeration Documentation	107
14.5.2.1 E	107
14.5.2.2 NS	107
14.6 ArnBasicItem Class Reference	108
14.6.1 Detailed Description	111
14.6.2 Constructor & Destructor Documentation	112
14.6.2.1 ArnBasicItem()	112
14.6.2.2 ~ArnBasicItem()	112
14.6.3 Member Function Documentation	112
14.6.3.1 addMode()	112
14.6.3.2 addValue() [1/2]	113
14.6.3.3 addValue() [2/2]	113
14.6.3.4 arnExport()	114

14.6.3.5	<code>arnImport()</code>	114
14.6.3.6	<code>close()</code>	115
14.6.3.7	<code>destroyLink()</code>	115
14.6.3.8	<code>destroyLinkLocal()</code>	115
14.6.3.9	<code>eventHandler()</code>	116
14.6.3.10	<code>getMode()</code>	116
14.6.3.11	<code>isAssigning()</code>	116
14.6.3.12	<code>isAtomicOpProvider()</code>	117
14.6.3.13	<code>isAutoDestroy()</code>	117
14.6.3.14	<code>isBiDirMode()</code>	117
14.6.3.15	<code>isFolder()</code>	118
14.6.3.16	<code>isIgnoreSameValue()</code>	118
14.6.3.17	<code>isMaster()</code>	118
14.6.3.18	<code>isOpen()</code>	119
14.6.3.19	<code>isPipeMode()</code>	119
14.6.3.20	<code>isProvider()</code>	119
14.6.3.21	<code>isSaveMode()</code>	120
14.6.3.22	<code>isUncrossed()</code>	120
14.6.3.23	<code>itemId()</code>	121
14.6.3.24	<code>linkId()</code>	121
14.6.3.25	<code>name()</code>	121
14.6.3.26	<code>open()</code>	122
14.6.3.27	<code>operator+=()</code> [1/2]	122
14.6.3.28	<code>operator+=()</code> [2/2]	122
14.6.3.29	<code>operator=()</code> [1/10]	123
14.6.3.30	<code>operator=()</code> [2/10]	123
14.6.3.31	<code>operator=()</code> [3/10]	123
14.6.3.32	<code>operator=()</code> [4/10]	123
14.6.3.33	<code>operator=()</code> [5/10]	123
14.6.3.34	<code>operator=()</code> [6/10]	123

14.6.3.35 operator=() [7/10]	124
14.6.3.36 operator=() [8/10]	124
14.6.3.37 operator=() [9/10]	124
14.6.3.38 operator=() [10/10]	124
14.6.3.39 path()	124
14.6.3.40 refCount()	125
14.6.3.41 reference()	125
14.6.3.42 setAtomicOpProvider()	125
14.6.3.43 setAutoDestroy()	126
14.6.3.44 setBiDirMode()	126
14.6.3.45 setBits()	126
14.6.3.46 setEventHandler()	127
14.6.3.47 setIgnoreSameValue()	127
14.6.3.48 setMaster()	127
14.6.3.49 setPipeMode()	128
14.6.3.50 setReference()	128
14.6.3.51 setSaveMode()	129
14.6.3.52 setUncrossed()	129
14.6.3.53 setValue() [1/11]	129
14.6.3.54 setValue() [2/11]	129
14.6.3.55 setValue() [3/11]	130
14.6.3.56 setValue() [4/11]	130
14.6.3.57 setValue() [5/11]	131
14.6.3.58 setValue() [6/11]	131
14.6.3.59 setValue() [7/11]	132
14.6.3.60 setValue() [8/11]	132
14.6.3.61 setValue() [9/11]	132
14.6.3.62 setValue() [10/11]	133
14.6.3.63 setValue() [11/11]	133
14.6.3.64 syncMode()	134

14.6.3.65	thread()	134
14.6.3.66	toBool()	134
14.6.3.67	toByteArray()	135
14.6.3.68	toDouble()	135
14.6.3.69	toInt()	136
14.6.3.70	toInt64()	136
14.6.3.71	toReal()	136
14.6.3.72	toString()	137
14.6.3.73	toUInt()	137
14.6.3.74	toUInt64()	138
14.6.3.75	toVariant()	138
14.6.3.76	type()	138
14.6.4	Friends And Related Function Documentation	139
14.6.4.1	ArnBasicItemEventHandler	139
14.7	ArnClient Class Reference	139
14.7.1	Detailed Description	142
14.7.2	Member Typedef Documentation	142
14.7.2.1	ConnectStat	142
14.7.2.2	HostList	142
14.7.2.3	SyncMode	142
14.7.3	Constructor & Destructor Documentation	143
14.7.3.1	ArnClient()	143
14.7.3.2	~ArnClient()	143
14.7.4	Member Function Documentation	143
14.7.4.1	abortKillRequest()	143
14.7.4.2	addMountPoint()	143
14.7.4.3	addToArnList()	144
14.7.4.4	arnList()	144
14.7.4.5	chatReceived	145
14.7.4.6	chatSend()	145

14.7.4.7	<code>clearArnList()</code>	146
14.7.4.8	<code>close()</code>	146
14.7.4.9	<code>connectionStatusChanged</code>	146
14.7.4.10	<code>connectStatus()</code>	147
14.7.4.11	<code>connectToArn()</code>	147
14.7.4.12	<code>connectToArnList()</code>	148
14.7.4.13	<code>disconnectFromArn()</code>	148
14.7.4.14	<code>freePaths()</code>	148
14.7.4.15	<code>getClient()</code>	149
14.7.4.16	<code>getTraffic()</code>	150
14.7.4.17	<code>id()</code>	150
14.7.4.18	<code>isDemandLogin()</code>	151
14.7.4.19	<code>isReConnect()</code>	151
14.7.4.20	<code>isReContact()</code>	151
14.7.4.21	<code>killRequested</code>	152
14.7.4.22	<code>loginRequired</code>	152
14.7.4.23	<code>loginToArn()</code>	153
14.7.4.24	<code>loginToArnHashed()</code>	153
14.7.4.25	<code>passwordHash()</code>	154
14.7.4.26	<code>receiveTimeout()</code>	154
14.7.4.27	<code>registerClient()</code>	154
14.7.4.28	<code>remoteWhoIam()</code>	155
14.7.4.29	<code>removeMountPoint()</code>	155
14.7.4.30	<code>setAutoConnect()</code>	156
14.7.4.31	<code>setDemandLogin()</code>	156
14.7.4.32	<code>setMountPoint()</code>	157
14.7.4.33	<code>setReceiveTimeout()</code>	157
14.7.4.34	<code>setSyncMode()</code>	158
14.7.4.35	<code>setWhoIam()</code>	158
14.7.4.36	<code>syncMode()</code>	159

14.7.4.37 tcpConnected	159
14.7.4.38 tcpDisConnected	160
14.7.4.39 tcpError	160
14.8 ArnClientConnectStat Class Reference	160
14.8.1 Detailed Description	160
14.8.2 Member Enumeration Documentation	160
14.8.2.1 E	160
14.8.2.2 NS	161
14.9 ArnClientReg Class Reference	161
14.9.1 Detailed Description	161
14.9.2 Member Function Documentation	162
14.9.2.1 get()	162
14.9.2.2 instance()	162
14.9.2.3 remove() [1/2]	162
14.9.2.4 remove() [2/2]	162
14.9.2.5 store()	162
14.10 ArnCoreItem Class Reference	163
14.10.1 Detailed Description	164
14.10.2 Constructor & Destructor Documentation	164
14.10.2.1 ArnCoreItem()	164
14.10.2.2 ~ArnCoreItem()	164
14.10.3 Member Function Documentation	164
14.10.3.1 thread()	164
14.10.4 Friends And Related Function Documentation	165
14.10.4.1 ArnBasicItemEventHandler	165
14.11 ArnDepend Class Reference	165
14.11.1 Detailed Description	166
14.11.2 Member Typedef Documentation	167
14.11.2.1 DepSlot	167
14.11.3 Constructor & Destructor Documentation	167

14.11.3.1	ArnDepend()	167
14.11.3.2	~ArnDepend()	167
14.11.4	Member Function Documentation	167
14.11.4.1	add() [1/2]	167
14.11.4.2	add() [2/2]	168
14.11.4.3	completed	168
14.11.4.4	setMonitorName()	168
14.11.4.5	startMonitor()	168
14.12	ArnDependOffer Class Reference	169
14.12.1	Detailed Description	170
14.12.2	Constructor & Destructor Documentation	170
14.12.2.1	ArnDependOffer()	170
14.12.2.2	~ArnDependOffer()	170
14.12.3	Member Function Documentation	171
14.12.3.1	advertise()	171
14.12.3.2	setStateId()	172
14.12.3.3	setStateName()	172
14.12.3.4	stateId()	172
14.12.3.5	stateName()	173
14.13	ArnDiscoverAdvertise Class Reference	173
14.13.1	Detailed Description	175
14.13.2	Constructor & Destructor Documentation	175
14.13.2.1	ArnDiscoverAdvertise()	175
14.13.2.2	~ArnDiscoverAdvertise()	176
14.13.3	Member Function Documentation	176
14.13.3.1	addCustomProperty()	176
14.13.3.2	addGroup()	176
14.13.3.3	advertiseService()	177
14.13.3.4	currentService()	177
14.13.3.5	customProperties()	178

14.13.3.6 groups()	178
14.13.3.7 service()	179
14.13.3.8 serviceChanged	179
14.13.3.9 serviceChangeError	179
14.13.3.10setCustomProperties()	180
14.13.3.11setGroups()	180
14.13.3.12setService	181
14.13.3.13state()	182
14.14ArnDiscoverBrowser Class Reference	182
14.14.1 Detailed Description	183
14.14.2 Constructor & Destructor Documentation	184
14.14.2.1 ArnDiscoverBrowser()	184
14.14.3 Member Function Documentation	184
14.14.3.1 browse	184
14.14.3.2 isBrowsing()	185
14.14.3.3 setFilter() [1/2]	185
14.14.3.4 setFilter() [2/2]	186
14.14.3.5 stopBrowse	186
14.15ArnDiscoverBrowserB Class Reference	187
14.15.1 Detailed Description	188
14.15.2 Constructor & Destructor Documentation	188
14.15.2.1 ArnDiscoverBrowserB()	188
14.15.2.2 ~ArnDiscoverBrowserB()	188
14.15.3 Member Function Documentation	189
14.15.3.1 defaultStopState()	189
14.15.3.2 goTowardState()	189
14.15.3.3 IdToIndex()	190
14.15.3.4 indexTold()	190
14.15.3.5 infoById()	191
14.15.3.6 infoByIndex()	191

14.15.3.7 infoByName()	192
14.15.3.8 infoUpdated	192
14.15.3.9 serviceAdded	192
14.15.3.10serviceCount()	193
14.15.3.11serviceNameTold()	193
14.15.3.12serviceRemoved	194
14.15.3.13setDefaultStopState()	194
14.16ArnDiscoverConnector Class Reference	195
14.16.1 Detailed Description	196
14.16.2 Constructor & Destructor Documentation	197
14.16.2.1 ArnDiscoverConnector()	197
14.16.2.2 ~ArnDiscoverConnector()	197
14.16.3 Member Function Documentation	197
14.16.3.1 addToDirectHosts()	197
14.16.3.2 clearDirectHosts()	198
14.16.3.3 clientReadyToConnect	198
14.16.3.4 directHostPrio()	199
14.16.3.5 discoverHostPrio()	199
14.16.3.6 externalClientConnect()	200
14.16.3.7 id()	200
14.16.3.8 resolveRefreshTimeout()	200
14.16.3.9 service()	201
14.16.3.10setDirectHostPrio()	201
14.16.3.11setDiscoverHostPrio()	201
14.16.3.12setExternalClientConnect()	202
14.16.3.13setResolver()	202
14.16.3.14setResolveRefreshTimeout()	203
14.16.3.15setService	203
14.16.3.16start()	204
14.17ArnDiscoverInfo Class Reference	204

14.17.1 Detailed Description	205
14.17.2 Constructor & Destructor Documentation	206
14.17.2.1 ArnDiscoverInfo() [1/2]	206
14.17.2.2 ArnDiscoverInfo() [2/2]	206
14.17.2.3 ~ArnDiscoverInfo()	206
14.17.3 Member Function Documentation	206
14.17.3.1 domain()	206
14.17.3.2 groups()	207
14.17.3.3 hostIp()	207
14.17.3.4 hostIpString()	207
14.17.3.5 hostName()	208
14.17.3.6 hostPort()	208
14.17.3.7 hostPortString()	208
14.17.3.8 hostWithInfo()	209
14.17.3.9 inProgress()	209
14.17.3.10sError()	209
14.17.3.11operator=()	210
14.17.3.12properties()	210
14.17.3.13resolvCode()	210
14.17.3.14serviceName()	211
14.17.3.15state()	211
14.17.3.16stopState()	211
14.17.3.17type()	212
14.17.3.18typeString()	212
14.17.4 Friends And Related Function Documentation	212
14.17.4.1 ArnDiscoverBrowserB	212
14.18 ArnDiscoverRemote Class Reference	213
14.18.1 Detailed Description	214
14.18.2 Constructor & Destructor Documentation	215
14.18.2.1 ArnDiscoverRemote()	215

14.18.2.2 ~ArnDiscoverRemote()	215
14.18.3 Member Function Documentation	215
14.18.3.1 clientReadyToConnect	215
14.18.3.2 defaultService()	216
14.18.3.3 initialServiceTimeout()	216
14.18.3.4 newConnector()	216
14.18.3.5 setDefaultService()	217
14.18.3.6 setInitialServiceTimeout()	217
14.18.3.7 setService	218
14.18.3.8 startUseNewServer()	218
14.18.3.9 startUseServer()	219
14.19 ArnDiscoverResolver Class Reference	219
14.19.1 Detailed Description	221
14.19.2 Constructor & Destructor Documentation	221
14.19.2.1 ArnDiscoverResolver()	221
14.19.3 Member Function Documentation	221
14.19.3.1 defaultService()	222
14.19.3.2 resolve	222
14.19.3.3 setDefaultService()	223
14.20 ArnError Class Reference	223
14.20.1 Detailed Description	223
14.20.2 Member Enumeration Documentation	224
14.20.2.1 E	224
14.21 ArnEvAtomicOp Class Reference	224
14.21.1 Detailed Description	226
14.21.2 Member Typedef Documentation	226
14.21.2.1 Op	226
14.21.3 Constructor & Destructor Documentation	226
14.21.3.1 ArnEvAtomicOp()	226
14.21.3.2 ~ArnEvAtomicOp()	226

14.21.4 Member Function Documentation	226
14.21.4.1 arg1()	227
14.21.4.2 arg2()	227
14.21.4.3 makeHeapClone()	227
14.21.4.4 op()	227
14.21.4.5 type()	227
14.22 ArnEvent Class Reference	228
14.22.1 Detailed Description	229
14.22.2 Member Typedef Documentation	229
14.22.2.1 Idx	229
14.22.3 Constructor & Destructor Documentation	229
14.22.3.1 ArnEvent()	230
14.22.3.2 ~ArnEvent()	230
14.22.4 Member Function Documentation	230
14.22.4.1 baseType()	230
14.22.4.2 copyOpt()	230
14.22.4.3 inhibitPendingChain()	230
14.22.4.4 isArnEvent()	231
14.22.4.5 makeHeapClone()	231
14.22.4.6 setTarget()	231
14.22.4.7 setTargetMutex()	231
14.22.4.8 setTargetPendingChain()	231
14.22.4.9 target()	231
14.22.4.10 toIdx() [1/2]	232
14.22.4.11 toIdx() [2/2]	232
14.22.4.12 toString() [1/2]	232
14.22.4.13 toString() [2/2]	232
14.23 ArnEventIdx Class Reference	232
14.23.1 Detailed Description	233
14.23.2 Member Enumeration Documentation	233

14.23.2.1 E	233
14.24 ArnEvLinkCreate Class Reference	234
14.24.1 Detailed Description	235
14.24.2 Constructor & Destructor Documentation	235
14.24.2.1 ArnEvLinkCreate()	235
14.24.3 Member Function Documentation	236
14.24.3.1 arnLink()	236
14.24.3.2 isLastLink()	236
14.24.3.3 makeHeapClone()	236
14.24.3.4 path()	236
14.24.3.5 type()	236
14.25 ArnEvModeChange Class Reference	237
14.25.1 Detailed Description	238
14.25.2 Constructor & Destructor Documentation	238
14.25.2.1 ArnEvModeChange()	238
14.25.3 Member Function Documentation	238
14.25.3.1 linkId()	238
14.25.3.2 makeHeapClone()	238
14.25.3.3 mode()	239
14.25.3.4 path()	239
14.25.3.5 type()	239
14.26 ArnEvMonitor Class Reference	239
14.26.1 Detailed Description	240
14.26.2 Constructor & Destructor Documentation	240
14.26.2.1 ArnEvMonitor()	240
14.26.3 Member Function Documentation	241
14.26.3.1 data()	241
14.26.3.2 isLocal()	241
14.26.3.3 makeHeapClone()	241
14.26.3.4 monEvType()	241

14.26.3.5 sessionHandler()	241
14.26.3.6 type()	242
14.27 ArnEvRefChange Class Reference	242
14.27.1 Detailed Description	243
14.27.2 Constructor & Destructor Documentation	243
14.27.2.1 ArnEvRefChange()	243
14.27.2.2 ~ArnEvRefChange()	243
14.27.3 Member Function Documentation	243
14.27.3.1 makeHeapClone()	243
14.27.3.2 refStep()	244
14.27.3.3 type()	244
14.28 ArnEvRetired Class Reference	244
14.28.1 Detailed Description	245
14.28.2 Constructor & Destructor Documentation	245
14.28.2.1 ArnEvRetired()	245
14.28.3 Member Function Documentation	246
14.28.3.1 isBelow()	246
14.28.3.2 isGlobal()	246
14.28.3.3 makeHeapClone()	246
14.28.3.4 startLink()	246
14.28.3.5 type()	246
14.29 ArnEvValueChange Class Reference	247
14.29.1 Detailed Description	248
14.29.2 Constructor & Destructor Documentation	248
14.29.2.1 ArnEvValueChange()	248
14.29.2.2 ~ArnEvValueChange()	248
14.29.3 Member Function Documentation	248
14.29.3.1 handleData()	248
14.29.3.2 makeHeapClone()	249
14.29.3.3 sendId()	249

14.29.3.4 type()	249
14.29.3.5 valueData()	249
14.30 ArnEvZeroRef Class Reference	250
14.30.1 Detailed Description	251
14.30.2 Constructor & Destructor Documentation	251
14.30.2.1 ArnEvZeroRef()	251
14.30.3 Member Function Documentation	251
14.30.3.1 arnLink()	251
14.30.3.2 makeHeapClone()	251
14.30.3.3 type()	252
14.31 ArnInterface Class Reference	252
14.31.1 Detailed Description	254
14.31.2 Member Enumeration Documentation	254
14.31.2.1 DataType	254
14.31.2.2 NameF	255
14.31.2.3 ObjectMode	255
14.31.2.4 SameValue	255
14.31.3 Member Function Documentation	256
14.31.3.1 bytes	256
14.31.3.2 changeBasePath	256
14.31.3.3 childPath	256
14.31.3.4 exist	256
14.31.3.5 intNum	257
14.31.3.6 isFolder	257
14.31.3.7 isFolderPath	257
14.31.3.8 isLeaf	257
14.31.3.9 isProviderPath	257
14.31.3.10 itemName	258
14.31.3.11 items	258
14.31.3.12 makePath	258

14.31.3.13num	258
14.31.3.14providerPath	258
14.31.3.15setBytes	259
14.31.3.16setIntNum	259
14.31.3.17setNum	259
14.31.3.18setString	259
14.31.3.19setValue	260
14.31.3.20setVariant	260
14.31.3.21string	260
14.31.3.22winPath	260
14.31.3.23value	261
14.31.3.24variant	261
14.31.4 Property Documentation	261
14.31.4.1 info	261
14.32ArnItem Class Reference	262
14.32.1 Detailed Description	266
14.32.2 Constructor & Destructor Documentation	266
14.32.2.1 ArnItem() [1/3]	266
14.32.2.2 ArnItem() [2/3]	267
14.32.2.3 ArnItem() [3/3]	267
14.32.2.4 ~ArnItem()	268
14.32.3 Member Function Documentation	268
14.32.3.1 addMode()	268
14.32.3.2 addValue() [1/2]	268
14.32.3.3 addValue() [2/2]	269
14.32.3.4 arnExport()	269
14.32.3.5 arnImport()	269
14.32.3.6 arnItemCreated	270
14.32.3.7 arnModeChanged	270
14.32.3.8 bypassDelayPending()	271

14.32.3.9	changed [1/7]	271
14.32.3.10	changed [2/7]	271
14.32.3.11	changed [3/7]	272
14.32.3.12	changed [4/7]	272
14.32.3.13	changed [5/7]	272
14.32.3.14	changed [6/7]	272
14.32.3.15	changed [7/7]	273
14.32.3.16	delay()	273
14.32.3.17	getMode()	273
14.32.3.18	isAutoDestroy()	273
14.32.3.19	isBiDirMode()	274
14.32.3.20	isDelayPending()	274
14.32.3.21	isFolder()	275
14.32.3.22	isIgnoreSameValue()	275
14.32.3.23	isMaster()	275
14.32.3.24	isPipeMode()	275
14.32.3.25	isProvider()	276
14.32.3.26	isSaveMode()	276
14.32.3.27	isTemplate()	277
14.32.3.28	isUncrossed()	277
14.32.3.29	modeChanged	277
14.32.3.30	openFolder()	278
14.32.3.31	openUid()	278
14.32.3.32	openUidPipe()	278
14.32.3.33	operator+=() [1/2]	279
14.32.3.34	operator+=() [2/2]	279
14.32.3.35	operator=() [1/10]	279
14.32.3.36	operator=() [2/10]	279
14.32.3.37	operator=() [3/10]	280
14.32.3.38	operator=() [4/10]	280

14.32.3.39operator=() [5/10]	280
14.32.3.40operator=() [6/10]	280
14.32.3.41operator=() [7/10]	280
14.32.3.42operator=() [8/10]	280
14.32.3.43operator=() [9/10]	281
14.32.3.44operator=() [10/10]	281
14.32.3.45setAutoDestroy()	281
14.32.3.46setBiDirMode()	281
14.32.3.47setBits()	281
14.32.3.48setBlockEcho()	282
14.32.3.49setDelay()	282
14.32.3.50setIgnoreSameValue()	283
14.32.3.51setMaster()	283
14.32.3.52setPipeMode()	284
14.32.3.53setSaveMode()	284
14.32.3.54setTemplate()	284
14.32.3.55setUncrossed()	285
14.32.3.56setValue() [1/18]	285
14.32.3.57setValue() [2/18]	286
14.32.3.58setValue() [3/18]	286
14.32.3.59setValue() [4/18]	287
14.32.3.60setValue() [5/18]	287
14.32.3.61setValue() [6/18]	287
14.32.3.62setValue() [7/18]	289
14.32.3.63setValue() [8/18]	289
14.32.3.64setValue() [9/18]	290
14.32.3.65setValue() [10/18]	290
14.32.3.66setValue() [11/18]	291
14.32.3.67setValue [12/18]	291
14.32.3.68setValue [13/18]	291

14.32.3.69	setValue [14/18]	292
14.32.3.70	setValue [15/18]	292
14.32.3.71	setValue [16/18]	293
14.32.3.72	setValue [17/18]	293
14.32.3.73	setValue [18/18]	293
14.32.3.74	syncMode()	294
14.32.3.75	toBool()	294
14.32.3.76	toByteArray()	294
14.32.3.77	toDouble()	295
14.32.3.78	toggleBool	295
14.32.3.79	toInt()	295
14.32.3.80	toInt64()	296
14.32.3.81	toReal()	296
14.32.3.82	toString()	297
14.32.3.83	toUInt()	297
14.32.3.84	toUInt64()	297
14.32.3.85	toVariant()	298
14.32.3.86	type()	298
14.33	ArnItemB Class Reference	299
14.33.1	Detailed Description	300
14.33.2	Constructor & Destructor Documentation	300
14.33.2.1	ArnItemB()	300
14.33.2.2	~ArnItemB()	301
14.33.3	Member Function Documentation	301
14.33.3.1	arnLinkDestroyed	301
14.33.3.2	open()	301
14.34	ArnItemQml Class Reference	302
14.34.1	Detailed Description	304
14.34.2	Member Function Documentation	305
14.34.2.1	addIntNum	305

14.34.2.2 addMode	305
14.34.2.3 addNum	306
14.34.2.4 getMode	306
14.34.2.5 setBits	306
14.34.3 Property Documentation	306
14.34.3.1 atomicOpProvider	307
14.34.3.2 autoDestroyMode	307
14.34.3.3 biDirMode	307
14.34.3.4 bytes	307
14.34.3.5 delay	307
14.34.3.6 ignoreSameValue	308
14.34.3.7 intNum	308
14.34.3.8 masterMode	308
14.34.3.9 num	308
14.34.3.10path	308
14.34.3.11pipeMode	309
14.34.3.12saveMode	309
14.34.3.13string	309
14.34.3.14type	309
14.34.3.15useUuid	309
14.34.3.16variant	310
14.34.3.17variantType	310
14.35ArnItemValve Class Reference	310
14.35.1 Detailed Description	312
14.35.2 Constructor & Destructor Documentation	312
14.35.2.1 ArnItemValve()	312
14.35.3 Member Function Documentation	312
14.35.3.1 changed	312
14.35.3.2 isAutoDestroy()	312
14.35.3.3 isMaster()	313

14.35.3.4 isSaveMode()	313
14.35.3.5 operator=()	314
14.35.3.6 setAutoDestroy()	314
14.35.3.7 setMaster()	314
14.35.3.8 setSaveMode()	315
14.35.3.9 setTarget()	315
14.35.3.10 setValue	315
14.35.3.11 switchMode()	315
14.35.3.12 toBool()	316
14.36 ArnLinkValue Struct Reference	316
14.36.1 Detailed Description	316
14.36.2 Constructor & Destructor Documentation	316
14.36.2.1 ArnLinkValue()	316
14.36.3 Member Data Documentation	317
14.36.3.1 localUpdateCount	317
14.36.3.2 valueByteArray	317
14.36.3.3 valueInt	317
14.36.3.4 valueReal	317
14.36.3.5 valueString	317
14.36.3.6 valueVariant	318
14.37 ArnM Class Reference	318
14.37.1 Detailed Description	320
14.37.2 Member Function Documentation	320
14.37.2.1 defaultIgnoreSameValue()	320
14.37.2.2 destroyLink	321
14.37.2.3 destroyLinkLocal()	321
14.37.2.4 errorLog()	321
14.37.2.5 errorLogSig	322
14.37.2.6 errorSysName()	322
14.37.2.7 exist()	322

14.37.2.8 info()	322
14.37.2.9 instance()	323
14.37.2.10sAtomicOpProvider()	323
14.37.2.11isFolder()	323
14.37.2.12sLeaf()	324
14.37.2.13sMainThread()	324
14.37.2.14sThreadedApp()	324
14.37.2.15tems()	325
14.37.2.16oadFromDirRoot()	325
14.37.2.17oadFromFile()	326
14.37.2.18saveToFile()	326
14.37.2.19setAtomicOpProvider()	326
14.37.2.20setConsoleError()	327
14.37.2.21setDefaultIgnoreSameValue()	327
14.37.2.22setSkipLocalSysLoading()	327
14.37.2.23setupErrorlog	328
14.37.2.24setValue() [1/6]	328
14.37.2.25setValue() [2/6]	328
14.37.2.26setValue() [3/6]	329
14.37.2.27setValue() [4/6]	329
14.37.2.28setValue() [5/6]	329
14.37.2.29setValue() [6/6]	330
14.37.2.30skipLocalSysLoading()	330
14.37.2.31valueByteArray()	330
14.37.2.32valueDouble()	331
14.37.2.33valueInt()	331
14.37.2.34valueReal()	332
14.37.2.35valueString()	332
14.37.2.36valueVariant()	332
14.37.3 Friends And Related Function Documentation	333

14.37.3.1 ArnBasicItem	333
14.38 ArnMonEventType Class Reference	333
14.38.1 Detailed Description	333
14.38.2 Member Enumeration Documentation	333
14.38.2.1 E	333
14.38.2.2 NS	334
14.39 ArnMonitor Class Reference	334
14.39.1 Detailed Description	337
14.39.2 Constructor & Destructor Documentation	337
14.39.2.1 ArnMonitor() [1/2]	337
14.39.2.2 ArnMonitor() [2/2]	337
14.39.2.3 ~ArnMonitor()	338
14.39.3 Member Function Documentation	338
14.39.3.1 arnChildDeleted	338
14.39.3.2 arnChildFound	338
14.39.3.3 arnChildFoundFolder	339
14.39.3.4 arnChildFoundLeaf	339
14.39.3.5 arnChildModeChanged	340
14.39.3.6 arnItemCreated	340
14.39.3.7 arnItemDeleted	341
14.39.3.8 arnItemModeChanged	341
14.39.3.9 client()	341
14.39.3.10 clientId()	342
14.39.3.11 foundChildDeleted	342
14.39.3.12 monitorClosed	342
14.39.3.13 monitorPath()	343
14.39.3.14 reference()	343
14.39.3.15 reStart()	343
14.39.3.16 setClient() [1/2]	343
14.39.3.17 setClient() [2/2]	344

14.39.3.18	setMonitorPath()	344
14.39.3.19	setReference()	345
14.39.3.20	start() [1/2]	345
14.39.3.21	start() [2/2]	345
14.40	ArnMonitorQml Class Reference	346
14.40.1	Detailed Description	347
14.40.2	Member Function Documentation	348
14.40.2.1	reStart	348
14.40.3	Property Documentation	348
14.40.3.1	clientId	348
14.40.3.2	monitorPath	349
14.41	ArnNullptr Struct Reference	349
14.41.1	Detailed Description	349
14.41.2	Member Function Documentation	349
14.41.2.1	operator T*()	349
14.42	ArnPersist Class Reference	350
14.42.1	Detailed Description	351
14.42.2	Constructor & Destructor Documentation	351
14.42.2.1	ArnPersist()	351
14.42.2.2	~ArnPersist()	351
14.42.3	Member Function Documentation	352
14.42.3.1	doArchive	352
14.42.3.2	flush()	352
14.42.3.3	setArchiveDir()	353
14.42.3.4	setMountPoint()	353
14.42.3.5	setPersistDir()	354
14.42.3.6	setupDataBase()	354
14.42.3.7	setVcs()	355
14.43	ArnPipe Class Reference	355
14.43.1	Detailed Description	357

14.43.2 Constructor & Destructor Documentation	357
14.43.2.1 ArnPipe() [1/2]	357
14.43.2.2 ArnPipe() [2/2]	358
14.43.2.3 ~ArnPipe()	358
14.43.3 Member Function Documentation	358
14.43.3.1 changed	358
14.43.3.2 isAutoDestroy()	359
14.43.3.3 isCheckSeq()	359
14.43.3.4 isMaster()	359
14.43.3.5 isSendSeq()	360
14.43.3.6 openUuid()	360
14.43.3.7 operator=()	361
14.43.3.8 outOfSequence	361
14.43.3.9 setAutoDestroy()	361
14.43.3.10setCheckSeq()	361
14.43.3.11setMaster()	362
14.43.3.12setSendSeq()	362
14.43.3.13setValue	363
14.43.3.14setValueOverwrite()	363
14.44 ArnQml Class Reference	364
14.44.1 Detailed Description	365
14.44.2 Member Function Documentation	366
14.44.2.1 arnRootPath()	366
14.44.2.2 instance()	367
14.44.2.3 setArnRootPath()	367
14.44.2.4 setup()	367
14.45 ArnRpc Class Reference	368
14.45.1 Detailed Description	370
14.45.2 Member Typedef Documentation	371
14.45.2.1 Mode	371

14.45.3 Constructor & Destructor Documentation	371
14.45.3.1 ArnRpc()	371
14.45.3.2 ~ArnRpc()	371
14.45.4 Member Function Documentation	371
14.45.4.1 addSenderSignals()	372
14.45.4.2 batchConnect() [1/3]	372
14.45.4.3 batchConnect() [2/3]	372
14.45.4.4 batchConnect() [3/3]	373
14.45.4.5 defaultCall	373
14.45.4.6 getHeartBeatCheck()	375
14.45.4.7 getHeartBeatSend()	375
14.45.4.8 heartBeatChanged	375
14.45.4.9 heartBeatReceived	376
14.45.4.10 invoke() [1/2]	376
14.45.4.11 invoke() [2/2]	377
14.45.4.12 isHeartBeatOk()	377
14.45.4.13 methodPrefix()	378
14.45.4.14 mode()	378
14.45.4.15 open()	378
14.45.4.16 outOfSequence	378
14.45.4.17 pipe()	379
14.45.4.18 pipeClosed	379
14.45.4.19 pipePath()	379
14.45.4.20 receiver()	379
14.45.4.21 rpcSender() [1/2]	380
14.45.4.22 rpcSender() [2/2]	380
14.45.4.23 sendText	380
14.45.4.24 setHeartBeatCheck()	380
14.45.4.25 setHeartBeatSend()	381
14.45.4.26 setIncludeSender()	381

14.45.4.27	setMethodPrefix()	381
14.45.4.28	setMode()	382
14.45.4.29	setPipe()	382
14.45.4.30	setReceiver()	382
14.45.4.31	textReceived	382
14.46	ArnRpcMode Class Reference	383
14.46.1	Detailed Description	383
14.46.2	Member Enumeration Documentation	383
14.46.2.1	E	384
14.47	ArnSapi Class Reference	384
14.47.1	Detailed Description	386
14.47.2	Constructor & Destructor Documentation	387
14.47.2.1	ArnSapi() [1/2]	387
14.47.2.2	ArnSapi() [2/2]	387
14.47.3	Member Function Documentation	387
14.47.3.1	batchConnectFrom()	387
14.47.3.2	batchConnectTo()	388
14.47.3.3	defaultPath()	388
14.47.3.4	open()	389
14.47.3.5	setDefaultPath()	389
14.48	ArnSapiQml Class Reference	390
14.48.1	Detailed Description	391
14.48.2	Member Enumeration Documentation	392
14.48.2.1	Mode	392
14.48.3	Member Function Documentation	392
14.48.3.1	isHeartBeatOk	393
14.48.4	Property Documentation	393
14.48.4.1	heartBeatCheck	393
14.48.4.2	heartBeatSend	393
14.48.4.3	mode	393

14.48.4.4 pipePath	394
14.48.4.5 receiver	394
14.49 ArnScript Class Reference	394
14.49.1 Detailed Description	395
14.49.2 Constructor & Destructor Documentation	395
14.49.2.1 ArnScript() [1/2]	396
14.49.2.2 ArnScript() [2/2]	396
14.49.3 Member Function Documentation	396
14.49.3.1 addObject()	396
14.49.3.2 callFunc()	396
14.49.3.3 engine()	396
14.49.3.4 errorLog()	397
14.49.3.5 errorText	397
14.49.3.6 evaluate()	397
14.49.3.7 evaluateFile()	397
14.49.3.8 globalProperty()	397
14.49.3.9 idName()	398
14.49.3.10 logUncaughtError()	398
14.49.3.11 printFunction()	398
14.49.3.12 setInterruptedText()	398
14.49.4 Member Data Documentation	398
14.49.4.1 _depOfferProto	398
14.49.4.2 _depProto	399
14.49.4.3 _engine	399
14.49.4.4 _itemProto	399
14.49.4.5 _monitorProto	399
14.50 ArnScriptJob Class Reference	400
14.50.1 Detailed Description	401
14.50.2 Constructor & Destructor Documentation	401
14.50.2.1 ArnScriptJob()	401

14.50.3 Member Function Documentation	401
14.50.3.1 errorLog	401
14.50.3.2 quit	401
14.50.3.3 setWatchDogTime	402
14.50.3.4 sigQuit	402
14.50.3.5 yield	402
14.50.4 Property Documentation	402
14.50.4.1 name	402
14.50.4.2 poll	402
14.50.4.3 running	402
14.50.4.4 sleepState	403
14.50.4.5 watchDog	403
14.51 ArnScriptJobControl Class Reference	403
14.51.1 Detailed Description	404
14.51.2 Constructor & Destructor Documentation	404
14.51.2.1 ArnScriptJobControl()	404
14.51.3 Member Function Documentation	404
14.51.3.1 addConfig()	405
14.51.3.2 addInterface()	405
14.51.3.3 addInterfaceList()	405
14.51.3.4 config()	405
14.51.3.5 doSetupJob()	405
14.51.3.6 errorText	406
14.51.3.7 id()	406
14.51.3.8 loadScriptFile()	406
14.51.3.9 name()	406
14.51.3.10script()	406
14.51.3.11scriptChanged	406
14.51.3.12setConfig()	407
14.51.3.13setName()	407

14.51.3.14	setScript	407
14.51.3.15	setThreaded()	407
14.52	ArnScriptJobFactory Class Reference	407
14.52.1	Detailed Description	408
14.52.2	Constructor & Destructor Documentation	408
14.52.2.1	ArnScriptJobFactory()	408
14.52.2.2	~ArnScriptJobFactory()	408
14.52.3	Member Function Documentation	408
14.52.3.1	installExtension()	408
14.52.3.2	setupInterface()	409
14.52.3.3	setupJsObj()	409
14.53	ArnScriptJobs Class Reference	409
14.53.1	Detailed Description	410
14.53.2	Constructor & Destructor Documentation	410
14.53.2.1	ArnScriptJobs()	410
14.53.3	Member Function Documentation	410
14.53.3.1	addJob()	411
14.53.3.2	setFactory()	411
14.53.3.3	start()	411
14.54	ArnServer Class Reference	411
14.54.1	Detailed Description	413
14.54.2	Constructor & Destructor Documentation	413
14.54.2.1	ArnServer()	413
14.54.2.2	~ArnServer()	413
14.54.3	Member Function Documentation	414
14.54.3.1	addAccess()	414
14.54.3.2	addFreePath()	414
14.54.3.3	freePaths()	415
14.54.3.4	isDemandLogin()	415
14.54.3.5	isDemandLoginNet()	415

14.54.3.6 listenAddress()	416
14.54.3.7 noLoginNets()	416
14.54.3.8 port()	416
14.54.3.9 setDemandLogin()	417
14.54.3.10setNoLoginNets()	417
14.54.3.11setWhoIAm()	418
14.54.3.12start()	418
14.55ArnServerRemote Class Reference	419
14.55.1 Detailed Description	420
14.55.2 Constructor & Destructor Documentation	420
14.55.2.1 ArnServerRemote()	420
14.55.2.2 ~ArnServerRemote()	420
14.55.3 Member Function Documentation	420
14.55.3.1 startUseServer()	420
14.56ArnServerRemoteSession Class Reference	421
14.56.1 Detailed Description	422
14.56.2 Member Typedef Documentation	422
14.56.2.1 KillMode	422
14.56.3 Constructor & Destructor Documentation	422
14.56.3.1 ArnServerRemoteSession()	422
14.57ArnServerRemoteSessionKillMode Class Reference	422
14.57.1 Detailed Description	423
14.57.2 Member Enumeration Documentation	423
14.57.2.1 E	423
14.58ArnServerSession Class Reference	423
14.58.1 Detailed Description	424
14.58.2 Constructor & Destructor Documentation	424
14.58.2.1 ArnServerSession()	424
14.58.3 Member Function Documentation	425
14.58.3.1 getAllow()	425

14.58.3.2	getTraffic()	425
14.58.3.3	infoReceived	425
14.58.3.4	loginCompleted	425
14.58.3.5	loginUserName()	425
14.58.3.6	messageReceived	425
14.58.3.7	remoteWhoIam()	426
14.58.3.8	sendMessage()	426
14.58.3.9	socket()	426
14.59	ArnZeroConfB Class Reference	426
14.59.1	Detailed Description	427
14.59.2	Constructor & Destructor Documentation	427
14.59.2.1	ArnZeroConfB()	428
14.59.2.2	~ArnZeroConfB()	428
14.59.3	Member Function Documentation	428
14.59.3.1	domain()	428
14.59.3.2	fullServiceType()	428
14.59.3.3	serviceType()	429
14.59.3.4	setDomain()	429
14.59.3.5	setServiceType()	429
14.59.3.6	setSocketType()	430
14.59.3.7	socketType()	430
14.59.3.8	state()	431
14.60	ArnZeroConfBrowser Class Reference	431
14.60.1	Detailed Description	433
14.60.2	Constructor & Destructor Documentation	433
14.60.2.1	ArnZeroConfBrowser() [1/2]	434
14.60.2.2	ArnZeroConfBrowser() [2/2]	435
14.60.2.3	~ArnZeroConfBrowser()	435
14.60.3	Member Function Documentation	435
14.60.3.1	activeServiceNames()	435

14.60.3.2 browse	436
14.60.3.3 browseError	436
14.60.3.4 getNextId()	437
14.60.3.5 isBrowsing()	437
14.60.3.6 serviceAdded	437
14.60.3.7 serviceChanged	438
14.60.3.8 serviceNameTold()	438
14.60.3.9 serviceRemoved	439
14.60.3.10setSubType()	439
14.60.3.11stopBrowse	440
14.60.3.12subType()	440
14.60.4 Friends And Related Function Documentation	440
14.60.4.1 ArnZeroConfIntern	440
14.61 ArnZeroConfLookup Class Reference	441
14.61.1 Detailed Description	442
14.61.2 Constructor & Destructor Documentation	443
14.61.2.1 ArnZeroConfLookup() [1/2]	443
14.61.2.2 ArnZeroConfLookup() [2/2]	443
14.61.2.3 ~ArnZeroConfLookup()	443
14.61.3 Member Function Documentation	444
14.61.3.1 host()	444
14.61.3.2 hostAddr()	444
14.61.3.3 id()	444
14.61.3.4 isForceQtDnsLookup()	445
14.61.3.5 lookup()	445
14.61.3.6 lookuped	446
14.61.3.7 lookupError	446
14.61.3.8 releaseLookup()	446
14.61.3.9 setForceQtDnsLookup()	447
14.61.3.10setHost()	447

14.61.3.11setId()	447
14.61.4 Friends And Related Function Documentation	448
14.61.4.1 ArnZeroConfIntern	448
14.62ArnZeroConfRegister Class Reference	448
14.62.1 Detailed Description	450
14.62.2 Constructor & Destructor Documentation	451
14.62.2.1 ArnZeroConfRegister() [1/3]	451
14.62.2.2 ArnZeroConfRegister() [2/3]	451
14.62.2.3 ArnZeroConfRegister() [3/3]	451
14.62.2.4 ~ArnZeroConfRegister()	453
14.62.3 Member Function Documentation	453
14.62.3.1 addSubType()	453
14.62.3.2 currentServiceName()	454
14.62.3.3 getTxtRecordMap()	454
14.62.3.4 host()	455
14.62.3.5 port()	455
14.62.3.6 registered	455
14.62.3.7 registerService()	456
14.62.3.8 registrationError	456
14.62.3.9 releaseService()	457
14.62.3.10serviceName()	457
14.62.3.11setHost()	457
14.62.3.12setPort()	458
14.62.3.13setServiceName()	458
14.62.3.14setSubTypes()	459
14.62.3.15setTxtRecord()	459
14.62.3.16setTxtRecordMap()	459
14.62.3.17subTypes()	460
14.62.3.18xtRecord()	460
14.62.4 Friends And Related Function Documentation	461

14.62.4.1 ArnZeroConfIntern	461
14.63 ArnZeroConfResolve Class Reference	461
14.63.1 Detailed Description	463
14.63.2 Constructor & Destructor Documentation	463
14.63.2.1 ArnZeroConfResolve() [1/3]	463
14.63.2.2 ArnZeroConfResolve() [2/3]	464
14.63.2.3 ArnZeroConfResolve() [3/3]	464
14.63.2.4 ~ArnZeroConfResolve()	465
14.63.3 Member Function Documentation	465
14.63.3.1 getTxtRecordMap()	465
14.63.3.2 host()	465
14.63.3.3 id()	466
14.63.3.4 port()	466
14.63.3.5 releaseResolve()	466
14.63.3.6 resolve()	467
14.63.3.7 resolved	467
14.63.3.8 resolveError	467
14.63.3.9 serviceName()	468
14.63.3.10 setId()	468
14.63.3.11 setServiceName()	469
14.63.3.12 txtRecord()	469
14.63.4 Friends And Related Function Documentation	469
14.63.4.1 ArnZeroConfIntern	470
14.64 Arn::ClientSyncMode Struct Reference	470
14.64.1 Detailed Description	470
14.64.2 Member Enumeration Documentation	470
14.64.2.1 E	470
14.65 Arn::Coding Struct Reference	471
14.65.1 Detailed Description	471
14.65.2 Member Enumeration Documentation	471

14.65.2.1 E	471
14.66Arn::DataType Class Reference	471
14.66.1 Detailed Description	472
14.66.2 Member Enumeration Documentation	472
14.66.2.1 E	472
14.67Arn::EnumTxt Class Reference	472
14.67.1 Detailed Description	474
14.67.2 Constructor & Destructor Documentation	475
14.67.2.1 EnumTxt() [1/2]	475
14.67.2.2 EnumTxt() [2/2]	476
14.67.2.3 ~EnumTxt()	476
14.67.3 Member Function Documentation	476
14.67.3.1 addBitSet()	476
14.67.3.2 addBitSetTo()	477
14.67.3.3 addEnumSet()	477
14.67.3.4 addEnumSetTo()	478
14.67.3.5 addFlagsTo()	478
14.67.3.6 addSubEnum()	479
14.67.3.7 addSubEnumPlainTo()	479
14.67.3.8 addSubEnumTo()	480
14.67.3.9 clear()	481
14.67.3.10enumCount()	481
14.67.3.11flagsFromString()	481
14.67.3.12flagsFromStringList()	482
14.67.3.13flagsToString()	483
14.67.3.14flagsToStringList()	483
14.67.3.15getBasicTextList()	484
14.67.3.16getBitSet()	485
14.67.3.17getEnumSet()	485
14.67.3.18getEnumVal() [1/2]	486

14.67.3.19	getEnumVal() [2/2]	486
14.67.3.20	getSubEnumVal() [1/2]	487
14.67.3.21	getSubEnumVal() [2/2]	488
14.67.3.22	getTxt()	489
14.67.3.23	getTxtString()	489
14.67.3.24	humanize()	490
14.67.3.25	sFlag()	490
14.67.3.26	loadBitSet() [1/2]	491
14.67.3.27	loadBitSet() [2/2]	491
14.67.3.28	loadEnumSet() [1/2]	492
14.67.3.29	loadEnumSet() [2/2]	492
14.67.3.30	name()	493
14.67.3.31	numToStr()	493
14.67.3.32	setMissingTxt()	493
14.67.3.33	setTxt()	494
14.67.3.34	setTxtRef()	495
14.67.3.35	setTxtString()	495
14.67.3.36	strToBitpos()	495
14.67.3.37	strToNum()	495
14.67.3.38	subEnumAt()	496
14.67.3.39	subEnumCount()	496
14.67.3.40	subEnumNameAt()	497
14.67.3.41	subEnumPropAt()	497
14.68	ArnZeroConf::Error Struct Reference	498
14.68.1	Detailed Description	498
14.68.2	Member Enumeration Documentation	498
14.68.2.1	E	498
14.69	Arn::ExportCode Class Reference	499
14.69.1	Detailed Description	499
14.69.2	Member Enumeration Documentation	499

14.69.2.1 E	499
14.70 ArnCoreItem::Heritage Struct Reference	500
14.70.1 Detailed Description	500
14.70.2 Member Enumeration Documentation	500
14.70.2.1 E	500
14.71 ArnClient::HostAddrPort Struct Reference	500
14.71.1 Detailed Description	501
14.71.2 Constructor & Destructor Documentation	501
14.71.2.1 HostAddrPort()	501
14.71.3 Member Data Documentation	501
14.71.3.1 addr	501
14.71.3.2 port	501
14.72 Arn::EnumTxt::IncludeMode Struct Reference	501
14.72.1 Detailed Description	502
14.72.2 Member Enumeration Documentation	502
14.72.2.1 E	502
14.73 Arn::InfoType Struct Reference	502
14.73.1 Detailed Description	502
14.73.2 Member Enumeration Documentation	503
14.73.2.1 E	503
14.74 ArnRpc::Invoke Struct Reference	503
14.74.1 Detailed Description	503
14.74.2 Member Enumeration Documentation	503
14.74.2.1 E	503
14.75 Arn::LinkFlags Struct Reference	504
14.75.1 Detailed Description	504
14.75.2 Member Enumeration Documentation	504
14.75.2.1 E	504
14.76 MQArgument< T > Class Template Reference	505
14.76.1 Detailed Description	506

14.76.2 Constructor & Destructor Documentation	506
14.76.2.1 MQArgument()	506
14.77MQBasicTimer Class Reference	506
14.77.1 Detailed Description	507
14.77.2 Constructor & Destructor Documentation	507
14.77.2.1 MQBasicTimer()	507
14.77.3 Member Function Documentation	507
14.77.3.1 interval()	507
14.77.3.2 setInterval()	508
14.77.3.3 start() [1/2]	508
14.77.3.4 start() [2/2]	508
14.78MQGenericArgument Class Reference	508
14.78.1 Detailed Description	509
14.78.2 Constructor & Destructor Documentation	509
14.78.2.1 MQGenericArgument() [1/2]	509
14.78.2.2 MQGenericArgument() [2/2]	509
14.78.3 Member Function Documentation	510
14.78.3.1 label()	510
14.79Arn::NameF Struct Reference	510
14.79.1 Detailed Description	510
14.79.2 Member Enumeration Documentation	510
14.79.2.1 E	510
14.80Arn::ObjectMode Class Reference	511
14.80.1 Detailed Description	511
14.80.2 Member Enumeration Documentation	511
14.80.2.1 E	511
14.81Arn::ObjectSyncMode Class Reference	511
14.81.1 Detailed Description	512
14.81.2 Member Enumeration Documentation	512
14.81.2.1 E	512

14.82ArnRpc::MethodsParam::Params Struct Reference	512
14.82.1 Detailed Description	512
14.82.2 Member Data Documentation	513
14.82.2.1 allMethodIds	513
14.82.2.2 methodIdsTab	513
14.82.2.3 paramNames	513
14.83Arn::QmIMFileIO Class Reference	513
14.83.1 Detailed Description	514
14.83.2 Constructor & Destructor Documentation	514
14.83.2.1 QmIMFileIO()	515
14.83.3 Member Function Documentation	515
14.83.3.1 error	515
14.83.3.2 path()	515
14.83.3.3 pathChanged	515
14.83.3.4 read()	515
14.83.3.5 readBytes()	515
14.83.3.6 setPath	516
14.83.3.7 write()	516
14.83.3.8 writeBytes()	516
14.83.4 Property Documentation	516
14.83.4.1 path	516
14.84Arn::QmIMQtObject Class Reference	517
14.84.1 Detailed Description	518
14.84.2 Constructor & Destructor Documentation	518
14.84.2.1 QmIMQtObject()	518
14.84.2.2 ~QmIMQtObject()	518
14.84.3 Member Function Documentation	518
14.84.3.1 classBegin()	518
14.84.3.2 completed	519
14.84.3.3 componentComplete()	519

14.84.3.4 data()	519
14.84.3.5 data_append()	519
14.84.3.6 data_at()	519
14.84.3.7 data_clear()	519
14.84.3.8 data_count()	520
14.84.3.9 parentChanged	520
14.84.3.10parentItem()	520
14.84.3.11setParentItem()	520
14.84.4 Property Documentation	520
14.84.4.1 data	520
14.84.4.2 parent	521
14.85Arn::QmlMSys Class Reference	521
14.85.1 Detailed Description	522
14.85.2 Member Function Documentation	522
14.85.2.1 xstringToEnum	522
14.85.3 Property Documentation	522
14.85.3.1 quickTypeRun	522
14.86Arn::SameValue Struct Reference	522
14.86.1 Detailed Description	523
14.86.2 Member Enumeration Documentation	523
14.86.2.1 E	523
14.87ArnDiscoverInfo::State Struct Reference	523
14.87.1 Detailed Description	523
14.87.2 Member Enumeration Documentation	523
14.87.2.1 E	523
14.88ArnZeroConf::State Struct Reference	524
14.88.1 Detailed Description	524
14.88.2 Member Enumeration Documentation	524
14.88.2.1 E	524
14.89ArnDiscoverAdvertise::State Struct Reference	525

14.89.1 Detailed Description	525
14.89.2 Member Enumeration Documentation	525
14.89.2.1 E	525
14.90 ArnError::StdCode Struct Reference	526
14.90.1 Detailed Description	526
14.90.2 Member Enumeration Documentation	526
14.90.2.1 E	526
14.91 ArnItemValve::SwitchMode Struct Reference	527
14.91.1 Detailed Description	527
14.91.2 Member Enumeration Documentation	527
14.91.2.1 E	527
14.92 ArnServer::Type Struct Reference	527
14.92.1 Detailed Description	528
14.92.2 Member Enumeration Documentation	528
14.92.2.1 E	528
14.93 ArnScriptJobs::Type Struct Reference	528
14.93.1 Detailed Description	528
14.93.2 Member Enumeration Documentation	528
14.93.2.1 E	528
14.94 ArnDiscover::Type Struct Reference	529
14.94.1 Detailed Description	529
14.94.2 Member Enumeration Documentation	529
14.94.2.1 E	529
14.95 ArnQml::UseFlags Struct Reference	530
14.95.1 Detailed Description	530
14.95.2 Member Enumeration Documentation	530
14.95.2.1 E	530
14.96 Arn::XStringMap Class Reference	530
14.96.1 Detailed Description	533
14.96.2 Member Typedef Documentation	533

14.96.2.1 Options	534
14.96.3 Constructor & Destructor Documentation	534
14.96.3.1 XStringMap() [1/4]	534
14.96.3.2 XStringMap() [2/4]	534
14.96.3.3 XStringMap() [3/4]	534
14.96.3.4 XStringMap() [4/4]	534
14.96.3.5 ~XStringMap()	535
14.96.4 Member Function Documentation	535
14.96.4.1 add() [1/10]	535
14.96.4.2 add() [2/10]	535
14.96.4.3 add() [3/10]	535
14.96.4.4 add() [4/10]	535
14.96.4.5 add() [5/10]	536
14.96.4.6 add() [6/10]	536
14.96.4.7 add() [7/10]	536
14.96.4.8 add() [8/10]	536
14.96.4.9 add() [9/10]	536
14.96.4.10add() [10/10]	537
14.96.4.11addNum() [1/9]	537
14.96.4.12addNum() [2/9]	537
14.96.4.13addNum() [3/9]	537
14.96.4.14addNum() [4/9]	537
14.96.4.15addNum() [5/9]	538
14.96.4.16addNum() [6/9]	538
14.96.4.17addNum() [7/9]	538
14.96.4.18addNum() [8/9]	538
14.96.4.19addNum() [9/9]	538
14.96.4.20addValues()	539
14.96.4.21append() [1/10]	539
14.96.4.22append() [2/10]	539

14.96.4.23	append() [3/10]	539
14.96.4.24	append() [4/10]	539
14.96.4.25	append() [5/10]	540
14.96.4.26	append() [6/10]	540
14.96.4.27	append() [7/10]	540
14.96.4.28	append() [8/10]	540
14.96.4.29	append() [9/10]	540
14.96.4.30	append() [10/10]	541
14.96.4.31	clear()	541
14.96.4.32	fromXString() [1/2]	541
14.96.4.33	fromXString() [2/2]	541
14.96.4.34	indexOf()	541
14.96.4.35	indexOf()	542
14.96.4.36	indexOf()	542
14.96.4.37	indexOfValue()	542
14.96.4.38	indexOfValue()	542
14.96.4.39	info()	542
14.96.4.40	key() [1/3]	543
14.96.4.41	key() [2/3]	543
14.96.4.42	key() [3/3]	543
14.96.4.43	keyRef()	543
14.96.4.44	keys()	543
14.96.4.45	keyString() [1/2]	544
14.96.4.46	keyString() [2/2]	544
14.96.4.47	maxEnumOf()	544
14.96.4.48	operator+=() [1/2]	544
14.96.4.49	operator+=() [2/2]	544
14.96.4.50	operator=()	545
14.96.4.51	options()	545
14.96.4.52	remove() [1/4]	545

14.96.4.53remove() [2/4]	545
14.96.4.54remove() [3/4]	545
14.96.4.55remove() [4/4]	546
14.96.4.56removeValue() [1/2]	546
14.96.4.57removeValue() [2/2]	546
14.96.4.58reverseOrder()	546
14.96.4.59set() [1/8]	546
14.96.4.60set() [2/8]	547
14.96.4.61set() [3/8]	547
14.96.4.62set() [4/8]	547
14.96.4.63set() [5/8]	547
14.96.4.64set() [6/8]	547
14.96.4.65set() [7/8]	548
14.96.4.66set() [8/8]	548
14.96.4.67setEmptyKeysToValue()	548
14.96.4.68setKey()	548
14.96.4.69setOptions()	548
14.96.4.70size()	549
14.96.4.71squeeze()	549
14.96.4.72stringCode()	549
14.96.4.73stringDecode()	549
14.96.4.74toVariantMap()	549
14.96.4.75toXString()	549
14.96.4.76toXStringString()	550
14.96.4.77value() [1/5]	550
14.96.4.78value() [2/5]	550
14.96.4.79value() [3/5]	550
14.96.4.80value() [4/5]	550
14.96.4.81value() [5/5]	551
14.96.4.82valueRef()	551

14.96.4.83	values()	551
14.96.4.84	valueString() [1/5]	551
14.96.4.85	valueString() [2/5]	551
14.96.4.86	valueString() [3/5]	552
14.96.4.87	valueString() [4/5]	552
14.96.4.88	valueString() [5/5]	552
14.97	Arn::XStringMapOptions Class Reference	552
14.97.1	Detailed Description	552
14.97.2	Member Enumeration Documentation	552
14.97.2.1	E	552
14.98	Arn::XStringMapQml Class Reference	553
14.98.1	Detailed Description	554
14.98.2	Member Function Documentation	554
14.98.2.1	add [1/2]	554
14.98.2.2	add [2/2]	555
14.98.2.3	clear	555
14.98.2.4	indexOf	555
14.98.2.5	indexOfValue	555
14.98.2.6	key [1/2]	555
14.98.2.7	key [2/2]	556
14.98.2.8	keys	556
14.98.2.9	remove [1/2]	556
14.98.2.10	remove [2/2]	556
14.98.2.11	removeValue	556
14.98.2.12	set [1/2]	557
14.98.2.13	set [2/2]	557
14.98.2.14	setEmptyKeysToValue	557
14.98.2.15	toMap	557
14.98.2.16	value [1/2]	557
14.98.2.17	value [2/2]	558
14.98.2.18	values	558
14.98.3	Property Documentation	558
14.98.3.1	size	558
14.98.3.2	xstring	558

15 File Documentation	559
15.1 doc/Changelog_Todo.md File Reference	559
15.2 doc/Description.md File Reference	559
15.3 doc/HelpIndex.txt File Reference	559
15.4 doc/Install.md File Reference	559
15.5 doc/Internals.md File Reference	559
15.6 examples/Examples.txt File Reference	559
15.7 README.md File Reference	559
15.8 src/Arn.cpp File Reference	559
15.9 src/ArnAdaptItem.cpp File Reference	561
15.9.1 Macro Definition Documentation	561
15.9.1.1 MUTEX_CALL	562
15.9.1.2 MUTEX_CALL_RET	562
15.10src/ArnBasicItem.cpp File Reference	562
15.11src/ArnClient.cpp File Reference	563
15.12src/ArnCompat.cpp File Reference	563
15.13src/ArnCoreItem.cpp File Reference	564
15.14src/ArnDepend.cpp File Reference	564
15.14.1 Variable Documentation	565
15.14.1.1 ArnDependPath	565
15.15src/ArnDiscover.cpp File Reference	565
15.16src/ArnDiscoverConnect.cpp File Reference	566
15.17src/ArnDiscoverRemote.cpp File Reference	566
15.18src/ArnEvent.cpp File Reference	566
15.18.1 Macro Definition Documentation	567
15.18.1.1 TO_IDX_RETVAL	567
15.19src/ArnInc/Arn.hpp File Reference	567
15.19.1 Macro Definition Documentation	569
15.19.1.1 ARNREAL	570
15.19.1.2 DATASTREAM_VER	570

15.20src/ArnInc/ArnAdaptItem.hpp File Reference	570
15.21src/ArnInc/ArnBasicItem.hpp File Reference	571
15.22src/ArnInc/ArnClient.hpp File Reference	572
15.23src/ArnInc/ArnCompat.hpp File Reference	572
15.23.1 Macro Definition Documentation	573
15.23.1.1 ARN_ModeRecursiveMutex	573
15.23.1.2 ARN_RecursiveMutex	574
15.23.1.3 ARN_RegExp	574
15.23.1.4 ARN_RegExpValidator	574
15.23.1.5 ARN_SIZETYPE	574
15.23.1.6 ARN_ToRegExp	574
15.24src/ArnInc/ArnCoreItem.hpp File Reference	575
15.25src/ArnInc/ArnDepend.hpp File Reference	575
15.26src/ArnInc/ArnDiscover.hpp File Reference	576
15.27src/ArnInc/ArnDiscoverConnect.hpp File Reference	578
15.28src/ArnInc/ArnDiscoverRemote.hpp File Reference	579
15.29src/ArnInc/ArnError.hpp File Reference	579
15.30src/ArnInc/ArnEvent.hpp File Reference	580
15.31src/ArnInc/ArnInterface.hpp File Reference	581
15.32src/ArnInc/ArnItem.hpp File Reference	582
15.32.1 Function Documentation	583
15.32.1.1 operator<<()	583
15.33src/ArnInc/ArnItemB.hpp File Reference	583
15.34src/ArnInc/ArnItemValve.hpp File Reference	584
15.35src/ArnInc/ArnLib.hpp File Reference	585
15.36src/ArnInc/ArnLib_global.hpp File Reference	586
15.36.1 Macro Definition Documentation	586
15.36.1.1 ARNLIBSHARED_EXPORT	586
15.37src/ArnInc/ArnLinkHandle.hpp File Reference	587
15.38src/ArnInc/ArnM.hpp File Reference	587

15.39src/ArnInc/ArnMonEvent.hpp File Reference	588
15.40src/ArnInc/ArnMonitor.hpp File Reference	589
15.41src/ArnInc/ArnPersist.hpp File Reference	590
15.42src/ArnInc/ArnPersistSapi.hpp File Reference	591
15.43src/ArnInc/ArnPipe.hpp File Reference	592
15.44src/ArnInc/ArnQml.hpp File Reference	593
15.44.1 Macro Definition Documentation	594
15.44.1.1 QML_ENGINE	594
15.44.1.2 QML_LIST_PROPERTY	594
15.44.1.3 QML_NETACC_FACTORY	595
15.44.1.4 QML_PARSER_STATUS	595
15.44.1.5 QML_Qt4	595
15.44.1.6 QML_QUICK_TYPE	595
15.45src/ArnInc/ArnQmlMQt.hpp File Reference	595
15.46src/ArnInc/ArnQmlMSystem.hpp File Reference	596
15.47src/ArnInc/ArnRpc.hpp File Reference	597
15.47.1 Macro Definition Documentation	598
15.47.1.1 MQ_ARG	598
15.47.1.2 no_queue	599
15.48src/ArnInc/ArnSapi.hpp File Reference	599
15.48.1 Macro Definition Documentation	600
15.48.1.1 MQ_PUBLIC_ACCESS	600
15.49src/ArnInc/ArnScript.hpp File Reference	600
15.49.1 Macro Definition Documentation	601
15.49.1.1 ARN_JSCONTEXT	601
15.49.1.2 ARN_JSENGINE	601
15.49.1.3 ARN_JSVALUE	602
15.49.1.4 ARN_JSVALUE_LIST	602
15.50src/ArnInc/ArnScriptJob.hpp File Reference	602
15.51src/ArnInc/ArnScriptJobs.hpp File Reference	603

15.52src/ArnInc/ArnServer.hpp File Reference	604
15.53src/ArnInc/ArnServerRemote.hpp File Reference	605
15.54src/ArnInc/ArnZeroConf.hpp File Reference	606
15.54.1 Typedef Documentation	607
15.54.1.1 DNSServiceRef	607
15.55src/ArnInc/Math.hpp File Reference	608
15.56src/ArnInc/MQFlags.hpp File Reference	609
15.56.1 Macro Definition Documentation	610
15.56.1.1 MQ_DECLARE_ENUM_NSTXT	610
15.56.1.2 MQ_DECLARE_ENUMTXT	610
15.56.1.3 MQ_DECLARE_FLAGS_NSTXT	611
15.56.1.4 MQ_DECLARE_FLAGSTXT	611
15.56.1.5 MQ_DECLARE_SUBETXT	611
15.56.1.6 MQ_NSTXT_FILL_MISSING	612
15.56.1.7 MQ_NSTXT_FILL_MISSING_FROM	612
15.56.1.8 MQ_SUBETXT_ADD_ABSDEF	612
15.56.1.9 MQ_SUBETXT_ADD_ABSOP	612
15.56.1.10MQ_SUBETXT_ADD_RELDEF	612
15.56.1.11MQ_SUBETXT_ADD_RELOP	613
15.57src/ArnInc/MQFlagsBase.hpp File Reference	613
15.57.1 Macro Definition Documentation	614
15.57.1.1 MQ_DECLARE_ENUM	614
15.57.1.2 MQ_DECLARE_FLAGS	614
15.57.1.3 MQ_DECLARE_OPERATORS_FOR_FLAGS	614
15.58src/ArnInc/XStringMap.hpp File Reference	615
15.58.1 Macro Definition Documentation	616
15.58.1.1 ARNXSTRINGMAP_VER	616
15.58.2 Typedef Documentation	616
15.58.2.1 MQVariantMap	616
15.59src/ArnItem.cpp File Reference	616

15.59.1 Function Documentation	617
15.59.1.1 operator<<()	617
15.60src/ArnItemB.cpp File Reference	617
15.61src/ArnItemNet.cpp File Reference	617
15.62src/ArnItemNet.hpp File Reference	618
15.63src/ArnItemValve.cpp File Reference	618
15.64src/ArnLib.cpp File Reference	619
15.65src/ArnLink.cpp File Reference	620
15.66src/ArnLink.hpp File Reference	620
15.66.1 Typedef Documentation	621
15.66.1.1 ArnCoreItemList	621
15.66.1.2 ArnLinkList	621
15.67src/ArnLinkHandle.cpp File Reference	622
15.68src/ArnM.cpp File Reference	622
15.69src/ArnMath.cpp File Reference	623
15.70src/ArnMonitor.cpp File Reference	624
15.71src/ArnPersist.cpp File Reference	624
15.72src/ArnPipe.cpp File Reference	625
15.73src/ArnQml.cpp File Reference	625
15.74src/ArnQmlMQt.cpp File Reference	626
15.75src/ArnQmlMSystem.cpp File Reference	626
15.76src/ArnRpc.cpp File Reference	627
15.76.1 Macro Definition Documentation	627
15.76.1.1 RPC_STORAGE_NAME	627
15.77src/ArnSapi.cpp File Reference	627
15.78src/ArnScript.cpp File Reference	628
15.79src/ArnScriptJob.cpp File Reference	628
15.79.1 Variable Documentation	629
15.79.1.1 EventQuit	629
15.80src/ArnScriptJobs.cpp File Reference	629
15.81src/ArnServer.cpp File Reference	630
15.82src/ArnServerRemote.cpp File Reference	630
15.83src/ArnSync.cpp File Reference	631
15.83.1 Macro Definition Documentation	631
15.83.1.1 ARNSYNCVER	631
15.84src/ArnSync.hpp File Reference	631
15.84.1 Macro Definition Documentation	632
15.84.1.1 ARNRECNAME	632
15.85src/ArnSyncLogin.cpp File Reference	633
15.86src/ArnSyncLogin.hpp File Reference	633
15.87src/ArnXStringMap.cpp File Reference	634
15.88src/ArnZeroConf.cpp File Reference	634
15.89src/MQFlags.cpp File Reference	635

16 Example Documentation	637
16.1 ArnDemoChat/main.cpp	637
16.2 ArnDemoChat/MainWindow.cpp	637
16.3 ArnDemoChat/MainWindow.hpp	638
16.4 ArnDemoChatServer/ChatSapi.hpp	639
16.5 ArnDemoChatServer/main.cpp	640
16.6 ArnDemoChatServer/MainWindow.cpp	640
16.7 ArnDemoChatServer/MainWindow.hpp	642
Index	645

Chapter 1

README

Copyright (C) 2010-2022 Michael Wiklund. All rights reserved. Contact: arnlib@wiklunden.se

ArnLib - Active Registry Network.

This Qt based library makes it easy to distribute changing data objects. It also gives a central place to find all your systems' current data. By using the ArnBrowser, all data objects are real time presented in a tree view.

Comparison to similar concepts

- **Data mart:** Statistical data gathered from different systems. This makes it possible to run cross system analysis.
- **Windows Active Directory (R):** Centralized configuration data. All in one place easily shared.
- **ArnLib:** Hot changing data from different systems. Enables easy cross system data exchange, debugging, etc.

Installation and usage

Read <doc/Install.md> how to build, install and use.

ArnLib could be beneficial in a lot of projects. It should be well suited to the following conditions:

- *A lot of configurations and changing values.*
ArnLib helps giving out-of-the-box diagnostics and ability to change values not yet available in the custom application user interface.
- *Hardware with a lot of sensors and controls.*
Arnlib helps giving a common interface and diagnostic.
- *Distributed systems.*
ArnLib helps giving an out-of-the-box data sharing system that replicates [Arn](#) objects.
- *Networked services by RPC (remote procedure call).*
Will be quite the same as setting up signals and slots for local calls. You can find an easy example in the ArnLib package, showing a simple chat Client and Server.
- *ZeroConfig detection of present services.*
Helps advertise and browse a service (ftp, http, arn, ...) on a local network. This is similar to UPNP discovery of units.

Main features

- Based on Qt (4, 5 and 6), multiple platform and OS support.
- Qt based [Arn](#) browser available. Allows you to access all data objects in a tree view (see ArnBrowser).
- Web based [Arn](#) browser available, allowing you to use a standard web browser (see WebArnBrowser).

Arn Data Objects

- Hierarchical storage of hot changing data objects.
- [Arn Data objects](#) can be: integers, floats, strings, byte arrays and variants (most Qt data types, e.g. QImage).
- Data objects can typically be: measures, settings, data streams, documents, scripts (js), etc.
- [Arn Data objects](#) are thread-safe.
- Native support for data validation and double direction pipes (streams).
- Enums, Flags and SubEnums in code available as text for output and parse.
- Metrics of [Arn](#) available in [Arn](#) tree.

Sharing

- Data objects can be shared in a single program, among threads or between programs, at different computers. This division of program modules can be changed and is transparent to usage of ArnLib.
- Support for temporary session data objects. Optional auto-delete of objects when tcp/ip closes and unique uuid names.
- Dependency system with custom offered services and getting signals when all needed services are available.
- Monitoring of newly created data objects and any mode change.
- Login system, to give access protection and different privileges.
- Remote access to [Arn](#) sessions, to view and control currently connected clients.

Persistent storage

- Optional persistent storage of object in SQLite or in a file.
- Support for version control (VCS) of objects stored in files. This can be git.

Java Script

- Native support in JavaScript for: [Arn Data Objects](#), Dependency system and Monitoring of changed objects.
- Java Script jobstack with preemptive and cooperative scripts running at different priorities.

Data streams and *Remote Procedure Call*

- All data streams (pipes) can easily be monitored and manual test data can be inserted (see ArnBrowser).
- Service Api, for calling routines anywhere in connected [Arn](#). *Remote Procedure Call* (RPC) simple to use as "remote signal slots".
- Service Api has an automatically generated help for giving syntax when doing debug manual typed calls to a RPC service.

ZeroConfig and Discover

- Any service (ftp, http, arn, etc) can be advertised, browsed and resolved for its host address and port number.
- High level, fully automatic support specialised for *arn* service, can e.g. remotely change the advertised *service name*.
- Optional internal DNS_SD/mDNS routines for no dependency to any extra library.

Qml

- Support in Qml for: [Arn Data Objects](#), monitoring of changed objects and Service Api (RPC).
- Added support in Qml for url like "arn:///test.qml".
- Possibility to create a remote generic Qml running environment, comparable to a web browser running an arbitrary web application. This is done by ArnBrowser.

Chapter 2

ArnLib Changelog / Todo

Major

- Script support for Sapi.
- ArnObject Link to other ArnObject (like in a filesystem).
- General access system with privileges at ArnObject level.
- Add more examples.
- Add Function tests.
- Add more Unit tests.
- API to Sync ArnObjects with other protocols (e.g. JSON-based).
- API to Sync ArnObjects over other media (e.g. CAN).
- Javascript based ArnLib for Web-applications over WebSocket.

Minor

- Optimize data transfer with minimal copying.
- Converter classes for ArnPipes to other streams (e.g UART, TCP etc).
- Addition to login a system to "pair" [ArnServer](#) and [ArnClient](#).

Done in 4.0

- Adopted to Qt6 Now with core5compat and some other deprecated parts.
- Added ArnCompat This helps using same code for any Qt-version, relating to [Arn](#) based code.
- Added support for QJSEngine. Previous support for QScript still works when using Qt version before Qt6. Also some compatibility is handled by [ArnScript](#).
- Added [ArnScriptJobs](#) general watchdog. This is for QJSEngine but also backported to QScript.
- Added demo for [ArnScriptJobs](#) in examples.
- Added scriptauto to use in pro-file This select QJSEngine or QScript depending on Qt-version. Preferred is QJSEngine.

- Added hostIp-list to [ArnDiscover](#) advertise. This can be seen as new properties in ArnBrowser during "Arn Discover".
- Added [ArnDiscover](#) logic to also use "HostIp" property to select addr. As MDns only can have 1 IP for a hostname, this helps choosing an connectable IP.
- Added SubEnum in MQFlags. One or more Enums can be included in Flags as SubEnum. This is also fully supported in text output, parse and XString-representation.
- EnumTxt is now a general usable class. It can be used in the application for dynamic handling of Enums, Flags and SubEnums.
- Added XStringMap to QML.
- Added support for Enum (from [Arn](#) set) in QML.
- XStringMap has now options for minimizing length of XString. This is backwards compatible and new formats are detected automatically. Options include: null-substitution (\0), repeated characters and a framing format.
- ArnSync is using new XStringMap optimization. This is backward compatible. SyncVersion is changed to 4.0.
- XStringMap has new option AnyKey When used, any key or value can be used, also binary blobs.
- Added atomic operations in [ArnItem](#) for: "bitSet" and "+=". Single items are local and bidir works remotely by enabling a included provider.
- Added ArnMath General math with templates and also some cpu-optimized Includes: modulo, circleVal, is↔Power2, log2, minLim, maxLim, rangeLim
- Added [ArnBasicItem](#) isAssigning(). This can be used to cancel echo.
- Changed [ArnDepend](#) to be more robust and independent of [ArnClient](#) connection state.

Done in 3.1

- Added [ArnAdaptItem](#). Can be used in threads without eventloop or even non Qt threads.
- Added [ArnClient](#) syncMode for different client sync methods.
- Now all Bidir Objects has no echo, this was true only for pipes before. The official value comes always from one provider. The requested value can be from many.
- Single objects has echo with better logic to avoid bad echoes that restores old values.
- Persistent values to client has more robust logic, especially for Master objects.
- Added [ArnItem::setUncrossed\(\)](#), will make it easier to build [Arn](#) Bridges etc.

Done in 3.0

- Delete ArnObject, but only local (remove any sync of it).
- [ArnClient](#) disconnect and close.
- Optimized memory consumption with pointers to different data in ArnLink.
- Minimized signal/slot:s in ArnLink by change to [ArnEvent](#).
- Distributed deletion of folders.
- Distributed create of folder.
- [ArnMonitor](#) detects destructions of [Arn Objects](#).

- Added `setDelay` in [ArnItemQml](#), rework `changed()` and using timer events.
- Access system for Server/Client login with session level privilege.
- Allow read access to "freePaths" without login. Used to view for example licenses.
- Option for free nets, e.g. "localnet", that don't need login for full access.
- A flush mechanism for [ArnPersist](#) to force saving.
- Pimpl: Converted to d-pointer for making binary compatible library in the future.
- Started unit tests
- Optimized `HandleData` class with Null-state that can be `this == 0`.
- Made `ArnObject` (`ArnLink`) none `QObject` to save memory and independent on main-thread-create. New methods and data for `parent()` etc.
- Changed to `ArnLink::toInt(bool* isOk = 0)`. To make `ignoreSameValue` work as expected for "" -> `int=0` and similar. Same for all `toXXX()`.
- Changed to `ArnItem::toInt(bool* isOk = 0)`. To give application the possibility to detect data type conversion errors.
- [ArnBasicItem](#) with no `QObject`, only inherited to give [ArnEvent](#) (`QEvent`). Small footprint!
- `ArnItemNet` (`Arn` syncing item) inherited from [ArnBasicItem](#) for small footprint.
- [ArnMonitor](#) no dependency to `ArnItemNet`, that can be in other thread.
- [ArnItem](#) none native data-types: `uint`, `int64` & `uint64`.
- Put [ArnServer](#) client sessions in `"/Local/..."` to be viewed and controlled (e.g kill). Added [ArnServerRemote](#) class. Also chat between server (pipe in [Arn](#)) and client is supported.
- Browsing and controlling connected clients.
- [Arn](#) Registry metrics available in `"/local/..."`
- Added auto "humanize" logic to `MQFlags` text. This will convert e.g. enum value `WriteDelay200Ms` to "Write delay 200 ms".
- `XStringMap` improved, e.g. `addNumber()`.

Done in 2.3

- Added `ArnReal` to be either float or double.
- Fixed zero reference to be more robust when deleting [Arn](#) objects in threads.
- Changed `ArnM::valueXXX` to create none existent `ArnObjects`.
- In Signal Slot (and more) use `"const Type&"`.
- QML with "files" as `ArnObject` and other integration with [Arn](#).
- QML support for Sapi.
- [ArnClient](#) stored centrally with an id. Also accessible by the id.
- External engine can be assigned to [ArnScript](#).
- [ArnSapi](#) default path, not needing path for the pipe.
- Persistent values can be flushed to storage on demand.
- Enums (and flags) using `MQFlags` can use `toString` and more.
- Unit test sub project with tests for enum text.
- `ArnQmlMQt` with `MQtObject` for non gui qml (like `Item/QtObject`).

Chapter 3

General Description

This document describes the general concepts of the ArnLib.

3.1 Arn Data Objects

All objects are stored in a tree hierarchy and the naming is similar to typical file systems, e.g. `"/Measure/Water/←Temperature/value"`.

To get a handle to a folder, use a path ending with `"/"`, e.g. `"/Measure/Water/"`.

Folder names can be empty. In the above example, the first level folder is empty and the second level folder is "Measure". The empty folder name can also be referred as `"@"`. Again, the example can equally be written `"/@/←Measure/Water/Temperature/value"`. This `"@"` is typically used when an empty name is unacceptable, e.g. in the tree viewer of the ArnBrowser tool.

A relative path is also called the `local path`, e.g. `"Sys/Discover/This/Service/value"`.

Each part in a given path is dynamically added as needed, i.e. any path can be used without explicitly creating each folder in advance.

3.1.1 ArnItem access

To access an *ARN Data Object* one can use `ArnM::setValue()` and `ArnM::valueInt()` etc. This is a polled access, and gives no signals / events for changed objects. Also this method is rather slow as it has to locate the object via a path lookup. However its good for application assign object "once".

For continous access to an *ARN Data Object* its better to use an `ArnItem`. This will be a handle to the object that give fast access. It will also provide signals for changed object. `ArnItem` is QObject based and has its characteristics.

Yet another way to access an *ARN Data Object*, is an `ArnBasicItem`. This will give a basic handle to the object. It is fast, small and is not based on QObject. As such it can not use signals and slots, but it can provide ArnEvents.

Normally `ArnItem` should be used, as it has a higher level interface with QObject signals and slots. Typically `ArnBasicItem` is used when no signal is needed, i.e only using direct access with `setValue` and `toXXX` methods. If you need a lot of ArnBasicItems and memory foot print (or speed) is important, You can consider to use `ArnBasicItem` with ArnEvents even if it will be harder to code.

You can expect `ArnBasicItem` to be lees than a third of the size of an `ArnItem`. Tests has shown `ArnBasicItem` to take half the time assigning an integer, compared to `ArnItem`.

3.1.2 Modes

Mode change is a one direction process. Once a specific *mode* is set, it can't be reset.

If the [ArnItem](#) is in a closed state when the *mode* change is done, the added modes will be stored and the real *mode* change is done when the [ArnItem](#) is opened to an *ARN Data Object*.

If the *general mode* change is done to a [shared](#) object, the change of *general mode* is also done at the server and any connected clients.

The following *general modes* are available:

- **BiDir** A two-way object, typically for validation or pipe. See [bidirectional](#) objects.
- **Pipe** Implies *BiDir* and all data is preserved as a stream during [sharing](#). Without *Pipe mode*, [sharing](#) is optimized to sync latest value and not all values in a stream.
- **Save** Sets the *ARN Data Object* as persistent and any data assigned to it will be saved. The persistent service must be started at the server. See [persistent](#) objects.

Additionally there are some *sync modes*. These modes are used by the local client session and are not shared with others. The *sync modes* must be set before the [ArnItem](#) is opened to an *ARN Data Object*.

Following *sync_modes* are available:

- **Master** The *ARN Data Object* (at client side) is set as *default generator* of data. Normally the server is the *default generator* of data. See [Sync Rules](#).
- **AutoDestroy** The *ARN Data Object* (at client side) is set up for auto destruction. When the client closes *tcp/ip*, the server side will destroy the *ARN Data Object* and this will also be done at any connected clients.

Note: It's convenient to always set all the needed modes before an [ArnItem](#) is opened or an [ArnItem](#) is used as a template. See [ArnItem::setTemplate\(\)](#).

3.1.3 Local

A relative path is also called the *local path*, e.g. the [Discover remote service name](#) at path "Sys/Discover/↔ This/Service/value". The *local path* is mapped to the absolute path "/Local/". The example is then equal to "↔ Local/Sys/Discover/This/Service/value". The *local path* should not be [shared](#) as it will contain specific data for its running program.

The exception to not sharing *local path* is for some kind of remote client that must be able to change an *ARN Data Object* in the *local path* at the remoted target. For example this is used to change the [Discover remote service name](#) for a target host.

Note: Do always mount the *local path* of the server at a different path at the client. This is to avoid collision with the client's own *local path* data.

In the above example, a remote client using [ArnClient::addMountPoint\("@HostLocal/", "/Local/"\)](#) will share and access the [Discover remote service name](#) at the path "@HostLocal/Sys/Discover/This/Service/value".

3.1.4 Naming conventions

These rules must not be obeyed, but are recommended, to get the most benefits of the Arn echo system, like the ArnBrowser tool.

- First level folder empty, e.g. `"/MyGlobalFolder/Date/value"`, is a global path and is `shared` to ARN server and clients.
- First level folder starts with "@", e.g. `"/@SomeServer/MyFolder/Date/value"`, is a shared path and is `shared` to an ARN server (typically with some other remote path).
- First level folder is `/Local`, e.g. `"/Local/Key/value"`, is a `local path` and is not `shared`.
- Path is relative, e.g. `"Key/value"`, is a `local path` and is not `shared`.
- When a leaf is used as an attribute, the following names are reserved:
 - **value** the value of the above closest folder denotation, e.g. `"Temperature/value"` (=10).
 - **name** the description of the above closest folder denotation, e.g. `"Server-1/name"` ("Hugin").
 - **set** allowed values and conversion to a more descriptive form, e.g. `"0=Off 1=On"`.
 - **bitSet** used bits and conversion to a more descriptive form, e.g. `"B0=Read B1=Write"`.
 - **property** like precision and unit, e.g. `"prec=1 unit=°C"`.
 - **info** like tool tips, e.g. `"<tt>Standard UV radiation index</tt>"`.
 - `**help.**XXX` like `"help.xhtml"` contains help in `xhtml` format.

3.1.5 Bidirectional Arn Data Objects

A bidirectional *ARN Data Object* is actually a double object, a twin. Each part has its own path but their life span is depending on each other.

One part is the normal "official" and the other part is *provider*. The provider has an added "!" to the normal path, e.g. normal = `"/Measure/Depth/value"`, provider = `"/Measure/Depth/value!"`.

Data written to one part ends up in the other. This can be compared to crossing electrical lines from one unit to another unit regarding transmit and receive signals in each unit. When a provider slot is connected to the provider part (`ArnItem`), the slot will receive "request" data from the normal part. The provider slot processes the request data and writes the result to the same provider part. This way the result will end up in the normal "official" part.

This functionality can typically be used for data validation and limiting.

The crossing property of BiDir can be suppressed by using `ArnItem::setUncrossed()`. Again this can be compared to uncrossed electrical lines from a unit to a "communicator" (modem, switch, hub ...) regarding transmit and receive signals. Not surprisingly this is usually a easier mode when making some kind of Bridge for ARN.

3.1.6 Pipe Arn Data Objects

Pipes also use the [bidirectional](#) functionality. The two (twin) parts are then named *requester* and *provider*.

All data put into a pipe are part of a stream and as such will be fully transferred (synchronized) if they are [shared](#) with a server and other clients.

[ArnPipe](#) is a specialized class for handling pipes. It contains logic for handling [sequence check](#) and [anti congest](#).

Data stream to and from a pipe can be controlled using [ArnItemValve](#) class. Actually [ArnItemValve](#) can control any [ArnItemB](#) derived class.

3.1.6.1 Pipe sequence check

Sequence check is used to make sure everything is received and nothing is lost or comes twice. This might happen when a tcp/ip connection goes up and down.

The sequence check uses a hidden sequence number not visible in the pipe stream. The sequence number is increased for each assignment to the pipe. The sending and checking of this sequence number is activated at each end of the pipe.

When checking is activated and the received sequence number is unexpected, a signal will be generated.

See also [ArnPipe::setSendSeq\(\)](#), [ArnPipe::setCheckSeq\(\)](#), [ArnPipe::outOfSequence\(\)](#).

3.1.6.2 Pipe anti congest

When the pipe is a [shared object](#), all assignment to the pipe is queued up in a send queue. If there is a disconnect in the tcp/ip, an [ArnServer](#) will drop the send queue. But in an [ArnClient](#), this send queue will grow out of control if assignments to the pipe keeps coming. This problem can also arise with a fast rate of status messages on a slow network.

One possibility is to keep track of the connection status, but this involves knowing about which [ArnClient](#) (if many) to get status from. It also doesn't handle the problem with a slow network.

A probably better way is to use the *Pipe anti congest* logic.

We identify *messages* that can be sent any number of times and are used to check the data flow, resending, status and alike. Typically this can be *Heart beat*, *ping*, *request update*, *current time* etc. These *async messages* are assigned using [ArnPipe::setValueOverwrite\(\)](#).

A regular expression is needed to identify "equal" *async messages*, that can be overwritten in the send queue. If *async messages* are repeatedly assigned to a pipe by [ArnPipe::setValueOverwrite\(\)](#), the send queue will then not grow.

All other *messages* will be normally assigned to the pipe. But these *messages* will only be assigned when normal data flow is present. Typically there is some expected *feedback message* from the receiving part to block uncontrolled assignment from one side of the pipe.

3.1.7 Persistent Arn Data Objects

The *server* must use [ArnPersist](#) to support the persistence service. As a standard *persist storage*, *ARN Data Objects* are stored in a SQLite database. It's also possible to store each object as a file.

The *mount point* (path) for collecting the persistent *ARN Data objects* is set by [ArnPersist::setMountPoint\(\)](#). For server applications this is typically set to `"/"`, which makes all *ARN Data Objects* potential persistent. In client applications the *mount point* is typically restricted to [Arn::pathLocal](#), which only saves local *ARN Data Objects* in the local *persist storage*.

Any connected *client* or the *server* can make an *ARN Data Object* persistent. Just open an [ArnItem](#) to the object and change *mode* to *Save*.

```
ArnItem arnMaxLevel;  
arnMaxLevel.addMode( Arn::ObjectMode::Save);  
arnMaxLevel.open("//Config/Level/Max/value");
```

When the *ARN Data Object* is set to *Save* mode, it's automatically loaded by the [ArnPersist](#). At the *server* this is instantly done. A *client* has to wait for the value to get synced from the *server*. It's convenient to use [ArnDepend](#) to get a signal when the value is loaded and ready to use.

When the *ARN Data Object* is changed, it will be automatically saved by [ArnPersist](#). There is a delay from first change of the object until the saving is done, see [ArnItem::setDelay\(\)](#). This allows for intensive updates of the object without choking down the server with saving operations.

It's possible to mark an object in the SQLite data base as *mandatory*. In this way the *ARN Data Object* is set as *persistent* and gets loaded at start of [ArnPersist](#).

3.1.7.1 Saving objects in files

To use the *persistent* storing of *ARN Data Objects* in files, the *root* directory is set by: [ArnPersist::setPersistDir\(\)](#). This can also be combined with support of VCS (version control system). See [ArnPersist::setVcs\(\)](#). Currently there is a support module for *git*.

In the *root* directory and below, all (VCS) persistent files are stored. The *root* directory corresponds to the *root* in Arn tree.

Example: *root* directory is set to `"/usr/local/arn_persist"`. There is a file stored at `"/usr/local/arn_persist/@/doc/help.xhtml"`. This file will be mapped to Arn at `"/doc/help.xhtml"`.

Any files stored in the *root* directory and below, get loaded into their *ARN Data Object* with *mode* set as *persistent* at start of [ArnPersist](#).

The files get updated in a similar way to the data base update.

3.1.8 Sharing Arn Data Objects

A fundamental aspect of Arn is that *ARN Data Objects* can be shared. This is centralized to the *ARN Server*, which stores all shared objects. It's still a distributed model as each client and server has their own set of *ARN Data Objects* that operate independent of any connection.

Each *ARN Client* connects to the *ARN Server* and decides which part of the *ARN Data Object* tree to be shared.

`ArnClient::addMountPoint("/Share/")` will make the tree `/Share/` shared.

This doesn't mean that everything in the shared tree at the server now will be available at the client. The client has to create an *ARN Data Object* in the shared tree. The client can then decide the exact objects of interest.

`ArnItem::Open("/Share/Test/value")` will open a shared object in previous example.

Note: Normally `/"` or `/@.../` is used for shared. See [naming conventions](#).

The remote tree can be at a different path than the local tree (mount point).

```
ArnClient::addMountPoint("/@Host/", "/") // Makes the server shared at "/@Host/".
ArnItem::open("/@Host/Share/Test/value") // Open the shared object in previous example.
```

3.1.8.1 Dynamic port

An *ArnServer* can be created with *port* set to 0. This will be handled as a *dynamic port* and the system will assign a free *port number* to the server. The *port number* will be taken from a range specified by IANA.

This can typically be used to skip configuring static port numbers and be able to have multiple instances of the *ArnServer* on the same machine. As an *ArnClient* must find its *ArnServer*, this can be used together with *ArnDiscoverRemote* / *ArnDiscover*.

3.1.9 Sync rules

Syncing between client and server is normally handled automatically, but for special needs and reference this chapter gives an idea of the rules. Also this describes the rules when connection is established. After that, normal syncing is done almost symmetrically between client and server.

An *ARN Data Object* with Master *Mode* is used as *default generator* of data. Normally the server is the *default generator* of data. This makes difference when client connects or reconnects to the server. The data from the *default generator* is then used and synced.

Also to have minimal data exchange when using non *BiDirectional ARN Data Object*, one should take Master mode into consideration. This is more important for big objects.

When a Null value is synced, the receiver store this as an empty value, i.e. it's not stored as Null which is impossible.

3.1.9.1 Sync rules for Pipe

- Pipes should be considered to carry a flow, not a value.
- The pipe flow (to server) is enabled after `ArnClient::connectToArn()`, and is disabled after `ArnClient::close()`.
- In client, an enabled flow can queue up the stream of data when there is no connection to server.
- In client, the flow keeps being enabled even if the `ArnClient::connectToArn()` fails or there is a TCP disconnect.
- When the flow is disabled (`ArnClient::close`), all queued stream data will be sent if possible.
- Server can never queue anything when disconnected, as the server session is only living when connected.

3.1.9.2 ClientSyncMode

ClientSyncMode can be set with `ArnClient::setSyncMode()`. Basically this controls if a client *ARN Data Object* is considered as a Master object (see also [Modes](#)).

ClientSyncMode doesn't affect a pipe. Default mode is StdAutoMaster.

- **StdAutoMaster** Dynamic auto master mode, general purpose, prohibit Null value sync. Can be used for one time initial setup, thereafter server can be Master for an object.
 - Master can be set explicitly with `ArnItem::setMaster()`. This is overriden if the *ARN Data Object* has a Null value (not assigned), then the object becomes temporary Slave for next connection.
 - If client has an unsynced local update (during not connected state), this *ARN Data Object* becomes temporary Master for just next connection.
 - If the client is not Master for an *ARN Data Object* but the server only has a Null value, the clients value (non Null) is still used.
- **ImplicitMaster** First local assign gives permanent Master mode, typically a client value reporter.
 - Master can be set explicitly with `ArnItem::setMaster()`.
 - Client local assign to an *ARN Data Object* gives permanent Master mode for this object. This implicit Master mode setting is done once when next connection is established.
 - Null values can be synced both from client and server.
 - If a client *ARN Data Object* is set booth as `Persistent` and Master with a Null value before connection, the Master mode is initially overridden and the servers value is synced to the client.
- **ExplicitMaster** Explicit permanent Master mode, typically an observer or manually setup Master mode. Can be used for UI (User Interface) with no Master set to any *ARN Data Object*, i.e. the server is always holding the "true" value.
 - Master can be set explicitly with `ArnItem::setMaster()`. Client has no other way to become Master for an *ARN Data Object*.
 - Null values can be synced both from client and server.
 - If a client *ARN Data Object* is set booth as `Persistent` and Master with a Null value before connection, the Master mode is initially overridden and the servers value is synced to the client.

3.2 RPC and SAPI

[ArnRpc](#) is the basic functionality of RPC (Remote Procedure Call). [ArnSapi](#) implements SAPI (Service Application Programming Interface) and is using [ArnRpc](#) as its base. It's recommended to use [ArnSapi](#) which has a higher level model.

The SAPI works by a model which can be described as RPC by *remote signal slots*. The *provider* is usually assumed to wait for a *requester* to initiate the session and then react to different remote calls from the *requester*. However, this is full duplex, so any side can make a remote call at any time.

A good example of the usage of SAPI is the "Arn Demo Chat", which is included in the source package of the ArnLib.

[ArnRpc](#) uses [pipes](#) to communicate. The *pipes* can be monitored and receive test stimuli from the "Arn Browser" program. The used [protocol](#) is XString based and quite easy to handtype when common data types are used. "\$help" will give the syntax for the actual custom SAPI.

A SAPI is setup by deriving the [ArnSapi](#) class to a new class that defines the *custom SAPI*. This custom-declared class is included at both the *provider* and *requester* ends. The *custom SAPI* class by itself doesn't implement any *services*. It's merely a hub for connections to *external signals and slots*. The base [ArnSapi](#) class automatically transfers all *custom signal* (SAPI) calls to the remote connected ends, which also have the [ArnSapi](#) derived class and that emits the transferred signal. See example in [ArnSapi Detailed Description](#).

The provider connects the signals from custom SAPI that are prefixed with "pv_" (as default) to each external slot that implements the services. In the same way the *requester* connects the signals prefixed with "rq_" to its external "service" slots.

When there is a naming pattern between the *SAPI services* and the *external signals and slots*, it's a great convenience to use [ArnRpc::batchConnect\(\)](#), [ArnSapi::batchConnectTo\(\)](#) or [ArnSapi::batchConnectFrom\(\)](#). This saves a lot of `QObject::connect()` calls. Also newly added services in the SAPI, that obey the naming scheme, will automatically be connected to the newly matching *external signals and slots* for implementation of the *service*.

An extended feature comparing to normal *signals* is that the *SAPI signals* are *public* and can be called by non-derived classes. This makes it optional to use both *signal to signal* connections or direct *signal* calls (`emit`), when issuing a RPC to the remote side.

The *service* slot can get the emitting *custom SAPI* object by using normal `QObject::sender()` functionality.

3.2.1 RPC and SAPI method name overload

Under the hood Qt converts a signal that uses default argument(s) into methods with same name and all variation of the arguments. I.e. One method with all arguments, one with all but the last default argument, and so on until there is no more default arguments. When emitting the signal with some number of arguments, all of the signal methods will be exited.

[ArnRpc](#) has to deal with this default argument mechanism, otherwise there would be multiple calling messages for just one original signal emit.

The problem arises when there also can be normal signals that are overloaded, i.e. using same method name but different arguments. [ArnRpc](#) has to be able to differentiate between these normal overloaded signals and the default argument signals described earlier.

These are the alternatives, how you can help [ArnRpc](#) make your SAPI work:

- Don't overload arguments or make sure they don't have a common start of equal names and types. E.g. its ok with: `f(int a, int b); f(int b); f(int c); f(uint a);`
- Set [ArnRpc::Mode::NoDefaultArgs](#) and never use any default arguments in the SAPI. It's then ok to use any kind of normal overloading.

3.2.2 RPC and SAPI communication format

The RPC calling has a basic format as XString (see [Arn::XStringMap](#)). A call message can have 3 possible argument formats: positional, named and typed. The positional format is always possible to use and is most comparable to a standard c++ call.

The method name always come first in the message. After that comes arguments that have the argument data in the value part of its key/value pair. The key part can have the argument type and name, but this depends on the used argument format.

The following RPC data types are available:

RPC	Qt
int	int
uint	uint
int64	qint64
uint64	quint64
bool	bool
float	float
double	double
bytes	QByteArray
date	QDate
time	QTime
datetime	QDateTime
list	QStringList
string	QString

Also generic RPC data types can be formed as:

Textual like QColor t<QColor>
Binary like QPoint tb<QPoint>

Only textual types, i.e. those that can be converted to/from a string, are reasonable to be hand typed.

Lets have an example method to see the message when it is called.

Method: void put(QString id, int value);
Get called by: put("level", 123);

Alternatives in positional argument format:

```
put t<QString>.id=level t<int>.value=123
put string.id=level int.value=123
put string.=level int.=123
put string=level int=123
put level int=123
```

- Argument names are optional and only for human debugging.

- When no type is given, "string" is assumed.
- When `ArnRpc::Mode::NamedArg` is active, its not allowed to only use typename, e.g. "int=123" can be "int.=123" to enforce positional format.
- Both textual and binary arguments can be used.

Alternatives in named argument format:

```
put id=level value=123
put value=123 id=level
put value=123 dummy=ABC id=level garbage=321
```

- Only Argument names are used.
- Any order of arguments can be used.
- Extra arguments are discarded.
- If too few arguments, default constructor is used, e.g. "put value=123" will give id="".
- The methods parameter data type is used and only textual types are allowed.
- When `ArnRpc::Mode::NamedArg` is inactive, its not allowed to use an argument name that also is a RPC data type. See table above. E.g. "list" and "string" are not allowed.
- Only textual arguments can be used (as stated before).

Alternatives in typed argument format:

```
put id:t<QString>=level value:t<int>=123
put id:string=level value:int=123
put value:int=123 id:string=level
put value:int=123 dummy:bytes=ABC id:string=level
```

- Argument names and types are used.
- Only the name is used to match method parameter.
- The type is verified with the matching method parameter for error check.
- Any order of arguments can be used.
- Extra arguments are discarded.
- If too few arguments, default constructor is used, e.g. "put value:int=123" will give id="".
- Both textual and binary arguments can be used.

Named and typed argument format can be mixed, but positional format is never mixed.

List (QStringList) can be used. All examples below will get same resulting call.


```

For a function: void test( QStringList lst, int num)
test list=red green blu int=3
test list.lst=red green blu int.num=3
test list= +=red +=green +=blu int=3
test list=red +=green blu int=3
test lst:list=red green blu num=3
test num=3 lst:list=red green +=blu

```

- list is both a data type and a syntax for defining its data.
- list is only available for positional and typed argument format.

For special cases, like empty elements, the += syntax is needed. The example below has a first empty element followed by "green".

```
test list= += green blue int=2
```

The built-in call "\$help" will give an automatically generated list of the present SAPI with the syntax for each available service. The default argument format is positional. This can be changed to named format by giving "\$help named".

3.3 ZeroConfig

For getting a basic understanding of ZeroConfig and further references to relevant documentation, see: <http://zeroconf.org/>

ARN ZeroConfig is the lowest level support for advertising and discovering services on a local network. The implementation has very few dependencies to the rest of the ArnLib.

ARN ZeroConfig can use a built in implementation of Apple (R) *mDns / DNS_SD* that has no further dependencies to external libraries. For *mDns* the low end system abstraction layer has been written to use Qt for portability. The higher level *DNS_SD* has wrappers written to give a good c++ / Qt API.

It's also possible to use an external *DNS_SD* library, like *Avahi*. This gives better performance when many applications uses *ZeroConfig* on the same machine, as they share caching etc with a common daemon. However you have to deal with this external dependency.

ARN ZeroConfig implementation has two parts. The [ArnZeroConfRegister](#) can be used to advertise any *service* given a *host address* and a *port number*. The other part is the [ArnZeroConfBrowser](#) / [ArnZeroConfResolve](#) / [ArnZeroConfLookup](#). The browser is used to get a realtime list of available *services* on the network. The resolver takes a given *service* and resolves it into its *host name* and *port number*. Finally [ArnZeroConfLookup](#) takes a given *host name* and makes a DNS (mDNS) lookup to get its ip-address. Each of these classes are stand alone and has to be combined with glue logic for the complete process.

3.3.1 ZeroConfig definitions

A *ZeroConfig service* has a *service type*, that preferably should be registered at IANA. Examples of *service types* are "http", "ftp" and "arn". This type is mandatory when advertising a *service*. Also the *service* must have a *service name*.

3.3.1.1 Service name

Service names can be any human readable id. It should be easy to understand, without any cryptic coding. There should not be any attempts to make the *service name* unique as this is taken care of by the ZeroConfig system. It's common that the *service name* can be modified by the end user. The default starting name could be some system or product name. Example of *service name*: "My House Registry".

3.3.1.2 Sub types

Services can also have *sub types*. These are identifiers that can be used to filter out some sub group from a specific *service type*. All *services* having the same *service type* must still have some common protocol even if they belong to different *sub types*. A *service* can be advertised with many *sub types*, but browsing can only be filtered with one *sub type* or with no filter.

3.3.1.3 Text record

It's possible to add a *text record* to a *service*. The format of this record is specified by IANA. The purpose is to store properties by a *key / value* -pair. For convenience this can be done with [ArnZeroConfRegister::setTxtRecordMap\(\)](#) using an [Arn::XStringMap](#).

3.3.2 Discover

ARN Discover is the mid level support for advertising and discovering services on a local network. This implementation is only for the "arn" *service type* and is heavily dependent on the ArnLib. The "arn" *service type* is approved and registered by IANA.

ARN Discover implementation has two parts. The [ArnDiscoverAdvertise](#) can be used to advertise an Arn *service* given a *host address* and a *port number*. The other part is the [ArnDiscoverBrowser](#) / [ArnDiscoverResolver](#). The browser is used to get a realtime list of available Arn *services* on the network. The resolver is for taking a manual resolve when a *service name* is known in advance.

ARN Discover is designed to minimize external glue logic as these classes do all the common processing. Internally *ARN ZeroConfig* is used, but focus is on solving Arn specific needs in a powerful, yet flexible manner.

An *ARN service* needs an [ArnDiscover::Type](#) and a *service name*. The [ArnDiscover::Type](#) sets up a coarse division of the applications into the *groups* "server" and "client". The "client" typically only offer the service of [ArnDiscoverRemote](#).

ARN services can also have *groups*. These are identifiers that can be used to filter out some sub group. An *ARN service* can be advertised with many *groups*, but browsing can only be filtered with one *group* or with no filter.

It's possible to add a *custom property* to an *ARN service*. This can be done with [ArnDiscoverAdvertise::setCustomProperties\(\)](#) using an [Arn::XStringMap](#). The propertie has a *key / value* -pair. The custom property are advised to have a *key* starting with a capital letter to avoid name collision with the system. The added *groups* will be set as properties with naming as "group0", "group1" ...

[ArnDiscoverBrowser](#) collects found *Arn services*. Each of these *services* can automatically be further examined. This is chosen by calling [ArnDiscoverBrowserB::setDefaultStopState\(\)](#), which e.g. tells examination to stop after *host name* has been found. The *service* can then manually be ordered for further examination by [ArnDiscoverBrowserB::goTowardState\(\)](#), e.g. examination should now stop after *host ip* is found.

All the information about a *service* is stored in [ArnDiscoverInfo](#). Found *services* can be accessed by index, id or *service name*. Increasing index, starting at 0, gives a list of *services* alphabetically sorted by *service name*. The index is kind of volatile and should be used instantly, not be stored. The id gives a unique number for each service and can be stored. However the *service* given by the id might disappear.

3.3.3 Discover remote

ARN Discover Remote is the highest level support for advertising and discovering services on a local network. Its implementation is based on *ARN Discover*. The added functionality is to have a remote control for both advertising an [ArnServer](#) and multiple [ArnClient](#) connections. The remote control is done via *ARN Data Objects* in [local path](#) "Sys/Discover/".

ARN Discover Remote has one main class, [ArnDiscoverRemote](#) which act as a central point. The [ArnDiscoverRemote](#) class also takes an [ArnServer](#) and advertises it as a *service*. For remote control the *service name* is available at [local path](#) "Sys/Discover/This/Service/value".

[ArnDiscoverRemote](#) can make an internal [ArnServer](#), when there is no need to access the [ArnServer](#) class. This is usually the case in an client application. The [ArnServer](#) is then merely used to make the discover functionality remote controlled.

Remote controlled client connections can be added. Each [ArnClient](#) is handled by an [ArnDiscoverConnector](#) instance, which is made by [ArnDiscoverRemote::newConnector\(\)](#). Connections can be added to [ArnDiscoverConnector](#), both as a *direct host* list and a *discover host*.

The *discover host* is indirectly set, by adding an [ArnDiscoverResolver](#) to [ArnDiscoverConnector](#). A *service name* can then be resolved into the *discover host*.

The two connection methods can coexist and as standard the *discover host* has lower priority number than *direct host*, i.e. *discover host* is tried first.

The [ArnDiscoverConnector](#) is associated with an *id*, which should be chosen to describe the client target or its purpose. It's not a host address or necessarily a specific host, as there can be many possible addresses assigned to the [ArnDiscoverConnector](#).

The *id* will appear as an *ARN folder* in [local path](#), e.g. when *id* is "WeatherData-XYZ" the folder path will be "Sys/Discover/Connect/WeatherData-XYZ/". The folder and its sub folders will contain *ARN Data Objects* to remote control the [ArnClient](#). For a more comprehensive description of these objects, see [help discover description](#).

In the above example, a *discover host* can be remote controlled by setting the *service name* in [local path](#) "Sys/Discover/Connect/WeatherData-XYZ/DiscoverHost/Service/value", e.g. to "Region Weather XYZ".

Also in the above example, the first *direct host* can be remote controlled by setting the *host name* in [local path](#) "Sys/Discover/Connect/WeatherData-XYZ/DirectHosts/Host-0/value", e.g. to "localhost".

Normally it's wanted that any remote set values in the [local path](#) remains after power cycling. This is supported by the [Arn persist system](#).

Connecting via resolver uses the logic:

- If connection fails for a *discover host*, resolving is forced to be refreshed for the target *service name*. The Host for the *service name* might have changed since last resolved and doing a refresh can get the new *discover host*.
- If connection continues to fail for a *discover host*, refreshing the resolv will have a blocking time to avoid spamming the net. Typically this time is 30 seconds, but it can be changed by [ArnDiscoverConnector::setResolveRefreshTimeout\(\)](#).

3.4 Application notations

- If any graphics are used, Gui must be included.
- Qt4: For console application only using QImage, Windowing system can be off, like: `QApplication a(argc, argv, false);`
- Qt5: For console application needing QImage, use `QApplication a(argc, argv)` and start application with flags `"-platform offscreen"`.

Chapter 4

Installation and usage

4.1 Introduction

This software uses qmake to build all its components. qmake is part of a Qt distribution.

qmake reads project files, that contain the options and rules how to build a certain project. A project file ends with the suffix "*.pro". Files that end with the suffix "*.pri" are included by the project files and contain definitions, that are common for several project files.

To use a more automated config of ArnLib and related applications, the Qt Feature mechanism is used as default when available. To set the feature directory following can be executed once (in Linux):

```
qmake -set QMAKEFEATURES /usr/include/qtfeatures
```

When possible arnlib.prf and arnlib_config.pri is installed in selected directory. Applications using ArnLib can now at best e.g. use:

```
ARN += client CONFIG += arnlib
```

And all needed config for ArnLib is automatically loaded.

Local adaption to ArnLib.pro can be put in the file arnlib_local.pri If needed edit the *.pri / *.pro files to adjust them to your needs. Take care to select your deployment directories.

4.2 Documentation

The documentation is built by:

```
qmake  
make doc
```

ArnLib includes a class documentation, that is available in various formats:

- **Html files**

- **PDF document**

refman.pdf is built by:

```
cd doc/latex
make
```

- **Qt Compressed Help** (*.qch) for the Qt assistant or creator.

Load the doc/qthelp/arnlib.qch file into Qt Creator. Start Qt creator and go to Tools > Options, open up Help and Documentation. Click Add and browse for the qch file that was just created, then Apply. It's best to close Qt creator at this point, and restart it.

4.3 Building ArnLib

The software can be built both by command line and IDE (Qt Creator). When using IDE, don't forget the "make install" step.

4.3.1 A) Unix

```
qmake
make
make install
```

The easiest way of installing this library, is to let it be placed in a standard location for libraries and includes, e.g. /usr/lib and /usr/include/ArnInc. When using a shared library it's path has to be known to the run-time linker of your operating system. On Linux systems read "man ldconfig" (or google for it). Another option is to use the LD_LIBRARY_PATH (on some systems LIBPATH is used instead, on MacOSX it is called DYLD_LIBRARY_PATH) environment variable.

If you only want to check the library examples without installing something, you can set the LD_LIBRARY_PATH to the lib directory of your local build. it's also possible to compile the sources together by ArnLibCompile (see Using ArnLib below).

The examples is built this way:

```
cd examples/ArnDemoChat
qmake
make
```

4.3.2 B) Win32/MSVC

Has not been tested yet ...

Check that your Qt version has been built with MSVC - not with MinGW !

Please read the qmake documentation how to convert your *.pro files into your development environment.

For example MSVC with nmake:

```
qmake ArnLib.pro
nmake
nmake install
```

The examples is built this way:

```
cd examples\ArnDemoChat
qmake ArnDemoChat.pro
nmake
```

Windows doesn't like mixing of debug and release binaries.

In windows it's possible to install the dll files together with the application binary, as the application directory always is included in the search path for dll.

4.3.3 C) Win32/MinGW

Using Qt Creator for windows, will give you the needed tools for building a Qt project.

Check that your Qt version has been built with MinGW - not with MSVC !

Start a Shell, where Qt is initialized. (e.g. with "Programs->Qt by Trolltech ...->Qt 4.x.x Command Prompt"). Check if you can execute "make" or something like "mingw32-make".

```
qmake ArnLib.pro
make
make install
```

The examples is built this way:

```
cd examples\ArnDemoChat
qmake ArnDemoChat.pro
make
```

Windows doesn't like mixing of debug and release binaries.

In windows it's possible to install the dll files together with the application binary, as the application directory always is included in the search path for dll.

4.3.4 D) MacOSX

Has not been tested yet ...

Well, the Mac is only another Unix system. So read the instructions in A).

In the recent Qt4 releases the default target of qmake is to generate XCode project files instead of makefiles. So you might need to do the following:

```
qmake -spec macx-g++
```

4.3.5 E) Qt Embedded

ArnLib has been built with Qt Embedded using a Raspberry Pi. To build was as simple as for a regular Unix build.

4.4 Using ArnLib

In ArnLib the arnlib.prf -file contains template lines that can be used in the *.pro file of the application.

This will give a starting point for the configuration. It works well when using the same base directory for ArnLib as the application, e.g. basedir/ArnLib and basedir/myApp. In Unix-alike systems it's also needed to install the library files in a path known by the system, see a) Unix.

When Qt Features is used (default), the applications using ArnLib can e.g. use:

```
ARN += client CONFIG += arnlib
```

And all needed config for ArnLib is automatically loaded.

It's possible to include the ArnLib source in the application compiling by adding ArnLibCompile to CONFIG. The included part of the source can be selected by additions to ARN, e.g. ARN += server.

WARNING! Using source inclusion (static linking) excludes the right to use LGPL for ArnLib. Options are then to use GPL for the whole application or have a written agreement with Michael Wiklund for other terms using the ArnLib.

Internal mDNS (ZeroConfig) is selected by adding mDnsIntern to CONFIG.

If you don't use qmake you have to add the include path to find the ArnLib headers to your compiler flags and the ArnLib library to your linker list.

This Install.md file is based on documentation in the Qwt project.

Chapter 5

ArnLib Internals

This document describes internal processes that are relatively complex and by this needs some explanation.

5.1 ScriptJobs

- Each jobstack ScriptJobs is setup with a ScriptJobFactory wich makes custom interfaces etc.
- ScriptJobControl is setup with: Sriptfile, Config (QObject) and InterfaceList. Sriptfile is also copied to a [ArnItem](#).
- ScriptJobControl can be connected to update of script in [Arn](#), to make reload possible.
- Error text from ScriptJobControl can be connected to a pipe in [Arn](#) for logging.
- ScriptJobControl together with jobpriority define the ScriptJob and is added to ScriptJobs. Error text from Script job is connected to ScriptJobControl.
- Starting ScriptJobs in cooperative mode:
 1. Every ScriptJob is created and setup by corresponding ScriptJobControl
 2. Every ScriptJob is connected to Scheduler (yield etc).
 3. Every ScriptJobControl is connected to ScriptJobs for signaling update of script.
 4. Scheduler is started.
- Setup ScriptJob by ScriptJobControl:
 1. set ScriptJobFactory and Config
 2. Make and add the jobs Interfaces
 3. Evaluate the script (in js engine)
 4. run script function jobInit()
- Updating Script in cooperative mode:
 1. ScriptJobControl gets updated by [Arn](#) (or other).
 2. ScriptJobControl sends signal to ScriptJobs, which sets an updated flag for the corresponding Script Job.
 3. When scheduling, every updated script will get its sigQuit signal invoked and then reloaded.
 4. Reloading includes creating a new ScriptJob and setting up with ScriptJobControl etc.
- Starting ScriptJobs in preemitive mode:

1. Every ScriptJob gets its own thread which also is setup with ScriptJobControl and ScriptJobFactory.
 2. Thread is started and it create a ScriptJobSingle where following steps are done.
 3. ScriptJob is created and setup by ScriptJobControl
 4. ScriptJob is connected to Scheduler (yield etc).
 5. ScriptJobControl is connected to ScriptJobSingle for signaling update of script.
 6. Scheduler is started in ScriptJobSingle (just one job).
- Updating Script in preemptive mode:
 1. ScriptJobControl gets updated by [Arn](#) (or other).
 2. ScriptJobControl sends signal to ScriptJobSingle, which sets an updated flag and both invokes sigQuit signal to script and calls quit in scriptJob.
 3. ScriptJob aborts its js script engine and posts a custom Quit event with high prio.
 4. When ScriptJob get the Quit event, it will send a QuitRequest signal to ScriptJobSingle.
 5. ScriptJobSingle will get the signal and detect update flag, which means reloading.
 6. Reloading includes creating a new ScriptJob and setting up with ScriptJobControl etc.

5.2 ArnMonitor

- Monitor starts its actual connection job when its start method is called.
- Monitor (at client-side) results in creates an ItemNet with path to monitorPath.
- The ItemNet is also put in syncQueue (always main-thread).
- Monitor puts the arn-event "monitorStart" in event loop, which makes sure event is sent after Monitor (and its caller) has finished initializing.
- When "monitorStart" is received on local (client) side, the ItemNet will change SyncMode to Monitor. This will resync ItemNet to a Monitor at any server restart.
- Now 2 possibilities depending on threading:
 1. The ItemNet was sent before syncMode Monitor was set. Then server will receive an ordinary ItemNet and do standard setup.
 2. The ItemNet was sent with syncMode Monitor set. The server will detect this and do MonitorSetup on the ItemNet.
- When arn-event "monitorStart" is received on server-side, if SyncMode is not already set to "Monitor", server will do MonitorSetup on the ItemNet.
- When doing MonitorSetup (at server-side), logic are made to send arn-events when new childs are created, and present childs are directly sent as arn-event.

5.3 Destroy

- Destruction can be locally initiated and affects one link. Destruction can be set as local or global.
- Destruction can also be initiated for leaves by the destroy command and arrives with a netId. Or it can be with the delete command for a folder (tree).
- For leaf, corresponding ItemNet is disabled (set as defunct), which prohibit sending destroy command back to the originator of the command.

- The ItemNet is also destroyed in the same way as a locally initiated destruction and affects one link. Destruction is set to be global.
- The affected link:s tree is recursively traversed and all links are first marked as retired. Also the retire type is set as LeafLocal, LeafGlobal or Tree.
- As the last thing in this recursion each link is sending a Retired [ArnEvent](#), ie the leaves are the first to send. The event is sent to the subscriptions (ArnBasicItems or derived) of each link.
- If it's a destroy of a tree (folder), a Retired [ArnEvent](#) is also sent to the tree:s parent and all the way up to the root. The event is sent to the subscriptions (ArnBasicItems) of each link. These events have a marking telling destroy is below.
- The Retired [ArnEvent](#) is handled by each subscribing Item. For [ArnBasicItem](#) this is done by its eventhandler, which by default is an internal handler. For [ArnItem](#) this is done by sending a linkDestroyed signal to be handled by application code. The Items is finally closed and by this the link ref counter is decremented.
- When the links ref counter is reaching zero, a ZeroRef [ArnEvent](#) is sent. Also a ZeroRef pending counter is increased.
- The event is handled by ArnM::doZerRefLink(), in Main thread. First the ZeroRef pending counter is decreased. Next both ref counter and ZeroRef pending counter is checked to be zero, which indicates that this is the final ZeroRef for this link. This is to prohibit a scenario where the link has been reused during ZeroRef [ArnEvent](#) delivery. Also this reuse might have been followed by a dropped usage resulting in a second ZeroRef [ArnEvent](#).
- In ArnM::doZerRefLink() if this is the final ZeroRef, it will set the link ref counter to -1, to mark the link as fully de-referenced. The link and parent (and grand parents ...) are deleted if they don't have any children and ref = -1 and they are marked retired.
- When the ArnSync, which is eventHandler for ItemNet, is handling the Retired [ArnEvent](#), it will delete the corresponding ItemNet from sync map and all queues. Finally a command can be sent with its netId.
- The sent command depends on retire type. For Leaflocal, a nosync command is used. For LeafGlobal, a delete command is used to spread the destruction to server and other clients. The Tree type doesn't send a command at item level.
- For tree destroy, [ArnClient](#) is using a monitoring ArnItemNetEar at each mount point to catch the Retire [ArnEvent](#) for a tree below. Such an event is resulting in a delete or noSync command is sent, depending on global or local destroy. The command is sent with the path to the destroyed tree.
- For tree destroy, [ArnServer](#) is using a monitoring ArnItemNetEar at root to catch the Retire [ArnEvent](#) for a tree below. Such an event is resulting in a delete command is sent. The command is sent with the path to the destroyed tree.
- When a delete command is echoed back to the originator, it will stop with this only echo as the affected tree is already marked for retire and this will terminate the command.

Chapter 6

Example Collection

Here are some examples showing the use of the ArnLib described in this documentation.

- [Chat Demo](#)

6.1 Chat Demo

Demonstration with a simple chat program. It consists of a server and a client part. After starting the server, any number of clients can be started.

This demo is focused on the *Service API* (RPC) functionality of ArnLib. Slots are remotely called from clients to server and the other way back. All is done with standard function calls without any visual serializing.

It's also a demo of *Discover Remote*, althou client side is as simple as possible without any remote control.

Chat Server** [ChatSapi.hpp](#), [MainWindow.hpp](#), [MainWindow.cpp](#), [main.cpp](#)

Chat Client** [MainWindow.hpp](#), [MainWindow.cpp](#), [main.cpp](#)

6.1.1 Chat Server

6.1.1.1 ChatSapi.hpp

```
#ifndef CHATSAPI_HPP
#define CHATSAPI_HPP

#include <ArnInc/ArnSapi.hpp>

class ChatSapi : public ArnSapi
{
    Q_OBJECT
public:
    explicit ChatSapi( QObject* parent = 0 ) : ArnSapi( parent) {}

signals:
    MQ_PUBLIC_ACCESS
    no_queue void pv_list();
    void pv_newMsg( QString name, QString msg);
    void pv_infoQ();

    void rq_updateMsg( int seq, QString name, QString msg);
    void rq_info( QString name, QString ver);
};

#endif // CHATSAPI_HPP
```

6.1.1.2 MainWindow.hpp

```

#ifndef MAINWINDOW_HPP
#define MAINWINDOW_HPP

#include "ChatSapi.hpp"
#include <ArnInc/ArnItem.hpp>
#include <ArnInc/ArnServer.hpp>
#include <QTimer>
#include <QStringList>
#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class ArnDiscoverRemote;

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow( QWidget *parent = 0);
    ~MainWindow();

private slots:
    void doNewSession( QString path);
    void doSessionClosed();
    void doUpdateView();
    void on_shutDownButton_clicked();
    void doTimeUpdate();

    void sapiList();
    void sapiNewMsg( QString name, QString msg);
    void sapiInfoQ();
    void sapiDefault( const QByteArray& data);

private:
    Ui::MainWindow *_ui;
    QStringList _chatNameList;
    QStringList _chatMsgList;
    QTimer _timer1s;
    int _connectCount;

    ArnItem _arnTime;
    ArnServer* _server;
    ChatSapi* _commonSapi;
    ArnDiscoverRemote* _discoverRemote;
};

#endif // MAINWINDOW_HPP

```

6.1.1.3 MainWindow.cpp

```

#include "MainWindow.hpp"
#include "tmp/ui_MainWindow.h"
#include <ArnInc/ArnItem.hpp>
#include <ArnInc/ArnDiscoverRemote.hpp>
#include <QTime>
#include <QDebug>

MainWindow::MainWindow( QWidget *parent) :
    QMainWindow( parent, Qt::CustomizeWindowHint | Qt::WindowMinimizeButtonHint),
    _ui( new Ui::MainWindow)
{
    _ui->setupUi( this);
    _connectCount = 0;
    doUpdateView();

    _timer1s.start(1000);
    connect( &_amp;_timer1s, SIGNAL(timeout()), this, SLOT(doTimeUpdate()));

    _server = new ArnServer( ArnServer::Type::NetSync, this);
    _server->start(0); // Start server on dynamic port

    _discoverRemote = new ArnDiscoverRemote( this);
    _discoverRemote->setService("Demo Chat Server");
    _discoverRemote->addGroup("arndemo/chat");
    _discoverRemote->addCustomProperty("ChatProtoVer", "1.0");
    _discoverRemote->startUseServer( _server);

```

```

    _arnTime.open("//Chat/Time/value");

    typedef ArnSapi::Mode SMode;
    _commonSapi = new ChatSapi( this);
    _commonSapi->open("//Chat/Pipes/pipeCommon", SMode::Provider | SMode::UseDefaultCall);
    _commonSapi->batchConnectTo( this, "sapi");

    ArnItem* arnPipes = new ArnItem("//Chat/Pipes/", this);
    connect( arnPipes, SIGNAL(arnItemCreated(QString)), this, SLOT(doNewSession(QString)));
}

MainWindow::~MainWindow()
{
    delete _ui;
}

void MainWindow::doNewSession( QString path)
{
    if (!Arn::isProviderPath( path)) return; // Only provider pipe is used

    typedef ArnSapi::Mode SMode;
    ChatSapi* soleSapi = new ChatSapi( this);
    soleSapi->open( path, SMode::Provider | SMode::UseDefaultCall);
    soleSapi->batchConnectTo( this, "sapi");
    connect( soleSapi, SIGNAL(pipeClosed()), soleSapi, SLOT(deleteLater()));

    connect( soleSapi, SIGNAL(pipeClosed()), this, SLOT(doSessionClosed()));
    ++_connectCount;
    doUpdateView();
}

void MainWindow::doSessionClosed()
{
    --_connectCount;
    doUpdateView();
}

void MainWindow::doUpdateView()
{
    _ui->connectCount->setText( QString::number( _connectCount));
}

void MainWindow::on_shutDownButton_clicked()
{
    qWarning() << "About to shut down.";
    delete _discoverRemote; // Must be deleted while still in the main eventloop
    _discoverRemote = 0;
    QApplication::quit();
}

void MainWindow::doTimeUpdate()
{
    _arnTime = QTime::currentTime().toString();
}

void MainWindow::sapiList()
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    for (int i = 0; i < _chatNameList.size(); ++i) {
        sapi->rq_updateMsg( i, _chatNameList.at(i), _chatMsgList.at(i));
    }
}

void MainWindow::sapiNewMsg( QString name, QString msg)
{
    _chatNameList += name;
    _chatMsgList += msg;
    int seq = _chatNameList.size() - 1;

    _commonSapi->rq_updateMsg( seq, name, msg);
}

void MainWindow::sapiInfoQ()
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
}

```

```

    sapi->rq_info("Arn Chat Demo", "1.2");
}

void MainWindow::sapiDefault( const QByteArray& data)
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    qDebug() << "chatDefault:" << data;
    sapi->sendText("Chat Sapi: Can't find method, use $help.");
}

```

6.1.1.4 main.cpp

```

#include "MainWindow.hpp"
#include <QApplication>
#include <QDebug>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}

```

6.1.2 Chat Client

6.1.2.1 MainWindow.hpp

```

#ifndef MAINWINDOW_HPP
#define MAINWINDOW_HPP

#include "../ArnDemoChatServer/ChatSapi.hpp"
#include <ArnInc/ArnClient.hpp>
#include <ArnInc/ArnItem.hpp>
#include <QMainWindow>
#include <QVector>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow( QWidget *parent = 0);
    ~MainWindow();

private slots:
    void doSendLine();
    void doTimeUpdate( QString timeStr);

    void sapiUpdateMsg( int seq, QString name, QString msg);
    void sapiInfo( QString name, QString ver);

private:
    Ui::MainWindow *_ui;
    QVector<QString> _chatNameList;
    QVector<QString> _chatMsgList;

    ArnClient _arnClient;
    ChatSapi _commonSapi;
    ChatSapi _soleSapi;
    ArnItem _arnTime;
};

#endif // MAINWINDOW_HPP

```


6.1.2.2 MainWindow.cpp

```

#include "MainWindow.hpp"
#include "tmp/ui_MainWindow.h"
#include <ArnInc/ArnDiscoverRemote.hpp>

MainWindow::MainWindow( QWidget* parent) :
    QMainWindow( parent),
    _ui( new Ui::MainWindow)
{
    _ui->setupUi( this);
    _ui->userEdit->setFocus();
    connect( _ui->lineEdit, SIGNAL(returnPressed()), this, SLOT(doSendLine()));

    _arnClient.addMountPoint("/");
    _arnClient.setAutoConnect(true);

    ArnDiscoverConnector* connector = new
    ArnDiscoverConnector( _arnClient, "DemoChat");
    connector->setResolver( new ArnDiscoverResolver());
    connector->setService("Demo Chat Server");
    connector->start();

    _arnTime.open("/Chat/Time/value");
    connect( &_arnTime, SIGNAL(changed(QString)), this, SLOT(doTimeUpdate(QString)));

    _commonSapi.open("/Chat/Pipes/pipeCommon");
    _commonSapi.batchConnectTo( this, "sapi");

    _soleSapi.open("/Chat/Pipes/pipe", ArnSapi::Mode::UuidAutoDestroy);
    _soleSapi.batchConnectTo( this, "sapi");

    _soleSapi.pv_infoQ();
    _soleSapi.pv_list();
}

MainWindow::~MainWindow()
{
    delete _ui;
}

void MainWindow::doTimeUpdate( QString timeStr)
{
    _ui->timeEdit->setTime( QTime::fromString( timeStr));
}

void MainWindow::doSendLine()
{
    QString myName = _ui->userEdit->text();
    QString line = _ui->lineEdit->text();
    _ui->lineEdit->clear();

    _soleSapi.pv_newMsg( myName, line);
}

void MainWindow::sapiUpdateMsg( int seq, QString name, QString msg)
{
    if (seq >= _chatNameList.size()) {
        _chatNameList.resize( seq + 1);
        _chatMsgList.resize( seq + 1);
    }
    _chatNameList[ seq] = name;
    _chatMsgList[ seq] = msg;

    QString text;
    for (int i = 0; i < _chatNameList.size(); ++i) {
        text += _chatNameList.at(i) + ": " + _chatMsgList.at(i) + "\n";
    }
    _ui->textEdit->setText( text);
}

void MainWindow::sapiInfo( QString name, QString ver)
{
    _ui->appNameLabel->setText( name);
    _ui->verLabel->setText( ver);
}

```

6.1.2.3 main.cpp

```
#include "MainWindow.hpp"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

6.1.3 Pictures

Chapter 7

Help descriptions

Here are some help descriptions included in ArnLib

- [Discover](#)

7.1 Discover

The "parameter path" in the table have stripped the "value" attribute, e.g. "Service/value".

7.1.1 Description

Chapter 8

Deprecated List

Member [ArnClient::setMountPoint](#) (const QString &path)

Use [addMountPoint\(\)](#) and [removeMountPoint\(\)](#)

Member [ArnItem::arnItemCreated](#) (const QString &path)

use [ArnMonitor](#) instead.

Member [ArnItem::arnModeChanged](#) (const QString &path, uint linkId, [Arn::ObjectMode](#) mode)

use [ArnMonitor](#) instead.

Member [ArnMonitor::setMonitorPath](#) (const QString &path, [ArnClient](#) *client=arnNullptr)

Use [start\(\)](#) instead, *client* parameter is changed.

Member [ArnRpc::setIncludeSender](#) (bool v)

Use [rpcSender\(\)](#)

Chapter 9

Namespace Index

9.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Arn	53
ArnDiscover	70
ArnZeroConf	70

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

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

Arn::_InitEnumTxt	71
Arn::_InitSubEnum	72
Arn::Allow	73
ArnAtomicOp	107
ArnClientConnectStat	160
ArnClientReg	161
ArnCoreItem	163
ArnBasicItem	108
ArnAdaptItem	74
ArnItemB	299
ArnItem	262
ArnItemQml	302
ArnItemValve	310
ArnMonitor	334
ArnMonitorQml	346
ArnPipe	355
ArnDiscoverInfo	204
ArnError	223
ArnEventIdx	232
ArnLinkValue	316
ArnMonEventType	333
ArnNullptr	349
ArnRpcMode	383
ArnScriptJobB	
ArnScriptJob	400
ArnScriptJobFactory	407
ArnServerRemoteSessionKillMode	422
Arn::ClientSyncMode	470
Arn::Coding	471
Arn::DataType	471
Arn::EnumTxt	472
ArnZeroConf::Error	498
Arn::ExportCode	499
ArnCoreItem::Heritage	500
ArnClient::HostAddrPort	500

Arn::EnumTxt::IncludeMode	501
Arn::InfoType	502
ArnRpc::Invoke	503
Arn::LinkFlags	504
Arn::NameF	510
Arn::ObjectMode	511
Arn::ObjectSyncMode	511
ArnRpc::MethodsParam::Params	512
QBasicTimer	
MQBasicTimer	506
QEvent	
ArnEvent	228
ArnEvAtomicOp	224
ArnEvLinkCreate	234
ArnEvModeChange	237
ArnEvMonitor	239
ArnEvRefChange	242
ArnEvRetired	244
ArnEvValueChange	247
ArnEvZeroRef	250
QGenericArgument	
MQGenericArgument	508
MQArgument< T >	505
QML_PARSER_STATUS	
Arn::QmlMQtObject	517
Arn::XStringMapQml	553
ArnItemQml	302
ArnMonitorQml	346
ArnSapiQml	390
QObject	
Arn::QmlMFileIO	513
Arn::QmlMQtObject	517
Arn::QmlMSys	521
Arn::XStringMapQml	553
ArnClient	139
ArnDepend	165
ArnDependOffer	169
ArnDiscoverAdvertise	173
ArnDiscoverRemote	213
ArnDiscoverBrowserB	187
ArnDiscoverBrowser	182
ArnDiscoverResolver	219
ArnDiscoverConnector	195
ArnInterface	252
ArnItemB	299
ArnM	318
ArnPersist	350
ArnQml	364
ArnRpc	368
ArnSapi	384
ArnSapiQml	390
ArnScript	394
ArnScriptJobControl	403
ArnScriptJobs	409
ArnServer	411
ArnServerRemote	419
ArnServerRemoteSession	421
ArnServerSession	423

ArnZeroConfB	426
ArnZeroConfBrowser	431
ArnZeroConfLookup	441
ArnZeroConfRegister	448
ArnZeroConfResolve	461
Arn::SameValue	522
ArnDiscoverInfo::State	523
ArnZeroConf::State	524
ArnDiscoverAdvertise::State	525
ArnError::StdCode	526
ArnItemValve::SwitchMode	527
ArnServer::Type	527
ArnScriptJobs::Type	528
ArnDiscover::Type	529
ArnQml::UseFlags	530
Arn::XStringMap	530
Arn::XStringMapQml	553
Arn::XStringMapOptions	552

Chapter 11

Class Index

11.1 Class List

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

Arn::_InitEnumTxt	71
Arn::_InitSubEnum	72
Arn::Allow	73
ArnAdaptItem ! Non Qt and threadsafe handle for an <i>Arn Data Object</i>	74
ArnAtomicOp	107
ArnBasicItem Base class handle for an <i>Arn Data Object</i>	108
ArnClient Class for connecting to an <i>Arn Server</i>	139
ArnClientConnectStat	160
ArnClientReg	161
ArnCoreItem Core base class for the inherited <i>ArnItem</i> classes	163
ArnDepend Class for setting up dependencis to needed services	165
ArnDependOffer Class for advertising that a <i>service</i> is available	169
ArnDiscoverAdvertise Advertise an <i>Arn</i> service	173
ArnDiscoverBrowser Browsing for <i>Arn</i> services	182
ArnDiscoverBrowserB Browse() and resolve() together, may never be used to the same instance	187
ArnDiscoverConnector An automatic client discover connector	195
ArnDiscoverInfo Class for holding current discover info of one service	204
ArnDiscoverRemote Discover with remote setting	213
ArnDiscoverResolver Resolv an <i>Arn</i> service	219
ArnError	223
ArnEvAtomicOp	224
ArnEvent	228

ArnEventIdx	232
ArnEvLinkCreate	234
ArnEvModeChange	237
ArnEvMonitor	239
ArnEvRefChange	242
ArnEvRetired	244
ArnEvValueChange	247
ArnEvZeroRef	250
ArnInterface	252
ArnItem	
Handle for an <i>Arn Data Object</i>	262
ArnItemB	
Base class handle for an <i>Arn Data Object</i>	299
ArnItemQml	
ARN Item QML	302
ArnItemValve	
Valve for controlling stream to/from an ArnItemB	310
ArnLinkValue	316
ArnM	
<i>Arn</i> main class	318
ArnMonEventType	333
ArnMonitor	
A client remote monitor to detect changes at server	334
ArnMonitorQml	
ARN Monitor QML	346
ArnNullptr	349
ArnPersist	
Class for handling persistent <i>Arn Data object</i>	350
ArnPipe	
ArnItem specialized as a pipe	355
ArnQml	
ARN QML	364
ArnRpc	
Remote Procedure Call	368
ArnRpcMode	383
ArnSapi	
Service API	384
ArnSapiQml	
ARN Sapi QML	390
ArnScript	394
ArnScriptJob	
Interface class to be normally used, is also Script Job interface	400
ArnScriptJobControl	
Is thread-safe (except doSetupJob)	403
ArnScriptJobFactory	
Must be thread-safe as subclassed	407
ArnScriptJobs	409
ArnServer	
Class for making an <i>Arn Server</i>	411
ArnServerRemote	
Class for remote controlling an <i>Arn Server</i>	419
ArnServerRemoteSession	421
ArnServerRemoteSessionKillMode	422
ArnServerSession	423
ArnZeroConfB	
Base class for Zero Config	426
ArnZeroConfBrowser	
Browsing for ZeroConfig services	431

ArnZeroConfLookup	
Lookup a host	441
ArnZeroConfRegister	
Registering a ZeroConfig service	448
ArnZeroConfResolve	
Resolv a ZeroConfig service	461
Arn::ClientSyncMode	
The Client session Sync mode at connect & reconnect	470
Arn::Coding	471
Arn::DataType	
Data type of an <i>Arn Data Object</i>	471
Arn::EnumTxt	
Class Enum text	472
ArnZeroConf::Error	
Errors of ZeroConfig, other values are defined in dns_sd.h	498
Arn::ExportCode	
Code used in blob for arnExport() and arnImport()	499
ArnCoreItem::Heritage	500
ArnClient::HostAddrPort	500
Arn::EnumTxt::IncludeMode	501
Arn::InfoType	
Info type for exchange static (meta) info between ArnClient and ArnServer	502
ArnRpc::Invoke	503
Arn::LinkFlags	
Link flags when accessing an <i>Arn Data Object</i>	504
MQArgument< T >	
Similar to QArgument but with added argument label (parameter name)	505
MQBasicTimer	506
MQGenericArgument	
Similar to QGenericArgument but with added argument label (parameter name)	508
Arn::NameF	510
Arn::ObjectMode	511
Arn::ObjectSyncMode	511
ArnRpc::MethodsParam::Params	512
Arn::QmlMFileIO	513
Arn::QmlMQObject	517
Arn::QmlMSys	521
Arn::SameValue	
Action when assigning same value to an ArnItem	522
ArnDiscoverInfo::State	
State of Arn discover browse data. Can be tested by relative order	523
ArnZeroConf::State	
States of ZeroConfig, limited valid for each ArnZeroConfB subclass / These values must be synced with: ArnDiscover::State	524
ArnDiscoverAdvertise::State	
States of DiscoverAdvertise / These values must be synced with: ArnZeroConf::State	525
ArnError::StdCode	526
ArnItemValve::SwitchMode	527
ArnServer::Type	527
ArnScriptJobs::Type	528
ArnDiscover::Type	
Types of Arn discover advertise	529
ArnQml::UseFlags	530
Arn::XStringMap	
Container class with string representation for serialized data	530
Arn::XStringMapOptions	552
Arn::XStringMapQml	553

Chapter 12

File Index

12.1 File List

Here is a list of all files with brief descriptions:

src/Arn.cpp	559
src/ArnAdaptItem.cpp	561
src/ArnBasicItem.cpp	562
src/ArnClient.cpp	563
src/ArnCompat.cpp	563
src/ArnCoreItem.cpp	564
src/ArnDepend.cpp	564
src/ArnDiscover.cpp	565
src/ArnDiscoverConnect.cpp	566
src/ArnDiscoverRemote.cpp	566
src/ArnEvent.cpp	566
src/ArnItem.cpp	616
src/ArnItemB.cpp	617
src/ArnItemNet.cpp	617
src/ArnItemNet.hpp	618
src/ArnItemValve.cpp	618
src/ArnLib.cpp	619
src/ArnLink.cpp	620
src/ArnLink.hpp	620
src/ArnLinkHandle.cpp	622
src/ArnM.cpp	622
src/ArnMath.cpp	623
src/ArnMonitor.cpp	624
src/ArnPersist.cpp	624
src/ArnPipe.cpp	625
src/ArnQml.cpp	625
src/ArnQmlMQt.cpp	626
src/ArnQmlMSystem.cpp	626
src/ArnRpc.cpp	627
src/ArnSapi.cpp	627
src/ArnScript.cpp	628
src/ArnScriptJob.cpp	628
src/ArnScriptJobs.cpp	629
src/ArnServer.cpp	630
src/ArnServerRemote.cpp	630

src/ArnSync.cpp	631
src/ArnSync.hpp	631
src/ArnSyncLogin.cpp	633
src/ArnSyncLogin.hpp	633
src/ArnXStringMap.cpp	634
src/ArnZeroConf.cpp	634
src/MQFlags.cpp	635
src/ArnInc/Arn.hpp	567
src/ArnInc/ArnAdaptItem.hpp	570
src/ArnInc/ArnBasicItem.hpp	571
src/ArnInc/ArnClient.hpp	572
src/ArnInc/ArnCompat.hpp	572
src/ArnInc/ArnCoreItem.hpp	575
src/ArnInc/ArnDepend.hpp	575
src/ArnInc/ArnDiscover.hpp	576
src/ArnInc/ArnDiscoverConnect.hpp	578
src/ArnInc/ArnDiscoverRemote.hpp	579
src/ArnInc/ArnError.hpp	579
src/ArnInc/ArnEvent.hpp	580
src/ArnInc/ArnInterface.hpp	581
src/ArnInc/ArnItem.hpp	582
src/ArnInc/ArnItemB.hpp	583
src/ArnInc/ArnItemValve.hpp	584
src/ArnInc/ArnLib.hpp	585
src/ArnInc/ArnLib_global.hpp	586
src/ArnInc/ArnLinkHandle.hpp	587
src/ArnInc/ArnM.hpp	587
src/ArnInc/ArnMonEvent.hpp	588
src/ArnInc/ArnMonitor.hpp	589
src/ArnInc/ArnPersist.hpp	590
src/ArnInc/ArnPersistSapi.hpp	591
src/ArnInc/ArnPipe.hpp	592
src/ArnInc/ArnQml.hpp	593
src/ArnInc/ArnQmlMQt.hpp	595
src/ArnInc/ArnQmlMSystem.hpp	596
src/ArnInc/ArnRpc.hpp	597
src/ArnInc/ArnSapi.hpp	599
src/ArnInc/ArnScript.hpp	600
src/ArnInc/ArnScriptJob.hpp	602
src/ArnInc/ArnScriptJobs.hpp	603
src/ArnInc/ArnServer.hpp	604
src/ArnInc/ArnServerRemote.hpp	605
src/ArnInc/ArnZeroConf.hpp	606
src/ArnInc/Math.hpp	608
src/ArnInc/MQFlags.hpp	609
src/ArnInc/MQFlagsBase.hpp	613
src/ArnInc/XStringMap.hpp	615

Chapter 13

Namespace Documentation

13.1 Arn Namespace Reference

Classes

- struct [_InitEnumTxt](#)
- struct [_InitSubEnum](#)
- class [Allow](#)
- struct [ClientSyncMode](#)
 - The Client session Sync mode at connect & reconnect.*
- struct [Coding](#)
- class [DataType](#)
 - Data type of an [Arn](#) Data Object*
- class [EnumTxt](#)
 - Class Enum text.*
- class [ExportCode](#)
 - Code used in blob for [arnExport\(\)](#) and [arnImport\(\)](#)*
- struct [InfoType](#)
 - Info type for exchange static (meta) info between [ArnClient](#) and [ArnServer](#).*
- struct [LinkFlags](#)
 - Link flags when accessing an [Arn](#) Data Object*
- struct [NameF](#)
- class [ObjectMode](#)
- class [ObjectSyncMode](#)
- class [QmlMFileIO](#)
- class [QmlMQtObject](#)
- class [QmlMSys](#)
- struct [SameValue](#)
 - Action when assigning same value to an [ArnItem](#).*
- class [XStringMap](#)
 - Container class with string representation for serialized data.*
- class [XStringMapOptions](#)
- class [XStringMapQml](#)

Functions

- QString [convertName](#) (const QString &name, [Arn::NameF](#) nameF=[Arn::NameF\(\)](#))
Convert a name to a specific format.
- QString [fullPath](#) (const QString &path)
Convert a path to a full absolute path.
- QString [itemName](#) (const QString &path)
The last part of a path
- QString [childPath](#) (const QString &parentPath, const QString &posterityPath)
Get substring for child from a path (posterityPath)
- QString [changeBasePath](#) (const QString &oldBasePath, const QString &newBasePath, const QString &path)
Change the base (start) of a path.
- QString [makePath](#) (const QString &parentPath, const QString &itemName)
Make a path from a parent and an item name.
- QString [addPath](#) (const QString &parentPath, const QString &childRelPath, [Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#))
Make a path from a parent and an additional relative path.
- QString [convertPath](#) (const QString &path, [Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#))
Convert a path to a specific format.
- QString [parentPath](#) (const QString &path)
Get the parent to a given path
- QString [twinPath](#) (const QString &path)
Get the bidirectional twin to a given path
- QString [providerPathIf](#) (const QString &path, bool giveProviderPath=true)
Get provider path or requester path
- bool [isFolderPath](#) (const QString &path)
Test if path is a folder path
- bool [isProviderPath](#) (const QString &path)
Test if path is a provider path
- QString [uuidPath](#) (const QString &path)
Get a path to an [Arn](#) Object with a unique uuid name.
- QString [makeHostWithInfo](#) (const QString &host, const QString &info)
Make a combined host and info string, i.e. [HostWithInfo](#)
- QString [hostFromHostWithInfo](#) (const QString &hostWithInfo)
Get the host from the [HostWithInfo](#) string.
- bool [isNullPtr](#) (const void *ptr)
- uint [rand](#) ()
- int [_mod_i](#) (int x, int y)
- qlonglong [_mod_ll](#) (qlonglong x, qlonglong y)
- int [_log2_u](#) (uint x)
- int [_log2_ull](#) (qulonglong x)
- template<typename T >
T [mod](#) (T x, T y)
- template<typename T >
T [circVal](#) (T x, T lo, T hi)
- template<typename T >
bool [isPower2](#) (T x)
- template<typename T >
int [log2](#) (T x)
- template<typename T >
T [minLim](#) (const T &x, const T &lim)
- template<typename T >
T [maxLim](#) (const T &x, const T &lim)
- template<typename T >
T [rangeLim](#) (const T &x, const T &min, const T &max)

Variables

- const QString `pathLocal` = "/Local/"
- const QString `pathLocalSys` = "Sys/"
- const QString `pathDiscover` = "Sys/Discover/"
- const QString `pathDiscoverThis` = "Sys/Discover/This/"
- const QString `pathDiscoverConnect` = "Sys/Discover/Connect/"
- const QString `pathServer` = "Sys/Server/"
- const QString `pathServerSessions` = "Sys/Server/Sessions/"
- bool `debugSizes` = false
- bool `debugThreading` = false
- bool `debugLinkRef` = false
- bool `debugLinkDestroy` = false
- bool `debugReclnOut` = false
- bool `debugShareObj` = false
- bool `debugMonitor` = false
- bool `debugMonitorTest` = false
- bool `debugRPC` = false
- bool `debugDepend` = false
- bool `debugQmlNetwork` = false
- bool `debugDiscover` = false
- bool `debugZeroConf` = false
- bool `debugMDNS` = false
- bool `warningMDNS` = false
- bool `offHeartbeat` = false
- const QString `resourceArnLib` = ":/ArnLib/"
- const QString `resourceArnRoot` = ":/ArnLib/ArnRoot/"
- const quint16 `defaultTcpPort` = 2022

13.1.1 Function Documentation

13.1.1.1 `_log2_u()`

```
int Arn::_log2_u (
    uint x )
```

Definition at line 86 of file ArnMath.cpp.

13.1.1.2 `_log2_ull()`

```
int Arn::_log2_ull (
    qulonglong x )
```

Definition at line 98 of file ArnMath.cpp.

13.1.1.3 `_mod_i()`

```
int Arn::_mod_i (
    int x,
    int y )
```

Definition at line 74 of file ArnMath.cpp.

13.1.1.4 `_mod_ll()`

```
qlonglong Arn::_mod_ll (
    qlonglong x,
    qlonglong y )
```

Definition at line 80 of file ArnMath.cpp.

13.1.1.5 `addPath()`

```
QString Arn::addPath (
    const QString & parentPath,
    const QString & childRelPath,
    Arn::NameF nameF = Arn::NameF::EmptyOk )
```

Make a path from a parent and an additional relative path.

parentPath don't have to end with a "/", if missing it's added.

Example: *parentPath* = "//Measure/", *childRelPath* = "depth/value" ==> return = "//Measure/depth/value"

Parameters

in	<i>parentPath</i>	
in	<i>childRelPath</i>	
in	<i>nameF</i>	is the path naming format

Returns

The *path*

See also

[convertPath\(\)](#)

Definition at line 137 of file Arn.cpp.

13.1.1.6 `changeBasePath()`

```
QString Arn::changeBasePath (
    const QString & oldBasePath,
    const QString & newBasePath,
    const QString & path )
```

Change the base (start) of a path.

oldBasePath and *newBasePath* don't have to end with a "/", if missing it's added. If *path* not starts with *oldBasePath*, *path* is returned. Otherwise the path is returned with its base changed from *oldBasePath* to *newBasePath*.

Example: *path* = "//Measure/depth/value", *oldBasePath* = "//Measure/", *newBasePath* = "/Measure/Tmp/" ==> return = "/Measure/Tmp/depth/value"

Parameters

in	<i>oldBasePath</i>	
in	<i>newBasePath</i>	
in	<i>path</i>	

Returns

The changed path

Definition at line 114 of file Arn.cpp.

13.1.1.7 `childPath()`

```
QString Arn::childPath (
    const QString & parentPath,
    const QString & posterityPath )
```

Get substring for child from a path (*posterityPath*)

parentPath don't have to end with a "/", if missing it's added.

If *posterityPath* not starts with *parentPath*, `QString()` is returned. Otherwise given the *posterityPath* the child to *parentPath* is returned.

Example 1: *posterityPath* = "//Measure/depth/value", *parentPath* = "//Measure/" ==> return = "//Measure/depth/"

Example 2: *posterityPath* = "//Measure/depth/value", *parentPath* = "//Measure/depth/" ==> return = //↔ Measure/depth/value"

Parameters

in	<i>parentPath</i>	
in	<i>posterityPath</i>	

Returns

The *child path*

Definition at line 100 of file Arn.cpp.

13.1.1.8 circVal()

```
template<typename T >
T Arn::circVal (
    T x,
    T lo,
    T hi )
```

Definition at line 63 of file Math.hpp.

13.1.1.9 convertName()

```
QString Arn::convertName (
    const QString & name,
    Arn::NameF nameF = Arn::NameF() )
```

Convert a name to a specific format.

Name is a sub part from a *path*. Example: *name* = "value/", nameF = NoFolderMark ==> return = "value"

Parameters

in	<i>name</i>	
in	<i>nameF</i>	is the path naming format

Returns

The converted *name*

Definition at line 54 of file Arn.cpp.

13.1.1.10 convertPath()

```
QString Arn::convertPath (
    const QString & path,
    Arn::NameF nameF = Arn::NameF::EmptyOk )
```

Convert a path to a specific format.

Example: *path* = "//Measure/depth/value", nameF = Relative ==> return = "@/Measure/depth/value"

Parameters

in	<i>path</i>	
in	<i>nameF</i>	is the path naming format

Returns

The converted *path*

Definition at line 148 of file Arn.cpp.

13.1.1.11 fullPath()

```
QString Arn::fullPath (
    const QString & path )
```

Convert a path to a full absolute path.

Example: *path* = "Measure/depth/value" ==> return = "/Local/Measure/depth/value"

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The converted *path* full path

Definition at line 82 of file Arn.cpp.

13.1.1.12 hostFromHostWithInfo()

```
QString Arn::hostFromHostWithInfo (
    const QString & hostWithInfo )
```

Get the host from the *HostWithInfo* string.

This is typically used to extract only the host part without information, to be used in e.g. QTcpSocket for connection to the host.

Example: *hostWithInfo* = "192.168.1.1 [myhost.local]" ==> return = "192.168.1.1"

Parameters

in	<i>hostWithInfo</i>	The <i>HostWithInfo</i> string
----	---------------------	--------------------------------

Returns

The name or address of the host

See also

[makeHostWithInfo\(\)](#)

Note

As the format of the *HostWithInfo* string can be changed in the future, always use [makeHostWithInfo\(\)](#) and [hostFromHostWithInfo\(\)](#) for coding and decoding.

Definition at line 242 of file Arn.cpp.

13.1.1.13 isFolderPath()

```
bool Arn::isFolderPath (
    const QString & path )
```

Test if *path* is a *folder path*

Parameters

in	<i>path</i>	
----	-------------	--

Return values

<i>true</i>	if <i>path</i> is a <i>folder path</i> , i.e. ends with a "/".
-------------	--

Definition at line 210 of file Arn.cpp.

13.1.1.14 isNullPtr()

```
bool Arn::isNullPtr (
    const void * ptr )
```

Definition at line 253 of file Arn.cpp.

13.1.1.15 isPower2()

```
template<typename T >
bool Arn::isPower2 (
    T x )
```

Definition at line 69 of file Math.hpp.

13.1.1.16 isProviderPath()

```
bool Arn::isProviderPath (
    const QString & path )
```

Test if *path* is a *provider path*

[About Bidirectional Arn Data Objects](#)

Parameters

in	<i>path</i>	
----	-------------	--

Return values

<i>true</i>	if <i>path</i> is a <i>provider path</i> , i.e. ends with a "!".
-------------	--

Examples:

[ArnDemoChatServer/MainWindow.cpp](#).

Definition at line 216 of file Arn.cpp.

13.1.1.17 itemName()

```
QString Arn::itemName (
    const QString & path )
```

The last part of a *path*

Example: *path* = "//Measure/depth/value" ==> return = "value"

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The *itemName*, i.e. the last part of the path after last "/"

Definition at line 90 of file Arn.cpp.

13.1.1.18 log2()

```
template<typename T >
int Arn::log2 (
    T x ) [inline]
```

Definition at line 75 of file Math.hpp.

13.1.1.19 makeHostWithInfo()

```
QString Arn::makeHostWithInfo (
    const QString & host,
    const QString & info )
```

Make a combined host and info string, i.e. *HostWithInfo*

This is typically used to pass some extra information about the host, but still be used for connection to the host.

[ArnClient](#) and alike accepts such *HostWithInfo* strings for connection. Hosts discovered using e.g. [ArnDiscoverBrowser](#) will be using the ip-address as host and the host name as info.

Example: *host* = "192.168.1.1", *info* = "myhost.local" ==> return = "192.168.1.1 [myhost.local]"

Parameters

in	<i>host</i>	the name or address of the host
in	<i>info</i>	is corresponding info for the host

Returns

The *HostWithInfo* string

See also

[hostFromHostWithInfo\(\)](#)

Note

As the format of the *HostWithInfo* string can be changed in the future, always use [makeHostWithInfo\(\)](#) and [hostFromHostWithInfo\(\)](#) for coding and decoding.

Definition at line 235 of file Arn.cpp.

13.1.1.20 makePath()

```
QString Arn::makePath (
    const QString & parentPath,
    const QString & itemName )
```

Make a path from a parent and an item name.

parentPath don't have to end with a "/", if missing it's added. Empty folder *itemName* is allowed on returned path.

Example: *parentPath* = "//Measure/depth/", *itemName* = "value" ==> return = "//Measure/depth/value"

Parameters

in	<i>parentPath</i>	
in	<i>itemName</i>	

Returns

The *path*

Definition at line 128 of file Arn.cpp.

13.1.1.21 maxLim()

```
template<typename T >
T Arn::maxLim (
    const T & x,
    const T & lim )
```

Definition at line 87 of file Math.hpp.

13.1.1.22 minLim()

```
template<typename T >
T Arn::minLim (
    const T & x,
    const T & lim )
```

Definition at line 81 of file Math.hpp.

13.1.1.23 mod()

```
template<typename T >
T Arn::mod (
    T x,
    T y ) [inline]
```

Definition at line 56 of file Math.hpp.

13.1.1.24 parentPath()

```
QString Arn::parentPath (
    const QString & path )
```

Get the parent to a given *path*

Example: *path* = "//Measure/depth/value!" ==> return = "//Measure/depth/"

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The parent *path*

Definition at line 183 of file Arn.cpp.

13.1.1.25 providerPathIf()

```
QString Arn::providerPathIf (
    const QString & path,
    bool giveProviderPath = true )
```

Get *provider path* or *requester path*

[About Bidirectional Arn Data Objects](#)

Parameters

in	<i>path</i>	to be converted
in	<i>giveProviderPath</i>	choses between provider and requester path. false = requester path, default is true = provider path.

Return values

is	<i>provider path</i> or <i>requester path</i>
----	---

See also

[twinPath\(\)](#)
[isProviderPath\(\)](#)

Definition at line 204 of file Arn.cpp.

13.1.1.26 rand()

```
uint Arn::rand ( )
```

Definition at line 259 of file Arn.cpp.

13.1.1.27 rangeLim()

```
template<typename T >
T Arn::rangeLim (
    const T & x,
    const T & min,
    const T & max )
```

Definition at line 93 of file Math.hpp.

13.1.1.28 twinPath()

```
QString Arn::twinPath (
    const QString & path )
```

Get the bidirectional twin to a given *path*

Example: *path* = "//Measure/depth/value!" ==> return = "//Measure/depth/value"

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The twin *path*

See also

[Bidirectional Arn Data Objects](#)

Definition at line 195 of file Arn.cpp.

13.1.1.29 uuidPath()

```
QString Arn::uuidPath (
    const QString & path )
```

Get a path to an [Arn](#) Object with a unique uuid name.

Parameters

in	<i>path</i>	The prefix for Arn uuid path e.g. "//Names/name"
----	-------------	--

Returns

the unique path

Definition at line 222 of file Arn.cpp.

13.1.2 Variable Documentation**13.1.2.1 debugDepend**

```
bool Arn::debugDepend = false
```

Definition at line 46 of file ArnLib.cpp.

13.1.2.2 debugDiscover

```
bool Arn::debugDiscover = false
```

Definition at line 48 of file ArnLib.cpp.

13.1.2.3 debugLinkDestroy

```
bool Arn::debugLinkDestroy = false
```

Definition at line 40 of file ArnLib.cpp.

13.1.2.4 debugLinkRef

```
bool Arn::debugLinkRef = false
```

Definition at line 39 of file ArnLib.cpp.

13.1.2.5 debugMDNS

```
bool Arn::debugMDNS = false
```

Definition at line 50 of file ArnLib.cpp.

13.1.2.6 debugMonitor

```
bool Arn::debugMonitor = false
```

Definition at line 43 of file ArnLib.cpp.

13.1.2.7 debugMonitorTest

```
bool Arn::debugMonitorTest = false
```

Definition at line 44 of file ArnLib.cpp.

13.1.2.8 debugQmlNetwork

```
bool Arn::debugQmlNetwork = false
```

Definition at line 47 of file ArnLib.cpp.

13.1.2.9 debugReclnOut

```
bool Arn::debugRecInOut = false
```

Definition at line 41 of file ArnLib.cpp.

13.1.2.10 debugRPC

```
bool Arn::debugRPC = false
```

Definition at line 45 of file ArnLib.cpp.

13.1.2.11 debugShareObj

```
bool Arn::debugShareObj = false
```

Definition at line 42 of file ArnLib.cpp.

13.1.2.12 debugSizes

```
bool Arn::debugSizes = false
```

Definition at line 37 of file ArnLib.cpp.

13.1.2.13 debugThreading

```
bool Arn::debugThreading = false
```

Definition at line 38 of file ArnLib.cpp.

13.1.2.14 debugZeroConf

```
bool Arn::debugZeroConf = false
```

Definition at line 49 of file ArnLib.cpp.

13.1.2.15 defaultTcpPort

```
const quint16 Arn::defaultTcpPort = 2022
```

Definition at line 50 of file Arn.hpp.

13.1.2.16 offHeartbeat

```
bool Arn::offHeartbeat = false
```

Definition at line 52 of file ArnLib.cpp.

13.1.2.17 pathDiscover

```
const QString Arn::pathDiscover = "Sys/Discover/"
```

Definition at line 47 of file Arn.cpp.

13.1.2.18 pathDiscoverConnect

```
const QString Arn::pathDiscoverConnect = "Sys/Discover/Connect/"
```

Definition at line 49 of file Arn.cpp.

13.1.2.19 pathDiscoverThis

```
const QString Arn::pathDiscoverThis = "Sys/Discover/This/"
```

Definition at line 48 of file Arn.cpp.

13.1.2.20 pathLocal

```
const QString Arn::pathLocal = "/Local/"
```

Definition at line 45 of file Arn.cpp.

13.1.2.21 pathLocalSys

```
const QString Arn::pathLocalSys = "Sys/"
```

Definition at line 46 of file Arn.cpp.

13.1.2.22 pathServer

```
const QString Arn::pathServer = "Sys/Server/"
```

Definition at line 50 of file Arn.cpp.

13.1.2.23 pathServerSessions

```
const QString Arn::pathServerSessions = "Sys/Server/Sessions/"
```

Definition at line 51 of file Arn.cpp.

13.1.2.24 resourceArnLib

```
const QString Arn::resourceArnLib = ":/ArnLib/"
```

Definition at line 54 of file ArnLib.cpp.

13.1.2.25 resourceArnRoot

```
const QString Arn::resourceArnRoot = ":/ArnLib/ArnRoot/"
```

Definition at line 55 of file ArnLib.cpp.

13.1.2.26 warningMDNS

```
bool Arn::warningMDNS = false
```

Definition at line 51 of file ArnLib.cpp.

13.2 ArnDiscover Namespace Reference

Classes

- struct [Type](#)
Types of [Arn](#) discover advertise.

13.3 ArnZeroConf Namespace Reference

Classes

- struct [Error](#)
Errors of ZeroConfig, other values are defined in dns_sd.h.
- struct [State](#)
States of ZeroConfig, limited valid for each [ArnZeroConfB](#) subclass / These values must be synced with: [ArnDiscover::State](#).

Chapter 14

Class Documentation

14.1 Arn::_InitEnumTxt Struct Reference

```
#include <MQFlags.hpp>
```

Public Attributes

- int [ns](#)
- int [enumVal](#)
- const char * [enumTxt](#)

14.1.1 Detailed Description

Definition at line 131 of file MQFlags.hpp.

14.1.2 Member Data Documentation

14.1.2.1 enumTxt

```
const char* Arn::_InitEnumTxt::enumTxt
```

Definition at line 134 of file MQFlags.hpp.

14.1.2.2 enumVal

```
int Arn::_InitEnumTxt::enumVal
```

Definition at line 133 of file MQFlags.hpp.

14.1.2.3 ns

```
int Arn::_InitEnumTxt::ns
```

Definition at line 132 of file MQFlags.hpp.

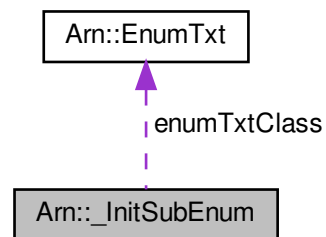
The documentation for this struct was generated from the following file:

- [src/ArnInc/MQFlags.hpp \(4.0.0\)](#)

14.2 Arn::_InitSubEnum Struct Reference

```
#include <MQFlags.hpp>
```

Collaboration diagram for Arn::_InitSubEnum:



Public Attributes

- [EnumTxt](#) * [enumTxtClass](#)
- uint [mask](#)
- uint [factor](#)

14.2.1 Detailed Description

Definition at line 137 of file MQFlags.hpp.

14.2.2 Member Data Documentation

14.2.2.1 enumTxtClass

```
EnumTxt* Arn::_InitSubEnum::enumTxtClass
```

Definition at line 138 of file MQFlags.hpp.

14.2.2.2 factor

```
uint Arn::_InitSubEnum::factor
```

Definition at line 140 of file MQFlags.hpp.

14.2.2.3 mask

```
uint Arn::_InitSubEnum::mask
```

Definition at line 139 of file MQFlags.hpp.

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

- [src/ArnInc/MQFlags.hpp \(4.0.0\)](#)

14.3 Arn::Allow Class Reference

```
#include <Arn.hpp>
```

Public Types

- enum `E` {
 `None` = 0x00, `Read` = 0x01, `Write` = 0x02, `Create` = 0x04,
 `Delete` = 0x08, `ModeChange` = 0x10, `ReadWrite` = 0x03, `All` = 0xff }

14.3.1 Detailed Description

Definition at line 206 of file Arn.hpp.

14.3.2 Member Enumeration Documentation

14.3.2.1 E

```
enum Arn::Allow::E
```

Enumerator

None	Nothing allowed.
Read	Read from Arn Objects.
Write	Write to Arn Objects.
Create	Create Arn Objects.
Delete	Delete Arn Objects.
ModeChange	Change Mode of Arn Objects.
ReadWrite	Convenience, allow read & write.
All	Convenience, allow all.

Definition at line 210 of file Arn.hpp.

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

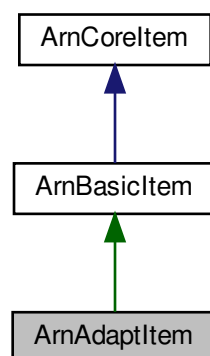
- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.4 ArnAdaptItem Class Reference

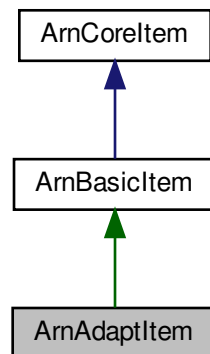
! Non Qt and threadsafe handle for an [Arn Data Object](#).

```
#include <ArnAdaptItem.hpp>
```

Inheritance diagram for ArnAdaptItem:



Collaboration diagram for ArnAdaptItem:



Public Types

- typedef void(* [ChangedCB](#)) ([ArnAdaptItem](#) &target, const QByteArray &value)
- typedef void(* [LinkDestroyedCB](#)) ([ArnAdaptItem](#) &target)
- typedef void(* [ArnEventCB](#)) (QEvent *ev, int arnEvIdx)

Public Member Functions

- [ArnAdaptItem](#) ()
Standard constructor of a closed handle.
- virtual [~ArnAdaptItem](#) ()
- bool [open](#) (const QString &path)
Open a handle to an [Arn Data Object](#)
- void [close](#) ()
Close the handle.
- void [destroyLink](#) (bool isGlobal=true)
Destroy the [Arn Data Object](#)
- void [destroyLinkLocal](#) ()
Destroy the local [Arn Data Object](#)
- bool [isOpen](#) () const
State of the handle.
- QString [path](#) ([Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#)) const
Path of the [Arn Data Object](#)
- QString [name](#) ([Arn::NameF](#) nameF) const
Name of the [Arn Data Object](#)
- void [setReference](#) (void *reference)
Set an associated external reference.
- void * [reference](#) () const
Get the stored external reference.
- uint [itemId](#) () const

- Get the id for this [ArnItem](#).
- uint [linkId](#) () const
- Get the id for this [Arn Data Object](#)
- int [refCount](#) () const
- Get the number of refs to this [Arn Data Object](#)
- bool [isFolder](#) () const
- bool [isProvider](#) () const
- [Arn::DataType](#) [type](#) () const
- The type stored in the [Arn Data Object](#)
- void [setIgnoreSameValue](#) (bool [isIgnore](#)=true)
- Set skipping of equal value.
- bool [isIgnoreSameValue](#) () const
- void [addMode](#) ([Arn::ObjectMode](#) [mode](#))
- Add general mode settings for this [Arn Data Object](#)
- [Arn::ObjectMode](#) [getMode](#) () const
- [Arn::ObjectSyncMode](#) [syncMode](#) () const
- [ArnAdaptItem](#) & [setBiDirMode](#) ()
- Set general mode as Bidirectional for this [Arn Data Object](#)
- bool [isBiDirMode](#) () const
- [ArnAdaptItem](#) & [setPipeMode](#) ()
- Set general mode as Pipe for this [Arn Data Object](#)
- bool [isPipeMode](#) () const
- [ArnAdaptItem](#) & [setSaveMode](#) ()
- Set general mode as Save for this [Arn Data Object](#)
- bool [isSaveMode](#) () const
- [ArnAdaptItem](#) & [setMaster](#) ()
- Set client session sync mode as Master for this [ArnItem](#).
- bool [isMaster](#) () const
- [ArnAdaptItem](#) & [setAutoDestroy](#) ()
- Set client session sync mode as AutoDestroy for this [ArnItem](#).
- bool [isAutoDestroy](#) () const
- void [arnImport](#) (const QByteArray &[data](#), int [ignoreSame](#)=[Arn::SameValue::DefaultAction](#))
- Import data to an [Arn Data Object](#)
- QByteArray [arnExport](#) () const
- int [toInt](#) (bool *[isOk](#)=arnNullptr) const
- double [toDouble](#) (bool *[isOk](#)=arnNullptr) const
- [ARNREAL](#) [toReal](#) (bool *[isOk](#)=arnNullptr) const
- QString [toString](#) (bool *[isOk](#)=arnNullptr) const
- QByteArray [toByteArray](#) (bool *[isOk](#)=arnNullptr) const
- QVariant [toVariant](#) (bool *[isOk](#)=arnNullptr) const
- bool [toBool](#) (bool *[isOk](#)=arnNullptr) const
- uint [toUInt](#) (bool *[isOk](#)=arnNullptr) const
- quint64 [toInt64](#) (bool *[isOk](#)=arnNullptr) const
- quint64 [toUInt64](#) (bool *[isOk](#)=arnNullptr) const
- [ArnAdaptItem](#) & [operator=](#) (const [ArnAdaptItem](#) &[other](#))
- [ArnAdaptItem](#) & [operator=](#) (int [val](#))
- [ArnAdaptItem](#) & [operator=](#) ([ARNREAL](#) [val](#))
- [ArnAdaptItem](#) & [operator=](#) (const QString &[val](#))
- [ArnAdaptItem](#) & [operator=](#) (const QByteArray &[val](#))
- [ArnAdaptItem](#) & [operator=](#) (const QVariant &[val](#))
- [ArnAdaptItem](#) & [operator=](#) (const char *[val](#))
- [ArnAdaptItem](#) & [operator=](#) (uint [val](#))
- [ArnAdaptItem](#) & [operator=](#) (quint64 [val](#))

- [ArnAdaptItem](#) & `operator=` (quint64 val)
- [ArnAdaptItem](#) & `operator+=` (int val)
- [ArnAdaptItem](#) & `operator+=` (ARNREAL val)
- void `setValue` (const [ArnAdaptItem](#) &other, int ignoreSame=[Arn::SameValue::DefaultAction](#))
- void `setValue` (int value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an integer to an [Arn](#) Data Object
- void `setValue` (ARNREAL value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an ARNREAL to an [Arn](#) Data Object
- void `setValue` (bool value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign a bool to an [Arn](#) Data Object
- void `setValue` (const QString &value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign a QString to an [Arn](#) Data Object
- void `setValue` (const QByteArray &value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign a QByteArray to an [Arn](#) Data Object
- void `setValue` (const QVariant &value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign a QVariant to an [Arn](#) Data Object
- void `setValue` (const char *value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign a char to an [Arn](#) Data Object*
- void `setValue` (uint value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an unsigned int to an [Arn](#) Data Object
- void `setValue` (quint64 value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an int 64 bit to an [Arn](#) Data Object
- void `setValue` (quint64 value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an unsigned int 64 bit to an [Arn](#) Data Object
- void `setBits` (int mask, int value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
AtomicOp assign an integer to specified bits in an [Arn](#) Data Object
- void `addValue` (int value)
AtomicOp adds an integer to an [Arn](#) Data Object
- void `addValue` (ARNREAL value)
AtomicOp adds an ARNREAL to an [Arn](#) Data Object
- [ARN_RecursiveMutex](#) & `mutex` () const
Get the mutex of this [ArnAdaptItem](#).
- QThread * `thread` () const
Get the thread affinity of this [ArnAdaptItem](#).
- void `setChangedCallback` ([ChangedCB](#) changedCB)
Set changed-callback for this [ArnAdaptItem](#).
- [ChangedCB](#) `ChangedCallback` () const
Get the changed-callback of this [ArnAdaptItem](#).
- void `setLinkDestroyedCallback` ([LinkDestroyedCB](#) linkDestroyedCB)
Set link-destroyed-callback for this [ArnAdaptItem](#).
- [LinkDestroyedCB](#) `linkDestroyedCallback` () const
Get the link-destroyed-callback of this [ArnAdaptItem](#).
- void `setArnEventCallback` ([ArnEventCB](#) evCB)
Set event callback for this [ArnAdaptItem](#).
- [ArnEventCB](#) `arnEventCallback` () const
Get the event callback of this [ArnAdaptItem](#).
- void `setUncrossed` (bool isUncrossed=true)
Set a Bidirectional item as Uncrossed.
- bool `isUncrossed` () const
Get the Uncrossed state of an object.

Additional Inherited Members

14.4.1 Detailed Description

! Non Qt and threadsafe handle for an *Arn Data Object*.

[About ArnItem access](#)

See [ArnItem](#).

[ArnAdaptItem](#) is based on [ArnBasicItem](#) and is used to get a handle (pointer) for accessing an *Arn Data Object*. It is very similar to [ArnBasicItem](#) but it is slower and its typical usage is in a non Qt thread. It don't use or need a Qt eventloop.

There can be any amount of [ArnAdaptItem](#):s opened (pointing) to the same *Arn Data object*. Deleting the [ArnAdaptItem](#) won't effect the *Arn Data object*.

This class is thread-safe, so any thread could use its instances. This includes booth Qt (based on QThread) and non Qt started thread.

For callbacks it's easiest to use [setChangedCallback\(\)](#) and [setLinkDestroyedCallback\(\)](#) when this is sufficient. For advanced usage it's also possible to use [setArnEventCallback\(\)](#) which gives all possible events but is more complicated and includes decoding of an event structure.

Example usage

```
// In class declare
ArnAdaptItem _arnTime;
static void arnEvCallback( QEvent* ev, int arnEvIdx);

// In class code
_arnTime.open("//Chat/Time/value");
_arnTime.setChangedCallback( &MyClass::changedCallback);
_arnTime.setLinkDestroyedCallback( &MyClass::linkDestroyedCallback);
_arnTime.setArnEventCallback( &MyClass::arnEvCallback);
_arnTime = "Undefined ...";

void MyClass::changedCallback( ArnAdaptItem& item, const QByteArray& value)
{
    // Is setup as Changed callback for my ArnAdaptItem.
    // Code must be threadsafe.

    qDebug() << "MyClass ValueChange: inItemPath=" << item.path()
               << " value=" << value;
}

void MyClass::linkDestroyedCallback( ArnAdaptItem& item)
{
    // Is setup as link-destroyed callback for my ArnAdaptItem.
    // Code must be threadsafe.

    qDebug() << "MyClass LinkDestroyed: inItemPath=" << item.path()
}

void MyClass::arnEvCallback( QEvent* ev, int arnEvIdx)
{
    // Is setup as ArnEvent callback for my ArnAdaptItem.
    // Code must be threadsafe.

    switch (arnEvIdx) {
    case ArnEvent::Idx::ValueChange:
    {
        ArnEvValueChange* e = static_cast<ArnEvValueChange*>( ev);
        ArnAdaptItem* item = static_cast<ArnAdaptItem*>( e->
target());
        if (!item) break; // No target, deleted/closed ...

        QByteArray val = e->valueData() ? *e->valueData() : item->
toByteArray();
        qDebug() << "MyClass EvValueChange: inItemPath=" << item->path()
                 << " value=" << val;
    }
    }
}
```

```

        break;
    }
    case ArnEvent::Idx::ModeChange:
    {
        ArnEvModeChange* e = static_cast<ArnEvModeChange*>( ev);
        ArnAdaptItem* item = static_cast<ArnAdaptItem*>( e->
target());
        if (!item) return; // No target, deleted/closed ...

        QMutexLocker mutexLocker( &item->mutex()); // Force atomic operation on target

        qDebug() << "EvModeChange: path=" << e->path() << " mode=" << e->
mode()
        << " inItemPath=" << item->path();
        break;
    }
    default:
        break;
    }
}
}

```

Definition at line 133 of file ArnAdaptItem.hpp.

14.4.2 Member Typedef Documentation

14.4.2.1 ArnEventCB

```
typedef void(* ArnAdaptItem::ArnEventCB) (QEvent *ev, int arnEvIdx)
```

Definition at line 140 of file ArnAdaptItem.hpp.

14.4.2.2 ChangedCB

```
typedef void(* ArnAdaptItem::ChangedCB) (ArnAdaptItem &target, const QByteArray &value)
```

Definition at line 138 of file ArnAdaptItem.hpp.

14.4.2.3 LinkDestroyedCB

```
typedef void(* ArnAdaptItem::LinkDestroyedCB) (ArnAdaptItem &target)
```

Definition at line 139 of file ArnAdaptItem.hpp.

14.4.3 Constructor & Destructor Documentation

14.4.3.1 ArnAdaptItem()

```
ArnAdaptItem::ArnAdaptItem ( )
```

Standard constructor of a closed handle.

Definition at line 70 of file ArnAdaptItem.cpp.

14.4.3.2 ~ArnAdaptItem()

```
ArnAdaptItem::~~ArnAdaptItem ( ) [virtual]
```

Definition at line 77 of file ArnAdaptItem.cpp.

14.4.4 Member Function Documentation

14.4.4.1 addMode()

```
void ArnAdaptItem::addMode (
    Arn::ObjectMode mode )
```

Add *general mode* settings for this *Arn Data Object*

If this *ArnItem* is in closed state, the added modes will be stored and the real mode change is done when this *ArnItem* is opened to an *Arn Data Object*. This implies that *ArnItems* can benefit from setting *modes* before opening.

Parameters

in	<i>mode</i>	The <i>modes</i> to be added.
----	-------------	-------------------------------

See also

[getMode\(\)](#)
[Modes](#)

Definition at line 221 of file ArnAdaptItem.cpp.

14.4.4.2 addValue() [1/2]

```
void ArnAdaptItem::addValue (
    int value )
```

AtomicOp adds an *integer* to an *Arn Data Object*

Operation is done atomicly. If *bidir*, it can also be done remotely by an *AtomicOpProvider*

Parameters

in	<i>value</i>	to be added to this Arn Data Object
----	--------------	---

See also

[setAtomicOpProvider\(\)](#)

Definition at line 644 of file ArnAdaptItem.cpp.

14.4.4.3 addValue() [2/2]

```
void ArnAdaptItem::addValue (
    ARNREAL value )
```

AtomicOp adds an *ARNREAL* to an [Arn Data Object](#)

Operation is done atomically. If *bidir*, it can also be done remotely by an AtomicOpProvider

Parameters

in	<i>value</i>	to be added to this Arn Data Object
----	--------------	---

See also

[setAtomicOpProvider\(\)](#)

Definition at line 652 of file ArnAdaptItem.cpp.

14.4.4.4 arnEventCallback()

```
ArnAdaptItem::ArnEventCB ArnAdaptItem::arnEventCallback ( ) const
```

Get the event callback of this [ArnAdaptItem](#).

Returns

the event callback

See also

[setArnEventCallback\(\)](#)
[thread\(\)](#)

Definition at line 714 of file ArnAdaptItem.cpp.

14.4.4.5 arnExport()

```
QByteArray ArnAdaptItem::arnExport ( ) const
```

Returns

A data blob representing the *Arn Data Object*

See also

[arnImport\(\)](#)

Definition at line 345 of file ArnAdaptItem.cpp.

14.4.4.6 arnImport()

```
void ArnAdaptItem::arnImport (
    const QByteArray & data,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Import data to an *Arn Data Object*

Data blob from a previous [arnExport\(\)](#) can be imported. This is essentially assigning the *Arn Data Object* with same as exported.

Parameters

in	<i>data</i>	is the data blob
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[arnExport\(\)](#)
[setIgnoreSameValue\(\)](#)

Definition at line 337 of file ArnAdaptItem.cpp.

14.4.4.7 ChangedCallback()

```
ArnAdaptItem::ChangedCB ArnAdaptItem::ChangedCallback ( ) const
```

Get the changed-callback of this *ArnAdaptItem*.

Returns

the changed-callback

See also

[setChangedCallback\(\)](#)
[thread\(\)](#)

Definition at line 682 of file ArnAdaptItem.cpp.

14.4.4.8 close()

```
void ArnAdaptItem::close ( )
```

Close the handle.

Definition at line 92 of file ArnAdaptItem.cpp.

14.4.4.9 destroyLink()

```
void ArnAdaptItem::destroyLink (
    bool isGlobal = true )
```

Destroy the *Arn Data Object*

The link (*Arn Data Object*) will be removed locally and optionally from server and all connected clients. Server is allways forcing global destroy.

Parameters

in	<i>isGlobal</i>	If true, removes from server and all connected clients, otherwise only local link.
----	-----------------	--

See also

[destroyLinkLocal\(\)](#)

Definition at line 100 of file ArnAdaptItem.cpp.

14.4.4.10 destroyLinkLocal()

```
void ArnAdaptItem::destroyLinkLocal ( ) [inline]
```

Destroy the local *Arn Data Object*

The link (*Arn Data Object*) will be removed locally. Server is allways forcing global destroy.

See also

[destroyLink\(\)](#)

Definition at line 172 of file ArnAdaptItem.hpp.

14.4.4.11 `getMode()`

```
Arn::ObjectMode ArnAdaptItem::getMode ( ) const
```

Returns

The *general mode* of the *Arn Data Object*

See also

[addMode\(\)](#)
[Modes](#)

Definition at line 229 of file ArnAdaptItem.cpp.

14.4.4.12 `isAutoDestroy()`

```
bool ArnAdaptItem::isAutoDestroy ( ) const
```

Return values

<i>true</i>	if <i>AutoDestroy mode</i>
-------------	----------------------------

See also

[setAutoDestroy\(\)](#)

Definition at line 328 of file ArnAdaptItem.cpp.

14.4.4.13 `isBiDirMode()`

```
bool ArnAdaptItem::isBiDirMode ( ) const
```

Return values

<i>true</i>	if Bidirectional
-------------	------------------

See also

[setBiDirMode\(\)](#)
[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 256 of file ArnAdaptItem.cpp.

14.4.4.14 isFolder()

```
bool ArnAdaptItem::isFolder ( ) const
```

Return values

<i>true</i>	if this ArnItem is a folder
-------------	---

Definition at line 177 of file ArnAdaptItem.cpp.

14.4.4.15 isIgnoreSameValue()

```
bool ArnAdaptItem::isIgnoreSameValue ( ) const
```

Return values

<i>true</i>	if skipping equal values
-------------	--------------------------

See also

[setIgnoreSameValue\(\)](#)

Definition at line 212 of file ArnAdaptItem.cpp.

14.4.4.16 isMaster()

```
bool ArnAdaptItem::isMaster ( ) const
```

Return values

<i>true</i>	if <i>Master mode</i>
-------------	-----------------------

See also

[setMaster\(\)](#)
[Modes](#)

Definition at line 310 of file ArnAdaptItem.cpp.

14.4.4.17 isOpen()

```
bool ArnAdaptItem::isOpen ( ) const
```

State of the handle.

Return values

<i>true</i>	if this ArnItem is open
-------------	---

Definition at line 108 of file ArnAdaptItem.cpp.

14.4.4.18 isPipeMode()

```
bool ArnAdaptItem::isPipeMode ( ) const
```

Return values

<i>true</i>	if <i>Pipe mode</i>
-------------	---------------------

See also

[setPipeMode\(\)](#)
[Modes](#)
[Pipe Arn Data Objects](#)

Definition at line 274 of file ArnAdaptItem.cpp.

14.4.4.19 isProvider()

```
bool ArnAdaptItem::isProvider ( ) const
```

Return values

<i>true</i>	if this ArnItem is a provider
-------------	---

See also

[setBiDirMode\(\)](#)
[Modes](#)

Definition at line 186 of file ArnAdaptItem.cpp.

14.4.4.20 isSaveMode()

```
bool ArnAdaptItem::isSaveMode ( ) const
```

Return values

<i>true</i>	if <i>Save mode</i>
-------------	---------------------

See also

[setSaveMode\(\)](#)
[Modes](#)
[Persistent Arn Data Objects](#)

Definition at line 292 of file ArnAdaptItem.cpp.

14.4.4.21 isUncrossed()

```
bool ArnAdaptItem::isUncrossed ( ) const
```

Get the Uncrossed state of an object.

Return values

<i>true</i>	if Uncrossed is set or <i>Arn Data Object</i> is not in Bidirectional mode.
-------------	---

See also

[setUncrossed\(\)](#)
[setBiDirMode\(\)](#)
[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 730 of file ArnAdaptItem.cpp.

14.4.4.22 itemId()

```
uint ArnAdaptItem::itemId ( ) const
```

Get the *id* for this [ArnItem](#).

The [ArnItem](#) *id* is unique within its running program. Even if 2 [ArnItems](#) are pointing to the same [Arn Data Object](#), they have different *item id*.

Returns

id for this [ArnItem](#)

See also

[linkId\(\)](#)

Definition at line 150 of file ArnAdaptItem.cpp.

14.4.4.23 linkDestroyedCallback()

```
ArnAdaptItem::LinkDestroyedCB ArnAdaptItem::linkDestroyedCallback ( ) const
```

Get the link-destroyed-callback of this [ArnAdaptItem](#).

Returns

the link-destroyed-callback

See also

[setLinkDestroyedCallback\(\)](#)
[thread\(\)](#)

Definition at line 698 of file ArnAdaptItem.cpp.

14.4.4.24 linkId()

```
uint ArnAdaptItem::linkId ( ) const
```

Get the *id* for this *Arn Data Object*

The link (*Arn Data Object*) *id* is unique within its running program. If 2 ArnItems are pointing to the same *Arn Data Object*, they have same *link id*.

Returns

Id for the *Arn Data Object*, 0 if closed

See also

[itemId\(\)](#)

Definition at line 159 of file ArnAdaptItem.cpp.

14.4.4.25 mutex()

```
ARN_RecursiveMutex & ArnAdaptItem::mutex ( ) const
```

Get the mutex of this [ArnAdaptItem](#).

This can be used for atomic operations etc on the item. The item it self is thread safe without the application code is using this mutex. Also this mutex is using QMutex::Recursive.

Returns

the items mutex

Definition at line 660 of file ArnAdaptItem.cpp.

14.4.4.26 name()

```
QString ArnAdaptItem::name (
    Arn::NameF nameF ) const
```

Name of the *Arn Data Object*

Parameters

in	<i>nameF</i>	The format of the returned name
----	--------------	---------------------------------

Returns

The object name

Definition at line 125 of file ArnAdaptItem.cpp.

14.4.4.27 open()

```
bool ArnAdaptItem::open (
    const QString & path )
```

Open a handle to an [Arn Data Object](#)

Parameters

in	<i>path</i>	The Arn Data Object path e.g. "//Measure/Water/Level/value"
----	-------------	---

Return values

<i>false</i>	if error
--------------	----------

Definition at line 82 of file ArnAdaptItem.cpp.

14.4.4.28 operator+=() [1/2]

```
ArnAdaptItem & ArnAdaptItem::operator+= (
    int val )
```

Definition at line 530 of file ArnAdaptItem.cpp.

14.4.4.29 operator+=() [2/2]

```
ArnAdaptItem & ArnAdaptItem::operator+= (
    ARNREAL val )
```

Definition at line 539 of file ArnAdaptItem.cpp.

14.4.4.30 operator=() [1/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    const ArnAdaptItem & other )
```

Definition at line 440 of file ArnAdaptItem.cpp.

14.4.4.31 operator=() [2/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    int val )
```

Definition at line 449 of file ArnAdaptItem.cpp.

14.4.4.32 operator=() [3/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    ARNREAL val )
```

Definition at line 458 of file ArnAdaptItem.cpp.

14.4.4.33 operator=() [4/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    const QString & val )
```

Definition at line 467 of file ArnAdaptItem.cpp.

14.4.4.34 operator=() [5/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    const QByteArray & val )
```

Definition at line 476 of file ArnAdaptItem.cpp.

14.4.4.35 operator=() [6/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    const QVariant & val )
```

Definition at line 485 of file ArnAdaptItem.cpp.

14.4.4.36 operator=() [7/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    const char * val )
```

Definition at line 494 of file ArnAdaptItem.cpp.

14.4.4.37 operator=() [8/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    uint val )
```

Definition at line 503 of file ArnAdaptItem.cpp.

14.4.4.38 operator=() [9/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    qint64 val )
```

Definition at line 512 of file ArnAdaptItem.cpp.

14.4.4.39 operator=() [10/10]

```
ArnAdaptItem & ArnAdaptItem::operator= (
    quint64 val )
```

Definition at line 521 of file ArnAdaptItem.cpp.

14.4.4.40 path()

```
QString ArnAdaptItem::path (
    Arn::NameF nameF = Arn::NameF::EmptyOk ) const
```

Path of the *Arn Data Object*

Parameters

in	<i>nameF</i>	The format of the returned path
----	--------------	---------------------------------

Returns

The object path

Definition at line 117 of file ArnAdaptItem.cpp.

14.4.4.41 refCount()

```
int ArnAdaptItem::refCount ( ) const
```

Get the number of refs to this *Arn Data Object*

Returns

The number of refs for the *Arn Data Object*, -1 if closed

Definition at line 168 of file ArnAdaptItem.cpp.

14.4.4.42 reference()

```
void * ArnAdaptItem::reference ( ) const
```

Get the stored external reference.

Returns

The associated external reference

See also

[setReference\(\)](#)

Definition at line 141 of file ArnAdaptItem.cpp.

14.4.4.43 setArnEventCallback()

```
void ArnAdaptItem::setArnEventCallback (
    ArnEventCB evCB )
```

Set event callback for this *ArnAdaptItem*.

Use e.g prototype: void myArnEventCB(QEvent* ev, int arnEvIdx); The event callback function must be threadsafe as it can be called from any thread.

Parameters

in	<i>evCB</i>	callback to be assigned
----	-------------	-------------------------

See also

[arnEventCallback\(\)](#)
[thread\(\)](#)

Definition at line 706 of file ArnAdaptItem.cpp.

14.4.4.44 setAutoDestroy()

```
ArnAdaptItem & ArnAdaptItem::setAutoDestroy ( )
```

Set client session *sync mode* as *AutoDestroy* for this [ArnItem](#).

This [ArnItem](#) at client side is setup for auto destruction.

Precondition

This must be set before [open\(\)](#).

Definition at line 319 of file ArnAdaptItem.cpp.

14.4.4.45 setBiDirMode()

```
ArnAdaptItem & ArnAdaptItem::setBiDirMode ( )
```

Set *general mode* as *Bidirectional* for this [Arn Data Object](#)

A two way object, typically for validation or pipe

See also

[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 247 of file ArnAdaptItem.cpp.

14.4.4.46 setBits()

```
void ArnAdaptItem::setBits (
    int mask,
    int value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

AtomicOp assign an *integer* to specified bits in an [Arn Data Object](#)

Operation is done atomicly. If *bidir*, it can also be done remotely by an AtomicOpProvider

Parameters

in	<i>mask</i>	to specify bits that is affected
in	<i>value</i>	to be assigned to affected bits
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setAtomicOpProvider\(\)](#)
[setIgnoreSameValue\(\)](#)

Definition at line 636 of file ArnAdaptItem.cpp.

14.4.4.47 setChangedCallback()

```
void ArnAdaptItem::setChangedCallback (
    ArnAdaptItem::ChangedCB changedCB )
```

Set changed-callback for this [ArnAdaptItem](#).

The callback is called when data in *Arn Data Object* is changed. Use e.g prototype: void myChangeCB(ArnAdaptItem& target, const QByteArray& value); The changed-callback function must be threadsafe as it can be called from any thread.

Parameters

in	<i>changedCB</i>	callback to be assigned
----	------------------	-------------------------

See also

[changedCallback\(\)](#)
[thread\(\)](#)

Definition at line 674 of file ArnAdaptItem.cpp.

14.4.4.48 setIgnoreSameValue()

```
void ArnAdaptItem::setIgnoreSameValue (
    bool isIgnore = true )
```

Set skipping of equal value.

Parameters

in	<i>isIgnore</i>	If true, assignment of equal value don't give a changed signal.
----	-----------------	---

Definition at line 204 of file ArnAdaptItem.cpp.

14.4.4.49 setLinkDestroyedCallback()

```
void ArnAdaptItem::setLinkDestroyedCallback (
    ArnAdaptItem::LinkDestroyedCB linkDestroyedCB )
```

Set link-destroyed-callback for this [ArnAdaptItem](#).

The callback is called when the *Arn Data Object* is destroyed. Use e.g prototype: void myLinkDestroyedCB(ArnAdaptItem& target); The link-destroyed-callback function must be threadsafe as it can be called from any thread.

Parameters

in	<i>linkDestroyedCB</i>	callback to be assigned
----	------------------------	-------------------------

See also

[linkDestroyedCallback\(\)](#)
[thread\(\)](#)

Definition at line 690 of file ArnAdaptItem.cpp.

14.4.4.50 setMaster()

```
ArnAdaptItem & ArnAdaptItem::setMaster ( )
```

Set client session *sync mode* as *Master* for this [ArnItem](#).

This [ArnItem](#) at client side is set as default generator of data.

Precondition

This must be set before [open\(\)](#).

See also

[Modes](#)

Definition at line 301 of file ArnAdaptItem.cpp.

14.4.4.51 setPipeMode()

[ArnAdaptItem](#) & [ArnAdaptItem::setPipeMode](#) ()

Set *general mode* as Pipe for this [Arn Data Object](#)

Implies *Bidir*.

See also

[Modes](#)
[Pipe Arn Data Objects](#)

Definition at line 265 of file [ArnAdaptItem.cpp](#).

14.4.4.52 setReference()

```
void ArnAdaptItem::setReference (
    void * reference )
```

Set an associated external reference.

This is typically used when having many *ArnItems* changed signal connected to a common slot. The slot can then discover the signalling [ArnItem](#):s associated structure for further processing.

Parameters

in	<i>reference</i>	Any external structure or id.
----	------------------	-------------------------------

See also

[reference\(\)](#)

Definition at line 133 of file [ArnAdaptItem.cpp](#).

14.4.4.53 setSaveMode()

[ArnAdaptItem](#) & [ArnAdaptItem::setSaveMode](#) ()

Set *general mode* as Save for this [Arn Data Object](#)

Data is persistent and will be saved

Precondition

The persistent service must be started at the server.

See also

[Modes](#)
[Persistent Arn Data Objects](#)

Definition at line 283 of file ArnAdaptItem.cpp.

14.4.4.54 setUncrossed()

```
void ArnAdaptItem::setUncrossed (
    bool isUncrossed = true )
```

Set a Bidirectional item as Uncrossed.

The two way object is not twisted at writes, i.e. exactly the same object is read and written. This has no effect on an *Arn Data Object* that not is in Bidirectional mode.

See also

[isUncrossed\(\)](#)
[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 722 of file ArnAdaptItem.cpp.

14.4.4.55 setValue() [1/11]

```
void ArnAdaptItem::setValue (
    const ArnAdaptItem & other,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Definition at line 548 of file ArnAdaptItem.cpp.

14.4.4.56 setValue() [2/11]

```
void ArnAdaptItem::setValue (
    int value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *integer* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 556 of file ArnAdaptItem.cpp.

14.4.4.57 setValue() [3/11]

```
void ArnAdaptItem::setValue (
    ARNREAL value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *ARNREAL* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 564 of file ArnAdaptItem.cpp.

14.4.4.58 setValue() [4/11]

```
void ArnAdaptItem::setValue (
    bool value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *bool* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 572 of file ArnAdaptItem.cpp.

14.4.4.59 setValue() [5/11]

```
void ArnAdaptItem::setValue (
    const QString & value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *QString* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 580 of file ArnAdaptItem.cpp.

14.4.4.60 setValue() [6/11]

```
void ArnAdaptItem::setValue (
    const QByteArray & value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *QByteArray* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 588 of file ArnAdaptItem.cpp.

14.4.4.61 setValue() [7/11]

```
void ArnAdaptItem::setValue (
    const QVariant & value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *QVariant* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 596 of file ArnAdaptItem.cpp.

14.4.4.62 setValue() [8/11]

```
void ArnAdaptItem::setValue (
    const char * value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *char** to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 604 of file ArnAdaptItem.cpp.

14.4.4.63 setValue() [9/11]

```
void ArnAdaptItem::setValue (
    uint value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *unsigned int* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 612 of file ArnAdaptItem.cpp.

14.4.4.64 setValue() [10/11]

```
void ArnAdaptItem::setValue (
    qint64 value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *int 64 bit* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 620 of file ArnAdaptItem.cpp.

14.4.4.65 setValue() [11/11]

```
void ArnAdaptItem::setValue (
    quint64 value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *unsigned int 64 bit* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 628 of file ArnAdaptItem.cpp.

14.4.4.66 syncMode()

```
Arn::ObjectSyncMode ArnAdaptItem::syncMode ( ) const
```

Returns

The client session *sync mode* of an *Arn Data Object*

See also

[Modes](#)

Definition at line 238 of file ArnAdaptItem.cpp.

14.4.4.67 thread()

```
QThread * ArnAdaptItem::thread ( ) const
```

Get the thread affinity of this [ArnAdaptItem](#).

The affinity is always the same as the caller thread.

Returns

the thread affinity (caller thread)

See also

[setArnEventCallback\(\)](#)

Definition at line 668 of file ArnAdaptItem.cpp.

14.4.4.68 toBool()

```
bool ArnAdaptItem::toBool (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *bool*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from Int.

Definition at line 404 of file ArnAdaptItem.cpp.

14.4.4.69 toByteArray()

```
QByteArray ArnAdaptItem::toByteArray (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *QByteArray*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 388 of file ArnAdaptItem.cpp.

14.4.4.70 toDouble()

```
double ArnAdaptItem::toDouble (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *double*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 362 of file ArnAdaptItem.cpp.

14.4.4.71 toInt()

```
int ArnAdaptItem::toInt (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *integer*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 353 of file ArnAdaptItem.cpp.

14.4.4.72 toInt64()

```
qint64 ArnAdaptItem::toInt64 (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *int 64 bit*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 422 of file ArnAdaptItem.cpp.

14.4.4.73 toReal()

```
ARNREAL ArnAdaptItem::toReal (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *ARNREAL*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 371 of file ArnAdaptItem.cpp.

14.4.4.74 toString()

```
QString ArnAdaptItem::toString (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *QString*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 380 of file ArnAdaptItem.cpp.

14.4.4.75 toUInt()

```
uint ArnAdaptItem::toUInt (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *unsigned int*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 413 of file ArnAdaptItem.cpp.

14.4.4.76 toUInt64()

```
quint64 ArnAdaptItem::toUInt64 (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *unsigned int 64 bit*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 431 of file ArnAdaptItem.cpp.

14.4.4.77 toVariant()

```
QVariant ArnAdaptItem::toVariant (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *QVariant*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 396 of file ArnAdaptItem.cpp.

14.4.4.78 type()

```
Arn::DataType ArnAdaptItem::type ( ) const
```

The type stored in the *Arn Data Object*

Returns

The type stored

Definition at line 195 of file ArnAdaptItem.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnAdaptItem.hpp \(4.0.0\)](#)
- [src/ArnAdaptItem.cpp \(4.0.0\)](#)

14.5 ArnAtomicOp Class Reference

```
#include <ArnEvent.hpp>
```

Public Types

- enum `E` {
`None = 0`, `BitSet`, `AddInt`, `AddReal`,
`N` }
- enum `NS` { `NsEnum`, `NsCom` }

14.5.1 Detailed Description

Definition at line 67 of file ArnEvent.hpp.

14.5.2 Member Enumeration Documentation

14.5.2.1 E

```
enum ArnAtomicOp::E
```

Enumerator

None	
BitSet	
AddInt	
AddReal	
N	Max index.

Definition at line 71 of file ArnEvent.hpp.

14.5.2.2 NS

```
enum ArnAtomicOp::NS
```

Enumerator

NsEnum	
NsCom	

Definition at line 81 of file ArnEvent.hpp.

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

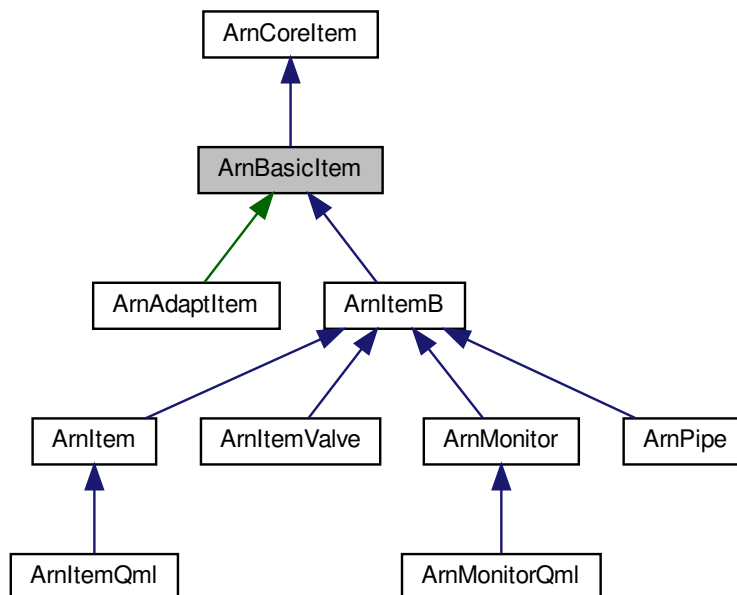
- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)

14.6 ArnBasicItem Class Reference

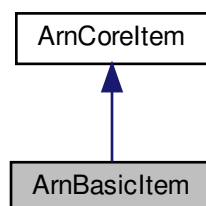
Base class handle for an *Arn Data Object*.

```
#include <ArnBasicItem.hpp>
```

Inheritance diagram for ArnBasicItem:



Collaboration diagram for ArnBasicItem:



Public Member Functions

- [ArnBasicItem](#) ()
Standard constructor of a closed handle.
- virtual [~ArnBasicItem](#) ()
- bool [open](#) (const QString &path)
Open a handle to an [Arn Data Object](#)
- void [close](#) ()
Close the handle.
- void [destroyLink](#) (bool isGlobal=true)
Destroy the [Arn Data Object](#)
- void [destroyLinkLocal](#) ()
Destroy the local [Arn Data Object](#)
- bool [isOpen](#) () const
State of the handle.
- QString [path](#) ([Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#)) const
Path of the [Arn Data Object](#)
- QString [name](#) ([Arn::NameF](#) nameF) const
Name of the [Arn Data Object](#)
- void [setReference](#) (void *reference)
Set an associated external reference.
- void * [reference](#) () const
Get the stored external reference.
- uint [itemId](#) () const
Get the id for this [ArnItem](#).
- uint [linkId](#) () const
Get the id for this [Arn Data Object](#)
- int [refCount](#) () const
Get the number of refs to this [Arn Data Object](#)
- bool [isFolder](#) () const
- bool [isProvider](#) () const
- [Arn::DataType](#) type () const
The type stored in the [Arn Data Object](#)
- void [setIgnoreSameValue](#) (bool isIgnore=true)
Set skipping of equal value.
- bool [isIgnoreSameValue](#) () const
- void [addMode](#) ([Arn::ObjectMode](#) mode)
Add general mode settings for this [Arn Data Object](#)
- [Arn::ObjectMode](#) [getMode](#) () const
Use with care, link must be "referenced" before use, otherwise it might have been deleted.
- [Arn::ObjectSyncMode](#) [syncMode](#) () const
- [ArnBasicItem](#) & [setBiDirMode](#) ()
Set general mode as Bidirectional for this [Arn Data Object](#)
- bool [isBiDirMode](#) () const
- [ArnBasicItem](#) & [setPipeMode](#) ()
Set general mode as Pipe for this [Arn Data Object](#)
- bool [isPipeMode](#) () const
- [ArnBasicItem](#) & [setSaveMode](#) ()
Set general mode as Save for this [Arn Data Object](#)
- bool [isSaveMode](#) () const
- void [setAtomicOpProvider](#) ()

Set this Arn Data Object as Atomic Operator Provider

- bool `isAtomicOpProvider ()` const
- `ArnBasicItem` & `setMaster ()`

Set client session sync mode as Master for this ArnItem.

- bool `isMaster ()` const
- `ArnBasicItem` & `setAutoDestroy ()`

Set client session sync mode as AutoDestroy for this ArnItem.

- bool `isAutoDestroy ()` const
- void `arnImport (const QByteArray &data, int ignoreSame=Arn::SameValue::DefaultAction)`

Import data to an Arn Data Object

- QByteArray `arnExport ()` const
- int `toInt (bool *isOk=arnNullptr)` const
- double `toDouble (bool *isOk=arnNullptr)` const
- `ARNREAL toReal (bool *isOk=arnNullptr)` const
- QString `toString (bool *isOk=arnNullptr)` const
- QByteArray `toByteArray (bool *isOk=arnNullptr)` const
- QVariant `toVariant (bool *isOk=arnNullptr)` const
- bool `toBool (bool *isOk=arnNullptr)` const
- uint `toUInt (bool *isOk=arnNullptr)` const
- quint64 `toInt64 (bool *isOk=arnNullptr)` const
- quint64 `toUInt64 (bool *isOk=arnNullptr)` const
- `ArnBasicItem` & `operator= (const ArnBasicItem &other)`
- `ArnBasicItem` & `operator= (int val)`
- `ArnBasicItem` & `operator= (ARNREAL val)`
- `ArnBasicItem` & `operator= (const QString &val)`
- `ArnBasicItem` & `operator= (const QByteArray &val)`
- `ArnBasicItem` & `operator= (const QVariant &val)`
- `ArnBasicItem` & `operator= (const char *val)`
- `ArnBasicItem` & `operator= (uint val)`
- `ArnBasicItem` & `operator= (quint64 val)`
- `ArnBasicItem` & `operator= (quint64 val)`
- `ArnBasicItem` & `operator+= (int val)`
- `ArnBasicItem` & `operator+= (ARNREAL val)`
- void `setValue (const ArnBasicItem &other, int ignoreSame=Arn::SameValue::DefaultAction)`
- void `setValue (int value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign an integer to an Arn Data Object

- void `setValue (ARNREAL value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign an ARNREAL to an Arn Data Object

- void `setValue (bool value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign a bool to an Arn Data Object

- void `setValue (const QString &value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign a QString to an Arn Data Object

- void `setValue (const QByteArray &value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign a QByteArray to an Arn Data Object

- void `setValue (const QVariant &value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign a QVariant to an Arn Data Object

- void `setValue (const char *value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign a char to an Arn Data Object*

- void `setValue (uint value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign an unsigned int to an Arn Data Object

- void `setValue (quint64 value, int ignoreSame=Arn::SameValue::DefaultAction)`

Assign an int 64 bit to an Arn Data Object

- void `setValue (quint64 value, int ignoreSame=Arn::SameValue::DefaultAction)`

- Assign an unsigned int 64 bit to an [Arn Data Object](#)*

 - void [setBits](#) (int mask, int value, int ignoreSame=[Arn::SameValue::DefaultAction](#))

AtomicOp assign an integer to specified bits in an [Arn Data Object](#)
- void [addValue](#) (int value)

AtomicOp adds an integer to an [Arn Data Object](#)
- void [addValue](#) ([ARNREAL](#) value)

AtomicOp adds an ARNREAL to an [Arn Data Object](#)
- QThread * [thread](#) () const

Get the thread affinity of this [ArnBasicItem](#).
- void [setEventHandler](#) (QObject *[eventHandler](#))

Set event handler for this [ArnBasicItem](#).
- QObject * [eventHandler](#) () const

Get the event handler of this [ArnBasicItem](#).
- void [setUncrossed](#) (bool [isUncrossed](#)=true)

Set a Bidirectional item as Uncrossed.
- bool [isUncrossed](#) () const

Get the Uncrossed state of an object.
- bool [isAssigning](#) () const

Tells if this [ArnItem](#) is assigned right now.

Friends

- class [ArnBasicItemEventHandler](#)

14.6.1 Detailed Description

Base class handle for an [Arn Data Object](#).

About ArnItem access

See [ArnItem](#).

[ArnBasicItem](#) is the basic way to get a handle (pointer) for accessing an [Arn Data Object](#). It is fast, small and is not based on QObject. As such it can not use signals and slots, but it can provide ArnEvents (based on QEvents) to be sent to any QObject based receiver.

There can be any amount of [ArnBasicItem](#):s opened (pointing) to the same [Arn Data object](#). Deleting the [ArnBasicItem](#) won't effect the [Arn Data object](#).

This class is not thread-safe, but the [Arn Data object](#) is, so each thread should have it's own handles i.e. [ArnBasicItem](#) instances.

Example usage

```

// In class declare
ArnBasicItem _arnTime;
MyReceiver _myRec; // QObject derived

// In class code
_arnTime.open("//Chat/Time/value");
_arnTime.setEventHandler( &_myRec);
_arnTime = "Undefined ...";

void MyReceiver::customEvent( QEvent* ev)
{
    // Is setup as ArnEvent handler for my ArnBasicItem.
    // Handler must finish with ArnBasicItemEventHandler::defaultEvent( ev).

    int evIdx = ev->type() - ArnEvent::baseType();
    switch (evIdx) {
    case ArnEvent::Idx::ValueChanged:
    {
        ArnEvValueChanged* e = static_cast<ArnEvValueChanged*>( ev);
        ArnBasicItem* item = static_cast<ArnBasicItem*>( e->
target());
        if (!item) break; // No target, deleted/closed ...

        QByteArray val = e->valueData() ? *e->valueData() : item->
toByteArray();
        qDebug() << "MyReceiver ArnEvValueChanged: inItemPath=" << item->path()
<< " value=" << val;
    }
    default:
        break;
    }

    ArnBasicItemEventHandler::defaultEvent( ev);
}

```

Definition at line 120 of file ArnBasicItem.hpp.

14.6.2 Constructor & Destructor Documentation

14.6.2.1 ArnBasicItem()

```
ArnBasicItem::ArnBasicItem ( )
```

Standard constructor of a closed handle.

Definition at line 92 of file ArnBasicItem.cpp.

14.6.2.2 ~ArnBasicItem()

```
ArnBasicItem::~~ArnBasicItem ( ) [virtual]
```

Definition at line 106 of file ArnBasicItem.cpp.

14.6.3 Member Function Documentation

14.6.3.1 addMode()

```
void ArnBasicItem::addMode (
    Arn::ObjectMode mode )
```

Add *general mode* settings for this *Arn Data Object*

If this *ArnItem* is in closed state, the added modes will be stored and the real mode change is done when this *ArnItem* is opened to an *Arn Data Object*. This implies that *ArnItems* can benefit from setting *modes* before opening.

Parameters

in	<i>mode</i>	The <i>modes</i> to be added.
----	-------------	-------------------------------

See also

[getMode\(\)](#)
[Modes](#)

Definition at line 421 of file ArnBasicItem.cpp.

14.6.3.2 addValue() [1/2]

```
void ArnBasicItem::addValue (
    int value )
```

AtomicOp adds an *integer* to an [Arn Data Object](#)

Operation is done atomically. If *bidir*, it can also be done remotely by an AtomicOpProvider

Parameters

in	<i>value</i>	to be added to this Arn Data Object
----	--------------	---

See also

[setAtomicOpProvider\(\)](#)

Definition at line 1101 of file ArnBasicItem.cpp.

14.6.3.3 addValue() [2/2]

```
void ArnBasicItem::addValue (
    ARNREAL value )
```

AtomicOp adds an *ARNREAL* to an [Arn Data Object](#)

Operation is done atomically. If *bidir*, it can also be done remotely by an AtomicOpProvider

Parameters

in	<i>value</i>	to be added to this Arn Data Object
----	--------------	---

See also

[setAtomicOpProvider\(\)](#)

Definition at line 1116 of file ArnBasicItem.cpp.

14.6.3.4 arnExport()

```
QByteArray ArnBasicItem::arnExport ( ) const
```

Returns

A data blob representing the *Arn Data Object*

See also

[arnImport\(\)](#)

Definition at line 623 of file ArnBasicItem.cpp.

14.6.3.5 arnImport()

```
void ArnBasicItem::arnImport (
    const QByteArray & data,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Import data to an *Arn Data Object*

Data blob from a previous [arnExport \(\)](#) can be imported. This is essentially assigning the *Arn Data Object* with same as exported.

Parameters

in	<i>data</i>	is the data blob
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[arnExport\(\)](#)
[setIgnoreSameValue\(\)](#)

Definition at line 512 of file ArnBasicItem.cpp.

14.6.3.6 close()

```
void ArnBasicItem::close ( )
```

Close the handle.

Definition at line 153 of file ArnBasicItem.cpp.

14.6.3.7 destroyLink()

```
void ArnBasicItem::destroyLink (
    bool isGlobal = true )
```

Destroy the *Arn Data Object*

The link (*Arn Data Object*) will be removed locally and optionally from server and all connected clients. Server is allways forcing global destroy.

Parameters

in	<i>isGlobal</i>	If true, removes from server and all connected clients, otherwise only local link.
----	-----------------	--

See also

[destroyLinkLocal\(\)](#)

Definition at line 178 of file ArnBasicItem.cpp.

14.6.3.8 destroyLinkLocal()

```
void ArnBasicItem::destroyLinkLocal ( ) [inline]
```

Destroy the local *Arn Data Object*

The link (*Arn Data Object*) will be removed locally. Server is allways forcing global destroy.

See also

[destroyLink\(\)](#)

Definition at line 156 of file ArnBasicItem.hpp.

14.6.3.9 eventHandler()

```
QObject * ArnBasicItem::eventHandler ( ) const
```

Get the event handler of this [ArnBasicItem](#).

Returns

the event handler

See also

[setEventHandler\(\)](#)
[thread\(\)](#)

Definition at line 1209 of file ArnBasicItem.cpp.

14.6.3.10 getMode()

```
Arn::ObjectMode ArnBasicItem::getMode ( ) const
```

Use with care, link must be "referenced" before use, otherwise it might have been deleted.

Returns

The *general mode* of the [Arn Data Object](#)

See also

[addMode\(\)](#)
[Modes](#)

Definition at line 439 of file ArnBasicItem.cpp.

14.6.3.11 isAssigning()

```
bool ArnBasicItem::isAssigning ( ) const
```

Tells if this [ArnItem](#) is assigned right now.

Typically used to stop endless recursion due to signal/slot direct call when assigned

Return values

<i>true</i>	if beeing assigned right now.
-------------	-------------------------------

Definition at line 1233 of file ArnBasicItem.cpp.

14.6.3.12 isAtomicOpProvider()

```
bool ArnBasicItem::isAtomicOpProvider ( ) const
```

Return values

<i>true</i>	if this is a <i>Atomic Operator Provider</i>
-------------	--

See also

[setAtomicOpProvider\(\)](#)

Definition at line 379 of file ArnBasicItem.cpp.

14.6.3.13 isAutoDestroy()

```
bool ArnBasicItem::isAutoDestroy ( ) const
```

Return values

<i>true</i>	if <i>AutoDestroy mode</i>
-------------	----------------------------

See also

[setAutoDestroy\(\)](#)

Definition at line 415 of file ArnBasicItem.cpp.

14.6.3.14 isBiDirMode()

```
bool ArnBasicItem::isBiDirMode ( ) const
```

Return values

<i>true</i>	if <i>Bidirectional</i>
-------------	-------------------------

See also

[setBiDirMode\(\)](#)
[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 309 of file ArnBasicItem.cpp.

14.6.3.15 isFolder()

```
bool ArnBasicItem::isFolder ( ) const
```

Return values

<i>true</i>	if this ArnItem is a folder
-------------	---

Definition at line 190 of file ArnBasicItem.cpp.

14.6.3.16 isIgnoreSameValue()

```
bool ArnBasicItem::isIgnoreSameValue ( ) const
```

Return values

<i>true</i>	if skipping equal values
-------------	--------------------------

See also

[setIgnoreSameValue\(\)](#)

Definition at line 464 of file ArnBasicItem.cpp.

14.6.3.17 isMaster()

```
bool ArnBasicItem::isMaster ( ) const
```

Return values

<i>true</i>	if <i>Master mode</i>
-------------	-----------------------

See also

[setMaster\(\)](#)
[Modes](#)

Definition at line 398 of file ArnBasicItem.cpp.

14.6.3.18 isOpen()

```
bool ArnBasicItem::isOpen ( ) const
```

State of the handle.

Return values

<i>true</i>	if this ArnItem is open
-------------	---

Definition at line 184 of file ArnBasicItem.cpp.

14.6.3.19 isPipeMode()

```
bool ArnBasicItem::isPipeMode ( ) const
```

Return values

<i>true</i>	if <i>Pipe mode</i>
-------------	---------------------

See also

[setPipeMode\(\)](#)
[Modes](#)
[Pipe Arn Data Objects](#)

Definition at line 338 of file ArnBasicItem.cpp.

14.6.3.20 isProvider()

```
bool ArnBasicItem::isProvider ( ) const
```

Return values

<i>true</i>	if this ArnItem is a provider
-------------	---

See also

[setBiDirMode\(\)](#)
[Modes](#)

Definition at line 198 of file ArnBasicItem.cpp.

14.6.3.21 isSaveMode()

```
bool ArnBasicItem::isSaveMode ( ) const
```

Return values

<i>true</i>	if <i>Save mode</i>
-------------	---------------------

See also

[setSaveMode\(\)](#)
[Modes](#)
[Persistent Arn Data Objects](#)

Definition at line 360 of file ArnBasicItem.cpp.

14.6.3.22 isUncrossed()

```
bool ArnBasicItem::isUncrossed ( ) const
```

Get the Uncrossed state of an object.

Return values

<i>true</i>	if Uncrossed is set or <i>Arn Data Object</i> is not in Bidirectional mode.
-------------	---

See also

[setUncrossed\(\)](#)
[setBiDirMode\(\)](#)
[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 1225 of file ArnBasicItem.cpp.

14.6.3.23 itemId()

```
uint ArnBasicItem::itemId ( ) const
```

Get the *id* for this [ArnItem](#).

The [ArnItem](#) *id* is unique within its running program. Even if 2 [ArnItems](#) are pointing to the same [Arn Data Object](#), they have different *item id*.

Returns

id for this [ArnItem](#)

See also

[linkId\(\)](#)

Definition at line 504 of file [ArnBasicItem.cpp](#).

14.6.3.24 linkId()

```
uint ArnBasicItem::linkId ( ) const
```

Get the *id* for this [Arn Data Object](#)

The link ([Arn Data Object](#)) *id* is unique within its running program. If 2 [ArnItems](#) are pointing to the same [Arn Data Object](#), they have same *link id*.

Returns

Id for the [Arn Data Object](#), 0 if closed

See also

[itemId\(\)](#)

Definition at line 214 of file [ArnBasicItem.cpp](#).

14.6.3.25 name()

```
QString ArnBasicItem::name (
    Arn::NameF nameF ) const
```

Name of the [Arn Data Object](#)

Parameters

in	<i>nameF</i>	The format of the returned name
----	--------------	---------------------------------

Returns

The object name

Definition at line 480 of file ArnBasicItem.cpp.

14.6.3.26 open()

```
bool ArnBasicItem::open (
    const QString & path )
```

Open a handle to an *Arn Data Object*

Parameters

in	<i>path</i>	The <i>Arn Data Object</i> path e.g. "//Measure/Water/Level/value"
----	-------------	--

Return values

<i>false</i>	if error
--------------	----------

Definition at line 147 of file ArnBasicItem.cpp.

14.6.3.27 operator+=() [1/2]

```
ArnBasicItem & ArnBasicItem::operator+= (
    int val )
```

Definition at line 827 of file ArnBasicItem.cpp.

14.6.3.28 operator+=() [2/2]

```
ArnBasicItem & ArnBasicItem::operator+= (
    ARNREAL val )
```

Definition at line 834 of file ArnBasicItem.cpp.

14.6.3.29 operator=() [1/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    const ArnBasicItem & other )
```

Definition at line 757 of file ArnBasicItem.cpp.

14.6.3.30 operator=() [2/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    int val )
```

Definition at line 764 of file ArnBasicItem.cpp.

14.6.3.31 operator=() [3/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    ARNREAL val )
```

Definition at line 771 of file ArnBasicItem.cpp.

14.6.3.32 operator=() [4/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    const QString & val )
```

Definition at line 778 of file ArnBasicItem.cpp.

14.6.3.33 operator=() [5/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    const QByteArray & val )
```

Definition at line 785 of file ArnBasicItem.cpp.

14.6.3.34 operator=() [6/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    const QVariant & val )
```

Definition at line 820 of file ArnBasicItem.cpp.

14.6.3.35 operator=() [7/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    const char * val )
```

Definition at line 792 of file ArnBasicItem.cpp.

14.6.3.36 operator=() [8/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    uint val )
```

Definition at line 799 of file ArnBasicItem.cpp.

14.6.3.37 operator=() [9/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    qint64 val )
```

Definition at line 806 of file ArnBasicItem.cpp.

14.6.3.38 operator=() [10/10]

```
ArnBasicItem & ArnBasicItem::operator= (
    quint64 val )
```

Definition at line 813 of file ArnBasicItem.cpp.

14.6.3.39 path()

```
QString ArnBasicItem::path (
    Arn::NameF nameF = Arn::NameF::EmptyOk ) const
```

Path of the *Arn Data Object*

Parameters

in	<i>nameF</i>	The format of the returned path
----	--------------	---------------------------------

Returns

The object path

Definition at line 472 of file ArnBasicItem.cpp.

14.6.3.40 refCount()

```
int ArnBasicItem::refCount ( ) const
```

Get the number of refs to this *Arn Data Object*

Returns

The number of refs for the *Arn Data Object*, -1 if closed

Definition at line 222 of file ArnBasicItem.cpp.

14.6.3.41 reference()

```
void * ArnBasicItem::reference ( ) const
```

Get the stored external reference.

Returns

The associated external reference

See also

[setReference\(\)](#)

Definition at line 496 of file ArnBasicItem.cpp.

14.6.3.42 setAtomicOpProvider()

```
void ArnBasicItem::setAtomicOpProvider ( )
```

Set this *Arn Data Object* as *Atomic Operator Provider*

The atomic operation is performed at this object

Definition at line 370 of file ArnBasicItem.cpp.

14.6.3.43 setAutoDestroy()

```
ArnBasicItem & ArnBasicItem::setAutoDestroy ( )
```

Set client session *sync mode* as *AutoDestroy* for this [ArnItem](#).

This [ArnItem](#) at client side is setup for auto destruction.

Precondition

This must be set before [open\(\)](#).

Definition at line 404 of file ArnBasicItem.cpp.

14.6.3.44 setBiDirMode()

```
ArnBasicItem & ArnBasicItem::setBiDirMode ( )
```

Set *general mode* as *Bidirectional* for this [Arn Data Object](#)

A two way object, typically for validation or pipe

See also

[Modes](#)
[Bidirectional Arn Data Objects](#)

Bidirectional-mode is the pair of value & provider

Definition at line 292 of file ArnBasicItem.cpp.

14.6.3.45 setBits()

```
void ArnBasicItem::setBits (
    int mask,
    int value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

AtomicOp assign an *integer* to specified bits in an [Arn Data Object](#)

Operation is done atomically. If bidir, it can also be done remotely by an AtomicOpProvider

Parameters

in	<i>mask</i>	to specify bits that is affected
in	<i>value</i>	to be assigned to affected bits
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setAtomicOpProvider\(\)](#)
[setIgnoreSameValue\(\)](#)

Definition at line 1075 of file ArnBasicItem.cpp.

14.6.3.46 setEventHandler()

```
void ArnBasicItem::setEventHandler (
    QObject * eventHandler )
```

Set event handler for this [ArnBasicItem](#).

The event handler must be QObject based

Parameters

in	<i>eventHandler</i>	to be assigned
----	---------------------	----------------

See also

[eventHandler\(\)](#)
[thread\(\)](#)

Definition at line 1200 of file ArnBasicItem.cpp.

14.6.3.47 setIgnoreSameValue()

```
void ArnBasicItem::setIgnoreSameValue (
    bool isIgnore = true )
```

Set skipping of equal value.

Parameters

in	<i>isIgnore</i>	If true, assignment of equal value don't give a changed signal.
----	-----------------	---

Definition at line 456 of file ArnBasicItem.cpp.

14.6.3.48 setMaster()

```
ArnBasicItem & ArnBasicItem::setMaster ( )
```

Set client session *sync mode* as *Master* for this [ArnItem](#).

This [ArnItem](#) at client side is set as default generator of data.

Precondition

This must be set before [open\(\)](#).

See also

[Modes](#)

Definition at line 387 of file ArnBasicItem.cpp.

14.6.3.49 setPipeMode()

```
ArnBasicItem & ArnBasicItem::setPipeMode ( )
```

Set *general mode* as *Pipe* for this [Arn Data Object](#)

Implies *Bidir*.

See also

[Modes](#)

[Pipe Arn Data Objects](#)

Definition at line 319 of file ArnBasicItem.cpp.

14.6.3.50 setReference()

```
void ArnBasicItem::setReference (
    void * reference )
```

Set an associated external reference.

This is typically used when having many *ArnItems* changed signal connected to a common slot. The slot can then discover the signalling [ArnItem](#):s associated structure for further processing.

Parameters

in	<i>reference</i>	Any external structure or id.
----	------------------	-------------------------------

See also

[reference\(\)](#)

Definition at line 488 of file ArnBasicItem.cpp.

14.6.3.51 setSaveMode()

```
ArnBasicItem & ArnBasicItem::setSaveMode ( )
```

Set *general mode* as *Save* for this *Arn Data Object*

Data is persistent and will be saved

Precondition

The persistent service must be started at the server.

See also

[Modes](#)

[Persistent Arn Data Objects](#)

Definition at line 348 of file ArnBasicItem.cpp.

14.6.3.52 setUncrossed()

```
void ArnBasicItem::setUncrossed (
    bool isUncrossed = true )
```

Set a Bidirectional item as Uncrossed.

The two way object is not twisted at writes, i.e. exactly the same object is read and written. This has no effect on an *Arn Data Object* that not is in Bidirectional mode.

See also

[isUncrossed\(\)](#)

[Modes](#)

[Bidirectional Arn Data Objects](#)

Definition at line 1217 of file ArnBasicItem.cpp.

14.6.3.53 setValue() [1/11]

```
void ArnBasicItem::setValue (
    const ArnBasicItem & other,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Definition at line 841 of file ArnBasicItem.cpp.

14.6.3.54 setValue() [2/11]

```
void ArnBasicItem::setValue (
    int value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *integer* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 883 of file ArnBasicItem.cpp.

14.6.3.55 setValue() [3/11]

```
void ArnBasicItem::setValue (
    ARNREAL value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *ARNREAL* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 908 of file ArnBasicItem.cpp.

14.6.3.56 setValue() [4/11]

```
void ArnBasicItem::setValue (
    bool value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *bool* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 933 of file ArnBasicItem.cpp.

14.6.3.57 setValue() [5/11]

```
void ArnBasicItem::setValue (
    const QString & value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *QString* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 958 of file ArnBasicItem.cpp.

14.6.3.58 setValue() [6/11]

```
void ArnBasicItem::setValue (
    const QByteArray & value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *QByteArray* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 983 of file ArnBasicItem.cpp.

14.6.3.59 setValue() [7/11]

```
void ArnBasicItem::setValue (
    const QVariant & value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *QVariant* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 1008 of file ArnBasicItem.cpp.

14.6.3.60 setValue() [8/11]

```
void ArnBasicItem::setValue (
    const char * value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign a *char** to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 1033 of file ArnBasicItem.cpp.

14.6.3.61 setValue() [9/11]

```
void ArnBasicItem::setValue (
    uint value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *unsigned int* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 1039 of file ArnBasicItem.cpp.

14.6.3.62 setValue() [10/11]

```
void ArnBasicItem::setValue (
    qint64 value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *int 64 bit* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 1051 of file ArnBasicItem.cpp.

14.6.3.63 setValue() [11/11]

```
void ArnBasicItem::setValue (
    quint64 value,
    int ignoreSame = Arn::SameValue::DefaultAction )
```

Assign an *unsigned int 64 bit* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 1063 of file ArnBasicItem.cpp.

14.6.3.64 syncMode()

```
Arn::ObjectSyncMode ArnBasicItem::syncMode ( ) const
```

Returns

The client session *sync mode* of an *Arn Data Object*

See also

[Modes](#)

Definition at line 281 of file ArnBasicItem.cpp.

14.6.3.65 thread()

```
QThread * ArnBasicItem::thread ( ) const
```

Get the thread affinity of this [ArnBasicItem](#).

The affinity (see QObject) is set when the [ArnBasicItem](#) is created and bound to an internal QObject based event handler. When a custom event handler is set, its affinity is used.

Returns

the thread affinity

See also

[setEventHandler\(\)](#)

Definition at line 1131 of file ArnBasicItem.cpp.

14.6.3.66 toBool()

```
bool ArnBasicItem::toBool (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *bool*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from Int.

Definition at line 725 of file ArnBasicItem.cpp.

14.6.3.67 toByteArray()

```
QByteArray ArnBasicItem::toByteArray (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *QByteArray*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 685 of file ArnBasicItem.cpp.

14.6.3.68 toDouble()

```
double ArnBasicItem::toDouble (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *double*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 709 of file ArnBasicItem.cpp.

14.6.3.69 toInt()

```
int ArnBasicItem::toInt (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *integer*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 701 of file ArnBasicItem.cpp.

14.6.3.70 toInt64()

```
qint64 ArnBasicItem::toInt64 (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *int 64 bit*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 741 of file ArnBasicItem.cpp.

14.6.3.71 toReal()

```
ARNREAL ArnBasicItem::toReal (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *ARNREAL*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 717 of file ArnBasicItem.cpp.

14.6.3.72 toString()

```
QString ArnBasicItem::toString (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *QString*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 677 of file ArnBasicItem.cpp.

14.6.3.73 toUInt()

```
uint ArnBasicItem::toUInt (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *unsigned int*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 733 of file ArnBasicItem.cpp.

14.6.3.74 toUInt64()

```
quint64 ArnBasicItem::toUInt64 (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to an *unsigned int 64 bit*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 749 of file ArnBasicItem.cpp.

14.6.3.75 toVariant()

```
QVariant ArnBasicItem::toVariant (
    bool * isOk = arnNullptr ) const
```

Returns

Convert *Arn Data Object* to a *QVariant*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 693 of file ArnBasicItem.cpp.

14.6.3.76 type()

```
Arn::DataType ArnBasicItem::type ( ) const
```

The type stored in the *Arn Data Object*

Returns

The type stored

Definition at line 206 of file ArnBasicItem.cpp.

14.6.4 Friends And Related Function Documentation

14.6.4.1 ArnBasicItemEventHandler

```
friend class ArnBasicItemEventHandler [friend]
```

Definition at line 123 of file ArnBasicItem.hpp.

The documentation for this class was generated from the following files:

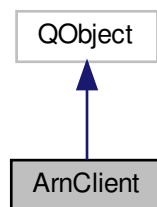
- [src/ArnInc/ArnBasicItem.hpp \(4.0.0\)](#)
- [src/ArnBasicItem.cpp \(4.0.0\)](#)

14.7 ArnClient Class Reference

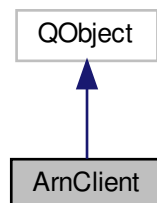
Class for connecting to an *Arn Server*.

```
#include <ArnClient.hpp>
```

Inheritance diagram for ArnClient:



Collaboration diagram for ArnClient:



Classes

- struct [HostAddrPort](#)

Public Types

- typedef [ArnClientConnectStat](#) [ConnectStat](#)
- typedef [Arn::ClientSyncMode](#) [SyncMode](#)
- typedef [QList< HostAddrPort >](#) [HostList](#)

Signals

- void [tcpError](#) (const [QString](#) &errorText, [QAbstractSocket::SocketError](#) socketError)
Signal emitted when a connection tcp error occur.
- void [tcpConnected](#) (const [QString](#) &arnHost, [quint16](#) port)
Signal emitted when the tcp connection is successfull.
- void [tcpDisConnected](#) ()
Signal emitted when the tcp connection is broken (has been successfull).
- void [connectionStatusChanged](#) (int status, int curPrio)
Signal emitted when the connection status is changed.
- void [loginRequired](#) (int contextCode)
Signal emitted when the remote [ArnServer](#) demands a login.
- void [killRequested](#) ()
Signal emitted when the server request this client to kill its connection.
- void [chatReceived](#) (const [QString](#) &text, int prioType)
Signal emitted when a chat message is received from the server.

Public Member Functions

- [ArnClient](#) ([QObject](#) *parent=[arnNullptr](#))
- [~ArnClient](#) ()
- void [clearArnList](#) (int prioFilter=-1)
Clear the [Arn](#) connection list.
- [HostList](#) [arnList](#) (int prioFilter=-1) const
Return the [Arn](#) connection list.
- void [addToArnList](#) (const [QString](#) &arnHost, [quint16](#) port=0, int prio=0)
Add an [Arn](#) Server to the [Arn](#) connection list.
- void [connectToArnList](#) ()
Connect to an [Arn](#) Server in the [Arn](#) connection list.
- void [connectToArn](#) (const [QString](#) &arnHost, [quint16](#) port=0)
Connect to an [Arn](#) Server
- void [disconnectFromArn](#) ()
Disconnect from an [Arn](#) Server
- void [loginToArn](#) (const [QString](#) &userName, const [QString](#) &password, [Arn::Allow](#) allow=[Arn::Allow::All](#))
Login to an [Arn](#) Server
- void [loginToArnHashed](#) (const [QString](#) &userName, const [QString](#) &passwordHashed, [Arn::Allow](#) allow=[Arn::Allow::All](#))
Login to an [Arn](#) Server using hashed password.
- void [close](#) ()

- Close sharing with an Arn Server*

 - bool `setMountPoint` (const QString &path)
 - Set the sharing tree path.*
 - bool `addMountPoint` (const QString &localPath, const QString &remotePath=QString())
 - Add a sharing tree path.*
 - bool `removeMountPoint` (const QString &localPath)
 - Remove a sharing tree path.*
 - `ConnectStat connectStatus` () const
 - Return the Arn connection status.*
 - void `setAutoConnect` (bool isAuto, int retryTime=2)
 - Set automatic reconnect.*
 - void `registerClient` (const QString &id)
 - Register this client to be available with id.*
 - QString `id` () const
 - Get the id of this client.*
 - int `receiveTimeout` () const
 - Get receive data timeout (base time)*
 - void `setReceiveTimeout` (int receiveTimeout)
 - Set receive data timeout (base time)*
 - bool `isDemandLogin` () const
 - Get clients demand for login.*
 - void `setDemandLogin` (bool isDemandLogin)
 - Set clients demand for login.*
 - `SyncMode syncMode` () const
 - Get ClientSyncMode.*
 - void `setSyncMode` (SyncMode syncMode)
 - Set ClientSyncMode.*
 - QStringList `freePaths` () const
 - Returns current list of freePaths.*
 - void `setWhoIAm` (const Arn::XStringMap &whoIAmXsm)
 - Set clients human readable identification information.*
 - Arn::XStringMap `remoteWhoIAm` () const
 - Returns remote side (server) readable identification information.*
 - bool `isReContact` () const
 - Is last TCP connection a reContact.*
 - bool `isReConnect` () const
 - Is last Arn Connection a reConnect.*
 - void `chatSend` (const QString &text, int prioType)
 - Send chat message to server.*
 - void `abortKillRequest` ()
 - Send abort kill request to server.*
 - bool `getTraffic` (quint64 &in, quint64 &out) const
 - Get traffic metrics.*

Static Public Member Functions

- static ArnClient * `getClient` (const QString &id)
- Get a client by its id.*
- static QString `passwordHash` (const QString &password)
- Generate a hashed password from clear text password.*

14.7.1 Detailed Description

Class for connecting to an [Arn Server](#).

[About Sharing Arn Data Objects About Sync Rules](#)

Connection can be made to a specific Host by [connectToArn\(\)](#). It's also possible to define an [Arn Connection List](#). Each host address is added to the list with a priority. The priority is used to control the order at which the host addresses will be tried for connection. Lowest priority number is tried first. Connection trials are started with [connectToArnList\(\)](#). The priority can also be used for selection in [clearArnList\(\)](#) and [arnList\(\)](#).

Example usage

```
// In class declare
ArnClient _arnClient;

// In class code
_arnClient.connectToArn("localhost");
_arnClient.addMountPoint("/");
_arnClient.setAutoConnect( true);
```

Examples:

[ArnDemoChat/MainWindow.hpp](#).

Definition at line 104 of file ArnClient.hpp.

14.7.2 Member Typedef Documentation

14.7.2.1 ConnectStat

```
typedef ArnClientConnectStat ArnClient::ConnectStat
```

Definition at line 110 of file ArnClient.hpp.

14.7.2.2 HostList

```
typedef QList<HostAddrPort> ArnClient::HostList
```

Definition at line 121 of file ArnClient.hpp.

14.7.2.3 SyncMode

```
typedef Arn::ClientSyncMode ArnClient::SyncMode
```

Definition at line 111 of file ArnClient.hpp.

14.7.3 Constructor & Destructor Documentation

14.7.3.1 ArnClient()

```
ArnClient::ArnClient (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 253 of file ArnClient.cpp.

14.7.3.2 ~ArnClient()

```
ArnClient::~ArnClient ( )
```

Definition at line 269 of file ArnClient.cpp.

14.7.4 Member Function Documentation

14.7.4.1 abortKillRequest()

```
void ArnClient::abortKillRequest ( )
```

Send abort kill request to server.

The server can request client to kill connection. This method is used to request an abort of the kill request. This method can be called any time but it will only be considered during a kill countdown.

See also

[killRequested\(\)](#)

Definition at line 621 of file ArnClient.cpp.

14.7.4.2 addMountPoint()

```
bool ArnClient::addMountPoint (
    const QString & localPath,
    const QString & remotePath = QString() )
```

Add a sharing tree path.

Mountpoint is an association to the similarity of mounting a "remote filesystem". In [Arn](#), the remote "file system" can be at different sub path than the local mountpoint, e.g. a client having mountpoint local="/a/b/" remote="/r/" and opening an [Arn Data Object](#) at "/a/b/c" will have the object *c* shared with the server at its path "/r/c". However if *remotePath* is not specified, it will be same as *localPath*. In the above example, the *c* object will then be shared with the server at its path "/a/b/c".

Parameters

in	<i>localPath</i>	is the local sharing tree.
in	<i>remotePath</i>	is the remote sharing tree. If empty, same as <i>localPath</i> .

Return values

<i>false</i>	if error.
--------------	-----------

See also

[Sharing Arn Data Objects](#)

Definition at line 385 of file ArnClient.cpp.

14.7.4.3 addToArnList()

```
void ArnClient::addToArnList (
    const QString & arnHost,
    quint16 port = 0,
    int prio = 0 )
```

Add an [Arn Server](#) to the [Arn](#) connection list.

Parameters

in	<i>arnHost</i>	is host name or ip address, e.g. "192.168.1.1".
in	<i>port</i>	is the host port, 0 gives Arn::defaultTcpPort .
in	<i>prio</i>	gives the sorting (connection) order and can be used for selection filter.

See also

[clearArnList\(\)](#)
[arnList\(\)](#)
[Arn::makeHostWithInfo\(\)](#)

Definition at line 293 of file ArnClient.cpp.

14.7.4.4 arnList()

```
ArnClient::HostList ArnClient::arnList (
    int prioFilter = -1 ) const
```

Return the [Arn](#) connection list.

Parameters

in	<i>prioFilter</i>	selects hosts in the list with this pri. Default -1 selects all.
----	-------------------	--

Return values

<i>the</i>	selected Arn connection list.
------------	---

See also

[addToArnList\(\)](#)

Definition at line 285 of file ArnClient.cpp.

14.7.4.5 chatReceived

```
void ArnClient::chatReceived (
    const QString & text,
    int prioType ) [signal]
```

Signal emitted when a chat message is received from the server.

Parameters

in	<i>text</i>	is the message.
in	<i>prioType</i>	is the priority of the message (1=Hi 2=Normal).

See also

[chatSend\(\)](#)

14.7.4.6 chatSend()

```
void ArnClient::chatSend (
    const QString & text,
    int prioType )
```

Send chat message to server.

This is used for a chat session between client and server.

Parameters

in	<i>text</i>	is the message.
in	<i>prioType</i>	is the priority of the message (1=Hi 2=Normal).

See also

[chatReceived\(\)](#)

Definition at line 611 of file ArnClient.cpp.

14.7.4.7 clearArnList()

```
void ArnClient::clearArnList (
    int prioFilter = -1 )
```

Clear the [Arn](#) connection list.

Typically used to start making a new [Arn](#) connection list.

Parameters

in	<i>prioFilter</i>	selects hosts in the list with this pri, to be removed. Default -1 removes all.
----	-------------------	---

See also

[addToArnList\(\)](#)

Definition at line 277 of file ArnClient.cpp.

14.7.4.8 close()

```
void ArnClient::close ( )
```

Close sharing with an [Arn Server](#)

Stop sharing [Arn objects](#) with the [Arn server](#). Similar to [disconnectFromArn\(\)](#). All pending data is written before disconnect. No synchronized [Arn objects](#) are remembered. This implies that it's not possible to continue previous session. This function is aimed at later starting a new session from scratch.

Auto connection is also disabled.

See also

[disconnectFromArn\(\)](#)

[setAutoConnect\(\)](#)

[connectToArn\(\)](#)

Definition at line 354 of file ArnClient.cpp.

14.7.4.9 connectionStatusChanged

```
void ArnClient::connectionStatusChanged (
    int status,
    int curPrio ) [signal]
```

Signal emitted when the connection status is changed.

Parameters

in	<i>status</i>	is the new connection status ArnClient::ConnectStat .
in	<i>curPrio</i>	is the current priority of the connection in ArnList

See also

[curPrio\(\)](#)

14.7.4.10 connectStatus()

```
ArnClient::ConnectStat ArnClient::connectStatus ( ) const
```

Return the [Arn](#) connection status.

Return values

<i>the</i>	Arn connection status.
------------	--

Definition at line 364 of file ArnClient.cpp.

14.7.4.11 connectToArn()

```
void ArnClient::connectToArn (
    const QString & arnHost,
    quint16 port = 0 )
```

Connect to an [Arn Server](#)

Parameters

in	<i>arnHost</i>	is host name or ip address, e.g. "192.168.1.1".
in	<i>port</i>	is the host port, 0 gives Arn::defaultTcpPort .

See also

[Arn::makeHostWithInfo\(\)](#)
[connectToArnList\(\)](#)

Definition at line 314 of file ArnClient.cpp.

14.7.4.12 connectToArnList()

```
void ArnClient::connectToArnList ( )
```

Connect to an *Arn Server* in the *Arn* connection list.

Will scan the connection list once until a successful connection is made. If the end of the list is reached without connection, the `tcpError()` signal

See also

[connectToArn\(\)](#)

Definition at line 301 of file ArnClient.cpp.

14.7.4.13 disconnectFromArn()

```
void ArnClient::disconnectFromArn ( )
```

Disconnect from an *Arn Server*

Force disconnect from the *Arn server*, similar behaviour to losing connection. All pending data is written before disconnect. All *Arn objects* that has been setup to be synronized is still kept. This implies that it's possible to continue previous session by just connecting to the *Arn server* again.

Auto connection is also disabled.

See also

[close\(\)](#)
[setAutoConnect\(\)](#)
[connectToArn\(\)](#)

Definition at line 327 of file ArnClient.cpp.

14.7.4.14 freePaths()

```
QStringList ArnClient::freePaths ( ) const
```

Returns current list of freePaths.

A freePath can be used even if not logged in to an *ArnServer* that demands login. Also all children below freePath is free to use. Usage is restricted to read operations and alike from *ArnServer* to *ArnClient*. The list of freePaths is used to enable the operation requests to be transfered to *ArnServer*. *ArnServer* still decides what's allowed. The list is automatically transfered from *ArnServer* to *ArnClient* during the negotiation phase.

Returns

the freePath list.

See also

[ArnServer::addFreePath\(\)](#)

Definition at line 563 of file ArnClient.cpp.

14.7.4.15 getClient()

```
ArnClient * ArnClient::getClient (
    const QString & id ) [static]
```

Get a client by its id.

Parameters

<code>in</code>	<code>id</code>	if "" will always return 0.
-----------------	-----------------	-----------------------------

Returns

the found client, 0 = not found or id == ""

See also

[registerClient\(\)](#)

Definition at line 492 of file ArnClient.cpp.

14.7.4.16 getTraffic()

```
bool ArnClient::getTraffic (
    quint64 & in,
    quint64 & out ) const
```

Get traffic metrics.

Return values

<code>true</code>	if ok.
-------------------	--------

Parameters

<code>out</code>	<code>in</code>	is the clients received number of bytes.
<code>out</code>	<code>out</code>	is the clients sent number of bytes.

Definition at line 629 of file ArnClient.cpp.

14.7.4.17 id()

```
QString ArnClient::id ( ) const
```

Get the id of this client.

Returns

the id, "" = none (local)

See also

[registerClient\(\)](#)

Definition at line 498 of file ArnClient.cpp.

14.7.4.18 isDemandLogin()

```
bool ArnClient::isDemandLogin ( ) const
```

Get clients demand for login.

If any of server or client demand login, it must be used.

Return values

<i>true</i>	if client demand login.
-------------	-------------------------

See also

[setDemandLogin\(\)](#)

Definition at line 522 of file ArnClient.cpp.

14.7.4.19 isReConnect()

```
bool ArnClient::isReConnect ( ) const
```

Is last [Arn](#) Connection a reConnect.

ReConnect occurs if an [Arn](#) connection is successful, then lost and then restored due to autoConnect. Successful [Arn](#) connection gives a state change to [ConnectStat::Connected](#).

Return values

<i>true</i>	if this is a reConnect.
-------------	-------------------------

See also

[isReContact\(\)](#)
[setAutoConnect\(\)](#)
[connectionStatusChanged\(\)](#)

Definition at line 595 of file ArnClient.cpp.

14.7.4.20 isReContact()

```
bool ArnClient::isReContact ( ) const
```

Is last TCP connection a reContact.

ReContact occurs if a TCP connection is successful, then lost and then restored due to autoConnect. Successful TCP connection gives a state change to [ConnectStat::Negotiating](#).

Return values

<code>true</code>	if this is a reContact.
-------------------	-------------------------

See also

[isReConnect\(\)](#)
[setAutoConnect\(\)](#)
[connectionStatusChanged\(\)](#)

Definition at line 587 of file ArnClient.cpp.

14.7.4.21 killRequested

```
void ArnClient::killRequested ( ) [signal]
```

Signal emitted when the server request this client to kill its connection.

This request should normally be obeyed by the client. I.e. it should usually result in a call to [close\(\)](#).

See also

[abortKillRequest\(\)](#)

14.7.4.22 loginRequired

```
void ArnClient::loginRequired (
    int contextCode ) [signal]
```

Signal emitted when the remote [ArnServer](#) demands a login.

When this signal is emitted, a call to [loginToArn\(\)](#) must be done to complete the connection process.

Parameters

in	<i>contextCode</i>	is the situation context as: 0 = First login trial 1 = Server deny, login retry 2 = Client deny, server gave bad password (fake server?) 3 = Client deny, server not support login 4 = Client deny, server bad negotiate sequence
----	--------------------	---

See also

[loginToArn\(\)](#)

14.7.4.23 loginToArn()

```
void ArnClient::loginToArn (
    const QString & userName,
    const QString & password,
    Arn::Allow allow = Arn::Allow::All )
```

Login to an [Arn Server](#)

This routine must be called when the signal [loginRequired\(\)](#) is emitted. Otherwise the client will not be fully connected to the server, ie the appropriate access privileges will not be setup at server and client. If a reconnect occurs, usually due to tcp breakage, login process is handled automatically by ArnLib using the last used login credentials. If this automatic login is failed, signal [loginRequired\(\)](#) is emitted.

Parameters

in	<i>userName</i>	
in	<i>password</i>	
in	<i>allow</i>	is the permissions for the server actions to this client.

See also

[Arn::Allow](#)
[loginRequired](#);
[loginToArnHashed\(\)](#)

Definition at line 338 of file ArnClient.cpp.

14.7.4.24 loginToArnHashed()

```
void ArnClient::loginToArnHashed (
    const QString & userName,
    const QString & passwordHashed,
    Arn::Allow allow = Arn::Allow::All )
```

Login to an [Arn Server](#) using hashed password.

This behaves exactly as [loginToArn\(\)](#), except for password being hashed. The hashed password which can be generated by [ArnClient::passwordHash\(\)](#) (see also ArnBrowser Settings).

Parameters

in	<i>userName</i>	
in	<i>passwordHashed</i>	
in	<i>allow</i>	is the permissions for the server actions to this client.

See also

[loginToArn\(\)](#)

[Arn::Allow](#)
[loginRequired](#);

Definition at line 345 of file ArnClient.cpp.

14.7.4.25 passwordHash()

```
QString ArnClient::passwordHash (  
    const QString & password ) [static]
```

Generate a hashed password from clear text password.

Parameters

in	<i>password</i>	is the clear text password.
----	-----------------	-----------------------------

Returns

the hashed password, e.g "{A5ha62Aug}"

Definition at line 557 of file ArnClient.cpp.

14.7.4.26 receiveTimeout()

```
int ArnClient::receiveTimeout ( ) const
```

Get receive data timeout (base time)

Returns

the timeout in seconds

See also

[setReceiveTimeout\(\)](#)

Definition at line 506 of file ArnClient.cpp.

14.7.4.27 registerClient()

```
void ArnClient::registerClient (  
    const QString & id )
```

Register this client to be available with id.

When instantiating an [ArnClient](#), it's always registered as id = "std", if that's not taken by another client.

Any previous registration of id for this client will be released when using [registerClient\(\)](#).

Parameters

in	<i>id</i>	must not be "".
----	-----------	-----------------

See also

[getClient\(\)](#)
[id\(\)](#)

Definition at line 481 of file ArnClient.cpp.

14.7.4.28 remoteWhoIam()

```
Arn::XStringMap ArnClient::remoteWhoIam ( ) const
```

Returns remote side (server) readable identification information.

This is used to identify the server side in session.

Returns

the information.

See also

[setWhoIam\(\)](#)

Definition at line 579 of file ArnClient.cpp.

14.7.4.29 removeMountPoint()

```
bool ArnClient::removeMountPoint (
    const QString & localPath )
```

Remove a sharing tree path.

Only the mount point will be removed, i.e any new *Arn Data Objects* created within the *localPath* tree will not be shared with the server. However already existing objects will not be affected and is still shared with the server.

Parameters

in	<i>localPath</i>	is the sharing tree to be removed. Only affects newly created objects.
----	------------------	--

Return values

<i>false</i>	if error.
--------------	-----------

See also

[Sharing Arn Data Objects](#)

Definition at line 440 of file ArnClient.cpp.

14.7.4.30 `setAutoConnect()`

```
void ArnClient::setAutoConnect (
    bool isAuto,
    int retryTime = 2 )
```

Set automatic reconnect.

If `connectToArnList()` is used, this auto connect functionality starts every time after the last host in the `Arn` connection list has failed. The connection list is retried after `retryTime`. When using `connectToArn()`, there will be a `retryTime` delay between each reConnect to the host.

Parameters

in	<i>isAuto</i>	true if using auto reconnect
in	<i>retryTime</i>	is the time between attempts in seconds

Definition at line 472 of file ArnClient.cpp.

14.7.4.31 `setDemandLogin()`

```
void ArnClient::setDemandLogin (
    bool isDemandLogin )
```

Set clients demand for login.

If any of server or client demand login, it must be used.

Parameters

in	<i>isDemandLogin</i>	true if client demand login.
----	----------------------	------------------------------

See also

[isDemandLogin\(\)](#)

Definition at line 530 of file ArnClient.cpp.

14.7.4.32 setMountPoint()

```
bool ArnClient::setMountPoint (
    const QString & path )
```

Set the sharing tree path.

For compatibility, this can only set one mount point and with same local as remote path. If exactly one mount point exist, it will be removed before this new one is added.

Parameters

in	<i>path</i>	is the sharing tree.
----	-------------	----------------------

Return values

<i>false</i>	if error.
--------------	-----------

See also

[Sharing Arn Data Objects](#)

Deprecated Use [addMountPoint\(\)](#) and [removeMountPoint\(\)](#)

Definition at line 372 of file ArnClient.cpp.

14.7.4.33 setReceiveTimeout()

```
void ArnClient::setReceiveTimeout (
    int receiveTimeout )
```

Set receive data timeout (base time)

The timeout deals with no received data. This base time T is used as follows: time passed == T/2, send a dummy request to [ArnServer](#) time passed == T, signal status [ConnectStat::Stopped](#) time passed == 3*T, abort [ArnClient](#) tcp socket.

Default base time T is set to 10 seconds.

Parameters

in	<i>receiveTimeout</i>	is the base time T in seconds. 0 = off (no timeout).
----	-----------------------	--

See also

[receiveTimeout\(\)](#)

Note

Must be set before client is connected

Definition at line 514 of file ArnClient.cpp.

14.7.4.34 setSyncMode()

```
void ArnClient::setSyncMode (
    ArnClient::SyncMode syncMode )
```

Set ClientSyncMode.

Default for [ArnClient](#) is StdAutoMaster.

Parameters

in	<i>syncMode</i>	the ClientSyncMode to be set.
----	-----------------	-------------------------------

See also

[ClientSyncMode](#)
[syncMode\(\)](#)

Definition at line 546 of file ArnClient.cpp.

14.7.4.35 setWhoIam()

```
void ArnClient::setWhoIAM (
    const Arn::XStringMap & whoIAMXsm )
```

Set clients human readable identification information.

This is used to identify the client session. Standard keys to use are: Agent, UserName, Contact, Location.

Example usage

```
Arn::XStringMap wimXsm;
wimXsm.add("Agent", "Arn Browser");
wimXsm.add("UserName", "Arn Magnusson");
wimXsm.add("Contact", "arn@arnas.se");
wimXsm.add("Location", "The Longhouse");
_arnClient->setWhoIAM( wimXsm);
```

Parameters

in	<i>whoIAmXsm</i>	contains the information.
----	------------------	---------------------------

See also

[remoteWhoIAm\(\)](#)

Definition at line 571 of file ArnClient.cpp.

14.7.4.36 syncMode()

```
ArnClient::SyncMode ArnClient::syncMode ( ) const
```

Get ClientSyncMode.

Default for [ArnClient](#) is StdAutoMaster.

Return values

<i>ClientSyncMode.</i>

See also

[ClientSyncMode](#)
[setSyncMode\(\)](#)

Definition at line 538 of file ArnClient.cpp.

14.7.4.37 tcpConnected

```
void ArnClient::tcpConnected (
    const QString & arnHost,
    quint16 port ) [signal]
```

Signal emitted when the tcp connection is successfull.

Parameters

in	<i>arnHost</i>	is host name or ip address, e.g. "192.168.1.1".
in	<i>port</i>	is the host port, e.g. 2022.

14.7.4.38 tcpDisConnected

```
void ArnClient::tcpDisConnected ( ) [signal]
```

Signal emitted when the tcp connection is broken (has been successfull).

14.7.4.39 tcpError

```
void ArnClient::tcpError (
    const QString & errorText,
    QAbstractSocket::SocketError socketError ) [signal]
```

Signal emitted when a connection tcp error occur.

Parameters

in	<i>errorText</i>	is the human readable description of the error.
in	<i>socketError</i>	is the error from tcp socket, see Qt doc.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnClient.hpp \(4.0.0\)](#)
- [src/ArnClient.cpp \(4.0.0\)](#)

14.8 ArnClientConnectStat Class Reference

```
#include <ArnClient.hpp>
```

Public Types

- enum [E](#) { [Init](#) = 0, [Connecting](#), [Negotiating](#), [Connected](#), [Stopped](#), [Error](#), [Disconnected](#), [TriedAll](#) }
- enum [NS](#) { [NsEnum](#), [NsHuman](#) }

14.8.1 Detailed Description

Definition at line 49 of file ArnClient.hpp.

14.8.2 Member Enumeration Documentation

14.8.2.1 E

```
enum ArnClientConnectStat::E
```

Enumerator

Init	Initialized, not yet any result of trying to connect ...
Connecting	Trying to connect to an Arn host.
Negotiating	Negotiating terms and compatibility with an Arn host.
Connected	Successfully connected to an Arn host.
Stopped	No data flow within set timeout (still connected)
Error	Unsuccessfull when trying to connect to an Arn host.
Disconnected	TCP connection is broken (has been successfull)
TriedAll	Unsuccessfully tried to connect to all hosts in the Arn connection List.

Definition at line 53 of file ArnClient.hpp.

14.8.2.2 NS

```
enum ArnClientConnectStat::NS
```

Enumerator

NsEnum	
NsHuman	

Definition at line 73 of file ArnClient.hpp.

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

- [src/ArnInc/ArnClient.hpp \(4.0.0\)](#)

14.9 ArnClientReg Class Reference

Public Member Functions

- bool [store](#) ([ArnClient](#) *client, const QString &id)
- [ArnClient](#) * [get](#) (const QString &id)
- int [remove](#) (const QString &id)
- int [remove](#) (const [ArnClient](#) *client)

Static Public Member Functions

- static [ArnClientReg](#) & [instance](#) ()

14.9.1 Detailed Description

Definition at line 48 of file ArnClient.cpp.

14.9.2 Member Function Documentation

14.9.2.1 get()

```
ArnClient * ArnClientReg::get (
    const QString & id )
```

Definition at line 79 of file ArnClient.cpp.

14.9.2.2 instance()

```
ArnClientReg & ArnClientReg::instance ( ) [static]
```

Definition at line 114 of file ArnClient.cpp.

14.9.2.3 remove() [1/2]

```
int ArnClientReg::remove (
    const QString & id )
```

Definition at line 87 of file ArnClient.cpp.

14.9.2.4 remove() [2/2]

```
int ArnClientReg::remove (
    const ArnClient * client )
```

Definition at line 95 of file ArnClient.cpp.

14.9.2.5 store()

```
bool ArnClientReg::store (
    ArnClient * client,
    const QString & id )
```

Definition at line 66 of file ArnClient.cpp.

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

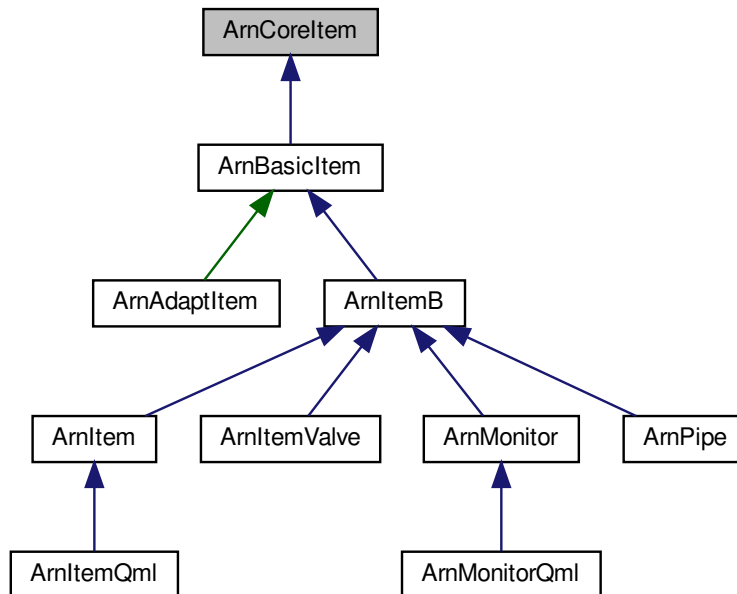
- [src/ArnClient.cpp \(4.0.0\)](#)

14.10 ArnCoreItem Class Reference

Core base class for the inherited [ArnItem](#) classes.

```
#include <ArnCoreItem.hpp>
```

Inheritance diagram for ArnCoreItem:



Classes

- struct [Heritage](#)

Public Member Functions

- [ArnCoreItem](#) ()
Standard constructor of a closed handle.
- virtual [~ArnCoreItem](#) ()
- `QThread * thread () const`
Get the thread affinity of this [ArnCoreItem](#).

Friends

- class [ArnBasicItemEventHandler](#)

14.10.1 Detailed Description

Core base class for the inherited [ArnItem](#) classes.

[About ArnItem access](#)

See [ArnItem](#).

[ArnCoreItem](#) is just a base class for [ArnBasicItem](#) and its inherited classes. Its purpose is to have a core API for meta handling ArnItems without having many virtual functions that increase the memory footprint for especially [ArnBasicItem](#).

It is the only real base class for all kinds of ArnItems.

Definition at line 56 of file ArnCoreItem.hpp.

14.10.2 Constructor & Destructor Documentation

14.10.2.1 ArnCoreItem()

```
ArnCoreItem::ArnCoreItem ( )
```

Standard constructor of a closed handle.

Definition at line 51 of file ArnCoreItem.cpp.

14.10.2.2 ~ArnCoreItem()

```
ArnCoreItem::~ArnCoreItem ( ) [virtual]
```

Definition at line 63 of file ArnCoreItem.cpp.

14.10.3 Member Function Documentation

14.10.3.1 thread()

```
QThread * ArnCoreItem::thread ( ) const
```

Get the thread affinity of this [ArnCoreItem](#).

The definition of affinity is different for different [ArnItem](#) classes. The returned value should still indicate for the caller if the item is in another thread and then the caller should treat the item with `isAlienThread=true`.

Returns

the thread affinity

See also

[setEventHandler\(\)](#)

Definition at line 69 of file ArnCoreItem.cpp.

14.10.4 Friends And Related Function Documentation

14.10.4.1 ArnBasicItemEventHandler

```
friend class ArnBasicItemEventHandler [friend]
```

Definition at line 59 of file ArnCoreItem.hpp.

The documentation for this class was generated from the following files:

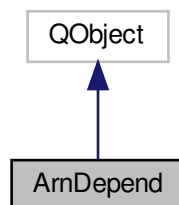
- [src/ArnInc/ArnCoreItem.hpp \(4.0.0\)](#)
- [src/ArnCoreItem.cpp \(4.0.0\)](#)

14.11 ArnDepend Class Reference

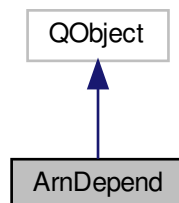
Class for setting up dependencis to needed services.

```
#include <ArnDepend.hpp>
```

Inheritance diagram for ArnDepend:



Collaboration diagram for ArnDepend:



Public Types

- typedef ArnDependSlot [DepSlot](#)

Signals

- void [completed](#) ()
Signal emitted when all dependent services are available.

Public Member Functions

- [ArnDepend](#) (QObject *parent=arnNullptr)
- [~ArnDepend](#) ()
- void [add](#) (const QString &serviceName, int stateId=-1)
Add a dependency for a service
- void [add](#) (const QString &serviceName, const QString &stateName)
Add a dependency for a service
- void [setMonitorName](#) (const QString &name)
Set an optional monitor name for debugging.
- void [startMonitor](#) ()
Starting the dependency monitor.

14.11.1 Detailed Description

Class for setting up dependencis to needed services.

The services can be both system types available by internal [Arn](#), and custom application types. The system types have a service name starting with "\$".

This is typically used when an application needs a service to continue. When using persistent values, a client will need to know when they have been synced from the server. Then it's convenient to setup a dependency for the system service "\$Persist".

When all dependent services are available, the [completed\(\)](#) signal is emitted.

Example usage

```
// In class declare
ArnDepend* _arnDepend;

// In class code
_arnDepend = new ArnDepend( this);
_arnDepend->setMonitorName("MyApp_Monitor"); // Optional for debug
_arnDepend->add("$Persist");
_arnDepend->add("MyService");
_arnDepend->startMonitor();
connect( _arnDepend, SIGNAL(completed()), this, SLOT(arnDependOk()));
```

Definition at line 132 of file ArnDepend.hpp.

14.11.2 Member Typedef Documentation

14.11.2.1 DepSlot

```
typedef ArnDependSlot ArnDepend::DepSlot
```

Definition at line 138 of file ArnDepend.hpp.

14.11.3 Constructor & Destructor Documentation

14.11.3.1 ArnDepend()

```
ArnDepend::ArnDepend (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 170 of file ArnDepend.cpp.

14.11.3.2 ~ArnDepend()

```
ArnDepend::~ArnDepend ( )
```

Definition at line 186 of file ArnDepend.cpp.

14.11.4 Member Function Documentation

14.11.4.1 add() [1/2]

```
void ArnDepend::add (
    const QString & serviceName,
    int stateId = -1 )
```

Add a dependency for a *service*

Parameters

in	<i>serviceName</i>	is the name of the needed <i>service</i> .
in	<i>stateId</i>	is the needed <i>state</i> id number. -1 is don't care.

Definition at line 221 of file ArnDepend.cpp.

14.11.4.2 add() [2/2]

```
void ArnDepend::add (
    const QString & serviceName,
    const QString & stateName )
```

Add a dependency for a *service*

Parameters

in	<i>serviceName</i>	is the name of the needed <i>service</i> .
in	<i>stateName</i>	is the needed <i>state</i> name.

Definition at line 213 of file ArnDepend.cpp.

14.11.4.3 completed

```
void ArnDepend::completed ( ) [signal]
```

Signal emitted when all dependent services are available.

14.11.4.4 setMonitorName()

```
void ArnDepend::setMonitorName (
    const QString & name )
```

Set an optional monitor name for debugging.

Parameters

in	<i>name</i>	is the monitor name.
----	-------------	----------------------

Definition at line 229 of file ArnDepend.cpp.

14.11.4.5 startMonitor()

```
void ArnDepend::startMonitor ( )
```

Starting the dependency monitor.

Definition at line 237 of file ArnDepend.cpp.

The documentation for this class was generated from the following files:

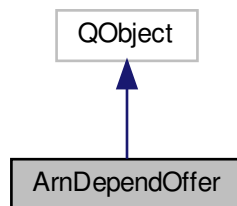
- [src/ArnInc/ArnDepend.hpp \(4.0.0\)](#)
- [src/ArnDepend.cpp \(4.0.0\)](#)

14.12 ArnDependOffer Class Reference

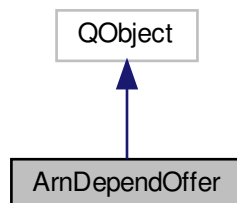
Class for advertising that a *service* is available.

```
#include <ArnDepend.hpp>
```

Inheritance diagram for ArnDependOffer:



Collaboration diagram for ArnDependOffer:



Public Member Functions

- [ArnDependOffer](#) (QObject *parent=arnNullptr)
- [~ArnDependOffer](#) ()
- void [advertise](#) (const QString &serviceName)
Advertise an available service
- void [setStateName](#) (const QString &name)
Set the state of the service by a logic name.
- QString [stateName](#) () const
- void [setStateId](#) (int id)
Set the state of the service by an id number.
- int [stateId](#) () const

14.12.1 Detailed Description

Class for advertising that a *service* is available.

Additionally it's possible to indicate the *state* of the *service*. The *state* can either be indicated by a logic name or by an id number whichever is preferred.

Example usage

```
// In class declare
ArnDependOffer* _depOffer;

// In class code
_depOffer = new ArnDependOffer( this);
_depOffer->advertise("MyService"); // Service now available
```

Definition at line 59 of file ArnDepend.hpp.

14.12.2 Constructor & Destructor Documentation

14.12.2.1 ArnDependOffer()

```
ArnDependOffer::ArnDependOffer (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 56 of file ArnDepend.cpp.

14.12.2.2 ~ArnDependOffer()

```
ArnDependOffer::~ArnDependOffer ( )
```

Definition at line 70 of file ArnDepend.cpp.

14.12.3 Member Function Documentation

14.12.3.1 advertise()

```
void ArnDependOffer::advertise (  
    const QString & serviceName )
```

Advertise an available *service*

Parameters

in	<i>serviceName</i>	is the name of the <i>service</i> .
----	--------------------	-------------------------------------

Definition at line 76 of file ArnDepend.cpp.

14.12.3.2 setStateId()

```
void ArnDependOffer::setStateId (
    int id )
```

Set the *state* of the *service* by an id number.

The *state* starts of by 0 as default.

Parameters

in	<i>id</i>	is the <i>state</i> id number.
----	-----------	--------------------------------

Definition at line 114 of file ArnDepend.cpp.

14.12.3.3 setStateName()

```
void ArnDependOffer::setStateName (
    const QString & name )
```

Set the *state* of the *service* by a logic name.

The *state* starts of by "Start" as default.

Parameters

in	<i>name</i>	is the <i>state</i> name.
----	-------------	---------------------------

Definition at line 98 of file ArnDepend.cpp.

14.12.3.4 stateId()

```
int ArnDependOffer::stateId ( ) const
```

Returns

The *state* id number.

See also

[setStateId\(\)](#)

Definition at line 122 of file ArnDepend.cpp.

14.12.3.5 stateName()

```
QString ArnDependOffer::stateName ( ) const
```

Returns

The logic *state* name, e.g. the default "Start"

See also

[setStateName\(\)](#)

Definition at line 106 of file ArnDepend.cpp.

The documentation for this class was generated from the following files:

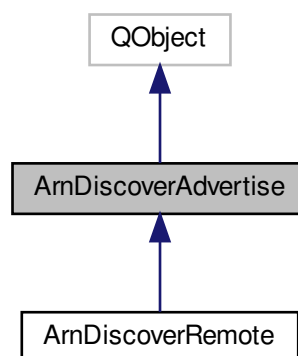
- src/ArnInc/ArnDepend.hpp (4.0.0)
- src/ArnDepend.cpp (4.0.0)

14.13 ArnDiscoverAdvertise Class Reference

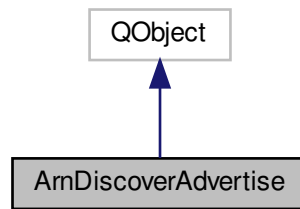
Advertise an [Arn](#) service.

```
#include <ArnDiscover.hpp>
```

Inheritance diagram for ArnDiscoverAdvertise:



Collaboration diagram for ArnDiscoverAdvertise:



Classes

- struct [State](#)

States of DiscoverAdvertise / These values must be synced with: [ArnZeroConf::State](#).

Public Slots

- virtual void [setService](#) (const QString &[service](#))

Set the service name.

Signals

- void [serviceChanged](#) (const QString &[serviceName](#))

Indicate successfull advertise of service.

- void [serviceChangeError](#) (int [code](#))

Indicate unsuccessful advertise of service.

Public Member Functions

- [ArnDiscoverAdvertise](#) (QObject *parent=arnNullptr)

- [~ArnDiscoverAdvertise](#) ()

- QStringList [groups](#) () const

Return service discover groups used for filter browsing.

- void [setGroups](#) (const QStringList &[groups](#))

Set service discover groups used for filter browsing.

- void [addGroup](#) (const QString &[group](#))

Add a service discover group.

- QString [service](#) () const

Returns the requested service name for this Advertise.

- QString [currentService](#) () const

Returns the current service name for this Advertise.

- [State state](#) () const

Returns the state for this Advertise.

- void [advertiseService](#) ([ArnDiscover::Type](#) discoverType, const QString &serviceName, int port=-1, const QString &hostName=QString())
Start advertising the service.
- [Arn::XStringMap customProperties](#) () const
Return service custom properties.
- void [setCustomProperties](#) (const [Arn::XStringMap](#) &customProperties)
Set service custom properties.
- void [addCustomProperty](#) (const QString &key, const QString &val)
Add service custom property.

14.13.1 Detailed Description

Advertise an [Arn](#) service.

About Arn Discover

Arn Discover is the mid level support for advertising services on an local network. For higher level support, use [ArnDiscoverRemote](#).

Example usage

```
// In class declare
ArnDiscoverAdvertise* _serviceAdvertiser;
ArnServer* _server;

// In class code
_server = new ArnServer( ArnServer::Type::NetSync, this);
_server->start(0); // Start server on dynamic port
int serverPort = _server->port();

_serviceAdvertiser = new ArnDiscoverAdvertise( this);
_serviceAdvertiser->addGroup("myId/myProduct");
_serviceAdvertiser->addCustomProperty("MyProtoVer", "1.0");
_serviceAdvertiser->advertiseService( ArnDiscover::Type::Server, "
    My service", serverPort);
```

Definition at line 629 of file ArnDiscover.hpp.

14.13.2 Constructor & Destructor Documentation

14.13.2.1 ArnDiscoverAdvertise()

```
ArnDiscoverAdvertise::ArnDiscoverAdvertise (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 831 of file ArnDiscover.cpp.

14.13.2.2 ~ArnDiscoverAdvertise()

```
ArnDiscoverAdvertise::~~ArnDiscoverAdvertise ( )
```

Definition at line 847 of file ArnDiscover.cpp.

14.13.3 Member Function Documentation

14.13.3.1 addCustomProperty()

```
void ArnDiscoverAdvertise::addCustomProperty (
    const QString & key,
    const QString & val )
```

Add service custom property.

The custom property are advised to have a *key* starting with a capital letter to avoid name collision with the system.

Parameters

in	<i>key</i>	property key (Start with capital letter) e.g. "MyProp"
in	<i>val</i>	property value can be any text e.g. "my data"

Note

Properties must be set before calling [advertiseService\(\)](#).

See also

[setCustomProperties\(\)](#)

Definition at line 932 of file ArnDiscover.cpp.

14.13.3.2 addGroup()

```
void ArnDiscoverAdvertise::addGroup (
    const QString & group )
```

Add a service discover group.

Parameters

in	<i>group</i>	e.g. "Any Group ID"
----	--------------	---------------------

Note

Groups must be set before calling [advertiseService\(\)](#).

See also

[setGroups\(\)](#)

Definition at line 1014 of file ArnDiscover.cpp.

14.13.3.3 advertiseService()

```
void ArnDiscoverAdvertise::advertiseService (
    ArnDiscover::Type discoverType,
    const QString & serviceName,
    int port = -1,
    const QString & hostName = QString() )
```

Start advertising the service.

Tries to advertise the service on the local network. Result is indicated by [serviceChanged\(\)](#) and [serviceChangeError\(\)](#) signals.

Empty *serviceName* will be ignored, no advertising until using [setService\(\)](#) with non empty name.

Parameters

in	<i>discoverType</i>	is used for discover filtering
in	<i>serviceName</i>	is requested name e.g. "My House Registry"
in	<i>port</i>	is the port of the service, -1 gives default Arn port number
in	<i>hostName</i>	is the host doing the service, empty gives this advertising host

See also

[setService\(\)](#)
[serviceChanged\(\)](#)
[serviceChangeError\(\)](#)

Definition at line 854 of file ArnDiscover.cpp.

14.13.3.4 currentService()

```
QString ArnDiscoverAdvertise::currentService ( ) const
```

Returns the current service name for this Advertise.

This is the really advertised name when it's available otherwise it's the requested service name.

Returns

service namen (se above) e.g. "My House Registry (2)"

See also

[setService\(\)](#)
[service\(\)](#)
[advertiseService\(\)](#)

Definition at line 964 of file ArnDiscover.cpp.

14.13.3.5 customProperties()

```
XStringMap ArnDiscoverAdvertise::customProperties ( ) const
```

Return service custom properties.

This is only the customer (application) properties, as there also are some [Arn](#) system properties.

Returns

custom properties

See also

[setCustomProperties\(\)](#)

Definition at line 916 of file ArnDiscover.cpp.

14.13.3.6 groups()

```
QStringList ArnDiscoverAdvertise::groups ( ) const
```

Return service discover groups used for filter browsing.

Returns

groups e.g. ("mydomain.se", "mydomain.se/House", "Any Group ID")

See also

[setGroups\(\)](#)

Definition at line 998 of file ArnDiscover.cpp.

14.13.3.7 service()

```
QString ArnDiscoverAdvertise::service ( ) const
```

Returns the requested service name for this Advertise.

This is always the requested service name, the really used name comes with the [serviceChanged\(\)](#) signal and [currentService\(\)](#).

Returns

requested service name, e.g. "My House Registry"

See also

[setService\(\)](#)
[currentService\(\)](#)
[advertiseService\(\)](#)

Definition at line 956 of file ArnDiscover.cpp.

14.13.3.8 serviceChanged

```
void ArnDiscoverAdvertise::serviceChanged (
    const QString & serviceName ) [signal]
```

Indicate successfull advertise of service.

Parameters

in	<i>serviceName</i>	is the really advertised name e.g. "My House Registry (2)"
----	--------------------	--

See also

[advertiseService\(\)](#)
[setService\(\)](#)

14.13.3.9 serviceChangeError

```
void ArnDiscoverAdvertise::serviceChangeError (
    int code ) [signal]
```

Indicate unsuccessfull advertise of service.

Parameters

in	<i>code</i>	error code.
----	-------------	-------------

See also

[advertiseService\(\)](#)

14.13.3.10 setCustomProperties()

```
void ArnDiscoverAdvertise::setCustomProperties (
    const Arn::XStringMap & customProperties )
```

Set service custom properties.

This is only the customer (application) properties, as there also are some [Arn](#) system properties.

These custom properties are advised to have a key starting with a capital letter to avoid name collision with the system.

Parameters

in	<i>customProperties</i>	e.g. Arn::XStringMap() .add("MyProp", "my data")
----	-------------------------	--

Note

Properties must be set before calling [advertiseService\(\)](#).

See also

[customProperties\(\)](#)
[addCustomProperty\(\)](#)
[ArnDiscoverInfo::properties\(\)](#)

Definition at line 924 of file ArnDiscover.cpp.

14.13.3.11 setGroups()

```
void ArnDiscoverAdvertise::setGroups (
    const QStringList & groups )
```

Set service discover groups used for filter browsing.

Groups are used for filtering discovered services. They will also be available as properties with naming as "group0", "group1" ...

Parameters

in	<i>groups</i>	e.g. ("mydomain.se", "mydomain.se/House", "Any Group ID")
----	---------------	---

Note

Groups must be set before calling [advertiseService\(\)](#).

See also

[groups\(\)](#)
[ArnDiscoverBrowser::setFilter\(\)](#)

Definition at line 1006 of file ArnDiscover.cpp.

14.13.3.12 setService

```
void ArnDiscoverAdvertise::setService (  
    const QString & service ) [virtual], [slot]
```

Set the service name.

Will update current advertised service name if this advertiser has been setup, otherwise the service name is stored for future use.

Service names can be any human readable id. It should be easy to understand, without any cryptic coding, and can usually be modified by the end user

Empty name is ignored. The requested service name is not guaranteed to be used for advertise, as it has to be unique within this local network. The really used name comes with the [serviceChanged\(\)](#) signal and [currentService\(\)](#).

Parameters

in	<i>service</i>	is the requested service name e.g. "My House Registry"
----	----------------	--

See also

[service\(\)](#)
[currentService\(\)](#)
[advertiseService\(\)](#)
[serviceChanged\(\)](#)
[serviceChangeError\(\)](#)

Definition at line 980 of file ArnDiscover.cpp.

14.13.3.13 state()

```
ArnDiscoverAdvertise::State ArnDiscoverAdvertise::state ( ) const
```

Returns the state for this Advertise.

Returns

current state

See also

[State](#)

Definition at line 972 of file ArnDiscover.cpp.

The documentation for this class was generated from the following files:

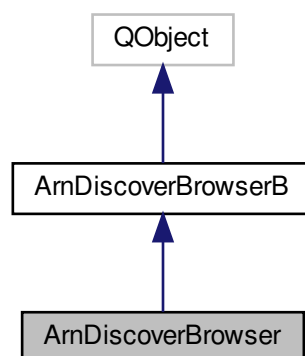
- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)
- [src/ArnDiscover.cpp \(4.0.0\)](#)

14.14 ArnDiscoverBrowser Class Reference

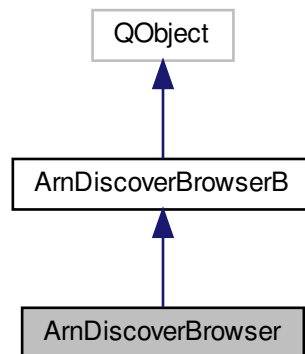
Browsing for [Arn](#) services.

```
#include <ArnDiscover.hpp>
```

Inheritance diagram for ArnDiscoverBrowser:



Collaboration diagram for ArnDiscoverBrowser:



Public Slots

- void [browse](#) (bool enable=true)
Change state of browsing.
- void [stopBrowse](#) ()
Stop browsing.

Public Member Functions

- [ArnDiscoverBrowser](#) (QObject *parent=arnNullptr)
- bool [isBrowsing](#) () const
Return the status of the browsing.
- void [setFilter](#) ([ArnDiscover::Type](#) typeFilter)
Set service discover filter using predefined types.
- void [setFilter](#) (const QString &group)
Set service discover filter using group name.

Additional Inherited Members

14.14.1 Detailed Description

Browsing for [Arn](#) services.

[About Arn Discover](#)

For a more complete example see the project ArnBrowser in DiscoverWindow.hpp and DiscoverWindow.cpp files.

Example usage

```

// In class declare
ArnDiscoverBrowser* _serviceBrowser;
QListWidget* _serviceTabView;
QLabel* _hostNameValue;

// In class code
_serviceBrowser = new ArnDiscoverBrowser( this);
connect(_serviceBrowser, SIGNAL(serviceAdded(int,QString)),
        this, SLOT(onServiceAdded(int,QString)));
connect(_serviceBrowser, SIGNAL(serviceRemoved(int)), this, SLOT(onServiceRemoved(int)));
connect(_serviceBrowser, SIGNAL(infoUpdated(int,
        ArnDiscoverInfo::State)),
        this, SLOT(onInfoUpdated(int,ArnDiscoverInfo::State)));

void XXX::onServiceAdded( int index, QString name)
{
    _serviceTabView->insertItem( index, name);
}

void XXX::onServiceRemoved( int index)
{
    QListWidgetItem* item = _serviceTabView->takeItem( index);
    if (item)
        delete item;
}

void XXX::onInfoUpdated( int index, ArnDiscoverInfo::State state)
{
    int curIndex = _serviceTabView->currentRow();
    if (index != curIndex) return; // The updated info is not for selected row

    const ArnDiscoverInfo& info = _serviceBrowser->infoByIndex( curIndex);
    _hostNameValue->setText( info.hostName());
}

```

Definition at line 477 of file ArnDiscover.hpp.

14.14.2 Constructor & Destructor Documentation

14.14.2.1 ArnDiscoverBrowser()

```

ArnDiscoverBrowser::ArnDiscoverBrowser (
    QObject * parent = arnNullptr ) [explicit]

```

Definition at line 237 of file ArnDiscover.cpp.

14.14.3 Member Function Documentation

14.14.3.1 browse

```

void ArnDiscoverBrowser::browse (
    bool enable = true ) [inline], [slot]

```

Change state of browsing.

When browsing is started, services will be discovered.

Parameters

in	<i>enable</i>	if true browsing is started, otherwise it is stopped
----	---------------	--

See also

[stopBrowse\(\)](#)
[serviceAdded\(\)](#)

Definition at line 516 of file ArnDiscover.hpp.

14.14.3.2 isBrowsing()

```
bool ArnDiscoverBrowser::isBrowsing ( ) const [inline]
```

Return the status of the browsing.

Return values

<i>true</i>	if browsing is started
-------------	------------------------

See also

[browse\(\)](#)

Definition at line 487 of file ArnDiscover.hpp.

14.14.3.3 setFilter() [1/2]

```
void ArnDiscoverBrowser::setFilter (
    ArnDiscover::Type typeFilter ) [inline]
```

Set service discover filter using predefined types.

When filter is enabled, only services that have the same type is discovered.

Parameters

in	<i>typeFilter</i>	
----	-------------------	--

See also

[ArnDiscoverAdvertise::advertiseService\(\)](#)

Definition at line 496 of file ArnDiscover.hpp.

14.14.3.4 setFilter() [2/2]

```
void ArnDiscoverBrowser::setFilter (
    const QString & group ) [inline]
```

Set service discover filter using group name.

If passing empty group, this is taken as subtype (filter) disabled. When subtype (filter) is enabled, only services that have the same group is discovered.

Parameters

in	<i>group</i>	the filter group name, e.g. "myGroup1"
----	--------------	--

See also

[ArnDiscoverAdvertise::setGroups\(\)](#)

Definition at line 506 of file ArnDiscover.hpp.

14.14.3.5 stopBrowse

```
void ArnDiscoverBrowser::stopBrowse ( ) [inline], [slot]
```

Stop browsing.

See also

[browse\(\)](#)

Definition at line 522 of file ArnDiscover.hpp.

The documentation for this class was generated from the following files:

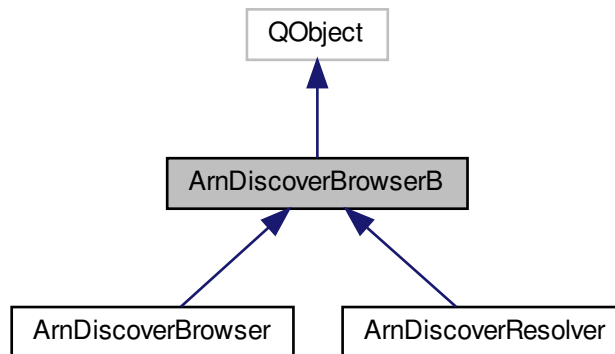
- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)
- [src/ArnDiscover.cpp \(4.0.0\)](#)

14.15 ArnDiscoverBrowserB Class Reference

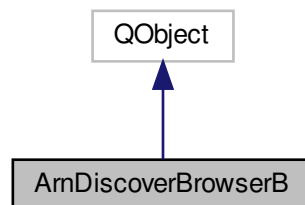
Browse() and resolve() together, may never be used to the same instance.

```
#include <ArnDiscover.hpp>
```

Inheritance diagram for ArnDiscoverBrowserB:



Collaboration diagram for ArnDiscoverBrowserB:



Signals

- void [serviceAdded](#) (int index, const QString &name)
Indicate service has been added (discovered)
- void [serviceRemoved](#) (int index)
Indicate service has been removed.
- void [infoUpdated](#) (int index, [ArnDiscoverInfo::State](#) state)
Indicate service has been updated.

Public Member Functions

- [ArnDiscoverBrowserB](#) (QObject *parent=arnNullptr)
- [~ArnDiscoverBrowserB](#) ()
- int [serviceCount](#) () const
Return the number of active discover services.
- const [ArnDiscoverInfo](#) & [infoByIndex](#) (int index)
Return the discover service info by its index.
- const [ArnDiscoverInfo](#) & [infoById](#) (int id)
Return the discover service info by its id.
- const [ArnDiscoverInfo](#) & [infoByName](#) (const QString &serviceName)
Return the discover service info by its name.
- int [indexTold](#) (int index)
Return the discover service id by its index.
- int [IdToIndex](#) (int id)
Return the discover service index by its id.
- int [serviceNameTold](#) (const QString &name)
Return the discover service id by its name.
- [ArnDiscoverInfo::State defaultStopState](#) () const
Return the default stop state for this service discover browser.
- void [setDefaultStopState](#) ([ArnDiscoverInfo::State defaultStopState](#))
Set the default stop state for this service discover browser.
- bool [goTowardState](#) (int index, [ArnDiscoverInfo::State](#) state)
Command a service to go towards a stop state.

14.15.1 Detailed Description

Browse() and resolve() together, may never be used to the same instance.

Definition at line 224 of file ArnDiscover.hpp.

14.15.2 Constructor & Destructor Documentation

14.15.2.1 ArnDiscoverBrowserB()

```
ArnDiscoverBrowserB::ArnDiscoverBrowserB (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 346 of file ArnDiscover.cpp.

14.15.2.2 ~ArnDiscoverBrowserB()

```
ArnDiscoverBrowserB::~ArnDiscoverBrowserB ( )
```

Definition at line 354 of file ArnDiscover.cpp.

14.15.3 Member Function Documentation

14.15.3.1 defaultStopState()

```
ArnDiscoverInfo::State ArnDiscoverBrowserB::defaultStopState ( ) const
```

Return the default stop state for this service discover browser.

This default stop state will be used for all services discovered by this browser.

Returns

default stop state

See also

[setDefaultStopState\(\)](#)
[goTowardState\(\)](#)
[ArnDiscoverInfo::stopState\(\)](#)
State

Definition at line 463 of file ArnDiscover.cpp.

14.15.3.2 goTowardState()

```
bool ArnDiscoverBrowserB::goTowardState (
    int index,
    ArnDiscoverInfo::State state )
```

Command a service to go towards a stop state.

The service is specified by its index. The wanted final state must be forward, otherwise it is ignored.

Parameters

in	<i>index</i>	for the service
in	<i>state</i>	is the wanted final state

See also

[defaultStopState\(\)](#)
[infoUpdated\(\)](#)
[ArnDiscoverInfo::stopState\(\)](#)
State

Definition at line 479 of file ArnDiscover.cpp.

14.15.3.3 IdToIndex()

```
int ArnDiscoverBrowserB::IdToIndex (
    int id )
```

Return the discover service index by its id.

The index for a service info is only valid valid for a given moment, it can change as services are added and removed. If given a non existent id, -1 will be returned.

Parameters

in	<i>id</i>	
----	-----------	--

Returns

selected service discover index

See also

[indexTold\(\)](#)
[infoByIndex\(\)](#)

Definition at line 413 of file ArnDiscover.cpp.

14.15.3.4 indexTold()

```
int ArnDiscoverBrowserB::indexToId (
    int index )
```

Return the discover service id by its index.

The index for a service info is only valid valid for a given moment, it can change as services are added and removed. If given an invalid index, -1 will be returned.

Parameters

in	<i>index</i>	
----	--------------	--

Returns

selected service discover id

See also

[IdToIndex\(\)](#)
[infoById\(\)](#)

Definition at line 403 of file ArnDiscover.cpp.

14.15.3.5 infoById()

```
const ArnDiscoverInfo & ArnDiscoverBrowserB::infoById (
    int id )
```

Return the discover service info by its id.

The id for a service info is unique and stays same over time, but the service can have been removed. If given a non existent service id, a Null discover info will be returned.

Parameters

in	<i>id</i>	
----	-----------	--

Returns

selected service discover info

See also

[infoByIndex\(\)](#)

Definition at line 388 of file ArnDiscover.cpp.

14.15.3.6 infoByIndex()

```
const ArnDiscoverInfo & ArnDiscoverBrowserB::infoByIndex (
    int index )
```

Return the discover service info by its index.

The index for a service info is only valid valid for a given moment, it can change as services are added and removed. If given an invalid index, a Null discover info will be returned.

Parameters

in	<i>index</i>	
----	--------------	--

Returns

selected service discover info

See also

[infoById\(\)](#)
[infoByName\(\)](#)
[indexTold\(\)](#)

Definition at line 376 of file ArnDiscover.cpp.

14.15.3.7 infoByName()

```
const ArnDiscoverInfo & ArnDiscoverBrowserB::infoByName (
    const QString & serviceName )
```

Return the discover service info by its name.

The service name is unique for a given moment, but the service can be removed and then reappear with a different service name. Also non used service names can be reused for a different service. If given a non existent service name, a Null discover info will be returned.

Parameters

in	<i>serviceName</i>	
----	--------------------	--

Returns

selected service discover info

See also

[serviceNameTold\(\)](#)

Definition at line 397 of file ArnDiscover.cpp.

14.15.3.8 infoUpdated

```
void ArnDiscoverBrowserB::infoUpdated (
    int index,
    ArnDiscoverInfo::State state ) [signal]
```

Indicate service has been updated.

Parameters

in	<i>index</i>	for the service
in	<i>state</i>	is the current state of the service info

See also

[goTowardState\(\)](#)
[serviceAdded\(\)](#)

14.15.3.9 serviceAdded

```
void ArnDiscoverBrowserB::serviceAdded (
    int index,
    const QString & name ) [signal]
```

Indicate service has been added (discovered)

The service has been added to a list sorted by ascending service names. The index is a reference to this sorted list.

Parameters

in	<i>index</i>	for the service
in	<i>name</i>	is the service name e.g. "My House Registry"

See also

[serviceRemoved\(\)](#)

[infoUpdated\(\)](#)

14.15.3.10 serviceCount()

```
int ArnDiscoverBrowserB::serviceCount ( ) const
```

Return the number of active discover services.

Returns

number of services

Definition at line 368 of file ArnDiscover.cpp.

14.15.3.11 serviceNameToId()

```
int ArnDiscoverBrowserB::serviceNameToId (
    const QString & name )
```

Return the discover service id by its name.

The service name is unique for a given moment. If given a non existent service name, -1 will be returned.

Parameters

in	<i>name</i>	
----	-------------	--

Returns

selected service discover id

See also

[IdToIndex\(\)](#)
[infoByName\(\)](#)

Definition at line 421 of file ArnDiscover.cpp.

14.15.3.12 serviceRemoved

```
void ArnDiscoverBrowserB::serviceRemoved (
    int index ) [signal]
```

Indicate service has been removed.

Parameters

in	<i>index</i>	for the service
----	--------------	-----------------

See also

[serviceAdded\(\)](#)

14.15.3.13 setDefaultStopState()

```
void ArnDiscoverBrowserB::setDefaultStopState (
    ArnDiscoverInfo::State defaultStopState )
```

Set the default stop state for this service discover browser.

This default stop state will be used for all services discovered by this browser.

Parameters

in	<i>defaultStopState</i>	
----	-------------------------	--

See also

[defaultStopState\(\)](#)
[goTowardState\(\)](#)
[ArnDiscoverInfo::stopState\(\)](#)
State

Definition at line 471 of file ArnDiscover.cpp.

The documentation for this class was generated from the following files:

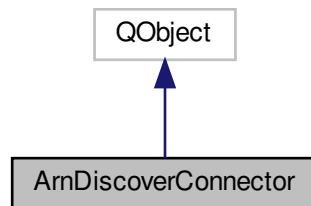
- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)
- [src/ArnDiscover.cpp \(4.0.0\)](#)

14.16 ArnDiscoverConnector Class Reference

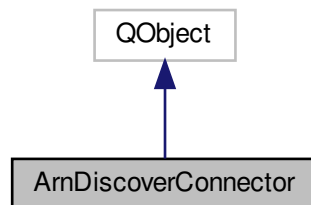
An automatic client discover connector.

```
#include <ArnDiscoverConnect.hpp>
```

Inheritance diagram for ArnDiscoverConnector:



Collaboration diagram for ArnDiscoverConnector:



Public Slots

- void [setService](#) (const QString &service)
Set the service name for the connection.

Signals

- void [clientReadyToConnect](#) (ArnClient *arnClient, const QString &id)
Signal for external client connection.

Public Member Functions

- [ArnDiscoverConnector](#) ([ArnClient](#) &client, const [QString](#) &id)
- [~ArnDiscoverConnector](#) ()
- void [clearDirectHosts](#) ()
 - Clear the direct host connection list.*
- void [addToDirectHosts](#) (const [QString](#) &arnHost, quint16 port=0)
 - Add an [Arn](#) Server to the direct host connection list.*
- void [setResolver](#) ([ArnDiscoverResolver](#) *resolver)
 - Set the [ArnDiscoverResolver](#) to be used.*
- void [start](#) ()
 - Start connector.*
- [QString](#) [id](#) () const
 - Return the identifier for this connector.*
- [QString](#) [service](#) () const
 - Returns the service name for this connection.*
- int [directHostPrio](#) () const
 - Return the priority for direct hosts*
- void [setDirectHostPrio](#) (int [directHostPrio](#))
 - Set the priority for direct hosts*
- int [discoverHostPrio](#) () const
 - Return the priority for discovered hosts*
- void [setDiscoverHostPrio](#) (int [discoverHostPrio](#))
 - Set the priority for discovered hosts*
- int [resolveRefreshTimeout](#) () const
 - Return the resolv refresh period.*
- void [setResolveRefreshTimeout](#) (int [resolveRefreshTimeout](#))
 - Set the resolv refresh period.*
- bool [externalClientConnect](#) () const
 - Return the external client connect mode.*
- void [setExternalClientConnect](#) (bool [externalClientConnect](#))
 - Set the external client connect mode.*

14.16.1 Detailed Description

An automatic client discover connector.

[About Arn Discover Remote](#)

This connector class manages client connections. Both as a list of possible *direct host* addresses and using a service name for resolving into a *discover host*. The two methods can coexist and as standard the *discover host* has lowest priority number, i.e. tried first.

An *id* is assigned to every connector. The *id* should be chosen to describe the client target or its purpose. It's not a host address or necessarily a specific host, as there can be many possible addresses assigned to the [ArnDiscoverConnector](#).

The *id* will appear as an *Arn folder*, e.g. when *id* is "WeatherData-XYZ" the *connector folder path* will be "Sys/←→Discover/Connect/WeatherData-XYZ!".

Example usage

```
// In class declare
ArnDiscoverConnector* _connector
ArnClient _arnClient;

// In class code
_arnClient.addMountPoint("/");
_arnClient.setAutoConnect(true);

_connector = new ArnDiscoverConnector( _arnClient, "MyConnectionId");
_connector->setResolver( new ArnDiscoverResolver());
_connector->setService("My Service");
_connector->addToDirectHosts("localhost");
_connector->start();
```

Examples:

[ArnDemoChat/MainWindow.cpp](#).

Definition at line 75 of file ArnDiscoverConnect.hpp.

14.16.2 Constructor & Destructor Documentation

14.16.2.1 ArnDiscoverConnector()

```
ArnDiscoverConnector::ArnDiscoverConnector (
    ArnClient & client,
    const QString & id )
```

Definition at line 70 of file ArnDiscoverConnect.cpp.

14.16.2.2 ~ArnDiscoverConnector()

```
ArnDiscoverConnector::~ArnDiscoverConnector ( )
```

Definition at line 93 of file ArnDiscoverConnect.cpp.

14.16.3 Member Function Documentation

14.16.3.1 addToDirectHosts()

```
void ArnDiscoverConnector::addToDirectHosts (
    const QString & arnHost,
    quint16 port = 0 )
```

Add an [Arn Server](#) to the *direct host* connection list.

Parameters

in	<i>arnHost</i>	is host name or ip address, e.g. "192.168.1.1".
in	<i>port</i>	is the host port, 0 gives Arn::defaultTcpPort .

See also

[clearDirectHosts\(\)](#)
[ArnClient](#)

Definition at line 107 of file ArnDiscoverConnect.cpp.

14.16.3.2 clearDirectHosts()

```
void ArnDiscoverConnector::clearDirectHosts ( )
```

Clear the *direct host* connection list.

Typically used to start making a new connection list.

See also

[addToDirectHosts\(\)](#)
[ArnClient](#)

Definition at line 99 of file ArnDiscoverConnect.cpp.

14.16.3.3 clientReadyToConnect

```
void ArnDiscoverConnector::clientReadyToConnect (
    ArnClient * arnClient,
    const QString & id ) [signal]
```

Signal for external client connection.

When activated external client connection by the method [setExternalClientConnect\(\)](#), this signal will be emitted when the client has been prepared to connect.

It's the responsibility of the receiver to do the actual client connect by [ArnClient::connectToArnList\(\)](#).

Parameters

in	<i>arnClient</i>	being ready for connection
in	<i>id</i>	is the identifier used in ArnDiscoverRemote::newConnector() , e.g "WeatherData-XYZ"

See also

[ArnDiscoverRemote::newConnector\(\)](#)
[setExternalClientConnect\(\)](#)

14.16.3.4 directHostPrio()

```
int ArnDiscoverConnector::directHostPrio ( ) const
```

Return the priority for *direct hosts*

Returns

direct host priority

See also

[setDirectHostPrio\(\)](#)

Definition at line 177 of file ArnDiscoverConnect.cpp.

14.16.3.5 discoverHostPrio()

```
int ArnDiscoverConnector::discoverHostPrio ( ) const
```

Return the priority for *discovered hosts*

Returns

discoverHostPrio is the priority.

See also

[setDiscoverHostPrio\(\)](#)

Definition at line 161 of file ArnDiscoverConnect.cpp.

14.16.3.6 externalClientConnect()

```
bool ArnDiscoverConnector::externalClientConnect ( ) const
```

Return the *external client connect* mode.

Returns

true when active.

See also

[setExternalClientConnect\(\)](#)

Definition at line 193 of file ArnDiscoverConnect.cpp.

14.16.3.7 id()

```
QString ArnDiscoverConnector::id ( ) const
```

Return the identifier for this connector.

Returns

the identifier, e.g "WeatherData-XYZ"

See also

[ArnDiscoverRemote::newConnector\(\)](#)

Definition at line 137 of file ArnDiscoverConnect.cpp.

14.16.3.8 resolveRefreshTimeout()

```
int ArnDiscoverConnector::resolveRefreshTimeout ( ) const
```

Return the resolv refresh period.

Returns

resolve refresh timeout in seconds.

See also

[setResolveRefreshTimeout\(\)](#)

Definition at line 145 of file ArnDiscoverConnect.cpp.

14.16.3.9 service()

```
QString ArnDiscoverConnector::service ( ) const
```

Returns the service name for this connection.

Returns

service name, e.g. "My House Registry"

See also

[setService\(\)](#)

Definition at line 209 of file ArnDiscoverConnect.cpp.

14.16.3.10 setDirectHostPrio()

```
void ArnDiscoverConnector::setDirectHostPrio (
    int directHostPrio )
```

Set the priority for *direct hosts*

This priority controls order between *direct hosts* and *discover host*. Low priority number give earlier try for its hosts.

Parameters

in	<i>directHostPrio</i>	is the priority.
----	-----------------------	------------------

Note

The priority for *direct hosts* and *discover hosts* must be different.

See also

[directHostPrio\(\)](#)

Definition at line 185 of file ArnDiscoverConnect.cpp.

14.16.3.11 setDiscoverHostPrio()

```
void ArnDiscoverConnector::setDiscoverHostPrio (
    int discoverHostPrio )
```

Set the priority for *discovered hosts*

This priority controls order between *direct hosts* and *discover host*. Low priority number give earlier try for its hosts.

Parameters

in	<i>discoverHostPrio</i>	is the priority.
----	-------------------------	------------------

Note

The priority for *direct hosts* and *discover hosts* must be different.

See also

[discoverHostPrio\(\)](#)

Definition at line 169 of file ArnDiscoverConnect.cpp.

14.16.3.12 setExternalClientConnect()

```
void ArnDiscoverConnector::setExternalClientConnect (
    bool externalClientConnect )
```

Set the *external client connect* mode.

This mode is used when there is a need to do special processing when connecting a client. Then `QObject::connect()` should be used for the signal [clientReadyToConnect\(\)](#) and a *receiver* doing the special processing.

It's the responsibility of the *receiver* to do the actual client connect by [ArnClient::connectToArnList\(\)](#).

Parameters

in	<i>externalClientConnect</i>	true to activate.
----	------------------------------	-------------------

See also

[externalClientConnect\(\)](#)

Definition at line 201 of file ArnDiscoverConnect.cpp.

14.16.3.13 setResolver()

```
void ArnDiscoverConnector::setResolver (
    ArnDiscoverResolver * resolver )
```

Set the [ArnDiscoverResolver](#) to be used.

The resolver handles resolving a known service name into a host name.

Ownership is taken of this resolver. Any previous set resolver will be deleted.

Parameters

in	<i>resolver</i>	is the used ArnDiscoverResolver . Use 0 (null) to set none.
----	-----------------	---

Examples:

[ArnDemoChat/MainWindow.cpp](#).

Definition at line 115 of file ArnDiscoverConnect.cpp.

14.16.3.14 setResolveRefreshTimeout()

```
void ArnDiscoverConnector::setResolveRefreshTimeout (
    int resolveRefreshTimeout )
```

Set the resolv refresh period.

The refresh period is used when there is a failure to connect to a *discover host*.

The rationale is that the current resolv might be outdated as there is an error when connecting to the resolved host. A refreshed resolv will be done at an interval of *resolveRefreshTimeout* until connection to resolved host is successful.

Parameters

in	<i>resolveRefreshTimeout</i>	is the period in seconds.
----	------------------------------	---------------------------

See also

[resolveRefreshTimeout\(\)](#)

Definition at line 153 of file ArnDiscoverConnect.cpp.

14.16.3.15 setService

```
void ArnDiscoverConnector::setService (
    const QString & service ) [slot]
```

Set the service name for the connection.

This is only functional if using [ArnDiscoverResolver](#), see [setResolver\(\)](#).

Will update connection service name if the resolver has been setup, otherwise the service name is only stored for future use.

For remote control the service name is also available as an *Arn Data Object* at **local path**: *connector folder path* + "Service/value", e.g. "Sys/Discover/Connect/WeatherData-XYZ/Service/value".

Parameters

in	<i>service</i>	is the requested connection service name e.g. "My House Registry"
----	----------------	---

See also

[ArnDiscoverAdvertise::setService\(\)](#)

Examples:

[ArnDemoChat/MainWindow.cpp](#).

Definition at line 217 of file ArnDiscoverConnect.cpp.

14.16.3.16 start()

```
void ArnDiscoverConnector::start ( )
```

Start connector.

See also

[addToDirectHosts\(\)](#)
[setResolver\(\)](#)

Examples:

[ArnDemoChat/MainWindow.cpp](#).

Definition at line 228 of file ArnDiscoverConnect.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnDiscoverConnect.hpp \(4.0.0\)](#)
- [src/ArnDiscoverConnect.cpp \(4.0.0\)](#)

14.17 ArnDiscoverInfo Class Reference

Class for holding current discover info of one service.

```
#include <ArnDiscover.hpp>
```

Classes

- struct [State](#)
State of Arn discover browse data. Can be tested by relative order.

Public Member Functions

- [ArnDiscoverInfo](#) ()
- [ArnDiscoverInfo](#) (const [ArnDiscoverInfo](#) &other)
- [ArnDiscoverInfo](#) & [operator=](#) (const [ArnDiscoverInfo](#) &other)
- [~ArnDiscoverInfo](#) ()
- bool [inProgress](#) () const
Is discover in progress for this service.
- bool [isError](#) () const
Is in an error state for this service.
- [State](#) [state](#) () const
Return the state for this service.
- [State](#) [stopState](#) () const
Return the stop state for this service.
- [ArnDiscover::Type](#) [type](#) () const
Return the discover type for this service.
- [QStringList](#) [groups](#) () const
Return the groups for this service.
- [QString](#) [serviceName](#) () const
Return the service name for this service.
- [QString](#) [domain](#) () const
Return the domain for this service.
- [QString](#) [hostName](#) () const
Return the host name for this service.
- [quint16](#) [hostPort](#) () const
Return the port for this service.
- [QHostAddress](#) [hostIp](#) () const
Return the host ip-address for this service.
- [Arn::XStringMap](#) [properties](#) () const
Return the properties for this service.
- [QString](#) [typeString](#) () const
Return the printable type for this service.
- [QString](#) [hostPortString](#) () const
Return the printable host port for this service.
- [QString](#) [hostIpString](#) () const
Return the printable host ip-address for this service.
- [QString](#) [hostWithInfo](#) () const
Get the the HostWithInfo string.
- int [resolvCode](#) () const
Return the latest resolv error code for this service.

Friends

- class [ArnDiscoverBrowserB](#)

14.17.1 Detailed Description

Class for holding current discover info of one service.

[About Arn Discover](#)

This class holds the service info and its discover state.

Definition at line 72 of file [ArnDiscover.hpp](#).

14.17.2 Constructor & Destructor Documentation

14.17.2.1 ArnDiscoverInfo() [1/2]

```
ArnDiscoverInfo::ArnDiscoverInfo ( )
```

Definition at line 60 of file ArnDiscover.cpp.

14.17.2.2 ArnDiscoverInfo() [2/2]

```
ArnDiscoverInfo::ArnDiscoverInfo (
    const ArnDiscoverInfo & other )
```

Definition at line 72 of file ArnDiscover.cpp.

14.17.2.3 ~ArnDiscoverInfo()

```
ArnDiscoverInfo::~ArnDiscoverInfo ( )
```

Definition at line 86 of file ArnDiscover.cpp.

14.17.3 Member Function Documentation

14.17.3.1 domain()

```
QString ArnDiscoverInfo::domain ( ) const
```

Return the domain for this service.

Returns

domain, e.g. "local."

Definition at line 149 of file ArnDiscover.cpp.

14.17.3.2 groups()

```
QStringList ArnDiscoverInfo::groups ( ) const
```

Return the groups for this service.

Groups are used for filtering discovered services. They will also be available as properties with naming as "group0", "group1" ...

Returns

groups, e.g. ("mydomain.se", "mydomain.se/House", "Any Group ID")

See also

[ArnDiscoverAdvertise::setGroups\(\)](#)

Definition at line 133 of file ArnDiscover.cpp.

14.17.3.3 hostIp()

```
QHostAddress ArnDiscoverInfo::hostIp ( ) const
```

Return the host ip-address for this service.

Returns

host ip-address

Definition at line 173 of file ArnDiscover.cpp.

14.17.3.4 hostIpString()

```
QString ArnDiscoverInfo::hostIpString ( ) const
```

Return the printable host ip-address for this service.

Will return empty string if no valid ip available

Returns

host ip-address, e.g. "192.168.1.1", "" etc

Definition at line 213 of file ArnDiscover.cpp.

14.17.3.5 hostName()

```
QString ArnDiscoverInfo::hostName ( ) const
```

Return the host name for this service.

Returns

host name, e.g. "myHost.local"

See also

[ArnDiscoverAdvertise::advertiseService\(\)](#)

Definition at line 157 of file ArnDiscover.cpp.

14.17.3.6 hostPort()

```
quint16 ArnDiscoverInfo::hostPort ( ) const
```

Return the port for this service.

Returns

port

See also

[ArnDiscoverAdvertise::advertiseService\(\)](#)

Definition at line 165 of file ArnDiscover.cpp.

14.17.3.7 hostPortString()

```
QString ArnDiscoverInfo::hostPortString ( ) const
```

Return the printable host port for this service.

Will return empty string if no valid port available

Returns

host port, e.g. "2022", "" etc

Definition at line 205 of file ArnDiscover.cpp.

14.17.3.8 hostWithInfo()

```
QString ArnDiscoverInfo::hostWithInfo ( ) const
```

Get the the *HostWithInfo* string.

[ArnClient](#) and alike accepts such *HostWithInfo* strings for connection.

Returns

The *HostWithInfo* string, e.g. "192.168.1.1 [myhost.local]"

See also

[Arn::makeHostWithInfo\(\)](#)

Definition at line 221 of file ArnDiscover.cpp.

14.17.3.9 inProgress()

```
bool ArnDiscoverInfo::inProgress ( ) const
```

Is discover in progress for this service.

Return values

<i>true</i>	if discover is in progress
-------------	----------------------------

See also

[state\(\)](#)

Definition at line 93 of file ArnDiscover.cpp.

14.17.3.10 isError()

```
bool ArnDiscoverInfo::isError ( ) const
```

Is in an error state for this service.

Return values

<i>true</i>	if in error state
-------------	-------------------

See also[state\(\)](#)

Definition at line 101 of file ArnDiscover.cpp.

14.17.3.11 operator=()

```
ArnDiscoverInfo & ArnDiscoverInfo::operator= (
    const ArnDiscoverInfo & other )
```

Definition at line 78 of file ArnDiscover.cpp.

14.17.3.12 properties()

```
XStringMap ArnDiscoverInfo::properties ( ) const
```

Return the properties for this service.

Will return both [Arn](#) system properties and custom (application) properties. System properties will always have a key starting with a lower case letter e.g. "protovers".

Returns

properties

See also[ArnDiscoverAdvertise::setCustomProperties\(\)](#)

Definition at line 181 of file ArnDiscover.cpp.

14.17.3.13 resolvCode()

```
int ArnDiscoverInfo::resolvCode ( ) const
```

Return the latest resolv error code for this service.

This code can come from both resolving a service and lookup ip-address.

Returns

error code

See also[ArnZeroConf::Error](#)

Definition at line 227 of file ArnDiscover.cpp.

14.17.3.14 serviceName()

```
QString ArnDiscoverInfo::serviceName ( ) const
```

Return the service name for this service.

Returns

service name, e.g. "My House Registry"

See also

[ArnDiscoverAdvertise::advertiseService\(\)](#)
[ArnDiscoverAdvertise::setService\(\)](#)

Definition at line 141 of file ArnDiscover.cpp.

14.17.3.15 state()

```
ArnDiscoverInfo::State ArnDiscoverInfo::state ( ) const
```

Return the state for this service.

Returns

state

See also

[State](#)

Definition at line 109 of file ArnDiscover.cpp.

14.17.3.16 stopState()

```
ArnDiscoverInfo::State ArnDiscoverInfo::stopState ( ) const
```

Return the stop state for this service.

The discover logic will stop when reaching the stop state for a service.

Returns

stop state

See also

[ArnDiscoverBrowserB::setDefaultStopState\(\)](#)
[ArnDiscoverBrowserB::goTowardState\(\)](#)
[State](#)

Definition at line 117 of file ArnDiscover.cpp.

14.17.3.17 type()

```
ArnDiscover::Type ArnDiscoverInfo::type ( ) const
```

Return the discover type for this service.

Returns

discover type

See also

Type
[ArnDiscoverAdvertise::advertiseService\(\)](#)

Definition at line 125 of file ArnDiscover.cpp.

14.17.3.18 typeString()

```
QString ArnDiscoverInfo::typeString ( ) const
```

Return the printable type for this service.

Returns

type, e.g. "Client"

Definition at line 189 of file ArnDiscover.cpp.

14.17.4 Friends And Related Function Documentation

14.17.4.1 ArnDiscoverBrowserB

```
friend class ArnDiscoverBrowserB [friend]
```

Definition at line 75 of file ArnDiscover.hpp.

The documentation for this class was generated from the following files:

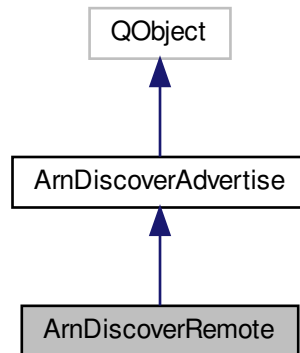
- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)
- [src/ArnDiscover.cpp \(4.0.0\)](#)

14.18 ArnDiscoverRemote Class Reference

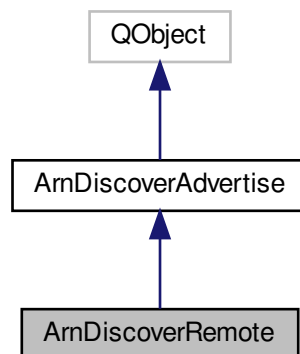
Discover with remote setting.

```
#include <ArnDiscoverRemote.hpp>
```

Inheritance diagram for ArnDiscoverRemote:



Collaboration diagram for ArnDiscoverRemote:



Public Slots

- virtual void `setService` (const QString &service)
Set the service name.

Signals

- void [clientReadyToConnect](#) ([ArnClient](#) *arnClient, const QString &id)
Central signal for external client connection.

Public Member Functions

- [ArnDiscoverRemote](#) (QObject *parent=arnNullptr)
- [~ArnDiscoverRemote](#) ()
- QString [defaultService](#) () const
Return the default service name.
- void [setDefaultService](#) (const QString &defaultService)
Set the default service name.
- int [initialServiceTimeout](#) () const
Return the time for initial timeout processing.
- void [setInitialServiceTimeout](#) (int initialServiceTimeout)
Set the time for initial timeout processing.
- void [startUseServer](#) ([ArnServer](#) *arnServer, [ArnDiscover::Type](#) discoverType=[ArnDiscover::Type::Server](#))
Start advertising the [ArnServer](#) as a service.
- void [startUseNewServer](#) ([ArnDiscover::Type](#) discoverType, int port=-1)
Start a new [ArnServer](#) and advertise as a service.
- [ArnDiscoverConnector](#) * [newConnector](#) ([ArnClient](#) &client, const QString &id)
Create and return an [ArnDiscoverConnector](#) for handling remote client.

14.18.1 Detailed Description

Discover with remote setting.

About Arn Discover Remote

This class is the main class for handling discover with remote setting.

Following rules apply:

- If service is set before start using server, this service will be used.
- If no persist is active or it gives an empty service name, timeout-processing is done.
- Timeout-processing can wait upto [initialServiceTimeout\(\)](#), after that [defaultService\(\)](#) will be used as service.
- If service is set by any method before timeout-processing has finished, that service is used. Timeout-processing is then also aborted.
- After initial advertise of the service, it can be changed by any method and the changed service will be used.
- The used service will also be saved if using persist.
- Methods to change service are [ArnDiscoverRemote::setService\(\)](#) and corresponding *Arn Data Objects* which can be changed locally or remote.

For a complete example of advertising a server, see the project [ArnServer](#) in ServerMain.hpp and ServerMain.cpp files.

Example usage

```

// In class declare
ArnDiscoverRemote* _discoverRemote;
ArnClient* _client;

// In class code
_client = new ArnClient;
_client->addMountPoint("/");
_client->setAutoConnect( true);

_discoverRemote = new ArnDiscoverRemote( this);
_discoverRemote->setDefaultService("My default service");
_discoverRemote->addGroup("myId/myProduct");
_discoverRemote->addCustomProperty("MyProtoVer", "1.0");
_discoverRemote->startUseNewServer( ArnDiscover::Type::Client, 0)
; // Dynamic server

ArnDiscoverConnector* connector = _discoverRemote->
newConnector( *_client, "House");
connector->setResolver( new ArnDiscoverResolver());
connector->start();

ArnPersist* persist = new ArnPersist( this);
persist->setupDataBase();
persist->setMountPoint( Arn::pathLocal);

```

Examples:

[ArnDemoChatServer/MainWindow.cpp](#), and [ArnDemoChatServer/MainWindow.hpp](#).

Definition at line 94 of file [ArnDiscoverRemote.hpp](#).

14.18.2 Constructor & Destructor Documentation**14.18.2.1 ArnDiscoverRemote()**

```

ArnDiscoverRemote::ArnDiscoverRemote (
    QObject * parent = arnNullptr ) [explicit]

```

Definition at line 63 of file [ArnDiscoverRemote.cpp](#).

14.18.2.2 ~ArnDiscoverRemote()

```

ArnDiscoverRemote::~ArnDiscoverRemote ( )

```

Definition at line 75 of file [ArnDiscoverRemote.cpp](#).

14.18.3 Member Function Documentation**14.18.3.1 clientReadyToConnect**

```

void ArnDiscoverRemote::clientReadyToConnect (
    ArnClient * arnClient,
    const QString & id ) [signal]

```

Central signal for external client connection.

When activated external client connection by the connector method [ArnDiscoverConnector::setExternalClientConnect\(\)](#), this signal will be emitted when the client has been prepared to connect.

It's the responsibility of the receiver to do the actual client connect by [ArnClient::connectToArnList\(\)](#).

Parameters

in	<i>arnClient</i>	being ready for connection
in	<i>id</i>	is the identifier used in newConnector() , e.g "WeatherData-XYZ"

See also

[newConnector\(\)](#)
[ArnDiscoverConnector::setExternalClientConnect\(\)](#)

14.18.3.2 defaultService()

```
QString ArnDiscoverRemote::defaultService ( ) const
```

Return the default service name.

Returns

default service name, e.g. "Arn Default Service"

See also

[setDefaultService\(\)](#)

Definition at line 238 of file ArnDiscoverRemote.cpp.

14.18.3.3 initialServiceTimeout()

```
int ArnDiscoverRemote::initialServiceTimeout ( ) const
```

Return the time for initial timeout processing.

Returns

time in seconds

See also

[setInitialServiceTimeout\(\)](#)

Definition at line 255 of file ArnDiscoverRemote.cpp.

14.18.3.4 newConnector()

```
ArnDiscoverConnector * ArnDiscoverRemote::newConnector (
    ArnClient & client,
    const QString & id )
```

Create and return an [ArnDiscoverConnector](#) for handling remote client.

The [ArnDiscoverConnector](#) is internally connected to this [ArnDiscoverRemote](#).

The *id* should be chosen to describe the client target or its purpose. It's not a host address or necessarily a specific host, as there can be many possible addresses assigned to the [ArnDiscoverConnector](#).

The *id* will appear as an *Arn folder*, e.g. when *id* is "WeatherData-XYZ" the folder path will be "Sys/Discover/↔ Connect/WeatherData-XYZ".

Parameters

in	<i>client</i>	
in	<i>id</i>	identifies the target of the client connection, e.g "WeatherData-XYZ"

Returns

The [ArnDiscoverConnector](#)

Definition at line 141 of file ArnDiscoverRemote.cpp.

14.18.3.5 setDefaultService()

```
void ArnDiscoverRemote::setDefaultService (
    const QString & defaultService )
```

Set the default service name.

This default service name will be used when no service has been set before timeout. If calling with *defaultService* empty, it's ignored.

Parameters

in	<i>defaultService</i>	e.g. "My Default Service"
----	-----------------------	---------------------------

See also

[defaultService\(\)](#)

Definition at line 246 of file ArnDiscoverRemote.cpp.

14.18.3.6 setInitialServiceTimeout()

```
void ArnDiscoverRemote::setInitialServiceTimeout (
    int initialServiceTimeout )
```

Set the time for initial timeout processing.

Initial timeout-processing can wait upto this time, after that [defaultService\(\)](#) will be used as service.

Parameters

in	<i>initialServiceTimeout</i>	in seconds
----	------------------------------	------------

See also

[initialServiceTimeout\(\)](#)

Definition at line 263 of file ArnDiscoverRemote.cpp.

14.18.3.7 setService

```
void ArnDiscoverRemote::setService (
    const QString & service ) [virtual], [slot]
```

Set the service name.

Will update current advertised service name if this advertiser has been setup, otherwise the service name is stored for future use.

For remote control the service name is also available as an *Arn Data Object* at [local path](#) "Sys/Discover/This/↔Service/value".

All the functionality from [ArnDiscoverAdvertise::setService\(\)](#) apply.

Parameters

in	<i>service</i>	is the requested service name e.g. "My House Registry"
----	----------------	--

See also

[ArnDiscoverAdvertise::setService\(\)](#)
[currentService\(\)](#)
[advertiseService\(\)](#)

Definition at line 221 of file ArnDiscoverRemote.cpp.

14.18.3.8 startUseNewServer()

```
void ArnDiscoverRemote::startUseNewServer (
    ArnDiscover::Type discoverType,
    int port = -1 )
```

Start a new [ArnServer](#) and advertise as a service.

Handle advertising an internally created [ArnServer](#) as a service on the local network.

This method is typically used when there is no need to access the [ArnServer](#) class, which usually is the case in a client application. The [ArnServer](#) is then merely used to make the discover functionality remote controlled.

All the functionality from [startUseServer\(\)](#) do apply.

Parameters

in	<i>discoverType</i>	is used for discover filtering
in	<i>port</i>	is the port of the service, -1 gives Arn::defaultTcpPort , 0 gives dynamic port

See also

[setService\(\)](#)
[setDefaultService\(\)](#)
[startUseServer\(\)](#)

Definition at line 128 of file ArnDiscoverRemote.cpp.

14.18.3.9 startUseServer()

```
void ArnDiscoverRemote::startUseServer (
    ArnServer * arnServer,
    ArnDiscover::Type discoverType = ArnDiscover::Type::Server )
```

Start advertising the [ArnServer](#) as a service.

Handle advertising of an existing [ArnServer](#) as a service on the local network. Everything is fully automatic, including remote setting service name and support for persistent storage of the name. Status can be accessed via [Arn Data Objects](#).

Parameters

in	<i>arnServer</i>	is the ArnServer to be advertised
in	<i>discoverType</i>	is used for discover filtering

See also

[setService\(\)](#)
[setDefaultService\(\)](#)
[startUseNewServer\(\)](#)

Definition at line 80 of file ArnDiscoverRemote.cpp.

The documentation for this class was generated from the following files:

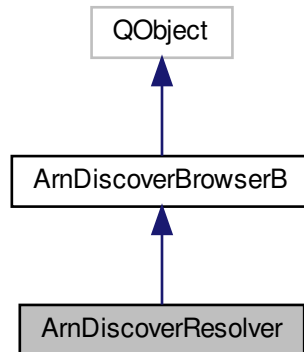
- [src/ArnInc/ArnDiscoverRemote.hpp \(4.0.0\)](#)
- [src/ArnDiscoverRemote.cpp \(4.0.0\)](#)

14.19 ArnDiscoverResolver Class Reference

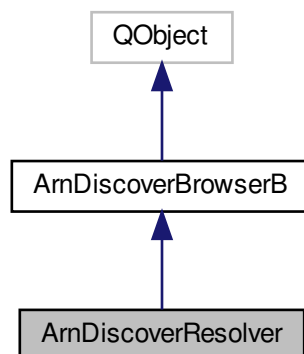
Resolv an [Arn](#) service.

```
#include <ArnDiscover.hpp>
```

Inheritance diagram for ArnDiscoverResolver:



Collaboration diagram for ArnDiscoverResolver:



Public Slots

- int [resolve](#) (const QString &serviceName, bool forceUpdate=true)
Resolve a specific service name.

Public Member Functions

- [ArnDiscoverResolver](#) (QObject *parent=arnNullptr)
- QString [defaultService](#) () const
Return the default service name.
- void [setDefaultService](#) (const QString &defaultService)
Set the default service name.

Additional Inherited Members

14.19.1 Detailed Description

Resolv an [Arn](#) service.

[About Arn Discover](#)

Example usage

```
// In class declare
ArnDiscoverResolver* _resolver;

// In class code
_resolver = new ArnDiscoverResolver( this);
connect( _resolver, SIGNAL(infoUpdated(int,ArnDiscoverInfo::State)),
        this, SLOT(doClientResolvChanged(int,ArnDiscoverInfo::State)));
_resolver->resolve("My service");

void XXX::doClientResolvChanged( int index, ArnDiscoverInfo::State state)
{
    const ArnDiscoverInfo& info = _resolver->infoByIndex( index);

    if (state == state.HostIp) {
        qDebug() << "Resolved service:" << info.serviceName()
                << " into host:" << info.hostWithInfo();
    }
    else if (info.isError()) {
        qDebug() << "Error resolving service:" << info.serviceName()
                << " code:" << info.resolveCode();
    }
}
```

Examples:

[ArnDemoChat/MainWindow.cpp](#).

Definition at line 556 of file ArnDiscover.hpp.

14.19.2 Constructor & Destructor Documentation

14.19.2.1 ArnDiscoverResolver()

```
ArnDiscoverResolver::ArnDiscoverResolver (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 256 of file ArnDiscover.cpp.

14.19.3 Member Function Documentation

14.19.3.1 defaultService()

```
QString ArnDiscoverResolver::defaultService ( ) const
```

Return the default service name.

This default service name will be used when [resolve\(\)](#) is called with empty service name.

Returns

default service name, e.g. "Arn Default Service"

See also

[setDefaultService\(\)](#)
[resolve\(\)](#)

Definition at line 276 of file ArnDiscover.cpp.

14.19.3.2 resolve

```
int ArnDiscoverResolver::resolve (
    const QString & serviceName,
    bool forceUpdate = true ) [slot]
```

Resolve a specific service name.

Only the specified service will be resolved, but there can be many ongoing resolves by calling this method multiple times with different service names. The [infoUpdated\(\)](#) signal will always be emitted when calling this method. The signal can also be emitted multiple times later regarding the same service.

Parameters

in	<i>serviceName</i>	is the service to be resolved
in	<i>forceUpdate</i>	when true, a new resolve is always done, otherwise a service name that already is resolved will not be resolved again.

Returns

index to service info

See also

[indexTold\(\)](#)
[infoUpdated\(\)](#)

Definition at line 268 of file ArnDiscover.cpp.

14.19.3.3 setDefaultService()

```
void ArnDiscoverResolver::setDefaultService (
    const QString & defaultService )
```

Set the default service name.

This default service name will be used when [resolve\(\)](#) is called with empty service name. If calling with *defaultService* empty, it is ignored.

Parameters

in	<i>defaultService</i>	e.g. "My Default Service"
----	-----------------------	---------------------------

See also

[defaultService\(\)](#)
[resolve\(\)](#)

Definition at line 284 of file ArnDiscover.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)
- [src/ArnDiscover.cpp \(4.0.0\)](#)

14.20 ArnError Class Reference

```
#include <ArnError.hpp>
```

Classes

- struct [StdCode](#)

Public Types

- enum [E](#) {
[Ok](#) = 0, [Info](#) = 1, [Warning](#) = 2, [Undef](#) = 15,
[Err_Undef](#) = 15, [CreateError](#) = 16, [Err_Custom](#) = 16, [NotFound](#),
[NotOpen](#), [AlreadyExist](#), [AlreadyOpen](#), [Retired](#),
[NotMainThread](#), [FolderNotOpen](#), [ItemNotOpen](#), [ItemNotSet](#),
[ConnectionError](#), [RecUnknown](#), [ScriptError](#), [RpcInvokeError](#),
[RpcReceiveError](#), [LoginBad](#), [RecNotExpected](#), [OpNotAllowed](#),
[Err_N](#) }

14.20.1 Detailed Description

Definition at line 38 of file ArnError.hpp.

14.20.2 Member Enumeration Documentation

14.20.2.1 E

```
enum ArnError::E
```

Enumerator

Ok	
Info	
Warning	
Undef	
Err_Undef	
CreateError	
Err_Custom	
NotFound	
NotOpen	
AlreadyExist	
AlreadyOpen	
Retired	
NotMainThread	
FolderNotOpen	
ItemNotOpen	
ItemNotSet	
ConnectionError	
RecUnknown	
ScriptError	
RpcInvokeError	
RpcReceiveError	
LoginBad	
RecNotExpected	
OpNotAllowed	
Err_N	

Definition at line 43 of file ArnError.hpp.

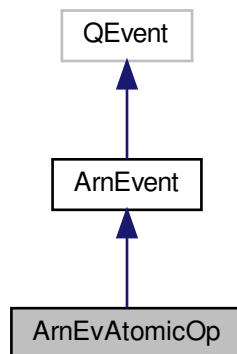
The documentation for this class was generated from the following file:

- [src/ArnInc/ArnError.hpp \(4.0.0\)](#)

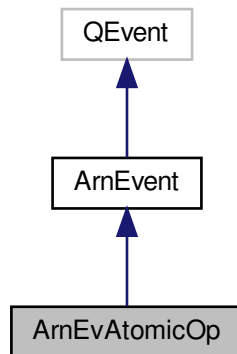
14.21 ArnEvAtomicOp Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvAtomicOp:



Collaboration diagram for ArnEvAtomicOp:



Public Types

- typedef [ArnAtomicOp](#) `Op`

Public Member Functions

- [ArnEvAtomicOp](#) (int `op`, const QVariant &`arg1`, const QVariant &`arg2`)
- virtual `~ArnEvAtomicOp` ()
- virtual [ArnEvent](#) * `makeHeapClone` ()
- const `Op` & `op` () const
- const QVariant & `arg1` () const
- const QVariant & `arg2` () const

Static Public Member Functions

- static QEvent::Type [type](#) ()

Additional Inherited Members

14.21.1 Detailed Description

Definition at line 258 of file ArnEvent.hpp.

14.21.2 Member Typedef Documentation

14.21.2.1 Op

```
typedef ArnAtomicOp ArnEvAtomicOp::Op
```

Definition at line 261 of file ArnEvent.hpp.

14.21.3 Constructor & Destructor Documentation

14.21.3.1 ArnEvAtomicOp()

```
ArnEvAtomicOp::ArnEvAtomicOp (
    int op,
    const QVariant & arg1,
    const QVariant & arg2 )
```

Definition at line 227 of file ArnEvent.cpp.

14.21.3.2 ~ArnEvAtomicOp()

```
ArnEvAtomicOp::~ArnEvAtomicOp ( ) [virtual]
```

Definition at line 236 of file ArnEvent.cpp.

14.21.4 Member Function Documentation

14.21.4.1 arg1()

```
const QVariant& ArnEvAtomicOp::arg1 ( ) const [inline]
```

Definition at line 277 of file ArnEvent.hpp.

14.21.4.2 arg2()

```
const QVariant& ArnEvAtomicOp::arg2 ( ) const [inline]
```

Definition at line 280 of file ArnEvent.hpp.

14.21.4.3 makeHeapClone()

```
ArnEvent * ArnEvAtomicOp::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 249 of file ArnEvent.cpp.

14.21.4.4 op()

```
const Op& ArnEvAtomicOp::op ( ) const [inline]
```

Definition at line 274 of file ArnEvent.hpp.

14.21.4.5 type()

```
QEvent::Type ArnEvAtomicOp::type ( ) [static]
```

Definition at line 241 of file ArnEvent.cpp.

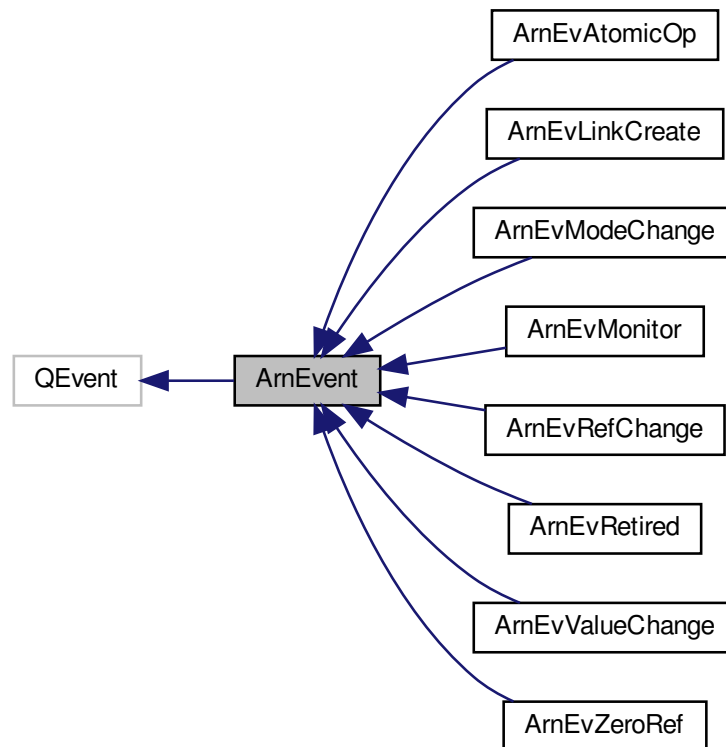
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

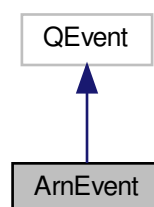
14.22 ArnEvent Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvent:



Collaboration diagram for ArnEvent:



Public Types

- typedef [ArnEventIdx](#) Idx

Public Member Functions

- [ArnEvent](#) (QEvent::Type type)
- virtual [~ArnEvent](#) ()
- int [toldx](#) () const
- QString [toString](#) () const
- virtual [ArnEvent * makeHeapClone](#) ()=0
- void * [target](#) () const
- void [setTarget](#) (void *target)
- void [setTargetPendingChain](#) ([ArnEvent **targetPendingChain=arnNullptr](#))
- void [setTargetMutex](#) (QMutex *targetMutex)
- void [inhibitPendingChain](#) ()

Static Public Member Functions

- static int [baseType](#) (int setVal=-1)
- static bool [isArnEvent](#) (int evType)
- static int [toldx](#) (QEvent::Type type)
- static QString [toString](#) (QEvent::Type type)

Protected Member Functions

- [ArnEvent * copyOpt](#) (const [ArnEvent *other](#))

14.22.1 Detailed Description

Definition at line 91 of file ArnEvent.hpp.

14.22.2 Member Typedef Documentation

14.22.2.1 Idx

```
typedef ArnEventIdx ArnEvent::Idx
```

Definition at line 101 of file ArnEvent.hpp.

14.22.3 Constructor & Destructor Documentation

14.22.3.1 ArnEvent()

```
ArnEvent::ArnEvent (
    QEvent::Type type )
```

Definition at line 37 of file ArnEvent.cpp.

14.22.3.2 ~ArnEvent()

```
ArnEvent::~ArnEvent ( ) [virtual]
```

Definition at line 48 of file ArnEvent.cpp.

14.22.4 Member Function Documentation

14.22.4.1 baseType()

```
int ArnEvent::baseType (
    int setVal = -1 ) [static]
```

Definition at line 62 of file ArnEvent.cpp.

14.22.4.2 copyOpt()

```
ArnEvent * ArnEvent::copyOpt (
    const ArnEvent * other ) [protected]
```

Definition at line 129 of file ArnEvent.cpp.

14.22.4.3 inhibitPendingChain()

```
void ArnEvent::inhibitPendingChain ( )
```

Definition at line 167 of file ArnEvent.cpp.

14.22.4.4 isArnEvent()

```
bool ArnEvent::isArnEvent (
    int evType ) [static]
```

Definition at line 89 of file ArnEvent.cpp.

14.22.4.5 makeHeapClone()

```
virtual ArnEvent* ArnEvent::makeHeapClone ( ) [pure virtual]
```

Implemented in [ArnEvRefChange](#), [ArnEvAtomicOp](#), [ArnEvValueChange](#), [ArnEvZeroRef](#), [ArnEvRetired](#), [ArnEvMonitor](#), [ArnEvModeChange](#), and [ArnEvLinkCreate](#).

14.22.4.6 setTarget()

```
void ArnEvent::setTarget (
    void * target )
```

Definition at line 138 of file ArnEvent.cpp.

14.22.4.7 setTargetMutex()

```
void ArnEvent::setTargetMutex (
    QMutex * targetMutex )
```

Definition at line 161 of file ArnEvent.cpp.

14.22.4.8 setTargetPendingChain()

```
void ArnEvent::setTargetPendingChain (
    ArnEvent ** targetPendingChain = arnNullptr )
```

Definition at line 144 of file ArnEvent.cpp.

14.22.4.9 target()

```
void* ArnEvent::target ( ) const [inline]
```

Definition at line 116 of file ArnEvent.hpp.

14.22.4.10 toIdx() [1/2]

```
int ArnEvent::toIdx (
    QEvent::Type type ) [static]
```

Definition at line 102 of file ArnEvent.cpp.

14.22.4.11 toIdx() [2/2]

```
int ArnEvent::toIdx ( ) const
```

Definition at line 109 of file ArnEvent.cpp.

14.22.4.12 toString() [1/2]

```
QString ArnEvent::toString (
    QEvent::Type type ) [static]
```

Definition at line 116 of file ArnEvent.cpp.

14.22.4.13 toString() [2/2]

```
QString ArnEvent::toString ( ) const
```

Definition at line 123 of file ArnEvent.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

14.23 ArnEventIdx Class Reference

```
#include <ArnEvent.hpp>
```

Public Types

- enum `E` {
 `QtEvent` = -1, `ValueChange` = 0, `AtomicOp`, `LinkCreate`,
 `ModeChange`, `Monitor`, `Retired`, `ZeroRef`,
 `RefChange`, `N` }

14.23.1 Detailed Description

Definition at line 47 of file ArnEvent.hpp.

14.23.2 Member Enumeration Documentation

14.23.2.1 E

```
enum ArnEventIdx::E
```

Enumerator

QtEvent	
ValueChanged	
AtomicOp	
LinkCreate	
ModeChange	
Monitor	
Retired	
ZeroRef	
RefChange	
N	Max index.

Definition at line 51 of file ArnEvent.hpp.

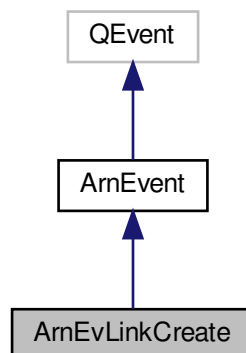
The documentation for this class was generated from the following file:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)

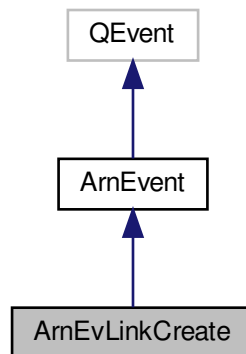
14.24 ArnEvLinkCreate Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvLinkCreate:



Collaboration diagram for ArnEvLinkCreate:



Public Member Functions

- [ArnEvLinkCreate](#) (const QString &path, ArnLink *arnLink, bool isLastLink)
- virtual [ArnEvent](#) * [makeHeapClone](#) ()
- const QString & [path](#) () const
- ArnLink * [arnLink](#) () const
- bool [isLastLink](#) () const

Static Public Member Functions

- static QEvent::Type [type](#) ()

Additional Inherited Members

14.24.1 Detailed Description

Definition at line 129 of file ArnEvent.hpp.

14.24.2 Constructor & Destructor Documentation

14.24.2.1 ArnEvLinkCreate()

```
ArnEvLinkCreate::ArnEvLinkCreate (
    const QString & path,
    ArnLink * arnLink,
    bool isLastLink )
```

Definition at line 255 of file ArnEvent.cpp.

14.24.3 Member Function Documentation

14.24.3.1 arnLink()

```
ArnLink* ArnEvLinkCreate::arnLink ( ) const [inline]
```

Definition at line 143 of file ArnEvent.hpp.

14.24.3.2 isLastLink()

```
bool ArnEvLinkCreate::isLastLink ( ) const [inline]
```

Definition at line 146 of file ArnEvent.hpp.

14.24.3.3 makeHeapClone()

```
ArnEvent * ArnEvLinkCreate::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 272 of file ArnEvent.cpp.

14.24.3.4 path()

```
const QString& ArnEvLinkCreate::path ( ) const [inline]
```

Definition at line 140 of file ArnEvent.hpp.

14.24.3.5 type()

```
QEvent::Type ArnEvLinkCreate::type ( ) [static]
```

Definition at line 264 of file ArnEvent.cpp.

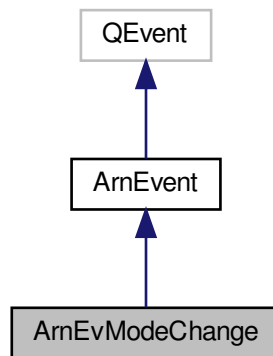
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

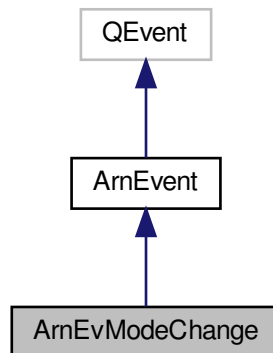
14.25 ArnEvModeChange Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvModeChange:



Collaboration diagram for ArnEvModeChange:



Public Member Functions

- [ArnEvModeChange](#) (const QString &path, uint linkId, Arn::ObjectMode mode)
- virtual [ArnEvent](#) * [makeHeapClone](#) ()
- const QString & [path](#) () const
- uint [linkId](#) () const
- [Arn::ObjectMode](#) mode () const

Static Public Member Functions

- static `QEvent::Type type ()`

Additional Inherited Members

14.25.1 Detailed Description

Definition at line 151 of file `ArnEvent.hpp`.

14.25.2 Constructor & Destructor Documentation

14.25.2.1 `ArnEvModeChange()`

```
ArnEvModeChange::ArnEvModeChange (
    const QString & path,
    uint linkId,
    Arn::ObjectMode mode )
```

Definition at line 279 of file `ArnEvent.cpp`.

14.25.3 Member Function Documentation

14.25.3.1 `linkId()`

```
uint ArnEvModeChange::linkId ( ) const [inline]
```

Definition at line 165 of file `ArnEvent.hpp`.

14.25.3.2 `makeHeapClone()`

```
ArnEvent * ArnEvModeChange::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 296 of file `ArnEvent.cpp`.

14.25.3.3 mode()

```
Arn::ObjectMode ArnEvModeChange::mode ( ) const [inline]
```

Definition at line 168 of file ArnEvent.hpp.

14.25.3.4 path()

```
const QString& ArnEvModeChange::path ( ) const [inline]
```

Definition at line 162 of file ArnEvent.hpp.

14.25.3.5 type()

```
QEvent::Type ArnEvModeChange::type ( ) [static]
```

Definition at line 288 of file ArnEvent.cpp.

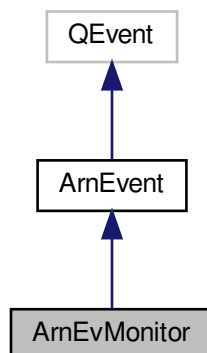
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

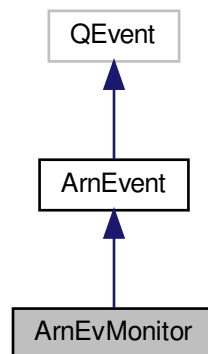
14.26 ArnEvMonitor Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvMonitor:



Collaboration diagram for ArnEvMonitor:



Public Member Functions

- [ArnEvMonitor](#) (int [monEvType](#), const QByteArray &[data](#), bool [isLocal](#), void *[sessionHandler](#))
- virtual [ArnEvent](#) * [makeHeapClone](#) ()
- int [monEvType](#) () const
- const QByteArray & [data](#) () const
- bool [isLocal](#) () const
- void * [sessionHandler](#) () const

Static Public Member Functions

- static QEvent::Type [type](#) ()

Additional Inherited Members

14.26.1 Detailed Description

Definition at line 173 of file ArnEvent.hpp.

14.26.2 Constructor & Destructor Documentation

14.26.2.1 ArnEvMonitor()

```
ArnEvMonitor::ArnEvMonitor (
    int monEvType,
    const QByteArray & data,
    bool isLocal,
    void * sessionHandler )
```

Definition at line 303 of file ArnEvent.cpp.

14.26.3 Member Function Documentation

14.26.3.1 data()

```
const QByteArray& ArnEvMonitor::data ( ) const [inline]
```

Definition at line 188 of file ArnEvent.hpp.

14.26.3.2 isLocal()

```
bool ArnEvMonitor::isLocal ( ) const [inline]
```

Definition at line 191 of file ArnEvent.hpp.

14.26.3.3 makeHeapClone()

```
ArnEvent * ArnEvMonitor::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 321 of file ArnEvent.cpp.

14.26.3.4 monEvType()

```
int ArnEvMonitor::monEvType ( ) const [inline]
```

Definition at line 185 of file ArnEvent.hpp.

14.26.3.5 sessionHandler()

```
void* ArnEvMonitor::sessionHandler ( ) const [inline]
```

Definition at line 194 of file ArnEvent.hpp.

14.26.3.6 type()

```
QEvent::Type ArnEvMonitor::type ( ) [static]
```

Definition at line 313 of file ArnEvent.cpp.

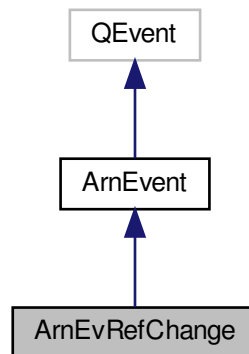
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

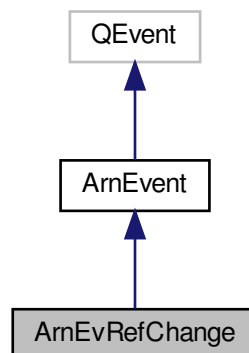
14.27 ArnEvRefChange Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvRefChange:



Collaboration diagram for ArnEvRefChange:



Public Member Functions

- [ArnEvRefChange](#) (int *refStep*)
- virtual [~ArnEvRefChange](#) ()
- virtual [ArnEvent](#) * [makeHeapClone](#) ()
- int [refStep](#) () const

Static Public Member Functions

- static [QEvent::Type](#) [type](#) ()

Additional Inherited Members

14.27.1 Detailed Description

Definition at line 285 of file [ArnEvent.hpp](#).

14.27.2 Constructor & Destructor Documentation

14.27.2.1 ArnEvRefChange()

```
ArnEvRefChange::ArnEvRefChange (
    int refStep )
```

Definition at line 374 of file [ArnEvent.cpp](#).

14.27.2.2 ~ArnEvRefChange()

```
ArnEvRefChange::~ArnEvRefChange ( ) [virtual]
```

Definition at line 381 of file [ArnEvent.cpp](#).

14.27.3 Member Function Documentation

14.27.3.1 makeHeapClone()

```
ArnEvent * ArnEvRefChange::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 394 of file [ArnEvent.cpp](#).

14.27.3.2 refStep()

```
int ArnEvRefChange::refStep ( ) const [inline]
```

Definition at line 295 of file ArnEvent.hpp.

14.27.3.3 type()

```
QEvent::Type ArnEvRefChange::type ( ) [static]
```

Definition at line 386 of file ArnEvent.cpp.

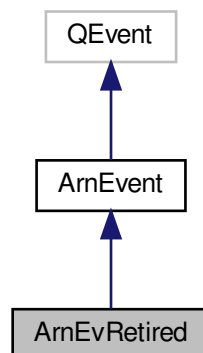
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

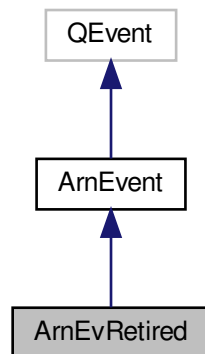
14.28 ArnEvRetired Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvRetired:



Collaboration diagram for ArnEvRetired:



Public Member Functions

- [ArnEvRetired](#) (ArnLink *[startLink](#), bool [isBelow](#), bool [isGlobal](#))
- virtual ArnEvent * [makeHeapClone](#) ()
- ArnLink * [startLink](#) () const
- bool [isBelow](#) () const
- bool [isGlobal](#) () const

Static Public Member Functions

- static QEvent::Type [type](#) ()

Additional Inherited Members

14.28.1 Detailed Description

Definition at line 199 of file ArnEvent.hpp.

14.28.2 Constructor & Destructor Documentation

14.28.2.1 ArnEvRetired()

```
ArnEvRetired::ArnEvRetired (  
    ArnLink * startLink,  
    bool isBelow,  
    bool isGlobal )
```

Definition at line 328 of file ArnEvent.cpp.

14.28.3 Member Function Documentation

14.28.3.1 isBelow()

```
bool ArnEvRetired::isBelow ( ) const [inline]
```

Definition at line 213 of file ArnEvent.hpp.

14.28.3.2 isGlobal()

```
bool ArnEvRetired::isGlobal ( ) const [inline]
```

Definition at line 216 of file ArnEvent.hpp.

14.28.3.3 makeHeapClone()

```
ArnEvent * ArnEvRetired::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 345 of file ArnEvent.cpp.

14.28.3.4 startLink()

```
ArnLink* ArnEvRetired::startLink ( ) const [inline]
```

Definition at line 210 of file ArnEvent.hpp.

14.28.3.5 type()

```
QEvent::Type ArnEvRetired::type ( ) [static]
```

Definition at line 337 of file ArnEvent.cpp.

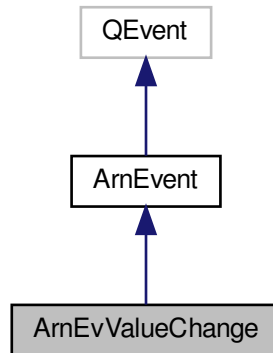
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

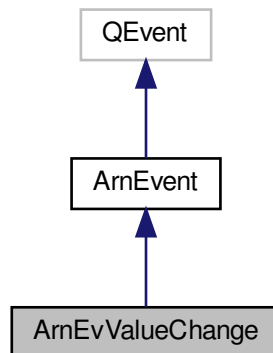
14.29 ArnEvValueChange Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvValueChange:



Collaboration diagram for ArnEvValueChange:



Public Member Functions

- [ArnEvValueChange](#) (int `sendId`, const QByteArray *`valueData`, const ArnLinkHandle &`handleData`)
- virtual `~ArnEvValueChange` ()
- virtual `ArnEvent` * `makeHeapClone` ()
- int `sendId` () const
- const QByteArray * `valueData` () const
- const ArnLinkHandle & `handleData` () const

Static Public Member Functions

- static QEvent::Type [type](#) ()

Additional Inherited Members

14.29.1 Detailed Description

Definition at line 235 of file ArnEvent.hpp.

14.29.2 Constructor & Destructor Documentation

14.29.2.1 ArnEvValueChange()

```
ArnEvValueChange::ArnEvValueChange (
    int sendId,
    const QByteArray * valueData,
    const ArnLinkHandle & handleData )
```

Definition at line 189 of file ArnEvent.cpp.

14.29.2.2 ~ArnEvValueChange()

```
ArnEvValueChange::~ArnEvValueChange ( ) [virtual]
```

Definition at line 204 of file ArnEvent.cpp.

14.29.3 Member Function Documentation

14.29.3.1 handleData()

```
const ArnLinkHandle& ArnEvValueChange::handleData ( ) const [inline]
```

Definition at line 253 of file ArnEvent.hpp.

14.29.3.2 makeHeapClone()

```
ArnEvent * ArnEvValueChange::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 221 of file ArnEvent.cpp.

14.29.3.3 sendId()

```
int ArnEvValueChange::sendId ( ) const [inline]
```

Definition at line 247 of file ArnEvent.hpp.

14.29.3.4 type()

```
QEvent::Type ArnEvValueChange::type ( ) [static]
```

Definition at line 213 of file ArnEvent.cpp.

14.29.3.5 valueData()

```
const QByteArray* ArnEvValueChange::valueData ( ) const [inline]
```

Definition at line 250 of file ArnEvent.hpp.

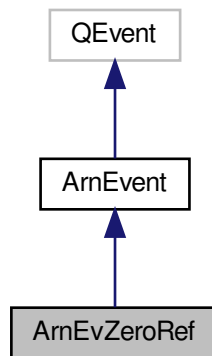
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

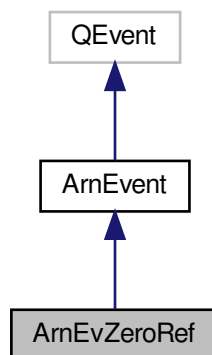
14.30 ArnEvZeroRef Class Reference

```
#include <ArnEvent.hpp>
```

Inheritance diagram for ArnEvZeroRef:



Collaboration diagram for ArnEvZeroRef:



Public Member Functions

- [ArnEvZeroRef](#) ([ArnLink](#) *[arnLink](#))
- virtual [ArnEvent](#) * [makeHeapClone](#) ()
- [ArnLink](#) * [arnLink](#) () const

Static Public Member Functions

- static `QEvent::Type type ()`

Additional Inherited Members

14.30.1 Detailed Description

Definition at line 221 of file ArnEvent.hpp.

14.30.2 Constructor & Destructor Documentation

14.30.2.1 ArnEvZeroRef()

```
ArnEvZeroRef::ArnEvZeroRef (  
    ArnLink * arnLink )
```

Definition at line 352 of file ArnEvent.cpp.

14.30.3 Member Function Documentation

14.30.3.1 arnLink()

```
ArnLink* ArnEvZeroRef::arnLink ( ) const [inline]
```

Definition at line 230 of file ArnEvent.hpp.

14.30.3.2 makeHeapClone()

```
ArnEvent * ArnEvZeroRef::makeHeapClone ( ) [virtual]
```

Implements [ArnEvent](#).

Definition at line 367 of file ArnEvent.cpp.

14.30.3.3 type()

```
QEvent::Type ArnEvZeroRef::type ( ) [static]
```

Definition at line 359 of file ArnEvent.cpp.

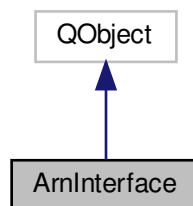
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(4.0.0\)](#)
- [src/ArnEvent.cpp \(4.0.0\)](#)

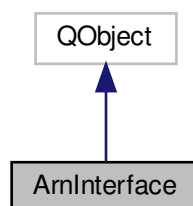
14.31 ArnInterface Class Reference

```
#include <ArnInterface.hpp>
```

Inheritance diagram for ArnInterface:



Collaboration diagram for ArnInterface:



Public Types

- enum [SameValue](#) { [SameValue_Accept](#) = Arn::SameValue::Accept, [SameValue_Ignore](#) = Arn::SameValue::Ignore, [SameValue_DefaultAction](#) = Arn::SameValue::DefaultAction }
- Action when assigning same value to an [ArnItem](#).*
- enum [DataType](#) { [DataType_Null](#) = Arn::DataType::Null, [DataType_Int](#) = Arn::DataType::Int, [DataType_Double](#) = Arn::DataType::Double, [DataType_Real](#) = Arn::DataType::Real, [DataType_ByteArray](#) = Arn::DataType::ByteArray, [DataType_String](#) = Arn::DataType::String, [DataType_Variant](#) = Arn::DataType::Variant }
- Data type of an [Arn Data Object](#)*
- enum [ObjectMode](#) { [ObjectMode_BiDir](#) = Arn::ObjectMode::BiDir, [ObjectMode_Pipe](#) = Arn::ObjectMode::Pipe, [ObjectMode_Save](#) = Arn::ObjectMode::Save }
- General global mode of an [Arn Data Object](#)*
- enum [NameF](#) { [NameF_Default](#) = Arn::NameF::Default, [NameF_NoFolderMark](#) = Arn::NameF::NoFolderMark, [NameF_EmptyOk](#) = Arn::NameF::EmptyOk, [NameF_Relative](#) = Arn::NameF::Relative }
- Selects a format for path or item name.*

Public Slots

- QVariant [value](#) (const QString &path)
See [ArnM::valueVariant\(\)](#)
- QVariant [variant](#) (const QString &path)
See [ArnM::valueVariant\(\)](#)
- QString [string](#) (const QString &path)
See [ArnM::valueString\(\)](#)
- QByteArray [bytes](#) (const QString &path)
See [ArnM::valueByteArray\(\)](#)
- double [num](#) (const QString &path)
See [ArnM::valueDouble\(\)](#)
- int [intNum](#) (const QString &path)
See [ArnM::valueInt\(\)](#)
- QStringList [items](#) (const QString &path)
See [ArnM::items\(\)](#)
- bool [exist](#) (const QString &path)
See [ArnM::exist\(\)](#)
- bool [isFolder](#) (const QString &path)
See [ArnM::isFolder\(\)](#)
- bool [isLeaf](#) (const QString &path)
See [ArnM::isLeaf\(\)](#)
- void [setValue](#) (const QString &path, const QVariant &value)
See [ArnM::setValue\(\)](#)
- void [setVariant](#) (const QString &path, const QVariant &value, const QString &typeName=QString())
See [ArnM::setValue\(\)](#)
- void [setString](#) (const QString &path, const QString &value)
See [ArnM::setValue\(\)](#)
- void [setBytes](#) (const QString &path, const QByteArray &value)
See [ArnM::setValue\(\)](#)
- void [setNum](#) (const QString &path, double value)
See [ArnM::setValue\(\)](#)
- void [setIntNum](#) (const QString &path, int value)

- See [ArnM::setValue\(\)](#)
- bool [isFolderPath](#) (const QString &path)
 - See [Arn::isFolderPath\(\)](#)
- bool [isProviderPath](#) (const QString &path)
 - See [Arn::isProviderPath\(\)](#)
- QString [itemName](#) (const QString &path)
 - See [Arn::itemName\(\)](#)
- QString [twinPath](#) (const QString &path)
 - See [Arn::twinPath\(\)](#)
- QString [changeBasePath](#) (const QString &oldBasePath, const QString &newBasePath, const QString &path)
 - See [Arn::changeBasePath\(\)](#)
- QString [childPath](#) (const QString &parentPath, const QString &posterityPath)
 - See [Arn::childPath\(\)](#)
- QString [makePath](#) (const QString &parentPath, const QString &itemName)
 - See [Arn::makePath\(\)](#)
- QString [providerPath](#) (const QString &path, bool giveProviderPath=true)
 - See [Arn::providerPath\(\)](#)

Properties

- QString [info](#)
 - See [ArnM::info\(\)](#)

14.31.1 Detailed Description

Definition at line 39 of file ArnInterface.hpp.

14.31.2 Member Enumeration Documentation

14.31.2.1 DataType

enum [ArnInterface::DataType](#)

Data type of an [Arn Data Object](#)

Enumerator

DataType_Null	
DataType_Int	
DataType_Double	
DataType_Real	
DataType_ByteArray	
DataType_String	
DataType_Variant	

Definition at line 57 of file ArnInterface.hpp.

14.31.2.2 NameF

```
enum ArnInterface::NameF
```

Selects a format for path or item name.

Enumerator

NameF_Default	Empty not ok, Path: Absolute Item: FolderMark.
NameF_NoFolderMark	Only on discrete names, no effect on path. "test/" ==> "test".
NameF_EmptyOk	Path: "@/test" ==> "/test", Item: "@" ==> "".
NameF_Relative	Only on path, no effect on discrete names. "/test/value" ==> "test/value".

Definition at line 80 of file ArnInterface.hpp.

14.31.2.3 ObjectMode

```
enum ArnInterface::ObjectMode
```

General global mode of an *Arn Data Object*

Enumerator

ObjectMode_BiDir	A two way object, typically for validation or pipe.
ObjectMode_Pipe	Implies <i>BiDir</i> and all data is preserved as a stream.
ObjectMode_Save	Data is persistent and will be saved.

Definition at line 69 of file ArnInterface.hpp.

14.31.2.4 SameValue

```
enum ArnInterface::SameValue
```

Action when assigning same value to an *ArnItem*.

Enumerator

SameValue_Accept	Assigning same value generates an update of the <i>Arn Data Object</i>
SameValue_Ignore	Assigning same value is ignored.
SameValue_DefaultAction	Assigning same value gives default action set in <i>ArnM</i> or <i>ArnItem</i> .

Definition at line 46 of file ArnInterface.hpp.

14.31.3 Member Function Documentation

14.31.3.1 bytes

```
QByteArray ArnInterface::bytes (  
    const QString & path ) [inline], [slot]
```

See [ArnM::valueByteArray\(\)](#)

Definition at line 110 of file ArnInterface.hpp.

14.31.3.2 changeBasePath

```
QString ArnInterface::changeBasePath (  
    const QString & oldBasePath,  
    const QString & newBasePath,  
    const QString & path ) [inline], [slot]
```

See [Arn::changeBasePath\(\)](#)

Definition at line 177 of file ArnInterface.hpp.

14.31.3.3 childPath

```
QString ArnInterface::childPath (  
    const QString & parentPath,  
    const QString & posterityPath ) [inline], [slot]
```

See [Arn::childPath\(\)](#)

Definition at line 181 of file ArnInterface.hpp.

14.31.3.4 exist

```
bool ArnInterface::exist (  
    const QString & path ) [inline], [slot]
```

See [ArnM::exist\(\)](#)

Definition at line 126 of file ArnInterface.hpp.

14.31.3.5 intNum

```
int ArnInterface::intNum (
    const QString & path ) [inline], [slot]
```

See [ArnM::valueInt\(\)](#)

Definition at line 120 of file ArnInterface.hpp.

14.31.3.6 isFolder

```
bool ArnInterface::isFolder (
    const QString & path ) [inline], [slot]
```

See [ArnM::isFolder\(\)](#)

Definition at line 129 of file ArnInterface.hpp.

14.31.3.7 isFolderPath

```
bool ArnInterface::isFolderPath (
    const QString & path ) [inline], [slot]
```

See [Arn::isFolderPath\(\)](#)

Definition at line 165 of file ArnInterface.hpp.

14.31.3.8 isLeaf

```
bool ArnInterface::isLeaf (
    const QString & path ) [inline], [slot]
```

See [ArnM::isLeaf\(\)](#)

Definition at line 132 of file ArnInterface.hpp.

14.31.3.9 isProviderPath

```
bool ArnInterface::isProviderPath (
    const QString & path ) [inline], [slot]
```

See [Arn::isProviderPath\(\)](#)

Definition at line 168 of file ArnInterface.hpp.

14.31.3.10 itemName

```
QString ArnInterface::itemName (
    const QString & path ) [inline], [slot]
```

See [Arn::itemName\(\)](#)

Definition at line 171 of file ArnInterface.hpp.

14.31.3.11 items

```
QStringList ArnInterface::items (
    const QString & path ) [inline], [slot]
```

See [ArnM::items\(\)](#)

Definition at line 123 of file ArnInterface.hpp.

14.31.3.12 makePath

```
QString ArnInterface::makePath (
    const QString & parentPath,
    const QString & itemName ) [inline], [slot]
```

See [Arn::makePath\(\)](#)

Definition at line 185 of file ArnInterface.hpp.

14.31.3.13 num

```
double ArnInterface::num (
    const QString & path ) [inline], [slot]
```

See [ArnM::valueDouble\(\)](#)

Definition at line 116 of file ArnInterface.hpp.

14.31.3.14 providerPath

```
QString ArnInterface::providerPath (
    const QString & path,
    bool giveProviderPath = true ) [inline], [slot]
```

See [Arn::providerPath\(\)](#)

Definition at line 189 of file ArnInterface.hpp.

14.31.3.15 setBytes

```
void ArnInterface::setBytes (
    const QString & path,
    const QByteArray & value ) [inline], [slot]
```

See [ArnM::setValue\(\)](#)

Definition at line 147 of file ArnInterface.hpp.

14.31.3.16 setIntNum

```
void ArnInterface::setIntNum (
    const QString & path,
    int value ) [inline], [slot]
```

See [ArnM::setValue\(\)](#)

Definition at line 159 of file ArnInterface.hpp.

14.31.3.17 setNum

```
void ArnInterface::setNum (
    const QString & path,
    double value ) [inline], [slot]
```

See [ArnM::setValue\(\)](#)

Definition at line 154 of file ArnInterface.hpp.

14.31.3.18 setString

```
void ArnInterface::setString (
    const QString & path,
    const QString & value ) [inline], [slot]
```

See [ArnM::setValue\(\)](#)

Definition at line 143 of file ArnInterface.hpp.

14.31.3.19 setValue

```
void ArnInterface::setValue (
    const QString & path,
    const QVariant & value ) [inline], [slot]
```

See [ArnM::setValue\(\)](#)

Definition at line 135 of file ArnInterface.hpp.

14.31.3.20 setVariant

```
void ArnInterface::setVariant (
    const QString & path,
    const QVariant & value,
    const QString & typeName = QString() ) [inline], [slot]
```

See [ArnM::setValue\(\)](#)

Definition at line 139 of file ArnInterface.hpp.

14.31.3.21 string

```
QString ArnInterface::string (
    const QString & path ) [inline], [slot]
```

See [ArnM::valueString\(\)](#)

Definition at line 107 of file ArnInterface.hpp.

14.31.3.22 twinPath

```
QString ArnInterface::twinPath (
    const QString & path ) [inline], [slot]
```

See [Arn::twinPath\(\)](#)

Definition at line 174 of file ArnInterface.hpp.

14.31.3.23 value

```
QVariant ArnInterface::value (
    const QString & path ) [inline], [slot]
```

See [ArnM::valueVariant\(\)](#)

Definition at line 101 of file ArnInterface.hpp.

14.31.3.24 variant

```
QVariant ArnInterface::variant (
    const QString & path ) [inline], [slot]
```

See [ArnM::valueVariant\(\)](#)

Definition at line 104 of file ArnInterface.hpp.

14.31.4 Property Documentation

14.31.4.1 info

```
QString ArnInterface::info [read]
```

See [ArnM::info\(\)](#)

Definition at line 43 of file ArnInterface.hpp.

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

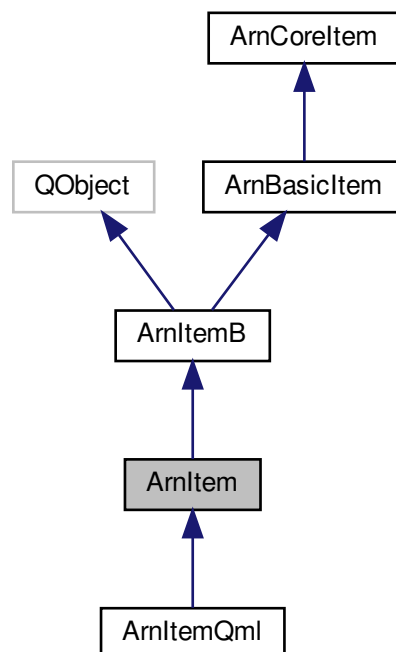
- [src/ArnInc/ArnInterface.hpp \(4.0.0\)](#)

14.32 ArnItem Class Reference

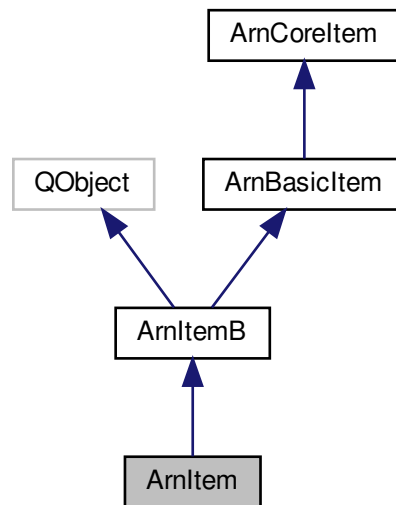
Handle for an *Arn Data Object*.

```
#include <ArnItem.hpp>
```

Inheritance diagram for ArnItem:



Collaboration diagram for ArnItem:



Public Slots

- void `setValue` (int value)
Assign an integer to an `Arn` Data Object
- void `setValue` (double value)
Assign an `ARNREAL` to an `Arn` Data Object
- void `setValue` (bool value)
Assign a bool to an `Arn` Data Object
- void `setValue` (const `QString` &value)
Assign a `QString` to an `Arn` Data Object
- void `setValue` (const `QByteArray` &value)
Assign a `QByteArray` to an `Arn` Data Object
- void `setValue` (const `QVariant` &value)
Assign a `QVariant` to an `Arn` Data Object
- void `setValue` (const char *value)
Assign a `char` to an `Arn` Data Object*
- void `toggleBool` ()
Toggle the bool at the `Arn` Data Object

Signals

- void `changed` ()
Signals emitted when data in `Arn` Data Object is changed.
- void `changed` (int value)
- void `changed` (double value)
- void `changed` (bool value)

- void `changed` (const QString &value)
- void `changed` (const QByteArray &value)
- void `changed` (const QVariant &value)
- void `modeChanged` (Arn::ObjectMode mode)
 - Signal emitted when mode in Arn Data Object is changed.*
- void `arnItemCreated` (const QString &path)
 - Signal emitted when an Arn Data Object is created in the tree below.*
- void `arnModeChanged` (const QString &path, uint linkId, Arn::ObjectMode mode)
 - Signal emitted when an Arn Data Object in the tree below has a general mode change.*

Public Member Functions

- `ArnItem` (QObject *parent=arnNullptr)
 - Standard constructor of a closed handle.*
- `ArnItem` (const QString &path, QObject *parent=arnNullptr)
 - Construction of a handle to a path.*
- `ArnItem` (const ArnItem &itemTemplate, const QString &path, QObject *parent=arnNullptr)
 - Construction of a handle to a path with a template for modes*
- virtual `~ArnItem` ()
- bool `openUuid` (const QString &path)
 - Open a handle to an Arn Object with a unique uuid name.*
- bool `openUuidPipe` (const QString &path)
 - Open a handle to an Arn Pipe Object with a unique uuid name.*
- bool `openFolder` (const QString &path)
 - Open a handle to an Arn folder.*
- bool `isFolder` () const
- bool `isProvider` () const
- `Arn::DataType` type () const
 - The type stored in the Arn Data Object*
- void `setIgnoreSameValue` (bool isIgnore=true)
 - Set skipping assignment of equal value.*
- bool `isIgnoreSameValue` ()
- void `addMode` (Arn::ObjectMode mode)
 - Add general mode settings for this Arn Data Object*
- `Arn::ObjectMode` `getMode` () const
- `Arn::ObjectSyncMode` `syncMode` () const
- `ArnItem` & `setTemplate` (bool isTemplate=true)
 - Mark this ArnItem as a template.*
- bool `isTemplate` () const
- `ArnItem` & `setBiDirMode` ()
 - Set general mode as Bidirectional for this Arn Data Object*
- bool `isBiDirMode` () const
- `ArnItem` & `setPipeMode` ()
 - Set general mode as Pipe for this Arn Data Object*
- bool `isPipeMode` () const
- `ArnItem` & `setSaveMode` ()
 - Set general mode as Save for this Arn Data Object*
- bool `isSaveMode` () const
- `ArnItem` & `setMaster` ()
 - Set client session sync mode as Master for this ArnItem.*
- bool `isMaster` () const

- [ArnItem](#) & [setAutoDestroy](#) ()
Set client session sync mode as AutoDestroy for this [ArnItem](#).
- bool [isAutoDestroy](#) () const
- void [setUncrossed](#) (bool [isUncrossed](#)=true)
Set a Bidirectional item as Uncrossed.
- bool [isUncrossed](#) () const
Get the Uncrossed state of an object.
- void [setBlockEcho](#) (bool [blockEcho](#)=true)
Control echo cancellation for this item.
- void [setDelay](#) (int [delay](#))
Set delay of data changed signal.
- int [delay](#) () const
Get delay of data changed signal.
- bool [isDelayPending](#) () const
- void [bypassDelayPending](#) ()
- void [arnImport](#) (const QByteArray &data, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Import data to an [Arn](#) Data Object
- QByteArray [arnExport](#) () const
- int [toInt](#) (bool *isOk=arnNullptr) const
- double [toDouble](#) (bool *isOk=arnNullptr) const
- [ARNREAL](#) [toReal](#) (bool *isOk=arnNullptr) const
- bool [toBool](#) (bool *isOk=arnNullptr) const
- QString [toString](#) (bool *isOk=arnNullptr) const
- QByteArray [toByteArray](#) (bool *isOk=arnNullptr) const
- QVariant [toVariant](#) (bool *isOk=arnNullptr) const
- uint [toUInt](#) (bool *isOk=arnNullptr) const
- qint64 [toInt64](#) (bool *isOk=arnNullptr) const
- quint64 [toUInt64](#) (bool *isOk=arnNullptr) const
- [ArnItem](#) & [operator=](#) (const [ArnItem](#) &other)
- [ArnItem](#) & [operator=](#) (int val)
- [ArnItem](#) & [operator=](#) ([ARNREAL](#) other)
- [ArnItem](#) & [operator=](#) (const QString &val)
- [ArnItem](#) & [operator=](#) (const QByteArray &val)
- [ArnItem](#) & [operator=](#) (const QVariant &val)
- [ArnItem](#) & [operator=](#) (const char *val)
- [ArnItem](#) & [operator=](#) (uint val)
- [ArnItem](#) & [operator=](#) (qint64 val)
- [ArnItem](#) & [operator=](#) (quint64 val)
- [ArnItem](#) & [operator+=](#) (int val)
- [ArnItem](#) & [operator+=](#) ([ARNREAL](#) val)
- void [setValue](#) (const [ArnItem](#) &other, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign the value of an other [ArnItem](#) to an [Arn](#) Data Object
- void [setValue](#) (uint value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an unsigned int to an [Arn](#) Data Object
- void [setValue](#) (qint64 value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an int 64 bit to an [Arn](#) Data Object
- void [setValue](#) (quint64 value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an unsigned int 64 bit to an [Arn](#) Data Object
- void [setValue](#) (int value, int ignoreSame)
Assign an integer to an [Arn](#) Data Object
- void [setValue](#) (double value, int ignoreSame)
Assign an ARNREAL to an [Arn](#) Data Object

- void `setValue` (bool value, int ignoreSame)
Assign a bool to an [Arn Data Object](#)
- void `setValue` (const QString &value, int ignoreSame)
Assign a QString to an [Arn Data Object](#)
- void `setValue` (const QByteArray &value, int ignoreSame)
Assign a QByteArray to an [Arn Data Object](#)
- void `setValue` (const QVariant &value, int ignoreSame)
Assign a QVariant to an [Arn Data Object](#)
- void `setValue` (const char *value, int ignoreSame)
Assign a char to an [Arn Data Object](#)*
- void `setBits` (int mask, int value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
AtomicOp assign an integer to specified bits in an [Arn Data Object](#)
- void `addValue` (int value)
AtomicOp adds an integer to an [Arn Data Object](#)
- void `addValue` ([ARNREAL](#) value)
AtomicOp adds an ARNREAL to an [Arn Data Object](#)

14.32.1 Detailed Description

Handle for an [Arn Data Object](#).

[About ArnItem access](#)

See [ArnBasicItem](#).

When opening an [ArnItem](#) to an [Arn Data object](#), the [ArnItem](#) act as a handle (pointer) to the object. There can be any amount of [ArnItem](#):s opened (pointing) to the same [Arn Data object](#). Deleting the [ArnItem](#) won't effect the [Arn Data object](#).

This class is not thread-safe, but the [Arn Data object](#) is, so each thread should have it's own handles i.e [ArnItem](#) instances.

Example usage

```
// In class declare
ArnItem _arnTime;

// In class code
_arnTime.open("//Chat/Time/value");
connect( &_arnTime, SIGNAL(changed(QString)), this, SLOT(doTimeUpdate(QString)));
_arnTime = "Undefined ...";
```

Examples:

[ArnDemoChat/MainWindow.hpp](#), [ArnDemoChatServer/MainWindow.cpp](#), and [ArnDemoChatServer/MainWindow.hpp](#).

Definition at line 72 of file [ArnItem.hpp](#).

14.32.2 Constructor & Destructor Documentation

14.32.2.1 ArnItem() [1/3]

```
ArnItem::ArnItem (
    QObject * parent = arnNullptr )
```

Standard constructor of a closed handle.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 109 of file ArnItem.cpp.

14.32.2.2 ArnItem() [2/3]

```
ArnItem::ArnItem (
    const QString & path,
    QObject * parent = arnNullptr )
```

Construction of a handle to a path.

Parameters

in	<i>path</i>	The <i>Arn Data Object</i> path e.g. <code>"//Measure/Water/Level/value"</code>
in	<i>parent</i>	

See also

[open\(\)](#)

Definition at line 116 of file ArnItem.cpp.

14.32.2.3 ArnItem() [3/3]

```
ArnItem::ArnItem (
    const ArnItem & itemTemplate,
    const QString & path,
    QObject * parent = arnNullptr )
```

Construction of a handle to a path with a template for *modes*

Parameters

in	<i>itemTemplate</i>	The template for setting <i>modes</i>
in	<i>path</i>	The <i>Arn Data Object</i> path e.g. <code>"//Measure/Water/Level/value"</code>
in	<i>parent</i>	

Definition at line 124 of file ArnItem.cpp.

14.32.2.4 ~ArnItem()

```
ArnItem::~~ArnItem ( ) [virtual]
```

Definition at line 545 of file ArnItem.cpp.

14.32.3 Member Function Documentation

14.32.3.1 addMode()

```
void ArnItem::addMode (
    Arn::ObjectMode mode ) [inline]
```

Add *general mode* settings for this *Arn Data Object*

If this *ArnItem* is in closed state, the added modes will be stored and the real mode change is done when this *ArnItem* is opened to an *Arn Data Object*. This implies that *ArnItems* can benefit from setting *modes* before opening.

Parameters

in	<i>mode</i>	The <i>modes</i> to be added.
----	-------------	-------------------------------

See also

[getMode\(\)](#)
[Modes](#)

Definition at line 159 of file ArnItem.hpp.

14.32.3.2 addValue() [1/2]

```
void ArnItem::addValue (
    int value ) [inline]
```

AtomicOp adds an *integer* to an *Arn Data Object*

Operation is done atomically. If *bidir*, it can also be done remotely by an *AtomicOpProvider*

Parameters

in	<i>value</i>	to be added to this <i>Arn Data Object</i>
----	--------------	--

See also

[setAtomicOpProvider\(\)](#)

Definition at line 548 of file ArnItem.hpp.

14.32.3.3 addValue() [2/2]

```
void ArnItem::addValue (
    ARNREAL value ) [inline]
```

AtomicOp adds an *ARNREAL* to an *Arn Data Object*

Operation is done atomically. If bidir, it can also be done remotely by an AtomicOpProvider

Parameters

in	value	to be added to this <i>Arn Data Object</i>
----	-------	--

See also

[setAtomicOpProvider\(\)](#)

Definition at line 557 of file ArnItem.hpp.

14.32.3.4 arnExport()

```
QByteArray ArnItem::arnExport ( ) const [inline]
```

Returns

A data blob representing the *Arn Data Object*

See also

[arnImport\(\)](#)

Definition at line 345 of file ArnItem.hpp.

14.32.3.5 arnImport()

```
void ArnItem::arnImport (
    const QByteArray & data,
    int ignoreSame = Arn::SameValue::DefaultAction ) [inline]
```

Import data to an *Arn Data Object*

Data blob from a previous `arnExport ()` can be imported. This is essentially assigning the *Arn Data Object* with same as exported.

Parameters

in	<i>data</i>	is the data blob
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[arnExport\(\)](#)
[setIgnoreSameValue\(\)](#)

Definition at line 339 of file ArnItem.hpp.

14.32.3.6 arnItemCreated

```
void ArnItem::arnItemCreated (
    const QString & path ) [signal]
```

Signal emitted when an [Arn Data Object](#) is created in the tree below.

The [ArnItem](#) is a folder. Created objects in this folder or its children will give this signal. Only created non folder objects will give this signal.

Parameters

in	<i>path</i>	to the created Arn Data Object
----	-------------	--

Deprecated use [ArnMonitor](#) instead.

14.32.3.7 arnModeChanged

```
void ArnItem::arnModeChanged (
    const QString & path,
    uint linkId,
    Arn::ObjectMode mode ) [signal]
```

Signal emitted when an [Arn Data Object](#) in the tree below has a *general mode* change.

The [ArnItem](#) is a folder. Objects changing *general mode* in this folder or its children will give this signal.

Parameters

in	<i>path</i>	to the <i>general mode</i> changing Arn Data Object
in	<i>linkId</i>	for the <i>general mode</i> changing Arn Data Object
in	<i>mode</i>	is the new <i>general mode</i>

See also

[linkId\(\)](#)
[Modes](#)

Deprecated use [ArnMonitor](#) instead.

14.32.3.8 bypassDelayPending()

```
void ArnItem::bypassDelayPending ( )
```

For delay pending, immediately signal changed If the changed signal is pending in a delay, the changed signal is immediately emitted and the delay is canceled. Otherwise nothing is done.

See also

[setDelay\(\)](#)
[isDelayPending\(\)](#)

Definition at line 228 of file ArnItem.cpp.

14.32.3.9 changed [1/7]

```
void ArnItem::changed ( ) [signal]
```

Signals emitted when data in [Arn Data Object](#) is changed.

Only the connected (used) signals are emitted for efficiency. When using pipes with queued connection to a slot, it's strongly advised to use the signal that carries the updated data. Otherwise some stream data can be lost and other will be doubled, because reading is done late in the slot.

changed(...) is using connectNotify & disconnectNotify. Must be updated if new types are added

See also

[setIgnoreSameValue\(\)](#)

14.32.3.10 changed [2/7]

```
void ArnItem::changed (
    int value ) [signal]
```

See also

[changed\(\)](#)

14.32.3.11 **changed** [3/7]

```
void ArnItem::changed (
    double value ) [signal]
```

See also[changed\(\)](#)**14.32.3.12** **changed** [4/7]

```
void ArnItem::changed (
    bool value ) [signal]
```

See also[changed\(\)](#)**14.32.3.13** **changed** [5/7]

```
void ArnItem::changed (
    const QString & value ) [signal]
```

See also[changed\(\)](#)**14.32.3.14** **changed** [6/7]

```
void ArnItem::changed (
    const QByteArray & value ) [signal]
```

See also[changed\(\)](#)

14.32.3.15 `changed` [7/7]

```
void ArnItem::changed (
    const QVariant & value ) [signal]
```

See also

[changed\(\)](#)

14.32.3.16 `delay()`

```
int ArnItem::delay ( ) const
```

Get *delay* of data changed signal.

Read current value of the delay.

Returns

the delay in ms.

See also

[setDelay\(\)](#)

Definition at line 210 of file ArnItem.cpp.

14.32.3.17 `getMode()`

```
Arn::ObjectMode ArnItem::getMode ( ) const [inline]
```

Returns

The *general mode* of the *Arn Data Object*

See also

[addMode\(\)](#)
[Modes](#)

Definition at line 166 of file ArnItem.hpp.

14.32.3.18 `isAutoDestroy()`

```
bool ArnItem::isAutoDestroy ( ) const [inline]
```

Return values

<i>true</i>	if <i>AutoDestroy mode</i>
-------------	----------------------------

See also[setAutoDestroy\(\)](#)

Definition at line 264 of file ArnItem.hpp.

14.32.3.19 isBiDirMode()

```
bool ArnItem::isBiDirMode ( ) const [inline]
```

Return values

<i>true</i>	if Bidirectional
-------------	------------------

See also[setBiDirMode\(\)](#)[Modes](#)[Bidirectional Arn Data Objects](#)

Definition at line 203 of file ArnItem.hpp.

14.32.3.20 isDelayPending()

```
bool ArnItem::isDelayPending ( ) const
```

Delay pending status

Return values

<i>true</i>	if the <i>Arn Data Object</i> is changed, but the changed signal is pending in a delay.
-------------	---

See also[setDelay\(\)](#)[bypassDelayPending\(\)](#)

Definition at line 220 of file ArnItem.cpp.

14.32.3.21 isFolder()

```
bool ArnItem::isFolder ( ) const [inline]
```

Return values

<i>true</i>	if this ArnItem is a folder
-------------	---

Definition at line 122 of file ArnItem.hpp.

14.32.3.22 isIgnoreSameValue()

```
bool ArnItem::isIgnoreSameValue ( ) [inline]
```

Return values

<i>true</i>	if skipping equal values
-------------	--------------------------

See also

[setIgnoreSameValue\(\)](#)

Definition at line 147 of file ArnItem.hpp.

14.32.3.23 isMaster()

```
bool ArnItem::isMaster ( ) const [inline]
```

Return values

<i>true</i>	if <i>Master mode</i>
-------------	-----------------------

See also

[setMaster\(\)](#)
[Modes](#)

Definition at line 251 of file ArnItem.hpp.

14.32.3.24 isPipeMode()

```
bool ArnItem::isPipeMode ( ) const [inline]
```

Return values

<i>true</i>	if <i>Pipe mode</i>
-------------	---------------------

See also[setPipeMode\(\)](#)[Modes](#)[Pipe Arn Data Objects](#)

Definition at line 219 of file ArnItem.hpp.

14.32.3.25 isProvider()

```
bool ArnItem::isProvider ( ) const [inline]
```

Return values

<i>true</i>	if this ArnItem is a provider
-------------	---

See also[setBiDirMode\(\)](#)[Modes](#)

Definition at line 129 of file ArnItem.hpp.

14.32.3.26 isSaveMode()

```
bool ArnItem::isSaveMode ( ) const [inline]
```

Return values

<i>true</i>	if <i>Save mode</i>
-------------	---------------------

See also[setSaveMode\(\)](#)[Modes](#)[Persistent Arn Data Objects](#)

Definition at line 236 of file ArnItem.hpp.

14.32.3.27 isTemplate()

```
bool ArnItem::isTemplate ( ) const
```

Return values

<i>true</i>	if this is a template
-------------	-----------------------

See also

[setTemplate\(\)](#)

Definition at line 191 of file ArnItem.cpp.

14.32.3.28 isUncrossed()

```
bool ArnItem::isUncrossed ( ) const [inline]
```

Get the Uncrossed state of an object.

Return values

<i>true</i>	if Uncrossed is set or <i>Arn Data Object</i> is not in Bidirectional mode.
-------------	---

See also

[setUncrossed\(\)](#)

[setBiDirMode\(\)](#)

[Modes](#)

[Bidirectional Arn Data Objects](#)

Definition at line 285 of file ArnItem.hpp.

14.32.3.29 modeChanged

```
void ArnItem::modeChanged (
    Arn::ObjectMode mode ) [signal]
```

Signal emitted when mode in *Arn Data Object* is changed.

Object changing *general mode* will give this signal.

Parameters

<i>in</i>	<i>mode</i>	is the new <i>general mode</i>
-----------	-------------	--------------------------------

See also

[Modes](#)

14.32.3.30 openFolder()

```
bool ArnItem::openFolder (
    const QString & path ) [inline]
```

Open a handle to an [Arn](#) folder.

Parameters

in	<i>path</i>	The Arn folder path e.g. "//Measure/Water" (the / is appended)
----	-------------	--

Return values

<i>false</i>	if error
--------------	----------

Definition at line 117 of file ArnItem.hpp.

14.32.3.31 openUuid()

```
bool ArnItem::openUuid (
    const QString & path ) [inline]
```

Open a handle to an [Arn](#) Object with a unique uuid name.

Parameters

in	<i>path</i>	The prefix for Arn uuid path e.g. "//Names/name"
----	-------------	--

Return values

<i>false</i>	if error
--------------	----------

Definition at line 103 of file ArnItem.hpp.

14.32.3.32 openUuidPipe()

```
bool ArnItem::openUuidPipe (
    const QString & path ) [inline]
```

Open a handle to an [Arn](#) Pipe Object with a unique uuid name.

Parameters

<code>in</code>	<code>path</code>	The prefix for <code>Arn</code> uuid pipe path e.g. <code>"/Pipes/pipe"</code>
-----------------	-------------------	--

Return values

<code>false</code>	if error
--------------------	----------

Definition at line 110 of file `ArnItem.hpp`.

14.32.3.33 operator+=() [1/2]

```
ArnItem & ArnItem::operator+= (
    int val )
```

Definition at line 306 of file `ArnItem.cpp`.

14.32.3.34 operator+=() [2/2]

```
ArnItem & ArnItem::operator+= (
    ARNREAL val )
```

Definition at line 313 of file `ArnItem.cpp`.

14.32.3.35 operator=() [1/10]

```
ArnItem & ArnItem::operator= (
    const ArnItem & other )
```

Definition at line 236 of file `ArnItem.cpp`.

14.32.3.36 operator=() [2/10]

```
ArnItem & ArnItem::operator= (
    int val )
```

Definition at line 243 of file `ArnItem.cpp`.

14.32.3.37 operator=() [3/10]

```
ArnItem & ArnItem::operator= (
    ARNREAL other )
```

Definition at line 250 of file ArnItem.cpp.

14.32.3.38 operator=() [4/10]

```
ArnItem & ArnItem::operator= (
    const QString & val )
```

Definition at line 257 of file ArnItem.cpp.

14.32.3.39 operator=() [5/10]

```
ArnItem & ArnItem::operator= (
    const QByteArray & val )
```

Definition at line 264 of file ArnItem.cpp.

14.32.3.40 operator=() [6/10]

```
ArnItem & ArnItem::operator= (
    const QVariant & val )
```

Definition at line 299 of file ArnItem.cpp.

14.32.3.41 operator=() [7/10]

```
ArnItem & ArnItem::operator= (
    const char * val )
```

Definition at line 271 of file ArnItem.cpp.

14.32.3.42 operator=() [8/10]

```
ArnItem & ArnItem::operator= (
    uint val )
```

Definition at line 278 of file ArnItem.cpp.

14.32.3.43 operator=() [9/10]

```
ArnItem & ArnItem::operator= (
    quint64 val )
```

Definition at line 285 of file ArnItem.cpp.

14.32.3.44 operator=() [10/10]

```
ArnItem & ArnItem::operator= (
    quint64 val )
```

Definition at line 292 of file ArnItem.cpp.

14.32.3.45 setAutoDestroy()

```
ArnItem& ArnItem::setAutoDestroy ( ) [inline]
```

Set client session *sync mode* as *AutoDestroy* for this [ArnItem](#).

This [ArnItem](#) at client side is setup for auto destruction.

Precondition

This must be set before [open\(\)](#).

Definition at line 258 of file ArnItem.hpp.

14.32.3.46 setBiDirMode()

```
ArnItem& ArnItem::setBiDirMode ( ) [inline]
```

Set *general mode* as *Bidirectional* for this *Arn Data Object*

A two way object, typically for validation or pipe

See also

[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 195 of file ArnItem.hpp.

14.32.3.47 setBits()

```
void ArnItem::setBits (
    int mask,
    int value,
    int ignoreSame = Arn::SameValue::DefaultAction ) [inline]
```

AtomicOp assign an *integer* to specified bits in an *Arn Data Object*

Operation is done atomicly. If *bidir*, it can also be done remotely by an *AtomicOpProvider*

Parameters

in	<i>mask</i>	to specify bits that is affected
in	<i>value</i>	to be assigned to affected bits
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setAtomicOpProvider\(\)](#)
[setIgnoreSameValue\(\)](#)

Definition at line 539 of file ArnItem.hpp.

14.32.3.48 setBlockEcho()

```
void ArnItem::setBlockEcho (
    bool blockEcho = true ) [inline]
```

Control echo cancellation for this item.

When an ArnObject is changed via this item, the [changed\(\)](#) signal on this item can be blocked.

Parameters

in	<i>blockEcho</i>	if true echo is blocked.
----	------------------	--------------------------

Definition at line 293 of file ArnItem.hpp.

14.32.3.49 setDelay()

```
void ArnItem::setDelay (
    int delay )
```

Set *delay* of data changed signal.

Normally any change of the *Arn Data Object* is immediately signalled. By setting this *delay*, intensive updates gives predictive and fewer signals. Signalling will not be faster than *delay* as period time. The latency from a change to a signal will not be more than *delay*.

Parameters

in	<i>delay</i>	in ms.
----	--------------	--------

See also

[delay\(\)](#)
[isDelayPending\(\)](#)
[bypassDelayPending\(\)](#)

Definition at line 199 of file ArnItem.cpp.

14.32.3.50 setIgnoreSameValue()

```
void ArnItem::setIgnoreSameValue (
    bool isIgnore = true ) [inline]
```

Set skipping assignment of equal value.

Parameters

<code>in</code>	<code><i>isIgnore</i></code>	If true, assignment of equal value don't give a changed signal.
-----------------	------------------------------	---

Definition at line 141 of file ArnItem.hpp.

14.32.3.51 setMaster()

```
ArnItem& ArnItem::setMaster ( ) [inline]
```

Set client session *sync mode* as *Master* for this [ArnItem](#).

This [ArnItem](#) at client side is set as default generator of data.

Precondition

This must be set before [open\(\)](#).

See also

[Modes](#)

Definition at line 244 of file ArnItem.hpp.

14.32.3.52 setPipeMode()

```
ArnItem& ArnItem::setPipeMode ( ) [inline]
```

Set *general mode* as Pipe for this [Arn Data Object](#)

Implies *Bidir*.

See also

[Modes](#)
[Pipe Arn Data Objects](#)

Definition at line 211 of file ArnItem.hpp.

14.32.3.53 setSaveMode()

```
ArnItem& ArnItem::setSaveMode ( ) [inline]
```

Set *general mode* as Save for this [Arn Data Object](#)

Data is persistent and will be saved

Precondition

The persistent service must be started at the server.

See also

[Modes](#)
[Persistent Arn Data Objects](#)

Definition at line 228 of file ArnItem.hpp.

14.32.3.54 setTemplate()

```
ArnItem & ArnItem::setTemplate (
    bool isTemplate = true )
```

Mark this [ArnItem](#) as a template.

When marked as a template it can be setup with a combination of *modes* which are used for other ArnItems using this template. The effected *modes* can be both *general modes* and *sync modes*.

Parameters

in	<i>isTemplate</i>	True for template mode.
----	-------------------	-------------------------

See also

[open\(\)](#)
[Modes](#)

Definition at line 182 of file ArnItem.cpp.

14.32.3.55 setUncrossed()

```
void ArnItem::setUncrossed (
    bool isUncrossed = true ) [inline]
```

Set a Bidirectional item as Uncrossed.

The two way object is not twisted at writes, i.e. exactly the same object is read and written. This has no effect on an *Arn Data Object* that not is in Bidirectional mode.

See also

[isUncrossed\(\)](#)
[Modes](#)
[Bidirectional Arn Data Objects](#)

Definition at line 275 of file ArnItem.hpp.

14.32.3.56 setValue() [1/18]

```
void ArnItem::setValue (
    const ArnItem & other,
    int ignoreSame = Arn::SameValue::DefaultAction ) [inline]
```

Assign the value of an other *ArnItem* to an *Arn Data Object*

Parameters

in	<i>other</i>	is the <i>ArnItem</i> containing the value to assign
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 440 of file ArnItem.hpp.

14.32.3.57 setValue() [2/18]

```
void ArnItem::setValue (
    uint value,
    int ignoreSame = Arn::SameValue::DefaultAction ) [inline]
```

Assign an *unsigned int* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 449 of file ArnItem.hpp.

14.32.3.58 setValue() [3/18]

```
void ArnItem::setValue (
    qint64 value,
    int ignoreSame = Arn::SameValue::DefaultAction ) [inline]
```

Assign an *int 64 bit* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 458 of file ArnItem.hpp.

14.32.3.59 setValue() [4/18]

```
void ArnItem::setValue (
    quint64 value,
    int ignoreSame = Arn::SameValue::DefaultAction ) [inline]
```

Assign an *unsigned int 64 bit* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 467 of file ArnItem.hpp.

14.32.3.60 setValue() [5/18]

```
void ArnItem::setValue (
    int value,
    int ignoreSame ) [inline]
```

Assign an *integer* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 475 of file ArnItem.hpp.

14.32.3.61 setValue() [6/18]

```
void ArnItem::setValue (
    double value,
    int ignoreSame ) [inline]
```

Assign an *ARNREAL* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 486 of file ArnItem.hpp.

14.32.3.62 setValue() [7/18]

```
void ArnItem::setValue (
    bool value,
    int ignoreSame ) [inline]
```

Assign a *bool* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 495 of file ArnItem.hpp.

14.32.3.63 setValue() [8/18]

```
void ArnItem::setValue (
    const QString & value,
    int ignoreSame ) [inline]
```

Assign a *QString* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 503 of file ArnItem.hpp.

14.32.3.64 setValue() [9/18]

```
void ArnItem::setValue (
    const QByteArray & value,
    int ignoreSame ) [inline]
```

Assign a *QByteArray* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 511 of file ArnItem.hpp.

14.32.3.65 setValue() [10/18]

```
void ArnItem::setValue (
    const QVariant & value,
    int ignoreSame ) [inline]
```

Assign a *QVariant* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 519 of file ArnItem.hpp.

14.32.3.66 setValue() [11/18]

```
void ArnItem::setValue (
    const char * value,
    int ignoreSame ) [inline]
```

Assign a *char** to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
in	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 527 of file ArnItem.hpp.

14.32.3.67 setValue [12/18]

```
void ArnItem::setValue (
    int value ) [inline], [slot]
```

Assign an *integer* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

See also

[setIgnoreSameValue\(\)](#)

Definition at line 565 of file ArnItem.hpp.

14.32.3.68 setValue [13/18]

```
void ArnItem::setValue (
    double value ) [inline], [slot]
```

Assign an *ARNREAL* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

See also[setIgnoreSameValue\(\)](#)

Definition at line 575 of file ArnItem.hpp.

14.32.3.69 setValue [14/18]

```
void ArnItem::setValue (  
    bool value ) [inline], [slot]
```

Assign a *bool* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

See also[setIgnoreSameValue\(\)](#)

Definition at line 583 of file ArnItem.hpp.

14.32.3.70 setValue [15/18]

```
void ArnItem::setValue (  
    const QString & value ) [inline], [slot]
```

Assign a *QString* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

See also[setIgnoreSameValue\(\)](#)

Definition at line 590 of file ArnItem.hpp.

14.32.3.71 setValue [16/18]

```
void ArnItem::setValue (
    const QByteArray & value ) [inline], [slot]
```

Assign a *QByteArray* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

See also

[setIgnoreSameValue\(\)](#)

Definition at line 597 of file ArnItem.hpp.

14.32.3.72 setValue [17/18]

```
void ArnItem::setValue (
    const QVariant & value ) [inline], [slot]
```

Assign a *QVariant* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

See also

[setIgnoreSameValue\(\)](#)

Definition at line 604 of file ArnItem.hpp.

14.32.3.73 setValue [18/18]

```
void ArnItem::setValue (
    const char * value ) [inline], [slot]
```

Assign a *char** to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

See also

[setIgnoreSameValue\(\)](#)

Definition at line 611 of file ArnItem.hpp.

14.32.3.74 syncMode()

```
Arn::ObjectSyncMode ArnItem::syncMode ( ) const [inline]
```

Returns

The client session *sync mode* of an *Arn Data Object*

See also

[Modes](#)

Definition at line 172 of file ArnItem.hpp.

14.32.3.75 toBool()

```
bool ArnItem::toBool (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to a *bool*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	-------------	---

Definition at line 373 of file ArnItem.hpp.

14.32.3.76 toByteArray()

```
QByteArray ArnItem::toByteArray (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to a *QByteArray*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	-------------	---

Definition at line 387 of file ArnItem.hpp.

14.32.3.77 toDouble()

```
double ArnItem::toDouble (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to a *double*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	-------------	---

Definition at line 359 of file ArnItem.hpp.

14.32.3.78 toggleBool

```
void ArnItem::toggleBool ( ) [slot]
```

Toggle the *bool* at the *Arn Data Object*

The *Arn Data Object* is first converted to a *bool*, then the toggled value is assigned back to the *Arn Data Object*.

Definition at line 320 of file ArnItem.cpp.

14.32.3.79 toInt()

```
int ArnItem::toInt (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to a *integer*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	-------------	---

Definition at line 352 of file ArnItem.hpp.

14.32.3.80 toInt64()

```
qint64 ArnItem::toInt64 (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to an *int 64 bit*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	-------------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 410 of file ArnItem.hpp.

14.32.3.81 toReal()

```
ARNREAL ArnItem::toReal (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to an *ARNREAL*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	-------------	---

Definition at line 366 of file ArnItem.hpp.

14.32.3.82 toString()

```
QString ArnItem::toString (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to a *QString*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Definition at line 380 of file ArnItem.hpp.

14.32.3.83 toUInt()

```
uint ArnItem::toUInt (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to an *unsigned int*

Parameters

out	isOk	If not 0 when a conversion error occurs, *isOk is set to false, otherwise *isOk is set to true.
-----	------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 402 of file ArnItem.hpp.

14.32.3.84 toUInt64()

```
quint64 ArnItem::toUInt64 (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to an *unsigned int 64 bit*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, <i>*isOk</i> is set to false, otherwise <i>*isOk</i> is set to true.
-----	-------------	---

Note

Not native ARN datatype. It's converted from ByteArray.

Definition at line 418 of file ArnItem.hpp.

14.32.3.85 toVariant()

```
QVariant ArnItem::toVariant (
    bool * isOk = arnNullptr ) const [inline]
```

Returns

Convert *Arn Data Object* to a *QVariant*

Parameters

out	<i>isOk</i>	If not 0 when a conversion error occurs, <i>*isOk</i> is set to false, otherwise <i>*isOk</i> is set to true.
-----	-------------	---

Definition at line 394 of file ArnItem.hpp.

14.32.3.86 type()

```
Arn::DataType ArnItem::type ( ) const [inline]
```

The type stored in the *Arn Data Object*

Returns

The type stored

Definition at line 135 of file ArnItem.hpp.

The documentation for this class was generated from the following files:

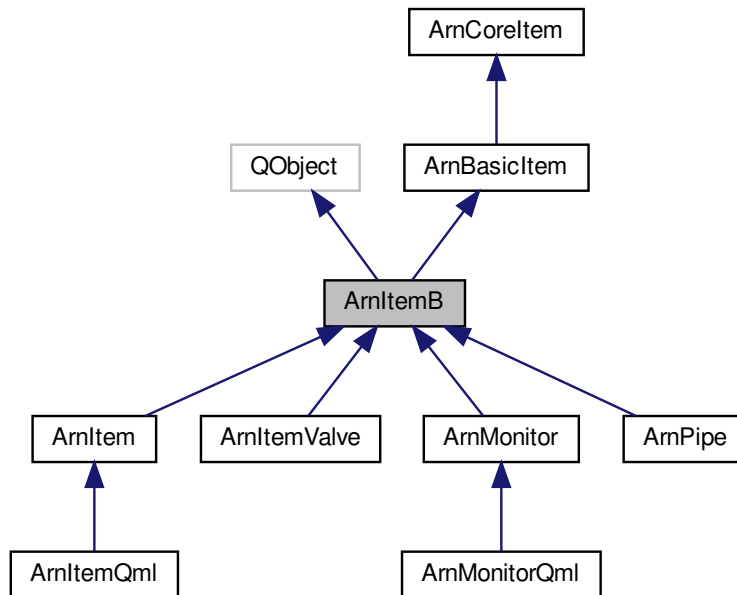
- [src/ArnInc/ArnItem.hpp \(4.0.0\)](#)
- [src/ArnItem.cpp \(4.0.0\)](#)

14.33 ArnItemB Class Reference

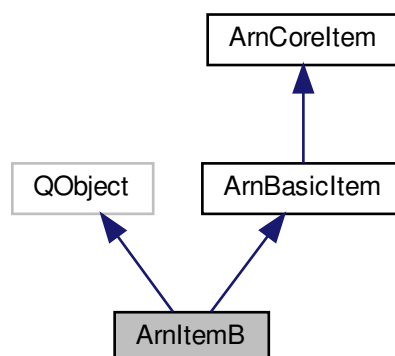
Base class handle for an *Arn Data Object*.

```
#include <ArnItemB.hpp>
```

Inheritance diagram for ArnItemB:



Collaboration diagram for ArnItemB:



Signals

- void [arnLinkDestroyed](#) ()
Signal emitted when the [Arn Data Object](#) is destroyed.

Public Member Functions

- [ArnItemB](#) (QObject *parent=arnNullptr)
Standard constructor of a closed handle.
- virtual [~ArnItemB](#) ()
- bool [open](#) (const QString &path)
Open a handle to an [Arn Data Object](#)

14.33.1 Detailed Description

Base class handle for an [Arn Data Object](#).

About Arn Data Object

This class contains the basic services, that should be appropriate for any derived class as public methods. Other non generic services that might be needed is available as protected methods. Typically derived classes can select among these protected methods and make any of them public.

See [ArnItem](#).

Definition at line 59 of file ArnItemB.hpp.

14.33.2 Constructor & Destructor Documentation

14.33.2.1 ArnItemB()

```
ArnItemB::ArnItemB (
    QObject * parent = arnNullptr )
```

Standard constructor of a closed handle.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 61 of file ArnItemB.cpp.

14.33.2.2 ~ArnItemB()

```
ArnItemB::~~ArnItemB ( ) [virtual]
```

Definition at line 77 of file ArnItemB.cpp.

14.33.3 Member Function Documentation

14.33.3.1 arnLinkDestroyed

```
void ArnItemB::arnLinkDestroyed ( ) [signal]
```

Signal emitted when the *Arn Data Object* is destroyed.

When the link (*Arn Data Object*) is destroyed, this *ArnItem* is closed and will give this signal. It's ok to assign values etc to a closed *ArnItem*, it's thrown away like a null device.

See also

[destroyLink\(\)](#)

14.33.3.2 open()

```
bool ArnItemB::open (
    const QString & path )
```

Open a handle to an *Arn Data Object*

Parameters

in	<i>path</i>	The <i>Arn Data Object</i> path e.g. "//Measure/Water/Level/value"
----	-------------	--

Return values

<i>false</i>	if error
--------------	----------

Definition at line 91 of file ArnItemB.cpp.

The documentation for this class was generated from the following files:

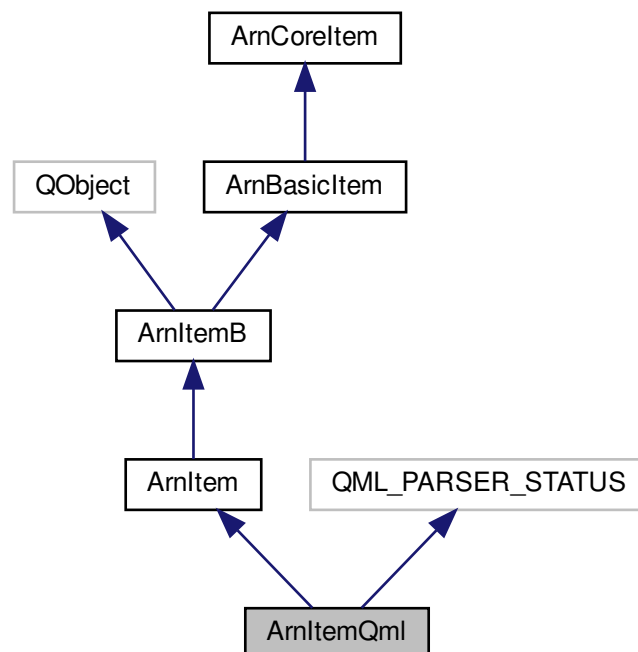
- [src/ArnInc/ArnItemB.hpp \(4.0.0\)](#)
- [src/ArnItemB.cpp \(4.0.0\)](#)

14.34 ArnItemQml Class Reference

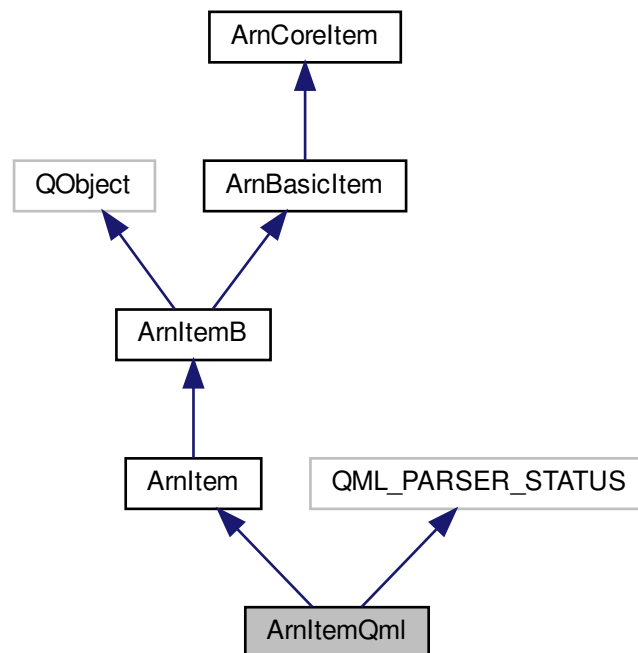
ARN Item QML.

```
#include <ArnQml.hpp>
```

Inheritance diagram for ArnItemQml:



Collaboration diagram for ArnItemQml:



Public Slots

- void [setBits](#) (int mask, int value)
AtomicOp assign an integer to specified bits in an Arn Data Object
- void [addIntNum](#) (int value)
AtomicOp adds an integer to an Arn Data Object
- void [addNum](#) (double value)
AtomicOp adds an ARNREAL to an Arn Data Object
- void [addMode](#) ([ArnInterface::ObjectMode](#) mode)
Add general mode settings for this Arn Data Object
- [ArnInterface::ObjectMode](#) [getMode](#) () const

Properties

- QString [variantType](#)
The type used inside the variant, e.g. QString.
- bool [useUuid](#)
Select to use ArnItem::openUuid()
- QString [path](#)
The path of this ArnItem.
- [ArnInterface::DataType](#) type
The Arn data type of this ArnItem.

- QVariant [variant](#)
The *ArnItem* value as a *QVariant*.
- QString [string](#)
The *ArnItem* value as a *QString*.
- QByteArray [bytes](#)
The *ArnItem* value as a *QByteArray*.
- double [num](#)
The *ArnItem* value as an *ARNREAL*.
- int [intNum](#)
The *ArnItem* value as an *int*.
- bool [biDirMode](#)
See *Arn::ObjectMode::BiDir*.
- bool [pipeMode](#)
See *Arn::ObjectMode::Pipe*.
- bool [saveMode](#)
See *Arn::ObjectMode::Save*.
- bool [masterMode](#)
See *Arn::ObjectSyncMode::Master*.
- bool [autoDestroyMode](#)
See *Arn::ObjectSyncMode::AutoDestroy*.
- bool [atomicOpProvider](#)
See *ArnBasicItem::setAtomicOpProvider()*
- bool [ignoreSameValue](#)
See *ArnItem::setIgnoreSameValue()*
- int [delay](#)
See *ArnItem::setDelay()*

Additional Inherited Members

14.34.1 Detailed Description

ARN Item QML.

This class is the Qml version of [ArnItem](#).

See also

[ArnQml](#)

Example usage

```
// In Qml
//
import QtQuick 2.0
import ArnLib 1.0

Rectangle {
    width: 370; height: 400

    property ArnItem arnT1: ArnItem {path: "//E1/UpdClock/value"}

    ArnItem {id: arnE1UpdClock; path: "//E1/UpdClock/value"}
    ArnItem {id: arnTest; path: "//Test/test"}
```

```
Rectangle {
    id: info
    anchors.bottom: parent.bottom; anchors.left: parent.left; anchors.right: parent.right
    height: 80
    Column {
        anchors.fill: parent;
        Text {text: "El updClock 1: " + arnElUpdClock.intNum}
        Text {text: "El updClock 2: " + arnTl.intNum}
    }
}

Component.onCompleted: {
    arnTest.setValue("Start ...", Arn.SameValue_Accept);
}
}
```

Definition at line 286 of file ArnQml.hpp.

14.34.2 Member Function Documentation

14.34.2.1 addIntNum

```
void ArnItemQml::addIntNum (
    int value ) [inline], [slot]
```

AtomicOp adds an *integer* to an *Arn Data Object*

See also

[ArnItem::addValue\(\)](#)

Definition at line 348 of file ArnQml.hpp.

14.34.2.2 addMode

```
void ArnItemQml::addMode (
    ArnInterface::ObjectMode mode ) [inline], [slot]
```

Add *general mode* settings for this *Arn Data Object*

See also

[ArnItem::addMode\(\)](#)

Definition at line 364 of file ArnQml.hpp.

14.34.2.3 addNum

```
void ArnItemQml::addNum (
    double value ) [inline], [slot]
```

AtomicOp adds an *ARNREAL* to an *Arn Data Object*

See also

[ArnItem::addValue\(\)](#)

Definition at line 357 of file ArnQml.hpp.

14.34.2.4 getMode

```
ArnInterface::ObjectMode ArnItemQml::getMode ( ) const [inline], [slot]
```

Returns

The *general mode* of the *Arn Data Object*

See also

[ArnItem::getMode\(\)](#)

Definition at line 370 of file ArnQml.hpp.

14.34.2.5 setBits

```
void ArnItemQml::setBits (
    int mask,
    int value ) [inline], [slot]
```

AtomicOp assign an *integer* to specified bits in an *Arn Data Object*

See also

[ArnItem::setBits\(\)](#)

Definition at line 342 of file ArnQml.hpp.

14.34.3 Property Documentation

14.34.3.1 atomicOpProvider

`bool ArnItemQml::atomicOpProvider [read], [write]`

See [ArnBasicItem::setAtomicOpProvider\(\)](#)

Definition at line 331 of file ArnQml.hpp.

14.34.3.2 autoDestroyMode

`bool ArnItemQml::autoDestroyMode [read], [write]`

See [Arn::ObjectSyncMode::AutoDestroy](#).

Definition at line 328 of file ArnQml.hpp.

14.34.3.3 biDirMode

`bool ArnItemQml::biDirMode [read], [write]`

See [Arn::ObjectMode::BiDir](#).

Definition at line 320 of file ArnQml.hpp.

14.34.3.4 bytes

`QByteArray ArnItemQml::bytes [read], [write]`

The [ArnItem](#) value as a QByteArray.

Definition at line 310 of file ArnQml.hpp.

14.34.3.5 delay

`int ArnItemQml::delay [read], [write]`

See [ArnItem::setDelay\(\)](#)

Definition at line 335 of file ArnQml.hpp.

14.34.3.6 ignoreSameValue

```
bool ArnItemQml::ignoreSameValue [read], [write]
```

See [ArnItem::setIgnoreSameValue\(\)](#)

Definition at line 333 of file ArnQml.hpp.

14.34.3.7 intNum

```
int ArnItemQml::intNum [read], [write]
```

The [ArnItem](#) value as an int.

Definition at line 318 of file ArnQml.hpp.

14.34.3.8 masterMode

```
bool ArnItemQml::masterMode [read], [write]
```

See [Arn::ObjectSyncMode::Master](#).

Definition at line 326 of file ArnQml.hpp.

14.34.3.9 num

```
double ArnItemQml::num [read], [write]
```

The [ArnItem](#) value as an ARNREAL.

Definition at line 315 of file ArnQml.hpp.

14.34.3.10 path

```
QString ArnItemQml::path [read], [write]
```

The path of this [ArnItem](#).

Definition at line 301 of file ArnQml.hpp.

14.34.3.11 pipeMode

```
bool ArnItemQml::pipeMode [read], [write]
```

See [Arn::ObjectMode::Pipe](#).

Definition at line 322 of file ArnQml.hpp.

14.34.3.12 saveMode

```
bool ArnItemQml::saveMode [read], [write]
```

See [Arn::ObjectMode::Save](#).

Definition at line 324 of file ArnQml.hpp.

14.34.3.13 string

```
QString ArnItemQml::string [read], [write]
```

The [ArnItem](#) value as a QString.

Definition at line 308 of file ArnQml.hpp.

14.34.3.14 type

```
ArnInterface::DataType ArnItemQml::type [read]
```

The [Arn](#) data type of this [ArnItem](#).

Definition at line 304 of file ArnQml.hpp.

14.34.3.15 useUuid

```
bool ArnItemQml::useUuid [read], [write]
```

Select to use [ArnItem::openUuid\(\)](#)

Definition at line 299 of file ArnQml.hpp.

14.34.3.16 variant

```
QVariant ArnItemQml::variant [read], [write]
```

The [ArnItem](#) value as a QVariant.

Definition at line 306 of file ArnQml.hpp.

14.34.3.17 variantType

```
QString ArnItemQml::variantType [read], [write]
```

The type used inside the variant, e.g. QString.

Definition at line 297 of file ArnQml.hpp.

The documentation for this class was generated from the following files:

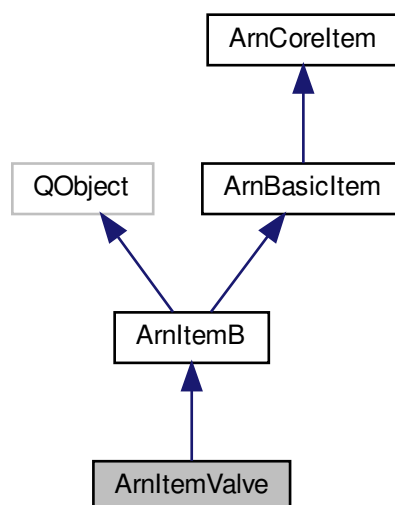
- [src/ArnInc/ArnQml.hpp \(4.0.0\)](#)
- [src/ArnQml.cpp \(4.0.0\)](#)

14.35 ArnItemValve Class Reference

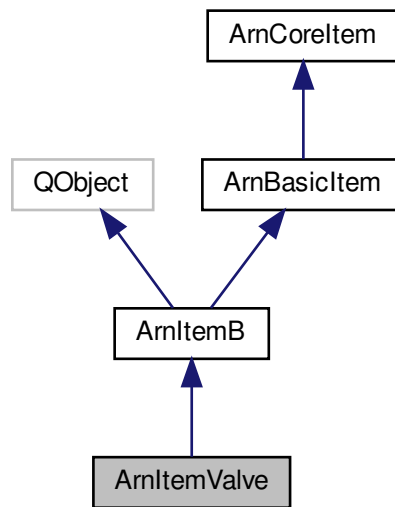
Valve for controlling stream to/from an [ArnItemB](#).

```
#include <ArnItemValve.hpp>
```

Inheritance diagram for ArnItemValve:



Collaboration diagram for ArnItemValve:



Classes

- struct [SwitchMode](#)

Public Slots

- void [setValue](#) (bool value)
Assign a bool to an [Arn](#) Data Object

Signals

- void [changed](#) (int value)

Public Member Functions

- [ArnItemValve](#) (QObject *parent=arnNullptr)
- bool [setTarget](#) ([ArnItemB](#) *targetItem, [SwitchMode](#) mode=[SwitchMode::InOutStream](#))
- [SwitchMode](#) [switchMode](#) () const
- [ArnItemValve](#) & [setSaveMode](#) ()
Set general mode as Save for this [Arn](#) Data Object
- bool [isSaveMode](#) () const
- [ArnItemValve](#) & [setMaster](#) ()
Set client session sync mode as Master for this [ArnItem](#).
- bool [isMaster](#) () const
- [ArnItemValve](#) & [setAutoDestroy](#) ()
Set client session sync mode as AutoDestroy for this [ArnItem](#).
- bool [isAutoDestroy](#) () const
- bool [toBool](#) () const
- [ArnItemValve](#) & [operator=](#) (bool value)

14.35.1 Detailed Description

Valve for controlling stream to/from an [ArnItemB](#).

About Arn Data Object

This valve class can control data stream to/from any [ArnItemB](#) derived class. The class itself is derived from [ArnItemB](#), so it could also be controlled by another [ArnItemValve](#). But most important, it has a subset of [ArnItem](#)'s methods to make it shareable in the ARN tree.

[ArnItemValve](#) can be used "standalone", i.e. not being opened to the ARN tree. In this case it is used by its `setValue` method and locally emits its `changed()` signal.

When opened to the ARN tree it can be used by its `setValue` method and it can also be remote controlled as any other [ArnItem](#). If locally set, this will as usual be reflected in the ARN tree.

It's possible to use one [ArnItemValve](#) for controlling *InStream* and another for controlling *OutStream*. The valve for each stream direction can then be set independently. The default is using one valve for both stream directions.

This class is not thread-safe, but the *Arn Data object* is, so this valve can be remote controlled by an [ArnItem](#).

Example usage

```
// In class code
_commonSapi = new ChatSapi( this);
_commonSapi->open("//Chat/Pipes/pipeCommon", ArnSapi::Mode::Provider);
_commonSapi->batchConnectTo( this, "sapi");

// Control message flow to and from service api _commonSapi
ArnItemValve* arnValve = new ArnItemValve( this);
arnValve->setTarget( _commonSapi->pipe());
arnValve->open("//Chat/Valves/pipeCommon");
*arnValve = true; // Set valve open for message flow
```

Definition at line 77 of file `ArnItemValve.hpp`.

14.35.2 Constructor & Destructor Documentation

14.35.2.1 ArnItemValve()

```
ArnItemValve::ArnItemValve (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 48 of file `ArnItemValve.cpp`.

14.35.3 Member Function Documentation

14.35.3.1 changed

```
void ArnItemValve::changed (
    int value ) [signal]
```

Signals emitted when data in *Arn Data Object* is changed.

14.35.3.2 isAutoDestroy()

```
bool ArnItemValve::isAutoDestroy ( ) const [inline]
```

Return values

<i>true</i>	if <i>AutoDestroy mode</i>
-------------	----------------------------

See also

[setAutoDestroy\(\)](#)

Definition at line 143 of file ArnItemValve.hpp.

14.35.3.3 isMaster()

```
bool ArnItemValve::isMaster ( ) const [inline]
```

Return values

<i>true</i>	if <i>Master mode</i>
-------------	-----------------------

See also

[setMaster\(\)](#)
[Modes](#)

Definition at line 130 of file ArnItemValve.hpp.

14.35.3.4 isSaveMode()

```
bool ArnItemValve::isSaveMode ( ) const [inline]
```

Return values

<i>true</i>	if <i>Save mode</i>
-------------	---------------------

See also

[setSaveMode\(\)](#)
[Modes](#)
[Persistent Arn Data Objects](#)

Definition at line 115 of file ArnItemValve.hpp.

14.35.3.5 operator=()

```
ArnItemValve & ArnItemValve::operator= (
    bool value )
```

Definition at line 91 of file ArnItemValve.cpp.

14.35.3.6 setAutoDestroy()

```
ArnItemValve& ArnItemValve::setAutoDestroy ( ) [inline]
```

Set client session *sync mode* as *AutoDestroy* for this [ArnItem](#).

This [ArnItem](#) at client side is setup for auto destruction.

Precondition

This must be set before [open\(\)](#).

Definition at line 137 of file ArnItemValve.hpp.

14.35.3.7 setMaster()

```
ArnItemValve& ArnItemValve::setMaster ( ) [inline]
```

Set client session *sync mode* as *Master* for this [ArnItem](#).

This [ArnItem](#) at client side is set as default generator of data.

Precondition

This must be set before [open\(\)](#).

See also

[Modes](#)

Definition at line 123 of file ArnItemValve.hpp.

14.35.3.8 setSaveMode()

```
ArnItemValve& ArnItemValve::setSaveMode ( ) [inline]
```

Set *general mode* as *Save* for this *Arn Data Object*

Data is persistent and will be saved

Precondition

The persistent service must be started at the server.

See also

[Modes](#)

[Persistent Arn Data Objects](#)

Definition at line 107 of file ArnItemValve.hpp.

14.35.3.9 setTarget()

```
bool ArnItemValve::setTarget (
    ArnItemB * targetItem,
    ArnItemValve::SwitchMode mode = SwitchMode::InOutputStream )
```

Definition at line 60 of file ArnItemValve.cpp.

14.35.3.10 setValue

```
void ArnItemValve::setValue (
    bool value ) [slot]
```

Assign a *bool* to an *Arn Data Object*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

Definition at line 98 of file ArnItemValve.cpp.

14.35.3.11 switchMode()

```
ArnItemValve::SwitchMode ArnItemValve::switchMode ( ) const
```

Definition at line 72 of file ArnItemValve.cpp.

14.35.3.12 toBool()

```
bool ArnItemValve::toBool ( ) const
```

Returns

state of this valve 1 = Enabled selected stream(s)

Definition at line 80 of file ArnItemValve.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnItemValve.hpp \(4.0.0\)](#)
- [src/ArnItemValve.cpp \(4.0.0\)](#)

14.36 ArnLinkValue Struct Reference

Public Member Functions

- [ArnLinkValue \(\)](#)

Public Attributes

- QString [valueString](#)
- QByteArray [valueByteArray](#)
- QVariant [valueVariant](#)
- volatile ARNREAL [valueReal](#)
- volatile int [valueInt](#)
- quint32 [localUpdateCount](#)

14.36.1 Detailed Description

Definition at line 43 of file ArnLink.cpp.

14.36.2 Constructor & Destructor Documentation

14.36.2.1 ArnLinkValue()

```
ArnLinkValue::ArnLinkValue ( ) [inline]
```

Definition at line 51 of file ArnLink.cpp.

14.36.3 Member Data Documentation

14.36.3.1 localUpdateCount

```
quint32 ArnLinkValue::localUpdateCount
```

Definition at line 49 of file ArnLink.cpp.

14.36.3.2 valueByteArray

```
QByteArray ArnLinkValue::valueByteArray
```

Definition at line 45 of file ArnLink.cpp.

14.36.3.3 valueInt

```
volatile int ArnLinkValue::valueInt
```

Definition at line 48 of file ArnLink.cpp.

14.36.3.4 valueReal

```
volatile ARNREAL ArnLinkValue::valueReal
```

Definition at line 47 of file ArnLink.cpp.

14.36.3.5 valueString

```
QString ArnLinkValue::valueString
```

Definition at line 44 of file ArnLink.cpp.

14.36.3.6 valueVariant

QVariant ArnLinkValue::valueVariant

Definition at line 46 of file ArnLink.cpp.

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

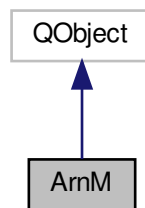
- [src/ArnLink.cpp \(4.0.0\)](#)

14.37 ArnM Class Reference

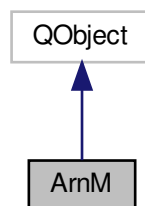
[Arn](#) main class.

```
#include <ArnM.hpp>
```

Inheritance diagram for ArnM:



Collaboration diagram for ArnM:



Public Slots

- static void [destroyLink](#) (const QString &path, bool isGlobal=true)
Destroy the Arn Data Object at path
- static void [setErrorlog](#) (QObject *errLog)

Signals

- void [errorLogSig](#) (const QString &errText, uint errCode, void *reference)

Public Member Functions

- bool [skipLocalSysLoading](#) () const
Return mode skip "/Local/Sys/" loading.
- void [setSkipLocalSysLoading](#) (bool [skipLocalSysLoading](#))
Set mode skip "/Local/Sys/" loading.

Static Public Member Functions

- static [ArnM](#) & [instance](#) ()
- static void [setConsoleError](#) (bool isConsoleError)
- static void [setDefaultIgnoreSameValue](#) (bool isIgnore=true)
Set system default skipping of equal assignment value.
- static bool [defaultIgnoreSameValue](#) ()
- static bool [isMainThread](#) ()
- static bool [isThreadedApp](#) ()
- static int [valueInt](#) (const QString &path)
Get the value of Arn Data Object at path
- static double [valueDouble](#) (const QString &path)
Get the value of Arn Data Object at path
- static [ARNREAL](#) [valueReal](#) (const QString &path)
Get the value of Arn Data Object at path
- static QString [valueString](#) (const QString &path)
Get the value of Arn Data Object at path
- static QByteArray [valueByteArray](#) (const QString &path)
Get the value of Arn Data Object at path
- static QVariant [valueVariant](#) (const QString &path)
Get the value of Arn Data Object at path
- static QStringList [items](#) (const QString &path)
Get the childrens of the folder at path
- static bool [exist](#) (const QString &path)
- static bool [isFolder](#) (const QString &path)
- static bool [isLeaf](#) (const QString &path)
- static void [setAtomicOpProvider](#) (const QString &path)
Set this Arn Data Object as Atomic Operator Provider
- static bool [isAtomicOpProvider](#) (const QString &path)
- static void [setValue](#) (const QString &path, int value)
Assign an integer to an Arn Data Object at path
- static void [setValue](#) (const QString &path, [ARNREAL](#) value)
Assign an ARNREAL to an Arn Data Object at path
- static void [setValue](#) (const QString &path, const QString &value)
Assign a QString to an Arn Data Object at path
- static void [setValue](#) (const QString &path, const QByteArray &value)
Assign a QByteArray to an Arn Data Object at path
- static void [setValue](#) (const QString &path, const QVariant &value, const char *typeName=arnNullptr)
Assign a QVariant to an Arn Data Object at path

- static void [setValue](#) (const QString &path, const char *value)
Assign a char to an [Arn Data Object](#) at path*
- static bool [loadFromFile](#) (const QString &path, const QString &fileName, [Arn::Coding](#) coding)
Load from a file to an [Arn Data Object](#) at path
- static bool [loadFromDirRoot](#) (const QString &path, const QDir &dirRoot, [Arn::Coding](#) coding)
Load relative a directory root to an [Arn Data Object](#) at path
- static bool [saveToFile](#) (const QString &path, const QString &fileName, [Arn::Coding](#) coding)
Save to a file from an [Arn Data Object](#) at path
- static void [errorLog](#) (QString errText, [ArnError](#) err=[ArnError::Undef](#), void *reference=arnNullptr)
- static QString [errorSysName](#) ()
- static QByteArray [info](#) ()
Give information about this library.
- static void [destroyLinkLocal](#) (const QString &path)
Destroy the local [Arn Data Object](#) at path

Friends

- class [ArnBasicItem](#)

14.37.1 Detailed Description

[Arn](#) main class.

About Arn Data Object

This singleton class is the main reference to the Active Registry Network.

Definition at line 106 of file ArnM.hpp.

14.37.2 Member Function Documentation

14.37.2.1 defaultIgnoreSameValue()

```
bool ArnM::defaultIgnoreSameValue ( ) [static]
```

Return values

<i>true</i>	if default skipping equal assignment value
-------------	--

See also

[setDefaultIgnoreSameValue\(\)](#)

Definition at line 1068 of file ArnM.cpp.

14.37.2.2 destroyLink

```
void ArnM::destroyLink (
    const QString & path,
    bool isGlobal = true ) [static], [slot]
```

Destroy the [Arn Data Object](#) at *path*

The link ([Arn Data Object](#)) will be removed locally and optionally from server and all connected clients. Server is always forcing global destroy.

Parameters

in	<i>path</i>	
in	<i>isGlobal</i>	If true, removes from server and all connected clients, otherwise only local link.

See also

[destroyLinkLocal\(\)](#)

Threaded version of `destroyLink`

Definition at line 853 of file ArnM.cpp.

14.37.2.3 destroyLinkLocal()

```
static void ArnM::destroyLinkLocal (
    const QString & path ) [inline], [static]
```

Destroy the local [Arn Data Object](#) at *path*

The link ([Arn Data Object](#)) will be removed locally. Server is always forcing global destroy.

Parameters

in	<i>path</i>	
----	-------------	--

See also

[destroyLink\(\)](#)

Definition at line 296 of file ArnM.hpp.

14.37.2.4 errorLog()

```
void ArnM::errorLog (
    QString errText,
```

```
ArnError err = ArnError::Undef,  
void * reference = arnNullptr ) [static]
```

Definition at line 1025 of file ArnM.cpp.

14.37.2.5 errorLogSig

```
void ArnM::errorLogSig (  
    const QString & errText,  
    uint errCode,  
    void * reference ) [signal]
```

14.37.2.6 errorSysName()

```
QString ArnM::errorSysName ( ) [static]
```

Definition at line 961 of file ArnM.cpp.

14.37.2.7 exist()

```
bool ArnM::exist (  
    const QString & path ) [static]
```

Parameters

in	<i>path</i>	
----	-------------	--

Return values

<i>true</i>	if <i>Arn Data Object</i> exist at <i>path</i>
-------------	--

Definition at line 402 of file ArnM.cpp.

14.37.2.8 info()

```
QByteArray ArnM::info ( ) [static]
```

Give information about this library.

Returns

The info, e.g. "Name=ArnLib Ver=1.0.0 Date=12-12-30 Time=00:37"

Definition at line 967 of file ArnM.cpp.

14.37.2.9 instance()

```
ArnM & ArnM::instance ( ) [static]
```

Definition at line 1048 of file ArnM.cpp.

14.37.2.10 isAtomicOpProvider()

```
bool ArnM::isAtomicOpProvider (
    const QString & path ) [static]
```

Return values

<i>true</i>	if this is a <i>Atomic Operator Provider</i>
-------------	--

Parameters

in	<i>path</i>	
----	-------------	--

See also

[setAtomicOpProvider\(\)](#)

Definition at line 449 of file ArnM.cpp.

14.37.2.11 isFolder()

```
bool ArnM::isFolder (
    const QString & path ) [static]
```

Parameters

in	<i>path</i>	
----	-------------	--

Return values

<i>true</i>	if <i>Arn Data Object</i> at <i>path</i> is a folder
-------------	--

Definition at line 413 of file ArnM.cpp.

14.37.2.12 isLeaf()

```
bool ArnM::isLeaf (
    const QString & path ) [static]
```

Parameters

in	<i>path</i>	
----	-------------	--

Return values

<i>true</i>	if <i>Arn Data Object</i> at <i>path</i> is a leaf (non folder)
-------------	---

Definition at line 424 of file ArnM.cpp.

14.37.2.13 isMainThread()

```
bool ArnM::isMainThread ( ) [static]
```

Return values

<i>true</i>	if this is the main thread in the application
-------------	---

Definition at line 379 of file ArnM.cpp.

14.37.2.14 isThreadedApp()

```
bool ArnM::isThreadedApp ( ) [static]
```

Return values

<i>true</i>	if this is a threaded application
-------------	-----------------------------------

Definition at line 396 of file ArnM.cpp.

14.37.2.15 items()

```
QStringList ArnM::items (
    const QString & path ) [static]
```

Get the childrens of the folder at *path*

Example: return list = {"test"; "folder/"; "@/"; "value"}

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The items (children)

Definition at line 315 of file ArnM.cpp.

14.37.2.16 loadFromDirRoot()

```
bool ArnM::loadFromDirRoot (
    const QString & path,
    const QDir & dirRoot,
    Arn::Coding coding ) [static]
```

Load relative a directory root to an *Arn Data Object* at *path*

Example: *path* = "//Doc/help.txt", *dirRoot* = "/usr/local", will load file from "/usr/local/@/Doc/help.txt" to *Arn* path at "//Doc/help.txt".

Parameters

in	<i>path</i>	is the path of the <i>Arn Data Object</i> and also path relative to <i>dirRoot</i>
in	<i>dirRoot</i>	is the file directory to be used as root for the <i>path</i>
in	<i>coding</i>	indicates if text or binary mode will be used

Return values

<i>true</i>	if loading from file is successful
-------------	------------------------------------

Definition at line 556 of file ArnM.cpp.

14.37.2.17 loadFromFile()

```
bool ArnM::loadFromFile (
    const QString & path,
    const QString & fileName,
    Arn::Coding coding ) [static]
```

Load from a file to an *Arn Data Object* at *path*

Parameters

in	<i>path</i>	is the path of the <i>Arn Data Object</i>
in	<i>fileName</i>	is the file to be loaded
in	<i>coding</i>	indicates if text or binary mode will be used

Return values

<i>true</i>	if loading from file is successful
-------------	------------------------------------

Definition at line 538 of file ArnM.cpp.

14.37.2.18 saveToFile()

```
bool ArnM::saveToFile (
    const QString & path,
    const QString & fileName,
    Arn::Coding coding ) [static]
```

Save to a file from an *Arn Data Object* at *path*

Parameters

in	<i>path</i>	is the path of the <i>Arn Data Object</i>
in	<i>fileName</i>	is the file to be saved
in	<i>coding</i>	indicates if text or binary mode will be used

Return values

<i>true</i>	if saving to file is successful
-------------	---------------------------------

Definition at line 565 of file ArnM.cpp.

14.37.2.19 setAtomicOpProvider()

```
void ArnM::setAtomicOpProvider (
    const QString & path ) [static]
```

Set this *Arn Data Object* as *Atomic Operator Provider*

The atomic operation is performed at this object

Parameters

in	<i>path</i>	
----	-------------	--

Definition at line 436 of file ArnM.cpp.

14.37.2.20 setConsoleError()

```
void ArnM::setConsoleError (
    bool isConsoleError ) [static]
```

Definition at line 1056 of file ArnM.cpp.

14.37.2.21 setDefaultIgnoreSameValue()

```
void ArnM::setDefaultIgnoreSameValue (
    bool isIgnore = true ) [static]
```

Set system default skipping of equal assignment value.

Parameters

in	<i>isIgnore</i>	If true, assignment of equal value don't give a changed signal.
----	-----------------	---

Definition at line 1062 of file ArnM.cpp.

14.37.2.22 setSkipLocalSysLoading()

```
void ArnM::setSkipLocalSysLoading (
    bool skipLocalSysLoading )
```

Set mode skip "/Local/Sys/" loading.

Can disable auto loading of *ARN Data Objects* into "/Local/Sys/ tree".

Parameters

in	<i>skipLocalSysLoading</i>	
----	----------------------------	--

Note

Must be called before entering the Qt event loop
 Check the rules for [Local path](#)

See also

[skipLocalSysLoading\(\)](#)

Definition at line 1080 of file ArnM.cpp.

14.37.2.23 setupErrorlog

```
void ArnM::setupErrorlog (
    QObject * errLog ) [static], [slot]
```

Definition at line 973 of file ArnM.cpp.

14.37.2.24 setValue() [1/6]

```
void ArnM::setValue (
    const QString & path,
    int value ) [static]
```

Assign an *integer* to an [Arn Data Object](#) at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned

Definition at line 473 of file ArnM.cpp.

14.37.2.25 setValue() [2/6]

```
void ArnM::setValue (
    const QString & path,
    ARNREAL value ) [static]
```

Assign an *ARNREAL* to an [Arn Data Object](#) at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned

Definition at line 484 of file ArnM.cpp.

14.37.2.26 setValue() [3/6]

```
void ArnM::setValue (
    const QString & path,
    const QString & value ) [static]
```

Assign a *QString* to an *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned

Definition at line 462 of file ArnM.cpp.

14.37.2.27 setValue() [4/6]

```
void ArnM::setValue (
    const QString & path,
    const QByteArray & value ) [static]
```

Assign a *QByteArray* to an *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned

Definition at line 495 of file ArnM.cpp.

14.37.2.28 setValue() [5/6]

```
void ArnM::setValue (
    const QString & path,
    const QVariant & value,
    const char * typeName = arnNullptr ) [static]
```

Assign a *QVariant* to an *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned
in	<i>typeName</i>	to convert variant into, default no conversion

Definition at line 506 of file ArnM.cpp.

14.37.2.29 setValue() [6/6]

```
void ArnM::setValue (
    const QString & path,
    const char * value ) [static]
```

Assign a *char** to an *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned

Definition at line 532 of file ArnM.cpp.

14.37.2.30 skipLocalSysLoading()

```
bool ArnM::skipLocalSysLoading ( ) const
```

Return mode skip "/Local/Sys/" loading.

Returns

mode skipLocalSysLoading

See also

[setSkipLocalSysLoading\(\)](#)

Definition at line 1074 of file ArnM.cpp.

14.37.2.31 valueByteArray()

```
QByteArray ArnM::valueByteArray (
    const QString & path ) [static]
```

Get the value of *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The *Arn Data Object* as a *QByteArray*

Definition at line 283 of file ArnM.cpp.

14.37.2.32 valueDouble()

```
double ArnM::valueDouble (
    const QString & path ) [static]
```

Get the value of *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The *Arn Data Object* as a *double*

Definition at line 255 of file ArnM.cpp.

14.37.2.33 valueInt()

```
int ArnM::valueInt (
    const QString & path ) [static]
```

Get the value of *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The *Arn Data Object* as an *integer*

Definition at line 244 of file ArnM.cpp.

14.37.2.34 valueReal()

```
ARNREAL ArnM::valueReal (
    const QString & path ) [static]
```

Get the value of *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The *Arn Data Object* as an *ARNREAL*

Definition at line 261 of file ArnM.cpp.

14.37.2.35 valueString()

```
QString ArnM::valueString (
    const QString & path ) [static]
```

Get the value of *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The *Arn Data Object* as a *QString*

Definition at line 272 of file ArnM.cpp.

14.37.2.36 valueVariant()

```
QVariant ArnM::valueVariant (
    const QString & path ) [static]
```

Get the value of *Arn Data Object* at *path*

Parameters

in	<i>path</i>	
----	-------------	--

Returns

The *Arn Data Object* as a *QVariant*

Definition at line 294 of file ArnM.cpp.

14.37.3 Friends And Related Function Documentation

14.37.3.1 ArnBasicItem

```
friend class ArnBasicItem [friend]
```

Definition at line 109 of file ArnM.hpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnM.hpp \(4.0.0\)](#)
- [src/ArnM.cpp \(4.0.0\)](#)

14.38 ArnMonEventType Class Reference

```
#include <ArnMonEvent.hpp>
```

Public Types

- enum [E](#) {
[None](#) = 0, [ItemCreated](#), [ItemFound](#), [ItemDeleted](#),
[ItemModeChg](#), [MonitorStart](#), [MonitorReStart](#) }
- enum [NS](#) { [NsEnum](#), [NsCom](#) }

14.38.1 Detailed Description

Definition at line 39 of file ArnMonEvent.hpp.

14.38.2 Member Enumeration Documentation

14.38.2.1 E

```
enum ArnMonEventType::E
```

Enumerator

None	Invalid.
ItemCreated	Newly created Arn object.
ItemFound	Found a present Arn object.
ItemDeleted	An Arn object was deleted.
ItemModeChg	An Arn object changed mode.
MonitorStart	Internal: start the Monitor.
MonitorReStart	Internal: restart the Monitor.

Definition at line 43 of file ArnMonEvent.hpp.

14.38.2.2 NS

```
enum ArnMonEventType::NS
```

Enumerator

NsEnum	
NsCom	

Definition at line 62 of file ArnMonEvent.hpp.

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

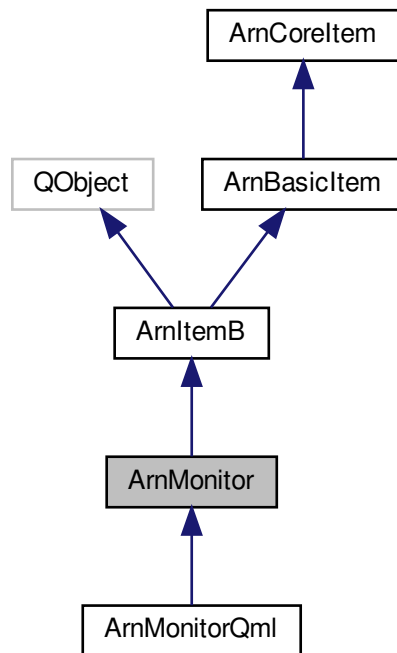
- [src/ArnInc/ArnMonEvent.hpp \(4.0.0\)](#)

14.39 ArnMonitor Class Reference

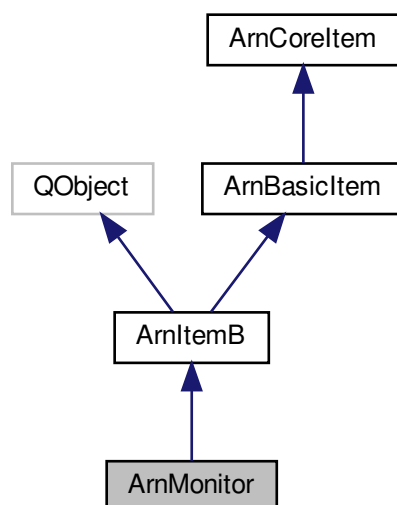
A client remote monitor to detect changes at server.

```
#include <ArnMonitor.hpp>
```

Inheritance diagram for ArnMonitor:



Collaboration diagram for ArnMonitor:



Public Slots

- void `foundChildDeleted` (const QString &path)
Help telling the monitor about deletion of a previous found child.

Signals

- void `monitorClosed` ()
Signal emitted when the Arn Monitor is closed down.
- void `arnItemCreated` (const QString &path)
Signal emitted when an Arn Data Object is created in the tree below.
- void `arnChildFound` (const QString &path)
Signal emitted for present and newly created childs in the monitor folder.
- void `arnChildFoundFolder` (const QString &path)
Signal emitted for present and newly created folder childs in the monitor folder.
- void `arnChildFoundLeaf` (const QString &path)
Signal emitted for present and newly created leaf childs in the monitor folder.
- void `arnItemDeleted` (const QString &path)
Signal emitted when an Arn Data Object is deleted in the tree below.
- void `arnChildDeleted` (const QString &path)
Signal emitted for deleted childs in the monitor folder.
- void `arnItemModeChanged` (const QString &path, int mode)
Signal emitted when an Arn Data Object changes mode in the tree below.
- void `arnChildModeChanged` (const QString &path, int mode)
Signal emitted for mode changing childs in the monitor folder.

Public Member Functions

- `ArnMonitor` (QObject *parent=arnNullptr)
- `ArnMonitor` (const QString &path, QObject *parent=arnNullptr)
Starts local monitoring.
- `~ArnMonitor` ()
- void `setClient` (ArnClient *client)
Set the client to be used.
- void `setClient` (const QString &id)
Set the client to be used by its id.
- QString `clientId` () const
Get the id name of the used client
- ArnClient * `client` () const
Get the used client
- void `setMonitorPath` (const QString &path, ArnClient *client=arnNullptr)
Set the path to be monitored.
- bool `start` (const QString &path, ArnClient *client)
Starts the monitoring.
- bool `start` (const QString &path)
Starts the monitoring.
- QString `monitorPath` () const
Get the monitored path
- void `reStart` ()
The monitor is restarted.
- void `setReference` (void *reference)
Set an associated external reference.
- void * `reference` () const
Get the stored external reference.

14.39.1 Detailed Description

A client remote monitor to detect changes at server.

The monitor must normally be set at a [shared](#) path. A none shared path can be used when client is set to 0, i.e. local monitoring.

When the monitor is started, all the *arnChildFound* signals are emitted for present childs. Later the signals are emitted for newly created childs.

Example usage

```
// In class declare
ArnMonitor* _arnMon;
ArnClient* _client;

// In class code
_arnMon = new ArnMonitor( this);
_arnMon->start("//Pipes/", _client);
connect( _arnMon, SIGNAL(arnChildFound(QString)), this, SLOT(netChildFound(QString)));
```

Definition at line 65 of file ArnMonitor.hpp.

14.39.2 Constructor & Destructor Documentation

14.39.2.1 ArnMonitor() [1/2]

```
ArnMonitor::ArnMonitor (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 64 of file ArnMonitor.cpp.

14.39.2.2 ArnMonitor() [2/2]

```
ArnMonitor::ArnMonitor (
    const QString & path,
    QObject * parent = arnNullptr )
```

Starts local monitoring.

Parameters

in	<i>path</i>	
in	<i>parent</i>	

See also

[start\(\)](#)

Definition at line 72 of file ArnMonitor.cpp.

14.39.2.3 ~ArnMonitor()

```
ArnMonitor::~ArnMonitor ( )
```

Definition at line 89 of file ArnMonitor.cpp.

14.39.3 Member Function Documentation

14.39.3.1 arnChildDeleted

```
void ArnMonitor::arnChildDeleted (
    const QString & path ) [signal]
```

Signal emitted for deleted childs in the monitor folder.

The [ArnMonitor](#) monitors a folder. Deleted objects in this folder will give this signal.

Example 1: monitorPath = "//Sensors/Temp1/", deleted object = "//Sensors/Temp1/value" ==> path to child = "//↔Sensors/Temp1/value"

Example 2: monitorPath = "//Sensors/Temp2/", deleted object = "//Sensors/Temp2/folder/" ==> path to child = "//↔Sensors/Temp2/folder/"

Parameters

in	<i>path</i>	to the child
----	-------------	--------------

See also

[arnItemDeleted\(\)](#)

14.39.3.2 arnChildFound

```
void ArnMonitor::arnChildFound (
    const QString & path ) [signal]
```


Signal emitted for present and newly created childs in the monitor folder.

The [ArnMonitor](#) monitors a folder. Present and newly created objects in this folder will give this signal. For newly created objects, the origin comes from the [arnItemCreated\(\)](#) signal.

Example 1: monitorPath = "//Sensors/", created object = "//Sensors/Temp1/value" ==> path to child = "//Sensors/↔Temp1/"

Example 2: monitorPath = "//Sensors/", created object = "//Sensors/Temp2/folder/" ==> path to child = "//Sensors/↔Temp2/"

Parameters

in	<i>path</i>	to the child
----	-------------	--------------

See also

[arnItemCreated\(\)](#)

14.39.3.3 arnChildFoundFolder

```
void ArnMonitor::arnChildFoundFolder (
    const QString & path ) [signal]
```

Signal emitted for present and newly created folder childs in the monitor folder.

The [ArnMonitor](#) monitors a folder. Present and newly created folder objects in this folder will give this signal. For newly created childs, the origin comes from the [arnItemCreated\(\)](#) signal.

Example: monitorPath = "//Sensors/", created object = "//Sensors/Temp1/value" ==> path to child = "//Sensors/↔Temp1/"

Parameters

in	<i>path</i>	to the child
----	-------------	--------------

See also

[arnItemCreated\(\)](#)
[arnChildFound\(\)](#)

14.39.3.4 arnChildFoundLeaf

```
void ArnMonitor::arnChildFoundLeaf (
    const QString & path ) [signal]
```

Signal emitted for present and newly created leaf childs in the monitor folder.

The [ArnMonitor](#) monitors a folder. Present and newly created leaf objects in this folder will give this signal. For newly created childs, the origin comes from the [arnItemCreated\(\)](#) signal.

Example: monitorPath = "//Sensors/", created object = "//Sensors/count" ==> path to child = "//Sensors/count"

Parameters

in	<i>path</i>	to the child
----	-------------	--------------

See also

[arnChildFound\(\)](#)

14.39.3.5 arnChildModeChanged

```
void ArnMonitor::arnChildModeChanged (
    const QString & path,
    int mode ) [signal]
```

Signal emitted for mode changing childs in the monitor folder.

The [ArnMonitor](#) monitors a folder. Objects changing mode in this folder will give this signal.

Example: monitorPath = "//Sensors/Temp1/", changing mode object = "//Sensors/Temp1/value" ==> path to child = "//Sensors/Temp1/value"

Parameters

in	<i>path</i>	to the child
in	<i>mode</i>	is the new Arn::ObjectMode

See also

[arnItemModeChanged\(\)](#)

14.39.3.6 arnItemCreated

```
void ArnMonitor::arnItemCreated (
    const QString & path ) [signal]
```

Signal emitted when an [Arn Data Object](#) is created in the tree below.

The [ArnMonitor](#) monitors a folder. Created objects in this folder or its children below will give this signal. Both created folder and leaf objects will give this signal.

Parameters

in	<i>path</i>	to the created <i>Arn Data Object</i>
----	-------------	---------------------------------------

14.39.3.7 arnItemDeleted

```
void ArnMonitor::arnItemDeleted (
    const QString & path ) [signal]
```

Signal emitted when an *Arn Data Object* is deleted in the tree below.

The *ArnMonitor* monitors a folder. Deleted objects in this folder or its children below will give this signal. Both deleted folder and leaf objects will give this signal.

Parameters

in	<i>path</i>	to the deleted <i>Arn Data Object</i>
----	-------------	---------------------------------------

14.39.3.8 arnItemModeChanged

```
void ArnMonitor::arnItemModeChanged (
    const QString & path,
    int mode ) [signal]
```

Signal emitted when an *Arn Data Object* changes mode in the tree below.

The *ArnMonitor* monitors a folder. Objects changing mode in this folder or its children below will give this signal.

Parameters

in	<i>path</i>	to the mode changing <i>Arn Data Object</i>
in	<i>mode</i>	is the new <i>Arn::ObjectMode</i>

14.39.3.9 client()

```
ArnClient * ArnMonitor::client ( ) const
```

Get the used *client*

Returns

The *client*

See also

[setClient\(\)](#)

Definition at line 126 of file ArnMonitor.cpp.

14.39.3.10 `clientId()`

```
QString ArnMonitor::clientId ( ) const
```

Get the id name of the used *client*

Returns

The *client* id name

See also

[setClient\(\)](#)

Definition at line 117 of file ArnMonitor.cpp.

14.39.3.11 `foundChildDeleted`

```
void ArnMonitor::foundChildDeleted (
    const QString & path ) [slot]
```

Help telling the monitor about deletion of a previous found child.

The monitor remembers every child it has signalled. If a deleted child reappears later it will not give a signal unless this function is used.

Since ArnLib 3.0 this function is called automatically when a child is deleted. This function is still available to manually handle any problems.

Parameters

in	<i>path</i>	to the deleted child
----	-------------	----------------------

Definition at line 377 of file ArnMonitor.cpp.

14.39.3.12 `monitorClosed`

```
void ArnMonitor::monitorClosed ( ) [signal]
```

Signal emitted when the *Arn Monitor* is closed down.

There is an internal (remote) pickup *ArnItem* at the monitor path. When the internal *ArnItem* is destroyed, this *ArnMonitor* is closed and will give this signal

14.39.3.13 monitorPath()

```
QString ArnMonitor::monitorPath ( ) const
```

Get the monitored *path*

Returns

The *path*

See also

[start\(\)](#)

Definition at line 214 of file ArnMonitor.cpp.

14.39.3.14 reference()

```
void * ArnMonitor::reference ( ) const
```

Get the stored external reference.

Returns

The associated external reference

See also

[setReference\(\)](#)

Definition at line 239 of file ArnMonitor.cpp.

14.39.3.15 reStart()

```
void ArnMonitor::reStart ( )
```

The monitor is restarted.

This makes the monitor forget the signals sent for present children and the *arnChildFound* signals are emitted again for present childs.

Definition at line 222 of file ArnMonitor.cpp.

14.39.3.16 setClient() [1/2]

```
void ArnMonitor::setClient (
    ArnClient * client )
```

Set the *client* to be used.

Parameters

in	<i>client</i>	to be used. If 0, local monitoring is done.
----	---------------	---

Definition at line 101 of file ArnMonitor.cpp.

14.39.3.17 `setClient()` [2/2]

```
void ArnMonitor::setClient (
    const QString & id )
```

Set the *client* to be used by its id.

Parameters

in	<i>id</i>	to identify client. If "", local monitoring is done.
----	-----------	--

Definition at line 109 of file ArnMonitor.cpp.

14.39.3.18 `setMonitorPath()`

```
void ArnMonitor::setMonitorPath (
    const QString & path,
    ArnClient * client = arnNullptr )
```

Set the *path* to be monitored.

The monitor must be set at a [shared path](#) that is shared using `client::addMountPoint()`. This function also starts the monitoring using `start()`.

Parameters

in	<i>path</i>	
in	<i>client</i>	to be used. If 0, keep previous set client.

See also

[start\(\)](#)

Deprecated Use [start\(\)](#) instead, *client* parameter is changed.

Definition at line 134 of file ArnMonitor.cpp.

14.39.3.19 setReference()

```
void ArnMonitor::setReference (
    void * reference )
```

Set an associated external reference.

This is typically used when having many *ArnMonitors* signal connected to a common slot. The slot can then discover the signalling *ArnMonitor*:s associated structure for further processing.

Parameters

in	<i>reference</i>	Any external structure or id.
----	------------------	-------------------------------

See also

[reference\(\)](#)

Definition at line 231 of file ArnMonitor.cpp.

14.39.3.20 start() [1/2]

```
bool ArnMonitor::start (
    const QString & path,
    ArnClient * client )
```

Starts the monitoring.

The monitor must normally be set at a [shared path](#) that is shared using `client::addMountPoint()`. A none shared path can be used when `client` is set to 0, i.e. local monitoring.

Parameters

in	<i>path</i>	
in	<i>client</i>	to be used. If 0, local monitoring is done.

Definition at line 142 of file ArnMonitor.cpp.

14.39.3.21 start() [2/2]

```
bool ArnMonitor::start (
    const QString & path )
```

Starts the monitoring.

The monitor must normally be set at a [shared path](#) that is shared using `client::addMountPoint()`. A none shared path can be used when `client` is set to 0, i.e. local monitoring.

Parameters

in	<i>path</i>	
----	-------------	--

Definition at line 208 of file ArnMonitor.cpp.

The documentation for this class was generated from the following files:

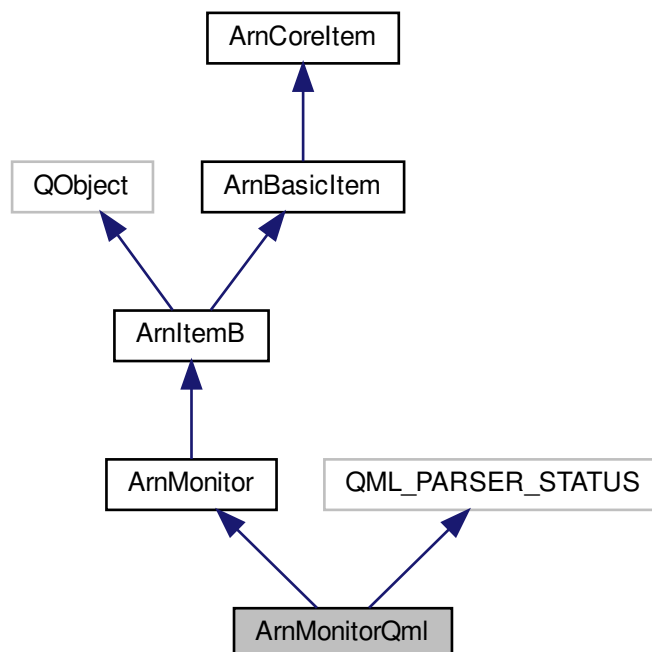
- [src/ArnInc/ArnMonitor.hpp \(4.0.0\)](#)
- [src/ArnMonitor.cpp \(4.0.0\)](#)

14.40 ArnMonitorQml Class Reference

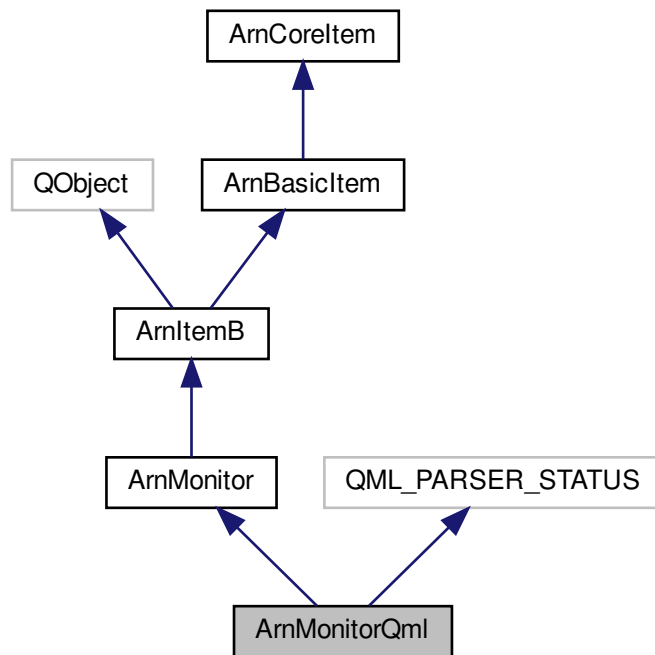
ARN Monitor QML.

```
#include <ArnQml.hpp>
```

Inheritance diagram for ArnMonitorQml:



Collaboration diagram for ArnMonitorQml:



Public Slots

- void `reStart()`
Restart the monitor.

Properties

- QString `clientId`
The client id. Set with `ArnClient::registerClient()`. Use "std" if not set.
- QString `monitorPath`
The path to be monitored at the server.

Additional Inherited Members

14.40.1 Detailed Description

ARN Monitor QML.

This class is the Qml version of the [ArnMonitor](#).

See also

[ArnQml](#)

Example usage

```
// In Qml
//
import QtQuick 2.0
import ArnLib 1.0

Rectangle {
    width: 370; height: 400

    ArnMonitor {
        clientId: "std"
        monitorPath: "//Test/List/"
        onArnChildFound: console.log("Found list item: " + path);
    }
}
```

Definition at line 448 of file ArnQml.hpp.

14.40.2 Member Function Documentation

14.40.2.1 reStart

```
void ArnMonitorQml::reStart ( ) [slot]
```

Restart the monitor.

All signals for found childs will be emitted again.

Definition at line 320 of file ArnQml.cpp.

14.40.3 Property Documentation

14.40.3.1 clientId

```
QString ArnMonitorQml::clientId [read], [write]
```

The client id. Set with [ArnClient::registerClient\(\)](#). Use "std" if not set.

Definition at line 459 of file ArnQml.hpp.

14.40.3.2 monitorPath

```
QString ArnMonitorQml::monitorPath [read], [write]
```

The path to be monitored at the server.

Definition at line 461 of file ArnQml.hpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQml.hpp \(4.0.0\)](#)
- [src/ArnQml.cpp \(4.0.0\)](#)

14.41 ArnNullptr Struct Reference

```
#include <ArnLib_global.hpp>
```

Public Member Functions

- [template<class T >
operator T* \(\)](#)

14.41.1 Detailed Description

Definition at line 19 of file ArnLib_global.hpp.

14.41.2 Member Function Documentation

14.41.2.1 operator T*()

```
template<class T >  
ArnNullptr::operator T* ( ) [inline]
```

Definition at line 22 of file ArnLib_global.hpp.

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

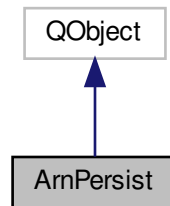
- [src/ArnInc/ArnLib_global.hpp \(4.0.0\)](#)

14.42 ArnPersist Class Reference

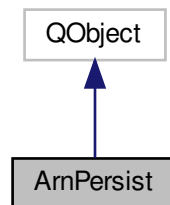
Class for handling persistent *Arn Data object*.

```
#include <ArnPersist.hpp>
```

Inheritance diagram for ArnPersist:



Collaboration diagram for ArnPersist:



Public Slots

- bool [doArchive](#) (const QString &name=QString())
Do a persistent database backup.

Public Member Functions

- [ArnPersist](#) (QObject *parent=arnNullptr)
- [~ArnPersist](#) ()
- bool [setMountPoint](#) (const QString &path)
Set the persistent enabled tree path.
- void [setPersistDir](#) (const QString &path)
Set the persistent file directory root

- void [setArchiveDir](#) (const QString &path)
Set the persistent database backup directory.
- void [setVcs](#) (ArnVcs *vcs)
Set the Version Control System to be used.
- bool [setupDataBase](#) (const QString &dbName="persist.db")
Setup the persistent database.
- bool [flush](#) (const QString &path=QString())
Save any pending values now.

14.42.1 Detailed Description

Class for handling persistent *Arn Data object*.

About Persistent Arn Data Object

This class is used at an *ArnServer* to implement persistent objects.

Example usage

```
// In class declare
ArnPersist *_persist;
VcsGit *_git;

// In class code
_persist = new ArnPersist( this);
_persist->setupDataBase("persist.db");
_persist->setArchiveDir("archive"); // Use this directory for backup
_persist->setPersistDir("persist"); // use this directory for VCS persist files
_persist->setMountPoint("/");
_persist->setVcs( _git);
```

Definition at line 168 of file ArnPersist.hpp.

14.42.2 Constructor & Destructor Documentation

14.42.2.1 ArnPersist()

```
ArnPersist::ArnPersist (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 210 of file ArnPersist.cpp.

14.42.2.2 ~ArnPersist()

```
ArnPersist::~ArnPersist ( )
```

Definition at line 226 of file ArnPersist.cpp.

14.42.3 Member Function Documentation

14.42.3.1 doArchive

```
bool ArnPersist::doArchive (
    const QString & name = QString() ) [slot]
```

Do a persistent database backup.

By default the backup file will be marked by date and clock. Optionally a custom name can be set for the backup file.

Parameters

in	<i>name</i>	is the file name of the backup. QString() gives default name.
----	-------------	---

See also

[setArchiveDir\(\)](#)

Definition at line 843 of file ArnPersist.cpp.

14.42.3.2 flush()

```
bool ArnPersist::flush (
    const QString & path = QString() )
```

Save any pending values now.

Persistent values are normally delayed before saving.

Parameters

in	<i>path</i>	is the starting path (tree) as filter. If empty, no filter.
----	-------------	---

Return values

<i>false</i>	if error.
--------------	-----------

See also

[Persistent Arn Data Objects](#)

Definition at line 527 of file ArnPersist.cpp.

14.42.3.3 setArchiveDir()

```
void ArnPersist::setArchiveDir (
    const QString & path )
```

Set the persistent database backup directory.

In this directory, all backup files are stored.

Parameters

in	<i>path</i>	is the persistent file directory <i>root</i> .
----	-------------	--

See also

[doArchive\(\)](#)
[Persistent Arn Data Objects](#)

Definition at line 240 of file ArnPersist.cpp.

14.42.3.4 setMountPoint()

```
bool ArnPersist::setMountPoint (
    const QString & path )
```

Set the persistent enabled tree path.

Mountpoint is a folder. When an *Arn Data Object* change to *Save* mode in this folder or anywhere below in the tree, it will be treated as a persistent object.

Parameters

in	<i>path</i>	is the persistent enabled tree.
----	-------------	---------------------------------

Return values

<i>false</i>	if error.
--------------	-----------

See also

[Persistent Arn Data Objects](#)

Definition at line 434 of file ArnPersist.cpp.

14.42.3.5 setPersistDir()

```
void ArnPersist::setPersistDir (
    const QString & path )
```

Set the persistent file directory *root*

In this directory and below, all persistent files are stored. The *path* correspond to the *root* in [Arn](#).

This file directory can optionally be managed by a *version control system*, set by using [setVcs\(\)](#).

Example: *path* is set to `"/usr/local/arn_persist"`. There is a file stored at `"/usr/local/arn_persist/@/doc/help.html"`. This file will be mapped to [Arn](#) at `"/doc/help.html"`.

Parameters

in	<i>path</i>	is the persistent file directory <i>root</i> .
----	-------------	--

See also

[setVcs\(\)](#)
[Persistent Arn Data Objects](#)

Definition at line 232 of file ArnPersist.cpp.

14.42.3.6 setupDataBase()

```
bool ArnPersist::setupDataBase (
    const QString & dbName = "persist.db" )
```

Setup the persistent database.

Starting a SQLite database to store persistent [Arn Data Object](#) in.

Parameters

in	<i>dbName</i>	is the name (and path) of the SQLite database file.
----	---------------	---

Return values

<i>false</i>	if error.
--------------	-----------

See also

[Persistent Arn Data Objects](#)

Definition at line 466 of file ArnPersist.cpp.

14.42.3.7 setVcs()

```
void ArnPersist::setVcs (
    ArnVcs * vcs )
```

Set the *Version Control System* to be used.

The VCS is implemented in a class derived from ArnVcs. Ownership is taken of this VCS. Any previous set VCS will be deleted.

Parameters

in	vcs	is the class implementing the VCS. Use 0 (null) to set none.
----	-----	--

See also

[setPersistDir\(\)](#)
[Persistent Arn Data Objects](#)

Definition at line 248 of file ArnPersist.cpp.

The documentation for this class was generated from the following files:

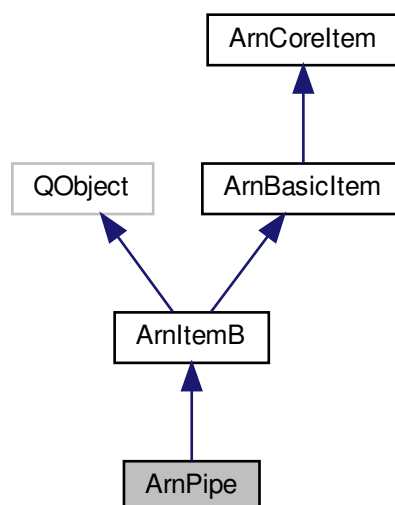
- [src/ArnInc/ArnPersist.hpp \(4.0.0\)](#)
- [src/ArnPersist.cpp \(4.0.0\)](#)

14.43 ArnPipe Class Reference

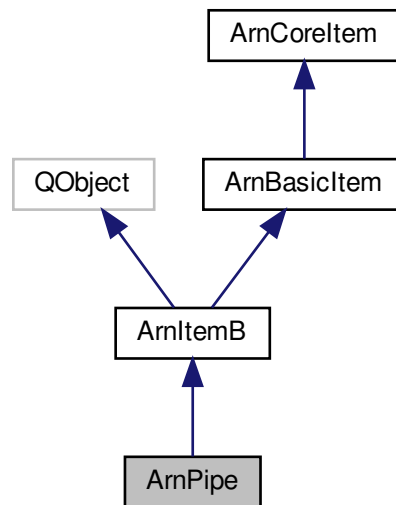
[ArnItem](#) specialized as a pipe.

```
#include <ArnPipe.hpp>
```

Inheritance diagram for ArnPipe:



Collaboration diagram for ArnPipe:



Public Slots

- void [setValue](#) (const QByteArray &value)
Assign a QByteArray to a Pipe

Signals

- void [changed](#) (const QByteArray &value)
Signal emitted when Pipe has received data.
- void [outOfSequence](#) ()
Signal emitted when the received sequence numbers are "out of sequence".

Public Member Functions

- [ArnPipe](#) (QObject *parent=arnNullptr)
Standard constructor of a closed handle.
- [ArnPipe](#) (const QString &path, QObject *parent=arnNullptr)
Construction of a pipe handle to a path
- virtual [~ArnPipe](#) ()
- bool [openUuid](#) (const QString &path)
Open a handle to an Arn Pipe Object with a unique uuid name.
- [ArnPipe](#) & [setMaster](#) ()
Set client session sync mode as Master for this ArnItem.
- bool [isMaster](#) () const
- [ArnPipe](#) & [setAutoDestroy](#) ()

Set client session sync mode as *AutoDestroy* for this [ArnItem](#).

- bool [isAutoDestroy](#) () const
- [ArnPipe](#) & [operator=](#) (const QByteArray &value)
- void [setValueOverwrite](#) (const QByteArray &value, const [ARN_RegExp](#) &rx)

Assign a *QByteArray* to a *Pipe* by using *Anti congest logic*.

- bool [isSendSeq](#) () const
Returns true if sending sequence numbers.
- void [setSendSeq](#) (bool useSendSeq)
Change usage of sending sequence numbers.
- bool [isCheckSeq](#) () const
Returns true if checking received sequence numbers.
- void [setCheckSeq](#) (bool useCheckSeq)
Change usage of checking received sequence numbers.

14.43.1 Detailed Description

[ArnItem](#) specialized as a pipe.

About Pipes

This class is not thread-safe, but the *Arn Data object* is, so each thread should have it's own handles i.e [ArnPipe](#) instances.

Example usage

```
// In class declare
ArnPipe _arnPipe;

// In class code
_arnPipe.open("//Pipes/Pipe/value");
_arnPipe.setSendSeq( true);
_arnPipe.setCheckSeq( true);
connect( &_arnPipe., SIGNAL(outOfSequence()), this, SLOT(doOutOfSequence()));
connect( &_arnPipe, SIGNAL(changed(QByteArray)), this, SLOT(doPipeInput(QByteArray)));

ARN_RegExp rx("^ping\\b");
_arnPipe.setValueOverwrite( "ping new", rx);
```

Definition at line 64 of file [ArnPipe.hpp](#).

14.43.2 Constructor & Destructor Documentation

14.43.2.1 ArnPipe() [1/2]

```
ArnPipe::ArnPipe (
    QObject * parent = arnNullptr )
```

Standard constructor of a closed handle.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 58 of file ArnPipe.cpp.

14.43.2.2 ArnPipe() [2/2]

```
ArnPipe::ArnPipe (
    const QString & path,
    QObject * parent = arnNullptr )
```

Construction of a pipe handle to a *path*

The mode for this handle is set to [Arn::ObjectMode::Pipe](#).

Parameters

in	<i>path</i>	The <i>Arn Data Object</i> path e.g. "//Pipes/myPipe/value"
in	<i>parent</i>	

See also

[open\(\)](#)

Definition at line 65 of file ArnPipe.cpp.

14.43.2.3 ~ArnPipe()

```
ArnPipe::~ArnPipe ( ) [virtual]
```

Definition at line 79 of file ArnPipe.cpp.

14.43.3 Member Function Documentation**14.43.3.1 changed**

```
void ArnPipe::changed (
    const QByteArray & value ) [signal]
```

Signal emitted when *Pipe* has received data.

This is implied by the *Arn Data Object* is changed.

Parameters

in	<i>value</i>	is the received bytes
----	--------------	-----------------------

14.43.3.2 isAutoDestroy()

```
bool ArnPipe::isAutoDestroy ( ) const [inline]
```

Return values

<i>true</i>	if <i>AutoDestroy mode</i>
-------------	----------------------------

See also

[setAutoDestroy\(\)](#)

Definition at line 118 of file ArnPipe.hpp.

14.43.3.3 isCheckSeq()

```
bool ArnPipe::isCheckSeq ( ) const
```

Returns true if checking received sequence numbers.

Return values

<i>true</i>	if checking received sequence numbers
-------------	---------------------------------------

See also

[setCheckSeq\(\)](#)

Definition at line 146 of file ArnPipe.cpp.

14.43.3.4 isMaster()

```
bool ArnPipe::isMaster ( ) const [inline]
```

Return values

<i>true</i>	if <i>Master mode</i>
-------------	-----------------------

See also

[setMaster\(\)](#)
[Modes](#)

Definition at line 105 of file ArnPipe.hpp.

14.43.3.5 isSendSeq()

```
bool ArnPipe::isSendSeq ( ) const
```

Returns true if sending sequence numbers.

Return values

<i>true</i>	if sending sequence numbers
-------------	-----------------------------

See also

[setSendSeq\(\)](#)

Definition at line 130 of file ArnPipe.cpp.

14.43.3.6 openUuid()

```
bool ArnPipe::openUuid (
    const QString & path ) [inline]
```

Open a handle to an [Arn](#) Pipe Object with a unique uuid name.

If *path* is marked as provider, the "!" marker will be moved to after uuid.

Parameters

in	<i>path</i>	The prefix for Arn uuid pipe path e.g. "//Pipes/pipe"
----	-------------	---

Return values

<i>false</i>	if error
--------------	----------

Definition at line 90 of file ArnPipe.hpp.

14.43.3.7 operator=()

```
ArnPipe & ArnPipe::operator= (
    const QByteArray & value )
```

Definition at line 98 of file ArnPipe.cpp.

14.43.3.8 outOfSequence

```
void ArnPipe::outOfSequence ( ) [signal]
```

Signal emitted when the received sequence numbers are "out of sequence".

See also

[setCheckSeq\(\)](#)
[setSendSeq\(\)](#)
[Pipe sequence check](#)

14.43.3.9 setAutoDestroy()

```
ArnPipe& ArnPipe::setAutoDestroy ( ) [inline]
```

Set client session *sync mode* as *AutoDestroy* for this [ArnItem](#).

This [ArnItem](#) at client side is setup for auto destruction.

Precondition

This must be set before [open\(\)](#).

Definition at line 112 of file ArnPipe.hpp.

14.43.3.10 setCheckSeq()

```
void ArnPipe::setCheckSeq (
    bool useCheckSeq )
```

Change usage of checking received sequence numbers.

Parameters

in	<i>useCheckSeq</i>	is true for activation
----	--------------------	------------------------

See also

[isCheckSeq\(\)](#)
[setSendSeq\(\)](#)
[outOfSequence\(\)](#)
[Pipe sequence check](#)

Definition at line 154 of file ArnPipe.cpp.

14.43.3.11 setMaster()

```
ArnPipe& ArnPipe::setMaster ( ) [inline]
```

Set client session *sync mode* as *Master* for this [ArnItem](#).

This [ArnItem](#) at client side is set as default generator of data.

Precondition

This must be set before [open\(\)](#).

See also

[Modes](#)

Definition at line 98 of file ArnPipe.hpp.

14.43.3.12 setSendSeq()

```
void ArnPipe::setSendSeq (
    bool useSendSeq )
```

Change usage of sending sequence numbers.

Parameters

in	<i>useSendSeq</i>	is true for activation
----	-------------------	------------------------

See also

[isSendSeq\(\)](#)
[setCheckSeq\(\)](#)
[outOfSequence\(\)](#)
[Pipe sequence check](#)

Definition at line 138 of file ArnPipe.cpp.

14.43.3.13 setValue

```
void ArnPipe::setValue (
    const QByteArray & value ) [slot]
```

Assign a *QByteArray* to a *Pipe*

Parameters

in	<i>value</i>	to be assigned
----	--------------	----------------

Definition at line 84 of file ArnPipe.cpp.

14.43.3.14 setValueOverwrite()

```
void ArnPipe::setValueOverwrite (
    const QByteArray & value,
    const ARN_RegExp & rx )
```

Assign a *QByteArray* to a *Pipe* by using *Anti congest* logic.

This is used to limit the filling of sendqueue with recurring messages during some kind of client disconnection. Matched message in sendqueue is overwritten by the new message *value*. Unmatched message is added to send queue as usual.

Example:

```
// Messages starts with a function name
// We want message with equal function name to overwrite
ARN_RegExp rx("^" + funcName + "\\b");
_pipe->setValueOverwrite( message, rx);
```

Parameters

in	<i>value</i>	to be assigned
in	<i>rx</i>	is regexp to be matched with items in send queue.

See also

[Pipe anti congest](#)

Definition at line 105 of file ArnPipe.cpp.

The documentation for this class was generated from the following files:

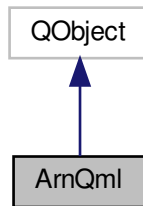
- [src/ArnInc/ArnPipe.hpp \(4.0.0\)](#)
- [src/ArnPipe.cpp \(4.0.0\)](#)

14.44 ArnQml Class Reference

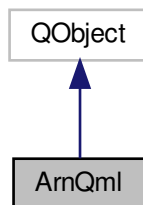
ARN QML.

```
#include <ArnQml.hpp>
```

Inheritance diagram for ArnQml:



Collaboration diagram for ArnQml:



Classes

- struct [UseFlags](#)

Static Public Member Functions

- static void [setup](#) ([QML_ENGINE](#) *qmlEngine, [UseFlags](#) flags=[UseFlags::ArnLib](#))
Add ArnLib support to a Qml instance.
- static [ArnQml](#) & [instance](#) ()
- static [QString](#) [arnRootPath](#) ()
Gives current ARN root path for all qml instances.
- static void [setArnRootPath](#) (const [QString](#) &path)
Change ARN root path for all qml instances.

14.44.1 Detailed Description

ARN QML.

Note

This class must be partly thread-safe

This class is the central point for [ArnQml](#). It's a singleton that is setup in the application. [ArnQml](#) can be used for creating GUI-applications in Qml that has integrated access to the ARN objects and some of the ArnLib functionality.

For information about available ArnLib components in Qml see:

QmlType	See
Arn	ArnInterface
ArnItem	ArnItemQml
ArnMonitor	ArnMonitorQml
ArnSapi	ArnSapiQml
XStringMap	XStringMapQml

If the Qml code must run in both Quick1 (Qt4) and Quick2 (Qt5), following apply: Only Quick1 code will be able to run in both environments. When this code is run in Quick2 its "import QtQuick 1" will be changed internally to "import QtQuick 2". "arn" is now an instantiation of [ArnInterface](#) and "Arn" is the type. In qml "arn.quickTypeRun" will give a 1 when running in a QtQuick1 environment and a 2 for QtQuick2.

When the Qml code only is to be run in Quick2 it should use "import QtQuick 2". In this case "Arn" will be a singleton instantiation of [ArnInterface](#). "arn" is then not needed.

ArnBrowser is using this class to run Qml applications in an opaque style, i.e. without specific application support. This resembles somewhat a web browser running a web application.

Note that you must not use any empty folders in QUrl for an ARN path. Example: path "//Qml/test.qml" can be set to the equal path "@/Qml/test.qml". Also this conversion can be made by [Arn::convertPath\("//Qml/test.qml", Arn::NameF\(\)\)](#).

Example usage

```
// In c++
//
QQuickView* view = new QQuickView;
ArnQml::setup( view->engine(), ArnQml::UseFlags::All);

QString qmlPathInArn = "//Qml/test.qml"
QUrl url;
url.setScheme("arn");
url.setPath( Arn::convertPath( qmlPathInArn, Arn::NameF()));
view->setSource( url);
view->show();

connect( engine(), SIGNAL(quit()), this, SLOT(onClose()));
connect( view, SIGNAL(closing(QQuickCloseEvent*)), this, SLOT(onClose()));

// In Qml
//
import QtQuick 2.0
import ArnLib 1.0

Rectangle {
    width: 370; height: 400

    ArnMonitor {
```

```

        clientId: "std"
        monitorPath: "//Test/List/"
        onArnChildFound: console.log("Found list item: " + path);
    }

    Image {
        anchors.top: parent.top; anchors.right: parent.right;
        source: "arn:///@/Test/Data/pic.png"
    }

    ArnItem {id: arnElUpdClock; path: "//El/UpdClock/value"}

    Item {
        id: sapiTest
        ArnSapi {pipePath: "//Test/pipe"}

        // Provider API
        signal pv_readFileTest( string fileName)

        // Requester API
        signal rq_test2( string parl)
        function rq_test( pl) {
            console.log("rq_test: pl=" + pl);
        }

        Component.onCompleted: {
            sapiTest.rq_test2.connect( info.setTestMsg);
            sapiTest.pv_readFileTest("myfile");
        }
    }

    Rectangle {
        id: info
        property string testMsg: ""
        anchors.bottom: parent.bottom; anchors.left: parent.left; anchors.right: parent.right
        height: 80
        Column {
            anchors.fill: parent;
            Text {text: "El updClock: " + arnElUpdClock.intNum}
            Text {text: "Msg: " + info.testMsg}
            Text {text: Arn.info} // ArnLib version info
        }

        function setTestMsg( msg) {
            info.testMsg = msg;
        }
    }
}

```

Definition at line 183 of file ArnQml.hpp.

14.44.2 Member Function Documentation

14.44.2.1 arnRootPath()

QString ArnQml::arnRootPath () [static]

Gives current ARN root path for all qml instances.

Returns

the root path

See also

[setArnRootPath](#)

Definition at line 60 of file ArnQml.cpp.

14.44.2.2 instance()

```
ArnQml & ArnQml::instance ( ) [static]
```

Definition at line 119 of file ArnQml.cpp.

14.44.2.3 setArnRootPath()

```
void ArnQml::setArnRootPath (
    const QString & path ) [static]
```

Change ARN root path for all qml instances.

This is set once in the application and must be set before any qml instances are setup.

Example: `setArnRootPath("/@myHost/");` will map a path `"/Test/value"` in Qml to an ARN object at path `"/@myHost/Test/value"`.

Parameters

in	<i>path</i>	is the root path
----	-------------	------------------

See also

[arnRootPath](#)

Definition at line 66 of file ArnQml.cpp.

14.44.2.4 setup()

```
void ArnQml::setup (
    QML_ENGINE * qmlEngine,
    ArnQml::UseFlags flags = UseFlags::ArnLib ) [static]
```

Add ArnLib support to a Qml instance.

ArnLib module is always included.

Parameters

in	<i>qmlEngine</i>	is the qml instance engine
in	<i>flags</i>	gives the modules to include

Definition at line 82 of file ArnQml.cpp.

The documentation for this class was generated from the following files:

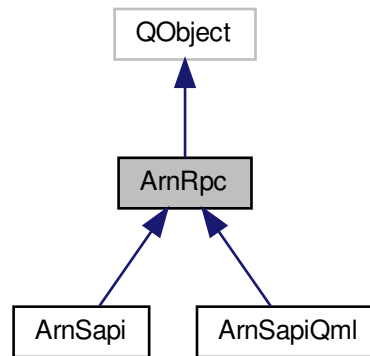
- [src/ArnInc/ArnQml.hpp \(4.0.0\)](#)
- [src/ArnQml.cpp \(4.0.0\)](#)

14.45 ArnRpc Class Reference

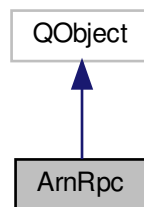
Remote Procedure Call.

```
#include <ArnRpc.hpp>
```

Inheritance diagram for ArnRpc:



Collaboration diagram for ArnRpc:



Classes

- struct [Invoke](#)

Public Types

- typedef [ArnRpcMode](#) [Mode](#)

Public Slots

- void [sendText](#) (const QString &txt)
Send a general text message to the other end of the used pipe

Signals

- void [pipeClosed](#) ()
Signal emitted when the used pipe is closed.
- void [textReceived](#) (const QString &text)
Signal emitted when a general text message is received.
- void [defaultCall](#) (const QByteArray &data)
Signal emitted when receiver method missing.
- void [outOfSequence](#) ()
Signal emitted when checked sequence order is wrong.
- void [heartBeatChanged](#) (bool isOk)
Signal emitted when Heart beat changes state.
- void [heartBeatReceived](#) ()
Signal emitted when Heart beat message is received.

Public Member Functions

- [ArnRpc](#) (QObject *parent=arnNullptr)
- [~ArnRpc](#) ()
- QString [pipePath](#) () const
Get the path for the used pipe
- bool [open](#) (const QString &[pipePath](#))
- void [setPipe](#) ([ArnPipe](#) *pipe)
Set pipe for this Rpc.
- [ArnPipe](#) * [pipe](#) () const
Get the used pipe
- bool [setReceiver](#) (QObject *receiver, bool useTrackRpcSender=true)
- QObject * [receiver](#) () const
- void [setMethodPrefix](#) (const QString &prefix)
- QString [methodPrefix](#) () const
- void [setIncludeSender](#) (bool v)
Add sender as argument when calling a rpc method.
- void [setMode](#) ([Mode](#) mode)
- [Mode](#) [mode](#) () const
Get the mode.
- void [setHeartBeatSend](#) (int time)
Set period time for sending heart beat message.
- int [getHeartBeatSend](#) () const
Get period time for sending heart beat message.
- void [setHeartBeatCheck](#) (int time)

- Set max time period for receiving heart beat message.*

 - int [getHeartBeatCheck](#) () const

Get max time period for receiving heart beat message.
- bool [isHeartBeatOk](#) () const

Get the state of heart beat.
- void [addSenderSignals](#) (QObject *sender, const QString &prefix)
- bool [invoke](#) (const QString &funcName, [MQGenericArgument](#) val0=[MQGenericArgument](#)(0), [MQGenericArgument](#) val1=[MQGenericArgument](#)(), [MQGenericArgument](#) val2=[MQGenericArgument](#)(), [MQGenericArgument](#) val3=[MQGenericArgument](#)(), [MQGenericArgument](#) val4=[MQGenericArgument](#)(), [MQGenericArgument](#) val5=[MQGenericArgument](#)(), [MQGenericArgument](#) val6=[MQGenericArgument](#)(), [MQGenericArgument](#) val7=[MQGenericArgument](#)())

Calls a named remote procedure.
- bool [invoke](#) (const QString &funcName, [Invoke](#) invokeFlags, [MQGenericArgument](#) val0=[MQGenericArgument](#)(0), [MQGenericArgument](#) val1=[MQGenericArgument](#)(), [MQGenericArgument](#) val2=[MQGenericArgument](#)(), [MQGenericArgument](#) val3=[MQGenericArgument](#)(), [MQGenericArgument](#) val4=[MQGenericArgument](#)(), [MQGenericArgument](#) val5=[MQGenericArgument](#)(), [MQGenericArgument](#) val6=[MQGenericArgument](#)(), [MQGenericArgument](#) val7=[MQGenericArgument](#)())

Calls a named remote procedure using invoke flags.
- [ArnRpc](#) * [rpcSender](#) ()
- void [batchConnect](#) (const [ARN_RegExp](#) &rgx, const QObject *receiver, const QString &replace, [Mode](#) mode=[Mode](#)())

Make batch connection from this [ArnRpc](#):s signals to another receivers slots/signals.
- void [batchConnect](#) (const QObject *sender, const [ARN_RegExp](#) &rgx, const QString &replace, [Mode](#) mode=[Mode](#)())

Make batch connection from one senders signals to this [ArnRpc](#):s slots/signals.

Static Public Member Functions

- static [ArnRpc](#) * [rpcSender](#) (QObject *receiver)
 - static void [batchConnect](#) (const QObject *sender, const [ARN_RegExp](#) &rgx, const QObject *receiver, const QString &replace, [Mode](#) mode=[Mode](#)())
- Make batch connection from one senders signals to another receivers slots/signals.*

14.45.1 Detailed Description

Remote Procedure Call.

About RPC and SAPI

This is the basic functionality of RPC. It's recommended to use [ArnSapi](#) which uses a higher level model. For now the [ArnRpc](#) class is more sparsely documented.

Example usage

```
// In class declare (MyClass)
ArnRpc* _rpcCommon;

// In class code (MyClass)
_rpcCommon = new ArnRpc( this);
_rpcCommon->setMethodPrefix("rpc_");
_rpcCommon->setReceiver( this);
_rpcCommon->setMode( ArnRpc::Mode::Provider);
_rpcCommon->open("//Pipes/pipeCommon");
.
```



```
void MyClass::rpc_test( QByteArray ba, QString str, int i)
{
    ArnRpc* sender = ArnRpc::rpcSender( this);
    if (sender) qDebug() << "RPC sender=" << sender->pipePath();
    qDebug() << "RPC-test ba=" << ba << " str=" << str << " int=" << i;
}

void MyClass::rpc_ver()
{
    ArnRpc* sender = ArnRpc::rpcSender( this);
    if (!sender) return;
    // Reply to requester the version text
    sender->invoke("ver", MQ_ARG( QString, verText, "MySystem Version 1.0"));
}
```

Definition at line 156 of file ArnRpc.hpp.

14.45.2 Member Typedef Documentation

14.45.2.1 Mode

```
typedef ArnRpcMode ArnRpc::Mode
```

Definition at line 162 of file ArnRpc.hpp.

14.45.3 Constructor & Destructor Documentation

14.45.3.1 ArnRpc()

```
ArnRpc::ArnRpc (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 204 of file ArnRpc.cpp.

14.45.3.2 ~ArnRpc()

```
ArnRpc::~ArnRpc ( )
```

Definition at line 220 of file ArnRpc.cpp.

14.45.4 Member Function Documentation

14.45.4.1 addSenderSignals()

```
void ArnRpc::addSenderSignals (
    QObject * sender,
    const QString & prefix )
```

Definition at line 440 of file ArnRpc.cpp.

14.45.4.2 batchConnect() [1/3]

```
void ArnRpc::batchConnect (
    const QObject * sender,
    const ARN_RegExp & rgx,
    const QObject * receiver,
    const QString & replace,
    Mode mode = Mode() ) [static]
```

Make batch connection from one senders signals to another receivers slots/signals.

Used when there is a pattern in the naming of the signals and slots. It's assumed that naming for slots are unique regardless of its case i.e. using both test() and tesT() are not allowed.

Example: `batchConnect(_commonSapi, ARN_RegExp("^rq_(.+)"), this, "chat\\1");`
 connects signal: `rq_info(QString,QString)` to slot: `chatInfo(QString,QString)`

Parameters

in	<i>sender</i>	is the sending QObject.
in	<i>rgx</i>	is the regular expression for selecting sender signals.
in	<i>receiver</i>	is the receiving QObject.
in	<i>replace</i>	is the conversion for naming the receiver slots/signals.
in	<i>mode</i>	Used modes: <i>Debug</i> , <i>NoDefaultArgs</i>

Definition at line 1487 of file ArnRpc.cpp.

14.45.4.3 batchConnect() [2/3]

```
void ArnRpc::batchConnect (
    const ARN_RegExp & rgx,
    const QObject * receiver,
    const QString & replace,
    Mode mode = Mode() ) [inline]
```

Make batch connection from this [ArnRpc](#):s signals to another receivers slots/signals.

Used when there is a pattern in the naming of the signals and slots. It's assumed that naming for slots are unique regardless of its case i.e. using both test() and tesT() are not allowed.

Example: `_commonSapi.batchConnect(ARN_RegExp("^rq_(.+)"), this, "chat\\1");`
 connects signal: `rq_info(QString,QString)` to slot: `chatInfo(QString,QString)`

Parameters

in	<i>rgx</i>	is the regular expression for selecting sender signals.
in	<i>receiver</i>	is the receiving QObject.
in	<i>replace</i>	is the conversion for naming the receiver slots/signals.
in	<i>mode</i>	

See also

[batchConnect](#)(const QObject*, const [ARN_Regex](#)&, const QObject*, const QString&, [Mode](#))

Definition at line 348 of file ArnRpc.hpp.

14.45.4.4 batchConnect() [3/3]

```
void ArnRpc::batchConnect (
    const QObject * sender,
    const ARN_Regex & rgx,
    const QString & replace,
    Mode mode = Mode() ) [inline]
```

Make batch connection from one senders signals to this [ArnRpc](#):s slots/signals.

Used when there is a pattern in the naming of the signals and slots. It's assumed that naming for slots are unique regardless of its case i.e. using both test() and testT() are not allowed.

Example: `_commonSapi.batchConnect(_commonSapi, ARN_Regex("^chat(.+)"), "rq_↵_\\1");` connects signal: chatinfo(QString,QString) to slot: rq_Info(QString,QString)

Parameters

in	<i>sender</i>	is the sending QObject.
in	<i>rgx</i>	is the regular expression for selecting sender signals.
in	<i>replace</i>	is the conversion for naming the receiver slots/signals.
in	<i>mode</i>	

See also

[batchConnect](#)(const QObject*, const [ARN_Regex](#)&, const QObject*, const QString&, [Mode](#))

Definition at line 369 of file ArnRpc.hpp.

14.45.4.5 defaultCall

```
void ArnRpc::defaultCall (
    const QByteArray & data ) [signal]
```

Signal emitted when receiver method missing.

This signal is only emitted if `Mode::useDefaultCall` is active. Error notification is then canceled.

Parameters

<code>in</code>	<code>data</code>	is the received call message in XString format.
-----------------	-------------------	---

14.45.4.6 getHeartBeatCheck()

```
int ArnRpc::getHeartBeatCheck ( ) const
```

Get max time period for receiving heart beat message.

Time zero is turned off checking.

Returns

time is the time period in seconds

See also

[setHeartBeatCheck\(\)](#)

Definition at line 424 of file ArnRpc.cpp.

14.45.4.7 getHeartBeatSend()

```
int ArnRpc::getHeartBeatSend ( ) const
```

Get period time for sending heart beat message.

Time zero is turned off sending.

Returns

time is the time period in seconds

See also

[setHeartBeatSend\(\)](#)

Definition at line 405 of file ArnRpc.cpp.

14.45.4.8 heartBeatChanged

```
void ArnRpc::heartBeatChanged (
    bool isOk ) [signal]
```

Signal emitted when Heart beat changes state.

Heart beat messages are detected and expected within a check time. If this is satisfied, the state of heart beat is ok.

Parameters

in	<i>isOk</i>	is the Heart beat state, false = Not received.
----	-------------	--

14.45.4.9 heartBeatReceived

```
void ArnRpc::heartBeatReceived ( ) [signal]
```

Signal emitted when Heart beat message is received.

14.45.4.10 invoke() [1/2]

```
bool ArnRpc::invoke (
    const QString & funcName,
    MQGenericArgument val0 = MQGenericArgument (0),
    MQGenericArgument val1 = MQGenericArgument (),
    MQGenericArgument val2 = MQGenericArgument (),
    MQGenericArgument val3 = MQGenericArgument (),
    MQGenericArgument val4 = MQGenericArgument (),
    MQGenericArgument val5 = MQGenericArgument (),
    MQGenericArgument val6 = MQGenericArgument (),
    MQGenericArgument val7 = MQGenericArgument () )
```

Calls a named remote procedure.

This is the low level way to call a remote procedure. It can freely call anything without declaring it. For high level calls use [ArnSapi](#).

This function works similar to `QMetaObject::invokeMethod()`. The called name is prefixed before the final call is made. Using the label in [MQ_ARG\(\)](#) makes debugging easier, as the parameter is named.

Example: `rpc->invoke("myfunc", MQ_ARG(QString, mypar, "Test XYZ"));`

Parameters

in	<i>funcName</i>	is the name of the called procedure.
in	<i>val0</i>	first arg.
in	<i>val1</i>	
in	<i>val2</i>	
in	<i>val3</i>	
in	<i>val4</i>	
in	<i>val5</i>	
in	<i>val6</i>	
in	<i>val7</i>	

Definition at line 494 of file ArnRpc.cpp.

14.45.4.11 invoke() [2/2]

```
bool ArnRpc::invoke (
    const QString & funcName,
    Invoke invokeFlags,
    MQGenericArgument val0 = MQGenericArgument (0),
    MQGenericArgument val1 = MQGenericArgument (),
    MQGenericArgument val2 = MQGenericArgument (),
    MQGenericArgument val3 = MQGenericArgument (),
    MQGenericArgument val4 = MQGenericArgument (),
    MQGenericArgument val5 = MQGenericArgument (),
    MQGenericArgument val6 = MQGenericArgument (),
    MQGenericArgument val7 = MQGenericArgument () )
```

Calls a named remote procedure using invoke flags.

This is the low level way to call a remote procedure. It can freely call anything without declaring it. For high level calls use [ArnSapi](#).

This function works similar to `QMetaObject::invokeMethod()`. The called name is prefixed before the final call is made. Using the label in [MQ_ARG\(\)](#) makes debugging easier, as the parameter is named.

Example: `rpc->invoke("myfunc", ArnRpc::Invoke::NoQueue, MQ_ARG(QString, mypar, "Test XYZ"));`

Parameters

in	<i>funcName</i>	is the name of the called procedure.
in	<i>invokeFlags</i>	is flags for controlling the invoke
in	<i>val0</i>	first arg.
in	<i>val1</i>	
in	<i>val2</i>	
in	<i>val3</i>	
in	<i>val4</i>	
in	<i>val5</i>	
in	<i>val6</i>	
in	<i>val7</i>	

Definition at line 533 of file ArnRpc.cpp.

14.45.4.12 isHeartBeatOk()

```
bool ArnRpc::isHeartBeatOk ( ) const
```

Get the state of heart beat.

Return values

<i>false</i>	if not getting heart beat in time
--------------	-----------------------------------

See also

[heartBeatChanged\(\)](#)

Definition at line 432 of file ArnRpc.cpp.

14.45.4.13 methodPrefix()

```
QString ArnRpc::methodPrefix ( ) const
```

Definition at line 346 of file ArnRpc.cpp.

14.45.4.14 mode()

```
ArnRpc::Mode ArnRpc::mode ( ) const
```

Get the mode.

Returns

current *mode*

Definition at line 386 of file ArnRpc.cpp.

14.45.4.15 open()

```
bool ArnRpc::open (
    const QString & pipePath )
```

Definition at line 236 of file ArnRpc.cpp.

14.45.4.16 outOfSequence

```
void ArnRpc::outOfSequence ( ) [signal]
```

Signal emitted when checked sequence order is wrong.

14.45.4.17 pipe()

```
ArnPipe * ArnRpc::pipe ( ) const
```

Get the used *pipe*

Returns

pipe

See also

[setPipe\(\)](#)

Definition at line 292 of file ArnRpc.cpp.

14.45.4.18 pipeClosed

```
void ArnRpc::pipeClosed ( ) [signal]
```

Signal emitted when the used pipe is closed.

The *pipe* closes when its *Arn Data Object* is destroyed, i.e. the session is considered ended.

14.45.4.19 pipePath()

```
QString ArnRpc::pipePath ( ) const
```

Get the path for the used *pipe*

Returns

path

Definition at line 226 of file ArnRpc.cpp.

14.45.4.20 receiver()

```
QObject * ArnRpc::receiver ( ) const
```

Definition at line 329 of file ArnRpc.cpp.

14.45.4.21 `rpcSender()` [1/2]

```
ArnRpc * ArnRpc::rpcSender ( )
```

Definition at line 473 of file `ArnRpc.cpp`.

14.45.4.22 `rpcSender()` [2/2]

```
ArnRpc * ArnRpc::rpcSender (
    QObject * receiver ) [static]
```

Definition at line 483 of file `ArnRpc.cpp`.

14.45.4.23 `sendText`

```
void ArnRpc::sendText (
    const QString & txt ) [slot]
```

Send a general text message to the other end of the used *pipe*

Is used by [ArnRpc](#) to give errors and help messages, mostly for debugging.

Parameters

in	<i>txt</i>	is the text to be sent
----	------------	------------------------

See also

[textReceived\(\)](#);

Definition at line 1466 of file `ArnRpc.cpp`.

14.45.4.24 `setHeartBeatCheck()`

```
void ArnRpc::setHeartBeatCheck (
    int time )
```

Set max time period for receiving heart beat message.

Setting time to zero will turn off checking.

Parameters

in	<i>time</i>	is the time period in seconds
----	-------------	-------------------------------

See also

[setHeartBeatSend\(\)](#);

Definition at line 413 of file ArnRpc.cpp.

14.45.4.25 setHeartBeatSend()

```
void ArnRpc::setHeartBeatSend (
    int time )
```

Set period time for sending heart beat message.

Setting time to zero will turn off sending.

Parameters

in	<i>time</i>	is the time period in seconds
----	-------------	-------------------------------

See also

[setHeartBeatCheck\(\)](#)

Definition at line 394 of file ArnRpc.cpp.

14.45.4.26 setIncludeSender()

```
void ArnRpc::setIncludeSender (
    bool v )
```

Add sender as argument when calling a rpc method.

Deprecated Use [rpcSender\(\)](#)

Definition at line 370 of file ArnRpc.cpp.

14.45.4.27 setMethodPrefix()

```
void ArnRpc::setMethodPrefix (
    const QString & prefix )
```

Definition at line 337 of file ArnRpc.cpp.

14.45.4.28 setMode()

```
void ArnRpc::setMode (
    Mode mode )
```

Definition at line 378 of file ArnRpc.cpp.

14.45.4.29 setPipe()

```
void ArnRpc::setPipe (
    ArnPipe * pipe )
```

Set pipe for this Rpc.

The Rpc will take ownership of the pip.

Parameters

in	<i>pipe</i>	
----	-------------	--

See also

[pipe\(\)](#)
[pipePath\(\)](#)

Definition at line 273 of file ArnRpc.cpp.

14.45.4.30 setReceiver()

```
bool ArnRpc::setReceiver (
    QObject * receiver,
    bool useTrackRpcSender = true )
```

Definition at line 300 of file ArnRpc.cpp.

14.45.4.31 textReceived

```
void ArnRpc::textReceived (
    const QString & text ) [signal]
```

Signal emitted when a general text message is received.

The text message is received from the other end of the used *pipe*.

Parameters

in	<i>text</i>	is the received text
----	-------------	----------------------

See also

[sendText\(\);](#)

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnRpc.hpp \(4.0.0\)](#)
- [src/ArnRpc.cpp \(4.0.0\)](#)

14.46 ArnRpcMode Class Reference

```
#include <ArnRpc.hpp>
```

Public Types

- enum `E` {
[Provider](#) = 0x0001, [AutoDestroy](#) = 0x0002, [UuidPipe](#) = 0x0004, [NoDefaultArgs](#) = 0x0008,
[SendSequence](#) = 0x0010, [CheckSequence](#) = 0x0020, [OnlyPosArgIn](#) = 0x0040, [NamedArg](#) = 0x0080,
[NamedTypedArg](#) = 0x0100, [UseDefaultCall](#) = 0x0200, [Debug](#) = 0x8000, [UuidAutoDestroy](#) = UuidPipe | Auto↔
Destroy,
[AnyNamedArg](#) = NamedArg | NamedTypedArg }

14.46.1 Detailed Description

Examples:

[ArnDemoChatServer/MainWindow.cpp](#).

Definition at line 85 of file ArnRpc.hpp.

14.46.2 Member Enumeration Documentation

Enumerator

14.46.2.1 E

```
enum ArnRpcMode::E
```

Enumerator

Provider	Provider side (opposed to requester)
AutoDestroy	Use <i>AutoDestroy</i> for the pipe, i.e. it is closed when tcp/ip is broken.
UuidPipe	Use an unique uuid in the pipe name.
NoDefaultArgs	If guaranteed no default arguments, full member name overload is ok.
SendSequence	Send sequence order information to pipe.
CheckSequence	Check sequence order information from pipe. Can generate signal <code>outOfSequence()</code> .
OnlyPosArgIn	Only allow calling in with positional argument (typed)
NamedArg	When calling out, uses named argument e.g "myFunc count=123".
NamedTypedArg	When calling out, uses named argument with type e.g "myFunc count:int=123".
UseDefaultCall	When receiver method missing, send <code>defaultCall()</code> signal instead of error.
Debug	Debug mode, dumping info for the batch connections.
UuidAutoDestroy	Convenience, combined <i>UuidPipe</i> and <i>AutoDestroy</i>
AnyNamedArg	Convenience, combined <i>NamedArg</i> and <i>NamedTypedArg</i>

Definition at line 89 of file `ArnRpc.hpp`.

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

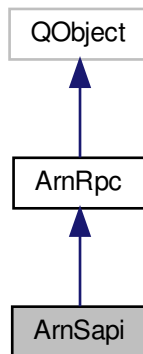
- [src/ArnInc/ArnRpc.hpp \(4.0.0\)](#)

14.47 ArnSapi Class Reference

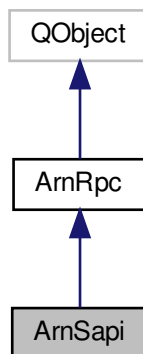
Service API.

```
#include <ArnSapi.hpp>
```

Inheritance diagram for ArnSapi:



Collaboration diagram for ArnSapi:



Public Member Functions

- [ArnSapi](#) (QObject *parent=arnNullptr)
- bool [open](#) (const QString &pipePath=QString(), [Mode mode=Mode\(\)](#), const char *providerPrefix=arnNullptr, const char *requesterPrefix=arnNullptr)
Open a new Service API.
- void [batchConnectTo](#) (const QObject *receiver, const QString &prefix=QString(), [Mode mode=Mode\(\)](#))
Make batch connection from this [ArnSapi:s](#) signals to another receivers slots/signals.
- void [batchConnectFrom](#) (const QObject *sender, const QString &prefix=QString(), [Mode mode=Mode\(\)](#))
Make batch connection from one senders signals to this [ArnSapi:s](#) signals.
- QString [defaultPath](#) () const
Get default path for the pipe to be used.

Protected Member Functions

- [ArnSapi](#) (const QString &defaultPath, QObject *parent=arnNullptr)
- void [setDefaultPath](#) (const QString &defaultPath)

Set default path for the pipe to be used.

Additional Inherited Members

14.47.1 Detailed Description

Service API.

About RPC and SAPI

This class serves as a base class for *Service Application Programming Interface*. It should be derived to a custom class that describe a specific *SAPI*.

By default all *provider* services are prefixed by "pv_" and all *requester* "services" are prefixed by "rq_". This standard can be changed.

The meta prefix *no_queue* is used to limit the filling of sendqueue with recurring RPC calls during some kind of client disconnection. Matched function name in sendqueue is overwritten by the last call. This functionality uses [pipe anti congest](#). This is internally used for *heart beat*, but other typical usages can be *ping*, *request update* etc.

Example usage

```
class ChatSapi : public ArnSapi
{
    Q_OBJECT
public:
    explicit ChatSapi( QObject* parent = arnNullptr) : ArnSapi( parent) {}

signals:
MQ_PUBLIC_ACCESS
    no_queue void pv_list();
    void pv_newMsg( QString name, QString msg);
    void pv_infoQ();

    void rq_updateMsg( int seq, QString name, QString msg);
    void rq_info( QString name, QString ver);
};

// In class declare (MyClass)
ChatSapi* _commonSapi;

// In class code (MyClass)
typedef ArnSapi::Mode SMode;
_commonSapi = new ChatSapi( this);
_commonSapi->open("//Chat/Pipes/pipeCommon", SMode::Provider | SMode::UseDefaultCall);
_commonSapi->batchConnectTo( this, "sapi");
.
.
void ServerMain::sapiNewMsg( QString name, QString msg)
{
    int seq = ...;
    _commonSapi->rq_updateMsg( seq, name, msg);
}

void MyClass::sapiInfoQ()
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    sapi->rq_info("Arn Chat Demo", "1.0");
}

void MainWindow::sapiDefault( const QByteArray& data)
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    qDebug() << "chatDefault:" << data;
    sapi->sendText("Chat Sapi: Can't find method, use $help.");
}
}
```

Examples:

[ArnDemoChatServer/ChatSapi.hpp](#).

Definition at line 115 of file ArnSapi.hpp.

14.47.2 Constructor & Destructor Documentation

14.47.2.1 ArnSapi() [1/2]

```
ArnSapi::ArnSapi (
    QObject * parent = arnNullptr ) [explicit]
```

Examples:

[ArnDemoChatServer/ChatSapi.hpp](#).

Definition at line 47 of file ArnSapi.cpp.

14.47.2.2 ArnSapi() [2/2]

```
ArnSapi::ArnSapi (
    const QString & defaultPath,
    QObject * parent = arnNullptr ) [protected]
```

Definition at line 53 of file ArnSapi.cpp.

14.47.3 Member Function Documentation

14.47.3.1 batchConnectFrom()

```
void ArnSapi::batchConnectFrom (
    const QObject * sender,
    const QString & prefix = QString(),
    ArnRpc::Mode mode = Mode() )
```

Make batch connection from one senders signals to this [ArnSapi:s](#) signals.

Used when there is a specific pattern in the naming of the signals. It's assumed that naming for signals are unique regardless of its case i.e. using both test() and tesT() are not allowed.

Example: Requester doing `_commonSapi.batchConnectFrom(mySender, "sapi");` Can connect signal: `sapiNewMsg(QString,QString)` to signal: `pv_newMsg(QString,QString)`

Parameters

in	<i>sender</i>	is the sending QObject.
in	<i>prefix</i>	is the prefix for sending signal names.
in	<i>mode</i>	

See also

[ArnRpc::batchConnect](#)(const QObject*, const [ARN_RegExp](#)&, const QObject*, const QString&, [Mode](#))

Definition at line 107 of file ArnSapi.cpp.

14.47.3.2 batchConnectTo()

```
void ArnSapi::batchConnectTo (
    const QObject * receiver,
    const QString & prefix = QString(),
    ArnRpc::Mode mode = Mode() )
```

Make batch connection from this [ArnSapi](#):s signals to another receivers slots/signals.

Used when there is a specific pattern in the naming of the signals and slots. It's assumed that naming for slots are unique regardless of its case i.e. using both test() and tesT() are not allowed.

When [Mode::UseDefaultCall](#) is active, then also the [defaultCall\(\)](#) signal is connected to the receiver. Method name will be using the prefix and end with "Default". E.g. prefix is "sapi" will give method name "sapiDefault".

Example: Provider doing `_commonSapi.batchConnectTo(myReceiver, "sapi");` Can connect signal: `pv_newMsg(QString,QString)` to slot: `sapiNewMsg(QString,QString)`

Parameters

in	<i>receiver</i>	is the receiving QObject.
in	<i>prefix</i>	is the prefix for receiving slot/signal names.
in	<i>mode</i>	

See also

[ArnRpc::batchConnect](#)(const QObject*, const [ARN_RegExp](#)&, const QObject*, const QString&, [Mode](#))

Definition at line 89 of file ArnSapi.cpp.

14.47.3.3 defaultPath()

```
QString ArnSapi::defaultPath ( ) const
```

Get default path for the *pipe* to be used.

Returns

default path

Definition at line 118 of file ArnSapi.cpp.

14.47.3.4 open()

```
bool ArnSapi::open (
    const QString & pipePath = QString(),
    Mode mode = Mode(),
    const char * providerPrefix = arnNullptr,
    const char * requesterPrefix = arnNullptr )
```

Open a new Service API.

The opened Sapi can be either the *provider* side or the *requester* side, which is indicated by *mode*. The provider marker "!" in the *pipePath* will automatically be set/removed in accordance to the *mode*.

Typically the *provider* is only using *mode Provider*. The *requester* can use default *mode* for a static *pipe* and typically use the *UuidAutoDestroy mode* for dynamic session *pipes*.

Parameters

in	<i>pipePath</i>	is the path used for Sapi. Empty string gives default.
in	<i>mode</i>	
in	<i>providerPrefix</i>	to set a custom prefix for <i>provider</i> signals.
in	<i>requesterPrefix</i>	to set a custom prefix for <i>requester</i> signals.

Return values

<i>false</i>	if error
--------------	----------

See also

[Pipe Arn Data Objects](#)
[setDefaultPath\(\)](#)

Definition at line 66 of file ArnSapi.cpp.

14.47.3.5 setDefaultPath()

```
void ArnSapi::setDefaultPath (
    const QString & defaultPath ) [protected]
```

Set default path for the *pipe* to be used.

A provider path will always be converted to a non provider path.

Parameters

in	<i>defaultPath</i>	
----	--------------------	--

See also

[defaultPath\(\)](#)
[open\(\)](#)

Definition at line 126 of file ArnSapi.cpp.

The documentation for this class was generated from the following files:

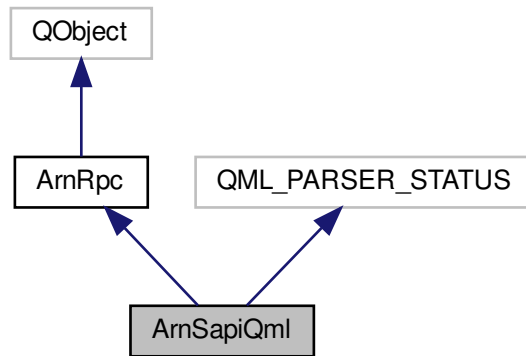
- [src/ArnInc/ArnSapi.hpp \(4.0.0\)](#)
- [src/ArnSapi.cpp \(4.0.0\)](#)

14.48 ArnSapiQml Class Reference

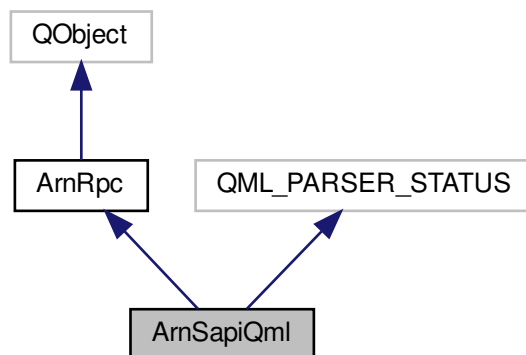
ARN Sapi QML.

```
#include <ArnQml.hpp>
```

Inheritance diagram for ArnSapiQml:



Collaboration diagram for ArnSapiQml:



Public Types

- enum [Mode](#) {
[Provider](#) = ArnRpc::Mode::Provider, [AutoDestroy](#) = ArnRpc::Mode::AutoDestroy, [UuidPipe](#) = ArnRpc::Mode::UuidPipe, [NoDefaultArgs](#) = ArnRpc::Mode::NoDefaultArgs,
[SendSequence](#) = ArnRpc::Mode::SendSequence, [CheckSequence](#) = ArnRpc::Mode::CheckSequence,
[NamedArg](#) = ArnRpc::Mode::NamedArg, [NamedTypedArg](#) = ArnRpc::Mode::NamedTypedArg,
[UseDefaultCall](#) = ArnRpc::Mode::UseDefaultCall, [UuidAutoDestroy](#) = int(UuidPipe) | int(AutoDestroy) }

Public Slots

- bool [isHeartBeatOk](#) ()

Properties

- QString [pipePath](#)
Path of the pipe for this Sapi.
- [Mode](#) [mode](#)
Sapi modes.
- QObject [receiver](#)
The receiving object of incomming Sapi calls. Default: parent.
- int [heartBeatSend](#)
Period time for sending heart beat message.
- int [heartBeatCheck](#)
Max time period for receiving heart beat message.

Additional Inherited Members

14.48.1 Detailed Description

ARN Sapi QML.

This class is the Qml version of the [ArnSapi](#).

See also

[ArnQml](#)

Example usage

```
// In Qml
//
import QtQuick 2.0
import ArnLib 1.0

Rectangle {
    width: 370; height: 400

    Item {
        id: sapiTest
        ArnSapi {
            pipePath: "//Test/pipe"
            mode: ArnSapi.NamedArg
        }
    }
}
```

```

// Provider API
signal pv_readFileTest( string fileName)

// Requester API
signal rq_test2( string par1)
function rq_test( p1) {
    console.log("rq_test: p1=" + p1);
}

Component.onCompleted: {
    sapiTest.rq_test2.connect( info.setTestMsg);
    sapiTest.pv_readFileTest("myfile");
}

Rectangle {
    id: info
    property string testMsg: ""
    anchors.bottom: parent.bottom; anchors.left: parent.left; anchors.right: parent.right
    height: 80
    Column {
        anchors.fill: parent;
        Text {text: "Msg: " + info.testMsg}
        Text {text: Arn.info} // ArnLib version info
    }

    function setTestMsg( msg) {
        info.testMsg = msg;
    }
}
}

```

Definition at line 551 of file ArnQml.hpp.

14.48.2 Member Enumeration Documentation

14.48.2.1 Mode

```
enum ArnSapiQml::Mode
```

Enumerator

Provider	Provider side (opposed to requester)
AutoDestroy	Use <i>AutoDestroy</i> for the pipe, i.e. it is closed when tcp/ip is broken.
UuidPipe	Use an unique uuid in the pipe name.
NoDefaultArgs	If guaranteed no default arguments, full member name overload is ok.
SendSequence	Send sequence order information to pipe.
CheckSequence	Check sequence order information from pipe. Can generate signal outOfSequence() .
NamedArg	When calling out, uses named argument e.g "myFunc count=123".
NamedTypedArg	When calling out, uses named argument with type e.g "myFunc count:int=123".
UseDefaultCall	When receiver method missing, send defaultCall() signal instead of error.
UuidAutoDestroy	Convenience, combined <i>UuidPipe</i> and <i>AutoDestroy</i>

Definition at line 578 of file ArnQml.hpp.

14.48.3 Member Function Documentation

14.48.3.1 isHeartBeatOk

```
bool ArnSapiQml::isHeartBeatOk ( ) [inline], [slot]
```

Definition at line 603 of file ArnQml.hpp.

14.48.4 Property Documentation

14.48.4.1 heartBeatCheck

```
int ArnSapiQml::heartBeatCheck [read], [write]
```

Max time period for receiving heart beat message.

See also

[ArnRpc::setHeartBeatCheck\(\)](#)

Definition at line 576 of file ArnQml.hpp.

14.48.4.2 heartBeatSend

```
int ArnSapiQml::heartBeatSend [read], [write]
```

Period time for sending heart beat message.

See also

[ArnRpc::setHeartBeatSend\(\)](#)

Definition at line 571 of file ArnQml.hpp.

14.48.4.3 mode

```
Mode ArnSapiQml::mode [read], [write]
```

Sapi modes.

Definition at line 564 of file ArnQml.hpp.

14.48.4.4 pipePath

```
QString ArnSapiQml::pipePath [read], [write]
```

Path of the pipe for this Sapi.

Definition at line 562 of file ArnQml.hpp.

14.48.4.5 receiver

```
QObject ArnSapiQml::receiver [read], [write]
```

The receiving object of incoming Sapi calls. Default: parent.

Definition at line 566 of file ArnQml.hpp.

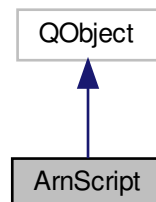
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQml.hpp \(4.0.0\)](#)
- [src/ArnQml.cpp \(4.0.0\)](#)

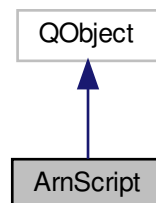
14.49 ArnScript Class Reference

```
#include <ArnScript.hpp>
```

Inheritance diagram for ArnScript:



Collaboration diagram for ArnScript:



Signals

- void [errorText](#) (QString txt)

Public Member Functions

- [ArnScript](#) (QObject *parent=arnNullptr)
- [ArnScript](#) (QScriptEngine *engine, QObject *parent=arnNullptr)
- QScriptEngine & [engine](#) () const
- void [addObject](#) (const QString &id, QObject *obj)
- bool [evaluate](#) (const QByteArray &script, const QString &idName, const QString &typeName=QString())
- bool [evaluateFile](#) (const QString &fileName)
- [ARN_JSVALUE](#) [globalProperty](#) (const QString &id)
- [ARN_JSVALUE](#) [callFunc](#) ([ARN_JSVALUE](#) &func, const [ARN_JSVALUE](#) &thisObj, const [ARN_JSVALUE_LIST](#) &args)
- bool [logUncaughtError](#) (QScriptValue &scriptValue, const QString &typeName=QString())
- QString [idName](#) () const
- void [setInterruptedText](#) (const QString &interruptedText)

Protected Member Functions

- void [errorLog](#) (const QString &errText, [ArnError](#) err=[ArnError::Undef](#), void *reference=arnNullptr)

Static Protected Member Functions

- static QScriptValue [printFunction](#) (QScriptContext *context, QScriptEngine *engine)

Protected Attributes

- QScriptEngine * [_engine](#)
- ArnItemProto * [_itemProto](#)
- ArnMonitorProto * [_monitorProto](#)
- ArnDepOfferProto * [_depOfferProto](#)
- ArnDepProto * [_depProto](#)

14.49.1 Detailed Description

Definition at line 405 of file ArnScript.hpp.

14.49.2 Constructor & Destructor Documentation

14.49.2.1 ArnScript() [1/2]

```
ArnScript::ArnScript (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 482 of file ArnScript.cpp.

14.49.2.2 ArnScript() [2/2]

```
ArnScript::ArnScript (
    QScriptEngine * engine,
    QObject * parent = arnNullptr )
```

Definition at line 489 of file ArnScript.cpp.

14.49.3 Member Function Documentation

14.49.3.1 addObject()

```
void ArnScript::addObject (
    const QString & id,
    QObject * obj )
```

Definition at line 502 of file ArnScript.cpp.

14.49.3.2 callFunc()

```
QScriptValue ArnScript::callFunc (
    ARN_JSVALUE & func,
    const ARN_JSVALUE & thisObj,
    const ARN_JSVALUE_LIST & args )
```

Definition at line 541 of file ArnScript.cpp.

14.49.3.3 engine()

```
QScriptEngine & ArnScript::engine ( ) const
```

Definition at line 496 of file ArnScript.cpp.

14.49.3.4 errorLog()

```
void ArnScript::errorLog (
    const QString & errText,
    ArnError err = ArnError::Undef,
    void * reference = arnNullptr ) [protected]
```

Definition at line 650 of file ArnScript.cpp.

14.49.3.5 errorText

```
void ArnScript::errorText (
    QString txt ) [signal]
```

14.49.3.6 evaluate()

```
bool ArnScript::evaluate (
    const QByteArray & script,
    const QString & idName,
    const QString & typeName = QString() )
```

Definition at line 515 of file ArnScript.cpp.

14.49.3.7 evaluateFile()

```
bool ArnScript::evaluateFile (
    const QString & fileName )
```

Definition at line 526 of file ArnScript.cpp.

14.49.3.8 globalProperty()

```
QScriptValue ArnScript::globalProperty (
    const QString & id )
```

Definition at line 535 of file ArnScript.cpp.

14.49.3.9 idName()

```
QString ArnScript::idName ( ) const
```

Definition at line 568 of file ArnScript.cpp.

14.49.3.10 logUncaughtError()

```
bool ArnScript::logUncaughtError (
    QScriptValue & scriptValue,
    const QString & typeName = QString() )
```

Definition at line 550 of file ArnScript.cpp.

14.49.3.11 printFunction()

```
QScriptValue ArnScript::printFunction (
    QScriptContext * context,
    QScriptEngine * engine ) [static], [protected]
```

Definition at line 582 of file ArnScript.cpp.

14.49.3.12 setInterruptedText()

```
void ArnScript::setInterruptedText (
    const QString & interruptedText )
```

14.49.4 Member Data Documentation

14.49.4.1 _depOfferProto

```
ArnDepOfferProto* ArnScript::_depOfferProto [protected]
```

Definition at line 439 of file ArnScript.hpp.

14.49.4.2 _depProto

```
ArnDepProto* ArnScript::_depProto [protected]
```

Definition at line 440 of file ArnScript.hpp.

14.49.4.3 _engine

```
QScriptEngine* ArnScript::_engine [protected]
```

Definition at line 436 of file ArnScript.hpp.

14.49.4.4 _itemProto

```
ArnItemProto* ArnScript::_itemProto [protected]
```

Definition at line 437 of file ArnScript.hpp.

14.49.4.5 _monitorProto

```
ArnMonitorProto* ArnScript::_monitorProto [protected]
```

Definition at line 438 of file ArnScript.hpp.

The documentation for this class was generated from the following files:

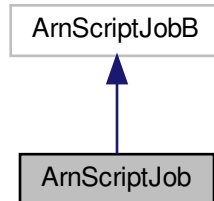
- [src/ArnInc/ArnScript.hpp \(4.0.0\)](#)
- [src/ArnScript.cpp \(4.0.0\)](#)

14.50 ArnScriptJob Class Reference

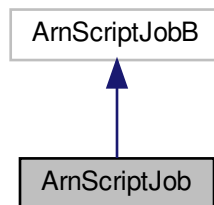
Interface class to be normally used, is also Script Job interface.

```
#include <ArnScriptJob.hpp>
```

Inheritance diagram for ArnScriptJob:



Collaboration diagram for ArnScriptJob:



Public Slots

- void [setWatchDogTime](#) (int time)
- void [yield](#) ()
- void [quit](#) ()
- void [errorLog](#) (const QString &txt)

Signals

- void [sigQuit](#) ()

Public Member Functions

- [ArnScriptJob](#) (int id, QObject *parent=arnNullptr)

Properties

- bool [sleepState](#)
- bool [running](#)
- int [watchDog](#)
- int [poll](#)
- QString [name](#)

14.50.1 Detailed Description

Interface class to be normally used, is also Script Job interface.

Definition at line 153 of file ArnScriptJob.hpp.

14.50.2 Constructor & Destructor Documentation

14.50.2.1 ArnScriptJob()

```
ArnScriptJob::ArnScriptJob (
    int id,
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 499 of file ArnScriptJob.cpp.

14.50.3 Member Function Documentation

14.50.3.1 errorLog

```
void ArnScriptJob::errorLog (
    const QString & txt ) [inline], [slot]
```

Definition at line 171 of file ArnScriptJob.hpp.

14.50.3.2 quit

```
void ArnScriptJob::quit ( ) [inline], [slot]
```

Definition at line 170 of file ArnScriptJob.hpp.

14.50.3.3 setWatchDogTime

```
void ArnScriptJob::setWatchDogTime (
    int time ) [inline], [slot]
```

Definition at line 168 of file ArnScriptJob.hpp.

14.50.3.4 sigQuit

```
void ArnScriptJob::sigQuit ( ) [signal]
```

14.50.3.5 yield

```
void ArnScriptJob::yield ( ) [inline], [slot]
```

Definition at line 169 of file ArnScriptJob.hpp.

14.50.4 Property Documentation

14.50.4.1 name

```
QString ArnScriptJob::name [read]
```

Definition at line 160 of file ArnScriptJob.hpp.

14.50.4.2 poll

```
int ArnScriptJob::poll [read], [write]
```

Definition at line 159 of file ArnScriptJob.hpp.

14.50.4.3 running

```
bool ArnScriptJob::running [read]
```

Definition at line 157 of file ArnScriptJob.hpp.

14.50.4.4 sleepState

```
bool ArnScriptJob::sleepState [read], [write]
```

Definition at line 156 of file ArnScriptJob.hpp.

14.50.4.5 watchDog

```
int ArnScriptJob::watchDog [read], [write]
```

Definition at line 158 of file ArnScriptJob.hpp.

The documentation for this class was generated from the following files:

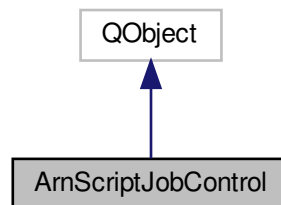
- [src/ArnInc/ArnScriptJob.hpp \(4.0.0\)](#)
- [src/ArnScriptJob.cpp \(4.0.0\)](#)

14.51 ArnScriptJobControl Class Reference

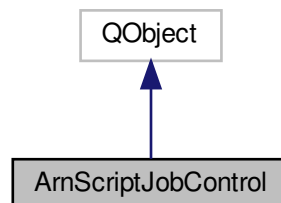
Is thread-safe (except doSetupJob)

```
#include <ArnScriptJob.hpp>
```

Inheritance diagram for ArnScriptJobControl:



Collaboration diagram for ArnScriptJobControl:



Public Slots

- void [setScript](#) (const QByteArray &script)

Signals

- void [scriptChanged](#) (int id)
- void [errorText](#) (const QString &txt)

Public Member Functions

- [ArnScriptJobControl](#) (QObject *parent=arnNullptr)
- int [id](#) ()
- QString [name](#) () const
- void [setName](#) (const QString &name)
- void [addInterface](#) (const QString &id)
- void [addInterfaceList](#) (const QStringList &interfaceList)
- QByteArray [script](#) () const
- void [loadScriptFile](#) (const QString &fileName)
- QVariant [config](#) (const char *name) const
- bool [setConfig](#) (const char *name, const QVariant &value)
- void [addConfig](#) (QObject *obj)
- void [setThreaded](#) (bool isThreaded)
- void [doSetupJob](#) ([ArnScriptJob](#) *job, [ArnScriptJobFactory](#) *jobFactory)

Not threadsafe, only run in same thread as script.

14.51.1 Detailed Description

Is thread-safe (except doSetupJob)

Definition at line 191 of file ArnScriptJob.hpp.

14.51.2 Constructor & Destructor Documentation

14.51.2.1 ArnScriptJobControl()

```
ArnScriptJobControl::ArnScriptJobControl (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 512 of file ArnScriptJob.cpp.

14.51.3 Member Function Documentation

14.51.3.1 addConfig()

```
void ArnScriptJobControl::addConfig (
    QObject * obj )
```

Definition at line 611 of file ArnScriptJob.cpp.

14.51.3.2 addInterface()

```
void ArnScriptJobControl::addInterface (
    const QString & id )
```

Definition at line 549 of file ArnScriptJob.cpp.

14.51.3.3 addInterfaceList()

```
void ArnScriptJobControl::addInterfaceList (
    const QStringList & interfaceList )
```

Definition at line 558 of file ArnScriptJob.cpp.

14.51.3.4 config()

```
QVariant ArnScriptJobControl::config (
    const char * name ) const
```

Definition at line 645 of file ArnScriptJob.cpp.

14.51.3.5 doSetupJob()

```
void ArnScriptJobControl::doSetupJob (
    ArnScriptJob * job,
    ArnScriptJobFactory * jobFactory )
```

Not threadsafe, only run in same thread as script.

Definition at line 629 of file ArnScriptJob.cpp.

14.51.3.6 errorText

```
void ArnScriptJobControl::errorText (
    const QString & txt ) [signal]
```

14.51.3.7 id()

```
int ArnScriptJobControl::id ( )
```

Definition at line 529 of file ArnScriptJob.cpp.

14.51.3.8 loadScriptFile()

```
void ArnScriptJobControl::loadScriptFile (
    const QString & fileName )
```

Definition at line 587 of file ArnScriptJob.cpp.

14.51.3.9 name()

```
QString ArnScriptJobControl::name ( ) const
```

Definition at line 539 of file ArnScriptJob.cpp.

14.51.3.10 script()

```
QByteArray ArnScriptJobControl::script ( ) const
```

Definition at line 577 of file ArnScriptJob.cpp.

14.51.3.11 scriptChanged

```
void ArnScriptJobControl::scriptChanged (
    int id ) [signal]
```

14.51.3.12 setConfig()

```
bool ArnScriptJobControl::setConfig (
    const char * name,
    const QVariant & value )
```

Definition at line 599 of file ArnScriptJob.cpp.

14.51.3.13 setName()

```
void ArnScriptJobControl::setName (
    const QString & name )
```

Definition at line 521 of file ArnScriptJob.cpp.

14.51.3.14 setScript

```
void ArnScriptJobControl::setScript (
    const QByteArray & script ) [slot]
```

Definition at line 567 of file ArnScriptJob.cpp.

14.51.3.15 setThreaded()

```
void ArnScriptJobControl::setThreaded (
    bool isThreaded )
```

Definition at line 622 of file ArnScriptJob.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnScriptJob.hpp \(4.0.0\)](#)
- [src/ArnScriptJob.cpp \(4.0.0\)](#)

14.52 ArnScriptJobFactory Class Reference

Must be thread-safe as subclassed.

```
#include <ArnScriptJob.hpp>
```

Public Member Functions

- [ArnScriptJobFactory](#) ()
- virtual [~ArnScriptJobFactory](#) ()
- virtual bool [installExtension](#) (const QString &id, [ARN_JSENGINE](#) &engine, const [ArnScriptJobControl](#) *job↔
Control=arnNullptr)=0

Static Protected Member Functions

- static void [setupJsObj](#) (const QString &id, const [ARN_JSVALUE](#) &jsObj, [ARN_JSENGINE](#) &engine)
- static bool [setupInterface](#) (const QString &id, QObject *interface, [ARN_JSENGINE](#) &engine)

14.52.1 Detailed Description

Must be thread-safe as subclassed.

Definition at line 176 of file ArnScriptJob.hpp.

14.52.2 Constructor & Destructor Documentation

14.52.2.1 ArnScriptJobFactory()

```
ArnScriptJobFactory::ArnScriptJobFactory ( ) [explicit]
```

Definition at line 390 of file ArnScriptJob.cpp.

14.52.2.2 ~ArnScriptJobFactory()

```
ArnScriptJobFactory::~ArnScriptJobFactory ( ) [virtual]
```

Definition at line 395 of file ArnScriptJob.cpp.

14.52.3 Member Function Documentation

14.52.3.1 installExtension()

```
virtual bool ArnScriptJobFactory::installExtension (
    const QString & id,
    ARN\_JSENGINE & engine,
    const ArnScriptJobControl * jobControl = arnNullptr ) [pure virtual]
```

14.52.3.2 setupInterface()

```
bool ArnScriptJobFactory::setupInterface (
    const QString & id,
    QObject * interface,
    ARN_JSENGINE & engine ) [static], [protected]
```

Definition at line 406 of file ArnScriptJob.cpp.

14.52.3.3 setupJsObj()

```
void ArnScriptJobFactory::setupJsObj (
    const QString & id,
    const ARN_JSVALUE & jsObj,
    ARN_JSENGINE & engine ) [static], [protected]
```

Definition at line 400 of file ArnScriptJob.cpp.

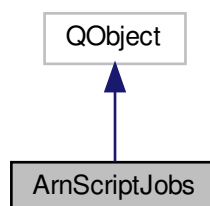
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnScriptJob.hpp \(4.0.0\)](#)
- [src/ArnScriptJob.cpp \(4.0.0\)](#)

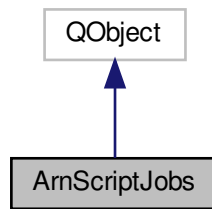
14.53 ArnScriptJobs Class Reference

```
#include <ArnScriptJobs.hpp>
```

Inheritance diagram for ArnScriptJobs:



Collaboration diagram for ArnScriptJobs:



Classes

- struct [Type](#)

Public Member Functions

- [ArnScriptJobs](#) (QObject *parent=arnNullptr)
- void [addJob](#) ([ArnScriptJobControl](#) *jobConfig, int prio=1)
- void [setFactory](#) ([ArnScriptJobFactory](#) *jobFactory)
- void [start](#) ([Type](#) type=[Type::Cooperative](#))

14.53.1 Detailed Description

TODO: Add destructor that deletes jobs in _jobSlots

Definition at line 160 of file ArnScriptJobs.hpp.

14.53.2 Constructor & Destructor Documentation

14.53.2.1 ArnScriptJobs()

```
ArnScriptJobs::ArnScriptJobs (  
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 307 of file ArnScriptJobs.cpp.

14.53.3 Member Function Documentation

14.53.3.1 addJob()

```
void ArnScriptJobs::addJob (
    ArnScriptJobControl * jobConfig,
    int prio = 1 )
```

Definition at line 318 of file ArnScriptJobs.cpp.

14.53.3.2 setFactory()

```
void ArnScriptJobs::setFactory (
    ArnScriptJobFactory * jobFactory )
```

Definition at line 330 of file ArnScriptJobs.cpp.

14.53.3.3 start()

```
void ArnScriptJobs::start (
    Type type = Type::Cooperative )
```

Definition at line 336 of file ArnScriptJobs.cpp.

The documentation for this class was generated from the following files:

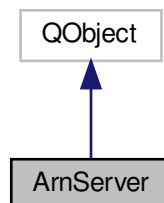
- [src/ArnInc/ArnScriptJobs.hpp \(4.0.0\)](#)
- [src/ArnScriptJobs.cpp \(4.0.0\)](#)

14.54 ArnServer Class Reference

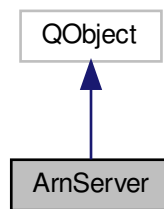
Class for making an *Arn Server*.

```
#include <ArnServer.hpp>
```

Inheritance diagram for ArnServer:



Collaboration diagram for ArnServer:



Classes

- struct [Type](#)

Public Member Functions

- [ArnServer](#) ([Type](#) serverType, `QObject *parent=arnNullptr`)
Create an [Arn](#) server object.
- [~ArnServer](#) ()
- void [start](#) (int [port](#)=-1, `QHostAddress listenAddr=QHostAddress::Any`)
Start the [Arn](#) server
- int [port](#) ()
Port number of the [Arn](#) server
- `QHostAddress` [listenAddress](#) ()
Address of the interface used to listening for connections to the [Arn](#) server
- void [addAccess](#) (const `QString` &userName, const `QString` &password, [Arn::Allow](#) allow)
Add an access entry.
- bool [isDemandLogin](#) () const
Get servers demand for login.
- void [setDemandLogin](#) (bool [isDemandLogin](#))
Set servers demand for login.
- void [setNoLoginNets](#) (const `QStringList` &[noLoginNets](#))
Set the nets not demanding login.
- `QStringList` [noLoginNets](#) () const
Get the nets not demanding login.
- bool [isDemandLoginNet](#) (const `QHostAddress` &remoteAddr) const
Return if a host address demands login.
- void [addFreePath](#) (const `QString` &path)
Add a new "freePath".
- `QStringList` [freePaths](#) () const
Returns current list of freePaths.
- void [setWhoIAm](#) (const [Arn::XStringMap](#) &whoIAmXsm)
Set servers human readable identification information.

14.54.1 Detailed Description

Class for making an *Arn* Server.

[About Sharing Arn Data Objects](#)

Example usage

```
// In class declare
ArnServer* _server;

// In class code
_server = new ArnServer( ArnServer::Type::NetSync, this);
_server->start();
```

Examples:

[ArnDemoChatServer/MainWindow.cpp](#), and [ArnDemoChatServer/MainWindow.hpp](#).

Definition at line 98 of file ArnServer.hpp.

14.54.2 Constructor & Destructor Documentation

14.54.2.1 ArnServer()

```
ArnServer::ArnServer (
    Type serverType,
    QObject * parent = arnNullptr )
```

Create an *Arn* server object.

Parameters

in	<i>serverType</i>	For now only <i>NetSync</i> is available.
in	<i>parent</i>	

Definition at line 198 of file ArnServer.cpp.

14.54.2.2 ~ArnServer()

```
ArnServer::~ArnServer ( )
```

Definition at line 212 of file ArnServer.cpp.

14.54.3 Member Function Documentation

14.54.3.1 addAccess()

```
void ArnServer::addAccess (
    const QString & userName,
    const QString & password,
    Arn::Allow allow )
```

Add an access entry.

This adds an entry to build an access table for the server. This access table restricts the operations of connected clients. Each client refer to one entry by its userName. Each entry must have a unique userName. Any equal userName is making the entry being replaced by the last added one. The password can be in clear text or a Hashed password which can be generated by [ArnClient::passwordHash\(\)](#) (see also ArnBrowser Settings).

Parameters

in	<i>userName</i>	
in	<i>password</i>	in clear text or Hashed
in	<i>allow</i>	have flags defining allowed basic operations (write, delete ...)

Definition at line 263 of file ArnServer.cpp.

14.54.3.2 addFreePath()

```
void ArnServer::addFreePath (
    const QString & path )
```

Add a new "freePath".

A freePath can be used even if not logged in to an [ArnServer](#) that demands login. Also all children below freePath is free to use. Usage is restricted to read operations and alike from [ArnServer](#) to [ArnClient](#). Setting a freePath at [ArnServer](#) gives the actual permission for read usage. All wanted freePaths must be added before [ArnServer](#) is started.

Parameters

in	<i>path</i>	is the freePath, eg "/Local/Sys/Legal".
----	-------------	---

See also

[freePaths\(\)](#)

Definition at line 363 of file ArnServer.cpp.

14.54.3.3 freePaths()

```
QStringList ArnServer::freePaths ( ) const
```

Returns current list of freePaths.

The list of freePaths is used to give permission for read uasge of the paths.

Returns

the freePath list.

See also

[addFreePath\(\)](#)

Definition at line 372 of file ArnServer.cpp.

14.54.3.4 isDemandLogin()

```
bool ArnServer::isDemandLogin ( ) const
```

Get servers demand for login.

If any of server or client demand login, it must be used.

Return values

<i>true</i>	if server demand login.
-------------	-------------------------

See also

[setDemandLogin\(\)](#)

Definition at line 271 of file ArnServer.cpp.

14.54.3.5 isDemandLoginNet()

```
bool ArnServer::isDemandLoginNet (
    const QHostAddress & remoteAddr ) const
```

Return if a host address demands login.

Parameters

<i>in</i>	<i>remoteAddr</i>	is the tested host address.
-----------	-------------------	-----------------------------

Return values

<i>false</i>	if the host address belongs to any net not demanding login
--------------	--

See also

[setNoLoginNets\(\)](#)

Definition at line 303 of file ArnServer.cpp.

14.54.3.6 listenAddress()

```
QHostAddress ArnServer::listenAddress ( )
```

Address of the interface used to listening for connections to the [Arn server](#)

Return values

<i>is</i>	the address (which usually is QHostAddress::Any).
-----------	---

See also

[start\(\)](#)

Definition at line 254 of file ArnServer.cpp.

14.54.3.7 noLoginNets()

```
QStringList ArnServer::noLoginNets ( ) const
```

Get the nets not demanding login.

Returns

the nets not demanding login.

See also

[setNoLoginNets\(\)](#)

Definition at line 295 of file ArnServer.cpp.

14.54.3.8 port()

```
int ArnServer::port ( )
```

Port number of the [Arn server](#)

Return values

<i>is</i>	the port number.
-----------	------------------

Definition at line 246 of file ArnServer.cpp.

14.54.3.9 setDemandLogin()

```
void ArnServer::setDemandLogin (
    bool isDemandLogin )
```

Set servers demand for login.

If any of server or client demand login, it must be used.

Parameters

in	<i>isDemandLogin</i>	true if server demand login.
----	----------------------	------------------------------

See also

[isDemandLogin\(\)](#)

Definition at line 279 of file ArnServer.cpp.

14.54.3.10 setNoLoginNets()

```
void ArnServer::setNoLoginNets (
    const QStringList & noLoginNets )
```

Set the nets not demanding login.

The net can be "localhost", "localnet", "any" or a subnet using syntax from `QHostAddress::parseSubnet()`. The "localnet" matches direct addresses on all of the available interfaces. The "any" will effectively turn off [setDemandLogin\(\)](#).

Parameters

in	<i>noLoginNets</i>	is the list of no login nets, e.g ("localhost" "192.168.1.0/255.255.255.0").
----	--------------------	--

See also

[noLoginNets\(\)](#)
[isDemandLoginNet\(\)](#)
[QHostAddress::parseSubnet\(\)](#)

Definition at line 287 of file ArnServer.cpp.

14.54.3.11 setWhoIam()

```
void ArnServer::setWhoIam (
    const Arn::XStringMap & whoIamXsm )
```

Set servers human readable identification information.

This is used to identify the server. Standard keys to use are: Contact, Location, Description.

Example usage

```
Arn::XStringMap xsm;
xsm.add("Contact", "arn@arnas.se");
xsm.add("Location", "The Longhouse");
xsm.add("Description", "Bring connection and integration to the people");
_arnServer->setWhoIam( xsm);
```

Parameters

in	<i>whoIamXsm</i>	contains the information.
----	------------------	---------------------------

See also

[remoteWhoIam\(\)](#)

Definition at line 404 of file ArnServer.cpp.

14.54.3.12 start()

```
void ArnServer::start (
    int port = -1,
    QHostAddress listenAddr = QHostAddress::Any )
```

Start the [Arn server](#)

Parameters

in	<i>port</i>	is the server port, -1 gives Arn::defaultTcpPort , 0 gives dynamic port
in	<i>listenAddr</i>	is the interface address to listen for connections (default any)

Definition at line 218 of file ArnServer.cpp.

The documentation for this class was generated from the following files:

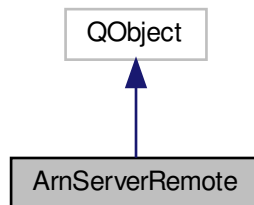
- [src/ArnInc/ArnServer.hpp \(4.0.0\)](#)
- [src/ArnServer.cpp \(4.0.0\)](#)

14.55 ArnServerRemote Class Reference

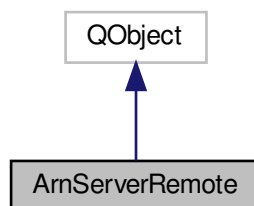
Class for remote controlling an *Arn Server*.

```
#include <ArnServerRemote.hpp>
```

Inheritance diagram for ArnServerRemote:



Collaboration diagram for ArnServerRemote:



Public Member Functions

- [ArnServerRemote](#) (QObject *parent=arnNullptr)
- [~ArnServerRemote](#) ()
- void [startUseServer](#) ([ArnServer](#) *arnServer)

Start making remote control objects for the [ArnServer](#).

14.55.1 Detailed Description

Class for remote controlling an [Arn Server](#).

About Sharing Arn Data Objects

The remote objects are available at [Arn](#) path `"/Local/Sys/Server/"`.

Example usage

```
// In class declare
ArnServer* _server;
ArnServerRemote* _serverRemote;

// In class code
_server = new ArnServer( ArnServer::Type::NetSync, this);
_server->start();
_serverRemote = new ArnServerRemote( this);
_serverRemote->startUseServer( _server);
```

Definition at line 122 of file ArnServerRemote.hpp.

14.55.2 Constructor & Destructor Documentation

14.55.2.1 ArnServerRemote()

```
ArnServerRemote::ArnServerRemote (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 310 of file ArnServerRemote.cpp.

14.55.2.2 ~ArnServerRemote()

```
ArnServerRemote::~ArnServerRemote ( )
```

Definition at line 326 of file ArnServerRemote.cpp.

14.55.3 Member Function Documentation

14.55.3.1 startUseServer()

```
void ArnServerRemote::startUseServer (
    ArnServer * arnServer )
```

Start making remote control objects for the [ArnServer](#).

The remote objects are available at [Arn](#) path `"/Local/Sys/Server/"`.

Parameters

in	<i>arnServer</i>	is the ArnServer to make remote controlled
----	------------------	--

See also

[ArnClient](#)

Definition at line 332 of file ArnServerRemote.cpp.

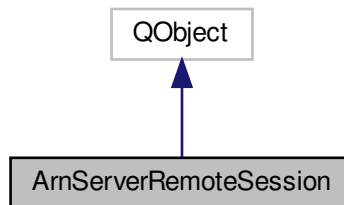
The documentation for this class was generated from the following files:

- src/ArnInc/ArnServerRemote.hpp (4.0.0)
- src/ArnServerRemote.cpp (4.0.0)

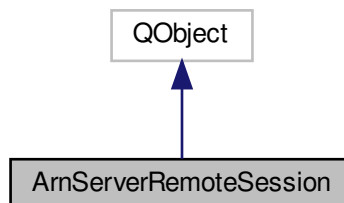
14.56 ArnServerRemoteSession Class Reference

```
#include <ArnServerRemote.hpp>
```

Inheritance diagram for ArnServerRemoteSession:



Collaboration diagram for ArnServerRemoteSession:



Public Types

- typedef [ArnServerRemoteSessionKillMode](#) [KillMode](#)

Public Member Functions

- [ArnServerRemoteSession](#) ([ArnServerSession](#) *arnServerSession, [ArnServerRemote](#) *arnServerRemote)

14.56.1 Detailed Description

Definition at line 61 of file ArnServerRemote.hpp.

14.56.2 Member Typedef Documentation

14.56.2.1 KillMode

```
typedef ArnServerRemoteSessionKillMode ArnServerRemoteSession::KillMode
```

Definition at line 65 of file ArnServerRemote.hpp.

14.56.3 Constructor & Destructor Documentation

14.56.3.1 ArnServerRemoteSession()

```
ArnServerRemoteSession::ArnServerRemoteSession (  
    ArnServerSession * arnServerSession,  
    ArnServerRemote * arnServerRemote )
```

Definition at line 44 of file ArnServerRemote.cpp.

The documentation for this class was generated from the following files:

- src/ArnInc/ArnServerRemote.hpp (4.0.0)
- src/ArnServerRemote.cpp (4.0.0)

14.57 ArnServerRemoteSessionKillMode Class Reference

```
#include <ArnServerRemote.hpp>
```

Public Types

- enum [E](#) { [Off](#), [Delay10Sec](#), [Delay60Sec](#) }

14.57.1 Detailed Description

Definition at line 48 of file ArnServerRemote.hpp.

14.57.2 Member Enumeration Documentation

14.57.2.1 E

```
enum ArnServerRemoteSessionKillMode::E
```

Enumerator

Off	
Delay10Sec	
Delay60Sec	

Definition at line 52 of file ArnServerRemote.hpp.

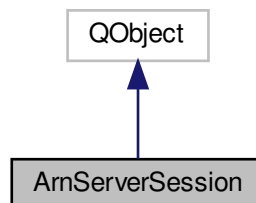
The documentation for this class was generated from the following file:

- src/ArnInc/[ArnServerRemote.hpp](#) (4.0.0)

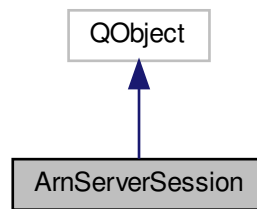
14.58 ArnServerSession Class Reference

```
#include <ArnServer.hpp>
```

Inheritance diagram for ArnServerSession:



Collaboration diagram for ArnServerSession:



Signals

- void [infoReceived](#) (int type)
- void [loginCompleted](#) ()
- void [messageReceived](#) (int type, const QByteArray &data)

Public Member Functions

- [ArnServerSession](#) (QTcpSocket *socket, ArnServer *arnServer)
- QTcpSocket * [socket](#) () const
- Arn::XStringMap [remoteWhoIAm](#) () const
- QString [loginUserName](#) () const
- Arn::Allow [getAllow](#) () const
- void [sendMessage](#) (int type, const QByteArray &data=QByteArray())
- bool [getTraffic](#) (quint64 &in, quint64 &out) const

14.58.1 Detailed Description

Definition at line 53 of file ArnServer.hpp.

14.58.2 Constructor & Destructor Documentation

14.58.2.1 ArnServerSession()

```
ArnServerSession::ArnServerSession (
    QTcpSocket * socket,
    ArnServer * arnServer )
```

Definition at line 51 of file ArnServer.cpp.

14.58.3 Member Function Documentation

14.58.3.1 getAllow()

```
Arn::Allow ArnServerSession::getAllow ( ) const
```

Definition at line 153 of file ArnServer.cpp.

14.58.3.2 getTraffic()

```
bool ArnServerSession::getTraffic (
    quint64 & in,
    quint64 & out ) const
```

Definition at line 169 of file ArnServer.cpp.

14.58.3.3 infoReceived

```
void ArnServerSession::infoReceived (
    int type ) [signal]
```

14.58.3.4 loginCompleted

```
void ArnServerSession::loginCompleted ( ) [signal]
```

14.58.3.5 loginUserName()

```
QString ArnServerSession::loginUserName ( ) const
```

Definition at line 145 of file ArnServer.cpp.

14.58.3.6 messageReceived

```
void ArnServerSession::messageReceived (
    int type,
    const QByteArray & data ) [signal]
```

14.58.3.7 remoteWhoIam()

```
Arn::XStringMap ArnServerSession::remoteWhoIAM ( ) const
```

Definition at line 137 of file ArnServer.cpp.

14.58.3.8 sendMessage()

```
void ArnServerSession::sendMessage (
    int type,
    const QByteArray & data = QByteArray() )
```

Definition at line 161 of file ArnServer.cpp.

14.58.3.9 socket()

```
QTcpSocket * ArnServerSession::socket ( ) const
```

Definition at line 131 of file ArnServer.cpp.

The documentation for this class was generated from the following files:

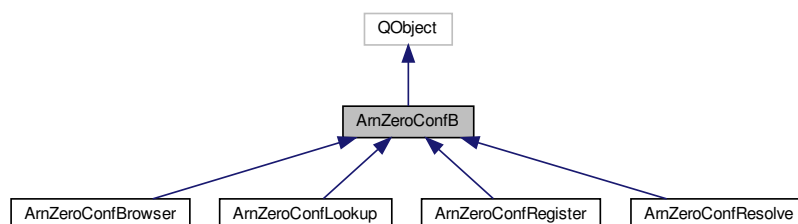
- [src/ArnInc/ArnServer.hpp \(4.0.0\)](#)
- [src/ArnServer.cpp \(4.0.0\)](#)

14.59 ArnZeroConfB Class Reference

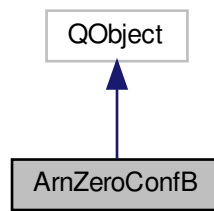
Base class for Zero Config.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfB:



Collaboration diagram for ArnZeroConfB:



Public Member Functions

- [ArnZeroConfB](#) (QObject *parent=arnNullptr)
- virtual [~ArnZeroConfB](#) ()
- QAbstractSocket::SocketType [socketType](#) () const
Returns the socket type for this Zero Config.
- void [setSocketType](#) (QAbstractSocket::SocketType type)
Sets the socket type for this Zero Config.
- QString [serviceType](#) () const
Returns the service type for this Zero Config.
- void [setServiceType](#) (const QString &type)
Returns the service type for this Zero Config.
- QString [domain](#) () const
Returns the domain for this Zero Config.
- void [setDomain](#) (const QString &domain)
Sets the domain for this Zero Config.
- [ArnZeroConf::State](#) [state](#) () const
Returns the current state of the service.
- QString [fullServiceType](#) () const
Returns the full service type for this Zero Config.

14.59.1 Detailed Description

Base class for Zero Config.

[About Zero Config](#)

This class contains methods and data which is usually a superset, i.e. not all data will be relevant / available for all uses.

Definition at line 112 of file ArnZeroConf.hpp.

14.59.2 Constructor & Destructor Documentation

14.59.2.1 ArnZeroConfB()

```
ArnZeroConfB::ArnZeroConfB (
    QObject * parent = arnNullptr )
```

Definition at line 85 of file ArnZeroConf.cpp.

14.59.2.2 ~ArnZeroConfB()

```
ArnZeroConfB::~ArnZeroConfB ( ) [virtual]
```

Definition at line 104 of file ArnZeroConf.cpp.

14.59.3 Member Function Documentation

14.59.3.1 domain()

```
QString ArnZeroConfB::domain ( ) const
```

Returns the domain for this Zero Config.

Returns

current domain.

See also

[setDomain\(\)](#)

Definition at line 295 of file ArnZeroConf.cpp.

14.59.3.2 fullServiceType()

```
QString ArnZeroConfB::fullServiceType ( ) const
```

Returns the full service type for this Zero Config.

Service types are standardized by IANA.

The full service type is the standard format used by the Zeroconf specification, e.g. "_arn._tcp".

Returns

current full service type (see above)

See also

[setServiceType\(\)](#)

Definition at line 330 of file ArnZeroConf.cpp.

14.59.3.3 serviceType()

```
QString ArnZeroConfB::serviceType ( ) const
```

Returns the service type for this Zero Config.

Returns

current service type, e.g. "arn", "ftp" ...

See also

[setServiceType\(\)](#)

Definition at line 266 of file ArnZeroConf.cpp.

14.59.3.4 setDomain()

```
void ArnZeroConfB::setDomain (
    const QString & domain )
```

Sets the domain for this Zero Config.

Default set by this class is "local".

Parameters

in	<i>domain</i>	
----	---------------	--

See also

[domain\(\)](#)

Definition at line 301 of file ArnZeroConf.cpp.

14.59.3.5 setServiceType()

```
void ArnZeroConfB::setServiceType (
    const QString & type )
```

Returns the service type for this Zero Config.

Service types are standardized by IANA.

The service type used here can be a name, like "arn", or the standard format used by the Zeroconf specification, e.g. "_arn._tcp".

Parameters

in	<i>type</i>	is the service type (se above).
----	-------------	---------------------------------

See also

[serviceType\(\)](#)

Definition at line 272 of file ArnZeroConf.cpp.

14.59.3.6 setSocketType()

```
void ArnZeroConfB::setSocketType (
    QAbstractSocket::SocketType type )
```

Sets the socket type for this Zero Config.

Allowed Socket type is: QAbstractSocket::TcpSocket, QAbstractSocket::UdpSocket.

Parameters

in	<i>type</i>	is one of the allowed types.
----	-------------	------------------------------

See also

[socketType\(\)](#)

Definition at line 260 of file ArnZeroConf.cpp.

14.59.3.7 socketType()

```
QAbstractSocket::SocketType ArnZeroConfB::socketType ( ) const
```

Returns the socket type for this Zero Config.

- Socket type can be: QAbstractSocket::TcpSocket, QAbstractSocket::UdpSocket, QAbstractSocket::↔ UnknownSocketType.
- Default set by this class is QAbstractSocket::TcpSocket.
- QAbstractSocket::UnknownSocketType is only used when socket type can't be determined.

Returns

current socket type.

See also

[setSocketType\(\)](#)

Definition at line 254 of file ArnZeroConf.cpp.

14.59.3.8 state()

```
ArnZeroConf::State ArnZeroConfB::state ( ) const
```

Returns the current state of the service.

Return values

<i>the</i>	state of the service
------------	----------------------

Definition at line 193 of file ArnZeroConf.cpp.

The documentation for this class was generated from the following files:

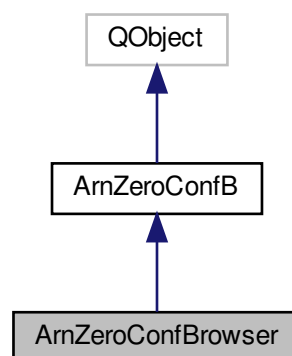
- [src/ArnInc/ArnZeroConf.hpp \(4.0.0\)](#)
- [src/ArnZeroConf.cpp \(4.0.0\)](#)

14.60 ArnZeroConfBrowser Class Reference

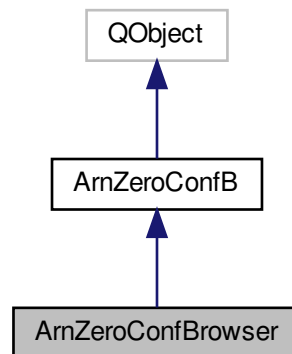
Browsing for ZeroConfig services.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfBrowser:



Collaboration diagram for ArnZeroConfBrowser:



Public Slots

- void [browse](#) (bool enable=true)
Change state of browsing.
- void [stopBrowse](#) ()
Stop browsing.

Signals

- void [serviceChanged](#) (bool isAdded, int id, const QString &serviceName, const QString &domain)
Indicate service has been added / removed.
- void [serviceAdded](#) (int id, const QString &serviceName, const QString &domain)
Indicate service has been added (discovered)
- void [serviceRemoved](#) (int id, const QString &serviceName, const QString &domain)
Indicate service has been removed.
- void [browseError](#) (int errorCode)
Indicate unsuccessfull browsing.

Public Member Functions

- [ArnZeroConfBrowser](#) (QObject *parent=arnNullptr)
Standard constructor of an [ArnZeroConfBrowser](#) object.
- [ArnZeroConfBrowser](#) (const QString &serviceType, QObject *parent=arnNullptr)
Constructor of an [ArnZeroConfBrowser](#) object.
- virtual [~ArnZeroConfBrowser](#) ()
Destructor of an [ArnZeroConfBrowser](#) object.
- void [setSubType](#) (const QString &subtype)
Set subtype (filter)
- QString [subType](#) ()

- Return current subtype (filter)*

 - QStringList [activeServiceNames](#) () const

Return current list of active service names.
- int [serviceNameTold](#) (const QString &name)
- Return the id for a service by its service name.*
- bool [isBrowsing](#) () const
- Return the status of the browsing.*

Static Public Member Functions

- static int [getNextId](#) ()
- Return the next id number for zero config objects.*

Friends

- class [ArnZeroConfIntern](#)

14.60.1 Detailed Description

Browsing for ZeroConfig services.

About Zero Config

This class handles browsing of ZeroConfig services.

Example usage

```
// In class declare
ArnZeroConfBrowser* _serviceBrowser;

// In class code
_serviceBrowser = new ArnZeroConfBrowser( this);
connect(_serviceBrowser, SIGNAL(browseError(int)),
        this, SLOT(onBrowseError(int)));
connect(_serviceBrowser, SIGNAL(serviceAdded(int,QString,QString)),
        this, SLOT(onServiceAdded(int,QString,QString)));
connect(_serviceBrowser, SIGNAL(serviceRemoved(int,QString,QString)),
        this, SLOT(onServiceRemoved(int,QString,QString)));

void XXX::onServiceAdded( int id, QString name, QString domain)
{
    ArnZeroConfResolve* ds = new ArnZeroConfResolve( name, this);
    ds->setId( id);
    connect( ds, SIGNAL(resolveError(int,int)), this, SLOT(onResolveError(int,int)));
    connect( ds, SIGNAL(resolved(int,QByteArray)), this, SLOT(onResolved(int,QByteArray)));
    ds->resolve();
}

void XXX::onServiceRemoved( int id, QString name, QString domain)
{
}
```

Definition at line 936 of file ArnZeroConf.hpp.

14.60.2 Constructor & Destructor Documentation

14.60.2.1 ArnZeroConfBrowser() [1/2]

```
ArnZeroConfBrowser::ArnZeroConfBrowser (
    QObject * parent = arnNullptr )
```

Standard constructor of an [ArnZeroConfBrowser](#) object.

All needed for browsing an "arn" service type.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 897 of file ArnZeroConf.cpp.

14.60.2.2 ArnZeroConfBrowser() [2/2]

```
ArnZeroConfBrowser::ArnZeroConfBrowser (
    const QString & serviceType,
    QObject * parent = arnNullptr )
```

Constructor of an [ArnZeroConfBrowser](#) object.

All needed parameters for browsing a service.

The service type can be a name or the standard format used by the Zeroconf specification, e.g. "_arn._tcp".

Parameters

in	<i>serviceType</i>	the service type, e.g. "arn" or "_arn._tcp".
in	<i>parent</i>	

Definition at line 904 of file ArnZeroConf.cpp.

14.60.2.3 ~ArnZeroConfBrowser()

```
ArnZeroConfBrowser::~ArnZeroConfBrowser ( ) [virtual]
```

Destructor of an [ArnZeroConfBrowser](#) object.

If browsing is active, it will be stopped.

Definition at line 912 of file ArnZeroConf.cpp.

14.60.3 Member Function Documentation

14.60.3.1 activeServiceNames()

```
QStringList ArnZeroConfBrowser::activeServiceNames ( ) const
```

Return current list of active service names.

Return values

<i>the</i>	active service names
------------	----------------------

See also[serviceAdded\(\)](#)

Definition at line 922 of file ArnZeroConf.cpp.

14.60.3.2 browse

```
void ArnZeroConfBrowser::browse (  
    bool enable = true ) [slot]
```

Change state of browsing.

When browsing is started, services will be discovered.

Parameters

in	<i>enable</i>	if true browsing is started, otherwise it is stopped
----	---------------	--

See also[stopBrowse\(\)](#)

Definition at line 954 of file ArnZeroConf.cpp.

14.60.3.3 browseError

```
void ArnZeroConfBrowser::browseError (  
    int errorCode ) [signal]
```

Indicate unsuccessful browsing.

Parameters

in	<i>errorCode</i>	
----	------------------	--

See also[browse\(\)](#)

14.60.3.4 getNextId()

```
static int ArnZeroConfBrowser::getNextId ( ) [inline], [static]
```

Return the next id number for zero config objects.

Returns

id number

Definition at line 1002 of file ArnZeroConf.hpp.

14.60.3.5 isBrowsing()

```
bool ArnZeroConfBrowser::isBrowsing ( ) const
```

Return the status of the browsing.

Return values

<i>true</i>	if browsing is started
-------------	------------------------

See also

[browse\(\)](#)

Definition at line 934 of file ArnZeroConf.cpp.

14.60.3.6 serviceAdded

```
void ArnZeroConfBrowser::serviceAdded (
    int id,
    const QString & serviceName,
    const QString & domain ) [signal]
```

Indicate service has been added (discovered)

id will not be reused for any other service, it is unique within this program.

Parameters

in	<i>id</i>	is the id number for the service
in	<i>serviceName</i>	e.g. "My House Registry"
in	<i>domain</i>	e.g. "local."

See also

[serviceRemoved\(\)](#)
[serviceChanged\(\)](#)

14.60.3.7 serviceChanged

```
void ArnZeroConfBrowser::serviceChanged (
    bool isAdded,
    int id,
    const QString & serviceName,
    const QString & domain ) [signal]
```

Indicate service has been added / removed.

id will not be reused for any other service, it is unique within this program.

Parameters

in	<i>isAdded</i>	is true when service has been added, otherwise false
in	<i>id</i>	is the id number for the service
in	<i>serviceName</i>	e.g. "My House Registry"
in	<i>domain</i>	e.g. "local."

See also

[serviceAdded\(\)](#)
[serviceRemoved\(\)](#)
[browse\(\)](#)

14.60.3.8 serviceNameToId()

```
int ArnZeroConfBrowser::serviceNameToId (
    const QString & name )
```

Return the id for a service by its service name.

Parameters

in	<i>name</i>	the service name, e.g. "My House Registry"
----	-------------	--

Returns

the id for the service

See also

[serviceAdded\(\)](#)

Definition at line 928 of file ArnZeroConf.cpp.

14.60.3.9 serviceRemoved

```
void ArnZeroConfBrowser::serviceRemoved (
    int id,
    const QString & serviceName,
    const QString & domain ) [signal]
```

Indicate service has been removed.

Parameters

in	<i>id</i>	is the id number for the service
in	<i>serviceName</i>	e.g. "My House Registry"
in	<i>domain</i>	e.g. "local."

See also

[serviceAdded\(\)](#)
[serviceChanged\(\)](#)

14.60.3.10 setSubType()

```
void ArnZeroConfBrowser::setSubType (
    const QString & subtype )
```

Set subtype (filter)

If passing empty subtype, this is taken as subtype (filter) disabled. When subtype (filter) is enabled, only services that have the same subtype is discovered.

Parameters

in	<i>subtype</i>	the filter, e.g. "myGroup1"
----	----------------	-----------------------------

See also

[subType\(\)](#)
[browse\(\)](#)
[ArnZeroConfRegister::setSubTypes\(\)](#)

Definition at line 940 of file ArnZeroConf.cpp.

14.60.3.11 stopBrowse

```
void ArnZeroConfBrowser::stopBrowse ( ) [slot]
```

Stop browsing.

See also

[browse\(\)](#)

Definition at line 988 of file ArnZeroConf.cpp.

14.60.3.12 subType()

```
QString ArnZeroConfBrowser::subType ( )
```

Return current subtype (filter)

Empty subtype, is taken as subtype (filter) disabled.

Returns

subtype, e.g. "myGroup1"

See also

[setSubType\(\)](#)

Definition at line 946 of file ArnZeroConf.cpp.

14.60.4 Friends And Related Function Documentation

14.60.4.1 ArnZeroConfIntern

```
friend class ArnZeroConfIntern [friend]
```

Definition at line 938 of file ArnZeroConf.hpp.

The documentation for this class was generated from the following files:

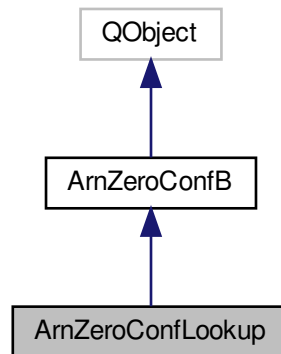
- [src/ArnInc/ArnZeroConf.hpp \(4.0.0\)](#)
- [src/ArnZeroConf.cpp \(4.0.0\)](#)

14.61 ArnZeroConfLookup Class Reference

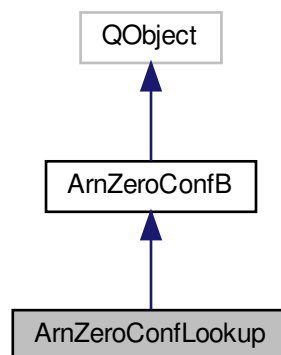
Lookup a host.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfLookup:



Collaboration diagram for ArnZeroConfLookup:



Signals

- void `lookuped` (int `id`)
Indicate successfull lookup of host.
- void `lookupError` (int `id`, int `code`)
Indicate unsuccessful lookup of host.

Public Member Functions

- [ArnZeroConfLookup](#) (QObject *parent=arnNullptr)
Standard constructor of an [ArnZeroConfLookup](#) object.
- [ArnZeroConfLookup](#) (const QString &hostName, QObject *parent=arnNullptr)
Constructor of an [ArnZeroConfLookup](#) object.
- virtual [~ArnZeroConfLookup](#) ()
Destructor of an [ArnZeroConfLookup](#) object.
- int [id](#) () const
Returns the id number for this lookup.
- void [setId](#) (int id)
Sets the id number for this this lookup.
- QString [host](#) () const
Returns the host name for this Lookup.
- void [setHost](#) (const QString &host)
Set the host name for this Lookup.
- QHostAddress [hostAddr](#) () const
Returns the host address for this Lookup.
- void [lookup](#) (bool forceMulticast=false)
Lookup the host address.
- void [releaseLookup](#) ()
Release the lookup.

Static Public Member Functions

- static bool [isForceQtDnsLookup](#) ()
Return Force using Qt for DNS lookup.
- static void [setForceQtDnsLookup](#) (bool [isForceQtDnsLookup](#))
Set Force using Qt for DNS lookup.

Friends

- class [ArnZeroConfIntern](#)

14.61.1 Detailed Description

Lookup a host.

About Zero Config

This class handles lookup of a host. It can be both Multicast and Unicast DNS lookup.

Example usage

```

ArnZeroConfLookup* ds = new ArnZeroConfLookup("myhost.local", this);
ds->setId(myId); // Optional id, later used in the signals
connect(ds, SIGNAL(lookupError(int,int)), this, SLOT(onLookupError(int,int)));
connect(ds, SIGNAL(lookuper(int)), this, SLOT(onLookuper(int)));
ds->lookup();

void XXX::onLookuper( int id)
{
    ArnZeroConfLookup* ds = qobject_cast<ArnZeroConfLookup*>( sender());
    QString hostName = ds->host();
    QHostAddress hostIp = ds->hostAddr();
    ds->releaseLookup();
    ds->deleteLater();
}

```

Definition at line 783 of file ArnZeroConf.hpp.

14.61.2 Constructor & Destructor Documentation

14.61.2.1 ArnZeroConfLookup() [1/2]

```
ArnZeroConfLookup::ArnZeroConfLookup (
    QObject * parent = arnNullptr )
```

Standard constructor of an [ArnZeroConfLookup](#) object.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 690 of file ArnZeroConf.cpp.

14.61.2.2 ArnZeroConfLookup() [2/2]

```
ArnZeroConfLookup::ArnZeroConfLookup (
    const QString & hostName,
    QObject * parent = arnNullptr )
```

Constructor of an [ArnZeroConfLookup](#) object.

All needed parameters for a lookup of a host.

Parameters

in	<i>hostName</i>	the name of the host.
in	<i>parent</i>	

Definition at line 697 of file ArnZeroConf.cpp.

14.61.2.3 ~ArnZeroConfLookup()

```
ArnZeroConfLookup::~ArnZeroConfLookup ( ) [virtual]
```

Destructor of an [ArnZeroConfLookup](#) object.

If the lookup is ongoing, it will be released.

Definition at line 706 of file ArnZeroConf.cpp.

14.61.3 Member Function Documentation

14.61.3.1 host()

```
QString ArnZeroConfLookup::host ( ) const [inline]
```

Returns the host name for this Lookup.

Returns

current host name

See also

[setHost\(\)](#)

Definition at line 824 of file ArnZeroConf.hpp.

14.61.3.2 hostAddr()

```
QHostAddress ArnZeroConfLookup::hostAddr ( ) const [inline]
```

Returns the host address for this Lookup.

Returns

current host adress

Definition at line 838 of file ArnZeroConf.hpp.

14.61.3.3 id()

```
int ArnZeroConfLookup::id ( ) const
```

Returns the id number for this lookup.

Return values

<i>the</i>	id number
------------	-----------

See also

[setId\(\)](#)

Definition at line 716 of file ArnZeroConf.cpp.

14.61.3.4 isForceQtDnsLookup()

```
bool ArnZeroConfLookup::isForceQtDnsLookup ( ) [static]
```

Return Force using Qt for DNS lookup.

Return values

<i>true</i>	if Force using Qt for DNS lookup
-------------	----------------------------------

See also

[setForceQtDnsLookup\(\)](#)

Definition at line 875 of file ArnZeroConf.cpp.

14.61.3.5 lookup()

```
void ArnZeroConfLookup::lookup (
    bool forceMulticast = false )
```

Lookup the host address.

Tries to lookup the host address necessary to establish a connection.

Result is indicated by [lookuper\(\)](#) and [lookupError\(\)](#) signals.

Parameters

in	<i>forceMulticast</i>	when true, ArnZeroConfLookup will use a mDns request to lookup the host address, even if the host name is a unicast address, i.e. outside the local network.
----	-----------------------	--

See also

[lookuper\(\)](#)
[lookupError\(\)](#)

Definition at line 728 of file ArnZeroConf.cpp.

14.61.3.6 lookedup

```
void ArnZeroConfLookup::lookuped (
    int id ) [signal]
```

Indicate successfull lookup of host.

Parameters

in	<i>id</i>	is the id number for this lookup
----	-----------	----------------------------------

See also

[lookup\(\)](#)

14.61.3.7 lookupError

```
void ArnZeroConfLookup::lookupError (
    int id,
    int code ) [signal]
```

Indicate unsuccessfull lookup of host.

Parameters

in	<i>id</i>	is the id number for this lookup
in	<i>code</i>	error code.

See also

[lookup\(\)](#)

14.61.3.8 releaseLookup()

```
void ArnZeroConfLookup::releaseLookup ( )
```

Release the lookup.

Any lookup attempts in progress will be aborted.

Definition at line 784 of file ArnZeroConf.cpp.

14.61.3.9 setForceQtDnsLookup()

```
void ArnZeroConfLookup::setForceQtDnsLookup (
    bool isForceQtDnsLookup ) [static]
```

Set Force using Qt for DNS lookup.

If mDns lookup doesn't work for a platform, try force using Qt:s built in DNS-lookup.

This is a global setting for all instances of [ArnZeroConfLookup](#).

Parameters

in	<i>isForceQtDnsLookup</i>	
----	---------------------------	--

See also

[isForceQtDnsLookup\(\)](#)

Definition at line 881 of file ArnZeroConf.cpp.

14.61.3.10 setHost()

```
void ArnZeroConfLookup::setHost (
    const QString & host ) [inline]
```

Set the host name for this Lookup.

Usually hostname contain domain, e.g. "myserver.local" but it can also be "myserver".

Parameters

in	<i>host</i>	is the current host name (se above)
----	-------------	-------------------------------------

See also

[host\(\)](#)

Definition at line 832 of file ArnZeroConf.hpp.

14.61.3.11 setId()

```
void ArnZeroConfLookup::setId (
    int id )
```

Sets the id number for this this lookup.

This id can be used to identify different lookup:s when using a common handler.

When not set, it will be automatically assigned during [lookup\(\)](#).

Parameters

in	<i>id</i>	the id number
----	-----------	---------------

See also

[id\(\)](#)

Definition at line 722 of file ArnZeroConf.cpp.

14.61.4 Friends And Related Function Documentation

14.61.4.1 ArnZeroConfIntern

```
friend class ArnZeroConfIntern [friend]
```

Definition at line 785 of file ArnZeroConf.hpp.

The documentation for this class was generated from the following files:

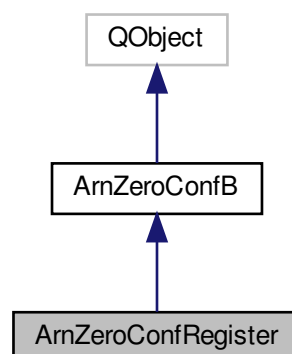
- [src/ArnInc/ArnZeroConf.hpp \(4.0.0\)](#)
- [src/ArnZeroConf.cpp \(4.0.0\)](#)

14.62 ArnZeroConfRegister Class Reference

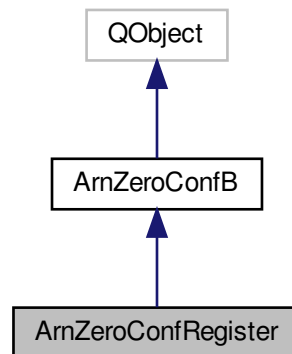
Registering a ZeroConfig service.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfRegister:



Collaboration diagram for ArnZeroConfRegister:



Signals

- void [registered](#) (const QString &[serviceName](#))
Indicate successfull registration of service.
- void [registrationError](#) (int code)
Indicate unsuccessful registration of service.

Public Member Functions

- [ArnZeroConfRegister](#) (QObject *parent=arnNullptr)
Standard constructor of an [ArnZeroConfRegister](#) object.
- [ArnZeroConfRegister](#) (const QString &[serviceName](#), QObject *parent=arnNullptr)
Constructor of an [ArnZeroConfRegister](#) object.
- [ArnZeroConfRegister](#) (const QString &[serviceName](#), const QString &[serviceType](#), quint16 [port](#), QObject *parent=arnNullptr)
Constructor of an [ArnZeroConfRegister](#) object.
- virtual [~ArnZeroConfRegister](#) ()
Destructor of an [ArnZeroConfRegister](#) object.
- QStringList [subTypes](#) () const
Returns the list of current subtypes.
- void [setSubTypes](#) (const QStringList &subtypes)
Sets the list of current subtypes.
- void [addSubType](#) (const QString &subtype)
Add a subtype to the list of current subtypes.
- quint16 [port](#) () const
Returns the port number for connecting to the service.
- void [setPort](#) (quint16 [port](#))
Sets the port number for connecting to the service.
- QString [serviceName](#) () const
Returns the service name for this Zero Config.

- `QString currentServiceName () const`
Returns the current service name for this Zero Config.
- `void setServiceName (const QString &name)`
Set the service name for this Zero Config.
- `QString host () const`
Returns the host name for this Zero Config.
- `void setHost (const QString &host)`
Set the host name for this Zero Config.
- `bool getTxtRecordMap (Arn::XStringMap &xsm)`
Load a XStringMap with parameters from the Txt Record.
- `void setTxtRecordMap (const Arn::XStringMap &xsm)`
Save a XStringMap with parameters to the Txt Record.
- `QByteArray txtRecord () const`
Return the Txt Record for this Zero Config.
- `void setTxtRecord (const QByteArray &txt)`
Set the Txt Record for this Zero Config.
- `void registerService (bool noAutoRename=false)`
Register the service.
- `void releaseService ()`
Release the service.

Friends

- class [ArnZeroConfIntern](#)

14.62.1 Detailed Description

Registering a ZeroConfig service.

About Zero Config

This class handles registration of a ZeroConfig service. The service name can be any string, giving a clear human readable naming of the service. If the given service name is already in use, it will have a number added to make it unique. A given TXT record can be registered together with the service.

Example usage

```
// In class declare
ArnZeroConfRegister* _advertService;

// In class code
_advertService = new ArnZeroConfRegister("My TestService. In the attic", this);
_advertService->addSubType("server");
Arn::XStringMap xsmPar;
xsmPar.add("ver", "1.0").add("server", "1");
_advertService->setTxtRecordMap( xsmPar);
connect(_advertService, SIGNAL(registered()), this, SLOT(onRegistered()));
connect(_advertService, SIGNAL(registrationError(int)),
        this, SLOT(onRegisterError(int)));
_advertService->registerService();
```

Definition at line 366 of file ArnZeroConf.hpp.

14.62.2 Constructor & Destructor Documentation

14.62.2.1 ArnZeroConfRegister() [1/3]

```
ArnZeroConfRegister::ArnZeroConfRegister (
    QObject * parent = arnNullptr )
```

Standard constructor of an [ArnZeroConfRegister](#) object.

The service name can be automatically generated based on the system's hostname.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 375 of file ArnZeroConf.cpp.

14.62.2.2 ArnZeroConfRegister() [2/3]

```
ArnZeroConfRegister::ArnZeroConfRegister (
    const QString & serviceName,
    QObject * parent = arnNullptr )
```

Constructor of an [ArnZeroConfRegister](#) object.

All needed parameters for an "arn" service type, using standard arn-port at this computer.

Parameters

in	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
in	<i>parent</i>	

Definition at line 382 of file ArnZeroConf.cpp.

14.62.2.3 ArnZeroConfRegister() [3/3]

```
ArnZeroConfRegister::ArnZeroConfRegister (
    const QString & serviceName,
    const QString & serviceType,
    quint16 port,
    QObject * parent = arnNullptr )
```

Constructor of an [ArnZeroConfRegister](#) object.

All needed parameters for a service at this computer.

The service type can be a name or the standard format used by the Zeroconf specification, e.g. "_arn._tcp".

Parameters

in	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
in	<i>serviceType</i>	the service type, e.g. "arn" or "_arn._tcp".
in	<i>port</i>	the service port num
in	<i>parent</i>	

Definition at line 391 of file ArnZeroConf.cpp.

14.62.2.4 ~ArnZeroConfRegister()

```
ArnZeroConfRegister::~ArnZeroConfRegister ( ) [virtual]
```

Destructor of an [ArnZeroConfRegister](#) object.

If the service is registered, it will be unregistered.

Definition at line 403 of file ArnZeroConf.cpp.

14.62.3 Member Function Documentation

14.62.3.1 addSubType()

```
void ArnZeroConfRegister::addSubType (
    const QString & subtype ) [inline]
```

Add a subtype to the list of current subtypes.

Parameters

in	<i>subtype</i>	the subtype to add, e.g. "myGroup1"
----	----------------	-------------------------------------

See also

[subTypes\(\)](#)
[setSubTypes\(\)](#)

Definition at line 427 of file ArnZeroConf.hpp.

14.62.3.2 `currentServiceName()`

```
QString ArnZeroConfRegister::currentServiceName ( ) const
```

Returns the current service name for this Zero Config.

At first, the requested service name is returned. Later the service name is internally updated with real name when [registered\(\)](#) signal is emitted.

Returns

current service name, e.g. "My House Registry (2)"

See also

[setServiceName\(\)](#)
[serviceName\(\)](#)
[registered\(\)](#)

Definition at line 414 of file ArnZeroConf.cpp.

14.62.3.3 `getTxtRecordMap()`

```
bool ArnZeroConfRegister::getTxtRecordMap (
    Arn::XStringMap & xsm ) [inline]
```

Load a XStringMap with parameters from the Txt Record.

It is assumed that the Txt Record has already been received.

After loading XStringMap is successfull it contains the parameters from the Txt Record, e.g. [Arn::XStringMap::toXString\(\)](#) can return "protovers=1.0 MyParam=xyz".

Parameters

<i>out</i>	<i>xsm</i>	is the loaded XStringMap if successfull, otherwise undefined.
------------	------------	---

Return values

<i>true</i>	if successfull.
-------------	-----------------

See also

[setTxtRecordMap\(\)](#)
[Arn::XStringMap](#)

Definition at line 509 of file ArnZeroConf.hpp.

14.62.3.4 host()

```
QString ArnZeroConfRegister::host ( ) const [inline]
```

Returns the host name for this Zero Config.

Usually hostname is empty, automatically using the computers name, but it can also be like "myserver".

Returns

current host name (se above)

See also

[setHost\(\)](#)

Definition at line 487 of file ArnZeroConf.hpp.

14.62.3.5 port()

```
quint16 ArnZeroConfRegister::port ( ) const [inline]
```

Returns the port number for connecting to the service.

Return values

<i>the</i>	port number
------------	-------------

See also

[setPort\(\)](#)

Definition at line 434 of file ArnZeroConf.hpp.

14.62.3.6 registered

```
void ArnZeroConfRegister::registered (
    const QString & serviceName ) [signal]
```

Indicate successfull registration of service.

The service name will also be internally updated, it can be accesed via [currentServiceName\(\)](#).

Parameters

<i>in</i>	<i>serviceName</i>	is the realy registered name e.g. "My House Registry (2)"
-----------	--------------------	---

See also

[registerService\(\)](#)
[setServiceName\(\)](#)
[serviceName\(\)](#)

14.62.3.7 registerService()

```
void ArnZeroConfRegister::registerService (
    bool noAutoRename = false )
```

Register the service.

Tries to register the service on the local network.

Result is indicated by [registered\(\)](#) and [registrationError\(\)](#) signals.

Parameters

in	<i>noAutoRename</i>	when true, registration will fail if another service with the same service type already is registered with the same service name.
----	---------------------	---

See also

[registered\(\)](#)
[registrationError\(\)](#)

Definition at line 427 of file ArnZeroConf.cpp.

14.62.3.8 registrationError

```
void ArnZeroConfRegister::registrationError (
    int code ) [signal]
```

Indicate unsuccessful registration of service.

Parameters

in	<i>code</i>	error code.
----	-------------	-------------

See also

[registerService\(\)](#)

14.62.3.9 releaseService()

```
void ArnZeroConfRegister::releaseService ( )
```

Release the service.

If the service is registered, it will be unregistered. Any registration attempts in progress will be aborted.

Definition at line 472 of file ArnZeroConf.cpp.

14.62.3.10 serviceName()

```
QString ArnZeroConfRegister::serviceName ( ) const [inline]
```

Returns the service name for this Zero Config.

The returned service name is always the requested name. For real name use [currentServiceName\(\)](#).

Returns

current service name, e.g. "My House Registry"

See also

[setServiceName\(\)](#)
[currentServiceName\(\)](#)
[registered\(\)](#)

Definition at line 454 of file ArnZeroConf.hpp.

14.62.3.11 setHost()

```
void ArnZeroConfRegister::setHost (
    const QString & host ) [inline]
```

Set the host name for this Zero Config.

Usually hostname is empty, automatically using the computers name, but it can also be like "myserver".

Parameters

<i>in</i>	<i>host</i>	is the current host name (se above)
-----------	-------------	-------------------------------------

See also

[host\(\)](#)

Definition at line 496 of file ArnZeroConf.hpp.

14.62.3.12 setPort()

```
void ArnZeroConfRegister::setPort (
    quint16 port ) [inline]
```

Sets the port number for connecting to the service.

When registering a service with a port number of 0, the service will not be found when browsing, but the service name will be marked as reserved.

Parameters

in	<i>port</i>	the port number
----	-------------	-----------------

See also

[port\(\)](#)

Definition at line 443 of file ArnZeroConf.hpp.

14.62.3.13 setServiceName()

```
void ArnZeroConfRegister::setServiceName (
    const QString & name )
```

Set the service name for this Zero Config.

Service names can be any human readable id. It should be easy to understand, without any cryptic coding, and can usually be modified by the end user.

The requested service name is not guaranteed to be registered, as it has to be unique within the local network. The really used name comes with the [registered\(\)](#) signal and can be accessed via [currentServiceName\(\)](#).

Parameters

in	<i>name</i>	is service name, e.g. "My House Registry"
----	-------------	---

See also

[serviceName\(\)](#)
[currentServiceName\(\)](#)
[registered\(\)](#)

Definition at line 420 of file ArnZeroConf.cpp.

14.62.3.14 `setSubTypes()`

```
void ArnZeroConfRegister::setSubTypes (
    const QStringList & subtypes ) [inline]
```

Sets the list of current subtypes.

Parameters

in	<i>subtypes</i>	The new list of subtypes, e.g. ("myGroup1", "myGroup2")
----	-----------------	---

See also

[subTypes\(\)](#)
[addSubType\(\)](#)
[ArnZeroConfBrowser::setSubType\(\)](#)

Definition at line 419 of file ArnZeroConf.hpp.

14.62.3.15 `setTxtRecord()`

```
void ArnZeroConfRegister::setTxtRecord (
    const QByteArray & txt ) [inline]
```

Set the Txt Record for this Zero Config.

The binary format should be the standardized from the Zeroconfig specification. This Txt Record will typically be used later for publishing in zero config.

Parameters

in	<i>txt</i>	is The Txt Record (in binary format)
----	------------	--------------------------------------

See also

[txtRecord\(\)](#)
[setTxtRecordMap\(\)](#)

Definition at line 540 of file ArnZeroConf.hpp.

14.62.3.16 `setTxtRecordMap()`

```
void ArnZeroConfRegister::setTxtRecordMap (
    const Arn::XStringMap & xsm ) [inline]
```

Save a XStringMap with parameters to the Txt Record.

The XStringMap contains the parameters to be saved into the Txt Record. This Txt Record will typically be used later for publishing in zero config.

Parameters

<i>in</i>	<i>xsm</i>	is the XStringMap to be saved into the Txt Record.
-----------	------------	--

See also

[getTxtRecordMap\(\)](#)
[Arn::XStringMap](#)

Definition at line 519 of file ArnZeroConf.hpp.

14.62.3.17 subTypes()

```
QStringList ArnZeroConfRegister::subTypes ( ) const [inline]
```

Returns the list of current subtypes.

Return values

<i>the</i>	subtype list, e.g. ("myGroup1", "myGroup2")
------------	---

See also

[setSubTypes\(\)](#)
[addSubType\(\)](#)

Definition at line 410 of file ArnZeroConf.hpp.

14.62.3.18 txtRecord()

```
QByteArray ArnZeroConfRegister::txtRecord ( ) const [inline]
```

Return the Txt Record for this Zero Config.

It is assumed that the Txt Record has already been received.

The binary format should be the standardized from the Zeroconfig specification.

Returns

The Txt Record (in binary format)

See also

[setTxtRecord\(\)](#)
[getTxtRecordMap\(\)](#)

Definition at line 530 of file ArnZeroConf.hpp.

14.62.4 Friends And Related Function Documentation

14.62.4.1 ArnZeroConfIntern

```
friend class ArnZeroConfIntern [friend]
```

Definition at line 368 of file ArnZeroConf.hpp.

The documentation for this class was generated from the following files:

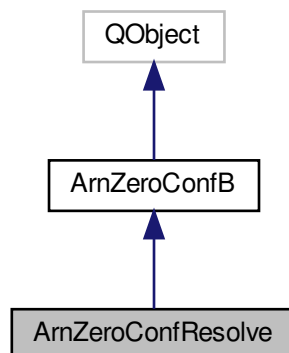
- [src/ArnInc/ArnZeroConf.hpp \(4.0.0\)](#)
- [src/ArnZeroConf.cpp \(4.0.0\)](#)

14.63 ArnZeroConfResolve Class Reference

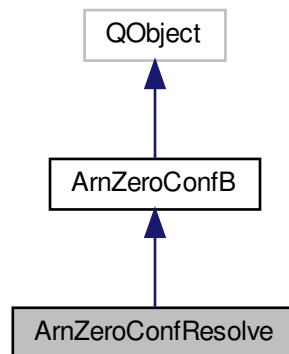
Resolv a ZeroConfig service.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfResolve:



Collaboration diagram for ArnZeroConfResolve:



Signals

- void [resolved](#) (int [id](#), const QByteArray &escFullDomain)
Indicate successfull resolve of service.
- void [resolveError](#) (int [id](#), int code)
Indicate unsuccessful resolve of service.

Public Member Functions

- [ArnZeroConfResolve](#) (QObject *parent=arnNullptr)
Standard constructor of an ArnZeroConfResolv object.
- [ArnZeroConfResolve](#) (const QString &[serviceName](#), QObject *parent=arnNullptr)
Constructor of an ArnZeroConfResolv object.
- [ArnZeroConfResolve](#) (const QString &[serviceName](#), const QString &[serviceType](#), QObject *parent=arnNullptr)
Constructor of an ArnZeroConfResolv object.
- virtual [~ArnZeroConfResolve](#) ()
Destructor of an ArnZeroConfResolv object.
- int [id](#) () const
Returns the id number for this resolv.
- void [setId](#) (int [id](#))
Sets the id number for this this resolv.
- QString [host](#) () const
Returns the host name for this resolv.
- quint16 [port](#) () const
Returns the port number for connecting to the service.
- QString [serviceName](#) () const
Returns the service name used for this resolv.
- void [setServiceName](#) (const QString &name)
Set the service name used for this resolv.

- bool [getTxtRecordMap](#) ([Arn::XStringMap](#) &xsm)
Load a XStringMap with parameters from the Txt Record.
- [QByteArray txtRecord](#) () const
Return the Txt Record for this Zero Config.
- void [resolve](#) (bool forceMulticast=false)
Resolve the service.
- void [releaseResolve](#) ()
Release the resolving.

Friends

- class [ArnZeroConfIntern](#)

14.63.1 Detailed Description

Resolv a ZeroConfig service.

About Zero Config

This class handles resolving of a ZeroConfig service. The service name can be given directly if known, but typically it comes from [ArnZeroConfBrowser](#).

Example usage

```
// In class code
ArnZeroConfResolve* ds = new ArnZeroConfResolve("My TestService.
    In the attic", this);
ds->setId( myId); // Optional id, later used in the signals
connect( ds, SIGNAL(resolveError(int,int)), this, SLOT(onResolveError(int,int)));
connect( ds, SIGNAL(resolved(int,QByteArray)), this, SLOT(onResolved(int,QByteArray)));
ds->resolve();

void XXX::onResolved( int id, QByteArray escFullDomain)
{
    ArnZeroConfResolve* ds = qobject_cast<ArnZeroConfResolve*>( sender
        ());
    Arn::XStringMap xsmPar;
    ds->getTxtRecordMap( xsmPar);
    QString info = QString()
        + " Domain=" + ds->domain()
        + " Host=" + ds->host()
        + " Port=" + QString::number( ds->port())
        + " Txt: " + QString::fromUtf8( xsmPar.toXString().constData());
    QString ver = xsmPar.valueString("MyVers");
    ds->releaseService();
    ds->deleteLater();
}
```

Definition at line 616 of file ArnZeroConf.hpp.

14.63.2 Constructor & Destructor Documentation

14.63.2.1 ArnZeroConfResolve() [1/3]

```
ArnZeroConfResolve::ArnZeroConfResolve (
    QObject * parent = arnNullptr )
```

Standard constructor of an ArnZeroConfResolv object.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 528 of file ArnZeroConf.cpp.

14.63.2.2 ArnZeroConfResolve() [2/3]

```
ArnZeroConfResolve::ArnZeroConfResolve (
    const QString & serviceName,
    QObject * parent = arnNullptr )
```

Constructor of an ArnZeroConfResolv object.

All needed parameters for an "arn" service type.

Parameters

in	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
in	<i>parent</i>	

Definition at line 535 of file ArnZeroConf.cpp.

14.63.2.3 ArnZeroConfResolve() [3/3]

```
ArnZeroConfResolve::ArnZeroConfResolve (
    const QString & serviceName,
    const QString & serviceType,
    QObject * parent = arnNullptr )
```

Constructor of an ArnZeroConfResolv object.

All needed parameters for a service.

The service type can be a name or the standard format used by the Zeroconf specification, e.g. "_arn._tcp".

Parameters

in	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
in	<i>serviceType</i>	the service type, e.g. "arn" or "_arn._tcp".
in	<i>parent</i>	

Definition at line 544 of file ArnZeroConf.cpp.

14.63.2.4 ~ArnZeroConfResolve()

```
ArnZeroConfResolve::~~ArnZeroConfResolve ( ) [virtual]
```

Destructor of an ArnZeroConfResolv object.

If the service is registered, it will be unregistered.

Definition at line 555 of file ArnZeroConf.cpp.

14.63.3 Member Function Documentation

14.63.3.1 getTxtRecordMap()

```
bool ArnZeroConfResolve::getTxtRecordMap (
    Arn::XStringMap & xsm ) [inline]
```

Load a XStringMap with parameters from the Txt Record.

It is assumed that the Txt Record has already been received.

After loading XStringMap is successfull it contains the parameters from the Txt Record, e.g. [Arn::XStringMap::toXString\(\)](#) can return "protovers=1.0 MyParam=xyz".

Parameters

out	xsm	is the loaded XStringMap if successfull, otherwise undefined.
-----	-----	---

Return values

true	if successfull.
------	-----------------

See also

[Arn::XStringMap](#)

Definition at line 703 of file ArnZeroConf.hpp.

14.63.3.2 host()

```
QString ArnZeroConfResolve::host ( ) const [inline]
```

Returns the host name for this resolv.

Hostname contain domain, e.g. "myserver.local".

Returns

current host name (se above)

Definition at line 670 of file ArnZeroConf.hpp.

14.63.3.3 id()

```
int ArnZeroConfResolve::id ( ) const
```

Returns the id number for this resolv.

Returns

the id number

See also

[setId\(\)](#)

Definition at line 565 of file ArnZeroConf.cpp.

14.63.3.4 port()

```
quint16 ArnZeroConfResolve::port ( ) const [inline]
```

Returns the port number for connecting to the service.

Return values

<i>the</i>	port number
------------	-------------

Definition at line 676 of file ArnZeroConf.hpp.

14.63.3.5 releaseResolve()

```
void ArnZeroConfResolve::releaseResolve ( )
```

Release the resolving.

Any resolve attempts in progress will be aborted.

Definition at line 615 of file ArnZeroConf.cpp.

14.63.3.6 resolve()

```
void ArnZeroConfResolve::resolve (
    bool forceMulticast = false )
```

Resolve the service.

Tries to resolve the service to determine the host and port necessary to establish a connection.

Result is indicated by [resolved\(\)](#) and [resolveError\(\)](#) signals.

Parameters

in	<i>forceMulticast</i>	when true, ArnZeroConfResolv will use a multicast request to resolve the service, even if the host name is a unicast address, i.e. outside the local network.
----	-----------------------	---

See also

[resolved\(\)](#)
[resolveError\(\)](#)

Definition at line 577 of file ArnZeroConf.cpp.

14.63.3.7 resolved

```
void ArnZeroConfResolve::resolved (
    int id,
    const QByteArray & escFullDomain ) [signal]
```

Indicate successfull resolve of service.

Parameters

in	<i>id</i>	is the id number for this resolve
in	<i>escFullDomain</i>	is the raw full domain with esc sequences

See also

[resolve\(\)](#)

14.63.3.8 resolveError

```
void ArnZeroConfResolve::resolveError (
    int id,
    int code ) [signal]
```

Indicate unsuccessful resolve of service.

Parameters

in	<i>id</i>	is the id number for this resolve
in	<i>code</i>	is the error code.

See also[resolve\(\)](#)**14.63.3.9 serviceName()**

```
QString ArnZeroConfResolve::serviceName ( ) const [inline]
```

Returns the service name used for this resolv.

Returns

current service name, e.g. "My House Registry"

Definition at line 682 of file ArnZeroConf.hpp.

14.63.3.10 setId()

```
void ArnZeroConfResolve::setId (
    int id )
```

Sets the id number for this this resolv.

This id can be used to identify different resolves when using a common handler.

When not set, it will be automatically assigned during [resolve\(\)](#).

Parameters

in	<i>id</i>	the id number
----	-----------	---------------

See also[id\(\)](#)

Definition at line 571 of file ArnZeroConf.cpp.

14.63.3.11 setServiceName()

```
void ArnZeroConfResolve::setServiceName (
    const QString & name ) [inline]
```

Set the service name used for this resolv.

Service names can be any human readable id. It will be used when resolving the service.

Parameters

in	<i>name</i>	is service name, e.g. "My House Registry"
----	-------------	---

See also

[serviceName\(\)](#)

Definition at line 691 of file ArnZeroConf.hpp.

14.63.3.12 txtRecord()

```
QByteArray ArnZeroConfResolve::txtRecord ( ) const [inline]
```

Return the Txt Record for this Zero Config.

It is assumed that the Txt Record has already been received.

The binary format should be the standardized from the Zeroconfig specification.

Returns

The Txt Record (in binary format)

See also

[getTxtRecordMap\(\)](#)

Definition at line 713 of file ArnZeroConf.hpp.

14.63.4 Friends And Related Function Documentation

14.63.4.1 ArnZeroConfIntern

```
friend class ArnZeroConfIntern [friend]
```

Definition at line 618 of file ArnZeroConf.hpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnZeroConf.hpp \(4.0.0\)](#)
- [src/ArnZeroConf.cpp \(4.0.0\)](#)

14.64 Arn::ClientSyncMode Struct Reference

The Client session Sync mode at connect & reconnect.

```
#include <Arn.hpp>
```

Public Types

- enum [E](#) { [Invalid](#), [StdAutoMaster](#), [ImplicitMaster](#), [ExplicitMaster](#) }

14.64.1 Detailed Description

The Client session Sync mode at connect & reconnect.

Definition at line 155 of file Arn.hpp.

14.64.2 Member Enumeration Documentation

14.64.2.1 E

```
enum Arn::ClientSyncMode::E
```

Enumerator

Invalid	Value for Server, can not be set in Client.
StdAutoMaster	Default dynamic auto master mode, general purpose, prohibit Null value sync.
ImplicitMaster	First local write gives permanent Master mode, typically a client value reporter.
ExplicitMaster	Explicit permanent Master mode, typically an observer or manually setup Master mode.

Definition at line 156 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.65 Arn::Coding Struct Reference

```
#include <Arn.hpp>
```

Public Types

- enum [E](#) { [Binary](#) = 0x0000, [Text](#) = 0x1000 }

14.65.1 Detailed Description

Definition at line 196 of file Arn.hpp.

14.65.2 Member Enumeration Documentation

14.65.2.1 E

```
enum Arn::Coding::E
```

Enumerator

Binary	No special coding, can be anything.
Text	Text coding, can be any character set.

Definition at line 197 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.66 Arn::DataType Class Reference

Data type of an [Arn Data Object](#)

```
#include <Arn.hpp>
```

Public Types

- enum [E](#) {
[Null](#) = 0, [Int](#) = 1, [Double](#) = 2, [Real](#) = 2,
[ByteArray](#) = 3, [String](#) = 4, [Variant](#) = 5 }

14.66.1 Detailed Description

Data type of an *Arn Data Object*

Definition at line 74 of file Arn.hpp.

14.66.2 Member Enumeration Documentation

14.66.2.1 E

enum `Arn::DataType::E`

Enumerator

Null	
Int	
Double	
Real	
ByteArray	
String	
Variant	

Definition at line 78 of file Arn.hpp.

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

- `src/ArnInc/Arn.hpp (4.0.0)`

14.67 Arn::EnumTxt Class Reference

Class Enum text.

```
#include <MQFlags.hpp>
```

Classes

- struct `IncludeMode`

Public Member Functions

- [EnumTxt](#) (const QMetaObject *metaObj, bool isFlag, const [_InitEnumTxt](#) *initTxt, const [_InitSubEnum](#) *initSubEnum, const char *name)
- [EnumTxt](#) (bool isFlag=false, const QString &name=QString())
Create a dynamic runtime handled EnumTxt.
- [~EnumTxt](#) ()
- void [setTxtRef](#) (const char *txt, int enumVal, quint16 nameSpace)
- void [setTxt](#) (const char *txt, int enumVal, quint16 nameSpace)
Set an additional text for an enum val in a namespace.
- const char * [getTxt](#) (int enumVal, quint16 nameSpace=0, bool *isFound=arnNullptr) const
Returns the text for a enum value in a namespace.
- void [setTxtString](#) (const QString &txt, int enumVal, quint16 nameSpace)
Set an additional text for an enum val in a namespace.
- QString [getTxtString](#) (int enumVal, quint16 nameSpace=0, bool *isFound=arnNullptr) const
Returns the text for a enum value in a namespace.
- int [getEnumVal](#) (const char *txt, int defaultVal=0, quint16 nameSpace=0, bool *isFound=arnNullptr) const
Returns the enum value for a text in a namespace.
- int [getEnumVal](#) (const QString &txt, int defaultVal=0, quint16 nameSpace=0, bool *isFound=arnNullptr) const
Returns the enum value for a text in a namespace.
- bool [getSubEnumVal](#) (const char *txt, int &subEnumVal, uint &bitMask, quint16 nameSpace=0) const
Returns the shifted enum value and the mask for a subEnum text in a namespace.
- bool [getSubEnumVal](#) (const QString &txt, int &subEnumVal, uint &bitMask, quint16 nameSpace=0) const
Returns the shifted enum value and the mask for a subEnum text in a namespace.
- void [addFlagsTo](#) ([Arn::XStringMap](#) &xsm, const [IncludeMode](#) &incMode, quint16 nameSpace=0, bool neverHumanize=false) const
Adds enum flags to a XStringMap.
- void [addSubEnumTo](#) ([Arn::XStringMap](#) &xsm, quint16 nameSpace=0, bool neverHumanize=false) const
Adds sub enum flags to a XStringMap.
- void [addBitSetTo](#) ([Arn::XStringMap](#) &xsm, quint16 nameSpace=0, bool neverHumanize=false) const
Adds bit set for enum flags to a XStringMap.
- void [addBitSet](#) ([Arn::XStringMap](#) &xsm, quint16 nameSpace=0, bool neverHumanize=false) const
- QString [getBitSet](#) (quint16 nameSpace=0, bool neverHumanize=false) const
returns the bit set string for enum flags
- void [addSubEnumPlainTo](#) ([Arn::XStringMap](#) &xsm, quint16 nameSpace=0, bool neverHumanize=false) const
Adds all sub enum plain and shifted to a XStringMap.
- QString [flagsToString](#) (int val, quint16 nameSpace=0) const
returns text string for enum flags
- QStringList [flagsToStringList](#) (int val, quint16 nameSpace=0) const
returns string list for enum flags
- int [flagsFromString](#) (const QString &flagString, quint16 nameSpace=0)
returns enum flags from string
- int [flagsFromStringList](#) (const QStringList &flagStrings, quint16 nameSpace=0)
returns enum flags from string list
- void [addEnumSetTo](#) ([Arn::XStringMap](#) &xsm, quint16 nameSpace=0, bool neverHumanize=false) const
Adds enum set to a XStringMap.
- void [addEnumSet](#) ([Arn::XStringMap](#) &xsm, quint16 nameSpace=0, bool neverHumanize=false) const
- QString [getEnumSet](#) (quint16 nameSpace=0, bool neverHumanize=false) const
returns the enum set string
- QStringList [getBasicTextList](#) (quint16 nameSpace=0, bool neverHumanize=false) const
returns a string list containing the most basic texts
- void [addSubEnum](#) (const [EnumTxt](#) &subEnum, uint bitMask, uint factor)

- Adds an other [EnumTxt](#) as a subEnum to this flag [EnumTxt](#).*
- `const char * name () const`
returns the name of the enum (class)
- `int enumCount () const`
returns number of enumerators in the enum (class)
- `void setMissingTxt (quint16 toNameSpace, quint16 fromNameSpace=0, bool neverHumanize=false)`
Copies missing enum texts from one namespace to another.
- `bool isFlag () const`
Returns true if this is a flag usage.
- `void clear ()`
Clear this dynamic instance to its starting state.
- `bool loadEnumSet (const Arn::XStringMap &xsm, const QString &name=QString())`
Loads the instance by an enum set [XStringMap](#).
- `bool loadEnumSet (const QString &xstr, const QString &name=QString())`
Loads the instance by an enum set [XString](#).
- `bool loadBitSet (const Arn::XStringMap &xsm, const QString &name=QString())`
Loads the instance by an bit set (flags) [XStringMap](#).
- `bool loadBitSet (const QString &xstr, const QString &name=QString())`
Loads the instance by an bit set (flags) [XString](#).
- `int subEnumCount () const`
Returns number of subEnums in this bitSet (class)
- `QString subEnumNameAt (int idx, quint16 nameSpace=0) const`
Returns the name of a SubEnum.
- `bool subEnumPropAt (int idx, uint &bitMask, uchar &bitPos) const`
Returns the properties of a SubEnum.
- `const EnumTxt * subEnumAt (int idx) const`
Returns a pointer to a SubEnum.

Static Public Member Functions

- `static QString humanize (const QString &txt)`
returns the humanized text
- `static QByteArray numToStr (uint num)`
- `static uint strToNum (const QByteArray &str, bool *isOk=arnNullptr)`
- `static uchar strToBitpos (const QByteArray &str, bool *isOk=arnNullptr)`

14.67.1 Detailed Description

Class Enum text.

Example usage

```
class AllowLevelT {
    Q_GADGET
    Q_ENUMS (E)
public:
    enum E {
        Low = 0,
        Mid,
        High
    };
    MQ_DECLARE_ENUMTXT ( AllowLevelT)
};
```



```

class AllowClassT {
    Q_GADGET
    Q_ENUMS(E)
public:
    enum E {
        None      = 0x00,
        Read      = 0x01,
        AllowLevB0 = 0x02,
        AllowLevB1 = 0x04,
        Create     = 0x08,
        Delete     = 0x10,
        AllowLev   = AllowLevB0 | AllowLevB1,
        All       = 0xff
    };
    MQ_DECLARE_FLAGSTXT( AllowClassT)

    MQ_DECLARE_SUBETXT(
        MQ_SUBETXT_ADD_RELDEF( AllowLevelT, AllowLev, AllowLevB0)
    )
    MQ_SUBETXT_ADD_RELOP( AllowLevelT, AllowLev, AllowLevB0)

    enum NS {NsEnum, NsHuman};
    MQ_DECLARE_ENUM_NSTXT(
        { NsHuman, Read, "Allow Read" },
        { NsHuman, Delete, "Allow Delete" }
    )
};

MQ_DECLARE_OPERATORS_FOR_FLAGS( AllowClassT)

class ConnectStatT {
    Q_GADGET
    Q_ENUMS(E)
public:
    enum E {
        Init = 0,
        Connected,
        Error,
        Disconnected,
        TriedAll
    };
    MQ_DECLARE_ENUMTXT( ConnectStatT)

    enum NS {NsEnum, NsHuman};
    MQ_DECLARE_ENUM_NSTXT(
        { NsHuman, Init, "Initialized" },
        { NsHuman, Error, "Connect error" },
        { NsHuman, MQ_NSTXT_FILL_MISSING_FROM( NsEnum) }
    )
};

```

Definition at line 212 of file MQFlags.hpp.

14.67.2 Constructor & Destructor Documentation

14.67.2.1 EnumTxt() [1/2]

```

Arn::EnumTxt::EnumTxt (
    const QMetaObject * metaObj,
    bool isFlag,
    const _InitEnumTxt * initTxt,
    const _InitSubEnum * initSubEnum,
    const char * name )

```

Definition at line 63 of file MQFlags.cpp.

14.67.2.2 EnumTxt() [2/2]

```
Arn::EnumTxt::EnumTxt (
    bool isFlag = false,
    const QString & name = QString() ) [explicit]
```

Create a dynamic runtime handled [EnumTxt](#).

This is used for handling general Enums that is not statically assigned via QObject. **Example usage**

```
Arn::EnumTxt myFlags( true, "MyFlags");
myFlags.loadBitSet( "B0=Flag1 B5=Flag2 0=None 0x21=FlagAll");
```

Parameters

in	<i>isFlag</i>	is true when using Flags (bitSet), otherwise use plain Enums.
in	<i>name</i>	is the name of these Enums / Flags.

Definition at line 77 of file MQFlags.cpp.

14.67.2.3 ~EnumTxt()

```
Arn::EnumTxt::~EnumTxt ( )
```

Definition at line 87 of file MQFlags.cpp.

14.67.3 Member Function Documentation

14.67.3.1 addBitSet()

```
void Arn::EnumTxt::addBitSet (
    Arn::XStringMap & xsm,
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const [inline]
```

Definition at line 399 of file MQFlags.hpp.

14.67.3.2 addBitSetTo()

```
void Arn::EnumTxt::addBitSetTo (
    Arn::XStringMap & xsm,
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const
```

Adds bit set for enum flags to a [XStringMap](#).

Example

```
Arn::XStringMap xsm;
xsm.add("T", "Test");
AllowClassT::txt().addBitSetTo( xsm, 0, true);
```

will give xsm containing: T=Test B0=Read B3=Create B4=Delete 0=None 0xff=All SE6:B1=AllowLev E0=Low E1=Mid E2=High

Parameters

out	<i>xsm</i>	is the XStringMap to be added to.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

See also

[addFlagsTo\(\)](#)
[addSubEnumTo\(\)](#)
[humanize\(\)](#)

Definition at line 262 of file MQFlags.cpp.

14.67.3.3 addEnumSet()

```
void Arn::EnumTxt::addEnumSet (
    Arn::XStringMap & xsm,
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const [inline]
```

Definition at line 503 of file MQFlags.hpp.

14.67.3.4 addEnumSetTo()

```
void Arn::EnumTxt::addEnumSetTo (
    Arn::XStringMap & xsm,
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const
```

Adds enum set to a [XStringMap](#).

Example

```
Arn::XStringMap xsm;
xsm.add("T", "Test");
ConnectStatT::txt().addEnumSetTo( xsm);
```

will give xsm containing: T=Test 0=Init 1=Connected 2=Error 3=Disconnected 4=Tried all

Parameters

out	<i>xsm</i>	is the XStringMap to be added to.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

See also

[getEnumSet\(\)](#)
[humanize\(\)](#)

Definition at line 406 of file MQFlags.cpp.

14.67.3.5 addFlagsTo()

```
void Arn::EnumTxt::addFlagsTo (
    Arn::XStringMap & xsm,
    const IncludeMode & incMode,
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const
```

Adds enum flags to a [XStringMap](#).

Example

```
Arn::XStringMap xsm;
xsm.add("T", "Test");
AllowClassT::txt().addFlagsTo( xsm, IncludeMode::OnlySingle1Bits, 0, true);
```

will give xsm containing: T=Test B0=Read B3=Create B4=Delete

Parameters

out	<i>xsm</i>	is the XStringMap to be added to.
in	<i>incMode</i>	specifies what to include (SingleBits / MultiBits / SubEnumBits / "Any").
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

See also

[addBitSetTo\(\)](#)
[humanize\(\)](#)

Definition at line 193 of file MQFlags.cpp.

14.67.3.6 addSubEnum()

```
void Arn::EnumTxt::addSubEnum (
    const EnumTxt & subEnum,
    uint bitMask,
    uint factor )
```

Adds an other [EnumTxt](#) as a subEnum to this flag [EnumTxt](#).

Example

```
// In class usage example, following is done behind the scene
AllowClassT::txt().addSubEnum( AllowLevelT::txt(), AllowClassT::AllowLev, AllowClassT::AllowLevB0);
```

Parameters

in	<i>subEnum</i>	is the other EnumTxt to be included as a subEnum.
in	<i>bitMask</i>	is used for selecting the subEnum among the flags.
in	<i>factor</i>	is multiplying the base enum before it is masked in as subEnum to flags.

Returns

the flags enum value.

Definition at line 451 of file MQFlags.cpp.

14.67.3.7 addSubEnumPlainTo()

```
void Arn::EnumTxt::addSubEnumPlainTo (
    Arn::XStringMap & xsm,
```

```

    quint16 nameSpace = 0,
    bool neverHumanize = false ) const

```

Adds all sub enum plain and shifted to a [XStringMap](#).

Also adds bitmask and name of the sub enum All enums must have unique names **Example**

```

Arn::XStringMap xsm;
xsm.add("T", "Test");
AllowClassT::txt().addSubEnumPlainTo( xsm, 0, true);

```

will give xsm containing: T=Test 6=AllowLev 0=Low 2=Mid 4=High

Parameters

out	<i>xsm</i>	is the XStringMap to be added to.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

See also

[addBitSetTo\(\)](#)
[humanize\(\)](#)

Definition at line 281 of file MQFlags.cpp.

14.67.3.8 addSubEnumTo()

```

void Arn::EnumTxt::addSubEnumTo (
    Arn::XStringMap & xsm,
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const

```

Adds sub enum flags to a [XStringMap](#).

Example

```

Arn::XStringMap xsm;
xsm.add("T", "Test");
AllowClassT::txt().addSubEnumTo( xsm, 0, true);

```

will give xsm containing: T=Test SE6:B1=AllowLev E0=Low E1=Mid E2=High

Parameters

out	<i>xsm</i>	is the XStringMap to be added to.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

See also

[addBitSetTo\(\)](#)
[humanize\(\)](#)

Definition at line 236 of file MQFlags.cpp.

14.67.3.9 clear()

```
void Arn::EnumTxt::clear ( )
```

Clear this dynamic instance to its starting state.

Clear will do nothing if this instance was created statically with a QMetaObject.

Definition at line 574 of file MQFlags.cpp.

14.67.3.10 enumCount()

```
int Arn::EnumTxt::enumCount ( ) const
```

returns number of enumerators in the enum (class)

Example

```
qDebug() << ConnectStatT::txt().enumCount();
```

will print: 5

Returns

the count of enumerators.

Definition at line 736 of file MQFlags.cpp.

14.67.3.11 flagsFromString()

```
int Arn::EnumTxt::flagsFromString (
    const QString & flagString,
    quint16 nameSpace = 0 )
```

returns enum flags from string

Example

```
QString flagString = "Create | Delete | Low";
int val = AllowClassT::txt().flagsFromString( flagString);
```

will give val: 0x18 (0x08 + 0x10 + 0x00)

Parameters

in	<i>flagString</i>	is the flags text.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

Returns

the flags enum value.

See also

[flagsFromStringList\(\)](#)

Definition at line 354 of file MQFlags.cpp.

14.67.3.12 flagsFromStringList()

```
int Arn::EnumTxt::flagsFromStringList (
    const QStringList & flagStrings,
    quint16 nameSpace = 0 )
```

returns enum flags from string list

Example

```
QStringList flagStrings;
flagStrings << "Create" << "Delete" << "High";
int val = AllowClassT::txt().flagsFromString( flagStrings);
```

will give val: 0x1c (0x08 + 0x10 + 0x04)

Parameters

in	<i>flagStrings</i>	is the flags text list.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

Returns

the flags enum value.

See also

[flagsFromString\(\)](#)

Definition at line 362 of file MQFlags.cpp.

14.67.3.13 flagsToString()

```
QString Arn::EnumTxt::flagsToString (
    int val,
    quint16 nameSpace = 0 ) const
```

returns text string for enum flags

Example

```
AllowClassT allow;
allow = allow.Create | allow.Delete;
allow.setSubEnum( AllowLevelT::Mid);
QDebug() << allow.txt().flagsToString( allow);
```

will print: "Create | Delete | Mid"

Parameters

in	<i>val</i>	is the flags enum value.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

Returns

the flags text string.

See also

[flagsToStringList\(\)](#)

Definition at line 305 of file MQFlags.cpp.

14.67.3.14 flagsToStringList()

```
QStringList Arn::EnumTxt::flagsToStringList (
    int val,
    quint16 nameSpace = 0 ) const
```

returns string list for enum flags

Example

```
AllowClassT allow;
allow = allow.Create | allow.Delete;
allow.setSubEnum( AllowLevelT::Low);
QStringList allowList = allow.txt().flagsToStringList( allow);
```

will give allowList containing: "Create", "Delete", "Low"

Parameters

in	<i>val</i>	is the flags enum value.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

Returns

the flags string list.

See also

[flagsToString\(\)](#)

Definition at line 313 of file MQFlags.cpp.

14.67.3.15 getBasicTextList()

```
QStringList Arn::EnumTxt::getBasicTextList (
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const
```

returns a string list containing the most basic texts

For a EnumSet this is the complete text list. For a BitSet this is the texts for all the single 1-bits except those used for SubEnums. Example

```
qDebug() << AllowClassT::txt().getBasicTextList( 0, true);
```

will print: "Read" "Create" "Delete"

Parameters

in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

Returns

the basic string list.

See also

[humanize\(\)](#)

Definition at line 437 of file MQFlags.cpp.

14.67.3.16 `getBitSet()`

```
QString Arn::EnumTxt::getBitSet (
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const
```

returns the bit set string for enum flags

Example

```
qDebug() << AllowClassT::txt().getBitSet();
```

will print: "B0=Read B3=Create B4=Delete 0=None 0xff=All SE6:B1=AllowLev E0=Low E1=Mid E2=High"

Parameters

in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

Returns

the bit set string.

See also

[humanize\(\)](#)

Definition at line 271 of file MQFlags.cpp.

14.67.3.17 `getEnumSet()`

```
QString Arn::EnumTxt::getEnumSet (
    quint16 nameSpace = 0,
    bool neverHumanize = false ) const
```

returns the enum set string

Example

```
qDebug() << ConnectStatT::txt().getEnumSet();
```

will print: "0=Init 1=Connected 2=Error 3=Disconnected 4=Tried_all"

Parameters

in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

Returns

the enum set string.

See also

[humanize\(\)](#)

Definition at line 429 of file MQFlags.cpp.

14.67.3.18 getEnumVal() [1/2]

```
int Arn::EnumTxt::getEnumVal (
    const char * txt,
    int defaultVal = 0,
    quint16 nameSpace = 0,
    bool * isFound = arnNullptr ) const
```

Returns the enum value for a text in a namespace.

Parameters

in	<i>txt</i>	is the enum text.
in	<i>defaultVal</i>	is the returned value when txt is not found.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
out	<i>isFound</i>	returns status when pointer is none null.

Returns

the enum value.

See also

[setTxt\(\);](#)

Definition at line 142 of file MQFlags.cpp.

14.67.3.19 getEnumVal() [2/2]

```
int Arn::EnumTxt::getEnumVal (
    const QString & txt,
    int defaultVal = 0,
    quint16 nameSpace = 0,
    bool * isFound = arnNullptr ) const
```

Returns the enum value for a text in a namespace.

Parameters

in	<i>txt</i>	is the enum text.
in	<i>defaultVal</i>	is the returned value when txt is not found.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
out	<i>isFound</i>	returns status when pointer is none null.

Returns

the enum value.

See also

[setTxt\(\);](#)
[setTxtString\(\);](#)

Definition at line 163 of file MQFlags.cpp.

14.67.3.20 `getSubEnumVal()` [1/2]

```
bool Arn::EnumTxt::getSubEnumVal (
    const char * txt,
    int & subEnumVal,
    uint & bitMask,
    quint16 nameSpace = 0 ) const
```

Returns the shifted enum value and the mask for a subEnum text in a namespace.

The enum value returned is shifted (with factor) to directly fit the flags enum. **Example usage**

```
int subEnumVal;
uint bitMask;
AllowClassT::txt().getSubEnumVal( "Mid", subEnumVal, bitMask);
qDebug() << subEnumVal << bitMask
```

will print: 2 and 6

Parameters

in	<i>txt</i>	is the subEnum text.
out	<i>subEnumVal</i>	is the returned shifted value when txt is found as a subEnum.
out	<i>bitMask</i>	is the returned value when txt is found as a subEnum.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

Return values

<i>is</i>	true when txt is found as a subEnum.
-----------	--------------------------------------

See also

[setTxt\(\);](#)
[addSubEnum\(\);](#)

Definition at line 169 of file MQFlags.cpp.

14.67.3.21 getSubEnumVal() [2/2]

```
bool Arn::EnumTxt::getSubEnumVal (
    const QString & txt,
    int & subEnumVal,
    uint & bitMask,
    quint16 nameSpace = 0 ) const
```

Returns the shifted enum value and the mask for a subEnum text in a namespace.

The enum value returned is shifted (with factor) to directly fit the flags enum. **Example usage**

```
int subEnumVal;
uint bitMask;
AllowClassT::txt().getSubEnumVal( "High", subEnumVal, bitMask);
qDebug() << subEnumVal << bitMask
```

will print: 4 and 6

Parameters

in	<i>txt</i>	is the subEnum text.
out	<i>subEnumVal</i>	is the returned shifted value when txt is found as a subEnum.
out	<i>bitMask</i>	is the returned value when txt is found as a subEnum.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

Return values

<i>is</i>	true when txt is found as a subEnum.
-----------	--------------------------------------

See also

[setTxtString\(\);](#)
[addSubEnum\(\);](#)

Definition at line 187 of file MQFlags.cpp.

14.67.3.22 `getTxt()`

```
const char * Arn::EnumTxt::getTxt (
    int enumVal,
    quint16 nameSpace = 0,
    bool * isFound = arnNullptr ) const
```

Returns the text for a enum value in a namespace.

Parameters

in	<i>enumVal</i>	is the referenced value.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
out	<i>isFound</i>	returns status when pointer is none null.

Returns

the enum text.

See also

[setTxt\(\);](#)

Definition at line 121 of file MQFlags.cpp.

14.67.3.23 `getTxtString()`

```
QString Arn::EnumTxt::getTxtString (
    int enumVal,
    quint16 nameSpace = 0,
    bool * isFound = arnNullptr ) const
```

Returns the text for a enum value in a namespace.

Parameters

in	<i>enumVal</i>	is the referenced value.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.
out	<i>isFound</i>	returns status when pointer is none null.

Returns

the enum text.

See also

[setTxt\(\);](#)
[setTxtString\(\);](#)

Definition at line 136 of file MQFlags.cpp.

14.67.3.24 `humanize()`

```
QString Arn::EnumTxt::humanize (
    const QString & txt ) [static]
```

returns the humanized text

The input text can be Chamel-case or '_' word separated. First output char will always be upper case and the following chars will always be lower case.

Example output

```
"MySimpelCase" ==> "My simpel case"
"My_Simpel_case" ==> "My simpel case"
"count123ms" ==> "Count 123 ms"
"DDTIsBad" ==> "DDT is bad"
```

Parameters

<i>in</i>	<i>txt</i>	is the text to be humanized.
-----------	------------	------------------------------

Returns

the humanized text.

Definition at line 497 of file MQFlags.cpp.

14.67.3.25 `isFlag()`

```
bool Arn::EnumTxt::isFlag ( ) const
```

Returns true if this is a flag usage.

Return values

<i>is</i>	true when this is a flag, false when plain enum.
-----------	--

Definition at line 568 of file MQFlags.cpp.

14.67.3.26 loadBitSet() [1/2]

```
bool Arn::EnumTxt::loadBitSet (
    const Arn::XStringMap & xsm,
    const QString & name = QString() )
```

Loads the instance by an bit set (flags) [XStringMap](#).

Example output

```
Arn::XStringMap xsm( "B0=Read B3=Create 0=None SE6:B1=AllowLev E0=Low E1=Mid E2=High");
Arn::EnumTxt myFlags;
myFlags.loadBitSet( xsm, "MyFlags");
```

Parameters

in	<i>xsm</i>	is the XStringMap containing the flags representation.
in	<i>name</i>	is the name of this flag collection.

Return values

<i>returns</i>	true if successful.
----------------	---------------------

Definition at line 625 of file MQFlags.cpp.

14.67.3.27 loadBitSet() [2/2]

```
bool Arn::EnumTxt::loadBitSet (
    const QString & xstr,
    const QString & name = QString() )
```

Loads the instance by an bit set (flags) [XString](#).

Example output

```
QString xstr( "B0=Read B3=Create 0=None SE6:B1=AllowLev E0=Low E1=Mid E2=High");
Arn::EnumTxt myFlags;
myFlags.loadBitSet( xstr, "MyFlags");
```

Parameters

in	<i>xstr</i>	is the XString containing the flags representation.
in	<i>name</i>	is the name of this flag collection.

Return values

Return values

<i>returns</i>	true if successful.
----------------	---------------------

Definition at line 689 of file MQFlags.cpp.

14.67.3.28 loadEnumSet() [1/2]

```
bool Arn::EnumTxt::loadEnumSet (
    const Arn::XStringMap & xsm,
    const QString & name = QString() )
```

Loads the instance by an enum set [XStringMap](#).

Example output

```
Arn::XStringMap xsm( "0=Arn 1=Is 0x2=Great");
Arn::EnumTxt myEnum;
myEnum.loadEnumSet( xsm, "MyEnum");
```

Parameters

in	<i>xsm</i>	is the XStringMap containing the enum representation.
in	<i>name</i>	is the name of this enum collection.

Return values

<i>returns</i>	true if successful.
----------------	---------------------

Definition at line 599 of file MQFlags.cpp.

14.67.3.29 loadEnumSet() [2/2]

```
bool Arn::EnumTxt::loadEnumSet (
    const QString & xstr,
    const QString & name = QString() )
```

Loads the instance by an enum set [XString](#).

Example output

```
QString xstr( "0=Arn 0x1=Is 2=Great");
Arn::EnumTxt myEnum;
myEnum.loadEnumSet( xstr, "MyEnum");
```

Parameters

in	<i>xstr</i>	is the XString containing the enum representation.
in	<i>name</i>	is the name of this enum collection.

Return values

<i>returns</i>	true if successful.
----------------	---------------------

Definition at line 619 of file MQFlags.cpp.

14.67.3.30 name()

```
const char * Arn::EnumTxt::name ( ) const
```

returns the name of the enum (class)

Example

```
qDebug() << ConnectStatT::txt().name();
```

will print: "ConnectStatT"

Returns

the enum (class) name.

Definition at line 730 of file MQFlags.cpp.

14.67.3.31 numToStr()

```
QByteArray Arn::EnumTxt::numToStr (
    uint num ) [static]
```

Definition at line 745 of file MQFlags.cpp.

14.67.3.32 setMissingTxt()

```
void Arn::EnumTxt::setMissingTxt (
    quint16 toNameSpace,
    quint16 fromNameSpace = 0,
    bool neverHumanize = false )
```

Copies missing enum texts from one namespace to another.

The standard 0 namespace contains all enum texts as defined and can not be altered. All the other wanted namespaces can have customized enum texts, but then there can be enum values without a text in such namespace. This function can be used to fill in those missing texts from another namespace, which typically is 0 as it contains all texts.

Parameters

in	<i>toNameSpace</i>	is the altered one. Can not be 0.
in	<i>fromNameSpace</i>	is the one to copy from.
in	<i>neverHumanize</i>	if true never applies the enum text humanize algorithm.

See also

[humanize\(\)](#)

Definition at line 469 of file MQFlags.cpp.

14.67.3.33 setTxt()

```
void Arn::EnumTxt::setTxt (
    const char * txt,
    int enumVal,
    quint16 nameSpace )
```

Set an additional text for an enum val in a namespace.

The namespace with index 0 is the standard namespace that automatically gets its texts from the definition of the enum.

Example usage

```
AllowClassT allow;
allow.txt().setTxt("Test - Create", allow.Create, AllowClassT::NsHuman);
allow = allow.Create;
QDebug() << allow.toString() << allow.toString( AllowClassT::NsHuman)
```

will print: "Create" and "Test - Create"

Parameters

in	<i>txt</i>	is the new enum text.
in	<i>enumVal</i>	is the referenced value.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

See also

[getTxt\(\)](#);

Definition at line 107 of file MQFlags.cpp.

14.67.3.34 setTxtRef()

```
void Arn::EnumTxt::setTxtRef (
    const char * txt,
    int enumVal,
    quint16 nameSpace )
```

Definition at line 100 of file MQFlags.cpp.

14.67.3.35 setTxtString()

```
void Arn::EnumTxt::setTxtString (
    const QString & txt,
    int enumVal,
    quint16 nameSpace )
```

Set an additional text for an enum val in a namespace.

Parameters

in	<i>txt</i>	is the new enum text.
in	<i>enumVal</i>	is the referenced value.
in	<i>nameSpace</i>	is the usage set for this enum, e.g human readable.

See also

[setTxt\(\)](#);
[getTxtString\(\)](#);

Definition at line 130 of file MQFlags.cpp.

14.67.3.36 strToBitpos()

```
uchar Arn::EnumTxt::strToBitpos (
    const QByteArray & str,
    bool * isOk = arnNullptr ) [static]
```

Definition at line 765 of file MQFlags.cpp.

14.67.3.37 strToNum()

```
uint Arn::EnumTxt::strToNum (
    const QByteArray & str,
    bool * isOk = arnNullptr ) [static]
```

Definition at line 754 of file MQFlags.cpp.

14.67.3.38 subEnumAt()

```
const EnumTxt * Arn::EnumTxt::subEnumAt (
    int idx ) const
```

Returns a pointer to a SubEnum.

Example output

```
QString xstr( "B0=Read B3=Create 0=None SE6:B1=AllowLev E0=Low E1=Mid E2=High");
Arn::EnumTxt myFlags;
myFlags.loadBitSet( xstr, "MyFlags");
const Arn::EnumTxt* sube = myFlags.subEnumAt( 0);
QDebug() << sube->getEnumSet( 0, false);
```

will print: "0=Low 1=Mid 2=High"

Parameters

in	<i>idx</i>	is the index of the SubEnum.
----	------------	------------------------------

Return values

<i>returns</i>	a pointer to SubEnum for inbound idx, otherwise arnNullPtr.
----------------	---

Definition at line 722 of file MQFlags.cpp.

14.67.3.39 subEnumCount()

```
int Arn::EnumTxt::subEnumCount ( ) const
```

Returns number of subEnums in this bitSet (class)

Example

```
QDebug() << AllowClassT::txt().subEnumCount();
```

will print: 1

Returns

the count of sub enums.

Definition at line 695 of file MQFlags.cpp.

14.67.3.40 subEnumNameAt()

```
QString Arn::EnumTxt::subEnumNameAt (
    int idx,
    quint16 nameSpace = 0 ) const
```

Returns the name of a SubEnum.

Name is taken by the registered bitmask of the SubEnum. When using static creation of this instance, there must be a declared enum for the bitmask of the SubEnum. E.g. in AllowClassT this enum is AllowLev. When using dynamic creation of flags, loading a BitSet, the name of the SubEnum is set in the process. This is also true when using [addSubEnum\(\)](#). Example

```
qDebug() << AllowClassT::txt().subEnumNameAt( 0);
```

will print: "AllowLev"

Parameters

in	<i>idx</i>	is the index of the SubEnum.
in	<i>nameSpace</i>	is the nameSpace for the registered name (by bitMask).

Return values

<i>returns</i>	the SubEnum name for inbound idx, otherwise QString().
----------------	--

Definition at line 702 of file MQFlags.cpp.

14.67.3.41 subEnumPropAt()

```
bool Arn::EnumTxt::subEnumPropAt (
    int idx,
    uint & bitMask,
    uchar & bitPos ) const
```

Returns the properties of a SubEnum.

Example

```
uint bitMask; uchar bitPos; qDebug() << AllowClassT::txt().subEnumNameAt( 0, bitMask, bitPos) <<
bitMask << bitPos;
```

will print: true 6 1

Parameters

in	<i>idx</i>	is the index of the SubEnum.
out	<i>bitMask</i>	is the bitmask for the SubEnum.
out	<i>bitPos</i>	is the position for the starting bit of the SubEnum.

Return values

<i>returns</i>	true for inbound idx, otherwise false.
----------------	--

Definition at line 711 of file MQFlags.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/MQFlags.hpp \(4.0.0\)](#)
- [src/MQFlags.cpp \(4.0.0\)](#)

14.68 ArnZeroConf::Error Struct Reference

Errors of ZeroConfig, other values are defined in dns_sd.h.

```
#include <ArnZeroConf.hpp>
```

Public Types

- enum `E` {
`Ok = 0`, `Running = -1`, `BadReqSeq = -2`, `Timeout = -3`,
`UDnsFail = -4` }

14.68.1 Detailed Description

Errors of ZeroConfig, other values are defined in dns_sd.h.

Definition at line 53 of file ArnZeroConf.hpp.

14.68.2 Member Enumeration Documentation

14.68.2.1 E

```
enum ArnZeroConf::Error::E
```

Enumerator

<code>Ok</code>	Ok, defined as <code>kDNSServiceErr_NoError</code> in <code>dns_sd.h</code> .
<code>Running</code>	Operation in progress.
<code>BadReqSeq</code>	Bad request sequence.
<code>Timeout</code>	Operation timeout.
<code>UDnsFail</code>	Unicast DNS lookup fail.

Definition at line 54 of file ArnZeroConf.hpp.

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

- [src/ArnInc/ArnZeroConf.hpp \(4.0.0\)](#)

14.69 Arn::ExportCode Class Reference

Code used in blob for arnExport() and arnImport()

```
#include <Arn.hpp>
```

Public Types

- enum `E` {
 [ByteArray](#) = 3, [String](#) = 4, [Variant](#) = 5, [VariantTxt](#) = 16,
 [VariantBin](#) = 17 }

14.69.1 Detailed Description

Code used in blob for arnExport() and arnImport()

Definition at line 92 of file Arn.hpp.

14.69.2 Member Enumeration Documentation

14.69.2.1 E

```
enum Arn::ExportCode::E
```

Enumerator

ByteArray	
String	
Variant	
VariantTxt	
VariantBin	

Definition at line 96 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.70 ArnCoreItem::Heritage Struct Reference

```
#include <ArnCoreItem.hpp>
```

Public Types

- enum [E](#) { [BasicItem](#) = 0x01, [ItemB](#) = 0x02, [AdaptItem](#) = 0x04, [None](#) = 0x00 }
The heritage track of this item.

14.70.1 Detailed Description

Definition at line 62 of file ArnCoreItem.hpp.

14.70.2 Member Enumeration Documentation

14.70.2.1 E

```
enum ArnCoreItem::Heritage::E
```

The heritage track of this item.

Enumerator

BasicItem	
ItemB	
AdaptItem	
None	

Definition at line 64 of file ArnCoreItem.hpp.

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

- [src/ArnInc/ArnCoreItem.hpp \(4.0.0\)](#)

14.71 ArnClient::HostAddrPort Struct Reference

```
#include <ArnClient.hpp>
```

Public Member Functions

- [HostAddrPort](#) ()

Public Attributes

- QString [addr](#)
- quint16 [port](#)

14.71.1 Detailed Description

Definition at line 113 of file ArnClient.hpp.

14.71.2 Constructor & Destructor Documentation

14.71.2.1 HostAddrPort()

```
ArnClient::HostAddrPort::HostAddrPort ( ) [inline]
```

Definition at line 117 of file ArnClient.hpp.

14.71.3 Member Data Documentation

14.71.3.1 addr

```
QString ArnClient::HostAddrPort::addr
```

Definition at line 114 of file ArnClient.hpp.

14.71.3.2 port

```
quint16 ArnClient::HostAddrPort::port
```

Definition at line 115 of file ArnClient.hpp.

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

- [src/ArnInc/ArnClient.hpp \(4.0.0\)](#)

14.72 Arn::EnumTxt::IncludeMode Struct Reference

```
#include <MQFlags.hpp>
```

Public Types

- enum [E](#) {
[OnlySingle1Bits](#), [OnlyMulti1Bits](#), [OnlySubEnumBits](#), [AnyButSubEnumBits](#),
[Any](#) }

14.72.1 Detailed Description

Definition at line 215 of file MQFlags.hpp.

14.72.2 Member Enumeration Documentation

14.72.2.1 E

```
enum Arn::EnumTxt::IncludeMode::E
```

Enumerator

OnlySingle1Bits	
OnlyMulti1Bits	
OnlySubEnumBits	
AnyButSubEnumBits	
Any	

Definition at line 216 of file MQFlags.hpp.

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

- [src/ArnInc/MQFlags.hpp \(4.0.0\)](#)

14.73 Arn::InfoType Struct Reference

Info type for exchange static (meta) info between [ArnClient](#) and [ArnServer](#).

```
#include <Arn.hpp>
```

Public Types

- enum [E](#) { [Custom](#) = 0, [N](#) }

14.73.1 Detailed Description

Info type for exchange static (meta) info between [ArnClient](#) and [ArnServer](#).

Definition at line 107 of file Arn.hpp.

14.73.2 Member Enumeration Documentation

14.73.2.1 E

```
enum Arn::InfoType::E
```

Enumerator

Custom	
N	

Definition at line 108 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.74 ArnRpc::Invoke Struct Reference

```
#include <ArnRpc.hpp>
```

Public Types

- enum E { NoQueue = 0x01 }

14.74.1 Detailed Description

Definition at line 164 of file ArnRpc.hpp.

14.74.2 Member Enumeration Documentation

14.74.2.1 E

```
enum ArnRpc::Invoke::E
```

Enumerator

NoQueue	This invoke is not queued, multiple calls to same method might overwrite.
---------	---

Definition at line 165 of file ArnRpc.hpp.

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

- [src/ArnInc/ArnRpc.hpp \(4.0.0\)](#)

14.75 Arn::LinkFlags Struct Reference

Link flags when accessing an *Arn Data Object*

```
#include <Arn.hpp>
```

Public Types

- enum `E` {
 `Folder` = 0x01, `CreateAllowed` = 0x02, `SilentError` = 0x04, `LastLink` = 0x08,
 `Threaded` = 0x80 }

14.75.1 Detailed Description

Link flags when accessing an *Arn Data Object*

Definition at line 170 of file Arn.hpp.

14.75.2 Member Enumeration Documentation

14.75.2.1 E

```
enum Arn::LinkFlags::E
```

Enumerator

Folder	
CreateAllowed	
SilentError	
LastLink	
Threaded	

Definition at line 171 of file Arn.hpp.

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

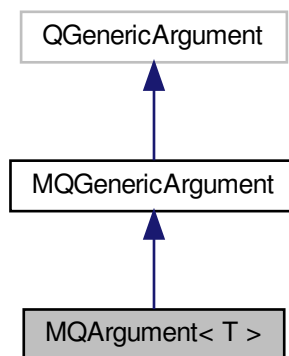
- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.76 MQArgument< T > Class Template Reference

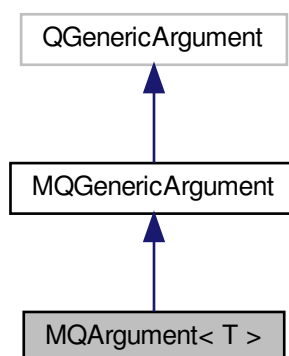
Similar to QArgument but with added argument label (parameter name)

```
#include <Arnrpc.hpp>
```

Inheritance diagram for MQArgument< T >:



Collaboration diagram for MQArgument< T >:



Public Member Functions

- [MQArgument](#) (const char *aName, const char *aLabel, const T &aData)

14.76.1 Detailed Description

```
template<class T>
class MQArgument< T >
```

Similar to QArgument but with added argument label (parameter name)

Definition at line 76 of file ArnRpc.hpp.

14.76.2 Constructor & Destructor Documentation

14.76.2.1 MQArgument()

```
template<class T >
MQArgument< T >::MQArgument (
    const char * aName,
    const char * aLabel,
    const T & aData ) [inline]
```

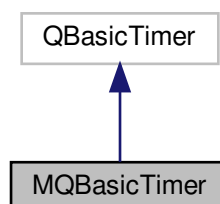
Definition at line 79 of file ArnRpc.hpp.

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

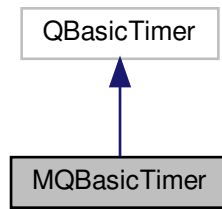
- [src/ArnInc/ArnRpc.hpp \(4.0.0\)](#)

14.77 MQBasicTimer Class Reference

Inheritance diagram for MQBasicTimer:



Collaboration diagram for MQBasicTimer:



Public Member Functions

- [MQBasicTimer](#) ()
- int [interval](#) () const
- void [setInterval](#) (int [interval](#))
- void [start](#) (QObject *obj)
- void [start](#) (int msec, QObject *obj)

14.77.1 Detailed Description

Definition at line 61 of file ArnItem.cpp.

14.77.2 Constructor & Destructor Documentation

14.77.2.1 MQBasicTimer()

```
MQBasicTimer::MQBasicTimer ( ) [inline]
```

Definition at line 64 of file ArnItem.cpp.

14.77.3 Member Function Documentation

14.77.3.1 interval()

```
int MQBasicTimer::interval ( ) const [inline]
```

Definition at line 69 of file ArnItem.cpp.

14.77.3.2 setInterval()

```
void MQBasicTimer::setInterval (
    int interval ) [inline]
```

Definition at line 70 of file ArnItem.cpp.

14.77.3.3 start() [1/2]

```
void MQBasicTimer::start (
    QObject * obj ) [inline]
```

Definition at line 71 of file ArnItem.cpp.

14.77.3.4 start() [2/2]

```
void MQBasicTimer::start (
    int msec,
    QObject * obj ) [inline]
```

Definition at line 72 of file ArnItem.cpp.

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

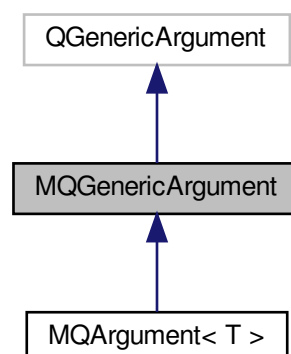
- [src/ArnItem.cpp \(4.0.0\)](#)

14.78 MQGenericArgument Class Reference

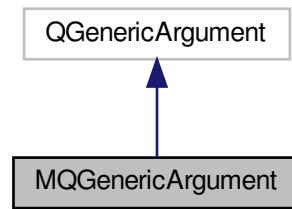
Similar to QGenericArgument but with added argument label (parameter name)

```
#include <ArnRpc.hpp>
```

Inheritance diagram for MQGenericArgument:



Collaboration diagram for MQGenericArgument:



Public Member Functions

- [MQGenericArgument](#) (const char *aName=arnNullptr, const char *aLabel=arnNullptr, const void *a↔Data=arnNullptr)
- [MQGenericArgument](#) (const QGenericArgument &qgenArg)
- const char * [label](#) () const

14.78.1 Detailed Description

Similar to QGenericArgument but with added argument label (parameter name)

Definition at line 59 of file ArnRpc.hpp.

14.78.2 Constructor & Destructor Documentation

14.78.2.1 MQGenericArgument() [1/2]

```
MQGenericArgument::MQGenericArgument (
    const char * aName = arnNullptr,
    const char * aLabel = arnNullptr,
    const void * aData = arnNullptr ) [inline]
```

Definition at line 62 of file ArnRpc.hpp.

14.78.2.2 MQGenericArgument() [2/2]

```
MQGenericArgument::MQGenericArgument (
    const QGenericArgument & qgenArg ) [inline]
```

Definition at line 65 of file ArnRpc.hpp.

14.78.3 Member Function Documentation

14.78.3.1 label()

```
const char* MQGenericArgument::label ( ) const [inline]
```

Definition at line 67 of file ArnRpc.hpp.

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

- [src/ArnInc/ArnRpc.hpp \(4.0.0\)](#)

14.79 Arn::NameF Struct Reference

```
#include <Arn.hpp>
```

Public Types

- enum [E](#) { [Default](#) = 0x00, [NoFolderMark](#) = 0x01, [EmptyOk](#) = 0x02, [Relative](#) = 0x04 }
Selects a format for path or item name.

14.79.1 Detailed Description

Definition at line 181 of file Arn.hpp.

14.79.2 Member Enumeration Documentation

14.79.2.1 E

```
enum Arn::NameF::E
```

Selects a format for path or item name.

Enumerator

Default	Empty not ok, Path: Absolute Item: FolderMark.
NoFolderMark	Only on discrete names, no effect on path. "test/" ==> "test".
EmptyOk	Path: "@/test" ==> "//test", Item: "@" ==> "".
Relative	Only on path, no effect on discrete names. "/test/value" ==> "test/value".

Definition at line 183 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.80 Arn::ObjectMode Class Reference

```
#include <Arn.hpp>
```

Public Types

- enum `E` { `Normal` = 0x00, `BiDir` = 0x01, `Pipe` = 0x02, `Save` = 0x04 }

14.80.1 Detailed Description

General global mode of an *Arn Data Object* Max 8 bit

Definition at line 118 of file Arn.hpp.

14.80.2 Member Enumeration Documentation

14.80.2.1 E

```
enum Arn::ObjectMode::E
```

Enumerator

Normal	default
BiDir	A two way object, typically for validation or pipe.
Pipe	Implies <i>BiDir</i> and all data is preserved as a stream.
Save	Data is persistent and will be saved.

Definition at line 122 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.81 Arn::ObjectSyncMode Class Reference

```
#include <Arn.hpp>
```

Public Types

- enum `E` { `Normal` = 0x00, `Monitor` = 0x01, `Master` = 0x02, `AutoDestroy` = 0x04 }

14.81.1 Detailed Description

The client session sync mode of an *Arn Data Object* Max 8 bit

Definition at line 137 of file Arn.hpp.

14.81.2 Member Enumeration Documentation

14.81.2.1 E

```
enum Arn::ObjectSyncMode::E
```

Enumerator

Normal	Default.
Monitor	Monitor of server object for client.
Master	The client is default generator of data.
AutoDestroy	Destroy this <i>Arn Data Object</i> when client (tcp/ip) closes.

Definition at line 141 of file Arn.hpp.

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

- src/ArnInc/Arn.hpp (4.0.0)

14.82 ArnRpc::MethodsParam::Params Struct Reference

```
#include <ArnRpc.hpp>
```

Public Attributes

- QList< QByteArray > `paramNames`
- QList< QList< int > > `methodIdsTab`
- QList< int > `allMethodIds`

14.82.1 Detailed Description

Definition at line 469 of file ArnRpc.hpp.

14.82.2 Member Data Documentation

14.82.2.1 allMethodIds

```
QList<int> ArnRpc::MethodsParam::Params::allMethodIds
```

Definition at line 472 of file ArnRpc.hpp.

14.82.2.2 methodIdsTab

```
QList< QList<int> > ArnRpc::MethodsParam::Params::methodIdsTab
```

Definition at line 471 of file ArnRpc.hpp.

14.82.2.3 paramNames

```
QList<QByteArray> ArnRpc::MethodsParam::Params::paramNames
```

Definition at line 470 of file ArnRpc.hpp.

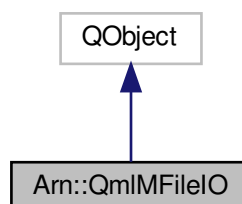
The documentation for this struct was generated from the following file:

- [src/ArnInc/ArnRpc.hpp \(4.0.0\)](#)

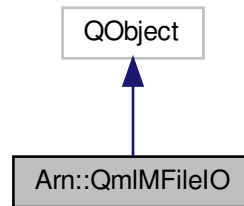
14.83 Arn::QmIMFileIO Class Reference

```
#include <ArnQmlMSystem.hpp>
```

Inheritance diagram for Arn::QmIMFileIO:



Collaboration diagram for Arn::QmIMFileIO:



Public Slots

- void [setPath](#) (const QString &path)

Signals

- void [pathChanged](#) (const QString &path)
- void [error](#) (const QString &msg)

Public Member Functions

- [QmIMFileIO](#) (QObject *parent=arnNullptr)
- Q_INVOKABLE QString [read](#) ()
- Q_INVOKABLE bool [write](#) (const QString &data)
- Q_INVOKABLE QByteArray [readBytes](#) ()
- Q_INVOKABLE bool [writeBytes](#) (const QByteArray &data)
- QString [path](#) ()

Properties

- QString [path](#)

14.83.1 Detailed Description

Definition at line 41 of file ArnQmIMSystem.hpp.

14.83.2 Constructor & Destructor Documentation

14.83.2.1 QmlFileIO()

```
Arn::QmlFileIO::QmlFileIO (
    QObject * parent = arnNullptr ) [explicit]
```

Definition at line 41 of file ArnQmlSystem.cpp.

14.83.3 Member Function Documentation

14.83.3.1 error

```
void Arn::QmlFileIO::error (
    const QString & msg ) [signal]
```

14.83.3.2 path()

```
QString Arn::QmlFileIO::path ( )
```

14.83.3.3 pathChanged

```
void Arn::QmlFileIO::pathChanged (
    const QString & path ) [signal]
```

14.83.3.4 read()

```
QString Arn::QmlFileIO::read ( )
```

Definition at line 47 of file ArnQmlSystem.cpp.

14.83.3.5 readBytes()

```
QByteArray Arn::QmlFileIO::readBytes ( )
```

Definition at line 95 of file ArnQmlSystem.cpp.

14.83.3.6 setPath

```
void Arn::QmlMFileIO::setPath (
    const QString & path ) [slot]
```

Definition at line 141 of file ArnQmlMSystem.cpp.

14.83.3.7 write()

```
bool Arn::QmlMFileIO::write (
    const QString & data )
```

Definition at line 77 of file ArnQmlMSystem.cpp.

14.83.3.8 writeBytes()

```
bool Arn::QmlMFileIO::writeBytes (
    const QByteArray & data )
```

Definition at line 118 of file ArnQmlMSystem.cpp.

14.83.4 Property Documentation

14.83.4.1 path

```
QString Arn::QmlMFileIO::path [read], [write]
```

Definition at line 46 of file ArnQmlMSystem.hpp.

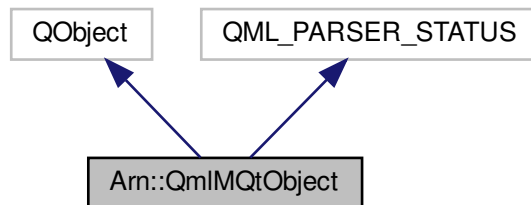
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQmlMSystem.hpp \(4.0.0\)](#)
- [src/ArnQmlMSystem.cpp \(4.0.0\)](#)

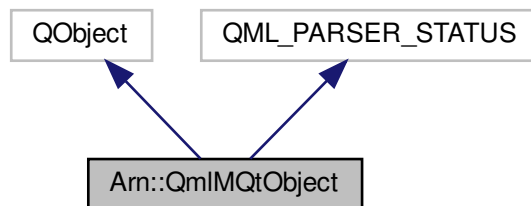
14.84 Arn::QmlMQtObject Class Reference

```
#include <ArnQmlMQt.hpp>
```

Inheritance diagram for Arn::QmlMQtObject:



Collaboration diagram for Arn::QmlMQtObject:



Signals

- void [parentChanged](#) (QmlMQtObject *obj)
- void [completed](#) ()

Public Member Functions

- [QmlMQtObject](#) (QmlMQtObject *parent=arnNullptr)
- virtual [~QmlMQtObject](#) ()
- [QmlMQtObject * parentItem](#) () const
- void [setParentItem](#) (QmlMQtObject *parent)
- [QML_LIST_PROPERTY](#)< QObject > [data](#) ()
- virtual void [classBegin](#) ()
- virtual void [componentComplete](#) ()

Static Public Member Functions

- static void [data_append](#) ([QML_LIST_PROPERTY](#)< [QObject](#) > *prop, [QObject](#) *obj)
- static [ARN_SIZETYPE](#) [data_count](#) ([QML_LIST_PROPERTY](#)< [QObject](#) > *prop)
- static [QObject](#) * [data_at](#) ([QML_LIST_PROPERTY](#)< [QObject](#) > *prop, [ARN_SIZETYPE](#) index)
- static void [data_clear](#) ([QML_LIST_PROPERTY](#)< [QObject](#) > *prop)

Properties

- [QDeclarativeListProperty](#)< [QObject](#) > [data](#)
- [QmlMQtObject](#) parent

14.84.1 Detailed Description

Definition at line 52 of file [ArnQmlMQt.hpp](#).

14.84.2 Constructor & Destructor Documentation

14.84.2.1 [QmlMQtObject](#)()

```
Arn::QmlMQtObject::QmlMQtObject (
    QmlMQtObject * parent = arnNullptr )
```

Definition at line 47 of file [ArnQmlMQt.cpp](#).

14.84.2.2 [~QmlMQtObject](#)()

```
Arn::QmlMQtObject::~QmlMQtObject ( ) [virtual]
```

Definition at line 53 of file [ArnQmlMQt.cpp](#).

14.84.3 Member Function Documentation

14.84.3.1 [classBegin](#)()

```
void Arn::QmlMQtObject::classBegin ( ) [virtual]
```

Definition at line 113 of file [ArnQmlMQt.cpp](#).

14.84.3.2 completed

```
void Arn::QmlMQtObject::completed ( ) [signal]
```

14.84.3.3 componentComplete()

```
void Arn::QmlMQtObject::componentComplete ( ) [virtual]
```

Definition at line 118 of file ArnQmlMQt.cpp.

14.84.3.4 data()

```
QML_LIST_PROPERTY<QObject> Arn::QmlMQtObject::data ( )
```

14.84.3.5 data_append()

```
void Arn::QmlMQtObject::data_append (
    QML_LIST_PROPERTY< QObject > * prop,
    QObject * obj ) [static]
```

Definition at line 77 of file ArnQmlMQt.cpp.

14.84.3.6 data_at()

```
QObject * Arn::QmlMQtObject::data_at (
    QML_LIST_PROPERTY< QObject > * prop,
    ARN_SIZE_TYPE index ) [static]
```

Definition at line 93 of file ArnQmlMQt.cpp.

14.84.3.7 data_clear()

```
void Arn::QmlMQtObject::data_clear (
    QML_LIST_PROPERTY< QObject > * prop ) [static]
```

Definition at line 103 of file ArnQmlMQt.cpp.

14.84.3.8 data_count()

```
ARN_SIZE_TYPE Arn::QmlMQtObject::data_count (
    QML_LIST_PROPERTY< QObject > * prop ) [static]
```

Definition at line 86 of file ArnQmlMQt.cpp.

14.84.3.9 parentChanged

```
void Arn::QmlMQtObject::parentChanged (
    QmlMQtObject * obj ) [signal]
```

14.84.3.10 parentItem()

```
QmlMQtObject * Arn::QmlMQtObject::parentItem ( ) const
```

Definition at line 58 of file ArnQmlMQt.cpp.

14.84.3.11 setParentItem()

```
void Arn::QmlMQtObject::setParentItem (
    QmlMQtObject * parent )
```

Definition at line 64 of file ArnQmlMQt.cpp.

14.84.4 Property Documentation

14.84.4.1 data

```
QML_LIST_PROPERTY< QObject > Arn::QmlMQtObject::data [read]
```

Definition at line 57 of file ArnQmlMQt.hpp.

14.84.4.2 parent

`QmlMQtObject` Arn::QmlMQtObject::parent [read], [write]

Definition at line 63 of file ArnQmlMQt.hpp.

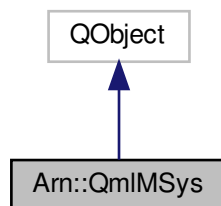
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQmlMQt.hpp \(4.0.0\)](#)
- [src/ArnQmlMQt.cpp \(4.0.0\)](#)

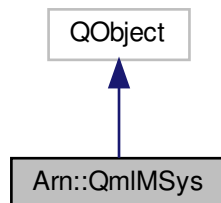
14.85 Arn::QmlMSys Class Reference

```
#include <ArnQml.hpp>
```

Inheritance diagram for Arn::QmlMSys:



Collaboration diagram for Arn::QmlMSys:



Public Slots

- `QVariantMap` [xstringToEnum](#) (const QString &xstring)

Properties

- int [quickTypeRun](#)

14.85.1 Detailed Description

Definition at line 734 of file ArnQml.hpp.

14.85.2 Member Function Documentation

14.85.2.1 xstringToEnum

```
QVariantMap Arn::QmlMSys::xstringToEnum (  
    const QString & xstring ) [slot]
```

Definition at line 476 of file ArnQml.cpp.

14.85.3 Property Documentation

14.85.3.1 quickTypeRun

```
int Arn::QmlMSys::quickTypeRun [read]
```

Definition at line 739 of file ArnQml.hpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQml.hpp \(4.0.0\)](#)
- [src/ArnQml.cpp \(4.0.0\)](#)

14.86 Arn::SameValue Struct Reference

Action when assigning same value to an [ArnItem](#).

```
#include <Arn.hpp>
```

Public Types

- enum [E](#) { [Accept](#) = 0, [Ignore](#) = 1, [DefaultAction](#) = -1 }

14.86.1 Detailed Description

Action when assigning same value to an [ArnItem](#).

Definition at line 61 of file Arn.hpp.

14.86.2 Member Enumeration Documentation

14.86.2.1 E

```
enum Arn::SameValue::E
```

Enumerator

Accept	Assigning same value generates an update of the <i>Arn Data Object</i>
Ignore	Assigning same value is ignored.
DefaultAction	Assigning same value gives default action set in ArnM or ArnItem .

Definition at line 62 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(4.0.0\)](#)

14.87 ArnDiscoverInfo::State Struct Reference

[State](#) of [Arn](#) discover browse data. Can be tested by relative order.

```
#include <ArnDiscover.hpp>
```

Public Types

- enum [E](#) {
[Init](#), [ServiceName](#), [HostInfoErr](#), [HostInfo](#),
[HostIpErr](#), [HostIp](#) }

14.87.1 Detailed Description

[State](#) of [Arn](#) discover browse data. Can be tested by relative order.

Definition at line 79 of file ArnDiscover.hpp.

14.87.2 Member Enumeration Documentation

14.87.2.1 E

```
enum ArnDiscoverInfo::State::E
```

Enumerator

Init	Initialized null state.
ServiceName	Got service name and domain (from browsing)
HostInfoErr	Got error during resolving HostName, HostPort, type and properties.
HostInfo	Also got HostName, HostPort, type and properties (from resolving)
HostIpErr	Got error during DNS lookup HostIp.
HostIp	Also got HostIp (from DNS lookup)

Definition at line 80 of file ArnDiscover.hpp.

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

- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)

14.88 ArnZeroConf::State Struct Reference

States of ZeroConfig, limited valid for each [ArnZeroConfB](#) subclass / These values must be synced with: [ArnDiscover::State](#).

```
#include <ArnZeroConf.hpp>
```

Public Types

- enum [E](#) {
[None](#) = 0x0000, [Registering](#) = 0x0100, [Registered](#) = 0x0001, [Register](#) = 0x0101,
[Browsing](#) = 0x0200, [Resolving](#) = 0x0400, [Resolved](#) = 0x0004, [Resolve](#) = 0x0404,
[LookingUp](#) = 0x0800, [Lookuped](#) = 0x0008, [Lookup](#) = 0x0808, [InProgress](#) = 0x0f00 }

14.88.1 Detailed Description

States of ZeroConfig, limited valid for each [ArnZeroConfB](#) subclass / These values must be synced with: [ArnDiscover::State](#).

Definition at line 71 of file ArnZeroConf.hpp.

14.88.2 Member Enumeration Documentation

14.88.2.1 E

```
enum ArnZeroConf::State::E
```

Enumerator

None	Inactive state.
Registering	Registering service in progress.
Registered	Registering service has finished successfully.
Register	isAny(): Registering service in progress or has finished successfully
Browsing	Browsing for service in progress.
Resolving	Resolving service in progress.
Resolved	Resolving service has finished successfully.
Resolve	isAny(): Resolving service in progress or has finished successfully
LookingUp	Lookup host in progress.
Lookuped	Lookup host has finished successfully.
Lookup	isAny(): Lookup host in progress or has finished successfully
InProgress	isAny(): Operation in progress

Definition at line 72 of file ArnZeroConf.hpp.

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

- [src/ArnInc/ArnZeroConf.hpp \(4.0.0\)](#)

14.89 ArnDiscoverAdvertise::State Struct Reference

States of DiscoverAdvertise / These values must be synced with: [ArnZeroConf::State](#).

```
#include <ArnDiscover.hpp>
```

Public Types

- enum [E](#) { [None](#) = 0x0000, [StartupAdvertise](#) = 0x0100, [Advertising](#) = 0x0001, [Advertise](#) = 0x0101 }

14.89.1 Detailed Description

States of DiscoverAdvertise / These values must be synced with: [ArnZeroConf::State](#).

Definition at line 637 of file ArnDiscover.hpp.

14.89.2 Member Enumeration Documentation

14.89.2.1 E

```
enum ArnDiscoverAdvertise::State::E
```

Enumerator

None	Inactive state.
StartupAdvertise	Startup advertising in progress.
Advertising	Is now advertising. Startup has finished successfully.
Advertise	isAny(): Startup advertising in progress or has finished successfully.

Definition at line 638 of file ArnDiscover.hpp.

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

- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)

14.90 ArnError::StdCode Struct Reference

```
#include <ArnError.hpp>
```

Public Types

- enum `E` {
`Ok = 0`, `Info = 1`, `Warning = 2`, `Err_Undef = 15`,
`Err_Custom = 16` }

14.90.1 Detailed Description

Definition at line 72 of file ArnError.hpp.

14.90.2 Member Enumeration Documentation**14.90.2.1 E**

```
enum ArnError::StdCode::E
```

Enumerator

Ok	
Info	
Warning	
Err_Undef	
Err_Custom	

Definition at line 74 of file ArnError.hpp.

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

- [src/ArnInc/ArnError.hpp \(4.0.0\)](#)

14.91 ArnItemValve::SwitchMode Struct Reference

```
#include <ArnItemValve.hpp>
```

Public Types

- enum [E](#) { [InStream](#) = 0x01, [OutStream](#) = 0x02, [InOutStream](#) = [InStream](#) | [OutStream](#) }

14.91.1 Detailed Description

Definition at line 83 of file [ArnItemValve.hpp](#).

14.91.2 Member Enumeration Documentation

14.91.2.1 E

```
enum ArnItemValve::SwitchMode::E
```

Enumerator

InStream	Control target item notifying (signal) updated value.
OutStream	Control target item accepting assign of value (setValue)
InOutStream	Convenience, combined <i>InStream</i> and <i>OutStream</i>

Definition at line 84 of file [ArnItemValve.hpp](#).

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

- [src/ArnInc/ArnItemValve.hpp \(4.0.0\)](#)

14.92 ArnServer::Type Struct Reference

```
#include <ArnServer.hpp>
```

Public Types

- enum [E](#) { [NetSync](#) }

14.92.1 Detailed Description

Definition at line 104 of file ArnServer.hpp.

14.92.2 Member Enumeration Documentation

14.92.2.1 E

```
enum ArnServer::Type::E
```

Enumerator

NetSync	
---------	--

Definition at line 105 of file ArnServer.hpp.

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

- [src/ArnInc/ArnServer.hpp \(4.0.0\)](#)

14.93 ArnScriptJobs::Type Struct Reference

```
#include <ArnScriptJobs.hpp>
```

Public Types

- enum [E](#) { [Null](#), [Cooperative](#), [Preemptive](#) }

14.93.1 Detailed Description

Definition at line 164 of file ArnScriptJobs.hpp.

14.93.2 Member Enumeration Documentation

14.93.2.1 E

```
enum ArnScriptJobs::Type::E
```

Enumerator

Null	
Cooperative	
Preemptive	

Definition at line 165 of file ArnScriptJobs.hpp.

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

- [src/ArnInc/ArnScriptJobs.hpp \(4.0.0\)](#)

14.94 ArnDiscover::Type Struct Reference

Types of [Arn](#) discover advertise.

```
#include <ArnDiscover.hpp>
```

Public Types

- enum [E](#) { [None](#), [Server](#), [Client](#) }

14.94.1 Detailed Description

Types of [Arn](#) discover advertise.

Definition at line 52 of file ArnDiscover.hpp.

14.94.2 Member Enumeration Documentation

14.94.2.1 E

```
enum ArnDiscover::Type::E
```

Enumerator

None	Undefined Arn discover.
Server	Server Arn discover.
Client	Client Arn discover.

Definition at line 53 of file ArnDiscover.hpp.

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

- [src/ArnInc/ArnDiscover.hpp \(4.0.0\)](#)

14.95 ArnQml::UseFlags Struct Reference

```
#include <ArnQml.hpp>
```

Public Types

- enum `E` { `ArnLib` = 0x01, `MSystem` = 0x02, `MQt` = 0x04, `All` = 0xff }

14.95.1 Detailed Description

Definition at line 187 of file ArnQml.hpp.

14.95.2 Member Enumeration Documentation

14.95.2.1 E

```
enum ArnQml::UseFlags::E
```

Enumerator

ArnLib	Note: ArnLib is always included.
MSystem	Include some system fuctions like file-io.
MQt	Include some Qt extensions like MQtObject.
All	Include everything.

Definition at line 188 of file ArnQml.hpp.

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

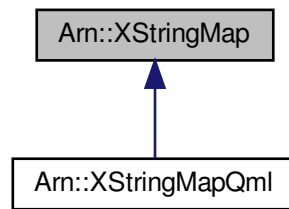
- [src/ArnInc/ArnQml.hpp \(4.0.0\)](#)

14.96 Arn::XStringMap Class Reference

Container class with string representation for serialized data.

```
#include <XStringMap.hpp>
```


Inheritance diagram for Arn::XStringMap:



Public Types

- typedef [XStringMapOptions](#) [Options](#)

Public Member Functions

- [XStringMap](#) ()
- [XStringMap](#) (const [XStringMap](#) &other)
 - Make shallow copy (Qt style)*
- [XStringMap](#) (const QByteArray &xString)
- [XStringMap](#) (const QVariantMap &variantMap)
- [~XStringMap](#) ()
- [XStringMap](#) & [operator=](#) (const [XStringMap](#) &other)
 - Make shallow copy (Qt style)*
- int [size](#) () const
- void [clear](#) (bool freeMem=false)
- void [squeeze](#) ()
- const [Options](#) & [options](#) () const
- void [setOptions](#) (const [Options](#) &newOptions)
- int [indexOf](#) (const char *key, int from=0) const
- int [indexOf](#) (const QByteArray &key, int from=0) const
- int [indexOf](#) (const QString &key, int from=0) const
- int [indexOfValue](#) (const QByteArray &value, int from=0) const
- int [indexOfValue](#) (const QString &value, int from=0) const
- int [maxEnumOf](#) (const char *keyPrefix) const
- [XStringMap](#) & [add](#) (const char *key, const QByteArray &val)
- [XStringMap](#) & [add](#) (const char *key, const char *val)
- [XStringMap](#) & [add](#) (const char *keyPrefix, uint eNum, const QByteArray &val)
- [XStringMap](#) & [add](#) (const QByteArray &key, const QByteArray &val)
- [XStringMap](#) & [add](#) (const char *key, const QString &val)
- [XStringMap](#) & [add](#) (const char *keyPrefix, uint eNum, const QString &val)
- [XStringMap](#) & [add](#) (const QByteArray &key, const QString &val)
- [XStringMap](#) & [add](#) (const QString &key, const QString &val)
- [XStringMap](#) & [add](#) (const [XStringMap](#) &other)
- [XStringMap](#) & [add](#) (const QVariantMap &variantMap)
- [XStringMap](#) & [addNum](#) (const char *key, int val)

- [XStringMap](#) & [addNum](#) (const QByteArray &[key](#), int val)
- [XStringMap](#) & [addNum](#) (const QString &[key](#), int val)
- [XStringMap](#) & [addNum](#) (const char *[key](#), uint val)
- [XStringMap](#) & [addNum](#) (const QByteArray &[key](#), uint val)
- [XStringMap](#) & [addNum](#) (const QString &[key](#), uint val)
- [XStringMap](#) & [addNum](#) (const char *[key](#), double val, int precision=-1)
- [XStringMap](#) & [addNum](#) (const QByteArray &[key](#), double val, int precision=-1)
- [XStringMap](#) & [addNum](#) (const QString &[key](#), double val, int precision=-1)
- [XStringMap](#) & [addValues](#) (const QStringList &stringList)
- [XStringMap](#) & [set](#) (int i, const QByteArray &val)
- [XStringMap](#) & [set](#) (int i, const QString &val)
- [XStringMap](#) & [set](#) (const char *[key](#), const QByteArray &val)
- [XStringMap](#) & [set](#) (const char *[key](#), const char *val)
- [XStringMap](#) & [set](#) (const QByteArray &[key](#), const QByteArray &val)
- [XStringMap](#) & [set](#) (const char *[key](#), const QString &val)
- [XStringMap](#) & [set](#) (const QByteArray &[key](#), const QString &val)
- [XStringMap](#) & [set](#) (const QString &[key](#), const QString &val)
- [XStringMap](#) & [setKey](#) (int i, const QByteArray &[key](#))
- const QByteArray & [keyRef](#) (int i) const
- QByteArray [key](#) (int i, const char *def=arnNullptr) const
- QByteArray [key](#) (const QByteArray &[value](#), const char *def=arnNullptr) const
- QByteArray [key](#) (const QString &[value](#), const char *def=arnNullptr) const
- QString [keyString](#) (int i, const QString &def=QString()) const
- QString [keyString](#) (const QString &[value](#), const QString &def=QString()) const
- const QByteArray & [valueRef](#) (int i) const
- QByteArray [value](#) (int i, const char *def=arnNullptr) const
- QByteArray [value](#) (const char *[key](#), const char *def=arnNullptr) const
- QByteArray [value](#) (const char *keyPrefix, uint eNum, const char *def=arnNullptr) const
- QByteArray [value](#) (const QByteArray &[key](#), const char *def=arnNullptr) const
- QByteArray [value](#) (const QByteArray &[key](#), const QByteArray &def) const
- QString [valueString](#) (int i, const QString &def=QString()) const
- QString [valueString](#) (const char *[key](#), const QString &def=QString()) const
- QString [valueString](#) (const char *keyPrefix, uint eNum, const QString &def=QString()) const
- QString [valueString](#) (const QByteArray &[key](#), const QString &def=QString()) const
- QString [valueString](#) (const QString &[key](#), const QString &def=QString()) const
- [XStringMap](#) & [remove](#) (int index)
- [XStringMap](#) & [remove](#) (const char *[key](#))
- [XStringMap](#) & [remove](#) (const QByteArray &[key](#))
- [XStringMap](#) & [remove](#) (const QString &[key](#))
- [XStringMap](#) & [removeValue](#) (const QByteArray &val)
- [XStringMap](#) & [removeValue](#) (const QString &val)
- QByteArray [toXString](#) () const
- QString [toXStringString](#) () const
- bool [fromXString](#) (const QByteArray &inXString, int [size](#)=-1)
- bool [fromXString](#) (const QString &inXString)
- void [setEmptyKeysToValue](#) ()
- void [reverseOrder](#) ()
- QStringList [keys](#) () const
- QStringList [values](#) (const char *keyPrefix=arnNullptr) const
- [MQVariantMap](#) [toVariantMap](#) (bool useStringVal) const
- void [stringCode](#) (QByteArray &dst, const QByteArray &src) const
- void [stringDecode](#) (QByteArray &dst, const QByteArray &src) const
- void [append](#) (const char *[key](#), const QByteArray &val)
- void [append](#) (const char *[key](#), const char *val)
- void [append](#) (const char *keyPrefix, uint eNum, const QByteArray &val)

- void [append](#) (const QByteArray &key, const QByteArray &val)
- void [append](#) (const char *key, const QString &val)
- void [append](#) (const char *keyPrefix, uint eNum, const QString &val)
- void [append](#) (const QByteArray &key, const QString &val)
- void [append](#) (const QString &key, const QString &val)
- void [append](#) (const XStringMap &other)
- void [append](#) (const QVariantMap &other)
- XStringMap & [operator+=](#) (const XStringMap &other)
- XStringMap & [operator+=](#) (const QVariantMap &other)
- QByteArray [info](#) ()

14.96.1 Detailed Description

Container class with string representation for serialized data.

The primary usage is for creating and parsing serialized data. it's optimized for giving an easy readable representation which never contains char codes below 32 (space).

This class can store data with a key like QMap. There is a guaranteed order of storing, i.e. its not sorted like QMap.

The stored data can be ascii as well as binary.

Following mapping is done when serialized to the XString:

```
Special codes below 32: code 0 -> "\0", code 10 -> "\n", code 13 -> "\r"
General codes below 32: code 1 -> "^A", code 2 -> "^B" and so on to code 31
code 32 (space) -> "_", "_" -> "\\_", "^" -> "\\^", "\" -> "\\\""
```

Key must be printable (char codes > 32). It must not contain " ", "=", "^" or "|" char. If this is not feasible, use option AnyKey.

The XString can be imported to the XStringMap. To get back stored values, XStringMap is Queried with the keys or by index.

```
Arn::XStringMap xsm;
xsm.add("", "put");
xsm.add("id", "level");
xsm.addNum("val", 12);
QDebug() << "XString: " << xsm.toXString();
```

This will print "XString: put id=level val=12"

Definition at line 107 of file XStringMap.hpp.

14.96.2 Member Typedef Documentation

14.96.2.1 Options

```
typedef XStringMapOptions Arn::XStringMap::Options
```

Definition at line 110 of file XStringMap.hpp.

14.96.3 Constructor & Destructor Documentation

14.96.3.1 XStringMap() [1/4]

```
Arn::XStringMap::XStringMap ( )
```

Definition at line 54 of file ArnXStringMap.cpp.

14.96.3.2 XStringMap() [2/4]

```
Arn::XStringMap::XStringMap (
    const XStringMap & other )
```

Make shallow copy (Qt style)

Definition at line 60 of file ArnXStringMap.cpp.

14.96.3.3 XStringMap() [3/4]

```
Arn::XStringMap::XStringMap (
    const QByteArray & xString ) [explicit]
```

Definition at line 68 of file ArnXStringMap.cpp.

14.96.3.4 XStringMap() [4/4]

```
Arn::XStringMap::XStringMap (
    const QVariantMap & variantMap ) [explicit]
```

Definition at line 75 of file ArnXStringMap.cpp.

14.96.3.5 ~XStringMap()

```
Arn::XStringMap::~XStringMap ( )
```

Definition at line 82 of file ArnXStringMap.cpp.

14.96.4 Member Function Documentation

14.96.4.1 add() [1/10]

```
XStringMap & Arn::XStringMap::add (
    const char * key,
    const QByteArray & val )
```

Definition at line 199 of file ArnXStringMap.cpp.

14.96.4.2 add() [2/10]

```
XStringMap & Arn::XStringMap::add (
    const char * key,
    const char * val )
```

Definition at line 214 of file ArnXStringMap.cpp.

14.96.4.3 add() [3/10]

```
XStringMap & Arn::XStringMap::add (
    const char * keyPrefix,
    uint eNum,
    const QByteArray & val )
```

Definition at line 220 of file ArnXStringMap.cpp.

14.96.4.4 add() [4/10]

```
XStringMap & Arn::XStringMap::add (
    const QByteArray & key,
    const QByteArray & val )
```

Definition at line 230 of file ArnXStringMap.cpp.

14.96.4.5 add() [5/10]

```
XStringMap & Arn::XStringMap::add (
    const char * key,
    const QString & val )
```

Definition at line 236 of file ArnXStringMap.cpp.

14.96.4.6 add() [6/10]

```
XStringMap & Arn::XStringMap::add (
    const char * keyPrefix,
    uint eNum,
    const QString & val )
```

Definition at line 242 of file ArnXStringMap.cpp.

14.96.4.7 add() [7/10]

```
XStringMap & Arn::XStringMap::add (
    const QByteArray & key,
    const QString & val )
```

Definition at line 248 of file ArnXStringMap.cpp.

14.96.4.8 add() [8/10]

```
XStringMap & Arn::XStringMap::add (
    const QString & key,
    const QString & val )
```

Definition at line 254 of file ArnXStringMap.cpp.

14.96.4.9 add() [9/10]

```
XStringMap & Arn::XStringMap::add (
    const XStringMap & other )
```

Definition at line 260 of file ArnXStringMap.cpp.

14.96.4.10 add() [10/10]

```
XStringMap & Arn::XStringMap::add (
    const QVariantMap & variantMap )
```

Definition at line 270 of file ArnXStringMap.cpp.

14.96.4.11 addNum() [1/9]

```
XStringMap & Arn::XStringMap::addNum (
    const char * key,
    int val )
```

Definition at line 286 of file ArnXStringMap.cpp.

14.96.4.12 addNum() [2/9]

```
XStringMap & Arn::XStringMap::addNum (
    const QByteArray & key,
    int val )
```

Definition at line 292 of file ArnXStringMap.cpp.

14.96.4.13 addNum() [3/9]

```
XStringMap & Arn::XStringMap::addNum (
    const QString & key,
    int val )
```

Definition at line 298 of file ArnXStringMap.cpp.

14.96.4.14 addNum() [4/9]

```
XStringMap & Arn::XStringMap::addNum (
    const char * key,
    uint val )
```

Definition at line 304 of file ArnXStringMap.cpp.

14.96.4.15 addNum() [5/9]

```
XStringMap & Arn::XStringMap::addNum (
    const QByteArray & key,
    uint val )
```

Definition at line 310 of file ArnXStringMap.cpp.

14.96.4.16 addNum() [6/9]

```
XStringMap & Arn::XStringMap::addNum (
    const QString & key,
    uint val )
```

Definition at line 316 of file ArnXStringMap.cpp.

14.96.4.17 addNum() [7/9]

```
XStringMap & Arn::XStringMap::addNum (
    const char * key,
    double val,
    int precision = -1 )
```

Definition at line 322 of file ArnXStringMap.cpp.

14.96.4.18 addNum() [8/9]

```
XStringMap & Arn::XStringMap::addNum (
    const QByteArray & key,
    double val,
    int precision = -1 )
```

Definition at line 329 of file ArnXStringMap.cpp.

14.96.4.19 addNum() [9/9]

```
XStringMap & Arn::XStringMap::addNum (
    const QString & key,
    double val,
    int precision = -1 )
```

Definition at line 336 of file ArnXStringMap.cpp.

14.96.4.20 addValues()

```
XStringMap & Arn::XStringMap::addValues (
    const QStringList & stringList )
```

Definition at line 343 of file ArnXStringMap.cpp.

14.96.4.21 append() [1/10]

```
void Arn::XStringMap::append (
    const char * key,
    const QByteArray & val ) [inline]
```

Definition at line 208 of file XStringMap.hpp.

14.96.4.22 append() [2/10]

```
void Arn::XStringMap::append (
    const char * key,
    const char * val ) [inline]
```

Definition at line 210 of file XStringMap.hpp.

14.96.4.23 append() [3/10]

```
void Arn::XStringMap::append (
    const char * keyPrefix,
    uint eNum,
    const QByteArray & val ) [inline]
```

Definition at line 212 of file XStringMap.hpp.

14.96.4.24 append() [4/10]

```
void Arn::XStringMap::append (
    const QByteArray & key,
    const QByteArray & val ) [inline]
```

Definition at line 214 of file XStringMap.hpp.

14.96.4.25 `append()` [5/10]

```
void Arn::XStringMap::append (
    const char * key,
    const QString & val ) [inline]
```

Definition at line 216 of file XStringMap.hpp.

14.96.4.26 `append()` [6/10]

```
void Arn::XStringMap::append (
    const char * keyPrefix,
    uint eNum,
    const QString & val ) [inline]
```

Definition at line 218 of file XStringMap.hpp.

14.96.4.27 `append()` [7/10]

```
void Arn::XStringMap::append (
    const QByteArray & key,
    const QString & val ) [inline]
```

Definition at line 220 of file XStringMap.hpp.

14.96.4.28 `append()` [8/10]

```
void Arn::XStringMap::append (
    const QString & key,
    const QString & val ) [inline]
```

Definition at line 222 of file XStringMap.hpp.

14.96.4.29 `append()` [9/10]

```
void Arn::XStringMap::append (
    const XStringMap & other ) [inline]
```

Definition at line 224 of file XStringMap.hpp.

14.96.4.30 `append()` [10/10]

```
void Arn::XStringMap::append (
    const QVariantMap & other ) [inline]
```

Definition at line 226 of file XStringMap.hpp.

14.96.4.31 `clear()`

```
void Arn::XStringMap::clear (
    bool freeMem = false )
```

Definition at line 104 of file ArnXStringMap.cpp.

14.96.4.32 `fromXString()` [1/2]

```
bool Arn::XStringMap::fromXString (
    const QByteArray & inXString,
    int size = -1 )
```

Definition at line 740 of file ArnXStringMap.cpp.

14.96.4.33 `fromXString()` [2/2]

```
bool Arn::XStringMap::fromXString (
    const QString & inXString )
```

Definition at line 820 of file ArnXStringMap.cpp.

14.96.4.34 `indexOf()` [1/3]

```
int Arn::XStringMap::indexOf (
    const char * key,
    int from = 0 ) const
```

Definition at line 136 of file ArnXStringMap.cpp.

14.96.4.35 indexOf() [2/3]

```
int Arn::XStringMap::indexOf (
    const QByteArray & key,
    int from = 0 ) const
```

Definition at line 149 of file ArnXStringMap.cpp.

14.96.4.36 indexOf() [3/3]

```
int Arn::XStringMap::indexOf (
    const QString & key,
    int from = 0 ) const
```

Definition at line 160 of file ArnXStringMap.cpp.

14.96.4.37 indexOfValue() [1/2]

```
int Arn::XStringMap::indexOfValue (
    const QByteArray & value,
    int from = 0 ) const
```

Definition at line 166 of file ArnXStringMap.cpp.

14.96.4.38 indexOfValue() [2/2]

```
int Arn::XStringMap::indexOfValue (
    const QString & value,
    int from = 0 ) const
```

Definition at line 177 of file ArnXStringMap.cpp.

14.96.4.39 info()

```
QByteArray Arn::XStringMap::info ( )
```

Definition at line 1074 of file ArnXStringMap.cpp.

14.96.4.40 key() [1/3]

```
QByteArray Arn::XStringMap::key (
    int i,
    const char * def = arnNullptr ) const
```

Definition at line 431 of file ArnXStringMap.cpp.

14.96.4.41 key() [2/3]

```
QByteArray Arn::XStringMap::key (
    const QByteArray & value,
    const char * def = arnNullptr ) const
```

Definition at line 439 of file ArnXStringMap.cpp.

14.96.4.42 key() [3/3]

```
QByteArray Arn::XStringMap::key (
    const QString & value,
    const char * def = arnNullptr ) const
```

Definition at line 448 of file ArnXStringMap.cpp.

14.96.4.43 keyRef()

```
const QByteArray & Arn::XStringMap::keyRef (
    int i ) const
```

Definition at line 423 of file ArnXStringMap.cpp.

14.96.4.44 keys()

```
QStringList Arn::XStringMap::keys ( ) const
```

Definition at line 647 of file ArnXStringMap.cpp.

14.96.4.45 keyString() [1/2]

```
QString Arn::XStringMap::keyString (
    int i,
    const QString & def = QString() ) const
```

Definition at line 454 of file ArnXStringMap.cpp.

14.96.4.46 keyString() [2/2]

```
QString Arn::XStringMap::keyString (
    const QString & value,
    const QString & def = QString() ) const
```

Definition at line 463 of file ArnXStringMap.cpp.

14.96.4.47 maxEnumOf()

```
int Arn::XStringMap::maxEnumOf (
    const char * keyPrefix ) const
```

Definition at line 183 of file ArnXStringMap.cpp.

14.96.4.48 operator+=() [1/2]

```
XStringMap & Arn::XStringMap::operator+= (
    const XStringMap & other )
```

Definition at line 1068 of file ArnXStringMap.cpp.

14.96.4.49 operator+=() [2/2]

```
XStringMap & Arn::XStringMap::operator+= (
    const QVariantMap & other )
```

Definition at line 1062 of file ArnXStringMap.cpp.

14.96.4.50 operator=()

```
XStringMap & Arn::XStringMap::operator= (
    const XStringMap & other )
```

Make shallow copy (Qt style)

Definition at line 87 of file ArnXStringMap.cpp.

14.96.4.51 options()

```
const XStringMap::Options & Arn::XStringMap::options ( ) const
```

Definition at line 124 of file ArnXStringMap.cpp.

14.96.4.52 remove() [1/4]

```
XStringMap & Arn::XStringMap::remove (
    int index )
```

Definition at line 569 of file ArnXStringMap.cpp.

14.96.4.53 remove() [2/4]

```
XStringMap & Arn::XStringMap::remove (
    const char * key )
```

Definition at line 585 of file ArnXStringMap.cpp.

14.96.4.54 remove() [3/4]

```
XStringMap & Arn::XStringMap::remove (
    const QByteArray & key )
```

Definition at line 591 of file ArnXStringMap.cpp.

14.96.4.55 remove() [4/4]

```
XStringMap & Arn::XStringMap::remove (
    const QString & key )
```

Definition at line 597 of file ArnXStringMap.cpp.

14.96.4.56 removeValue() [1/2]

```
XStringMap & Arn::XStringMap::removeValue (
    const QByteArray & val )
```

Definition at line 603 of file ArnXStringMap.cpp.

14.96.4.57 removeValue() [2/2]

```
XStringMap & Arn::XStringMap::removeValue (
    const QString & val )
```

Definition at line 609 of file ArnXStringMap.cpp.

14.96.4.58 reverseOrder()

```
void Arn::XStringMap::reverseOrder ( )
```

Definition at line 626 of file ArnXStringMap.cpp.

14.96.4.59 set() [1/8]

```
XStringMap & Arn::XStringMap::set (
    int i,
    const QByteArray & val )
```

Definition at line 353 of file ArnXStringMap.cpp.

14.96.4.60 set() [2/8]

```
XStringMap & Arn::XStringMap::set (
    int i,
    const QString & val )
```

Definition at line 364 of file ArnXStringMap.cpp.

14.96.4.61 set() [3/8]

```
XStringMap & Arn::XStringMap::set (
    const char * key,
    const QByteArray & val )
```

Definition at line 370 of file ArnXStringMap.cpp.

14.96.4.62 set() [4/8]

```
XStringMap & Arn::XStringMap::set (
    const char * key,
    const char * val )
```

Definition at line 382 of file ArnXStringMap.cpp.

14.96.4.63 set() [5/8]

```
XStringMap & Arn::XStringMap::set (
    const QByteArray & key,
    const QByteArray & val )
```

Definition at line 388 of file ArnXStringMap.cpp.

14.96.4.64 set() [6/8]

```
XStringMap & Arn::XStringMap::set (
    const char * key,
    const QString & val )
```

Definition at line 394 of file ArnXStringMap.cpp.

14.96.4.65 set() [7/8]

```
XStringMap & Arn::XStringMap::set (
    const QByteArray & key,
    const QString & val )
```

Definition at line 400 of file ArnXStringMap.cpp.

14.96.4.66 set() [8/8]

```
XStringMap & Arn::XStringMap::set (
    const QString & key,
    const QString & val )
```

Definition at line 406 of file ArnXStringMap.cpp.

14.96.4.67 setEmptyKeysToValue()

```
void Arn::XStringMap::setEmptyKeysToValue ( )
```

Definition at line 615 of file ArnXStringMap.cpp.

14.96.4.68 setKey()

```
XStringMap & Arn::XStringMap::setKey (
    int i,
    const QByteArray & key )
```

Definition at line 412 of file ArnXStringMap.cpp.

14.96.4.69 setOptions()

```
void Arn::XStringMap::setOptions (
    const Options & newOptions )
```

Definition at line 130 of file ArnXStringMap.cpp.

14.96.4.70 size()

```
int Arn::XStringMap::size ( ) const [inline]
```

Definition at line 121 of file XStringMap.hpp.

14.96.4.71 squeeze()

```
void Arn::XStringMap::squeeze ( )
```

Definition at line 115 of file ArnXStringMap.cpp.

14.96.4.72 stringCode()

```
void Arn::XStringMap::stringCode (
    QByteArray & dst,
    const QByteArray & src ) const
```

Definition at line 826 of file ArnXStringMap.cpp.

14.96.4.73 stringDecode()

```
void Arn::XStringMap::stringDecode (
    QByteArray & dst,
    const QByteArray & src ) const
```

Definition at line 970 of file ArnXStringMap.cpp.

14.96.4.74 toVariantMap()

```
MQVariantMap Arn::XStringMap::toVariantMap (
    bool useStringValue ) const
```

Definition at line 673 of file ArnXStringMap.cpp.

14.96.4.75 toXString()

```
QByteArray Arn::XStringMap::toXString ( ) const
```

Definition at line 688 of file ArnXStringMap.cpp.

14.96.4.76 toXStringString()

```
QString Arn::XStringMap::toXStringString ( ) const
```

Definition at line 733 of file ArnXStringMap.cpp.

14.96.4.77 value() [1/5]

```
QByteArray Arn::XStringMap::value (
    int i,
    const char * def = arnNullptr ) const
```

Definition at line 478 of file ArnXStringMap.cpp.

14.96.4.78 value() [2/5]

```
QByteArray Arn::XStringMap::value (
    const char * key,
    const char * def = arnNullptr ) const
```

Definition at line 486 of file ArnXStringMap.cpp.

14.96.4.79 value() [3/5]

```
QByteArray Arn::XStringMap::value (
    const char * keyPrefix,
    uint eNum,
    const char * def = arnNullptr ) const
```

Definition at line 495 of file ArnXStringMap.cpp.

14.96.4.80 value() [4/5]

```
QByteArray Arn::XStringMap::value (
    const QByteArray & key,
    const char * def = arnNullptr ) const
```

Definition at line 508 of file ArnXStringMap.cpp.

14.96.4.81 value() [5/5]

```
QByteArray Arn::XStringMap::value (
    const QByteArray & key,
    const QByteArray & def ) const
```

Definition at line 517 of file ArnXStringMap.cpp.

14.96.4.82 valueRef()

```
const QByteArray & Arn::XStringMap::valueRef (
    int i ) const
```

Definition at line 470 of file ArnXStringMap.cpp.

14.96.4.83 values()

```
QStringList Arn::XStringMap::values (
    const char * keyPrefix = arnNullptr ) const
```

Definition at line 658 of file ArnXStringMap.cpp.

14.96.4.84 valueString() [1/5]

```
QString Arn::XStringMap::valueString (
    int i,
    const QString & def = QString() ) const
```

Definition at line 527 of file ArnXStringMap.cpp.

14.96.4.85 valueString() [2/5]

```
QString Arn::XStringMap::valueString (
    const char * key,
    const QString & def = QString() ) const
```

Definition at line 536 of file ArnXStringMap.cpp.

14.96.4.86 valueString() [3/5]

```
QString Arn::XStringMap::valueString (
    const char * keyPrefix,
    uint eNum,
    const QString & def = QString() ) const
```

Definition at line 543 of file ArnXStringMap.cpp.

14.96.4.87 valueString() [4/5]

```
QString Arn::XStringMap::valueString (
    const QByteArray & key,
    const QString & def = QString() ) const
```

Definition at line 555 of file ArnXStringMap.cpp.

14.96.4.88 valueString() [5/5]

```
QString Arn::XStringMap::valueString (
    const QString & key,
    const QString & def = QString() ) const
```

Definition at line 562 of file ArnXStringMap.cpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/XStringMap.hpp \(4.0.0\)](#)
- [src/ArnXStringMap.cpp \(4.0.0\)](#)

14.97 Arn::XStringMapOptions Class Reference

```
#include <XStringMap.hpp>
```

Public Types

- enum [E](#) {
[None](#) = 0x00, [NullTilde](#) = 0x01, [RepeatLen](#) = 0x02, [Frame](#) = 0x04,
[AnyKey](#) = 0x08, [Supported](#) = 0x0f }

14.97.1 Detailed Description

Definition at line 58 of file XStringMap.hpp.

14.97.2 Member Enumeration Documentation

14.97.2.1 E

```
enum Arn::XStringMapOptions::E
```

Enumerator

None	
NullTilde	
RepeatLen	
Frame	
AnyKey	
Supported	Convenience.

Definition at line 62 of file XStringMap.hpp.

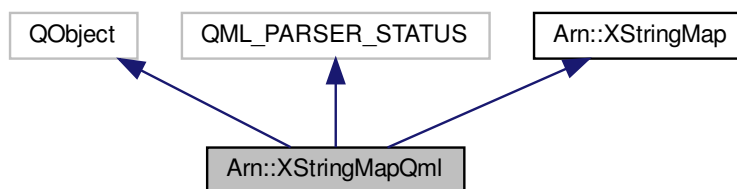
The documentation for this class was generated from the following file:

- [src/ArnInc/XStringMap.hpp \(4.0.0\)](#)

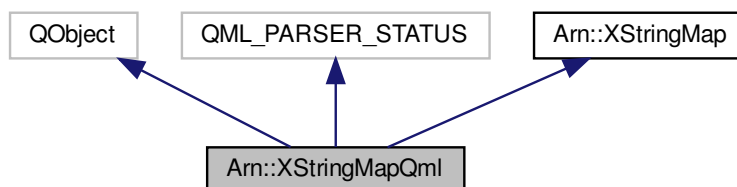
14.98 Arn::XStringMapQml Class Reference

```
#include <ArnQml.hpp>
```

Inheritance diagram for Arn::XStringMapQml:



Collaboration diagram for Arn::XStringMapQml:



Public Slots

- void `clear` ()
 - Clear and free mem.*
- int `indexOf` (const QString &key, int from=0) const
- int `indexOfValue` (const QString &value, int from=0) const
- QObject * `add` (const QString &key, const QString &val)
- QObject * `add` (QObject *other)
- QObject * `set` (int i, const QString &val)
- QObject * `set` (const QString &key, const QString &val)
- QString `key` (int i, const QString &def=QString()) const
- QString `key` (const QString &value, const QString &def=QString()) const
- QString `value` (int i, const QString &def=QString()) const
- QString `value` (const QString &key, const QString &def=QString()) const
- QObject * `remove` (int index)
- QObject * `remove` (const QString &key)
- QObject * `removeValue` (const QString &val)
- void `setEmptyKeysToValue` ()
- QStringList `keys` () const
- QStringList `values` () const
- `MQVariantMap toMap` () const

Properties

- QString `xstring`
 - The map serialized as xstring.*
- int `size`
 - Number of items.*

Additional Inherited Members

14.98.1 Detailed Description

Definition at line 637 of file ArnQml.hpp.

14.98.2 Member Function Documentation

14.98.2.1 `add` [1/2]

```
QObject* Arn::XStringMapQml::add (
    const QString & key,
    const QString & val ) [inline], [slot]
```

Definition at line 672 of file ArnQml.hpp.

14.98.2.2 add [2/2]

```
QObject * XStringMapQml::add (
    QObject * other ) [slot]
```

Definition at line 450 of file ArnQml.cpp.

14.98.2.3 clear

```
void Arn::XStringMapQml::clear ( ) [inline], [slot]
```

Clear and free mem.

Definition at line 663 of file ArnQml.hpp.

14.98.2.4 indexOf

```
int Arn::XStringMapQml::indexOf (
    const QString & key,
    int from = 0 ) const [inline], [slot]
```

Definition at line 666 of file ArnQml.hpp.

14.98.2.5 indexOfValue

```
int Arn::XStringMapQml::indexOfValue (
    const QString & value,
    int from = 0 ) const [inline], [slot]
```

Definition at line 669 of file ArnQml.hpp.

14.98.2.6 key [1/2]

```
QString Arn::XStringMapQml::key (
    int i,
    const QString & def = QString() ) const [inline], [slot]
```

Definition at line 683 of file ArnQml.hpp.

14.98.2.7 key [2/2]

```
QString Arn::XStringMapQml::key (
    const QString & value,
    const QString & def = QString() ) const [inline], [slot]
```

Definition at line 686 of file ArnQml.hpp.

14.98.2.8 keys

```
QStringList Arn::XStringMapQml::keys ( ) const [inline], [slot]
```

Definition at line 707 of file ArnQml.hpp.

14.98.2.9 remove [1/2]

```
QObject* Arn::XStringMapQml::remove (
    int index ) [inline], [slot]
```

Definition at line 695 of file ArnQml.hpp.

14.98.2.10 remove [2/2]

```
QObject* Arn::XStringMapQml::remove (
    const QString & key ) [inline], [slot]
```

Definition at line 698 of file ArnQml.hpp.

14.98.2.11 removeValue

```
QObject* Arn::XStringMapQml::removeValue (
    const QString & val ) [inline], [slot]
```

Definition at line 701 of file ArnQml.hpp.

14.98.2.12 set [1/2]

```
QObject* Arn::XStringMapQml::set (
    int i,
    const QString & val ) [inline], [slot]
```

Definition at line 677 of file ArnQml.hpp.

14.98.2.13 set [2/2]

```
QObject* Arn::XStringMapQml::set (
    const QString & key,
    const QString & val ) [inline], [slot]
```

Definition at line 680 of file ArnQml.hpp.

14.98.2.14 setEmptyKeysToValue

```
void Arn::XStringMapQml::setEmptyKeysToValue ( ) [inline], [slot]
```

Definition at line 704 of file ArnQml.hpp.

14.98.2.15 toMap

```
MQVariantMap Arn::XStringMapQml::toMap ( ) const [inline], [slot]
```

Definition at line 713 of file ArnQml.hpp.

14.98.2.16 value [1/2]

```
QString Arn::XStringMapQml::value (
    int i,
    const QString & def = QString() ) const [inline], [slot]
```

Definition at line 689 of file ArnQml.hpp.

14.98.2.17 value [2/2]

```
QString Arn::XStringMapQml::value (
    const QString & key,
    const QString & def = QString() ) const [inline], [slot]
```

Definition at line 692 of file ArnQml.hpp.

14.98.2.18 values

```
QStringList Arn::XStringMapQml::values ( ) const [inline], [slot]
```

Definition at line 710 of file ArnQml.hpp.

14.98.3 Property Documentation

14.98.3.1 size

```
int Arn::XStringMapQml::size [read]
```

Number of items.

Definition at line 650 of file ArnQml.hpp.

14.98.3.2 xstring

```
QString Arn::XStringMapQml::xstring [read], [write]
```

The map serialized as xstring.

Definition at line 648 of file ArnQml.hpp.

The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQml.hpp \(4.0.0\)](#)
- [src/ArnQml.cpp \(4.0.0\)](#)

Chapter 15

File Documentation

15.1 `doc/Changelog_Todo.md` File Reference

15.2 `doc/Description.md` File Reference

15.3 `doc/HelpIndex.txt` File Reference

15.4 `doc/Install.md` File Reference

15.5 `doc/Internals.md` File Reference

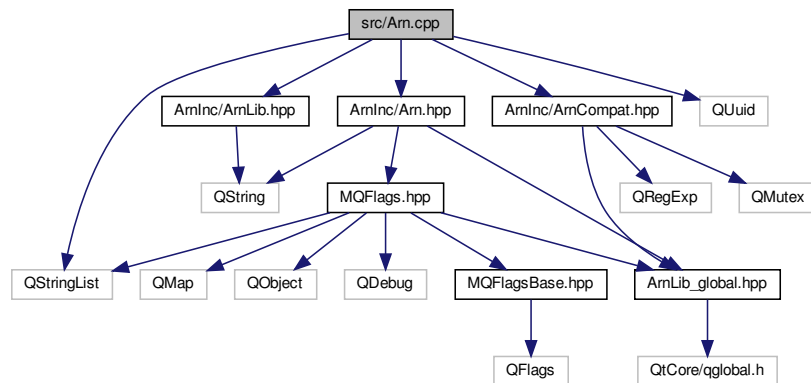
15.6 `examples/Examples.txt` File Reference

15.7 `README.md` File Reference

15.8 `src/Arn.cpp` File Reference

```
#include "ArnInc/Arn.hpp"  
#include "ArnInc/ArnLib.hpp"  
#include "ArnInc/ArnCompat.hpp"  
#include <QUuid>
```

```
#include <QStringList>
Include dependency graph for Arn.cpp:
```



Namespaces

- [Arn](#)

Functions

- `QString Arn::convertName` (const QString &name, [Arn::NameF](#) nameF=[Arn::NameF\(\)](#))
Convert a name to a specific format.
- `QString Arn::fullPath` (const QString &path)
Convert a path to a full absolute path.
- `QString Arn::itemName` (const QString &path)
The last part of a path
- `QString Arn::childPath` (const QString &parentPath, const QString &posterityPath)
Get substring for child from a path (posterityPath)
- `QString Arn::changeBasePath` (const QString &oldBasePath, const QString &newBasePath, const QString &path)
Change the base (start) of a path.
- `QString Arn::makePath` (const QString &parentPath, const QString &itemName)
Make a path from a parent and an item name.
- `QString Arn::addPath` (const QString &parentPath, const QString &childRelPath, [Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#))
Make a path from a parent and an additional relative path.
- `QString Arn::convertPath` (const QString &path, [Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#))
Convert a path to a specific format.
- `QString Arn::parentPath` (const QString &path)
Get the parent to a given path
- `QString Arn::twinPath` (const QString &path)
Get the bidirectional twin to a given path
- `QString Arn::providerPathIf` (const QString &path, bool giveProviderPath=true)
Get provider path or requester path
- `bool Arn::isFolderPath` (const QString &path)
Test if path is a folder path

- bool [Arn::isProviderPath](#) (const QString &path)
Test if path is a provider path
- QString [Arn::uuidPath](#) (const QString &path)
Get a path to an [Arn](#) Object with a unique uuid name.
- QString [Arn::makeHostWithInfo](#) (const QString &host, const QString &info)
Make a combined host and info string, i.e. HostWithInfo
- QString [Arn::hostFromHostWithInfo](#) (const QString &hostWithInfo)
Get the host from the HostWithInfo string.
- bool [Arn::isNullPtr](#) (const void *ptr)
- uint [Arn::rand](#) ()

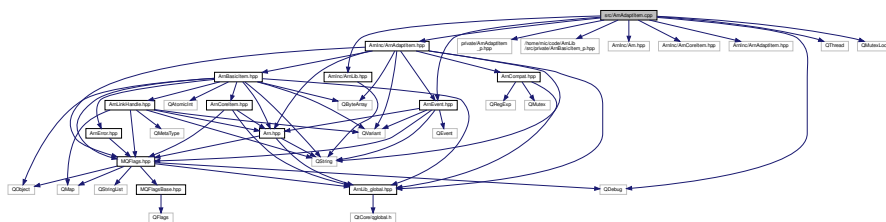
Variables

- const QString [Arn::pathLocal](#) = "/Local/"
- const QString [Arn::pathLocalSys](#) = "Sys/"
- const QString [Arn::pathDiscover](#) = "Sys/Discover/"
- const QString [Arn::pathDiscoverThis](#) = "Sys/Discover/This/"
- const QString [Arn::pathDiscoverConnect](#) = "Sys/Discover/Connect/"
- const QString [Arn::pathServer](#) = "Sys/Server/"
- const QString [Arn::pathServerSessions](#) = "Sys/Server/Sessions/"

15.9 src/ArnAdaptItem.cpp File Reference

```
#include "ArnInc/ArnAdaptItem.hpp"
#include "private/ArnAdaptItem_p.hpp"
#include "ArnInc/ArnEvent.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QThread>
#include <QMutexLocker>
#include <QDebug>
```

Include dependency graph for ArnAdaptItem.cpp:



Macros

- #define [MUTEX_CALL](#)(funcCall)
- #define [MUTEX_CALL_RET](#)(funcCall)

15.9.1 Macro Definition Documentation

15.9.1.1 MUTEX_CALL

```
#define MUTEX_CALL(  
    funcCall )
```

Value:

```
d->_mutex.lock(); \  
    funcCall; \  
    d->_mutex.unlock();
```

Definition at line 40 of file ArnAdaptItem.cpp.

15.9.1.2 MUTEX_CALL_RET

```
#define MUTEX_CALL_RET(  
    funcCall )
```

Value:

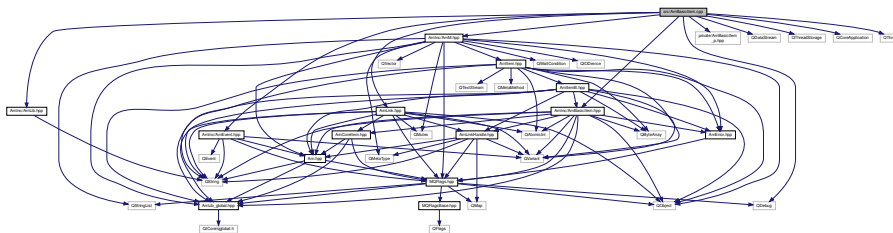
```
QMutexLocker mutexLocker( &d->_mutex); \  
    return funcCall;
```

Definition at line 45 of file ArnAdaptItem.cpp.

15.10 src/ArnBasicItem.cpp File Reference

```
#include "ArnInc/ArnBasicItem.hpp"  
#include "private/ArnBasicItem_p.hpp"  
#include "ArnInc/ArnM.hpp"  
#include "ArnInc/ArnEvent.hpp"  
#include "ArnInc/ArnLib.hpp"  
#include "ArnLink.hpp"  
#include <QDataStream>  
#include <QThreadStorage>  
#include <QCoreApplication>  
#include <QThread>  
#include <QDebug>
```

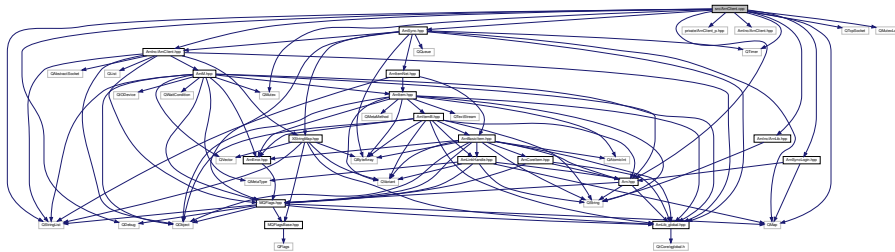
Include dependency graph for ArnBasicItem.cpp:



15.11 src/ArnClient.cpp File Reference

```
#include "ArnInc/ArnClient.hpp"
#include "private/ArnClient_p.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnSync.hpp"
#include "ArnSyncLogin.hpp"
#include <QTcpSocket>
#include <QStringList>
#include <QTimer>
#include <QMap>
#include <QMutexLocker>
#include <QDebug>
```

Include dependency graph for ArnClient.cpp:



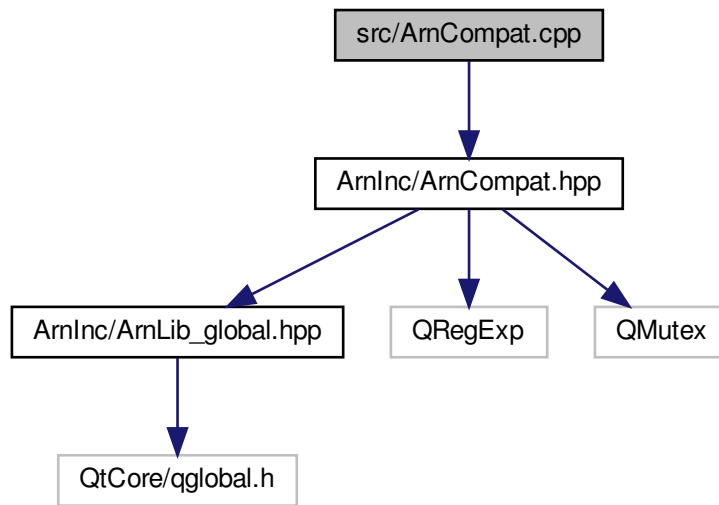
Classes

- class [ArnClientReg](#)

15.12 src/ArnCompat.cpp File Reference

```
#include "ArnInc/ArnCompat.hpp"
```

Include dependency graph for ArnCompat.cpp:



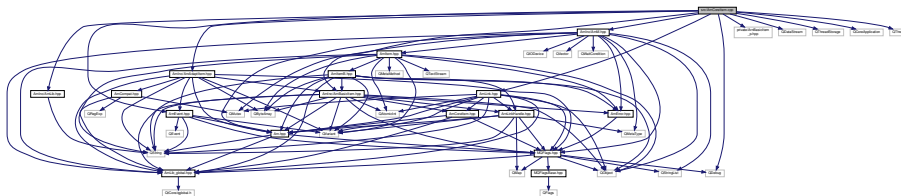
15.13 src/ArnCoreItem.cpp File Reference

```

#include "ArnInc/ArnBasicItem.hpp"
#include "ArnInc/ArnAdaptItem.hpp"
#include "private/ArnBasicItem_p.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnInc/ArnEvent.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnLink.hpp"
#include <QDataStream>
#include <QThreadStorage>
#include <QCoreApplication>
#include <QThread>
#include <QDebug>

```

Include dependency graph for ArnCoreItem.cpp:



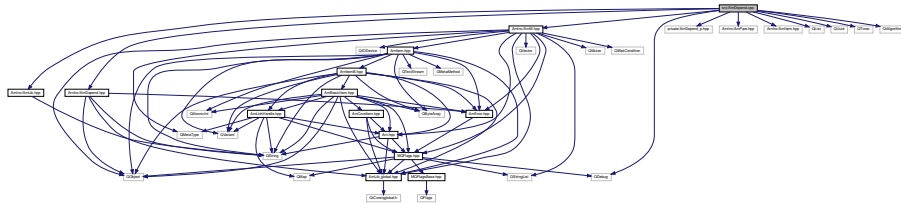
15.14 src/ArnDepend.cpp File Reference

```

#include "ArnInc/ArnDepend.hpp"
#include "private/ArnDepend_p.hpp"

```

```
#include "ArnInc/ArnM.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QUuid>
#include <QTimer>
#include <QtAlgorithms>
#include <QDebug>
Include dependency graph for ArnDepend.cpp:
```



Variables

- const char * `ArnDependPath` = `"//.sys/Depend/"`

15.14.1 Variable Documentation

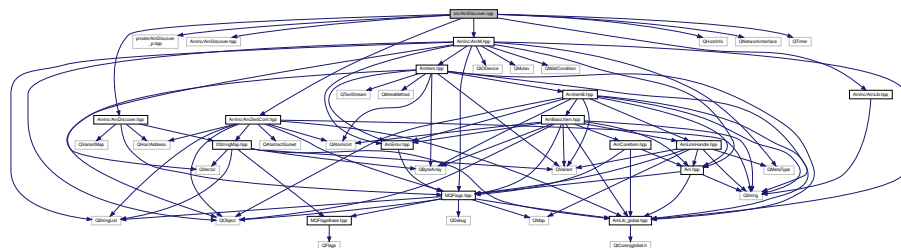
15.14.1.1 ArnDependPath

```
const char* ArnDependPath = "//.sys/Depend/"
```

Definition at line 41 of file ArnDepend.cpp.

15.15 src/ArnDiscover.cpp File Reference

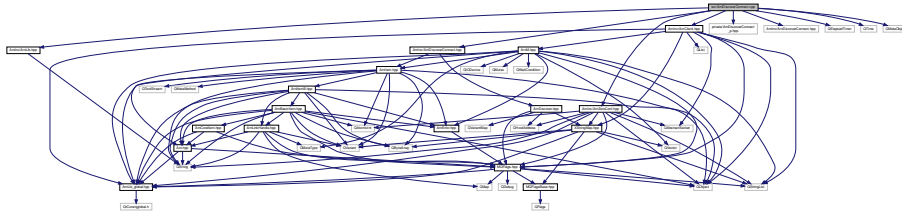
```
#include "ArnInc/ArnDiscover.hpp"
#include "private/ArnDiscover_p.hpp"
#include "ArnInc/ArnZeroConf.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QHostInfo>
#include <QNetworkInterface>
#include <QTimer>
Include dependency graph for ArnDiscover.cpp:
```



15.16 src/ArnDiscoverConnect.cpp File Reference

```
#include "ArnInc/ArnDiscoverConnect.hpp"
#include "private/ArnDiscoverConnect_p.hpp"
#include "ArnInc/ArnZeroConf.hpp"
#include "ArnInc/ArnClient.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QElapsedTimer>
#include <QTime>
#include <QMetaObject>
```

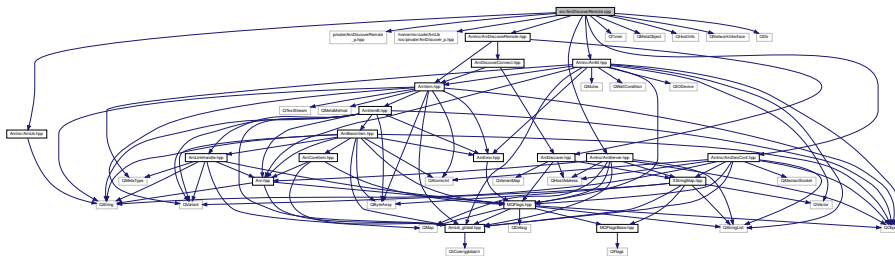
Include dependency graph for ArnDiscoverConnect.cpp:



15.17 src/ArnDiscoverRemote.cpp File Reference

```
#include "ArnInc/ArnDiscoverRemote.hpp"
#include "private/ArnDiscoverRemote_p.hpp"
#include "ArnInc/ArnZeroConf.hpp"
#include "ArnInc/ArnServer.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QTimer>
#include <QMetaObject>
#include <QHostInfo>
#include <QNetworkInterface>
#include <QDir>
```

Include dependency graph for ArnDiscoverRemote.cpp:

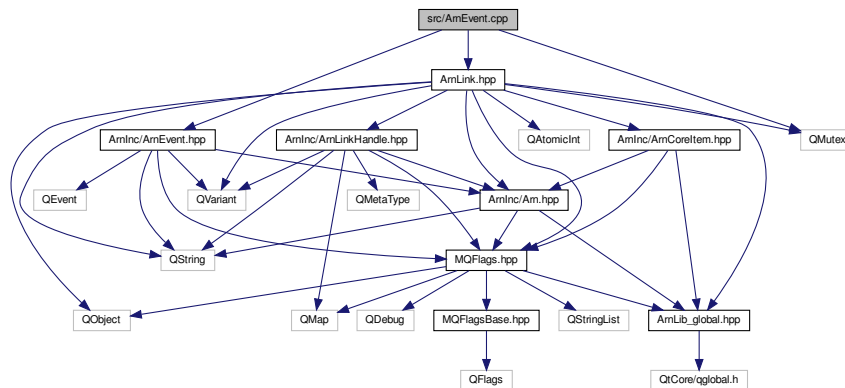


15.18 src/ArnEvent.cpp File Reference

```
#include <ArnInc/ArnEvent.hpp>
#include <ArnLink.hpp>
```

```
#include <QMutex>
```

Include dependency graph for ArnEvent.cpp:



Macros

- `#define TO_IDX_RETVAL(evType)`

15.18.1 Macro Definition Documentation

15.18.1.1 TO_IDX_RETVAL

```
#define TO_IDX_RETVAL(  
    evType )
```

Value:

```
int retVal = (evType) - baseType(); \  
retVal = ((retVal >= 0) && (retVal < Idx::N)) ? retVal : Idx::QtEvent;
```

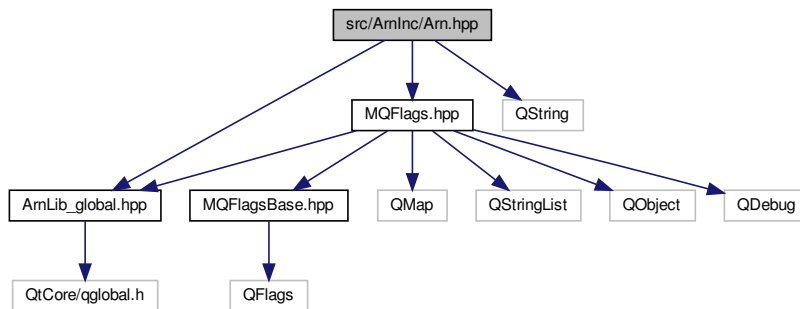
Definition at line 97 of file ArnEvent.cpp.

15.19 src/ArnInc/Arn.hpp File Reference

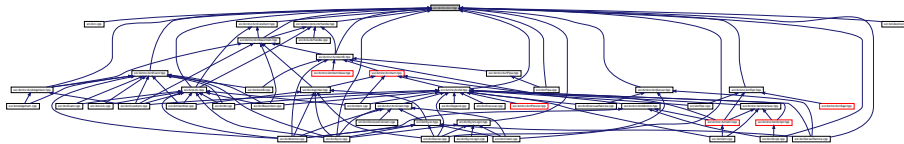
```
#include "MQFlags.hpp"  
#include "ArnLib_global.hpp"
```

```
#include <QString>
```

Include dependency graph for Arn.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct [Arn::SameValue](#)
Action when assigning same value to an [ArnItem](#).
- class [Arn::DataType](#)
Data type of an [Arn](#) Data Object
- class [Arn::ExportCode](#)
Code used in blob for `arnExport()` and `arnImport()`
- struct [Arn::InfoType](#)
Info type for exchange static (meta) info between [ArnClient](#) and [ArnServer](#).
- class [Arn::ObjectMode](#)
- class [Arn::ObjectSyncMode](#)
- struct [Arn::ClientSyncMode](#)
The Client session Sync mode at connect & reconnect.
- struct [Arn::LinkFlags](#)
Link flags when accessing an [Arn](#) Data Object
- struct [Arn::NameF](#)
- struct [Arn::Coding](#)
- class [Arn::Allow](#)

Namespaces

- [Arn](#)

Macros

- #define [DATASTREAM_VER](#) QDataStream::Qt_4_6
- #define [ARNREAL](#) double

Functions

- QString [Arn::convertName](#) (const QString &name, [Arn::NameF](#) nameF=[Arn::NameF\(\)](#))
Convert a name to a specific format.
- QString [Arn::fullPath](#) (const QString &path)
Convert a path to a full absolute path.
- bool [Arn::isFolderPath](#) (const QString &path)
Test if path is a folder path
- bool [Arn::isProviderPath](#) (const QString &path)
Test if path is a provider path
- QString [Arn::itemName](#) (const QString &path)
The last part of a path
- QString [Arn::childPath](#) (const QString &parentPath, const QString &posterityPath)
Get substring for child from a path (posterityPath)
- QString [Arn::changeBasePath](#) (const QString &oldBasePath, const QString &newBasePath, const QString &path)
Change the base (start) of a path.
- QString [Arn::makePath](#) (const QString &parentPath, const QString &itemName)
Make a path from a parent and an item name.
- QString [Arn::addPath](#) (const QString &parentPath, const QString &childRelPath, [Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#))
Make a path from a parent and an additional relative path.
- QString [Arn::convertPath](#) (const QString &path, [Arn::NameF](#) nameF=[Arn::NameF::EmptyOk](#))
Convert a path to a specific format.
- QString [Arn::parentPath](#) (const QString &path)
Get the parent to a given path
- QString [Arn::twinPath](#) (const QString &path)
Get the bidirectional twin to a given path
- QString [Arn::providerPathlf](#) (const QString &path, bool giveProviderPath=true)
Get provider path or requester path
- QString [Arn::uuidPath](#) (const QString &path)
Get a path to an Arn Object with a unique uuid name.
- QString [Arn::makeHostWithInfo](#) (const QString &host, const QString &info)
Make a combined host and info string, i.e. HostWithInfo
- QString [Arn::hostFromHostWithInfo](#) (const QString &hostWithInfo)
Get the host from the HostWithInfo string.
- bool [Arn::isNullPtr](#) (const void *ptr)
- uint [Arn::rand](#) ()

Variables

- const quint16 [Arn::defaultTcpPort](#) = 2022

15.19.1 Macro Definition Documentation

15.19.1.1 ARNREAL

```
#define ARNREAL double
```

Definition at line 44 of file Arn.hpp.

15.19.1.2 DATASTREAM_VER

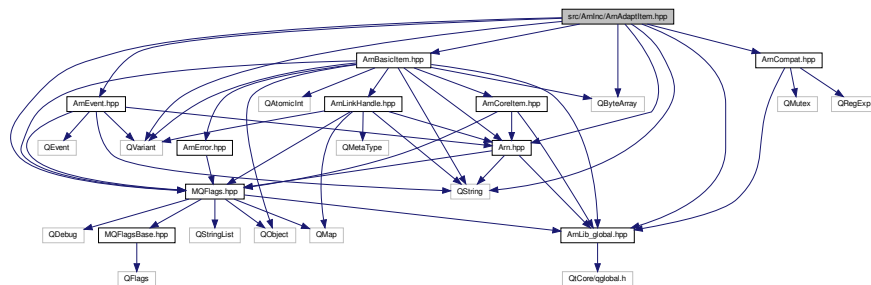
```
#define DATASTREAM_VER QDataStream::Qt_4_6
```

Definition at line 39 of file Arn.hpp.

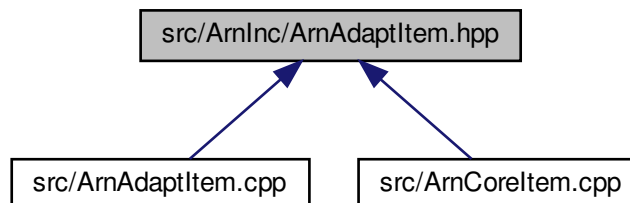
15.20 src/ArnInc/ArnAdaptItem.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "Arn.hpp"
#include "ArnBasicItem.hpp"
#include "ArnEvent.hpp"
#include "ArnCompat.hpp"
#include "MQFlags.hpp"
#include <QString>
#include <QByteArray>
#include <QVariant>
```

Include dependency graph for ArnAdaptItem.hpp:



This graph shows which files directly or indirectly include this file:



Classes

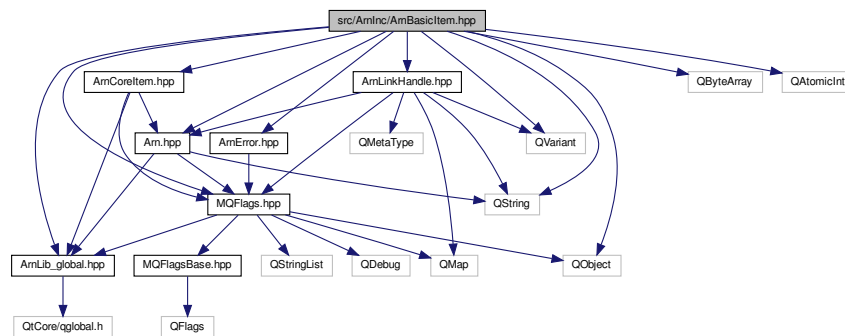
- class [ArnAdaptItem](#)

! Non Qt and threadsafe handle for an [Arn](#) Data Object.

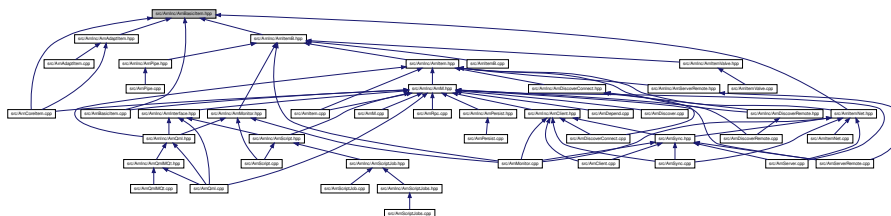
15.21 src/ArnInc/ArnBasicItem.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnCoreItem.hpp"
#include "ArnLinkHandle.hpp"
#include "ArnError.hpp"
#include "Arn.hpp"
#include "MQFlags.hpp"
#include <QString>
#include <QByteArray>
#include <QVariant>
#include <QAtomicInt>
#include <QObject>
```

Include dependency graph for ArnBasicItem.hpp:



This graph shows which files directly or indirectly include this file:



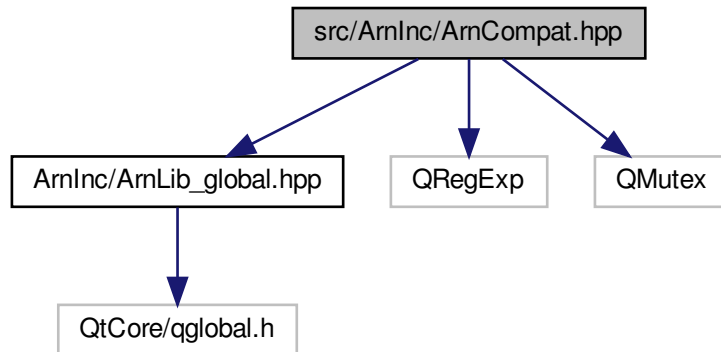
Classes

- class [ArnBasicItem](#)

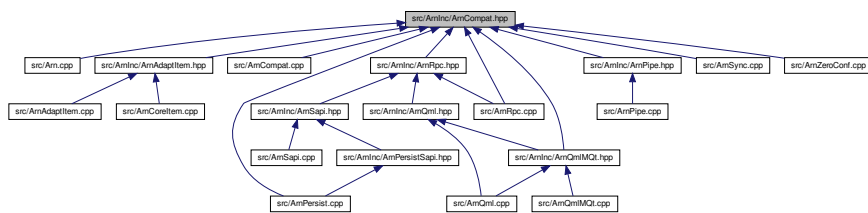
Base class handle for an [Arn](#) Data Object.


```
#include <QMutex>
```

Include dependency graph for ArnCompat.hpp:



This graph shows which files directly or indirectly include this file:



Macros

- `#define ARN_RegExp` `QRegExp`
- `#define ARN_RegExpValidator` `QRegExpValidator`
- `#define ARN_ToRegExp` `toRegExp`
- `#define ARN_RecursiveMutex` `QMutex`
- `#define ARN_ModeRecursiveMutex` `QMutex::Recursive`
- `#define ARN_SIZETYPE` `int`

15.23.1 Macro Definition Documentation

15.23.1.1 ARN_ModeRecursiveMutex

```
#define ARN_ModeRecursiveMutex QMutex::Recursive
```

Definition at line 74 of file `ArnCompat.hpp`.

15.23.1.2 ARN_RecursiveMutex

```
#define ARN_RecursiveMutex QMutex
```

Definition at line 73 of file ArnCompat.hpp.

15.23.1.3 ARN_RegExp

```
#define ARN_RegExp QRegExp
```

Definition at line 70 of file ArnCompat.hpp.

15.23.1.4 ARN_RegExpValidator

```
#define ARN_RegExpValidator QRegExpValidator
```

Definition at line 71 of file ArnCompat.hpp.

15.23.1.5 ARN_SIZETYPE

```
#define ARN_SIZETYPE int
```

Definition at line 75 of file ArnCompat.hpp.

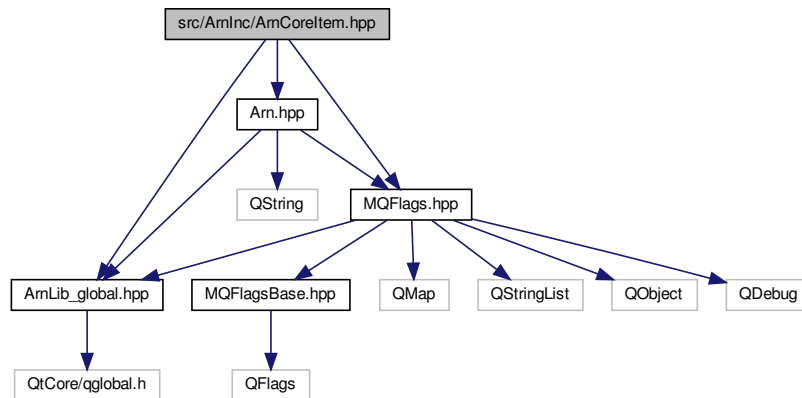
15.23.1.6 ARN_ToRegExp

```
#define ARN_ToRegExp toRegExp
```

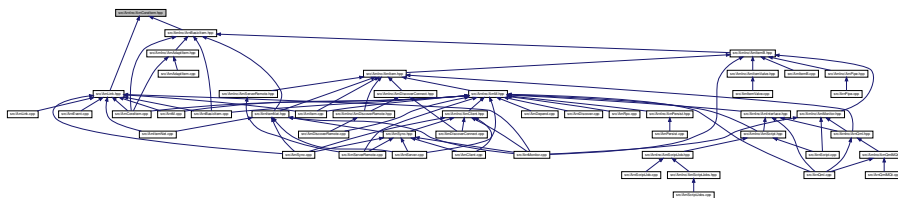
Definition at line 72 of file ArnCompat.hpp.

15.24 src/ArnInc/ArnCoreItem.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "Arn.hpp"
#include "MQFlags.hpp"
Include dependency graph for ArnCoreItem.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

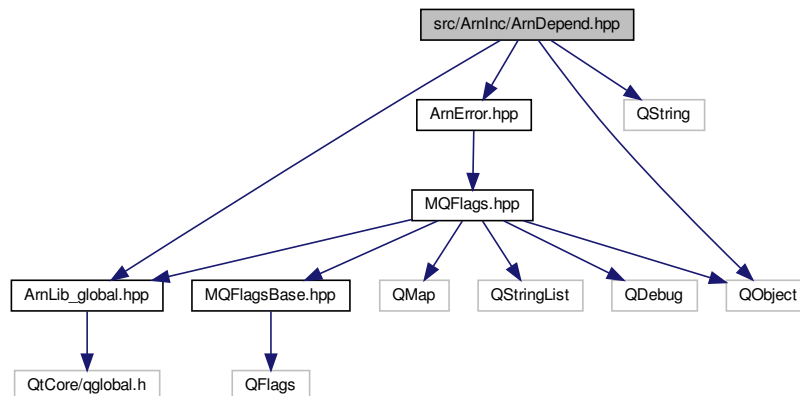
- class [ArnCoreItem](#)
Core base class for the inherited [ArnItem](#) classes.
- struct [ArnCoreItem::Heritage](#)

15.25 src/ArnInc/ArnDepend.hpp File Reference

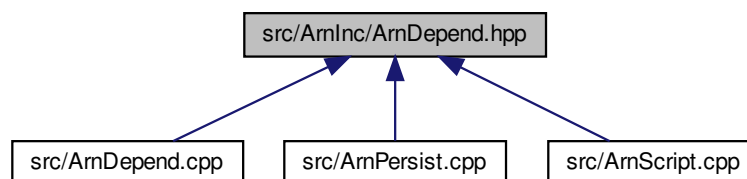
```
#include "ArnLib_global.hpp"
#include "ArnError.hpp"
#include <QString>
```

```
#include <QObject>
```

Include dependency graph for ArnDepend.hpp:



This graph shows which files directly or indirectly include this file:



Classes

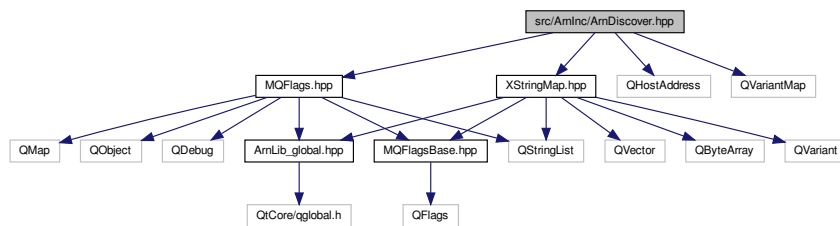
- class [ArnDependOffer](#)
Class for advertising that a service is available.
- class [ArnDepend](#)
Class for setting up dependencis to needed services.

15.26 src/ArnInc/ArnDiscover.hpp File Reference

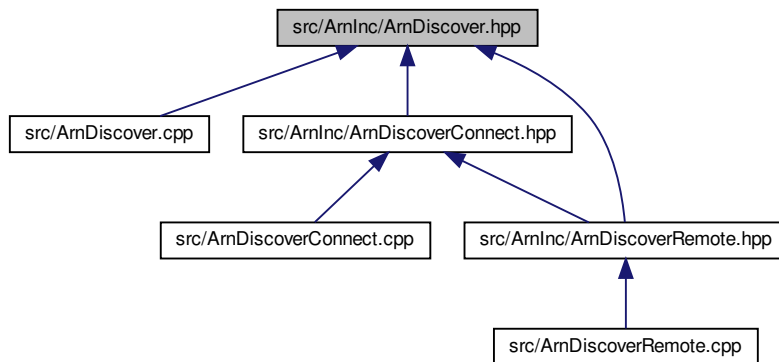
```
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QHostAddress>
```

```
#include <QVariantMap>
```

Include dependency graph for ArnDiscover.hpp:



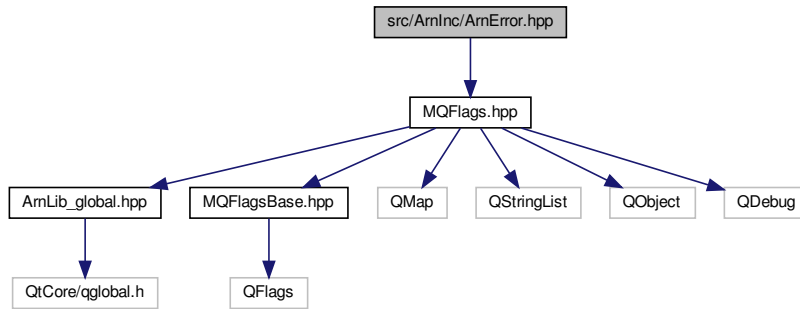
This graph shows which files directly or indirectly include this file:



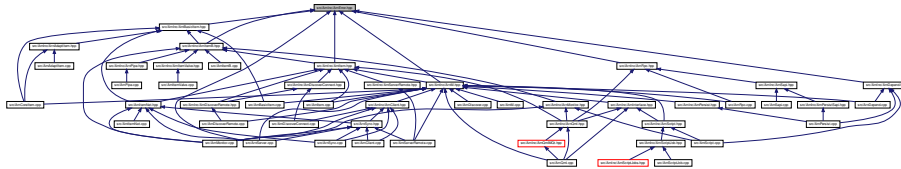
Classes

- struct [ArnDiscover::Type](#)
Types of Arn discover advertise.
- class [ArnDiscoverInfo](#)
Class for holding current discover info of one service.
- struct [ArnDiscoverInfo::State](#)
State of Arn discover browse data. Can be tested by relative order.
- class [ArnDiscoverBrowserB](#)
Browse() and resolve() together, may never be used to the same instance.
- class [ArnDiscoverBrowser](#)
Browsing for Arn services.
- class [ArnDiscoverResolver](#)
Resolv an Arn service.
- class [ArnDiscoverAdvertise](#)
Advertise an Arn service.
- struct [ArnDiscoverAdvertise::State](#)
States of DiscoverAdvertise / These values must be synced with: [ArnZeroConf::State](#).

Include dependency graph for ArnError.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnError](#)
- struct [ArnError::StdCode](#)

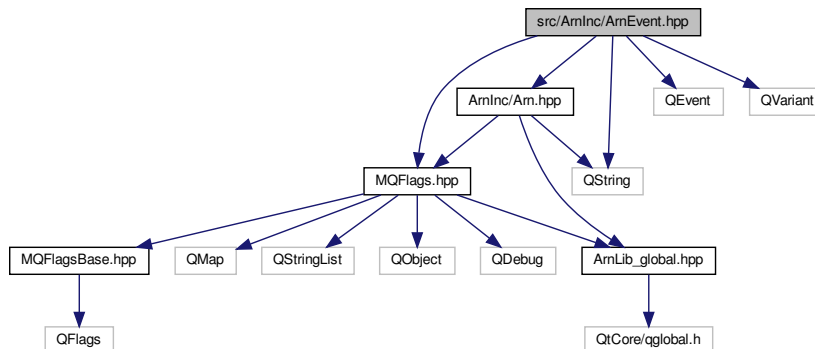
15.30 src/ArnInc/ArnEvent.hpp File Reference

```

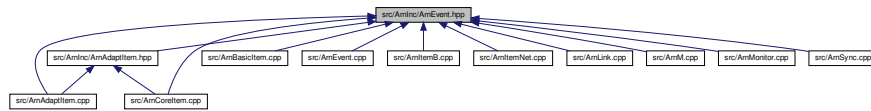
#include "ArnInc/Arn.hpp"
#include "ArnInc/MQFlags.hpp"
#include <QEvent>
#include <QString>
#include <QVariant>

```

Include dependency graph for ArnEvent.hpp:



This graph shows which files directly or indirectly include this file:

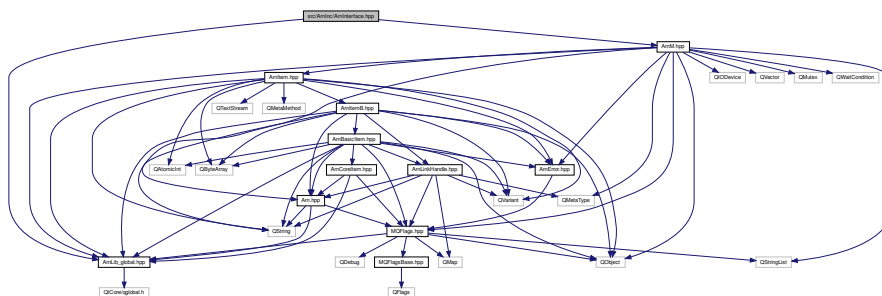


Classes

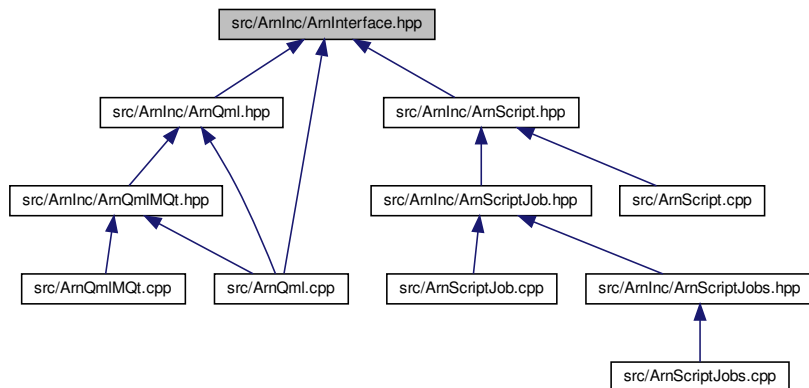
- class [ArnEventIdx](#)
- class [ArnAtomicOp](#)
- class [ArnEvent](#)
- class [ArnEvLinkCreate](#)
- class [ArnEvModeChange](#)
- class [ArnEvMonitor](#)
- class [ArnEvRetired](#)
- class [ArnEvZeroRef](#)
- class [ArnEvValueChange](#)
- class [ArnEvAtomicOp](#)
- class [ArnEvRefChange](#)

15.31 src/ArnInc/ArnInterface.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnM.hpp"
Include dependency graph for ArnInterface.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnInterface](#)

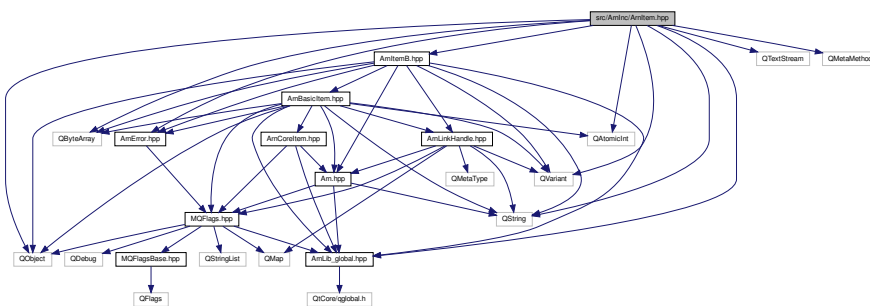
15.32 src/ArnInc/ArnItem.hpp File Reference

```

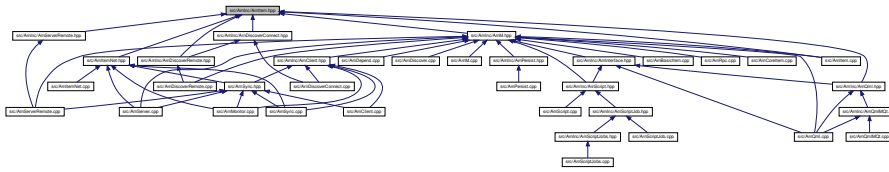
#include "ArnLib_global.hpp"
#include "ArnItemB.hpp"
#include "ArnError.hpp"
#include <QTextStream>
#include <QObject>
#include <QMetaMethod>
#include <QString>
#include <QByteArray>
#include <QVariant>
#include <QAtomicInt>

```

Include dependency graph for ArnItem.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnItem](#)
Handle for an *Arn* Data Object.

Functions

- QTextStream & [operator<<](#) (QTextStream &out, const [ArnItem](#) &item)

15.32.1 Function Documentation

15.32.1.1 operator<<()

```
QTextStream& operator<< (
    QTextStream & out,
    const ArnItem & item )
```

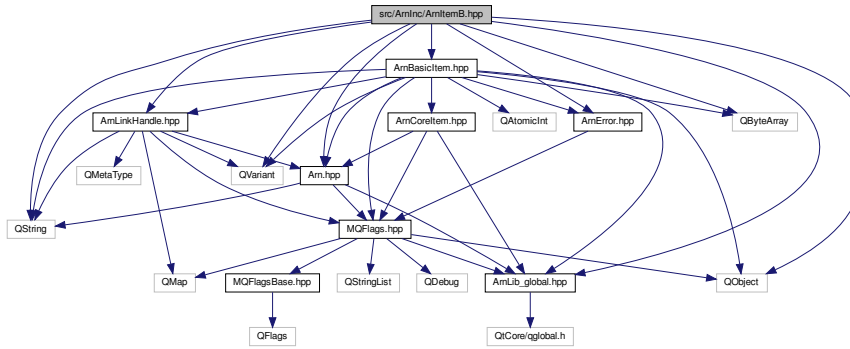
Definition at line 550 of file ArnItem.cpp.

15.33 src/ArnInc/ArnItemB.hpp File Reference

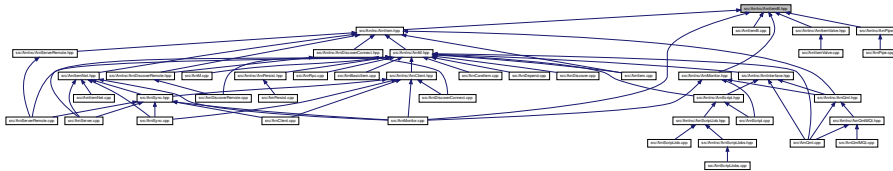
```
#include "ArnLib_global.hpp"
#include "ArnLinkHandle.hpp"
#include "ArnError.hpp"
#include "Arn.hpp"
#include "ArnBasicItem.hpp"
#include <QObject>
#include <QString>
#include <QByteArray>
```

```
#include <QVariant>
```

Include dependency graph for ArnItemB.hpp:



This graph shows which files directly or indirectly include this file:



Classes

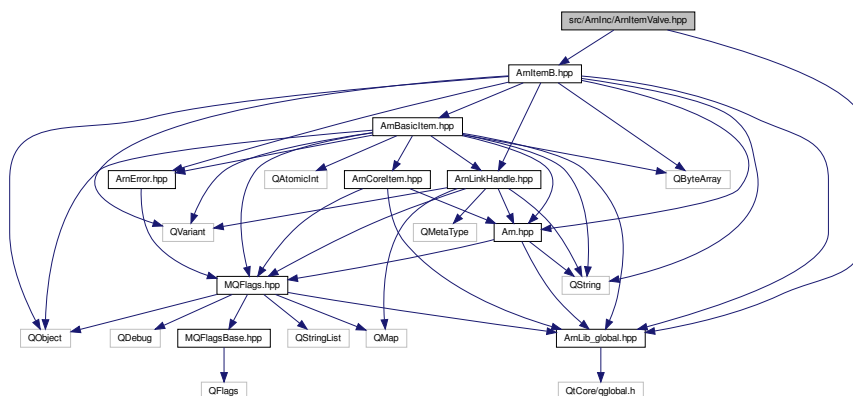
- class [ArnItemB](#)
Base class handle for an *Arn* Data Object.

15.34 src/ArnInc/ArnItemValve.hpp File Reference

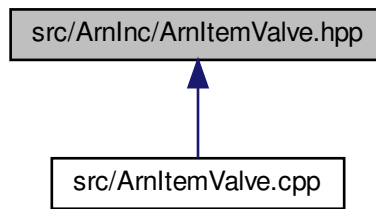
```
#include "ArnLib_global.hpp"
```

```
#include "ArnItemB.hpp"
```

Include dependency graph for ArnItemValve.hpp:



This graph shows which files directly or indirectly include this file:



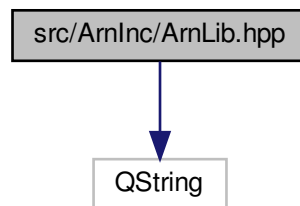
Classes

- class [ArnItemValve](#)
Valve for controlling stream to/from an [ArnItemB](#).
- struct [ArnItemValve::SwitchMode](#)

15.35 src/ArnInc/ArnLib.hpp File Reference

```
#include <QString>
```

Include dependency graph for ArnLib.hpp:



This graph shows which files directly or indirectly include this file:



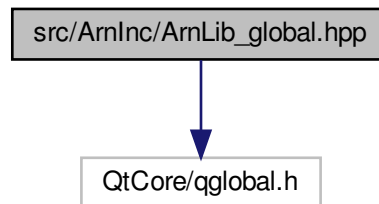
Namespaces

- [Arn](#)

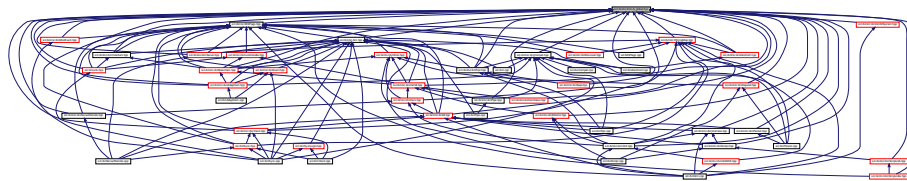
15.36 src/ArnInc/ArnLib_global.hpp File Reference

```
#include <QtCore/qglobal.h>
```

Include dependency graph for ArnLib_global.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct [ArnNullptr](#)

Macros

- #define [ARNLIBSHARED_EXPORT](#) Q_DECL_IMPORT

15.36.1 Macro Definition Documentation

15.36.1.1 ARNLIBSHARED_EXPORT

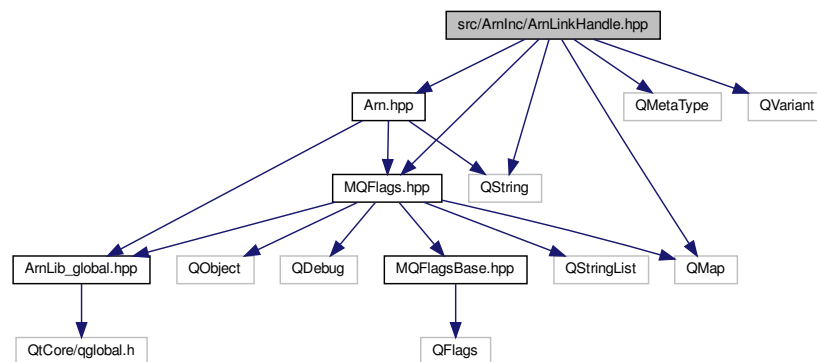
```
#define ARNLIBSHARED_EXPORT Q_DECL_IMPORT
```

Definition at line 11 of file ArnLib_global.hpp.

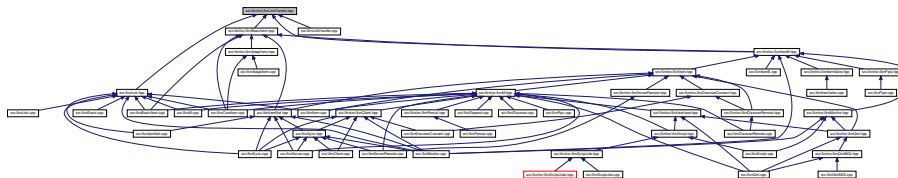
15.37 src/ArnInc/ArnLinkHandle.hpp File Reference

```
#include "Arn.hpp"
#include "MQFlags.hpp"
#include <QMetaType>
#include <QString>
#include <QVariant>
#include <QMap>
```

Include dependency graph for ArnLinkHandle.hpp:



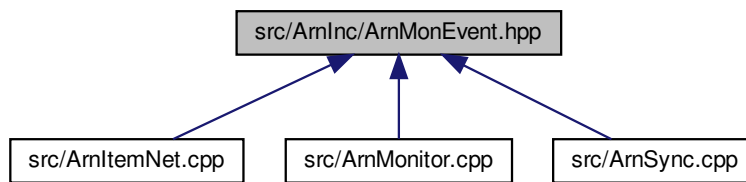
This graph shows which files directly or indirectly include this file:



15.38 src/ArnInc/ArnM.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "Arn.hpp"
#include "MQFlags.hpp"
#include "ArnError.hpp"
#include "ArnItem.hpp"
#include <QIODevice>
#include <QStringList>
#include <QVector>
#include <QMetaType>
#include <QObject>
#include <QMutex>
```


This graph shows which files directly or indirectly include this file:



Classes

- class [ArnMonEventType](#)

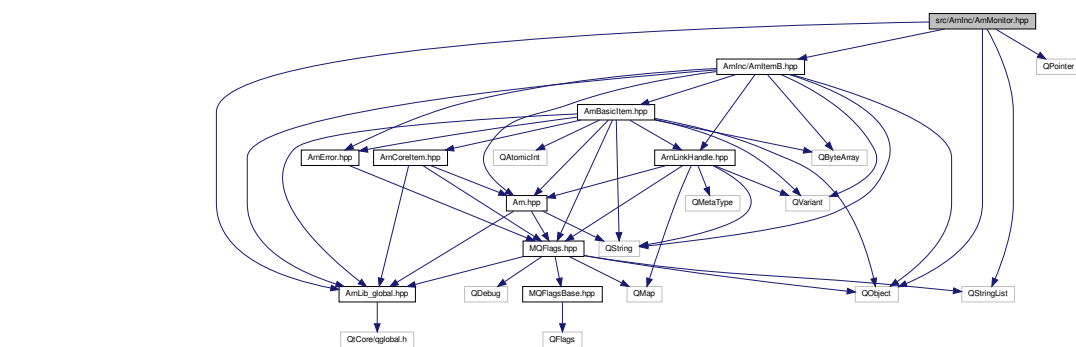
15.40 src/ArnInc/ArnMonitor.hpp File Reference

```

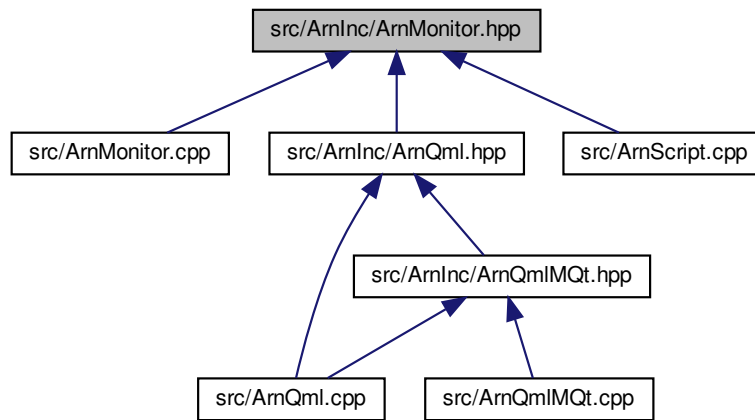
#include "ArnLib_global.hpp"
#include "ArnInc/ArnItemB.hpp"
#include <QStringList>
#include <QObject>
#include <QPointer>

```

Include dependency graph for ArnMonitor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnMonitor](#)

A client remote monitor to detect changes at server.

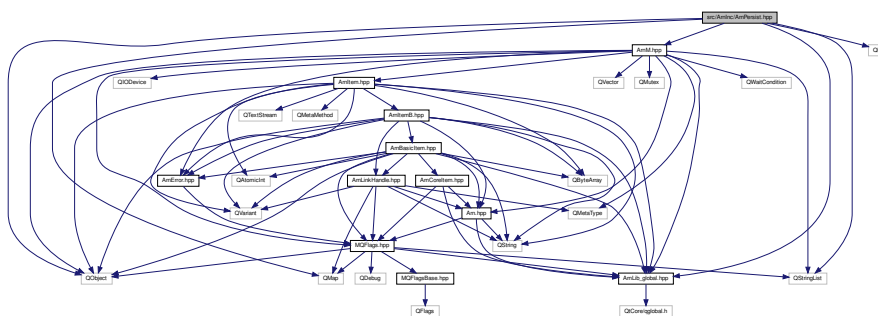
15.41 src/ArnInc/ArnPersist.hpp File Reference

```

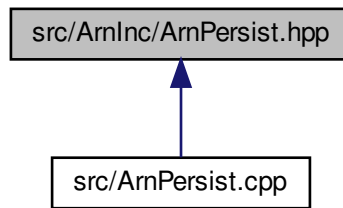
#include "ArnLib_global.hpp"
#include "ArnM.hpp"
#include <QMap>
#include <QList>
#include <QStringList>
#include <QObject>

```

Include dependency graph for ArnPersist.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnPersist](#)
Class for handling persistent [Arn](#) Data object.

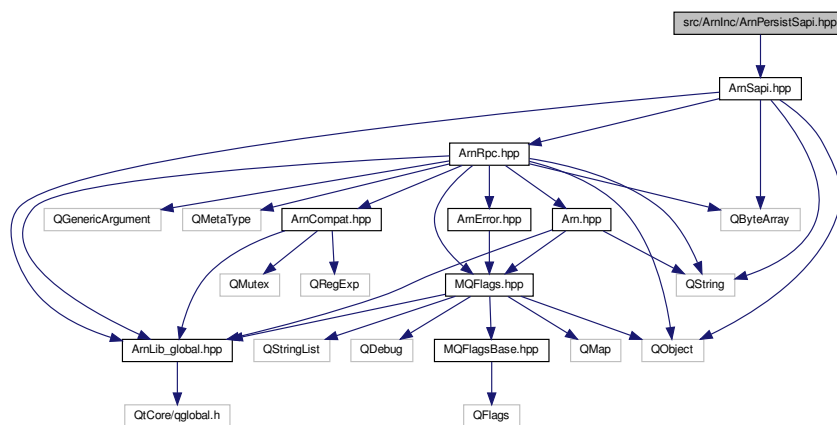
Namespaces

- [Arn](#)

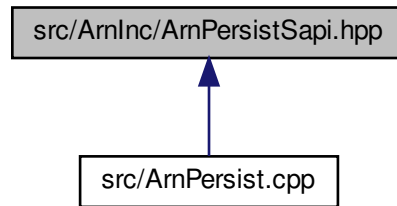
15.42 src/ArnInc/ArnPersistSapi.hpp File Reference

```
#include "ArnSapi.hpp"
```

Include dependency graph for ArnPersistSapi.hpp:



This graph shows which files directly or indirectly include this file:



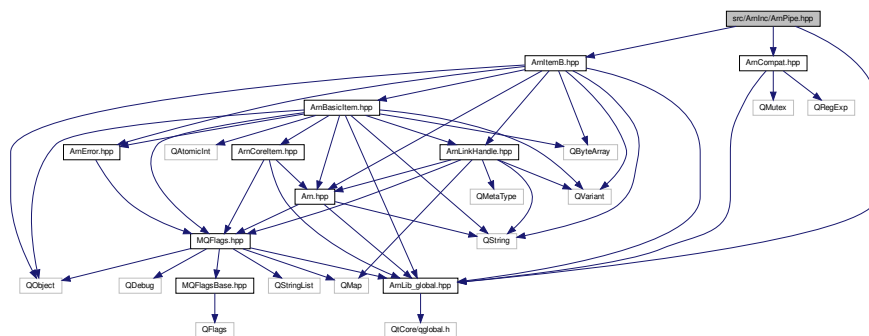
15.43 src/ArnInc/ArnPipe.hpp File Reference

```

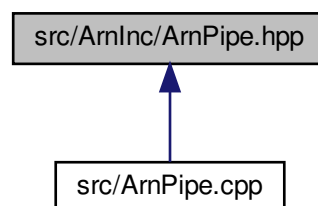
#include "ArnLib_global.hpp"
#include "ArnItemB.hpp"
#include "ArnCompat.hpp"

```

Include dependency graph for ArnPipe.hpp:



This graph shows which files directly or indirectly include this file:



Classes

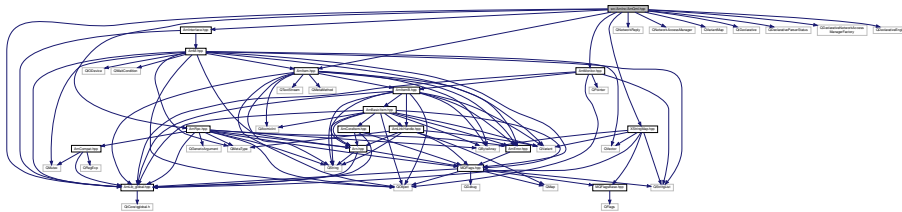
- class [ArnPipe](#)

ArnItem specialized as a pipe.

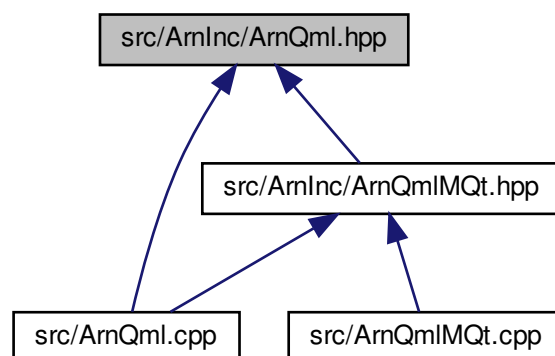
15.44 src/ArnInc/ArnQml.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnInterface.hpp"
#include "ArnItem.hpp"
#include "ArnMonitor.hpp"
#include "ArnRpc.hpp"
#include "XStringMap.hpp"
#include <QNetworkReply>
#include <QNetworkAccessManager>
#include <QVariantMap>
#include <QtDeclarative>
#include <QDeclarativeParserStatus>
#include <QDeclarativeNetworkAccessManagerFactory>
#include <QDeclarativeEngine>
```

Include dependency graph for ArnQml.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnQml](#)
ARN QML.
- struct [ArnQml::UseFlags](#)
- class [ArnItemQml](#)
ARN Item QML.
- class [ArnMonitorQml](#)
ARN Monitor QML.
- class [ArnSapiQml](#)
ARN Sapi QML.
- class [Arn::XStringMapQml](#)
- class [Arn::QmlMSys](#)

Namespaces

- [Arn](#)

Macros

- `#define QML_Qt4`
- `#define QML_QUICK_TYPE 1`
- `#define QML_ENGINE QDeclarativeEngine`
- `#define QML_PARSER_STATUS QDeclarativeParserStatus`
- `#define QML_NETACC_FACTORY QDeclarativeNetworkAccessManagerFactory`
- `#define QML_LIST_PROPERTY QDeclarativeListProperty`

15.44.1 Macro Definition Documentation

15.44.1.1 QML_ENGINE

```
#define QML_ENGINE QDeclarativeEngine
```

Definition at line 62 of file ArnQml.hpp.

15.44.1.2 QML_LIST_PROPERTY

```
#define QML_LIST_PROPERTY QDeclarativeListProperty
```

Definition at line 65 of file ArnQml.hpp.

15.44.1.3 QML_NETACC_FACTORY

```
#define QML_NETACC_FACTORY QDeclarativeNetworkAccessManagerFactory
```

Definition at line 64 of file ArnQml.hpp.

15.44.1.4 QML_PARSER_STATUS

```
#define QML_PARSER_STATUS QDeclarativeParserStatus
```

Definition at line 63 of file ArnQml.hpp.

15.44.1.5 QML_Qt4

```
#define QML_Qt4
```

Definition at line 60 of file ArnQml.hpp.

15.44.1.6 QML_QUICK_TYPE

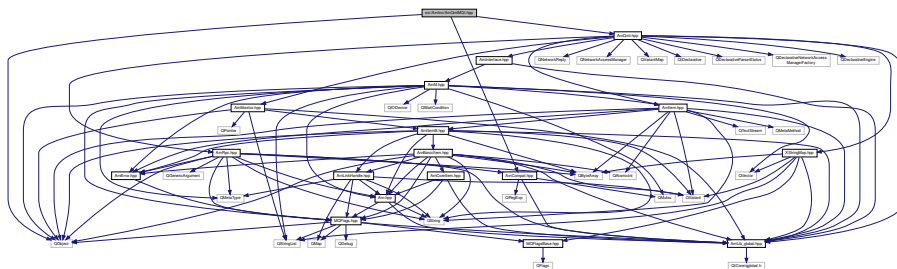
```
#define QML_QUICK_TYPE 1
```

Definition at line 61 of file ArnQml.hpp.

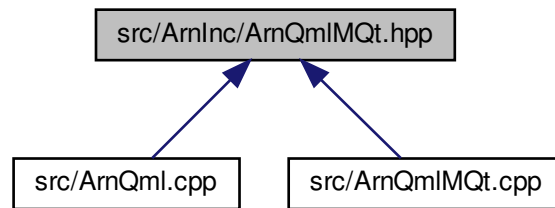
15.45 src/ArnInc/ArnQmIMQt.hpp File Reference

```
#include "ArnQml.hpp"  
#include "ArnCompat.hpp"  
#include <QObject>
```

Include dependency graph for ArnQmIMQt.hpp:



This graph shows which files directly or indirectly include this file:



Classes

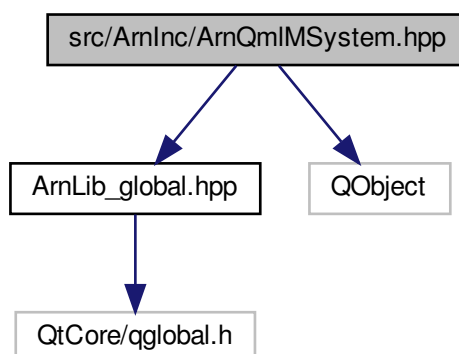
- class [Arn::QmlMQtObject](#)

Namespaces

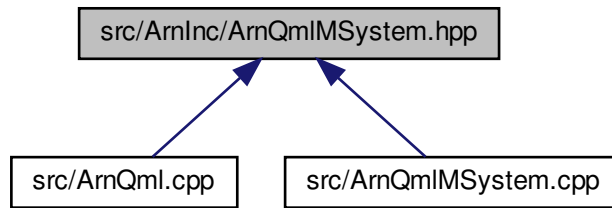
- [Arn](#)

15.46 src/ArnInc/ArnQmlMSystem.hpp File Reference

```
#include "ArnLib_global.hpp"  
#include <QObject>  
Include dependency graph for ArnQmlMSystem.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Arn::QmlMFileIO](#)

Namespaces

- [Arn](#)

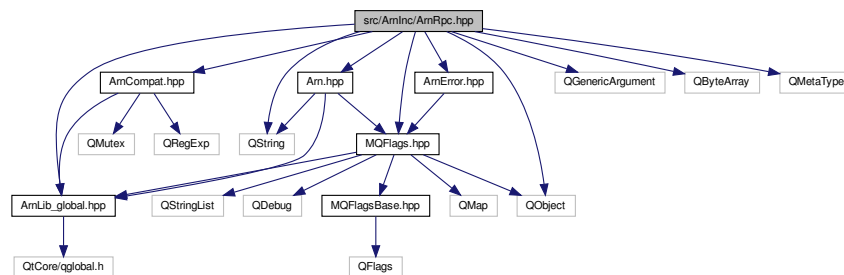
15.47 src/ArnInc/ArnRpc.hpp File Reference

```

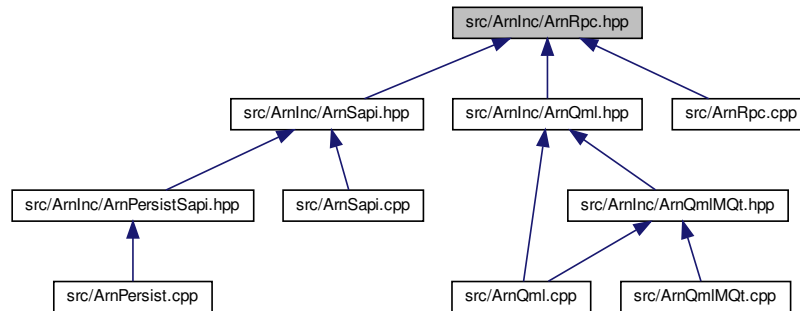
#include "ArnLib_global.hpp"
#include "Arn.hpp"
#include "ArnError.hpp"
#include "MQFlags.hpp"
#include "ArnCompat.hpp"
#include <QGenericArgument>
#include <QString>
#include <QByteArray>
#include <QObject>
#include <QMetaType>

```

Include dependency graph for ArnRpc.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [MQGenericArgument](#)
Similar to QGenericArgument but with added argument label (parameter name)
- class [MQArgument< T >](#)
Similar to QArgument but with added argument label (parameter name)
- class [ArnrpcMode](#)
- class [Arnrpc](#)
Remote Procedure Call.
- struct [Arnrpc::Invoke](#)
- struct [Arnrpc::MethodsParam::Params](#)

Macros

- `#define no_queue`
- `#define MQ_ARG(type, label, data) MQArgument<type>(#type, #label, data)`
Similar to Q_ARG but with added argument label (parameter name)

15.47.1 Macro Definition Documentation

15.47.1.1 MQ_ARG

```
#define MQ_ARG(
    type,
    label,
    data ) MQArgument<type>(#type, #label, data)
```

Similar to `Q_ARG` but with added argument label (parameter name)

Definition at line 49 of file `Arnrpc.hpp`.

15.47.1.2 no_queue

```
#define no_queue
```

Examples:

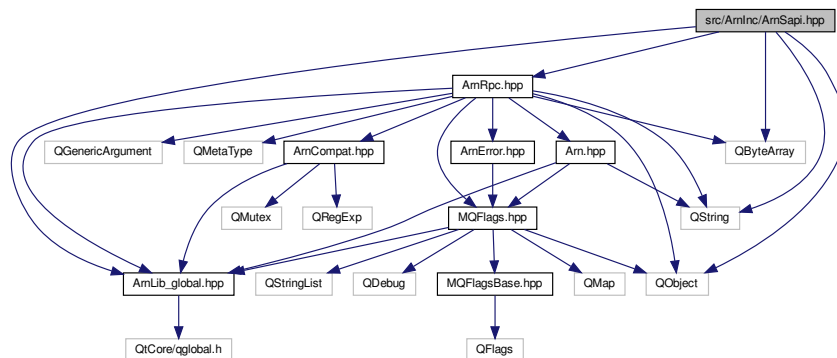
[ArnDemoChatServer/ChatSapi.hpp](#).

Definition at line 35 of file ArnRpc.hpp.

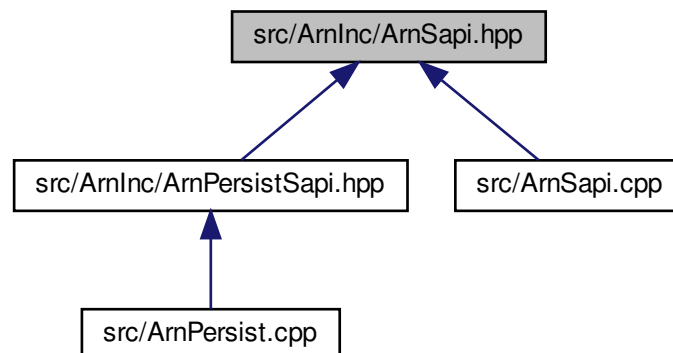
15.48 src/ArnInc/ArnSapi.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnRpc.hpp"
#include <QString>
#include <QByteArray>
#include <QObject>
```

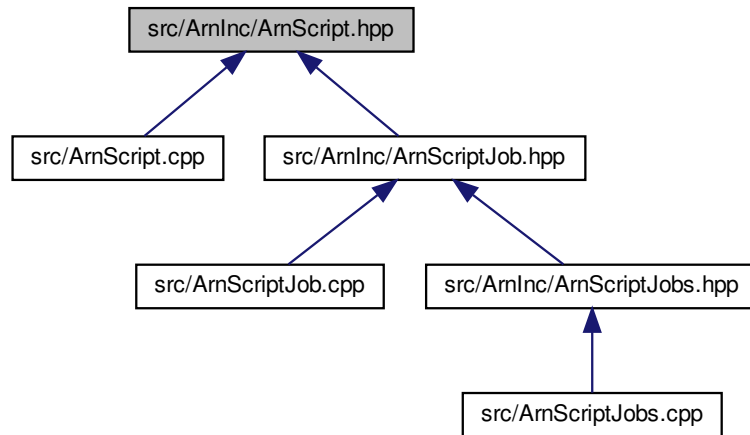
Include dependency graph for ArnSapi.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnScript](#)

Macros

- #define [ARN_JSENGINE](#) QScriptEngine
- #define [ARN_JSVALUE](#) QScriptValue
- #define [ARN_JSVALUE_LIST](#) QScriptValueList
- #define [ARN_JSCONTEXT](#) QScriptContext

15.49.1 Macro Definition Documentation

15.49.1.1 ARN_JSCONTEXT

```
#define ARN_JSCONTEXT QScriptContext
```

Definition at line 57 of file ArnScript.hpp.

15.49.1.2 ARN_JSENGINE

```
#define ARN_JSENGINE QScriptEngine
```

Definition at line 54 of file ArnScript.hpp.

15.49.1.3 ARN_JSVALUE

```
#define ARN_JSVALUE QScriptValue
```

Definition at line 55 of file ArnScript.hpp.

15.49.1.4 ARN_JSVALUE_LIST

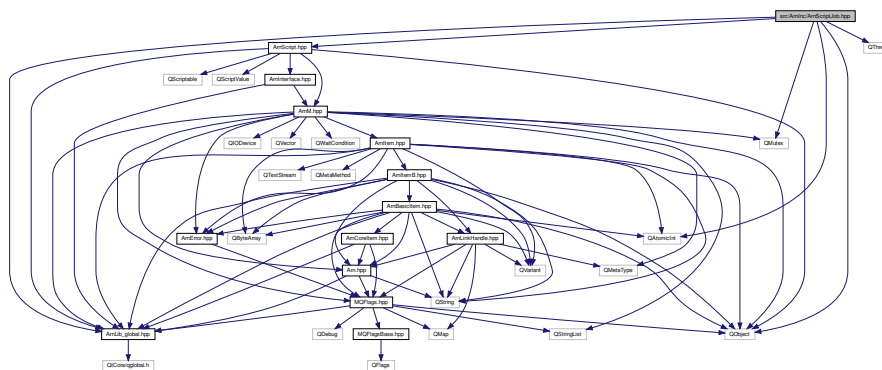
```
#define ARN_JSVALUE_LIST QScriptValueList
```

Definition at line 56 of file ArnScript.hpp.

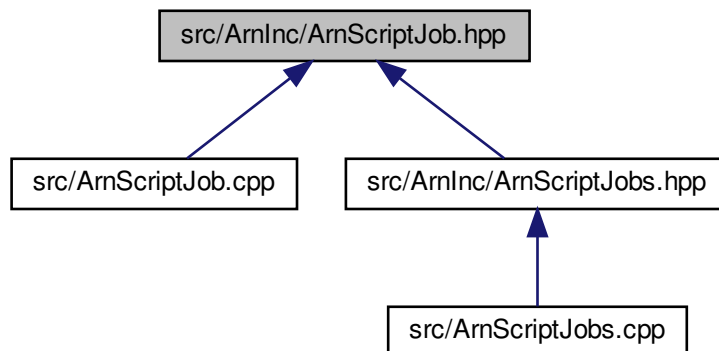
15.50 src/ArnInc/ArnScriptJob.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnScript.hpp"
#include <QObject>
#include <QAtomicInt>
#include <QMutex>
#include <QThread>
```

Include dependency graph for ArnScriptJob.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnScriptJob](#)
Interface class to be normally used, is also Script Job interface.
- class [ArnScriptJobFactory](#)
Must be thread-safe as subclassed.
- class [ArnScriptJobControl](#)
Is thread-safe (except doSetupJob)

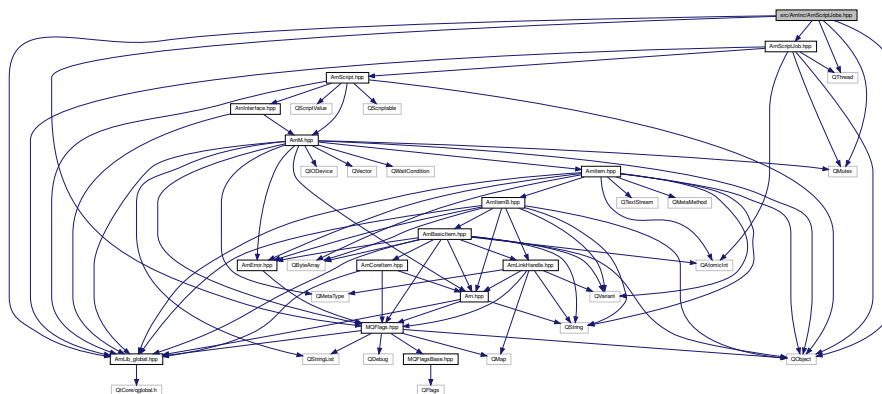
15.51 src/ArnInc/ArnScriptJobs.hpp File Reference

```

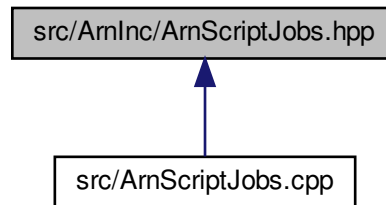
#include "ArnLib_global.hpp"
#include "ArnScriptJob.hpp"
#include "MQFlags.hpp"
#include <QThread>
#include <QMutex>
#include <QObject>

```

Include dependency graph for ArnScriptJobs.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnScriptJobs](#)
- struct [ArnScriptJobs::Type](#)

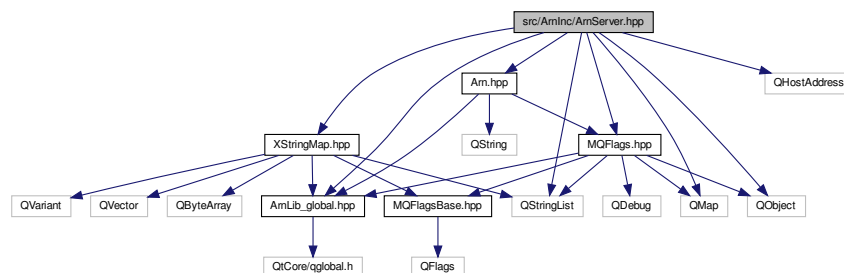
15.52 src/ArnInc/ArnServer.hpp File Reference

```

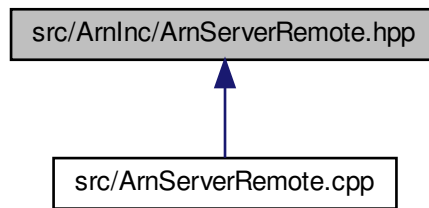
#include "ArnLib_global.hpp"
#include "Arn.hpp"
#include "MQFlags.hpp"
#include "XStringMap.hpp"
#include <QObject>
#include <QHostAddress>
#include <QMap>
#include <QStringList>

```

Include dependency graph for ArnServer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnServerRemoteSessionKillMode](#)
- class [ArnServerRemoteSession](#)
- class [ArnServerRemote](#)

Class for remote controlling an [Arn](#) Server.

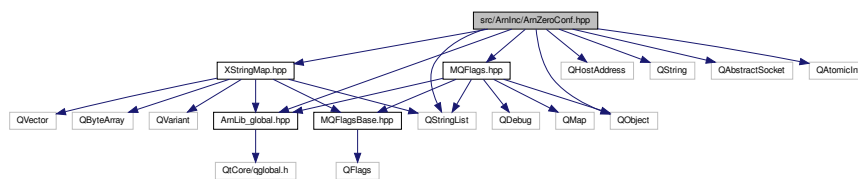
15.54 src/ArnInc/ArnZeroConf.hpp File Reference

```

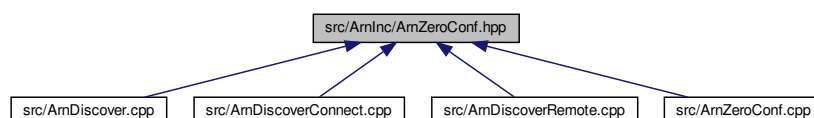
#include "ArnLib_global.hpp"
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QHostAddress>
#include <QObject>
#include <QStringList>
#include <QString>
#include <QAbstractSocket>
#include <QAtomicInt>

```

Include dependency graph for ArnZeroConf.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct [ArnZeroConf::Error](#)
Errors of ZeroConfig, other values are defined in dns_sd.h.
- struct [ArnZeroConf::State](#)
States of ZeroConfig, limited valid for each [ArnZeroConfB](#) subclass / These values must be synced with: [ArnDiscover::State](#).
- class [ArnZeroConfB](#)
Base class for Zero Config.
- class [ArnZeroConfRegister](#)
Registering a ZeroConfig service.
- class [ArnZeroConfResolve](#)
Resolv a ZeroConfig service.
- class [ArnZeroConfLookup](#)
Lookup a host.
- class [ArnZeroConfBrowser](#)
Browsing for ZeroConfig services.

Namespaces

- [ArnZeroConf](#)

Typedefs

- typedef struct _DNSServiceRef_t * [DNSServiceRef](#)

15.54.1 Typedef Documentation

15.54.1.1 DNSServiceRef

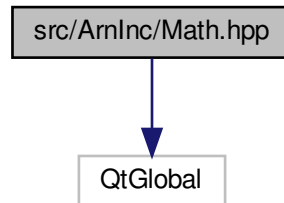
```
typedef struct _DNSServiceRef_t* DNSServiceRef
```

Definition at line 45 of file ArnZeroConf.hpp.

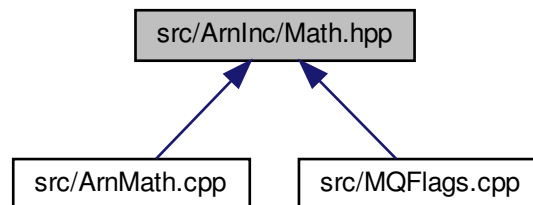
15.55 src/ArnInc/Math.hpp File Reference

```
#include <QtGlobal>
```

Include dependency graph for Math.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- [Arn](#)

Functions

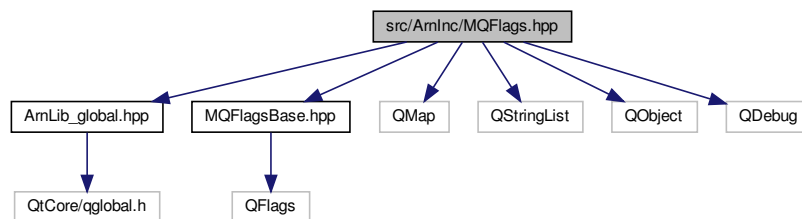
- `int Arn::_mod_i` (int x, int y)
- `qulonglong Arn::_mod_ll` (qulonglong x, qulonglong y)
- `int Arn::_log2_u` (uint x)
- `int Arn::_log2_ull` (qulonglong x)
- `template<typename T >`
`T Arn::mod` (T x, T y)
- `template<typename T >`
`T Arn::circVal` (T x, T lo, T hi)
- `template<typename T >`
`bool Arn::isPower2` (T x)

- `template<typename T >`
`int Arn::log2 (T x)`
- `template<typename T >`
`T Arn::minLim (const T &x, const T &lim)`
- `template<typename T >`
`T Arn::maxLim (const T &x, const T &lim)`
- `template<typename T >`
`T Arn::rangeLim (const T &x, const T &min, const T &max)`

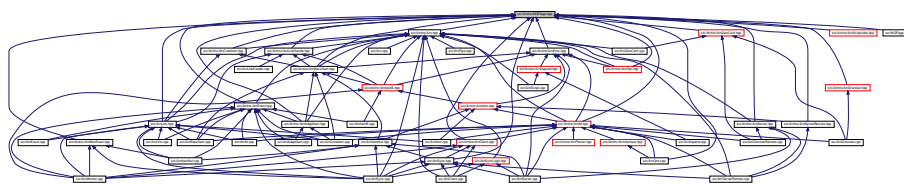
15.56 src/ArnInc/MQFlags.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "MQFlagsBase.hpp"
#include <QMap>
#include <QStringList>
#include <QObject>
#include <QDebug>
```

Include dependency graph for MQFlags.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct `Arn::_InitEnumTxt`
- struct `Arn::_InitSubEnum`
- class `Arn::EnumTxt`
Class Enum text.
- struct `Arn::EnumTxt::IncludeMode`

Namespaces

- `Arn`

Macros

- #define `MQ_NSTXT_FILL_MISSING` 0, 0
- #define `MQ_NSTXT_FILL_MISSING_FROM`(FromNs) FromNs, 0
- #define `MQ_DECLARE_FLAGSTXT`(EStruct)
 - Flags text.
- #define `MQ_DECLARE_FLAGS_NSTXT`(...)
- #define `MQ_DECLARE_SUBETXT`(...)
- #define `MQ_SUBETXT_ADD_RELDEF`(EStruct, Mask, Factor) { &EStruct::txt(), Mask, Factor}
- #define `MQ_SUBETXT_ADD_ABSDEF`(EStruct, Mask) `MQ_SUBETXT_ADD_RELDEF`(EStruct, Mask, 1)
- #define `MQ_SUBETXT_ADD_RELOP`(EStruct, Mask, Factor)
- #define `MQ_SUBETXT_ADD_ABSOP`(EStruct, Mask) `MQ_SUBETXT_ADD_RELOP`(EStruct, Mask, 1) \
- #define `MQ_DECLARE_ENUMTXT`(EStruct)
 - Enums text.
- #define `MQ_DECLARE_ENUM_NSTXT`(...)

15.56.1 Macro Definition Documentation

15.56.1.1 MQ_DECLARE_ENUM_NSTXT

```
#define MQ_DECLARE_ENUM_NSTXT(
    ... )
```

Value:

```
static const Arn::_InitEnumTxt* _setNs(int dummy) { \
    Q_UNUSED(dummy) \
    static Arn::_InitEnumTxt initTxt[] = { __VA_ARGS__ , { 0, 0, 0 } }; \
    return initTxt; \
};
```

Definition at line 113 of file MQFlags.hpp.

15.56.1.2 MQ_DECLARE_ENUMTXT

```
#define MQ_DECLARE_ENUMTXT(
    EStruct )
```

Value:

```
MQ_DECLARE_ENUM( EStruct) \
    static Arn::EnumTxt& txt() {static Arn::EnumTxt in( &staticMetaObject, false, \
        _setNs(0), arnNullptr, \
            #EStruct); return in;} \
    inline static const Arn::_InitEnumTxt* _setNs( const \
        Arn::_InitEnumTxt* ieTxt) {return ieTxt;} \
    inline static const char* name() {return txt().name();} \
    inline QString toString( quint16 nameSpace = 0) const {return txt().getTxtString( e, nameSpace);} \
    inline static EStruct fromString( const QString& text, quint16 nameSpace = 0) \
        {return EStruct( E( txt().getEnumVal( text, 0, nameSpace)));}
```

Enums text.

Definition at line 102 of file MQFlags.hpp.

15.56.1.3 MQ_DECLARE_FLAGS_NSTXT

```
#define MQ_DECLARE_FLAGS_NSTXT(
    ... )
```

Value:

```
static const Arn::_InitEnumTxt* _setNs(int dummy) { \
    Q_UNUSED(dummy) \
    static Arn::_InitEnumTxt initTxt[] = { __VA_ARGS__ , { 0, 0, arnNullptr } }; \
    return initTxt; \
};
```

Definition at line 69 of file MQFlags.hpp.

15.56.1.4 MQ_DECLARE_FLAGSTXT

```
#define MQ_DECLARE_FLAGSTXT(
    FEStruct )
```

Value:

```
MQ_DECLARE_FLAGS( FEStruct) \
    static Arn::EnumTxt& txt() {static Arn::EnumTxt in( &staticMetaObject, true,
    _setNs(0), _setSe(0), \
    #FEStruct); return in;} \
    inline static const Arn::_InitEnumTxt* _setNs( const
    Arn::_InitEnumTxt* ieTxt) {return ieTxt;} \
    inline static const Arn::_InitSubEnum* _setSe( const
    Arn::_InitSubEnum* isEnum) {return isEnum;} \
    inline static const char* name() {return txt().name();} \
    inline QString toString( quint16 nameSpace = 0) const {return txt().flagsToString( f, nameSpace);} \
    inline static FEStruct fromString( const QString& text, quint16 nameSpace = 0) \
    {return FEStruct( F( txt().flagsFromString( text, nameSpace)));}
```

Flags text.

Definition at line 57 of file MQFlags.hpp.

15.56.1.5 MQ_DECLARE_SUBETXT

```
#define MQ_DECLARE_SUBETXT(
    ... )
```

Value:

```
static const Arn::_InitSubEnum* _setSe( int dummy) { \
    Q_UNUSED(dummy) \
    static Arn::_InitSubEnum initSubEnum[] = { __VA_ARGS__ , { arnNullptr, 0, 0 } }; \
    return initSubEnum; \
};
```

Definition at line 76 of file MQFlags.hpp.

15.56.1.6 MQ_NSTXT_FILL_MISSING

```
#define MQ_NSTXT_FILL_MISSING 0, 0
```

Definition at line 52 of file MQFlags.hpp.

15.56.1.7 MQ_NSTXT_FILL_MISSING_FROM

```
#define MQ_NSTXT_FILL_MISSING_FROM(  
    FromNs ) FromNs, 0
```

Definition at line 53 of file MQFlags.hpp.

15.56.1.8 MQ_SUBETXT_ADD_ABSDEF

```
#define MQ_SUBETXT_ADD_ABSDEF(  
    EStruct,  
    Mask ) MQ_SUBETXT_ADD_RELDEF( EStruct, Mask, 1)
```

Definition at line 86 of file MQFlags.hpp.

15.56.1.9 MQ_SUBETXT_ADD_ABSOP

```
#define MQ_SUBETXT_ADD_ABSOP(  
    EStruct,  
    Mask ) MQ_SUBETXT_ADD_RELOP( EStruct, Mask, 1) \
```

Definition at line 97 of file MQFlags.hpp.

15.56.1.10 MQ_SUBETXT_ADD_RELDEF

```
#define MQ_SUBETXT_ADD_RELDEF(  
    EStruct,  
    Mask,  
    Factor ) { &EStruct::txt(), Mask, Factor}
```

Definition at line 83 of file MQFlags.hpp.

15.56.1.11 MQ_SUBETXT_ADD_RELOP

```
#define MQ_SUBETXT_ADD_RELOP(
    EStruct,
    Mask,
    Factor )
```

Value:

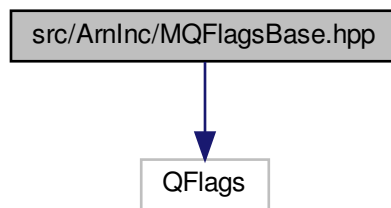
```
inline void setSubEnum( EStruct::E v_) { \
    setBits( Mask, v_ * Factor); \
} \
inline EStruct::E getSubEnum_##EStruct() { \
    return EStruct::E( (f & Mask) / Factor); \
}
```

Definition at line 89 of file MQFlags.hpp.

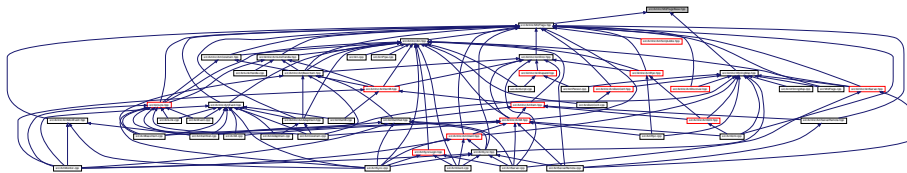
15.57 src/ArnInc/MQFlagsBase.hpp File Reference

```
#include <QFlags>
```

Include dependency graph for MQFlagsBase.hpp:



This graph shows which files directly or indirectly include this file:

**Macros**

- #define [MQ_DECLARE_FLAGS](#)(FEStruct)
Flags.
- #define [MQ_DECLARE_OPERATORS_FOR_FLAGS](#)(FEStruct) Q_DECLARE_OPERATORS_FOR_FLAGS(FEStruct::F)
- #define [MQ_DECLARE_ENUM](#)(EStruct)
Enums.

15.57.1 Macro Definition Documentation

15.57.1.1 MQ_DECLARE_ENUM

```
#define MQ_DECLARE_ENUM(  
    EStruct )
```

Value:

```
E e; \  
    inline EStruct(E v_ = E(0)) : e( v_ ) {setup(0);} \  
    inline static EStruct fromInt( int v_ ) {return EStruct( E( v_));} \  
    inline int toInt() const {return e;} \  
    inline operator int() const {return e;} \  
    inline bool operator!() const {return !e;} \  
    inline void setup( char* dummy ) {Q_UNUSED(dummy)}
```

Enums.

Definition at line 70 of file MQFlagsBase.hpp.

15.57.1.2 MQ_DECLARE_FLAGS

```
#define MQ_DECLARE_FLAGS(  
    FEStruct )
```

Value:

```
Q_DECLARE_FLAGS(F, E) \  
    F f; \  
    inline FEStruct(F v_ = F(QFlag(0))) : f( v_ ) {setup(0);} \  
    inline FEStruct(E e_) : f( e_ ) {setup(0);} \  
    inline static E flagIf( bool test, E e ) {return test ? e : E(0);} \  
    inline bool is(E e) const {return f.testFlag(e);} \  
    inline bool isAny(E e) const {return ((f & e) != 0) && (e != 0 || f == 0);} \  
    inline FEStruct& set(E e, bool v_ = true) {f = v_ ? (f | e) : (f & ~e); return *this;} \  
    inline void setBits(E e, int v_) {f = (f & ~e) | E(v_);} \  
    inline static FEStruct fromInt( int v_ ) {return FEStruct( F( v_));} \  
    inline int toInt() const {return f;} \  
    inline operator int() const {return f;} \  
    inline bool operator!() const {return !f;} \  
    inline void setup( char* dummy ) {Q_UNUSED(dummy)}
```

Flags.

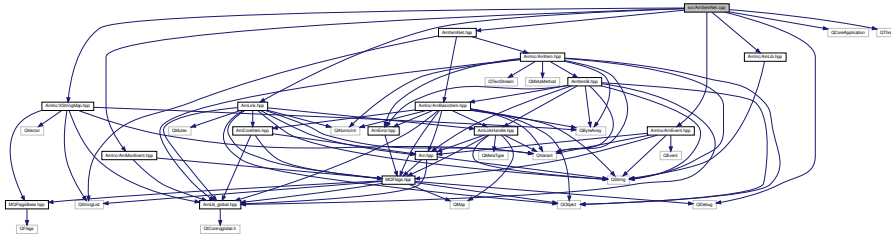
Definition at line 49 of file MQFlagsBase.hpp.

15.57.1.3 MQ_DECLARE_OPERATORS_FOR_FLAGS

```
#define MQ_DECLARE_OPERATORS_FOR_FLAGS(  
    FEStruct ) Q_DECLARE_OPERATORS_FOR_FLAGS( FEStruct::F)
```

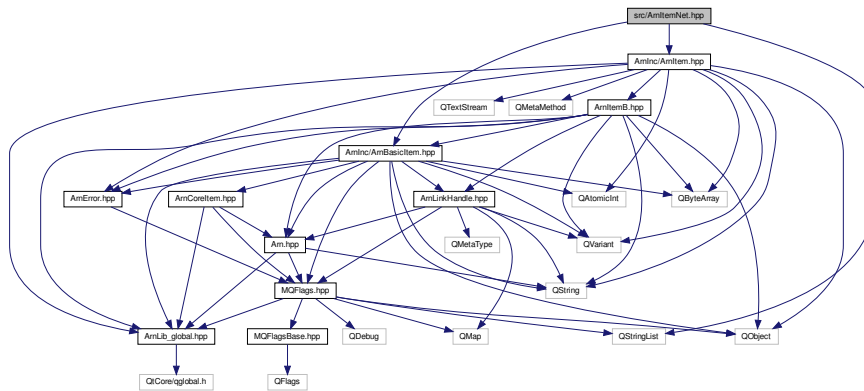
Definition at line 65 of file MQFlagsBase.hpp.


```
#include <QDebug>
Include dependency graph for ArnItemNet.cpp:
```

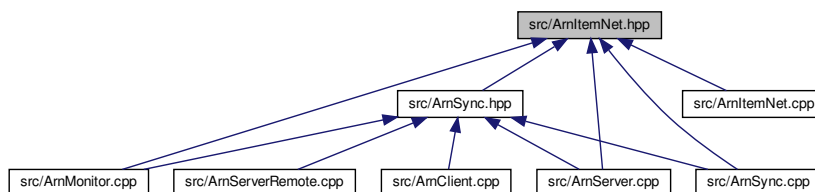


15.62 src/ArnItemNet.hpp File Reference

```
#include "ArnInc/ArnBasicItem.hpp"
#include "ArnInc/ArnItem.hpp"
#include <QStringList>
Include dependency graph for ArnItemNet.hpp:
```



This graph shows which files directly or indirectly include this file:



15.63 src/ArnItemValve.cpp File Reference

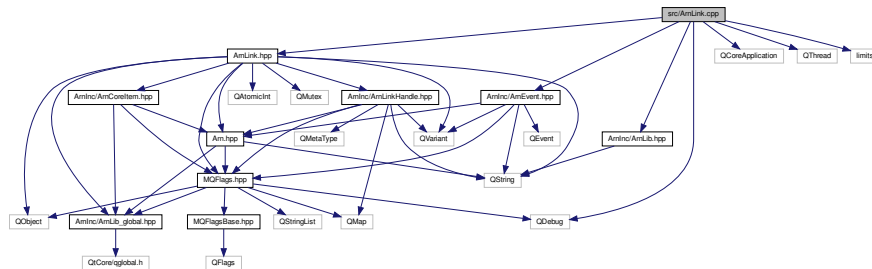
```
#include "ArnInc/ArnItemValve.hpp"
```


- bool `Arn::debugShareObj` = false
- bool `Arn::debugMonitor` = false
- bool `Arn::debugMonitorTest` = false
- bool `Arn::debugRPC` = false
- bool `Arn::debugDepend` = false
- bool `Arn::debugQmlNetwork` = false
- bool `Arn::debugDiscover` = false
- bool `Arn::debugZeroConf` = false
- bool `Arn::debugMDNS` = false
- bool `Arn::warningMDNS` = false
- bool `Arn::offHeartbeat` = false
- const QString `Arn::resourceArnLib` = ":/ArnLib/"
- const QString `Arn::resourceArnRoot` = ":/ArnLib/ArnRoot/"

15.65 src/ArnLink.cpp File Reference

```
#include "ArnLink.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnInc/ArnEvent.hpp"
#include <QCoreApplication>
#include <QThread>
#include <limits>
#include <QDebug>
```

Include dependency graph for ArnLink.cpp:



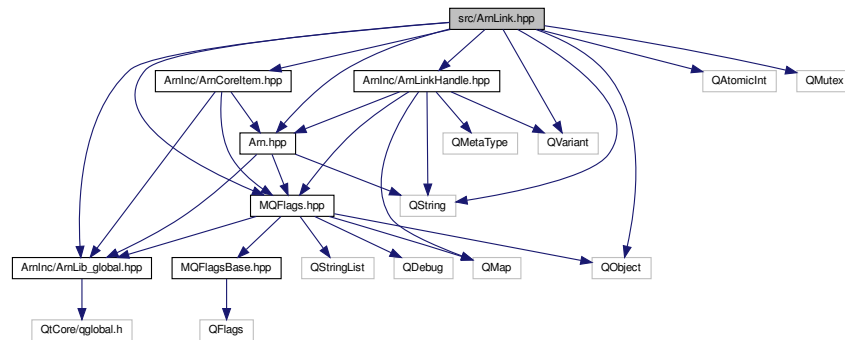
Classes

- struct `ArnLinkValue`

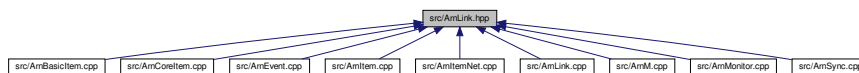
15.66 src/ArnLink.hpp File Reference

```
#include "ArnInc/ArnLib_global.hpp"
#include "ArnInc/ArnLinkHandle.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnInc/ArnCoreItem.hpp"
#include "ArnInc/MQFlags.hpp"
#include <QObject>
#include <QString>
#include <QVariant>
```

```
#include <QAtomicInt>
#include <QMutex>
Include dependency graph for ArnLink.hpp:
```



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef QList< ArnLink * > [ArnLinkList](#)
- typedef QList< [ArnCoreItem](#) * > [ArnCoreItemList](#)

15.66.1 Typedef Documentation

15.66.1.1 ArnCoreItemList

```
typedef QList<ArnCoreItem*> ArnCoreItemList
```

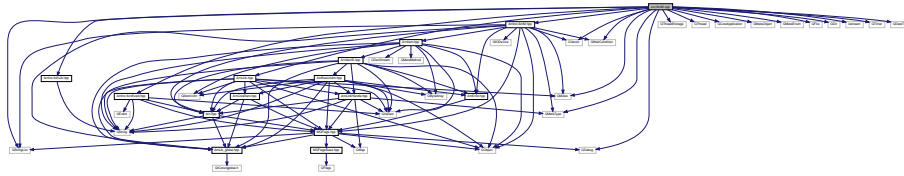
Definition at line 51 of file ArnLink.hpp.

15.66.1.2 ArnLinkList

```
typedef QList<ArnLink*> ArnLinkList
```

Definition at line 48 of file ArnLink.hpp.

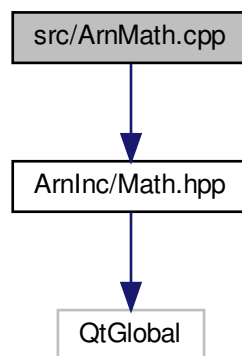
Include dependency graph for ArnM.cpp:



15.69 src/ArnMath.cpp File Reference

```
#include "ArnInc/Math.hpp"
```

Include dependency graph for ArnMath.cpp:



Namespaces

- [Arn](#)

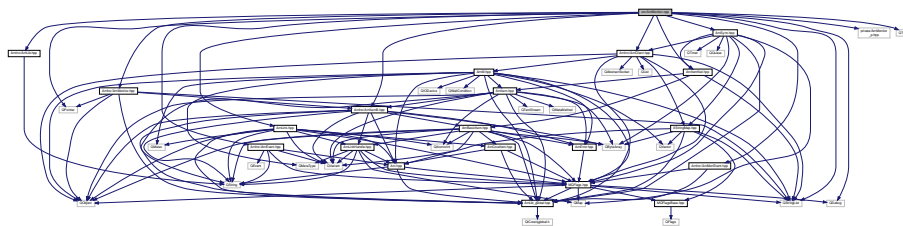
Functions

- [int Arn::_mod_i](#) (int x, int y)
- [qulonglong Arn::_mod_ll](#) (qulonglong x, qulonglong y)
- [int Arn::_log2_u](#) (uint x)
- [int Arn::_log2_ull](#) (qulonglong x)

15.70 src/ArnMonitor.cpp File Reference

```
#include "ArnInc/ArnMonitor.hpp"
#include "private/ArnMonitor_p.hpp"
#include "ArnInc/ArnMonEvent.hpp"
#include "ArnInc/ArnClient.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnInc/ArnEvent.hpp"
#include "ArnInc/ArnItemB.hpp"
#include "ArnItemNet.hpp"
#include "ArnSync.hpp"
#include "ArnLink.hpp"
#include <QDebug>
#include <QTime>
```

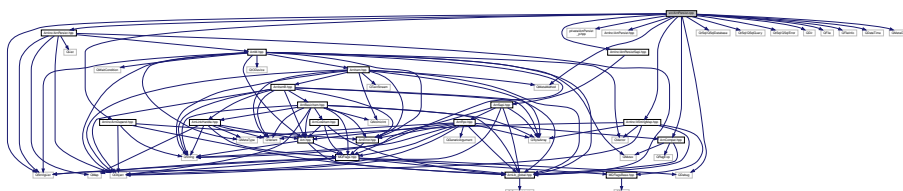
Include dependency graph for ArnMonitor.cpp:



15.71 src/ArnPersist.cpp File Reference

```
#include "ArnInc/ArnPersist.hpp"
#include "private/ArnPersist_p.hpp"
#include "ArnInc/ArnPersistSapi.hpp"
#include "ArnInc/ArnDepend.hpp"
#include "ArnInc/XStringMap.hpp"
#include "ArnInc/ArnCompat.hpp"
#include <QtSql/QtSqlDatabase>
#include <QtSql/QtSqlQuery>
#include <QtSql/QtSqlError>
#include <QDir>
#include <QFile>
#include <QFileInfo>
#include <QDateTime>
#include <QStringList>
#include <QDebug>
#include <QMetaObject>
#include <QMetaMethod>
```

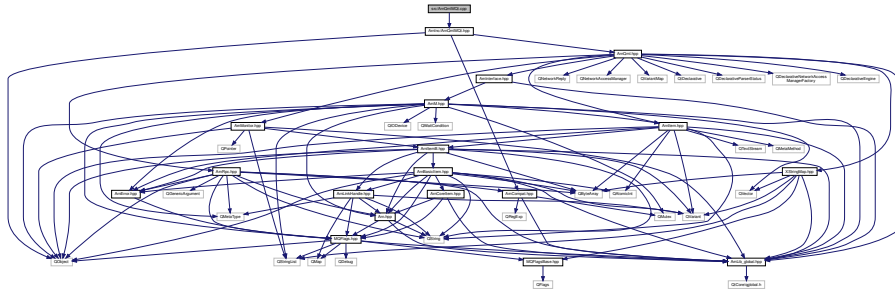
Include dependency graph for ArnPersist.cpp:



15.74 src/ArnQmlMQt.cpp File Reference

```
#include "ArnInc/ArnQmlMQt.hpp"
```

Include dependency graph for ArnQmlMQt.cpp:



Namespaces

- [Arn](#)

15.75 src/ArnQmlMSystem.cpp File Reference

```
#include "ArnInc/ArnQmlMSystem.hpp"
```

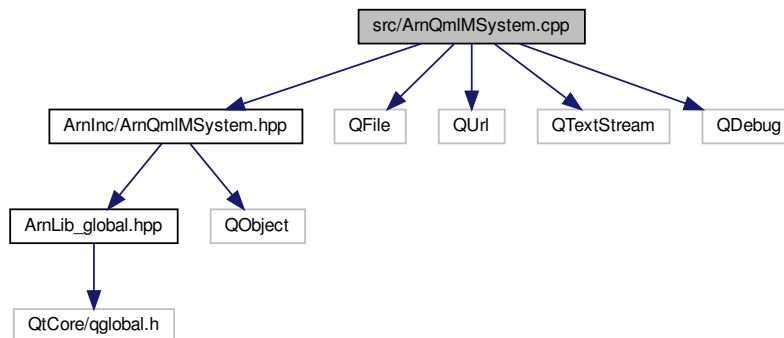
```
#include <QFile>
```

```
#include <QUrl>
```

```
#include <QTextStream>
```

```
#include <QDebug>
```

Include dependency graph for ArnQmlMSystem.cpp:



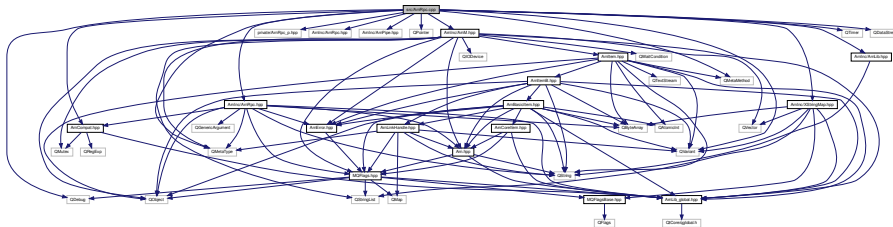
Namespaces

- [Arn](#)

15.76 src/ArnRpc.cpp File Reference

```
#include "ArnInc/ArnRpc.hpp"
#include "private/ArnRpc_p.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnInc/XStringMap.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnInc/ArnCompat.hpp"
#include <QMetaType>
#include <QMetaMethod>
#include <QTimer>
#include <QDataStream>
#include <QVariant>
#include <QDebug>
```

Include dependency graph for ArnRpc.cpp:



Macros

- `#define RPC_STORAGE_NAME "_ArnRpcStorage"`

15.76.1 Macro Definition Documentation

15.76.1.1 `RPC_STORAGE_NAME`

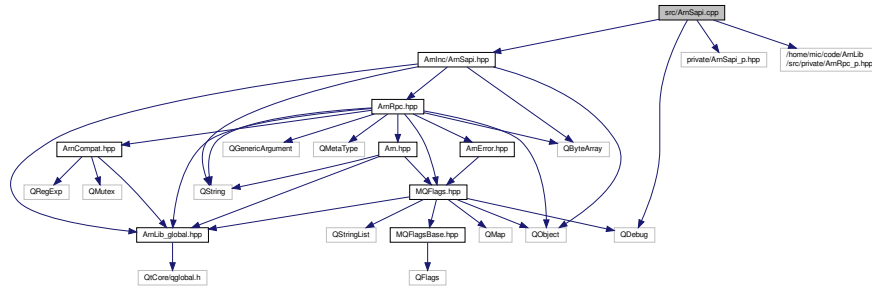
```
#define RPC_STORAGE_NAME "_ArnRpcStorage"
```

Definition at line 47 of file ArnRpc.cpp.

15.77 src/ArnSapi.cpp File Reference

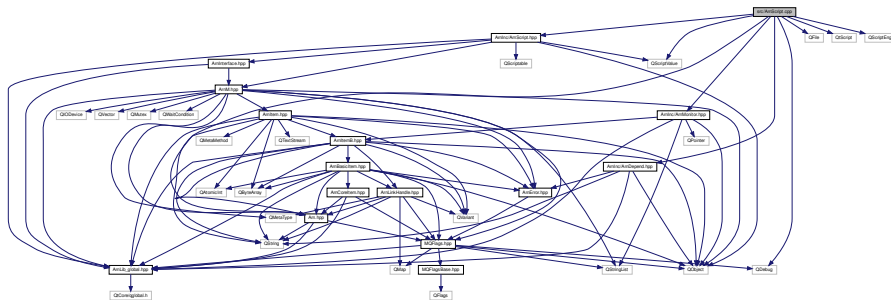
```
#include "ArnInc/ArnSapi.hpp"
#include "private/ArnSapi_p.hpp"
```

```
#include <QDebug>
Include dependency graph for ArnSapi.cpp:
```



15.78 src/ArnScript.cpp File Reference

```
#include "ArnInc/ArnScript.hpp"
#include "ArnInc/ArnDepend.hpp"
#include "ArnInc/ArnMonitor.hpp"
#include "ArnInc/Arn.hpp"
#include <QFile>
#include <QDebug>
#include <QtScript>
#include <QScriptValue>
#include <QScriptEngine>
Include dependency graph for ArnScript.cpp:
```



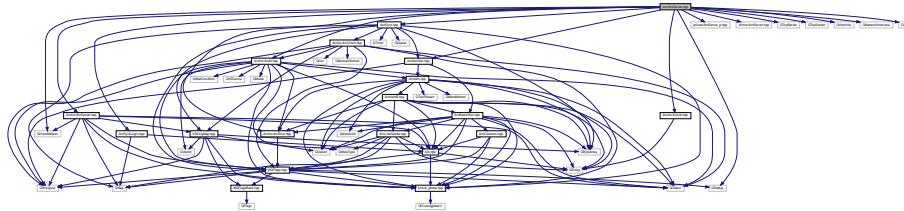
15.79 src/ArnScriptJob.cpp File Reference

```
#include "ArnInc/ArnScriptJob.hpp"
#include <QFileInfo>
#include <QTimer>
#include <QEvent>
#include <QCoreApplication>
#include <QDebug>
#include <QScriptable>
#include <QtScript>
```


15.81 src/ArnServer.cpp File Reference

```
#include "ArnInc/ArnServer.hpp"
#include "private/ArnServer_p.hpp"
#include "ArnInc/ArnError.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnSync.hpp"
#include "ArnSyncLogin.hpp"
#include "ArnItemNet.hpp"
#include <QTcpServer>
#include <QTcpSocket>
#include <QHostInfo>
#include <QNetworkInterface>
#include <QHostAddress>
#include <QPair>
#include <QDebug>
```

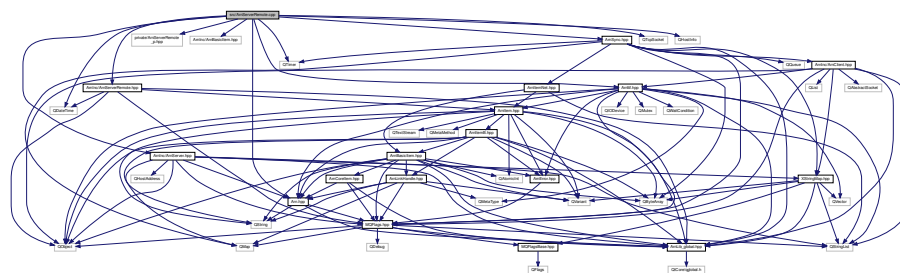
Include dependency graph for ArnServer.cpp:



15.82 src/ArnServerRemote.cpp File Reference

```
#include "ArnInc/ArnServerRemote.hpp"
#include "private/ArnServerRemote_p.hpp"
#include "ArnInc/ArnServer.hpp"
#include "ArnSync.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnInc/Arn.hpp"
#include <QTcpSocket>
#include <QHostInfo>
```

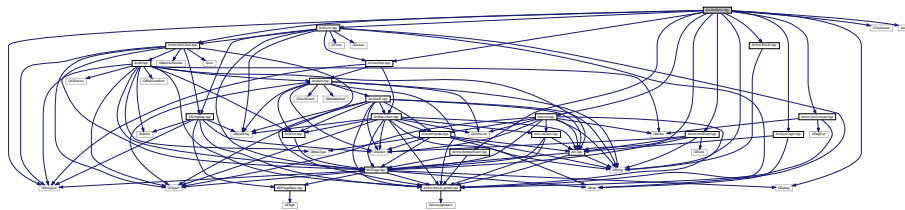
Include dependency graph for ArnServerRemote.cpp:



15.83 src/ArnSync.cpp File Reference

```
#include "ArnSync.hpp"
#include "ArnSyncLogin.hpp"
#include "ArnItemNet.hpp"
#include "ArnLink.hpp"
#include "ArnInc/ArnClient.hpp"
#include "ArnInc/ArnMonEvent.hpp"
#include "ArnInc/ArnEvent.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnInc/ArnCompat.hpp"
#include <QTcpSocket>
#include <QString>
#include <QStringList>
#include <QDebug>
#include <limits.h>
```

Include dependency graph for ArnSync.cpp:



Macros

- `#define ARNSYNCVER "4.0"`

15.83.1 Macro Definition Documentation

15.83.1.1 ARNSYNCVER

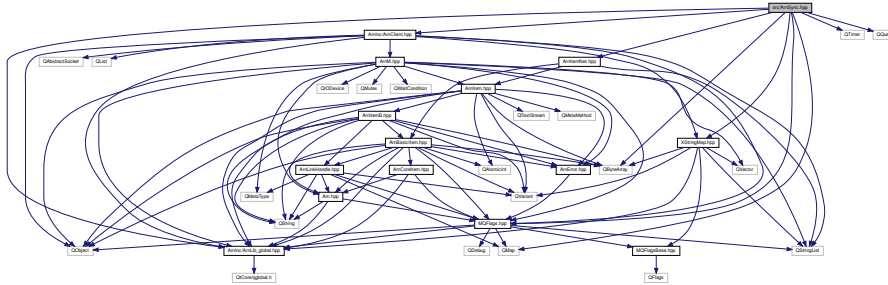
```
#define ARNSYNCVER "4.0"
```

Definition at line 48 of file ArnSync.cpp.

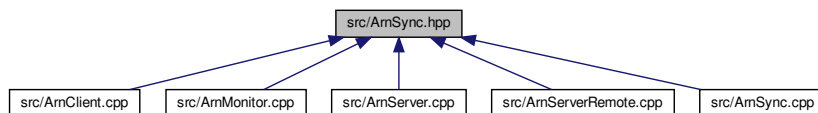
15.84 src/ArnSync.hpp File Reference

```
#include "ArnInc/ArnLib_global.hpp"
#include "ArnInc/ArnClient.hpp"
#include "ArnInc/XStringMap.hpp"
#include "ArnItemNet.hpp"
#include "ArnInc/MQFlags.hpp"
#include <QTimer>
```

```
#include <QByteArray>
#include <QMap>
#include <QQueue>
Include dependency graph for ArnSync.hpp:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define ARNRECNAME ""`

15.84.1 Macro Definition Documentation

15.84.1.1 ARNRECNAME

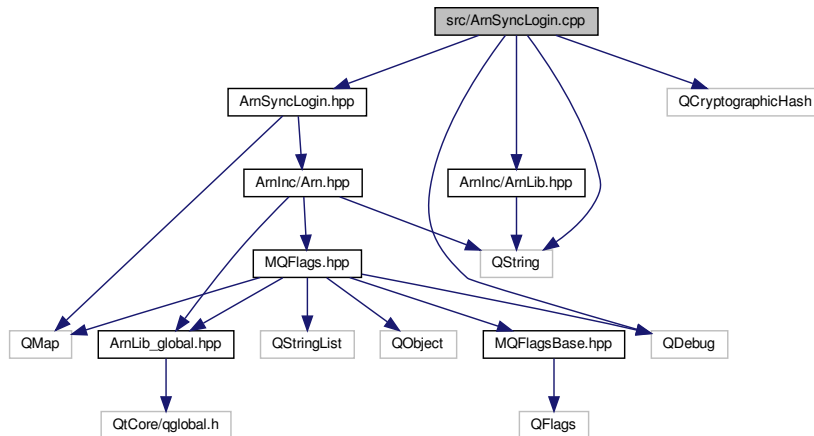
```
#define ARNRECNAME ""
```

Definition at line 45 of file ArnSync.hpp.

15.85 src/ArnSyncLogin.cpp File Reference

```
#include "ArnSyncLogin.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QCryptographicHash>
#include <QString>
#include <QDebug>
```

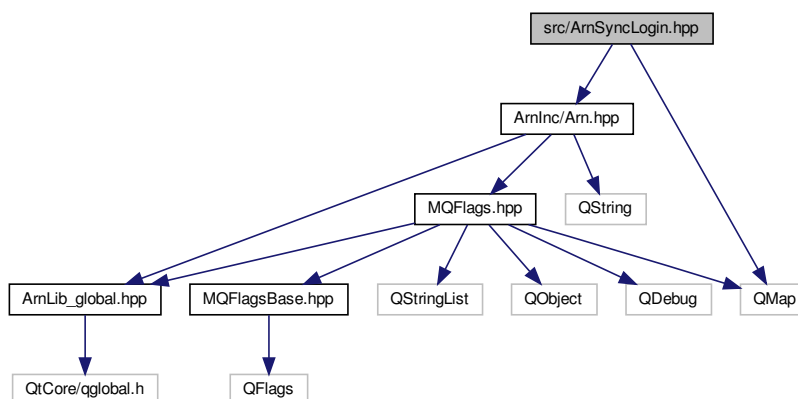
Include dependency graph for ArnSyncLogin.cpp:



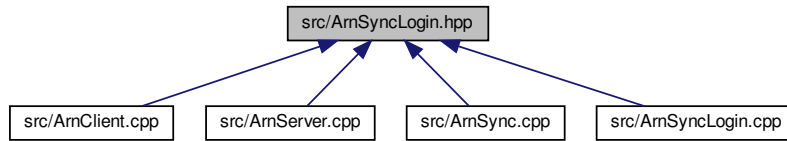
15.86 src/ArnSyncLogin.hpp File Reference

```
#include "ArnInc/Arn.hpp"
#include <QMap>
```

Include dependency graph for ArnSyncLogin.hpp:



This graph shows which files directly or indirectly include this file:



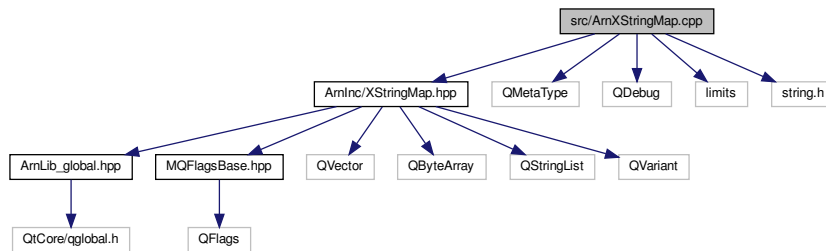
15.87 src/ArnXStringMap.cpp File Reference

```

#include "ArnInc/XStringMap.hpp"
#include <QMetaType>
#include <QDebug>
#include <limits>
#include <string.h>

```

Include dependency graph for ArnXStringMap.cpp:



Namespaces

- [Arn](#)

15.88 src/ArnZeroConf.cpp File Reference

```

#include "ArnInc/ArnZeroConf.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnInc/ArnCompat.hpp"
#include <dns_sd.h>
#include <QSocketNotifier>
#include <QHostInfo>
#include <QTimer>
#include <QtEndian>

```


Chapter 16

Example Documentation

16.1 ArnDemoChat/main.cpp

Demo Chat Client

```
#include "MainWindow.hpp"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

16.2 ArnDemoChat/MainWindow.cpp

Demo Chat Client

```
// Copyright (C) 2010-2014 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklund.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their own
// licenses. ArnDemoChat is independent of these licenses; however, use of these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#include "MainWindow.hpp"
```

```

#include "tmp/ui_MainWindow.h"
#include <ArnInc/ArnDiscoverRemote.hpp>

MainWindow::MainWindow( QWidget* parent) :
    QMainWindow( parent),
    _ui( new Ui::MainWindow)
{
    _ui->setupUi( this);
    _ui->userEdit->setFocus();
    connect( _ui->lineEdit, SIGNAL(returnPressed()), this, SLOT(doSendLine()));

    _arnClient.addMountPoint("/");
    _arnClient.setAutoConnect(true);

    ArnDiscoverConnector* connector = new
        ArnDiscoverConnector( _arnClient, "DemoChat");
    connector->setResolver( new ArnDiscoverResolver());
    connector->setService("Demo Chat Server");
    connector->start();

    _arnTime.open("//Chat/Time/value");
    connect( &_arnTime, SIGNAL(changed(QString)), this, SLOT(doTimeUpdate(QString)));

    _commonSapi.open("//Chat/Pipes/pipeCommon");
    _commonSapi.batchConnectTo( this, "sapi");

    _soleSapi.open("//Chat/Pipes/pipe", ArnSapi::Mode::UuidAutoDestroy);
    _soleSapi.batchConnectTo( this, "sapi");

    _soleSapi.pv_infoQ();
    _soleSapi.pv_list();
}

MainWindow::~MainWindow()
{
    delete _ui;
}

void MainWindow::doTimeUpdate( QString timeStr)
{
    _ui->timeEdit->setTime( QTime::fromString( timeStr));
}

void MainWindow::doSendLine()
{
    QString myName = _ui->userEdit->text();
    QString line = _ui->lineEdit->text();
    _ui->lineEdit->clear();

    _soleSapi.pv_newMsg( myName, line);
}

void MainWindow::sapiUpdateMsg( int seq, QString name, QString msg)
{
    if (seq >= _chatNameList.size()) {
        _chatNameList.resize( seq + 1);
        _chatMsgList.resize( seq + 1);
    }
    _chatNameList[ seq] = name;
    _chatMsgList[ seq] = msg;

    QString text;
    for (int i = 0; i < _chatNameList.size(); ++i) {
        text += _chatNameList.at(i) + ": " + _chatMsgList.at(i) + "\n";
    }
    _ui->textEdit->setText( text);
}

void MainWindow::sapiInfo( QString name, QString ver)
{
    _ui->appNameLabel->setText( name);
    _ui->verLabel->setText( ver);
}

```

16.3 ArnDemoChat/MainWindow.hpp

Demo Chat Client

```

// Copyright (C) 2010-2014 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their own
// licenses. ArnDemoChat is independent of these licenses; however, use of these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#ifndef MAINWINDOW_HPP
#define MAINWINDOW_HPP

#include "../ArnDemoChatServer/ChatSapi.hpp"
#include <ArnInc/ArnClient.hpp>
#include <ArnInc/ArnItem.hpp>
#include <QMainWindow>
#include <QVector>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow( QWidget *parent = 0);
    ~MainWindow();

private slots:
    void doSendLine();
    void doTimeUpdate( QString timeStr);

    void sapiUpdateMsg( int seq, QString name, QString msg);
    void sapiInfo( QString name, QString ver);

private:
    Ui::MainWindow *_ui;
    QVector<QString> _chatNameList;
    QVector<QString> _chatMsgList;

    ArnClient _arnClient;
    ChatSapi _commonSapi;
    ChatSapi _soleSapi;
    ArnItem _arnTime;
};

#endif // MAINWINDOW_HPP

```

16.4 ArnDemoChatServer/ChatSapi.hpp

Demo Chat Server

```

// Copyright (C) 2010-2014 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their own

```

```

// licenses. ArnDemoChat is independent of these licenses; however, use of these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#ifndef CHATSAPI_HPP
#define CHATSAPI_HPP

#include <ArnInc/ArnSapi.hpp>

class ChatSapi : public ArnSapi
{
    Q_OBJECT
public:
    explicit ChatSapi( QObject* parent = 0) : ArnSapi( parent) {}

signals:
MQ_PUBLIC_ACCESS
    no_queue void pv_list();
    void pv_newMsg( QString name, QString msg);
    void pv_infoQ();

    void rq_updateMsg( int seq, QString name, QString msg);
    void rq_info( QString name, QString ver);
};

#endif // CHATSAPI_HPP

```

16.5 ArnDemoChatServer/main.cpp

Demo Chat Server

```

#include "MainWindow.hpp"
#include <QApplication>
#include <QDebug>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}

```

16.6 ArnDemoChatServer/MainWindow.cpp

Demo Chat Server

```

// Copyright (C) 2010-2014 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//

```

```

// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their own
// licenses. ArnDemoChat is independent of these licenses; however, use of these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#include "MainWindow.hpp"
#include "tmp/ui_MainWindow.h"
#include <ArnInc/ArnItem.hpp>
#include <ArnInc/ArnDiscoverRemote.hpp>
#include <QTime>
#include <QDebug>

MainWindow::MainWindow( QWidget *parent) :
    QMainWindow( parent, Qt::CustomizeWindowHint | Qt::WindowMinimizeButtonHint),
    _ui( new Ui::MainWindow)
{
    _ui->setupUi( this);
    _connectCount = 0;
    doUpdateView();

    _timer1s.start(1000);
    connect( &_timer1s, SIGNAL(timeout()), this, SLOT(doTimeUpdate()));

    _server = new ArnServer( ArnServer::Type::NetSync, this);
    _server->start(0); // Start server on dynamic port

    _discoverRemote = new ArnDiscoverRemote( this);
    _discoverRemote->setService("Demo Chat Server");
    _discoverRemote->addGroup("arndemo/chat");
    _discoverRemote->addCustomProperty("ChatProtoVer", "1.0");
    _discoverRemote->startUseServer( _server);

    _arnTime.open("//Chat/Time/value");

    typedef ArnSapi::Mode SMode;
    _commonSapi = new ChatSapi( this);
    _commonSapi->open("//Chat/Pipes/pipeCommon", SMode::Provider | SMode::UseDefaultCall);
    _commonSapi->batchConnectTo( this, "sapi");

    ArnItem* arnPipes = new ArnItem("//Chat/Pipes/", this);
    connect( arnPipes, SIGNAL(arnItemCreated(QString)), this, SLOT(doNewSession(QString)));
}

MainWindow::~MainWindow()
{
    delete _ui;
}

void MainWindow::doNewSession( QString path)
{
    if (!Arn::isProviderPath( path)) return; // Only provider pipe is used

    typedef ArnSapi::Mode SMode;
    ChatSapi* soleSapi = new ChatSapi( this);
    soleSapi->open( path, SMode::Provider | SMode::UseDefaultCall);
    soleSapi->batchConnectTo( this, "sapi");
    connect( soleSapi, SIGNAL(pipeClosed()), soleSapi, SLOT(deleteLater()));

    connect( soleSapi, SIGNAL(pipeClosed()), this, SLOT(doSessionClosed()));
    ++_connectCount;
    doUpdateView();
}

void MainWindow::doSessionClosed()

```

```

{
    --_connectCount;
    doUpdateView();
}

void MainWindow::doUpdateView()
{
    _ui->connectCount->setText( QString::number( _connectCount));
}

void MainWindow::on_shutDownButton_clicked()
{
    qWarning() << "About to shut down.";
    delete _discoverRemote; // Must be deleted while still in the main eventloop
    _discoverRemote = 0;
    QApplication::quit();
}

void MainWindow::doTimeUpdate()
{
    _arnTime = QTime::currentTime().toString();
}

void MainWindow::sapiList()
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    for (int i = 0; i < _chatNameList.size(); ++i) {
        sapi->rq_updateMsg( i, _chatNameList.at(i), _chatMsgList.at(i));
    }
}

void MainWindow::sapiNewMsg( QString name, QString msg)
{
    _chatNameList += name;
    _chatMsgList += msg;
    int seq = _chatNameList.size() - 1;

    _commonSapi->rq_updateMsg( seq, name, msg);
}

void MainWindow::sapiInfoQ()
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    sapi->rq_info("Arn Chat Demo", "1.2");
}

void MainWindow::sapiDefault( const QByteArray& data)
{
    ChatSapi* sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    qDebug() << "chatDefault:" << data;
    sapi->sendText("Chat Sapi: Can't find method, use $help.");
}

```

16.7 ArnDemoChatServer/MainWindow.hpp

Demo Chat Server

```

// Copyright (C) 2010-2014 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklund.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their own
// licenses. ArnDemoChat is independent of these licenses; however, use of these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation

```



```

// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#ifndef MAINWINDOW_HPP
#define MAINWINDOW_HPP

#include "ChatSapi.hpp"
#include <ArnInc/ArnItem.hpp>
#include <ArnInc/ArnServer.hpp>
#include <QTimer>
#include <QStringList>
#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class ArnDiscoverRemote;

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow( QWidget *parent = 0);
    ~MainWindow();

private slots:
    void doNewSession( QString path);
    void doSessionClosed();
    void doUpdateView();
    void on_shutDownButton_clicked();
    void doTimeUpdate();

    void sapiList();
    void sapiNewMsg( QString name, QString msg);
    void sapiInfoQ();
    void sapiDefault( const QByteArray& data);

private:
    Ui::MainWindow *_ui;
    QStringList _chatNameList;
    QStringList _chatMsgList;
    QTimer _timer1s;
    int _connectCount;

    ArnItem _arnTime;
    ArnServer* _server;
    ChatSapi* _commonSapi;
    ArnDiscoverRemote* _discoverRemote;
};

#endif // MAINWINDOW_HPP

```


Index

- [_depOfferProto](#)
 - [ArnScript, 398](#)
 - [_depProto](#)
 - [ArnScript, 398](#)
 - [_engine](#)
 - [ArnScript, 399](#)
 - [_itemProto](#)
 - [ArnScript, 399](#)
 - [_log2_u](#)
 - [Arn, 55](#)
 - [_log2_ull](#)
 - [Arn, 55](#)
 - [_mod_i](#)
 - [Arn, 55](#)
 - [_mod_ll](#)
 - [Arn, 56](#)
 - [_monitorProto](#)
 - [ArnScript, 399](#)
 - [~ArnAdaptItem](#)
 - [ArnAdaptItem, 80](#)
 - [~ArnBasicItem](#)
 - [ArnBasicItem, 112](#)
 - [~ArnClient](#)
 - [ArnClient, 143](#)
 - [~ArnCoreItem](#)
 - [ArnCoreItem, 164](#)
 - [~ArnDepend](#)
 - [ArnDepend, 167](#)
 - [~ArnDependOffer](#)
 - [ArnDependOffer, 170](#)
 - [~ArnDiscoverAdvertise](#)
 - [ArnDiscoverAdvertise, 175](#)
 - [~ArnDiscoverBrowserB](#)
 - [ArnDiscoverBrowserB, 188](#)
 - [~ArnDiscoverConnector](#)
 - [ArnDiscoverConnector, 197](#)
 - [~ArnDiscoverInfo](#)
 - [ArnDiscoverInfo, 206](#)
 - [~ArnDiscoverRemote](#)
 - [ArnDiscoverRemote, 215](#)
 - [~ArnEvAtomicOp](#)
 - [ArnEvAtomicOp, 226](#)
 - [~ArnEvRefChange](#)
 - [ArnEvRefChange, 243](#)
 - [~ArnEvValueChange](#)
 - [ArnEvValueChange, 248](#)
 - [~ArnEvent](#)
 - [ArnEvent, 230](#)
 - [~ArnItem](#)
 - [ArnItem, 267](#)
 - [~ArnItemB](#)
 - [ArnItemB, 300](#)
 - [~ArnMonitor](#)
 - [ArnMonitor, 338](#)
 - [~ArnPersist](#)
 - [ArnPersist, 351](#)
 - [~ArnPipe](#)
 - [ArnPipe, 358](#)
 - [~ArnRpc](#)
 - [ArnRpc, 371](#)
 - [~ArnScriptJobFactory](#)
 - [ArnScriptJobFactory, 408](#)
 - [~ArnServer](#)
 - [ArnServer, 413](#)
 - [~ArnServerRemote](#)
 - [ArnServerRemote, 420](#)
 - [~ArnZeroConfB](#)
 - [ArnZeroConfB, 428](#)
 - [~ArnZeroConfBrowser](#)
 - [ArnZeroConfBrowser, 435](#)
 - [~ArnZeroConfLookup](#)
 - [ArnZeroConfLookup, 443](#)
 - [~ArnZeroConfRegister](#)
 - [ArnZeroConfRegister, 453](#)
 - [~ArnZeroConfResolve](#)
 - [ArnZeroConfResolve, 464](#)
 - [~EnumTxt](#)
 - [Arn::EnumTxt, 476](#)
 - [~QmlMQtObject](#)
 - [Arn::QmlMQtObject, 518](#)
 - [~XStringMap](#)
 - [Arn::XStringMap, 534](#)
- [ARN_JSCONTEXT](#)
 - [ArnScript.hpp, 601](#)
- [ARN_JSENGINE](#)
 - [ArnScript.hpp, 601](#)
- [ARN_JSVALUE_LIST](#)
 - [ArnScript.hpp, 602](#)
- [ARN_JSVALUE](#)
 - [ArnScript.hpp, 601](#)
- [ARN_ModeRecursiveMutex](#)
 - [ArnCompat.hpp, 573](#)
- [ARN_RecursiveMutex](#)
 - [ArnCompat.hpp, 573](#)
- [ARN_RegExp](#)
 - [ArnCompat.hpp, 574](#)
- [ARN_RegExpValidator](#)
 - [ArnCompat.hpp, 574](#)

- ARN_SIZETYPE
 - ArnCompat.hpp, 574
- ARN_ToRegExp
 - ArnCompat.hpp, 574
- ARNLIBSHARED_EXPORT
 - ArnLib_global.hpp, 586
- ARNREAL
 - Arn.hpp, 569
- ARNRECNAME
 - ArnSync.hpp, 632
- ARNSYNCVER
 - ArnSync.cpp, 631
- ARNXSTRINGMAP_VER
 - XStringMap.hpp, 616
- abortKillRequest
 - ArnClient, 143
- activeServiceNames
 - ArnZeroConfBrowser, 435
- add
 - Arn::XStringMap, 535, 536
 - Arn::XStringMapQml, 554
 - ArnDepend, 167, 168
- addAccess
 - ArnServer, 414
- addBitSet
 - Arn::EnumTxt, 476
- addBitSetTo
 - Arn::EnumTxt, 476
- addConfig
 - ArnScriptJobControl, 404
- addCustomProperty
 - ArnDiscoverAdvertise, 176
- addEnumSet
 - Arn::EnumTxt, 477
- addEnumSetTo
 - Arn::EnumTxt, 477
- addFlagsTo
 - Arn::EnumTxt, 478
- addFreePath
 - ArnServer, 414
- addGroup
 - ArnDiscoverAdvertise, 176
- addIntNum
 - ArnItemQml, 305
- addInterface
 - ArnScriptJobControl, 405
- addInterfaceList
 - ArnScriptJobControl, 405
- addJob
 - ArnScriptJobs, 410
- addMode
 - ArnAdaptItem, 80
 - ArnBasicItem, 112
 - ArnItem, 268
 - ArnItemQml, 305
- addMountPoint
 - ArnClient, 143
- addNum
 - Arn::XStringMap, 537, 538
 - ArnItemQml, 305
- addObject
 - ArnScript, 396
- addPath
 - Arn, 56
- addSenderSignals
 - ArnRpc, 371
- addSubEnum
 - Arn::EnumTxt, 479
- addSubEnumPlainTo
 - Arn::EnumTxt, 479
- addSubEnumTo
 - Arn::EnumTxt, 480
- addSubType
 - ArnZeroConfRegister, 453
- addToArnList
 - ArnClient, 144
- addToDirectHosts
 - ArnDiscoverConnector, 197
- addValue
 - ArnAdaptItem, 80, 81
 - ArnBasicItem, 113
 - ArnItem, 268, 269
- addValues
 - Arn::XStringMap, 538
- addr
 - ArnClient::HostAddrPort, 501
- advertise
 - ArnDependOffer, 171
- advertiseService
 - ArnDiscoverAdvertise, 177
- allMethodIds
 - ArnRpc::MethodsParam::Params, 513
- append
 - Arn::XStringMap, 539, 540
- arg1
 - ArnEvAtomicOp, 226
- arg2
 - ArnEvAtomicOp, 227
- Arn, 53
 - _log2_u, 55
 - _log2_ull, 55
 - _mod_i, 55
 - _mod_ll, 56
 - addPath, 56
 - changeBasePath, 56
 - childPath, 57
 - circVal, 58
 - convertName, 58
 - convertPath, 58
 - debugDepend, 66
 - debugDiscover, 66
 - debugLinkDestroy, 66
 - debugLinkRef, 66
 - debugMDNS, 66
 - debugMonitor, 66
 - debugMonitorTest, 67

- debugQmlNetwork, 67
- debugRPC, 67
- debugRecInOut, 67
- debugShareObj, 67
- debugSizes, 67
- debugThreading, 68
- debugZeroConf, 68
- defaultTcpPort, 68
- fullPath, 59
- hostFromHostWithInfo, 59
- isFolderPath, 60
- isNullPtr, 60
- isPower2, 60
- isProviderPath, 60
- itemName, 61
- log2, 61
- makeHostWithInfo, 62
- makePath, 62
- maxLim, 63
- minLim, 63
- mod, 63
- offHeartbeat, 68
- parentPath, 63
- pathDiscover, 68
- pathDiscoverConnect, 68
- pathDiscoverThis, 69
- pathLocal, 69
- pathLocalSys, 69
- pathServer, 69
- pathServerSessions, 69
- providerPathIf, 64
- rand, 64
- rangeLim, 64
- resourceArnLib, 69
- resourceArnRoot, 70
- twinPath, 65
- uuidPath, 65
- warningMDNS, 70
- Arn.hpp
 - ARNREAL, 569
 - DATASTREAM_VER, 570
- Arn::_InitEnumTxt, 71
 - enumTxt, 71
 - enumVal, 71
 - ns, 71
- Arn::_InitSubEnum, 72
 - enumTxtClass, 72
 - factor, 73
 - mask, 73
- Arn::Allow, 73
 - E, 73
- Arn::ClientSyncMode, 470
 - E, 470
- Arn::Coding, 471
 - E, 471
- Arn::DataType, 471
 - E, 472
- Arn::EnumTxt, 472
 - ~EnumTxt, 476
 - addBitSet, 476
 - addBitSetTo, 476
 - addEnumSet, 477
 - addEnumSetTo, 477
 - addFlagsTo, 478
 - addSubEnum, 479
 - addSubEnumPlainTo, 479
 - addSubEnumTo, 480
 - clear, 481
 - enumCount, 481
 - EnumTxt, 475
 - flagsFromString, 481
 - flagsFromStringList, 482
 - flagsToString, 482
 - flagsToStringList, 483
 - getBasicTextList, 484
 - getBitSet, 484
 - getEnumSet, 485
 - getEnumVal, 486
 - getSubEnumVal, 487, 488
 - getTxt, 488
 - getTxtString, 489
 - humanize, 490
 - isFlag, 490
 - loadBitSet, 490, 491
 - loadEnumSet, 492
 - name, 493
 - numToStr, 493
 - setMissingTxt, 493
 - setTxt, 494
 - setTxtRef, 494
 - setTxtString, 495
 - strToBitpos, 495
 - strToNum, 495
 - subEnumAt, 495
 - subEnumCount, 496
 - subEnumNameAt, 496
 - subEnumPropAt, 497
- Arn::EnumTxt::IncludeMode, 501
 - E, 502
- Arn::ExportCode, 499
 - E, 499
- Arn::InfoType, 502
 - E, 503
- Arn::LinkFlags, 504
 - E, 504
- Arn::NameF, 510
 - E, 510
- Arn::ObjectMode, 511
 - E, 511
- Arn::ObjectSyncMode, 511
 - E, 512
- Arn::QmlMFileIO, 513
 - error, 515
 - path, 515, 516
 - pathChanged, 515
 - QmlMFileIO, 514

- read, 515
- readBytes, 515
- setPath, 515
- write, 516
- writeBytes, 516
- Arn::QmlMQtObject, 517
 - ~QmlMQtObject, 518
 - classBegin, 518
 - completed, 518
 - componentComplete, 519
 - data, 519, 520
 - data_append, 519
 - data_at, 519
 - data_clear, 519
 - data_count, 519
 - parent, 520
 - parentChanged, 520
 - parentItem, 520
 - QmlMQtObject, 518
 - setParentItem, 520
- Arn::QmlMSys, 521
 - quickTypeRun, 522
 - xstringToEnum, 522
- Arn::SameValue, 522
 - E, 523
- Arn::XStringMap, 530
 - ~XStringMap, 534
 - add, 535, 536
 - addNum, 537, 538
 - addValues, 538
 - append, 539, 540
 - clear, 541
 - fromXString, 541
 - indexOf, 541, 542
 - indexOfValue, 542
 - info, 542
 - key, 542, 543
 - keyRef, 543
 - keyString, 543, 544
 - keys, 543
 - maxEnumOf, 544
 - operator+=", 544
 - operator=, 544
 - Options, 533
 - options, 545
 - remove, 545
 - removeValue, 546
 - reverseOrder, 546
 - set, 546–548
 - setEmptyKeysToValue, 548
 - setKey, 548
 - setOptions, 548
 - size, 548
 - squeeze, 549
 - stringCode, 549
 - stringDecode, 549
 - toVariantMap, 549
 - toXString, 549
 - toXStringString, 549
 - value, 550
 - valueRef, 551
 - valueString, 551, 552
 - values, 551
 - XStringMap, 534
- Arn::XStringMapOptions, 552
 - E, 552
- Arn::XStringMapQml, 553
 - add, 554
 - clear, 555
 - indexOf, 555
 - indexOfValue, 555
 - key, 555
 - keys, 556
 - remove, 556
 - removeValue, 556
 - set, 556, 557
 - setEmptyKeysToValue, 557
 - size, 558
 - toMap, 557
 - value, 557
 - values, 558
 - xstring, 558
- ArnAdaptItem, 74
 - ~ArnAdaptItem, 80
 - addMode, 80
 - addValue, 80, 81
 - ArnAdaptItem, 79
 - arnEventCallback, 81
 - ArnEventCB, 79
 - arnExport, 81
 - arnImport, 82
 - ChangedCallback, 82
 - ChangedCB, 79
 - close, 83
 - destroyLink, 83
 - destroyLinkLocal, 83
 - getMode, 83
 - isAutoDestroy, 84
 - isBiDirMode, 84
 - isFolder, 84
 - isIgnoreSameValue, 85
 - isMaster, 85
 - isOpen, 85
 - isPipeMode, 86
 - isProvider, 86
 - isSaveMode, 86
 - isUncrossed, 87
 - itemId, 87
 - linkDestroyedCallback, 87
 - LinkDestroyedCB, 79
 - linkId, 88
 - mutex, 88
 - name, 88
 - open, 89
 - operator+=", 89
 - operator=, 89–91

- path, 91
- refCount, 92
- reference, 92
- setArnEventCallback, 92
- setAutoDestroy, 93
- setBiDirMode, 93
- setBits, 93
- setChangedCallback, 94
- setIgnoreSameValue, 94
- setLinkDestroyedCallback, 95
- setMaster, 95
- setPipeMode, 95
- setReference, 96
- setSaveMode, 96
- setUncrossed, 97
- setValue, 97–101
- syncMode, 102
- thread, 102
- toBool, 102
- toByteArray, 103
- toDouble, 103
- toInt, 103
- toInt64, 104
- toReal, 104
- toString, 105
- toUInt, 105
- toUInt64, 105
- toVariant, 106
- type, 106
- ArnAdaptItem.cpp
 - MUTEX_CALL_RET, 562
 - MUTEX_CALL, 561
- ArnAtomicOp, 107
 - E, 107
 - NS, 107
- ArnBasicItem, 108
 - ~ArnBasicItem, 112
 - addMode, 112
 - addValue, 113
 - ArnBasicItem, 112
 - ArnBasicItemEventHandler, 139
 - arnExport, 114
 - arnImport, 114
 - ArnM, 333
 - close, 114
 - destroyLink, 115
 - destroyLinkLocal, 115
 - eventHandler, 115
 - getMode, 116
 - isAssigning, 116
 - isAtomicOpProvider, 117
 - isAutoDestroy, 117
 - isBiDirMode, 117
 - isFolder, 118
 - isIgnoreSameValue, 118
 - isMaster, 118
 - isOpen, 119
 - isPipeMode, 119
 - isProvider, 119
 - isSaveMode, 120
 - isUncrossed, 120
 - itemId, 120
 - linkId, 121
 - name, 121
 - open, 122
 - operator+=", 122
 - operator=", 122–124
 - path, 124
 - refCount, 125
 - reference, 125
 - setAtomicOpProvider, 125
 - setAutoDestroy, 125
 - setBiDirMode, 126
 - setBits, 126
 - setEventHandler, 127
 - setIgnoreSameValue, 127
 - setMaster, 127
 - setPipeMode, 128
 - setReference, 128
 - setSaveMode, 129
 - setUncrossed, 129
 - setValue, 129–133
 - syncMode, 134
 - thread, 134
 - toBool, 134
 - toByteArray, 135
 - toDouble, 135
 - toInt, 135
 - toInt64, 136
 - toReal, 136
 - toString, 137
 - toUInt, 137
 - toUInt64, 137
 - toVariant, 138
 - type, 138
- ArnBasicItemEventHandler
 - ArnBasicItem, 139
 - ArnCoreItem, 165
- arnChildDeleted
 - ArnMonitor, 338
- arnChildFound
 - ArnMonitor, 338
- arnChildFoundFolder
 - ArnMonitor, 339
- arnChildFoundLeaf
 - ArnMonitor, 339
- arnChildModeChanged
 - ArnMonitor, 340
- ArnClient, 139
 - ~ArnClient, 143
 - abortKillRequest, 143
 - addMountPoint, 143
 - addToArnList, 144
 - ArnClient, 143
 - arnList, 144
 - chatReceived, 145

- chatSend, 145
- clearArnList, 146
- close, 146
- ConnectStat, 142
- connectStatus, 147
- connectToArn, 147
- connectToArnList, 147
- connectionStatusChanged, 146
- disconnectFromArn, 148
- freePaths, 148
- getClient, 148
- getTraffic, 150
- HostList, 142
- id, 150
- isDemandLogin, 150
- isReConnect, 151
- isReContact, 151
- killRequested, 152
- loginRequired, 152
- loginToArn, 152
- loginToArnHashed, 153
- passwordHash, 154
- receiveTimeout, 154
- registerClient, 154
- remoteWhoIAm, 155
- removeMountPoint, 155
- setAutoConnect, 156
- setDemandLogin, 156
- setMountPoint, 157
- setReceiveTimeout, 157
- setSyncMode, 158
- setWhoIAm, 158
- SyncMode, 142
- syncMode, 159
- tcpConnected, 159
- tcpDisConnected, 159
- tcpError, 160
- ArnClient::HostAddrPort, 500
 - addr, 501
 - HostAddrPort, 501
 - port, 501
- ArnClientConnectStat, 160
 - E, 160
 - NS, 161
- ArnClientReg, 161
 - get, 162
 - instance, 162
 - remove, 162
 - store, 162
- ArnCompat.hpp
 - ARN_ModeRecursiveMutex, 573
 - ARN_RecursiveMutex, 573
 - ARN_RegExp, 574
 - ARN_RegExpValidator, 574
 - ARN_SIZETYPE, 574
 - ARN_ToRegExp, 574
- ArnCoreItem, 163
 - ~ArnCoreItem, 164
 - ArnBasicItemEventHandler, 165
 - ArnCoreItem, 164
 - thread, 164
- ArnCoreItem::Heritage, 500
 - E, 500
- ArnCoreItemList
 - ArnLink.hpp, 621
- ArnDepend, 165
 - ~ArnDepend, 167
 - add, 167, 168
 - ArnDepend, 167
 - completed, 168
 - DepSlot, 167
 - setMonitorName, 168
 - startMonitor, 168
- ArnDepend.cpp
 - ArnDependPath, 565
- ArnDependOffer, 169
 - ~ArnDependOffer, 170
 - advertise, 171
 - ArnDependOffer, 170
 - setStateId, 172
 - setStateName, 172
 - stateId, 172
 - stateName, 173
- ArnDependPath
 - ArnDepend.cpp, 565
- ArnDiscover, 70
- ArnDiscover::Type, 529
 - E, 529
- ArnDiscoverAdvertise, 173
 - ~ArnDiscoverAdvertise, 175
 - addCustomProperty, 176
 - addGroup, 176
 - advertiseService, 177
 - ArnDiscoverAdvertise, 175
 - currentService, 177
 - customProperties, 178
 - groups, 178
 - service, 178
 - serviceChangeError, 179
 - serviceChanged, 179
 - setCustomProperties, 180
 - setGroups, 180
 - setService, 181
 - state, 181
- ArnDiscoverAdvertise::State, 525
 - E, 525
- ArnDiscoverBrowser, 182
 - ArnDiscoverBrowser, 184
 - browse, 184
 - isBrowsing, 185
 - setFilter, 185, 186
 - stopBrowse, 186
- ArnDiscoverBrowserB, 187
 - ~ArnDiscoverBrowserB, 188
 - ArnDiscoverBrowserB, 188
 - ArnDiscoverInfo, 212

- defaultStopState, 189
- goTowardState, 189
- IdToIndex, 189
- indexTold, 190
- infoById, 190
- infoByIndex, 191
- infoByName, 191
- infoUpdated, 192
- serviceAdded, 192
- serviceCount, 193
- serviceNameTold, 193
- serviceRemoved, 194
- setDefaultStopState, 194
- ArnDiscoverConnector, 195
 - ~ArnDiscoverConnector, 197
 - addToDirectHosts, 197
 - ArnDiscoverConnector, 197
 - clearDirectHosts, 198
 - clientReadyToConnect, 198
 - directHostPrio, 199
 - discoverHostPrio, 199
 - externalClientConnect, 199
 - id, 200
 - resolveRefreshTimeout, 200
 - service, 200
 - setDirectHostPrio, 201
 - setDiscoverHostPrio, 201
 - setExternalClientConnect, 202
 - setResolveRefreshTimeout, 203
 - setResolver, 202
 - setService, 203
 - start, 204
- ArnDiscoverInfo, 204
 - ~ArnDiscoverInfo, 206
 - ArnDiscoverBrowserB, 212
 - ArnDiscoverInfo, 206
 - domain, 206
 - groups, 206
 - hostIp, 207
 - hostIpString, 207
 - hostName, 207
 - hostPort, 208
 - hostPortString, 208
 - hostWithInfo, 208
 - inProgress, 209
 - isError, 209
 - operator=, 210
 - properties, 210
 - resolvCode, 210
 - serviceName, 210
 - state, 211
 - stopState, 211
 - type, 211
 - typeString, 212
- ArnDiscoverInfo::State, 523
 - E, 523
- ArnDiscoverRemote, 213
 - ~ArnDiscoverRemote, 215
 - ArnDiscoverRemote, 215
 - clientReadyToConnect, 215
 - defaultService, 216
 - initialServiceTimeout, 216
 - newConnector, 216
 - setDefaultService, 217
 - setInitialServiceTimeout, 217
 - setService, 218
 - startUseNewServer, 218
 - startUseServer, 219
- ArnDiscoverResolver, 219
 - ArnDiscoverResolver, 221
 - defaultService, 221
 - resolve, 222
 - setDefaultService, 222
- ArnError, 223
 - E, 224
- ArnError::StdCode, 526
 - E, 526
- ArnEvAtomicOp, 224
 - ~ArnEvAtomicOp, 226
 - arg1, 226
 - arg2, 227
 - ArnEvAtomicOp, 226
 - makeHeapClone, 227
 - Op, 226
 - op, 227
 - type, 227
- ArnEvLinkCreate, 234
 - ArnEvLinkCreate, 235
 - arnLink, 236
 - isLastLink, 236
 - makeHeapClone, 236
 - path, 236
 - type, 236
- ArnEvModeChange, 237
 - ArnEvModeChange, 238
 - linkId, 238
 - makeHeapClone, 238
 - mode, 238
 - path, 239
 - type, 239
- ArnEvMonitor, 239
 - ArnEvMonitor, 240
 - data, 241
 - isLocal, 241
 - makeHeapClone, 241
 - monEvType, 241
 - sessionHandler, 241
 - type, 241
- ArnEvRefChange, 242
 - ~ArnEvRefChange, 243
 - ArnEvRefChange, 243
 - makeHeapClone, 243
 - refStep, 243
 - type, 244
- ArnEvRetired, 244
 - ArnEvRetired, 245

- isBelow, 246
- isGlobal, 246
- makeHeapClone, 246
- startLink, 246
- type, 246
- ArnEvValueChange, 247
 - ~ArnEvValueChange, 248
 - ArnEvValueChange, 248
 - handleData, 248
 - makeHeapClone, 248
 - sendId, 249
 - type, 249
 - valueData, 249
- ArnEvZeroRef, 250
 - ArnEvZeroRef, 251
 - arnLink, 251
 - makeHeapClone, 251
 - type, 251
- ArnEvent, 228
 - ~ArnEvent, 230
 - ArnEvent, 229
 - baseType, 230
 - copyOpt, 230
 - Idx, 229
 - inhibitPendingChain, 230
 - isArnEvent, 230
 - makeHeapClone, 231
 - setTarget, 231
 - setTargetMutex, 231
 - setTargetPendingChain, 231
 - target, 231
 - toldx, 231, 232
 - toString, 232
- ArnEvent.cpp
 - TO_IDX_RETVAL, 567
- arnEventCallback
 - ArnAdaptItem, 81
- ArnEventCB
 - ArnAdaptItem, 79
- ArnEventIdx, 232
 - E, 233
- arnExport
 - ArnAdaptItem, 81
 - ArnBasicItem, 114
 - ArnItem, 269
- arnImport
 - ArnAdaptItem, 82
 - ArnBasicItem, 114
 - ArnItem, 269
- ArnInterface, 252
 - bytes, 256
 - changeBasePath, 256
 - childPath, 256
 - Data Type, 254
 - exist, 256
 - info, 261
 - intNum, 256
 - isFolder, 257
 - isFolderPath, 257
 - isLeaf, 257
 - isProviderPath, 257
 - itemName, 257
 - items, 258
 - makePath, 258
 - NameF, 255
 - num, 258
 - ObjectMode, 255
 - providerPath, 258
 - SameValue, 255
 - setBytes, 258
 - setIntNum, 259
 - setNum, 259
 - setString, 259
 - setValue, 259
 - setVariant, 260
 - string, 260
 - twinPath, 260
 - value, 260
 - variant, 261
- ArnItem, 262
 - ~ArnItem, 267
 - addMode, 268
 - addValue, 268, 269
 - arnExport, 269
 - arnImport, 269
 - ArnItem, 266, 267
 - arnItemCreated, 270
 - arnModeChanged, 270
 - bypassDelayPending, 271
 - changed, 271, 272
 - delay, 273
 - getMode, 273
 - isAutoDestroy, 273
 - isBiDirMode, 274
 - isDelayPending, 274
 - isFolder, 274
 - isIgnoreSameValue, 275
 - isMaster, 275
 - isPipeMode, 275
 - isProvider, 276
 - isSaveMode, 276
 - isTemplate, 276
 - isUncrossed, 277
 - modeChanged, 277
 - openFolder, 278
 - openUuid, 278
 - openUuidPipe, 278
 - operator+=", 279
 - operator=", 279–281
 - setAutoDestroy, 281
 - setBiDirMode, 281
 - setBits, 281
 - setBlockEcho, 282
 - setDelay, 282
 - setIgnoreSameValue, 283
 - setMaster, 283

- setPipeMode, 283
- setSaveMode, 284
- setTemplate, 284
- setUncrossed, 285
- setValue, 285–287, 289–293
- syncMode, 294
- toBool, 294
- toByteArray, 294
- toDouble, 295
- toInt, 295
- toInt64, 296
- toReal, 296
- toString, 296
- toUInt, 297
- toUInt64, 297
- toVariant, 298
- toggleBool, 295
- type, 298
- ArnItem.cpp
 - operator<<, 617
- ArnItem.hpp
 - operator<<, 583
- arnItemCreated
 - ArnItem, 270
 - ArnMonitor, 340
- arnItemDeleted
 - ArnMonitor, 341
- arnItemModeChanged
 - ArnMonitor, 341
- ArnItemQml, 302
 - addIntNum, 305
 - addMode, 305
 - addNum, 305
 - atomicOpProvider, 306
 - autoDestroyMode, 307
 - biDirMode, 307
 - bytes, 307
 - delay, 307
 - getMode, 306
 - ignoreSameValue, 307
 - intNum, 308
 - masterMode, 308
 - num, 308
 - path, 308
 - pipeMode, 308
 - saveMode, 309
 - setBits, 306
 - string, 309
 - type, 309
 - useUuid, 309
 - variant, 309
 - variantType, 310
- ArnItemValve, 310
 - ArnItemValve, 312
 - changed, 312
 - isAutoDestroy, 312
 - isMaster, 313
 - isSaveMode, 313
 - operator=, 313
 - setAutoDestroy, 314
 - setMaster, 314
 - setSaveMode, 314
 - setTarget, 315
 - setValue, 315
 - switchMode, 315
 - toBool, 316
- ArnItemValve::SwitchMode, 527
 - E, 527
- ArnItemB, 299
 - ~ArnItemB, 300
 - ArnItemB, 300
 - arnLinkDestroyed, 301
 - open, 301
- ArnLib_global.hpp
 - ARNLIBSHARED_EXPORT, 586
- arnLink
 - ArnEvLinkCreate, 236
 - ArnEvZeroRef, 251
- ArnLink.hpp
 - ArnCoreItemList, 621
 - ArnLinkList, 621
- arnLinkDestroyed
 - ArnItemB, 301
- ArnLinkList
 - ArnLink.hpp, 621
- ArnLinkValue, 316
 - ArnLinkValue, 316
 - localUpdateCount, 317
 - valueByteArray, 317
 - valueInt, 317
 - valueReal, 317
 - valueString, 317
 - valueVariant, 317
- arnList
 - ArnClient, 144
- arnModeChanged
 - ArnItem, 270
- ArnMonEventType, 333
 - E, 333
 - NS, 334
- ArnMonitor, 334
 - ~ArnMonitor, 338
 - arnChildDeleted, 338
 - arnChildFound, 338
 - arnChildFoundFolder, 339
 - arnChildFoundLeaf, 339
 - arnChildModeChanged, 340
 - arnItemCreated, 340
 - arnItemDeleted, 341
 - arnItemModeChanged, 341
 - ArnMonitor, 337
 - client, 341
 - clientId, 342
 - foundChildDeleted, 342
 - monitorClosed, 342
 - monitorPath, 343

- reStart, [343](#)
- reference, [343](#)
- setClient, [343](#), [344](#)
- setMonitorPath, [344](#)
- setReference, [344](#)
- start, [345](#)
- ArnMonitorQml, [346](#)
 - clientId, [348](#)
 - monitorPath, [348](#)
 - reStart, [348](#)
- ArnNullptr, [349](#)
 - operator T*, [349](#)
- ArnPersist, [350](#)
 - ~ArnPersist, [351](#)
 - ArnPersist, [351](#)
 - doArchive, [352](#)
 - flush, [352](#)
 - setArchiveDir, [352](#)
 - setMountPoint, [353](#)
 - setPersistDir, [353](#)
 - setVcs, [354](#)
 - setupDataBase, [354](#)
- ArnPipe, [355](#)
 - ~ArnPipe, [358](#)
 - ArnPipe, [357](#), [358](#)
 - changed, [358](#)
 - isAutoDestroy, [359](#)
 - isCheckSeq, [359](#)
 - isMaster, [359](#)
 - isSendSeq, [360](#)
 - openUuid, [360](#)
 - operator=, [360](#)
 - outOfSequence, [361](#)
 - setAutoDestroy, [361](#)
 - setCheckSeq, [361](#)
 - setMaster, [362](#)
 - setSendSeq, [362](#)
 - setValue, [362](#)
 - setValueOverwrite, [363](#)
- ArnQml, [364](#)
 - arnRootPath, [366](#)
 - instance, [366](#)
 - setArnRootPath, [367](#)
 - setup, [367](#)
- ArnQml.hpp
 - QML_ENGINE, [594](#)
 - QML_LIST_PROPERTY, [594](#)
 - QML_NETACC_FACTORY, [594](#)
 - QML_PARSER_STATUS, [595](#)
 - QML_QUICK_TYPE, [595](#)
 - QML_Qt4, [595](#)
- ArnQml::UseFlags, [530](#)
 - E, [530](#)
- arnRootPath
 - ArnQml, [366](#)
- ArnRpc, [368](#)
 - ~ArnRpc, [371](#)
 - addSenderSignals, [371](#)
 - ArnRpc, [371](#)
 - batchConnect, [372](#), [373](#)
 - defaultCall, [373](#)
 - getHeartBeatCheck, [375](#)
 - getHeartBeatSend, [375](#)
 - heartBeatChanged, [375](#)
 - heartBeatReceived, [376](#)
 - invoke, [376](#)
 - isHeartBeatOk, [377](#)
 - methodPrefix, [378](#)
 - Mode, [371](#)
 - mode, [378](#)
 - open, [378](#)
 - outOfSequence, [378](#)
 - pipe, [378](#)
 - pipeClosed, [379](#)
 - pipePath, [379](#)
 - receiver, [379](#)
 - rpcSender, [379](#), [380](#)
 - sendText, [380](#)
 - setHeartBeatCheck, [380](#)
 - setHeartBeatSend, [381](#)
 - setIncludeSender, [381](#)
 - setMethodPrefix, [381](#)
 - setMode, [381](#)
 - setPipe, [382](#)
 - setReceiver, [382](#)
 - textReceived, [382](#)
- ArnRpc.cpp
 - RPC_STORAGE_NAME, [627](#)
- ArnRpc.hpp
 - MQ_ARG, [598](#)
 - no_queue, [598](#)
- ArnRpc::Invoke, [503](#)
 - E, [503](#)
- ArnRpc::MethodsParam::Params, [512](#)
 - allMethodIds, [513](#)
 - methodIdsTab, [513](#)
 - paramNames, [513](#)
- ArnRpcMode, [383](#)
 - E, [383](#)
- ArnSapi, [384](#)
 - ArnSapi, [387](#)
 - batchConnectFrom, [387](#)
 - batchConnectTo, [388](#)
 - defaultPath, [388](#)
 - open, [388](#)
 - setDefaultPath, [389](#)
- ArnSapi.hpp
 - MQ_PUBLIC_ACCESS, [600](#)
- ArnSapiQml, [390](#)
 - heartBeatCheck, [393](#)
 - heartBeatSend, [393](#)
 - isHeartBeatOk, [392](#)
 - Mode, [392](#)
 - mode, [393](#)
 - pipePath, [393](#)
 - receiver, [394](#)

- ArnScript, 394
 - _depOfferProto, 398
 - _depProto, 398
 - _engine, 399
 - _itemProto, 399
 - _monitorProto, 399
 - addObject, 396
 - ArnScript, 395, 396
 - callFunc, 396
 - engine, 396
 - errorLog, 396
 - errorText, 397
 - evaluate, 397
 - evaluateFile, 397
 - globalProperty, 397
 - idName, 397
 - logUncaughtError, 398
 - printFunction, 398
 - setInterruptedText, 398
- ArnScript.hpp
 - ARN_JSCONTEXT, 601
 - ARN_JSENGINE, 601
 - ARN_JSVALUE_LIST, 602
 - ARN_JSVALUE, 601
- ArnScriptJob, 400
 - ArnScriptJob, 401
 - errorLog, 401
 - name, 402
 - poll, 402
 - quit, 401
 - running, 402
 - setWatchDogTime, 401
 - sigQuit, 402
 - sleepState, 402
 - watchDog, 403
 - yield, 402
- ArnScriptJob.cpp
 - EventQuit, 629
- ArnScriptJobControl, 403
 - addConfig, 404
 - addInterface, 405
 - addInterfaceList, 405
 - ArnScriptJobControl, 404
 - config, 405
 - doSetupJob, 405
 - errorText, 405
 - id, 406
 - loadScriptFile, 406
 - name, 406
 - script, 406
 - scriptChanged, 406
 - setConfig, 406
 - setName, 407
 - setScript, 407
 - setThreaded, 407
- ArnScriptJobFactory, 407
 - ~ArnScriptJobFactory, 408
 - ArnScriptJobFactory, 408
 - installExtension, 408
 - setupInterface, 408
 - setupJsObj, 409
- ArnScriptJobs, 409
 - addJob, 410
 - ArnScriptJobs, 410
 - setFactory, 411
 - start, 411
- ArnScriptJobs::Type, 528
 - E, 528
- ArnServer, 411
 - ~ArnServer, 413
 - addAccess, 414
 - addFreePath, 414
 - ArnServer, 413
 - freePaths, 414
 - isDemandLogin, 415
 - isDemandLoginNet, 415
 - listenAddress, 416
 - noLoginNets, 416
 - port, 416
 - setDemandLogin, 417
 - setNoLoginNets, 417
 - setWhoIam, 418
 - start, 418
- ArnServer::Type, 527
 - E, 528
- ArnServerRemote, 419
 - ~ArnServerRemote, 420
 - ArnServerRemote, 420
 - startUseServer, 420
- ArnServerRemoteSession, 421
 - ArnServerRemoteSession, 422
 - KillMode, 422
- ArnServerRemoteSessionKillMode, 422
 - E, 423
- ArnServerSession, 423
 - ArnServerSession, 424
 - getAllow, 425
 - getTraffic, 425
 - infoReceived, 425
 - loginCompleted, 425
 - loginUserName, 425
 - messageReceived, 425
 - remoteWhoIam, 425
 - sendMessage, 426
 - socket, 426
- ArnSync.cpp
 - ARNSYNCOVER, 631
- ArnSync.hpp
 - ARNRECNAME, 632
- ArnZeroConf, 70
- ArnZeroConf.hpp
 - DNSServiceRef, 607
- ArnZeroConf::Error, 498
 - E, 498
- ArnZeroConf::State, 524
 - E, 524

- ArnZeroConfBrowser, 431
 - ~ArnZeroConfBrowser, 435
 - activeServiceNames, 435
 - ArnZeroConfBrowser, 433, 435
 - ArnZeroConfIntern, 440
 - browse, 436
 - browseError, 436
 - getNextId, 436
 - isBrowsing, 437
 - serviceAdded, 437
 - serviceChanged, 438
 - serviceNameTold, 438
 - serviceRemoved, 439
 - setSubType, 439
 - stopBrowse, 440
 - subType, 440
- ArnZeroConfIntern
 - ArnZeroConfBrowser, 440
 - ArnZeroConfLookup, 448
 - ArnZeroConfRegister, 461
 - ArnZeroConfResolve, 469
- ArnZeroConfLookup, 441
 - ~ArnZeroConfLookup, 443
 - ArnZeroConfIntern, 448
 - ArnZeroConfLookup, 443
 - host, 444
 - hostAddr, 444
 - id, 444
 - isForceQtDnsLookup, 445
 - lookup, 445
 - lookupError, 446
 - lookped, 445
 - releaseLookup, 446
 - setForceQtDnsLookup, 446
 - setHost, 447
 - setId, 447
- ArnZeroConfRegister, 448
 - ~ArnZeroConfRegister, 453
 - addSubType, 453
 - ArnZeroConfIntern, 461
 - ArnZeroConfRegister, 451
 - currentServiceName, 453
 - getTxtRecordMap, 454
 - host, 454
 - port, 455
 - registerService, 456
 - registered, 455
 - registrationError, 456
 - releaseService, 456
 - serviceName, 457
 - setHost, 457
 - setPort, 458
 - setServiceName, 458
 - setSubTypes, 458
 - setTxtRecord, 459
 - setTxtRecordMap, 459
 - subTypes, 460
 - txtRecord, 460
- ArnZeroConfResolve, 461
 - ~ArnZeroConfResolve, 464
 - ArnZeroConfIntern, 469
 - ArnZeroConfResolve, 463, 464
 - getTxtRecordMap, 465
 - host, 465
 - id, 466
 - port, 466
 - releaseResolve, 466
 - resolve, 466
 - resolveError, 467
 - resolved, 467
 - serviceName, 468
 - setId, 468
 - setServiceName, 468
 - txtRecord, 469
- ArnZeroConfB, 426
 - ~ArnZeroConfB, 428
 - ArnZeroConfB, 427
 - domain, 428
 - fullServiceType, 428
 - serviceType, 428
 - setDomain, 429
 - setServiceType, 429
 - setSocketType, 430
 - socketType, 430
 - state, 430
- ArnM, 318
 - ArnBasicItem, 333
 - defaultIgnoreSameValue, 320
 - destroyLink, 320
 - destroyLinkLocal, 321
 - errorLog, 321
 - errorLogSig, 322
 - errorSysName, 322
 - exist, 322
 - info, 322
 - instance, 323
 - isAtomicOpProvider, 323
 - isFolder, 323
 - isLeaf, 324
 - isMainThread, 324
 - isThreadedApp, 324
 - items, 325
 - loadFromDirRoot, 325
 - loadFromFile, 325
 - saveToFile, 326
 - setAtomicOpProvider, 326
 - setConsoleError, 327
 - setDefaultIgnoreSameValue, 327
 - setSkipLocalSysLoading, 327
 - setValue, 328–330
 - setErrorlog, 328
 - skipLocalSysLoading, 330
 - valueByteArray, 330
 - valueDouble, 331
 - valueInt, 331
 - valueReal, 331

- valueString, [332](#)
- valueVariant, [332](#)
- atomicOpProvider
 - ArnItemQml, [306](#)
- autoDestroyMode
 - ArnItemQml, [307](#)
- baseType
 - ArnEvent, [230](#)
- batchConnect
 - ArnRpc, [372](#), [373](#)
- batchConnectFrom
 - ArnSapi, [387](#)
- batchConnectTo
 - ArnSapi, [388](#)
- biDirMode
 - ArnItemQml, [307](#)
- browse
 - ArnDiscoverBrowser, [184](#)
 - ArnZeroConfBrowser, [436](#)
- browseError
 - ArnZeroConfBrowser, [436](#)
- bypassDelayPending
 - ArnItem, [271](#)
- bytes
 - ArnInterface, [256](#)
 - ArnItemQml, [307](#)
- callFunc
 - ArnScript, [396](#)
- changeBasePath
 - Arn, [56](#)
 - ArnInterface, [256](#)
- changed
 - ArnItem, [271](#), [272](#)
 - ArnItemValve, [312](#)
 - ArnPipe, [358](#)
- ChangedCallback
 - ArnAdaptItem, [82](#)
- ChangedCB
 - ArnAdaptItem, [79](#)
- chatReceived
 - ArnClient, [145](#)
- chatSend
 - ArnClient, [145](#)
- childPath
 - Arn, [57](#)
 - ArnInterface, [256](#)
- circVal
 - Arn, [58](#)
- classBegin
 - Arn::QmlMQtObject, [518](#)
- clear
 - Arn::EnumTxt, [481](#)
 - Arn::XStringMap, [541](#)
 - Arn::XStringMapQml, [555](#)
- clearArnList
 - ArnClient, [146](#)
- clearDirectHosts
 - ArnDiscoverConnector, [198](#)
- client
 - ArnMonitor, [341](#)
- clientId
 - ArnMonitor, [342](#)
 - ArnMonitorQml, [348](#)
- clientReadyToConnect
 - ArnDiscoverConnector, [198](#)
 - ArnDiscoverRemote, [215](#)
- close
 - ArnAdaptItem, [83](#)
 - ArnBasicItem, [114](#)
 - ArnClient, [146](#)
- completed
 - Arn::QmlMQtObject, [518](#)
 - ArnDepend, [168](#)
- componentComplete
 - Arn::QmlMQtObject, [519](#)
- config
 - ArnScriptJobControl, [405](#)
- ConnectStat
 - ArnClient, [142](#)
- connectStatus
 - ArnClient, [147](#)
- connectToArn
 - ArnClient, [147](#)
- connectToArnList
 - ArnClient, [147](#)
- connectionStatusChanged
 - ArnClient, [146](#)
- convertName
 - Arn, [58](#)
- convertPath
 - Arn, [58](#)
- copyOpt
 - ArnEvent, [230](#)
- currentService
 - ArnDiscoverAdvertise, [177](#)
- currentServiceName
 - ArnZeroConfRegister, [453](#)
- customProperties
 - ArnDiscoverAdvertise, [178](#)
- DATASTREAM_VER
 - Arn.hpp, [570](#)
- DNSServiceRef
 - ArnZeroConf.hpp, [607](#)
- data
 - Arn::QmlMQtObject, [519](#), [520](#)
 - ArnEvMonitor, [241](#)
- data_append
 - Arn::QmlMQtObject, [519](#)
- data_at
 - Arn::QmlMQtObject, [519](#)
- data_clear
 - Arn::QmlMQtObject, [519](#)
- data_count
 - Arn::QmlMQtObject, [519](#)
- DataType

- ArnInterface, [254](#)
- debugDepend
 - Arn, [66](#)
- debugDiscover
 - Arn, [66](#)
- debugLinkDestroy
 - Arn, [66](#)
- debugLinkRef
 - Arn, [66](#)
- debugMDNS
 - Arn, [66](#)
- debugMonitor
 - Arn, [66](#)
- debugMonitorTest
 - Arn, [67](#)
- debugQmlNetwork
 - Arn, [67](#)
- debugRPC
 - Arn, [67](#)
- debugReclnOut
 - Arn, [67](#)
- debugShareObj
 - Arn, [67](#)
- debugSizes
 - Arn, [67](#)
- debugThreading
 - Arn, [68](#)
- debugZeroConf
 - Arn, [68](#)
- defaultCall
 - ArnRpc, [373](#)
- defaultIgnoreSameValue
 - ArnM, [320](#)
- defaultPath
 - ArnSapi, [388](#)
- defaultService
 - ArnDiscoverRemote, [216](#)
 - ArnDiscoverResolver, [221](#)
- defaultStopState
 - ArnDiscoverBrowserB, [189](#)
- defaultTcpPort
 - Arn, [68](#)
- delay
 - ArnItem, [273](#)
 - ArnItemQml, [307](#)
- DepSlot
 - ArnDepend, [167](#)
- destroyLink
 - ArnAdaptItem, [83](#)
 - ArnBasicItem, [115](#)
 - ArnM, [320](#)
- destroyLinkLocal
 - ArnAdaptItem, [83](#)
 - ArnBasicItem, [115](#)
 - ArnM, [321](#)
- directHostPrio
 - ArnDiscoverConnector, [199](#)
- disconnectFromArn
 - ArnClient, [148](#)
- discoverHostPrio
 - ArnDiscoverConnector, [199](#)
- doArchive
 - ArnPersist, [352](#)
- doSetupJob
 - ArnScriptJobControl, [405](#)
- doc/Changelog_Todo.md(4.0.0), [559](#)
- doc/Description.md(4.0.0), [559](#)
- doc/HelpIndex.txt(4.0.0), [559](#)
- doc/Install.md(4.0.0), [559](#)
- doc/Internals.md(4.0.0), [559](#)
- domain
 - ArnDiscoverInfo, [206](#)
 - ArnZeroConfB, [428](#)
- E
 - Arn::Allow, [73](#)
 - Arn::ClientSyncMode, [470](#)
 - Arn::Coding, [471](#)
 - Arn::DataType, [472](#)
 - Arn::EnumTxt::IncludeMode, [502](#)
 - Arn::ExportCode, [499](#)
 - Arn::InfoType, [503](#)
 - Arn::LinkFlags, [504](#)
 - Arn::NameF, [510](#)
 - Arn::ObjectMode, [511](#)
 - Arn::ObjectSyncMode, [512](#)
 - Arn::SameValue, [523](#)
 - Arn::XStringMapOptions, [552](#)
 - ArnAtomicOp, [107](#)
 - ArnClientConnectStat, [160](#)
 - ArnCoreItem::Heritage, [500](#)
 - ArnDiscover::Type, [529](#)
 - ArnDiscoverAdvertise::State, [525](#)
 - ArnDiscoverInfo::State, [523](#)
 - ArnError, [224](#)
 - ArnError::StdCode, [526](#)
 - ArnEventIdx, [233](#)
 - ArnItemValve::SwitchMode, [527](#)
 - ArnMonEventType, [333](#)
 - ArnQml::UseFlags, [530](#)
 - ArnRpc::Invoke, [503](#)
 - ArnRpcMode, [383](#)
 - ArnScriptJobs::Type, [528](#)
 - ArnServer::Type, [528](#)
 - ArnServerRemoteSessionKillMode, [423](#)
 - ArnZeroConf::Error, [498](#)
 - ArnZeroConf::State, [524](#)
- engine
 - ArnScript, [396](#)
- enumCount
 - Arn::EnumTxt, [481](#)
- EnumTxt
 - Arn::EnumTxt, [475](#)
- enumTxt
 - Arn::_InitEnumTxt, [71](#)
- enumTxtClass
 - Arn::_InitSubEnum, [72](#)

- enumVal
 - Arn::_InitEnumTxt, 71
- error
 - Arn::QmlMFileIO, 515
- errorLog
 - ArnScript, 396
 - ArnScriptJob, 401
 - ArnM, 321
- errorLogSig
 - ArnM, 322
- errorSysName
 - ArnM, 322
- errorText
 - ArnScript, 397
 - ArnScriptJobControl, 405
- evaluate
 - ArnScript, 397
- evaluateFile
 - ArnScript, 397
- eventHandler
 - ArnBasicItem, 115
- EventQuit
 - ArnScriptJob.cpp, 629
- examples/Examples.txt(4.0.0), 559
- exist
 - ArnInterface, 256
 - ArnM, 322
- externalClientConnect
 - ArnDiscoverConnector, 199
- factor
 - Arn::_InitSubEnum, 73
- flagsFromString
 - Arn::EnumTxt, 481
- flagsFromStringList
 - Arn::EnumTxt, 482
- flagsToString
 - Arn::EnumTxt, 482
- flagsToStringList
 - Arn::EnumTxt, 483
- flush
 - ArnPersist, 352
- foundChildDeleted
 - ArnMonitor, 342
- freePaths
 - ArnClient, 148
 - ArnServer, 414
- fromXString
 - Arn::XStringMap, 541
- fullPath
 - Arn, 59
- fullServiceType
 - ArnZeroConfB, 428
- get
 - ArnClientReg, 162
- getAllow
 - ArnServerSession, 425
- getBasicTextList
 - Arn::EnumTxt, 484
- getBitSet
 - Arn::EnumTxt, 484
- getClient
 - ArnClient, 148
- getEnumSet
 - Arn::EnumTxt, 485
- getEnumVal
 - Arn::EnumTxt, 486
- getHeartBeatCheck
 - ArnRpc, 375
- getHeartBeatSend
 - ArnRpc, 375
- getMode
 - ArnAdaptItem, 83
 - ArnBasicItem, 116
 - ArnItem, 273
 - ArnItemQml, 306
- getNextId
 - ArnZeroConfBrowser, 436
- getSubEnumVal
 - Arn::EnumTxt, 487, 488
- getTraffic
 - ArnClient, 150
 - ArnServerSession, 425
- getTxt
 - Arn::EnumTxt, 488
- getTxtRecordMap
 - ArnZeroConfRegister, 454
 - ArnZeroConfResolve, 465
- getTxtString
 - Arn::EnumTxt, 489
- globalProperty
 - ArnScript, 397
- goTowardState
 - ArnDiscoverBrowserB, 189
- groups
 - ArnDiscoverAdvertise, 178
 - ArnDiscoverInfo, 206
- handleData
 - ArnEvValueChange, 248
- heartBeatChanged
 - ArnRpc, 375
- heartBeatCheck
 - ArnSapiQml, 393
- heartBeatReceived
 - ArnRpc, 376
- heartBeatSend
 - ArnSapiQml, 393
- host
 - ArnZeroConfLookup, 444
 - ArnZeroConfRegister, 454
 - ArnZeroConfResolve, 465
- hostAddr
 - ArnZeroConfLookup, 444
- HostAddrPort
 - ArnClient::HostAddrPort, 501
- hostFromHostWithInfo

- Arn, [59](#)
- hostIp
 - ArnDiscoverInfo, [207](#)
- hostIpString
 - ArnDiscoverInfo, [207](#)
- HostList
 - ArnClient, [142](#)
- hostName
 - ArnDiscoverInfo, [207](#)
- hostPort
 - ArnDiscoverInfo, [208](#)
- hostPortString
 - ArnDiscoverInfo, [208](#)
- hostWithInfo
 - ArnDiscoverInfo, [208](#)
- humanize
 - Arn::EnumTxt, [490](#)
- id
 - ArnClient, [150](#)
 - ArnDiscoverConnector, [200](#)
 - ArnScriptJobControl, [406](#)
 - ArnZeroConfLookup, [444](#)
 - ArnZeroConfResolve, [466](#)
- idName
 - ArnScript, [397](#)
- IdToIndex
 - ArnDiscoverBrowserB, [189](#)
- Idx
 - ArnEvent, [229](#)
- ignoreSameValue
 - ArnItemQml, [307](#)
- inProgress
 - ArnDiscoverInfo, [209](#)
- indexOf
 - Arn::XStringMap, [541](#), [542](#)
 - Arn::XStringMapQml, [555](#)
- indexOfValue
 - Arn::XStringMap, [542](#)
 - Arn::XStringMapQml, [555](#)
- indexTold
 - ArnDiscoverBrowserB, [190](#)
- info
 - Arn::XStringMap, [542](#)
 - ArnInterface, [261](#)
 - ArnM, [322](#)
- infoById
 - ArnDiscoverBrowserB, [190](#)
- infoByIndex
 - ArnDiscoverBrowserB, [191](#)
- infoByName
 - ArnDiscoverBrowserB, [191](#)
- infoReceived
 - ArnServerSession, [425](#)
- infoUpdated
 - ArnDiscoverBrowserB, [192](#)
- inhibitPendingChain
 - ArnEvent, [230](#)
- initialServiceTimeout
 - ArnDiscoverRemote, [216](#)
- installExtension
 - ArnScriptJobFactory, [408](#)
- instance
 - ArnClientReg, [162](#)
 - ArnQml, [366](#)
 - ArnM, [323](#)
- intNum
 - ArnInterface, [256](#)
 - ArnItemQml, [308](#)
- interval
 - MQBasicTimer, [507](#)
- invoke
 - ArnRpc, [376](#)
- isArnEvent
 - ArnEvent, [230](#)
- isAssigning
 - ArnBasicItem, [116](#)
- isAtomicOpProvider
 - ArnBasicItem, [117](#)
 - ArnM, [323](#)
- isAutoDestroy
 - ArnAdaptItem, [84](#)
 - ArnBasicItem, [117](#)
 - ArnItem, [273](#)
 - ArnItemValve, [312](#)
 - ArnPipe, [359](#)
- isBelow
 - ArnEvRetired, [246](#)
- isBiDirMode
 - ArnAdaptItem, [84](#)
 - ArnBasicItem, [117](#)
 - ArnItem, [274](#)
- isBrowsing
 - ArnDiscoverBrowser, [185](#)
 - ArnZeroConfBrowser, [437](#)
- isCheckSeq
 - ArnPipe, [359](#)
- isDelayPending
 - ArnItem, [274](#)
- isDemandLogin
 - ArnClient, [150](#)
 - ArnServer, [415](#)
- isDemandLoginNet
 - ArnServer, [415](#)
- isError
 - ArnDiscoverInfo, [209](#)
- isFlag
 - Arn::EnumTxt, [490](#)
- isFolder
 - ArnAdaptItem, [84](#)
 - ArnBasicItem, [118](#)
 - ArnInterface, [257](#)
 - ArnItem, [274](#)
 - ArnM, [323](#)
- isFolderPath
 - Arn, [60](#)
 - ArnInterface, [257](#)

- isForceQtDnsLookup
 - ArnZeroConfLookup, 445
- isGlobal
 - ArnEvRetired, 246
- isHeartBeatOk
 - ArnRpc, 377
 - ArnSapiQml, 392
- isIgnoreSameValue
 - ArnAdaptItem, 85
 - ArnBasicItem, 118
 - ArnItem, 275
- isLastLink
 - ArnEvLinkCreate, 236
- isLeaf
 - ArnInterface, 257
 - ArnM, 324
- isLocal
 - ArnEvMonitor, 241
- isMainThread
 - ArnM, 324
- isMaster
 - ArnAdaptItem, 85
 - ArnBasicItem, 118
 - ArnItem, 275
 - ArnItemValve, 313
 - ArnPipe, 359
- isNullPtr
 - Arn, 60
- isOpen
 - ArnAdaptItem, 85
 - ArnBasicItem, 119
- isPipeMode
 - ArnAdaptItem, 86
 - ArnBasicItem, 119
 - ArnItem, 275
- isPower2
 - Arn, 60
- isProvider
 - ArnAdaptItem, 86
 - ArnBasicItem, 119
 - ArnItem, 276
- isProviderPath
 - Arn, 60
 - ArnInterface, 257
- isReConnect
 - ArnClient, 151
- isReContact
 - ArnClient, 151
- isSaveMode
 - ArnAdaptItem, 86
 - ArnBasicItem, 120
 - ArnItem, 276
 - ArnItemValve, 313
- isSendSeq
 - ArnPipe, 360
- isTemplate
 - ArnItem, 276
- isThreadedApp
 - ArnM, 324
- isUncrossed
 - ArnAdaptItem, 87
 - ArnBasicItem, 120
 - ArnItem, 277
- itemId
 - ArnAdaptItem, 87
 - ArnBasicItem, 120
- itemName
 - Arn, 61
 - ArnInterface, 257
- items
 - ArnInterface, 258
 - ArnM, 325
- key
 - Arn::XStringMap, 542, 543
 - Arn::XStringMapQml, 555
- keyRef
 - Arn::XStringMap, 543
- keyString
 - Arn::XStringMap, 543, 544
- keys
 - Arn::XStringMap, 543
 - Arn::XStringMapQml, 556
- KillMode
 - ArnServerRemoteSession, 422
- killRequested
 - ArnClient, 152
- label
 - MQGenericArgument, 510
- linkDestroyedCallback
 - ArnAdaptItem, 87
- LinkDestroyedCB
 - ArnAdaptItem, 79
- linkId
 - ArnAdaptItem, 88
 - ArnBasicItem, 121
 - ArnEvModeChange, 238
- listenAddress
 - ArnServer, 416
- loadBitSet
 - Arn::EnumTxt, 490, 491
- loadEnumSet
 - Arn::EnumTxt, 492
- loadFromDirRoot
 - ArnM, 325
- loadFromFile
 - ArnM, 325
- loadScriptFile
 - ArnScriptJobControl, 406
- localUpdateCount
 - ArnLinkValue, 317
- log2
 - Arn, 61
- logUncaughtError
 - ArnScript, 398
- loginCompleted

- ArnServerSession, 425
- loginRequired
 - ArnClient, 152
- loginToArn
 - ArnClient, 152
- loginToArnHashed
 - ArnClient, 153
- loginUserName
 - ArnServerSession, 425
- lookup
 - ArnZeroConfLookup, 445
- lookupError
 - ArnZeroConfLookup, 446
- lookupid
 - ArnZeroConfLookup, 445
- MQ_ARG
 - ArnRpc.hpp, 598
- MQ_DECLARE_ENUM_NSTXT
 - MQFlags.hpp, 610
- MQ_DECLARE_ENUMTXT
 - MQFlags.hpp, 610
- MQ_DECLARE_ENUM
 - MQFlagsBase.hpp, 614
- MQ_DECLARE_FLAGS_NSTXT
 - MQFlags.hpp, 610
- MQ_DECLARE_FLAGSTXT
 - MQFlags.hpp, 611
- MQ_DECLARE_FLAGS
 - MQFlagsBase.hpp, 614
- MQ_DECLARE_OPERATORS_FOR_FLAGS
 - MQFlagsBase.hpp, 614
- MQ_DECLARE_SUBETXT
 - MQFlags.hpp, 611
- MQ_NSTXT_FILL_MISSING_FROM
 - MQFlags.hpp, 612
- MQ_NSTXT_FILL_MISSING
 - MQFlags.hpp, 611
- MQ_PUBLIC_ACCESS
 - ArnSapi.hpp, 600
- MQ_SUBETXT_ADD_ABSDEF
 - MQFlags.hpp, 612
- MQ_SUBETXT_ADD_ABSOP
 - MQFlags.hpp, 612
- MQ_SUBETXT_ADD_RELDEF
 - MQFlags.hpp, 612
- MQ_SUBETXT_ADD_RELOP
 - MQFlags.hpp, 612
- MQArgument
 - MQArgument, 506
- MQArgument< T >, 505
- MQBasicTimer, 506
 - interval, 507
 - MQBasicTimer, 507
 - setInterval, 507
 - start, 508
- MQFlags.hpp
 - MQ_DECLARE_ENUM_NSTXT, 610
 - MQ_DECLARE_ENUMTXT, 610
 - MQ_DECLARE_FLAGS_NSTXT, 610
 - MQ_DECLARE_FLAGSTXT, 611
 - MQ_DECLARE_SUBETXT, 611
 - MQ_NSTXT_FILL_MISSING_FROM, 612
 - MQ_NSTXT_FILL_MISSING, 611
 - MQ_SUBETXT_ADD_ABSDEF, 612
 - MQ_SUBETXT_ADD_ABSOP, 612
 - MQ_SUBETXT_ADD_RELDEF, 612
 - MQ_SUBETXT_ADD_RELOP, 612
- MQFlagsBase.hpp
 - MQ_DECLARE_ENUM, 614
 - MQ_DECLARE_FLAGS, 614
 - MQ_DECLARE_OPERATORS_FOR_FLAGS, 614
- MQGenericArgument, 508
 - label, 510
 - MQGenericArgument, 509
- MQVariantMap
 - XStringMap.hpp, 616
- MUTEX_CALL_RET
 - ArnAdaptItem.cpp, 562
- MUTEX_CALL
 - ArnAdaptItem.cpp, 561
- makeHeapClone
 - ArnEvAtomicOp, 227
 - ArnEvLinkCreate, 236
 - ArnEvModeChange, 238
 - ArnEvMonitor, 241
 - ArnEvRefChange, 243
 - ArnEvRetired, 246
 - ArnEvValueChange, 248
 - ArnEvZeroRef, 251
 - ArnEvent, 231
- makeHostWithInfo
 - Arn, 62
- makePath
 - Arn, 62
 - ArnInterface, 258
- mask
 - Arn::InitSubEnum, 73
- masterMode
 - ArnItemQml, 308
- maxEnumOf
 - Arn::XStringMap, 544
- maxLim
 - Arn, 63
- messageReceived
 - ArnServerSession, 425
- methodIdsTab
 - ArnRpc::MethodsParam::Params, 513
- methodPrefix
 - ArnRpc, 378
- minLim
 - Arn, 63
- mod
 - Arn, 63
- Mode
 - ArnRpc, 371
 - ArnSapiQml, 392

- mode
 - ArnEvModeChange, 238
 - ArnRpc, 378
 - ArnSapiQml, 393
- modeChanged
 - ArnItem, 277
- monEvType
 - ArnEvMonitor, 241
- monitorClosed
 - ArnMonitor, 342
- monitorPath
 - ArnMonitor, 343
 - ArnMonitorQml, 348
- mutex
 - ArnAdaptItem, 88
- name
 - Arn::EnumTxt, 493
 - ArnAdaptItem, 88
 - ArnBasicItem, 121
 - ArnScriptJob, 402
 - ArnScriptJobControl, 406
- NameF
 - ArnInterface, 255
- newConnector
 - ArnDiscoverRemote, 216
- no_queue
 - ArnRpc.hpp, 598
- noLoginNets
 - ArnServer, 416
- NS
 - ArnAtomicOp, 107
 - ArnClientConnectStat, 161
 - ArnMonEventType, 334
- ns
 - Arn::_InitEnumTxt, 71
- num
 - ArnInterface, 258
 - ArnItemQml, 308
- numToStr
 - Arn::EnumTxt, 493
- ObjectMode
 - ArnInterface, 255
- offHeartbeat
 - Arn, 68
- Op
 - ArnEvAtomicOp, 226
- op
 - ArnEvAtomicOp, 227
- open
 - ArnAdaptItem, 89
 - ArnBasicItem, 122
 - ArnItemB, 301
 - ArnRpc, 378
 - ArnSapi, 388
- openFolder
 - ArnItem, 278
- openUuid
 - ArnItem, 278
 - ArnPipe, 360
- openUuidPipe
 - ArnItem, 278
- operator T*
 - ArnNullptr, 349
- operator <<
 - ArnItem.cpp, 617
 - ArnItem.hpp, 583
- operator +=
 - Arn::XStringMap, 544
 - ArnAdaptItem, 89
 - ArnBasicItem, 122
 - ArnItem, 279
- operator =
 - Arn::XStringMap, 544
 - ArnAdaptItem, 89–91
 - ArnBasicItem, 122–124
 - ArnDiscoverInfo, 210
 - ArnItem, 279–281
 - ArnItemValve, 313
 - ArnPipe, 360
- Options
 - Arn::XStringMap, 533
- options
 - Arn::XStringMap, 545
- outOfSequence
 - ArnPipe, 361
 - ArnRpc, 378
- paramNames
 - ArnRpc::MethodsParam::Params, 513
- parent
 - Arn::QmlMQtObject, 520
- parentChanged
 - Arn::QmlMQtObject, 520
- parentItem
 - Arn::QmlMQtObject, 520
- parentPath
 - Arn, 63
- passwordHash
 - ArnClient, 154
- path
 - Arn::QmlMFileIO, 515, 516
 - ArnAdaptItem, 91
 - ArnBasicItem, 124
 - ArnEvLinkCreate, 236
 - ArnEvModeChange, 239
 - ArnItemQml, 308
- pathChanged
 - Arn::QmlMFileIO, 515
- pathDiscover
 - Arn, 68
- pathDiscoverConnect
 - Arn, 68
- pathDiscoverThis
 - Arn, 69
- pathLocal
 - Arn, 69

- pathLocalSys
 - Arn, [69](#)
- pathServer
 - Arn, [69](#)
- pathServerSessions
 - Arn, [69](#)
- pipe
 - ArnRpc, [378](#)
- pipeClosed
 - ArnRpc, [379](#)
- pipeMode
 - ArnItemQml, [308](#)
- pipePath
 - ArnRpc, [379](#)
 - ArnSapiQml, [393](#)
- poll
 - ArnScriptJob, [402](#)
- port
 - ArnClient::HostAddrPort, [501](#)
 - ArnServer, [416](#)
 - ArnZeroConfRegister, [455](#)
 - ArnZeroConfResolve, [466](#)
- printFunction
 - ArnScript, [398](#)
- properties
 - ArnDiscoverInfo, [210](#)
- providerPath
 - ArnInterface, [258](#)
- providerPathIf
 - Arn, [64](#)
- QML_ENGINE
 - ArnQml.hpp, [594](#)
- QML_LIST_PROPERTY
 - ArnQml.hpp, [594](#)
- QML_NETACC_FACTORY
 - ArnQml.hpp, [594](#)
- QML_PARSER_STATUS
 - ArnQml.hpp, [595](#)
- QML_QUICK_TYPE
 - ArnQml.hpp, [595](#)
- QML_Qt4
 - ArnQml.hpp, [595](#)
- QmlMFileIO
 - Arn::QmlMFileIO, [514](#)
- QmlMQtObject
 - Arn::QmlMQtObject, [518](#)
- quickTypeRun
 - Arn::QmlMSys, [522](#)
- quit
 - ArnScriptJob, [401](#)
- README.md(4.0.0), [559](#)
- RPC_STORAGE_NAME
 - ArnRpc.cpp, [627](#)
- rand
 - Arn, [64](#)
- rangeLim
 - Arn, [64](#)
- reStart
 - ArnMonitor, [343](#)
 - ArnMonitorQml, [348](#)
- read
 - Arn::QmlMFileIO, [515](#)
- readBytes
 - Arn::QmlMFileIO, [515](#)
- receiveTimeout
 - ArnClient, [154](#)
- receiver
 - ArnRpc, [379](#)
 - ArnSapiQml, [394](#)
- refCount
 - ArnAdaptItem, [92](#)
 - ArnBasicItem, [125](#)
- refStep
 - ArnEvRefChange, [243](#)
- reference
 - ArnAdaptItem, [92](#)
 - ArnBasicItem, [125](#)
 - ArnMonitor, [343](#)
- registerClient
 - ArnClient, [154](#)
- registerService
 - ArnZeroConfRegister, [456](#)
- registered
 - ArnZeroConfRegister, [455](#)
- registrationError
 - ArnZeroConfRegister, [456](#)
- releaseLookup
 - ArnZeroConfLookup, [446](#)
- releaseResolve
 - ArnZeroConfResolve, [466](#)
- releaseService
 - ArnZeroConfRegister, [456](#)
- remoteWhoIAm
 - ArnClient, [155](#)
 - ArnServerSession, [425](#)
- remove
 - Arn::XStringMap, [545](#)
 - Arn::XStringMapQml, [556](#)
 - ArnClientReg, [162](#)
- removeMountPoint
 - ArnClient, [155](#)
- removeValue
 - Arn::XStringMap, [546](#)
 - Arn::XStringMapQml, [556](#)
- resolveCode
 - ArnDiscoverInfo, [210](#)
- resolve
 - ArnDiscoverResolver, [222](#)
 - ArnZeroConfResolve, [466](#)
- resolveError
 - ArnZeroConfResolve, [467](#)
- resolveRefreshTimeout
 - ArnDiscoverConnector, [200](#)
- resolved
 - ArnZeroConfResolve, [467](#)

- resourceArnLib
 - Arn, [69](#)
- resourceArnRoot
 - Arn, [70](#)
- reverseOrder
 - Arn::XStringMap, [546](#)
- rpcSender
 - ArnRpc, [379](#), [380](#)
- running
 - ArnScriptJob, [402](#)
- SameValue
 - ArnInterface, [255](#)
- saveMode
 - ArnItemQml, [309](#)
- saveToFile
 - ArnM, [326](#)
- script
 - ArnScriptJobControl, [406](#)
- scriptChanged
 - ArnScriptJobControl, [406](#)
- sendId
 - ArnEvValueChange, [249](#)
- sendMessage
 - ArnServerSession, [426](#)
- sendText
 - ArnRpc, [380](#)
- service
 - ArnDiscoverAdvertise, [178](#)
 - ArnDiscoverConnector, [200](#)
- serviceAdded
 - ArnDiscoverBrowserB, [192](#)
 - ArnZeroConfBrowser, [437](#)
- serviceChangeError
 - ArnDiscoverAdvertise, [179](#)
- serviceChanged
 - ArnDiscoverAdvertise, [179](#)
 - ArnZeroConfBrowser, [438](#)
- serviceCount
 - ArnDiscoverBrowserB, [193](#)
- serviceName
 - ArnDiscoverInfo, [210](#)
 - ArnZeroConfRegister, [457](#)
 - ArnZeroConfResolve, [468](#)
- serviceNameTold
 - ArnDiscoverBrowserB, [193](#)
 - ArnZeroConfBrowser, [438](#)
- serviceRemoved
 - ArnDiscoverBrowserB, [194](#)
 - ArnZeroConfBrowser, [439](#)
- serviceType
 - ArnZeroConfB, [428](#)
- sessionHandler
 - ArnEvMonitor, [241](#)
- set
 - Arn::XStringMap, [546](#)–[548](#)
 - Arn::XStringMapQml, [556](#), [557](#)
- setArchiveDir
 - ArnPersist, [352](#)
- setArnEventCallback
 - ArnAdaptItem, [92](#)
- setArnRootPath
 - ArnQml, [367](#)
- setAtomicOpProvider
 - ArnBasicItem, [125](#)
 - ArnM, [326](#)
- setAutoConnect
 - ArnClient, [156](#)
- setAutoDestroy
 - ArnAdaptItem, [93](#)
 - ArnBasicItem, [125](#)
 - ArnItem, [281](#)
 - ArnItemValve, [314](#)
 - ArnPipe, [361](#)
- setBiDirMode
 - ArnAdaptItem, [93](#)
 - ArnBasicItem, [126](#)
 - ArnItem, [281](#)
- setBits
 - ArnAdaptItem, [93](#)
 - ArnBasicItem, [126](#)
 - ArnItem, [281](#)
 - ArnItemQml, [306](#)
- setBlockEcho
 - ArnItem, [282](#)
- setBytes
 - ArnInterface, [258](#)
- setChangedCallback
 - ArnAdaptItem, [94](#)
- setCheckSeq
 - ArnPipe, [361](#)
- setClient
 - ArnMonitor, [343](#), [344](#)
- setConfig
 - ArnScriptJobControl, [406](#)
- setConsoleError
 - ArnM, [327](#)
- setCustomProperties
 - ArnDiscoverAdvertise, [180](#)
- setDefaultIgnoreSameValue
 - ArnM, [327](#)
- setDefaultPath
 - ArnSapi, [389](#)
- setDefaultService
 - ArnDiscoverRemote, [217](#)
 - ArnDiscoverResolver, [222](#)
- setDefaultStopState
 - ArnDiscoverBrowserB, [194](#)
- setDelay
 - ArnItem, [282](#)
- setDemandLogin
 - ArnClient, [156](#)
 - ArnServer, [417](#)
- setDirectHostPrio
 - ArnDiscoverConnector, [201](#)
- setDiscoverHostPrio
 - ArnDiscoverConnector, [201](#)

- setDomain
 - ArnZeroConfB, 429
- setEmptyKeysToValue
 - Arn::XStringMap, 548
 - Arn::XStringMapQml, 557
- setEventHandler
 - ArnBasicItem, 127
- setExternalClientConnect
 - ArnDiscoverConnector, 202
- setFactory
 - ArnScriptJobs, 411
- setFilter
 - ArnDiscoverBrowser, 185, 186
- setForceQtDnsLookup
 - ArnZeroConfLookup, 446
- setGroups
 - ArnDiscoverAdvertise, 180
- setHeartBeatCheck
 - ArnRpc, 380
- setHeartBeatSend
 - ArnRpc, 381
- setHost
 - ArnZeroConfLookup, 447
 - ArnZeroConfRegister, 457
- setId
 - ArnZeroConfLookup, 447
 - ArnZeroConfResolve, 468
- setIgnoreSameValue
 - ArnAdaptItem, 94
 - ArnBasicItem, 127
 - ArnItem, 283
- setIncludeSender
 - ArnRpc, 381
- setInitialServiceTimeout
 - ArnDiscoverRemote, 217
- setIntNum
 - ArnInterface, 259
- setInterruptedText
 - ArnScript, 398
- setInterval
 - MQBasicTimer, 507
- setKey
 - Arn::XStringMap, 548
- setLinkDestroyedCallback
 - ArnAdaptItem, 95
- setMaster
 - ArnAdaptItem, 95
 - ArnBasicItem, 127
 - ArnItem, 283
 - ArnItemValve, 314
 - ArnPipe, 362
- setMethodPrefix
 - ArnRpc, 381
- setMissingTxt
 - Arn::EnumTxt, 493
- setMode
 - ArnRpc, 381
- setMonitorName
 - ArnDepend, 168
- setMonitorPath
 - ArnMonitor, 344
- setMountPoint
 - ArnClient, 157
 - ArnPersist, 353
- setName
 - ArnScriptJobControl, 407
- setNoLoginNets
 - ArnServer, 417
- setNum
 - ArnInterface, 259
- setOptions
 - Arn::XStringMap, 548
- setParentItem
 - Arn::QmlMQtObject, 520
- setPath
 - Arn::QmlMFileIO, 515
- setPersistDir
 - ArnPersist, 353
- setPipe
 - ArnRpc, 382
- setPipeMode
 - ArnAdaptItem, 95
 - ArnBasicItem, 128
 - ArnItem, 283
- setPort
 - ArnZeroConfRegister, 458
- setReceiveTimeout
 - ArnClient, 157
- setReceiver
 - ArnRpc, 382
- setReference
 - ArnAdaptItem, 96
 - ArnBasicItem, 128
 - ArnMonitor, 344
- setResolveRefreshTimeout
 - ArnDiscoverConnector, 203
- setResolver
 - ArnDiscoverConnector, 202
- setSaveMode
 - ArnAdaptItem, 96
 - ArnBasicItem, 129
 - ArnItem, 284
 - ArnItemValve, 314
- setScript
 - ArnScriptJobControl, 407
- setSendSeq
 - ArnPipe, 362
- setService
 - ArnDiscoverAdvertise, 181
 - ArnDiscoverConnector, 203
 - ArnDiscoverRemote, 218
- setServiceName
 - ArnZeroConfRegister, 458
 - ArnZeroConfResolve, 468
- setServiceType
 - ArnZeroConfB, 429

- setSkipLocalSysLoading
 - ArnM, 327
- setSocketType
 - ArnZeroConfB, 430
- setStateId
 - ArnDependOffer, 172
- setStateName
 - ArnDependOffer, 172
- setString
 - ArnInterface, 259
- setSubType
 - ArnZeroConfBrowser, 439
- setSubTypes
 - ArnZeroConfRegister, 458
- setSyncMode
 - ArnClient, 158
- setTarget
 - ArnEvent, 231
 - ArnItemValve, 315
- setTargetMutex
 - ArnEvent, 231
- setTargetPendingChain
 - ArnEvent, 231
- setTemplate
 - ArnItem, 284
- setThreaded
 - ArnScriptJobControl, 407
- setTxt
 - Arn::EnumTxt, 494
- setTxtRecord
 - ArnZeroConfRegister, 459
- setTxtRecordMap
 - ArnZeroConfRegister, 459
- setTxtRef
 - Arn::EnumTxt, 494
- setTxtString
 - Arn::EnumTxt, 495
- setUncrossed
 - ArnAdaptItem, 97
 - ArnBasicItem, 129
 - ArnItem, 285
- setValue
 - ArnAdaptItem, 97–101
 - ArnBasicItem, 129–133
 - ArnInterface, 259
 - ArnItem, 285–287, 289–293
 - ArnItemValve, 315
 - ArnPipe, 362
 - ArnM, 328–330
- setValueOverwrite
 - ArnPipe, 363
- setVariant
 - ArnInterface, 260
- setVcs
 - ArnPersist, 354
- setWatchDogTime
 - ArnScriptJob, 401
- setWhoIAm
 - ArnClient, 158
 - ArnServer, 418
- setup
 - ArnQml, 367
- setupDataBase
 - ArnPersist, 354
- setupErrorlog
 - ArnM, 328
- setupInterface
 - ArnScriptJobFactory, 408
- setupJsObj
 - ArnScriptJobFactory, 409
- sigQuit
 - ArnScriptJob, 402
- size
 - Arn::XStringMap, 548
 - Arn::XStringMapQml, 558
- skipLocalSysLoading
 - ArnM, 330
- sleepState
 - ArnScriptJob, 402
- socket
 - ArnServerSession, 426
- socketType
 - ArnZeroConfB, 430
- squeeze
 - Arn::XStringMap, 549
- src/Arn.cpp(4.0.0), 559
- src/ArnAdaptItem.cpp(4.0.0), 561
- src/ArnBasicItem.cpp(4.0.0), 562
- src/ArnClient.cpp(4.0.0), 563
- src/ArnCompat.cpp(4.0.0), 563
- src/ArnCoreItem.cpp(4.0.0), 564
- src/ArnDepend.cpp(4.0.0), 564
- src/ArnDiscover.cpp(4.0.0), 565
- src/ArnDiscoverConnect.cpp(4.0.0), 566
- src/ArnDiscoverRemote.cpp(4.0.0), 566
- src/ArnEvent.cpp(4.0.0), 566
- src/ArnInc/Arn.hpp(4.0.0), 567
- src/ArnInc/ArnAdaptItem.hpp(4.0.0), 570
- src/ArnInc/ArnBasicItem.hpp(4.0.0), 571
- src/ArnInc/ArnClient.hpp(4.0.0), 572
- src/ArnInc/ArnCompat.hpp(4.0.0), 572
- src/ArnInc/ArnCoreItem.hpp(4.0.0), 575
- src/ArnInc/ArnDepend.hpp(4.0.0), 575
- src/ArnInc/ArnDiscover.hpp(4.0.0), 576
- src/ArnInc/ArnDiscoverConnect.hpp(4.0.0), 578
- src/ArnInc/ArnDiscoverRemote.hpp(4.0.0), 579
- src/ArnInc/ArnError.hpp(4.0.0), 579
- src/ArnInc/ArnEvent.hpp(4.0.0), 580
- src/ArnInc/ArnInterface.hpp(4.0.0), 581
- src/ArnInc/ArnItem.hpp(4.0.0), 582
- src/ArnInc/ArnItemB.hpp(4.0.0), 583
- src/ArnInc/ArnItemValve.hpp(4.0.0), 584
- src/ArnInc/ArnLib.hpp(4.0.0), 585
- src/ArnInc/ArnLib_global.hpp(4.0.0), 586
- src/ArnInc/ArnLinkHandle.hpp(4.0.0), 587
- src/ArnInc/ArnM.hpp(4.0.0), 587

- src/ArnInc/ArnMonEvent.hpp(4.0.0), 588
- src/ArnInc/ArnMonitor.hpp(4.0.0), 589
- src/ArnInc/ArnPersist.hpp(4.0.0), 590
- src/ArnInc/ArnPersistSapi.hpp(4.0.0), 591
- src/ArnInc/ArnPipe.hpp(4.0.0), 592
- src/ArnInc/ArnQml.hpp(4.0.0), 593
- src/ArnInc/ArnQmlMQT.hpp(4.0.0), 595
- src/ArnInc/ArnQmlMSystem.hpp(4.0.0), 596
- src/ArnInc/ArnRpc.hpp(4.0.0), 597
- src/ArnInc/ArnSapi.hpp(4.0.0), 599
- src/ArnInc/ArnScript.hpp(4.0.0), 600
- src/ArnInc/ArnScriptJob.hpp(4.0.0), 602
- src/ArnInc/ArnScriptJobs.hpp(4.0.0), 603
- src/ArnInc/ArnServer.hpp(4.0.0), 604
- src/ArnInc/ArnServerRemote.hpp(4.0.0), 605
- src/ArnInc/ArnZeroConf.hpp(4.0.0), 606
- src/ArnInc/MQFlags.hpp(4.0.0), 609
- src/ArnInc/MQFlagsBase.hpp(4.0.0), 613
- src/ArnInc/Math.hpp(4.0.0), 608
- src/ArnInc/XStringMap.hpp(4.0.0), 615
- src/ArnItem.cpp(4.0.0), 616
- src/ArnItemB.cpp(4.0.0), 617
- src/ArnItemNet.cpp(4.0.0), 617
- src/ArnItemNet.hpp(4.0.0), 618
- src/ArnItemValve.cpp(4.0.0), 618
- src/ArnLib.cpp(4.0.0), 619
- src/ArnLink.cpp(4.0.0), 620
- src/ArnLink.hpp(4.0.0), 620
- src/ArnLinkHandle.cpp(4.0.0), 622
- src/ArnM.cpp(4.0.0), 622
- src/ArnMath.cpp(4.0.0), 623
- src/ArnMonitor.cpp(4.0.0), 624
- src/ArnPersist.cpp(4.0.0), 624
- src/ArnPipe.cpp(4.0.0), 625
- src/ArnQml.cpp(4.0.0), 625
- src/ArnQmlMQT.cpp(4.0.0), 626
- src/ArnQmlMSystem.cpp(4.0.0), 626
- src/ArnRpc.cpp(4.0.0), 627
- src/ArnSapi.cpp(4.0.0), 627
- src/ArnScript.cpp(4.0.0), 628
- src/ArnScriptJob.cpp(4.0.0), 628
- src/ArnScriptJobs.cpp(4.0.0), 629
- src/ArnServer.cpp(4.0.0), 630
- src/ArnServerRemote.cpp(4.0.0), 630
- src/ArnSync.cpp(4.0.0), 631
- src/ArnSync.hpp(4.0.0), 631
- src/ArnSyncLogin.cpp(4.0.0), 633
- src/ArnSyncLogin.hpp(4.0.0), 633
- src/ArnXStringMap.cpp(4.0.0), 634
- src/ArnZeroConf.cpp(4.0.0), 634
- src/MQFlags.cpp(4.0.0), 635
- start
 - ArnDiscoverConnector, 204
 - ArnMonitor, 345
 - ArnScriptJobs, 411
 - ArnServer, 418
 - MQBasicTimer, 508
- startLink
 - ArnEvRetired, 246
- startMonitor
 - ArnDepend, 168
- startUseNewServer
 - ArnDiscoverRemote, 218
- startUseServer
 - ArnDiscoverRemote, 219
 - ArnServerRemote, 420
- state
 - ArnDiscoverAdvertise, 181
 - ArnDiscoverInfo, 211
 - ArnZeroConfB, 430
- stateld
 - ArnDependOffer, 172
- stateName
 - ArnDependOffer, 173
- stopBrowse
 - ArnDiscoverBrowser, 186
 - ArnZeroConfBrowser, 440
- stopState
 - ArnDiscoverInfo, 211
- store
 - ArnClientReg, 162
- strToBitpos
 - Arn::EnumTxt, 495
- strToNum
 - Arn::EnumTxt, 495
- string
 - ArnInterface, 260
 - ArnItemQml, 309
- stringCode
 - Arn::XStringMap, 549
- stringDecode
 - Arn::XStringMap, 549
- subEnumAt
 - Arn::EnumTxt, 495
- subEnumCount
 - Arn::EnumTxt, 496
- subEnumNameAt
 - Arn::EnumTxt, 496
- subEnumPropAt
 - Arn::EnumTxt, 497
- subType
 - ArnZeroConfBrowser, 440
- subTypes
 - ArnZeroConfRegister, 460
- switchMode
 - ArnItemValve, 315
- SyncMode
 - ArnClient, 142
- syncMode
 - ArnAdaptItem, 102
 - ArnBasicItem, 134
 - ArnClient, 159
 - ArnItem, 294
- TO_IDX_RETVAL
 - ArnEvent.cpp, 567
- target

- ArnEvent, [231](#)
- tcpConnected
 - ArnClient, [159](#)
- tcpDisConnected
 - ArnClient, [159](#)
- tcpError
 - ArnClient, [160](#)
- textReceived
 - ArnRpc, [382](#)
- thread
 - ArnAdaptItem, [102](#)
 - ArnBasicItem, [134](#)
 - ArnCoreItem, [164](#)
- toBool
 - ArnAdaptItem, [102](#)
 - ArnBasicItem, [134](#)
 - ArnItem, [294](#)
 - ArnItemValve, [316](#)
- toByteArray
 - ArnAdaptItem, [103](#)
 - ArnBasicItem, [135](#)
 - ArnItem, [294](#)
- toDouble
 - ArnAdaptItem, [103](#)
 - ArnBasicItem, [135](#)
 - ArnItem, [295](#)
- toldx
 - ArnEvent, [231](#), [232](#)
- toInt
 - ArnAdaptItem, [103](#)
 - ArnBasicItem, [135](#)
 - ArnItem, [295](#)
- toInt64
 - ArnAdaptItem, [104](#)
 - ArnBasicItem, [136](#)
 - ArnItem, [296](#)
- toMap
 - Arn::XStringMapQml, [557](#)
- toReal
 - ArnAdaptItem, [104](#)
 - ArnBasicItem, [136](#)
 - ArnItem, [296](#)
- toString
 - ArnAdaptItem, [105](#)
 - ArnBasicItem, [137](#)
 - ArnEvent, [232](#)
 - ArnItem, [296](#)
- toUInt
 - ArnAdaptItem, [105](#)
 - ArnBasicItem, [137](#)
 - ArnItem, [297](#)
- toUInt64
 - ArnAdaptItem, [105](#)
 - ArnBasicItem, [137](#)
 - ArnItem, [297](#)
- toVariant
 - ArnAdaptItem, [106](#)
 - ArnBasicItem, [138](#)
- ArnItem, [298](#)
- toVariantMap
 - Arn::XStringMap, [549](#)
- toXString
 - Arn::XStringMap, [549](#)
- toXStringString
 - Arn::XStringMap, [549](#)
- toggleBool
 - ArnItem, [295](#)
- twinPath
 - Arn, [65](#)
 - ArnInterface, [260](#)
- txtRecord
 - ArnZeroConfRegister, [460](#)
 - ArnZeroConfResolve, [469](#)
- type
 - ArnAdaptItem, [106](#)
 - ArnBasicItem, [138](#)
 - ArnDiscoverInfo, [211](#)
 - ArnEvAtomicOp, [227](#)
 - ArnEvLinkCreate, [236](#)
 - ArnEvModeChange, [239](#)
 - ArnEvMonitor, [241](#)
 - ArnEvRefChange, [244](#)
 - ArnEvRetired, [246](#)
 - ArnEvValueChange, [249](#)
 - ArnEvZeroRef, [251](#)
 - ArnItem, [298](#)
 - ArnItemQml, [309](#)
- typeString
 - ArnDiscoverInfo, [212](#)
- useUuid
 - ArnItemQml, [309](#)
- uuidPath
 - Arn, [65](#)
- value
 - Arn::XStringMap, [550](#)
 - Arn::XStringMapQml, [557](#)
 - ArnInterface, [260](#)
- valueByteArray
 - ArnLinkValue, [317](#)
 - ArnM, [330](#)
- valueData
 - ArnEvValueChange, [249](#)
- valueDouble
 - ArnM, [331](#)
- valueInt
 - ArnLinkValue, [317](#)
 - ArnM, [331](#)
- valueReal
 - ArnLinkValue, [317](#)
 - ArnM, [331](#)
- valueRef
 - Arn::XStringMap, [551](#)
- valueString
 - Arn::XStringMap, [551](#), [552](#)
 - ArnLinkValue, [317](#)

- ArnM, [332](#)
- valueVariant
 - ArnLinkValue, [317](#)
 - ArnM, [332](#)
- values
 - Arn::XStringMap, [551](#)
 - Arn::XStringMapQml, [558](#)
- variant
 - ArnInterface, [261](#)
 - ArnItemQml, [309](#)
- variantType
 - ArnItemQml, [310](#)
- warningMDNS
 - Arn, [70](#)
- watchDog
 - ArnScriptJob, [403](#)
- write
 - Arn::QmIMFileIO, [516](#)
- writeBytes
 - Arn::QmIMFileIO, [516](#)
- XStringMap
 - Arn::XStringMap, [534](#)
- XStringMap.hpp
 - ARNXSTRINGMAP_VER, [616](#)
 - MQVariantMap, [616](#)
- xstring
 - Arn::XStringMapQml, [558](#)
- xstringToEnum
 - Arn::QmIMSys, [522](#)
- yield
 - ArnScriptJob, [402](#)