

ArnLib

3.0.x

Generated by Doxygen 1.8.8

Wed May 11 2016 22:22:54

Contents

1	README	1
2	General Description	5
2.1	Arn Data Objects	5
2.1.1	ArnItem access	5
2.1.2	Modes	6
2.1.3	Local	6
2.1.4	Naming conventions	6
2.1.5	Bidirectional Arn Data Objects	7
2.1.6	Pipe Arn Data Objects	7
2.1.6.1	Pipe sequence check	7
2.1.6.2	Pipe anti congest	8
2.1.7	Persistent Arn Data Objects	8
2.1.7.1	Saving objects in files	9
2.1.8	Sharing Arn Data Objects	9
2.1.8.1	Dynamic port	9
2.2	RPC and SAPI	10
2.2.1	RPC and SAPI method name overload	10
2.2.2	RPC and SAPI communication format	11
2.3	ZeroConfig	13
2.3.1	ZeroConfig definitions	13
2.3.1.1	Service name	13
2.3.1.2	Sub types	13
2.3.1.3	Text record	13
2.3.2	Discover	14
2.3.3	Discover remote	14
2.4	Application notations	15
3	Installation and usage	17
3.1	Introduction	17
3.2	Documentation	17
3.3	Building ArnLib	18

3.3.1	A) Unix	18
3.3.2	B) Win32/MSVC	18
3.3.3	C) Win32/MinGW	18
3.3.4	D) MacOSX	19
3.3.5	E) Qt Embedded	19
3.4	Using ArnLib	19
4	ArnLib Internals	21
4.1	ScriptJobs	21
4.2	ArnMonitor	22
4.3	Destroy	22
5	ArnLib Todo	25
6	Example Collection	27
6.1	Chat Demo	27
6.1.1	Chat Server	27
6.1.1.1	ChatSapi.hpp	27
6.1.1.2	MainWindow.hpp	27
6.1.1.3	MainWindow.cpp	28
6.1.1.4	main.cpp	30
6.1.2	Chat Client	30
6.1.2.1	MainWindow.hpp	30
6.1.2.2	MainWindow.cpp	30
6.1.2.3	main.cpp	31
6.1.3	Pictures	32
7	Help descriptions	33
7.1	Discover	33
7.1.1	Description	33
8	Deprecated List	35
9	Namespace Index	37
9.1	Namespace List	37
10	Hierarchical Index	39
10.1	Class Hierarchy	39
11	Class Index	43
11.1	Class List	43
12	File Index	47
12.1	File List	47

13 Namespace Documentation	49
13.1 Arn Namespace Reference	49
13.1.1 Function Documentation	51
13.1.1.1 addPath	51
13.1.1.2 changeBasePath	51
13.1.1.3 childPath	51
13.1.1.4 convertName	52
13.1.1.5 convertPath	52
13.1.1.6 fullPath	52
13.1.1.7 hostFromHostWithInfo	53
13.1.1.8 isFolderPath	53
13.1.1.9 isPower2	53
13.1.1.10 isProviderPath	53
13.1.1.11 itemName	54
13.1.1.12 makeHostWithInfo	54
13.1.1.13 makePath	54
13.1.1.14 parentPath	55
13.1.1.15 providerPathIf	55
13.1.1.16 twinPath	55
13.1.1.17 uuidPath	56
13.1.2 Variable Documentation	56
13.1.2.1 debugDepend	56
13.1.2.2 debugDiscover	56
13.1.2.3 debugLinkDestroy	56
13.1.2.4 debugLinkRef	56
13.1.2.5 debugMDNS	56
13.1.2.6 debugMonitor	56
13.1.2.7 debugMonitorTest	57
13.1.2.8 debugQmlNetwork	57
13.1.2.9 debugReclnOut	57
13.1.2.10 debugRPC	57
13.1.2.11 debugShareObj	57
13.1.2.12 debugSizes	57
13.1.2.13 debugThreading	57
13.1.2.14 debugZeroConf	57
13.1.2.15 defaultTcpPort	57
13.1.2.16 pathDiscover	57
13.1.2.17 pathDiscoverConnect	57
13.1.2.18 pathDiscoverThis	57
13.1.2.19 pathLocal	58

13.1.2.20 pathLocalSys	58
13.1.2.21 pathServer	58
13.1.2.22 pathServerSessions	58
13.1.2.23 resourceArnLib	58
13.1.2.24 resourceArnRoot	58
13.1.2.25 warningMDNS	58
13.2 ArnDiscover Namespace Reference	58
13.3 ArnZeroConf Namespace Reference	58
14 Class Documentation	59
14.1 Arn::_InitEnumTxt Struct Reference	59
14.1.1 Detailed Description	59
14.1.2 Member Data Documentation	59
14.1.2.1 enumTxt	59
14.1.2.2 enumVal	59
14.1.2.3 ns	59
14.2 Arn::Allow Class Reference	59
14.2.1 Detailed Description	60
14.2.2 Member Enumeration Documentation	60
14.2.2.1 E	60
14.3 ArnBasicItem Class Reference	60
14.3.1 Detailed Description	63
14.3.2 Constructor & Destructor Documentation	64
14.3.2.1 ArnBasicItem	64
14.3.2.2 ~ArnBasicItem	64
14.3.3 Member Function Documentation	64
14.3.3.1 addMode	64
14.3.3.2 arnExport	64
14.3.3.3 arnImport	65
14.3.3.4 close	65
14.3.3.5 destroyLink	65
14.3.3.6 destroyLinkLocal	65
14.3.3.7 eventHandler	66
14.3.3.8 getMode	66
14.3.3.9 isAutoDestroy	66
14.3.3.10 isBiDirMode	66
14.3.3.11 isFolder	67
14.3.3.12 isIgnoreSameValue	67
14.3.3.13 isMaster	67
14.3.3.14 isOpen	67

14.3.3.15 isPipeMode	67
14.3.3.16 isProvider	68
14.3.3.17 isSaveMode	68
14.3.3.18 itemId	68
14.3.3.19 linkId	69
14.3.3.20 name	69
14.3.3.21 open	69
14.3.3.22 operator=	69
14.3.3.23 operator=	69
14.3.3.24 operator=	69
14.3.3.25 operator=	70
14.3.3.26 operator=	70
14.3.3.27 operator=	70
14.3.3.28 operator=	70
14.3.3.29 operator=	70
14.3.3.30 operator=	70
14.3.3.31 operator=	70
14.3.3.32 path	70
14.3.3.33 refCount	70
14.3.3.34 reference	71
14.3.3.35 setAutoDestroy	71
14.3.3.36 setBiDirMode	71
14.3.3.37 setEventHandler	71
14.3.3.38 setIgnoreSameValue	71
14.3.3.39 setMaster	72
14.3.3.40 setPipeMode	72
14.3.3.41 setReference	72
14.3.3.42 setSaveMode	72
14.3.3.43 setValue	73
14.3.3.44 setValue	73
14.3.3.45 setValue	73
14.3.3.46 setValue	73
14.3.3.47 setValue	74
14.3.3.48 setValue	74
14.3.3.49 setValue	74
14.3.3.50 setValue	74
14.3.3.51 setValue	75
14.3.3.52 setValue	75
14.3.3.53 setValue	75
14.3.3.54 syncMode	76

14.3.3.55 thread	76
14.3.3.56 toBool	76
14.3.3.57 toByteArray	77
14.3.3.58 toDouble	77
14.3.3.59 toInt	77
14.3.3.60 toInt64	77
14.3.3.61 toReal	78
14.3.3.62 toString	78
14.3.3.63 toUInt	78
14.3.3.64 toUInt64	78
14.3.3.65 toVariant	79
14.3.3.66 type	79
14.3.4 Friends And Related Function Documentation	79
14.3.4.1 ArnBasicItemEventHandler	79
14.4 ArnClient Class Reference	79
14.4.1 Detailed Description	82
14.4.2 Member Typedef Documentation	82
14.4.2.1 ConnectStat	82
14.4.2.2 HostList	83
14.4.3 Constructor & Destructor Documentation	83
14.4.3.1 ArnClient	83
14.4.3.2 ~ArnClient	83
14.4.4 Member Function Documentation	83
14.4.4.1 abortKillRequest	83
14.4.4.2 addMountPoint	83
14.4.4.3 addToArnList	83
14.4.4.4 arnList	84
14.4.4.5 chatReceived	84
14.4.4.6 chatSend	84
14.4.4.7 clearArnList	85
14.4.4.8 close	85
14.4.4.9 connectionStatusChanged	85
14.4.4.10 connectStatus	85
14.4.4.11 connectToArn	86
14.4.4.12 connectToArnList	86
14.4.4.13 disconnectFromArn	86
14.4.4.14 freePaths	86
14.4.4.15 getClient	87
14.4.4.16 getTraffic	87
14.4.4.17 id	87

14.4.4.18 isDemandLogin	87
14.4.4.19 isReConnect	88
14.4.4.20 isReContact	88
14.4.4.21 killRequested	88
14.4.4.22 loginRequired	89
14.4.4.23 loginToArn	90
14.4.4.24 loginToArnHashed	90
14.4.4.25 passwordHash	90
14.4.4.26 receiveTimeout	91
14.4.4.27 registerClient	91
14.4.4.28 remoteWholAm	91
14.4.4.29 removeMountPoint	92
14.4.4.30 setAutoConnect	92
14.4.4.31 setDemandLogin	92
14.4.4.32 setMountPoint	92
14.4.4.33 setReceiveTimeout	93
14.4.4.34 setWholAm	93
14.4.4.35 tcpConnected	94
14.4.4.36 tcpDisConnected	94
14.4.4.37 tcpError	94
14.5 ArnClientConnectStat Class Reference	94
14.5.1 Detailed Description	94
14.5.2 Member Enumeration Documentation	95
14.5.2.1 E	95
14.5.2.2 NS	95
14.6 ArnClientReg Class Reference	95
14.6.1 Detailed Description	95
14.6.2 Member Function Documentation	95
14.6.2.1 get	95
14.6.2.2 instance	96
14.6.2.3 remove	96
14.6.2.4 remove	96
14.6.2.5 store	96
14.7 ArnDepend Class Reference	96
14.7.1 Detailed Description	97
14.7.2 Member Typedef Documentation	98
14.7.2.1 DepSlot	98
14.7.3 Constructor & Destructor Documentation	98
14.7.3.1 ArnDepend	98
14.7.3.2 ~ArnDepend	98

14.7.4	Member Function Documentation	98
14.7.4.1	add	98
14.7.4.2	add	98
14.7.4.3	completed	98
14.7.4.4	setMonitorName	99
14.7.4.5	startMonitor	100
14.8	ArnDependOffer Class Reference	100
14.8.1	Detailed Description	101
14.8.2	Constructor & Destructor Documentation	101
14.8.2.1	ArnDependOffer	101
14.8.2.2	~ArnDependOffer	101
14.8.3	Member Function Documentation	101
14.8.3.1	advertise	101
14.8.3.2	setStateId	101
14.8.3.3	setStateName	102
14.8.3.4	statId	102
14.8.3.5	stateName	102
14.9	ArnDiscoverAdvertise Class Reference	102
14.9.1	Detailed Description	104
14.9.2	Constructor & Destructor Documentation	104
14.9.2.1	ArnDiscoverAdvertise	104
14.9.2.2	~ArnDiscoverAdvertise	105
14.9.3	Member Function Documentation	105
14.9.3.1	addCustomProperty	105
14.9.3.2	addGroup	105
14.9.3.3	advertiseService	105
14.9.3.4	currentService	106
14.9.3.5	customProperties	106
14.9.3.6	groups	106
14.9.3.7	service	107
14.9.3.8	serviceChanged	107
14.9.3.9	serviceChangeError	107
14.9.3.10	setCustomProperties	107
14.9.3.11	setGroups	108
14.9.3.12	setService	108
14.9.3.13	state	109
14.10	ArnDiscoverBrowser Class Reference	109
14.10.1	Detailed Description	110
14.10.2	Constructor & Destructor Documentation	111
14.10.2.1	ArnDiscoverBrowser	111

14.10.3 Member Function Documentation	111
14.10.3.1 browse	111
14.10.3.2 isBrowsing	111
14.10.3.3 setFilter	112
14.10.3.4 setFilter	112
14.10.3.5 stopBrowse	112
14.11 ArnDiscoverBrowserB Class Reference	113
14.11.1 Detailed Description	114
14.11.2 Constructor & Destructor Documentation	114
14.11.2.1 ArnDiscoverBrowserB	114
14.11.2.2 ~ArnDiscoverBrowserB	114
14.11.3 Member Function Documentation	114
14.11.3.1 defaultStopState	114
14.11.3.2 goTowardState	115
14.11.3.3 IdToIndex	115
14.11.3.4 indexTold	115
14.11.3.5 infoById	116
14.11.3.6 infoByIndex	116
14.11.3.7 infoByName	117
14.11.3.8 infoUpdated	117
14.11.3.9 serviceAdded	117
14.11.3.10 serviceCount	117
14.11.3.11 serviceNameTold	118
14.11.3.12 serviceRemoved	118
14.11.3.13 setDefaultStopState	118
14.12 ArnDiscoverConnector Class Reference	119
14.12.1 Detailed Description	120
14.12.2 Constructor & Destructor Documentation	121
14.12.2.1 ArnDiscoverConnector	121
14.12.2.2 ~ArnDiscoverConnector	121
14.12.3 Member Function Documentation	121
14.12.3.1 addToDirectHosts	121
14.12.3.2 clearDirectHosts	121
14.12.3.3 clientReadyToConnect	121
14.12.3.4 directHostPrio	122
14.12.3.5 discoverHostPrio	122
14.12.3.6 externalClientConnect	122
14.12.3.7 id	123
14.12.3.8 resolveRefreshTimeout	123
14.12.3.9 service	123

14.12.3.10	setDirectHostPrio	123
14.12.3.11	setDiscoverHostPrio	124
14.12.3.12	setExternalClientConnect	124
14.12.3.13	setResolver	124
14.12.3.14	setResolveRefreshTimeout	125
14.12.3.15	setService	125
14.12.3.16	start	125
14.13	ArnDiscoverInfo Class Reference	126
14.13.1	Detailed Description	127
14.13.2	Constructor & Destructor Documentation	127
14.13.2.1	ArnDiscoverInfo	127
14.13.2.2	ArnDiscoverInfo	127
14.13.2.3	~ArnDiscoverInfo	127
14.13.3	Member Function Documentation	127
14.13.3.1	domain	127
14.13.3.2	groups	127
14.13.3.3	hostIp	128
14.13.3.4	hostIpString	128
14.13.3.5	hostName	128
14.13.3.6	hostPort	128
14.13.3.7	hostPortString	129
14.13.3.8	hostWithInfo	129
14.13.3.9	inProgress	129
14.13.3.10	isError	129
14.13.3.11	operator=	129
14.13.3.12	properties	130
14.13.3.13	resolveCode	130
14.13.3.14	serviceName	130
14.13.3.15	state	130
14.13.3.16	stopState	131
14.13.3.17	type	131
14.13.3.18	typeString	131
14.13.4	Friends And Related Function Documentation	131
14.13.4.1	ArnDiscoverBrowserB	131
14.14	ArnDiscoverRemote Class Reference	132
14.14.1	Detailed Description	133
14.14.2	Constructor & Destructor Documentation	134
14.14.2.1	ArnDiscoverRemote	134
14.14.2.2	~ArnDiscoverRemote	134
14.14.3	Member Function Documentation	134

14.14.3.1 clientReadyToConnect	134
14.14.3.2 defaultService	134
14.14.3.3 initialServiceTimeout	135
14.14.3.4 newConnector	135
14.14.3.5 setDefaultService	135
14.14.3.6 setInitialServiceTimeout	136
14.14.3.7 setService	137
14.14.3.8 startUseNewServer	137
14.14.3.9 startUseServer	138
14.15ArnDiscoverResolver Class Reference	138
14.15.1 Detailed Description	139
14.15.2 Constructor & Destructor Documentation	140
14.15.2.1 ArnDiscoverResolver	140
14.15.3 Member Function Documentation	140
14.15.3.1 defaultService	140
14.15.3.2 resolve	140
14.15.3.3 setDefaultService	141
14.16ArnError Class Reference	141
14.16.1 Detailed Description	141
14.16.2 Member Enumeration Documentation	141
14.16.2.1 E	141
14.17ArnEvent Class Reference	142
14.17.1 Detailed Description	144
14.17.2 Member Typedef Documentation	144
14.17.2.1 Idx	144
14.17.3 Constructor & Destructor Documentation	144
14.17.3.1 ArnEvent	144
14.17.3.2 ~ArnEvent	144
14.17.4 Member Function Documentation	144
14.17.4.1 baseType	144
14.17.4.2 copyOpt	144
14.17.4.3 inhibitPendingChain	145
14.17.4.4 isArnEvent	145
14.17.4.5 makeHeapClone	145
14.17.4.6 setTarget	145
14.17.4.7 setTargetMutex	145
14.17.4.8 setTargetPendingChain	145
14.17.4.9 target	145
14.17.4.10oldx	145
14.17.4.11toldx	145

14.17.4.12oString	145
14.17.4.13oString	145
14.18ArnEventIdx Class Reference	146
14.18.1 Detailed Description	146
14.18.2 Member Enumeration Documentation	146
14.18.2.1 E	146
14.19ArnEvLinkCreate Class Reference	146
14.19.1 Detailed Description	148
14.19.2 Constructor & Destructor Documentation	148
14.19.2.1 ArnEvLinkCreate	148
14.19.3 Member Function Documentation	148
14.19.3.1 arnLink	148
14.19.3.2 isLastLink	148
14.19.3.3 makeHeapClone	148
14.19.3.4 path	148
14.19.3.5 type	148
14.20ArnEvModeChange Class Reference	148
14.20.1 Detailed Description	150
14.20.2 Constructor & Destructor Documentation	150
14.20.2.1 ArnEvModeChange	150
14.20.3 Member Function Documentation	150
14.20.3.1 linkId	150
14.20.3.2 makeHeapClone	150
14.20.3.3 mode	150
14.20.3.4 path	150
14.20.3.5 type	150
14.21ArnEvMonitor Class Reference	150
14.21.1 Detailed Description	152
14.21.2 Constructor & Destructor Documentation	152
14.21.2.1 ArnEvMonitor	152
14.21.3 Member Function Documentation	152
14.21.3.1 data	152
14.21.3.2 isLocal	152
14.21.3.3 makeHeapClone	152
14.21.3.4 monEvType	152
14.21.3.5 sessionHandler	152
14.21.3.6 type	152
14.22ArnEvRefChange Class Reference	152
14.22.1 Detailed Description	154
14.22.2 Constructor & Destructor Documentation	154

14.22.2.1 ArnEvRefChange	154
14.22.2.2 ~ArnEvRefChange	154
14.22.3 Member Function Documentation	154
14.22.3.1 makeHeapClone	154
14.22.3.2 refStep	154
14.22.3.3 type	154
14.23 ArnEvRetired Class Reference	154
14.23.1 Detailed Description	156
14.23.2 Constructor & Destructor Documentation	156
14.23.2.1 ArnEvRetired	156
14.23.3 Member Function Documentation	156
14.23.3.1 isBelow	156
14.23.3.2 isGlobal	156
14.23.3.3 makeHeapClone	156
14.23.3.4 startLink	156
14.23.3.5 type	156
14.24 ArnEvValueChange Class Reference	156
14.24.1 Detailed Description	158
14.24.2 Constructor & Destructor Documentation	158
14.24.2.1 ArnEvValueChange	158
14.24.2.2 ~ArnEvValueChange	158
14.24.3 Member Function Documentation	158
14.24.3.1 handleData	158
14.24.3.2 makeHeapClone	158
14.24.3.3 sendId	158
14.24.3.4 type	158
14.24.3.5 valueData	158
14.25 ArnEvZeroRef Class Reference	158
14.25.1 Detailed Description	160
14.25.2 Constructor & Destructor Documentation	160
14.25.2.1 ArnEvZeroRef	160
14.25.3 Member Function Documentation	160
14.25.3.1 arnLink	160
14.25.3.2 makeHeapClone	160
14.25.3.3 type	160
14.26 ArnInterface Class Reference	160
14.26.1 Detailed Description	162
14.26.2 Member Enumeration Documentation	162
14.26.2.1 DataType	162
14.26.2.2 NameF	163

14.26.2.3 ObjectMode	163
14.26.2.4 SameValue	163
14.26.3 Member Function Documentation	163
14.26.3.1 bytes	163
14.26.3.2 changeBasePath	163
14.26.3.3 childPath	164
14.26.3.4 exist	164
14.26.3.5 intNum	164
14.26.3.6 isFolder	164
14.26.3.7 isFolderPath	164
14.26.3.8 isLeaf	164
14.26.3.9 isProviderPath	164
14.26.3.10 itemName	164
14.26.3.11 items	164
14.26.3.12 makePath	165
14.26.3.13 num	165
14.26.3.14 providerPath	165
14.26.3.15 setBytes	165
14.26.3.16 setIntNum	165
14.26.3.17 setNum	165
14.26.3.18 setString	165
14.26.3.19 setValue	165
14.26.3.20 setVariant	165
14.26.3.21 string	166
14.26.3.22 winPath	166
14.26.3.23 value	166
14.26.3.24 variant	166
14.27 ArnItem Class Reference	166
14.27.1 Detailed Description	170
14.27.2 Constructor & Destructor Documentation	170
14.27.2.1 ArnItem	170
14.27.2.2 ArnItem	170
14.27.2.3 ArnItem	171
14.27.2.4 ~ArnItem	171
14.27.3 Member Function Documentation	171
14.27.3.1 addMode	171
14.27.3.2 arnExport	171
14.27.3.3 arnImport	172
14.27.3.4 arnItemCreated	172
14.27.3.5 arnModeChanged	172

14.27.3.6 bypassDelayPending	172
14.27.3.7 changed	173
14.27.3.8 changed	173
14.27.3.9 changed	173
14.27.3.10changed	173
14.27.3.11changed	173
14.27.3.12changed	173
14.27.3.13changed	173
14.27.3.14delay	174
14.27.3.15getMode	174
14.27.3.16sAutoDestroy	174
14.27.3.17sBiDirMode	174
14.27.3.18sDelayPending	174
14.27.3.19sFolder	175
14.27.3.20sIgnoreSameValue	175
14.27.3.21sIsMaster	175
14.27.3.22sPipeMode	175
14.27.3.23sProvider	176
14.27.3.24sSaveMode	177
14.27.3.25sTemplate	177
14.27.3.26modeChanged	177
14.27.3.27openFolder	177
14.27.3.28openUuid	178
14.27.3.29openUuidPipe	178
14.27.3.30operator=	178
14.27.3.31operator=	178
14.27.3.32operator=	178
14.27.3.33operator=	178
14.27.3.34operator=	178
14.27.3.35operator=	179
14.27.3.36operator=	179
14.27.3.37operator=	179
14.27.3.38operator=	179
14.27.3.39operator=	179
14.27.3.40setAutoDestroy	179
14.27.3.41setBiDirMode	179
14.27.3.42setBlockEcho	179
14.27.3.43setDelay	180
14.27.3.44setIgnoreSameValue	180
14.27.3.45setMaster	180

14.27.3.46	setPipeMode	180
14.27.3.47	setSaveMode	181
14.27.3.48	setTemplate	181
14.27.3.49	setValue	181
14.27.3.50	setValue	181
14.27.3.51	setValue	182
14.27.3.52	setValue	182
14.27.3.53	setValue	182
14.27.3.54	setValue	183
14.27.3.55	setValue	183
14.27.3.56	setValue	183
14.27.3.57	setValue	184
14.27.3.58	setValue	185
14.27.3.59	setValue	185
14.27.3.60	syncMode	185
14.27.3.61	toBool	186
14.27.3.62	toByteArray	186
14.27.3.63	toDouble	186
14.27.3.64	toggleBool	186
14.27.3.65	toInt	186
14.27.3.66	toInt64	187
14.27.3.67	toReal	187
14.27.3.68	toString	187
14.27.3.69	toUInt	187
14.27.3.70	toUInt64	188
14.27.3.71	toVariant	188
14.27.3.72	type	188
14.28	ArnItemB Class Reference	188
14.28.1	Detailed Description	190
14.28.2	Constructor & Destructor Documentation	190
14.28.2.1	ArnItemB	190
14.28.2.2	~ArnItemB	190
14.28.3	Member Function Documentation	190
14.28.3.1	arnLinkDestroyed	190
14.28.3.2	open	190
14.29	ArnItemQml Class Reference	191
14.29.1	Detailed Description	192
14.29.2	Member Function Documentation	192
14.29.2.1	addMode	192
14.29.2.2	getMode	193

14.30 ArnItemValve Class Reference	193
14.30.1 Detailed Description	195
14.30.2 Constructor & Destructor Documentation	195
14.30.2.1 ArnItemValve	195
14.30.3 Member Function Documentation	195
14.30.3.1 changed	195
14.30.3.2 isAutoDestroy	195
14.30.3.3 isMaster	195
14.30.3.4 isSaveMode	196
14.30.3.5 operator=	196
14.30.3.6 setAutoDestroy	196
14.30.3.7 setMaster	196
14.30.3.8 setSaveMode	197
14.30.3.9 setTarget	197
14.30.3.10 setValue	197
14.30.3.11 switchMode	197
14.30.3.12 toBool	197
14.31 ArnLinkValue Struct Reference	197
14.31.1 Detailed Description	198
14.31.2 Constructor & Destructor Documentation	198
14.31.2.1 ArnLinkValue	198
14.31.3 Member Data Documentation	198
14.31.3.1 valueByteArray	198
14.31.3.2 valueInt	198
14.31.3.3 valueReal	198
14.31.3.4 valueString	198
14.31.3.5 valueVariant	198
14.32 ArnM Class Reference	198
14.32.1 Detailed Description	201
14.32.2 Member Function Documentation	201
14.32.2.1 defaultIgnoreSameValue	201
14.32.2.2 destroyLink	201
14.32.2.3 destroyLinkLocal	201
14.32.2.4 errorLog	202
14.32.2.5 errorLogSig	202
14.32.2.6 errorSysName	202
14.32.2.7 exist	202
14.32.2.8 info	202
14.32.2.9 instance	202
14.32.2.10 sFolder	202

14.32.2.11isLeaf	202
14.32.2.12sMainThread	203
14.32.2.13sThreadedApp	203
14.32.2.14items	203
14.32.2.15oadFromDirRoot	203
14.32.2.16oadFromFile	204
14.32.2.17saveToFile	205
14.32.2.18setConsoleError	205
14.32.2.19setDefaultIgnoreSameValue	205
14.32.2.20setSkipLocalSysLoading	205
14.32.2.21setupErrorlog	206
14.32.2.22setValue	206
14.32.2.23setValue	206
14.32.2.24setValue	206
14.32.2.25setValue	206
14.32.2.26setValue	206
14.32.2.27setValue	207
14.32.2.28skipLocalSysLoading	207
14.32.2.29valueByteArray	207
14.32.2.30valueDouble	207
14.32.2.31valueInt	208
14.32.2.32valueReal	209
14.32.2.33valueString	209
14.32.2.34valueVariant	209
14.32.3 Friends And Related Function Documentation	209
14.32.3.1 ArnBasicItem	209
14.33ArnMonEventType Class Reference	210
14.33.1 Detailed Description	210
14.33.2 Member Enumeration Documentation	210
14.33.2.1 E	210
14.33.2.2 NS	210
14.34ArnMonitor Class Reference	211
14.34.1 Detailed Description	213
14.34.2 Constructor & Destructor Documentation	213
14.34.2.1 ArnMonitor	213
14.34.2.2 ArnMonitor	213
14.34.2.3 ~ArnMonitor	213
14.34.3 Member Function Documentation	213
14.34.3.1 arnChildDeleted	213
14.34.3.2 arnChildFound	214

14.34.3.3	arnChildFoundFolder	214
14.34.3.4	arnChildFoundLeaf	214
14.34.3.5	arnChildModeChanged	215
14.34.3.6	arnItemCreated	215
14.34.3.7	arnItemDeleted	215
14.34.3.8	arnItemModeChanged	215
14.34.3.9	client	216
14.34.3.10	clientId	216
14.34.3.11	foundChildDeleted	216
14.34.3.12	monitorClosed	216
14.34.3.13	monitorPath	217
14.34.3.14	reference	217
14.34.3.15	reStart	217
14.34.3.16	setClient	217
14.34.3.17	setClient	217
14.34.3.18	setMonitorPath	218
14.34.3.19	setReference	219
14.34.3.20	start	219
14.34.3.21	start	219
14.35	ArnMonitorQml Class Reference	220
14.35.1	Detailed Description	221
14.35.2	Member Function Documentation	221
14.35.2.1	reStart	221
14.36	ArnPersist Class Reference	221
14.36.1	Detailed Description	223
14.36.2	Constructor & Destructor Documentation	223
14.36.2.1	ArnPersist	223
14.36.2.2	~ArnPersist	223
14.36.3	Member Function Documentation	223
14.36.3.1	doArchive	223
14.36.3.2	flush	223
14.36.3.3	setArchiveDir	224
14.36.3.4	setMountPoint	224
14.36.3.5	setPersistDir	224
14.36.3.6	setupDataBase	225
14.36.3.7	setVcs	225
14.37	ArnPipe Class Reference	225
14.37.1	Detailed Description	227
14.37.2	Constructor & Destructor Documentation	227
14.37.2.1	ArnPipe	227

14.37.2.2 ArnPipe	228
14.37.2.3 ~ArnPipe	228
14.37.3 Member Function Documentation	228
14.37.3.1 changed	228
14.37.3.2 isAutoDestroy	228
14.37.3.3 isCheckSeq	228
14.37.3.4 isMaster	229
14.37.3.5 isSendSeq	229
14.37.3.6 openUuid	229
14.37.3.7 operator=	229
14.37.3.8 outOfSequence	230
14.37.3.9 setAutoDestroy	230
14.37.3.10setCheckSeq	230
14.37.3.11setMaster	230
14.37.3.12setSendSeq	230
14.37.3.13setValue	231
14.37.3.14setValueOverwrite	231
14.38 ArnQml Class Reference	231
14.38.1 Detailed Description	232
14.38.2 Member Function Documentation	234
14.38.2.1 arnRootPath	234
14.38.2.2 instance	234
14.38.2.3 setArnRootPath	234
14.38.2.4 setup	235
14.39 ArnRpc Class Reference	235
14.39.1 Detailed Description	238
14.39.2 Member Typedef Documentation	238
14.39.2.1 Mode	238
14.39.3 Constructor & Destructor Documentation	238
14.39.3.1 ArnRpc	238
14.39.3.2 ~ArnRpc	238
14.39.4 Member Function Documentation	238
14.39.4.1 addSenderSignals	238
14.39.4.2 batchConnect	239
14.39.4.3 batchConnect	239
14.39.4.4 batchConnect	239
14.39.4.5 defaultCall	240
14.39.4.6 getHeartBeatCheck	240
14.39.4.7 getHeartBeatSend	240
14.39.4.8 heartBeatChanged	240

14.39.4.9 heartbeatReceived	241
14.39.4.10 invoke	241
14.39.4.11 invoke	241
14.39.4.12 sHeartBeatOk	242
14.39.4.13 methodPrefix	242
14.39.4.14 mode	242
14.39.4.15 open	242
14.39.4.16 outOfSequence	242
14.39.4.17 pipe	242
14.39.4.18 pipeClosed	243
14.39.4.19 pipePath	243
14.39.4.20 receiver	243
14.39.4.21 rpcSender	243
14.39.4.22 pcSender	243
14.39.4.23 sendText	243
14.39.4.24 setHeartBeatCheck	244
14.39.4.25 setHeartBeatSend	245
14.39.4.26 setIncludeSender	245
14.39.4.27 setMethodPrefix	245
14.39.4.28 setMode	245
14.39.4.29 setPipe	245
14.39.4.30 setReceiver	246
14.39.4.31 textReceived	246
14.40 ArnRpcMode Class Reference	246
14.40.1 Detailed Description	246
14.40.2 Member Enumeration Documentation	246
14.40.2.1 E	246
14.41 ArnSapi Class Reference	247
14.41.1 Detailed Description	248
14.41.2 Constructor & Destructor Documentation	249
14.41.2.1 ArnSapi	249
14.41.2.2 ArnSapi	249
14.41.3 Member Function Documentation	250
14.41.3.1 batchConnectFrom	250
14.41.3.2 batchConnectTo	250
14.41.3.3 defaultPath	250
14.41.3.4 open	251
14.41.3.5 setDefaultPath	251
14.42 ArnSapiQml Class Reference	251
14.42.1 Detailed Description	253

14.42.2 Member Enumeration Documentation	253
14.42.2.1 Mode	253
14.42.3 Member Function Documentation	254
14.42.3.1 isHeartBeatOk	254
14.43 ArnScript Class Reference	254
14.43.1 Detailed Description	255
14.43.2 Constructor & Destructor Documentation	256
14.43.2.1 ArnScript	256
14.43.2.2 ArnScript	256
14.43.3 Member Function Documentation	256
14.43.3.1 engine	256
14.43.3.2 errorLog	256
14.43.3.3 errorText	256
14.43.3.4 evaluate	256
14.43.3.5 evaluateFile	256
14.43.3.6 idName	256
14.43.3.7 logUncaughtError	256
14.43.3.8 printFunction	256
14.43.4 Member Data Documentation	256
14.43.4.1 _depOfferProto	256
14.43.4.2 _depProto	257
14.43.4.3 _engine	257
14.43.4.4 _itemProto	257
14.43.4.5 _monitorProto	257
14.44 ArnScriptJob Class Reference	257
14.44.1 Detailed Description	258
14.44.2 Constructor & Destructor Documentation	258
14.44.2.1 ArnScriptJob	258
14.44.3 Member Function Documentation	258
14.44.3.1 errorLog	258
14.44.3.2 quit	258
14.44.3.3 setWatchDogTime	259
14.44.3.4 sigQuit	259
14.44.3.5 yield	259
14.45 ArnScriptJobControl Class Reference	259
14.45.1 Detailed Description	260
14.45.2 Constructor & Destructor Documentation	260
14.45.2.1 ArnScriptJobControl	260
14.45.3 Member Function Documentation	260
14.45.3.1 addConfig	260

14.45.3.2 addInterface	260
14.45.3.3 addInterfaceList	260
14.45.3.4 config	261
14.45.3.5 doSetupJob	261
14.45.3.6 errorText	261
14.45.3.7 id	261
14.45.3.8 loadScriptFile	261
14.45.3.9 name	261
14.45.3.10script	261
14.45.3.11scriptChanged	261
14.45.3.12setConfig	261
14.45.3.13setName	261
14.45.3.14setScript	261
14.45.3.15setThreaded	261
14.46ArnScriptJobFactory Class Reference	262
14.46.1 Detailed Description	262
14.46.2 Constructor & Destructor Documentation	262
14.46.2.1 ArnScriptJobFactory	262
14.46.2.2 ~ArnScriptJobFactory	262
14.46.3 Member Function Documentation	262
14.46.3.1 installExtension	262
14.46.3.2 setupInterface	262
14.46.3.3 setupJsObj	262
14.47ArnScriptJobs Class Reference	263
14.47.1 Detailed Description	263
14.47.2 Constructor & Destructor Documentation	264
14.47.2.1 ArnScriptJobs	264
14.47.3 Member Function Documentation	264
14.47.3.1 addJob	264
14.47.3.2 setFactory	264
14.47.3.3 start	264
14.48ArnServer Class Reference	264
14.48.1 Detailed Description	266
14.48.2 Constructor & Destructor Documentation	266
14.48.2.1 ArnServer	266
14.48.2.2 ~ArnServer	266
14.48.3 Member Function Documentation	266
14.48.3.1 addAccess	266
14.48.3.2 addFreePath	266
14.48.3.3 freePaths	267

14.48.3.4 isDemandLogin	267
14.48.3.5 isDemandLoginNet	267
14.48.3.6 listenAddress	268
14.48.3.7 noLoginNets	268
14.48.3.8 port	268
14.48.3.9 setDemandLogin	268
14.48.3.10setNoLoginNets	269
14.48.3.11setWhoIAm	269
14.48.3.12start	269
14.49ArnServerRemote Class Reference	270
14.49.1 Detailed Description	271
14.49.2 Constructor & Destructor Documentation	271
14.49.2.1 ArnServerRemote	271
14.49.2.2 ~ArnServerRemote	271
14.49.3 Member Function Documentation	271
14.49.3.1 startUseServer	271
14.50ArnServerRemoteSession Class Reference	271
14.50.1 Detailed Description	272
14.50.2 Member Typedef Documentation	272
14.50.2.1 KillMode	272
14.50.3 Constructor & Destructor Documentation	273
14.50.3.1 ArnServerRemoteSession	273
14.51ArnServerRemoteSessionKillMode Class Reference	273
14.51.1 Detailed Description	273
14.51.2 Member Enumeration Documentation	273
14.51.2.1 E	273
14.52ArnServerSession Class Reference	273
14.52.1 Detailed Description	274
14.52.2 Constructor & Destructor Documentation	275
14.52.2.1 ArnServerSession	275
14.52.3 Member Function Documentation	275
14.52.3.1 getAllow	275
14.52.3.2 getTraffic	275
14.52.3.3 infoReceived	275
14.52.3.4 loginCompleted	275
14.52.3.5 loginUserName	275
14.52.3.6 messageReceived	275
14.52.3.7 remoteWhoIAm	275
14.52.3.8 sendMessage	275
14.52.3.9 socket	275

14.53 ArnZeroConfB Class Reference	275
14.53.1 Detailed Description	277
14.53.2 Constructor & Destructor Documentation	277
14.53.2.1 ArnZeroConfB	277
14.53.2.2 ~ArnZeroConfB	277
14.53.3 Member Function Documentation	277
14.53.3.1 domain	277
14.53.3.2 fullServiceType	277
14.53.3.3 serviceType	278
14.53.3.4 setDomain	278
14.53.3.5 setServiceType	278
14.53.3.6 setSocketType	278
14.53.3.7 socketType	279
14.53.3.8 state	279
14.54 ArnZeroConfBrowser Class Reference	279
14.54.1 Detailed Description	281
14.54.2 Constructor & Destructor Documentation	282
14.54.2.1 ArnZeroConfBrowser	282
14.54.2.2 ArnZeroConfBrowser	282
14.54.2.3 ~ArnZeroConfBrowser	282
14.54.3 Member Function Documentation	282
14.54.3.1 activeServiceNames	282
14.54.3.2 browse	283
14.54.3.3 browseError	283
14.54.3.4 getNextId	283
14.54.3.5 isBrowsing	283
14.54.3.6 serviceAdded	284
14.54.3.7 serviceChanged	284
14.54.3.8 serviceNameTold	284
14.54.3.9 serviceRemoved	284
14.54.3.10 setSubType	285
14.54.3.11 stopBrowse	285
14.54.3.12 subType	285
14.54.4 Friends And Related Function Documentation	286
14.54.4.1 ArnZeroConfIntern	286
14.55 ArnZeroConfLookup Class Reference	286
14.55.1 Detailed Description	288
14.55.2 Constructor & Destructor Documentation	288
14.55.2.1 ArnZeroConfLookup	288
14.55.2.2 ArnZeroConfLookup	288

14.55.2.3 ~ArnZeroConfLookup	289
14.55.3 Member Function Documentation	289
14.55.3.1 host	289
14.55.3.2 hostAddr	289
14.55.3.3 id	289
14.55.3.4 isForceQtDnsLookup	289
14.55.3.5 lookup	290
14.55.3.6 lookuped	290
14.55.3.7 lookupError	290
14.55.3.8 releaseLookup	291
14.55.3.9 setForceQtDnsLookup	291
14.55.3.10setHost	291
14.55.3.11setId	291
14.55.4 Friends And Related Function Documentation	292
14.55.4.1 ArnZeroConfIntern	292
14.56 ArnZeroConfRegister Class Reference	292
14.56.1 Detailed Description	294
14.56.2 Constructor & Destructor Documentation	294
14.56.2.1 ArnZeroConfRegister	294
14.56.2.2 ArnZeroConfRegister	295
14.56.2.3 ArnZeroConfRegister	295
14.56.2.4 ~ArnZeroConfRegister	295
14.56.3 Member Function Documentation	295
14.56.3.1 addSubType	295
14.56.3.2 currentServiceName	296
14.56.3.3 getTxtRecordMap	296
14.56.3.4 host	296
14.56.3.5 port	296
14.56.3.6 registered	297
14.56.3.7 registerService	297
14.56.3.8 registrationError	297
14.56.3.9 releaseService	298
14.56.3.10serviceName	298
14.56.3.11setHost	298
14.56.3.12setPort	298
14.56.3.13setServiceName	299
14.56.3.14setSubTypes	299
14.56.3.15setTxtRecord	299
14.56.3.16setTxtRecordMap	299
14.56.3.17subTypes	300

14.56.3.1 <code>txtRecord</code>	300
14.56.4 Friends And Related Function Documentation	300
14.56.4.1 <code>ArnZeroConfIntern</code>	300
14.57 <code>ArnZeroConfResolve</code> Class Reference	301
14.57.1 Detailed Description	302
14.57.2 Constructor & Destructor Documentation	303
14.57.2.1 <code>ArnZeroConfResolve</code>	303
14.57.2.2 <code>ArnZeroConfResolve</code>	303
14.57.2.3 <code>ArnZeroConfResolve</code>	303
14.57.2.4 <code>~ArnZeroConfResolve</code>	303
14.57.3 Member Function Documentation	304
14.57.3.1 <code>getTxtRecordMap</code>	304
14.57.3.2 <code>host</code>	304
14.57.3.3 <code>id</code>	304
14.57.3.4 <code>port</code>	304
14.57.3.5 <code>releaseResolve</code>	305
14.57.3.6 <code>resolve</code>	305
14.57.3.7 <code>resolved</code>	305
14.57.3.8 <code>resolveError</code>	305
14.57.3.9 <code>serviceName</code>	306
14.57.3.10 <code>setId</code>	306
14.57.3.11 <code>setServiceName</code>	306
14.57.3.12 <code>txtRecord</code>	306
14.57.4 Friends And Related Function Documentation	307
14.57.4.1 <code>ArnZeroConfIntern</code>	307
14.58 <code>Arn::Coding Struct</code> Reference	307
14.58.1 Detailed Description	307
14.58.2 Member Enumeration Documentation	307
14.58.2.1 <code>E</code>	307
14.59 <code>Arn::DataType</code> Class Reference	307
14.59.1 Detailed Description	308
14.59.2 Member Enumeration Documentation	308
14.59.2.1 <code>E</code>	308
14.60 <code>Arn::EnumTxt</code> Class Reference	308
14.60.1 Detailed Description	309
14.60.2 Constructor & Destructor Documentation	310
14.60.2.1 <code>EnumTxt</code>	310
14.60.3 Member Function Documentation	310
14.60.3.1 <code>addBitSet</code>	310
14.60.3.2 <code>addEnumSet</code>	310

14.60.3.3 flagsFromString	311
14.60.3.4 flagsFromStringList	311
14.60.3.5 flagsToString	312
14.60.3.6 flagsToStringList	312
14.60.3.7 getBitSet	312
14.60.3.8 getEnumSet	313
14.60.3.9 getEnumVal	313
14.60.3.10getEnumVal	314
14.60.3.11getTxt	315
14.60.3.12getTxtString	315
14.60.3.13humanize	316
14.60.3.14name	316
14.60.3.15setMissingTxt	316
14.60.3.16setTxt	317
14.60.3.17setTxtRef	317
14.60.3.18setTxtString	317
14.61 ArnZeroConf::Error Struct Reference	317
14.61.1 Detailed Description	318
14.61.2 Member Enumeration Documentation	318
14.61.2.1 E	318
14.62 Arn::ExportCode Class Reference	318
14.62.1 Detailed Description	318
14.62.2 Member Enumeration Documentation	318
14.62.2.1 E	318
14.63 ArnClient::HostAddrPort Struct Reference	319
14.63.1 Detailed Description	319
14.63.2 Constructor & Destructor Documentation	319
14.63.2.1 HostAddrPort	319
14.63.3 Member Data Documentation	319
14.63.3.1 addr	319
14.63.3.2 port	319
14.64 Arn::InfoType Struct Reference	320
14.64.1 Detailed Description	320
14.64.2 Member Enumeration Documentation	320
14.64.2.1 E	320
14.65 ArnRpc::Invoke Struct Reference	320
14.65.1 Detailed Description	320
14.65.2 Member Enumeration Documentation	320
14.65.2.1 E	320
14.66 Arn::LinkFlags Struct Reference	321

14.66.1 Detailed Description	321
14.66.2 Member Enumeration Documentation	321
14.66.2.1 E	321
14.67MQArgument< T > Class Template Reference	321
14.67.1 Detailed Description	322
14.67.2 Constructor & Destructor Documentation	323
14.67.2.1 MQArgument	323
14.68MQBasicTimer Class Reference	323
14.68.1 Detailed Description	324
14.68.2 Constructor & Destructor Documentation	324
14.68.2.1 MQBasicTimer	324
14.68.3 Member Function Documentation	324
14.68.3.1 interval	324
14.68.3.2 setInterval	324
14.68.3.3 start	324
14.68.3.4 start	324
14.69MQGenericArgument Class Reference	324
14.69.1 Detailed Description	325
14.69.2 Constructor & Destructor Documentation	325
14.69.2.1 MQGenericArgument	326
14.69.2.2 MQGenericArgument	326
14.69.3 Member Function Documentation	326
14.69.3.1 label	326
14.70Arn::NameF Struct Reference	326
14.70.1 Detailed Description	326
14.70.2 Member Enumeration Documentation	326
14.70.2.1 E	326
14.71Arn::ObjectMode Class Reference	327
14.71.1 Detailed Description	327
14.71.2 Member Enumeration Documentation	327
14.71.2.1 E	327
14.72Arn::ObjectSyncMode Class Reference	327
14.72.1 Detailed Description	327
14.72.2 Member Enumeration Documentation	327
14.72.2.1 E	327
14.73ArnRpc::MethodsParam::Params Struct Reference	328
14.73.1 Detailed Description	328
14.73.2 Member Data Documentation	328
14.73.2.1 allMethodIds	328
14.73.2.2 methodIdsTab	328

14.73.2.3 paramNames	328
14.74Arn::QmlFileIO Class Reference	328
14.74.1 Detailed Description	330
14.74.2 Constructor & Destructor Documentation	330
14.74.2.1 QmlFileIO	330
14.74.3 Member Function Documentation	330
14.74.3.1 error	330
14.74.3.2 path	330
14.74.3.3 pathChanged	330
14.74.3.4 read	330
14.74.3.5 readBytes	330
14.74.3.6 setPath	330
14.74.3.7 write	330
14.74.3.8 writeBytes	330
14.74.4 Property Documentation	330
14.74.4.1 path	330
14.75Arn::QmlMQtObject Class Reference	331
14.75.1 Detailed Description	332
14.75.2 Constructor & Destructor Documentation	332
14.75.2.1 QmlMQtObject	332
14.75.2.2 ~QmlMQtObject	332
14.75.3 Member Function Documentation	332
14.75.3.1 classBegin	332
14.75.3.2 completed	332
14.75.3.3 componentComplete	332
14.75.3.4 data	332
14.75.3.5 data_append	332
14.75.3.6 data_at	332
14.75.3.7 data_clear	332
14.75.3.8 data_count	333
14.75.3.9 parentChanged	333
14.75.3.10parentItem	333
14.75.3.11setParentItem	333
14.76Arn::SameValue Struct Reference	333
14.76.1 Detailed Description	333
14.76.2 Member Enumeration Documentation	333
14.76.2.1 E	333
14.77ArnDiscoverInfo::State Struct Reference	334
14.77.1 Detailed Description	334
14.77.2 Member Enumeration Documentation	334

14.77.2.1 E	334
14.78 ArnZeroConf::State Struct Reference	334
14.78.1 Detailed Description	335
14.78.2 Member Enumeration Documentation	335
14.78.2.1 E	335
14.79 ArnDiscoverAdvertise::State Struct Reference	335
14.79.1 Detailed Description	335
14.79.2 Member Enumeration Documentation	336
14.79.2.1 E	336
14.80 ArnError::StdCode Struct Reference	336
14.80.1 Detailed Description	336
14.80.2 Member Enumeration Documentation	336
14.80.2.1 E	336
14.81 ArnItemValve::SwitchMode Struct Reference	336
14.81.1 Detailed Description	337
14.81.2 Member Enumeration Documentation	337
14.81.2.1 E	337
14.82 ArnScriptJobs::Type Struct Reference	337
14.82.1 Detailed Description	337
14.82.2 Member Enumeration Documentation	337
14.82.2.1 E	337
14.83 ArnDiscover::Type Struct Reference	338
14.83.1 Detailed Description	338
14.83.2 Member Enumeration Documentation	338
14.83.2.1 E	338
14.84 ArnServer::Type Struct Reference	338
14.84.1 Detailed Description	338
14.84.2 Member Enumeration Documentation	338
14.84.2.1 E	338
14.85 ArnQml::UseFlags Struct Reference	339
14.85.1 Detailed Description	339
14.85.2 Member Enumeration Documentation	339
14.85.2.1 E	339
14.86 Arn::XStringMap Class Reference	339
14.86.1 Detailed Description	341
14.86.2 Constructor & Destructor Documentation	342
14.86.2.1 XStringMap	342
14.86.2.2 XStringMap	342
14.86.2.3 XStringMap	342
14.86.2.4 XStringMap	342

14.86.2.5 ~XStringMap	342
14.86.3 Member Function Documentation	342
14.86.3.1 add	342
14.86.3.2 add	342
14.86.3.3 add	342
14.86.3.4 add	342
14.86.3.5 add	342
14.86.3.6 add	342
14.86.3.7 add	343
14.86.3.8 add	343
14.86.3.9 add	343
14.86.3.10 add	343
14.86.3.11 addNum	343
14.86.3.12 addNum	343
14.86.3.13 addNum	343
14.86.3.14 addNum	343
14.86.3.15 addNum	343
14.86.3.16 addNum	343
14.86.3.17 addNum	343
14.86.3.18 addNum	343
14.86.3.19 addNum	344
14.86.3.20 addValues	344
14.86.3.21 append	344
14.86.3.22 append	344
14.86.3.23 append	344
14.86.3.24 append	344
14.86.3.25 append	344
14.86.3.26 append	344
14.86.3.27 append	344
14.86.3.28 append	344
14.86.3.29 append	344
14.86.3.30 append	344
14.86.3.31 clear	345
14.86.3.32 fromXString	345
14.86.3.33 indexOf	345
14.86.3.34 indexOf	345
14.86.3.35 indexOf	345
14.86.3.36 indexOfValue	345
14.86.3.37 indexOfValue	345
14.86.3.38 info	345

14.86.3.39key	345
14.86.3.40key	345
14.86.3.41key	345
14.86.3.42keyRef	345
14.86.3.43keys	346
14.86.3.44keyString	346
14.86.3.45keyString	346
14.86.3.46maxEnumOf	346
14.86.3.47operator+=	346
14.86.3.48operator+=	346
14.86.3.49operator=	346
14.86.3.50remove	346
14.86.3.51remove	346
14.86.3.52remove	346
14.86.3.53remove	346
14.86.3.54set	346
14.86.3.55set	347
14.86.3.56set	347
14.86.3.57set	347
14.86.3.58set	347
14.86.3.59set	347
14.86.3.60set	347
14.86.3.61setEmptyKeysToValue	347
14.86.3.62size	347
14.86.3.63squeeze	347
14.86.3.64stringCode	347
14.86.3.65stringDecode	347
14.86.3.66toVariantMap	347
14.86.3.67toXString	348
14.86.3.68value	348
14.86.3.69value	348
14.86.3.70value	348
14.86.3.71value	348
14.86.3.72value	348
14.86.3.73valueRef	348
14.86.3.74values	348
14.86.3.75valueString	348
14.86.3.76valueString	348
14.86.3.77valueString	348
14.86.3.78valueString	348

14.86.3.79valueString	349
15 File Documentation	351
15.1 doc/Description.md File Reference	351
15.2 doc/HelpIndex.txt File Reference	351
15.3 doc/Install.md File Reference	351
15.4 doc/Internals.md File Reference	351
15.5 doc/ToDo.md File Reference	351
15.6 examples/Examples.txt File Reference	351
15.7 README.md File Reference	351
15.8 src/Arn.cpp File Reference	351
15.9 src/ArnBasicItem.cpp File Reference	353
15.10src/ArnClient.cpp File Reference	353
15.11src/ArnDepend.cpp File Reference	354
15.11.1 Variable Documentation	354
15.11.1.1 ArnDependPath	355
15.12src/ArnDiscover.cpp File Reference	355
15.13src/ArnDiscoverConnect.cpp File Reference	355
15.14src/ArnDiscoverRemote.cpp File Reference	356
15.15src/ArnEvent.cpp File Reference	356
15.15.1 Macro Definition Documentation	356
15.15.1.1 TO_IDX_RETVAL	356
15.16src/ArnInc/Arn.hpp File Reference	357
15.16.1 Macro Definition Documentation	359
15.16.1.1 ARNREAL	359
15.16.1.2 DATASTREAM_VER	359
15.17src/ArnInc/ArnBasicItem.hpp File Reference	359
15.18src/ArnInc/ArnClient.hpp File Reference	360
15.19src/ArnInc/ArnDepend.hpp File Reference	360
15.20src/ArnInc/ArnDiscover.hpp File Reference	361
15.21src/ArnInc/ArnDiscoverConnect.hpp File Reference	363
15.22src/ArnInc/ArnDiscoverRemote.hpp File Reference	363
15.23src/ArnInc/ArnError.hpp File Reference	364
15.24src/ArnInc/ArnEvent.hpp File Reference	365
15.25src/ArnInc/ArnInterface.hpp File Reference	366
15.26src/ArnInc/ArnItem.hpp File Reference	367
15.26.1 Function Documentation	368
15.26.1.1 operator<<	368
15.27src/ArnInc/ArnItemB.hpp File Reference	368
15.28src/ArnInc/ArnItemValve.hpp File Reference	369

15.29src/ArnInc/ArnLib.hpp File Reference	370
15.30src/ArnInc/ArnLib_global.hpp File Reference	371
15.30.1 Macro Definition Documentation	371
15.30.1.1 ARNLIBSHARED_EXPORT	371
15.31src/ArnInc/ArnLinkHandle.hpp File Reference	371
15.32src/ArnInc/ArnM.hpp File Reference	372
15.33src/ArnInc/ArnMonEvent.hpp File Reference	373
15.34src/ArnInc/ArnMonitor.hpp File Reference	374
15.35src/ArnInc/ArnPersist.hpp File Reference	374
15.36src/ArnInc/ArnPersistSapi.hpp File Reference	375
15.37src/ArnInc/ArnPipe.hpp File Reference	376
15.38src/ArnInc/ArnQml.hpp File Reference	377
15.38.1 Macro Definition Documentation	379
15.38.1.1 QML_ENGINE	379
15.38.1.2 QML_LIST_PROPERTY	379
15.38.1.3 QML_NETACC_FACTORY	379
15.38.1.4 QML_PARSER_STATUS	379
15.38.1.5 QML_Qt4	379
15.38.1.6 QML_QUICK_TYPE	379
15.39src/ArnInc/ArnQmlMQt.hpp File Reference	379
15.40src/ArnInc/ArnQmlMSystem.hpp File Reference	380
15.41src/ArnInc/ArnRpc.hpp File Reference	381
15.41.1 Macro Definition Documentation	382
15.41.1.1 MQ_ARG	382
15.41.1.2 no_queue	382
15.42src/ArnInc/ArnSapi.hpp File Reference	383
15.42.1 Macro Definition Documentation	384
15.42.1.1 MQ_PUBLIC_ACCESS	384
15.43src/ArnInc/ArnScript.hpp File Reference	384
15.44src/ArnInc/ArnScriptJob.hpp File Reference	385
15.45src/ArnInc/ArnScriptJobs.hpp File Reference	386
15.46src/ArnInc/ArnServer.hpp File Reference	386
15.47src/ArnInc/ArnServerRemote.hpp File Reference	387
15.48src/ArnInc/ArnZeroConf.hpp File Reference	388
15.48.1 Typedef Documentation	389
15.48.1.1 DNSServiceRef	389
15.49src/ArnInc/MQFlags.hpp File Reference	390
15.49.1 Macro Definition Documentation	391
15.49.1.1 MQ_DECLARE_ENUM	391
15.49.1.2 MQ_DECLARE_ENUM_NSTXT	391

15.49.1.3 MQ_DECLARE_ENUMTXT	391
15.49.1.4 MQ_DECLARE_FLAGS	391
15.49.1.5 MQ_DECLARE_FLAGS_NSTXT	392
15.49.1.6 MQ_DECLARE_FLAGSTXT	392
15.49.1.7 MQ_DECLARE_OPERATORS_FOR_FLAGS	392
15.49.1.8 MQ_NSTXT_FILL_MISSING	392
15.49.1.9 MQ_NSTXT_FILL_MISSING_FROM	392
15.50src/ArnInc/XStringMap.hpp File Reference	393
15.50.1 Macro Definition Documentation	393
15.50.1.1 ARNXSTRINGMAP_VER	393
15.51src/ArnItem.cpp File Reference	394
15.51.1 Function Documentation	394
15.51.1.1 operator<<	394
15.52src/ArnItemB.cpp File Reference	394
15.53src/ArnItemNet.cpp File Reference	395
15.54src/ArnItemNet.hpp File Reference	395
15.55src/ArnItemValve.cpp File Reference	396
15.56src/ArnLib.cpp File Reference	396
15.57src/ArnLink.cpp File Reference	397
15.58src/ArnLink.hpp File Reference	398
15.58.1 Typedef Documentation	399
15.58.1.1 ArnBasicItemList	399
15.58.1.2 ArnLinkList	399
15.59src/ArnLinkHandle.cpp File Reference	399
15.60src/ArnM.cpp File Reference	400
15.61src/ArnMonitor.cpp File Reference	400
15.62src/ArnPersist.cpp File Reference	401
15.62.1 Variable Documentation	401
15.62.1.1 arnDbSaveVer	401
15.63src/ArnPipe.cpp File Reference	402
15.64src/ArnQml.cpp File Reference	402
15.65src/ArnQmlMQt.cpp File Reference	403
15.66src/ArnQmlMSystem.cpp File Reference	403
15.67src/ArnRpc.cpp File Reference	404
15.67.1 Macro Definition Documentation	404
15.67.1.1 RPC_STORAGE_NAME	404
15.68src/ArnSapi.cpp File Reference	404
15.69src/ArnScript.cpp File Reference	405
15.70src/ArnScriptJob.cpp File Reference	405
15.70.1 Variable Documentation	406

15.70.1.1 EventQuit	406
15.71src/ArnScriptJobs.cpp File Reference	406
15.72src/ArnServer.cpp File Reference	406
15.73src/ArnServerRemote.cpp File Reference	407
15.74src/ArnSync.cpp File Reference	407
15.74.1 Macro Definition Documentation	407
15.74.1.1 ARNSYNCVER	408
15.75src/ArnSync.hpp File Reference	408
15.75.1 Macro Definition Documentation	408
15.75.1.1 ARNRECNAME	408
15.76src/ArnSyncLogin.cpp File Reference	409
15.77src/ArnSyncLogin.hpp File Reference	409
15.78src/ArnXStringMap.cpp File Reference	410
15.79src/ArnZeroConf.cpp File Reference	410
15.80src/MQFlags.cpp File Reference	411
16 Example Documentation	413
16.1 ArnDemoChat/main.cpp	413
16.2 ArnDemoChat/MainWindow.cpp	413
16.3 ArnDemoChat/MainWindow.hpp	414
16.4 ArnDemoChatServer/ChatSapi.hpp	415
16.5 ArnDemoChatServer/main.cpp	416
16.6 ArnDemoChatServer/MainWindow.cpp	416
16.7 ArnDemoChatServer/MainWindow.hpp	418
Index	420

Chapter 1

README

Copyright (C) 2010-2016 Michael Wiklund. All rights reserved. Contact: arnlib@wiklund.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), 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).
- 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

General Description

This document describes the general concepts of the ArnLib.

2.1 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.

2.1.1 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 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`.

2.1.2 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. This makes difference when client connects or reconnects to the server. The data from the *default generator* is then used and synced. Also echo of data to the client side *ARN data object* is prohibited.
- **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\(\)](#).

2.1.3 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".

2.1.4 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.

2.1.5 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. 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.

2.1.6 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.

2.1.6.1 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\(\)](#).

2.1.6.2 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.

2.1.7 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](#).

2.1.7.1 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.

2.1.8 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.
```

2.1.8.1 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`.

2.2 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.

2.2.1 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.

2.2.2 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 asumed.
- 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".

2.3 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.

2.3.1 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*.

2.3.1.1 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".

2.3.1.2 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.

2.3.1.3 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](#).

2.3.2 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 [ArnDiscover↔Remote](#).

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::setCustom↔Properties\(\)](#) using an [Arn::XStringMap](#). The property 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 [ArnDiscover↔BrowserB::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* alfabetically 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 dissappear.

2.3.3 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 [ArnDiscover↔Remote](#) 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 [ArnDiscover↔Connector](#), both as a *direct host* list and a *discover host*.

The *discover host* is indirerctly 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::set↔ResolveRefreshTimeout\(\)](#).

2.4 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 3

Installation and usage

3.1 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.

The first step is to edit the *.pri / *.pro files to adjust them to your needs. Take care to select your deployment directories.

3.2 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.

3.3 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.

3.3.1 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
```

3.3.2 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.

3.3.3 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.

3.3.4 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++
```

3.3.5 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.

3.4 Using ArnLib

In the *.pro file of the application the below lines can be used.

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.

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.

```
CONFIG += ArnLibCompile
CONFIG += mDnsIntern

greaterThan(QT_MAJOR_VERSION, 4) {
    ARNLIB = Arn5
} else {
    ARNLIB = Arn4
}

ArnLibCompile {
    #ARN += client
    ARN += server
    ARN += discover
```

```
include(../ArnLib/src/ArnLib.pri)
INCLUDEPATH += $$PWD/../ArnLib/src
} else {
    win32: INCLUDEPATH += $$PWD/../ArnLib/src
    win32:CONFIG(release, debug|release): LIBS += -L$$OUT_PWD/../ArnLib/release/ -l${ARNLIB}
    else:win32:CONFIG(debug, debug|release): LIBS += -L$$OUT_PWD/../ArnLib/debug/ -l${ARNLIB}
    else:unix: LIBS += -L$$OUT_PWD/../ArnLib/ -l${ARNLIB}
}

!mDnsIntern {
    win32:CONFIG(release, debug|release): LIBS += -ldns_sd
    else:win32:CONFIG(debug, debug|release): LIBS += -ldns_sd
    else:unix: LIBS += -ldns_sd
}
```

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 4

ArnLib Internals

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

4.1 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 joblnit()
- 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 followning 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.

4.2 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.

4.3 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 5

ArnLib Todo

Major

- Script support for Sapi.
- ArnObject Link to other ArnObject (like in a filesystem).
- Add atomic operations in [ArnItem](#) for: "+=", "&=", "|=" and alike.
- 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 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 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=0)

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	49
ArnDiscover	58
ArnZeroConf	58

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

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

Arn::_InitEnumTxt	59
Arn::Allow	59
ArnBasicItem	60
ArnItemB	188
ArnItem	166
ArnItemQml	191
ArnItemValve	193
ArnMonitor	211
ArnMonitorQml	220
ArnPipe	225
ArnClientConnectStat	94
ArnClientReg	95
ArnDiscoverInfo	126
ArnError	141
ArnEventIdx	146
ArnLinkValue	197
ArnMonEventType	210
ArnRpcMode	246
ArnScriptJobB	
ArnScriptJob	257
ArnScriptJobFactory	262
ArnServerRemoteSessionKillMode	273
Arn::Coding	307
Arn::DataType	307
Arn::EnumTxt	308
ArnZeroConf::Error	317
Arn::ExportCode	318
ArnClient::HostAddrPort	319
Arn::InfoType	320
ArnRpc::Invoke	320
Arn::LinkFlags	321
Arn::NameF	326
Arn::ObjectMode	327
Arn::ObjectSyncMode	327
ArnRpc::MethodsParam::Params	328
QBasicTimer	
MQBasicTimer	323

QEvent	
ArnEvent	142
ArnEvLinkCreate	146
ArnEvModeChange	148
ArnEvMonitor	150
ArnEvRefChange	152
ArnEvRetired	154
ArnEvValueChange	156
ArnEvZeroRef	158
QGenericArgument	
MQGenericArgument	324
MQArgument< T >	321
QML_PARSER_STATUS	
Arn::QmlMQtObject	331
ArnItemQml	191
ArnMonitorQml	220
ArnSapiQml	251
QObject	
Arn::QmlMFileIO	328
Arn::QmlMQtObject	331
ArnClient	79
ArnDepend	96
ArnDependOffer	100
ArnDiscoverAdvertise	102
ArnDiscoverRemote	132
ArnDiscoverBrowserB	113
ArnDiscoverBrowser	109
ArnDiscoverResolver	138
ArnDiscoverConnector	119
ArnInterface	160
ArnItemB	188
ArnM	198
ArnPersist	221
ArnQml	231
ArnRpc	235
ArnSapi	247
ArnSapiQml	251
ArnScript	254
ArnScriptJobControl	259
ArnScriptJobs	263
ArnServer	264
ArnServerRemote	270
ArnServerRemoteSession	271
ArnServerSession	273
ArnZeroConfB	275
ArnZeroConfBrowser	279
ArnZeroConfLookup	286
ArnZeroConfRegister	292
ArnZeroConfResolve	301
Arn::SameValue	333
ArnDiscoverInfo::State	334
ArnZeroConf::State	334
ArnDiscoverAdvertise::State	335
ArnError::StdCode	336
ArnItemValve::SwitchMode	336
ArnScriptJobs::Type	337
ArnDiscover::Type	338
ArnServer::Type	338

ArnQml::UseFlags	339
Arn::XStringMap	339

Chapter 11

Class Index

11.1 Class List

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

Arn::_InitEnumTxt	59
Arn::Allow	59
ArnBasicItem	
Base class handle for an Arn Data Object	60
ArnClient	
Class for connecting to an Arn Server	79
ArnClientConnectStat	94
ArnClientReg	95
ArnDepend	
Class for setting up dependencis to needed services	96
ArnDependOffer	
Class for advertising that a <i>service</i> is available	100
ArnDiscoverAdvertise	
Advertise an Arn service	102
ArnDiscoverBrowser	
Browsing for Arn services	109
ArnDiscoverBrowserB	
Browse() and resolve() together, may never be used to the same instance	113
ArnDiscoverConnector	
An automatic client discover connector	119
ArnDiscoverInfo	
Class for holding current discover info of one service	126
ArnDiscoverRemote	
Discover with remote setting	132
ArnDiscoverResolver	
Resolv an Arn service	138
ArnError	141
ArnEvent	142
ArnEventIdx	146
ArnEvLinkCreate	146
ArnEvModeChange	148
ArnEvMonitor	150
ArnEvRefChange	152
ArnEvRetired	154
ArnEvValueChange	156
ArnEvZeroRef	158
ArnInterface	160

ArnItem	Handle for an Arn Data Object	166
ArnItemB	Base class handle for an Arn Data Object	188
ArnItemQml	ARN Item QML	191
ArnItemValve	Valve for controlling stream to/from an ArnItemB	193
ArnLinkValue		197
ArnM	Arn main class	198
ArnMonEventType		210
ArnMonitor	A client remote monitor to detect changes at server	211
ArnMonitorQml	ARN Monitor QML	220
ArnPersist	Class for handling persistent Arn Data object	221
ArnPipe	ArnItem specialized as a pipe	225
ArnQml	ARN QML	231
ArnRpc	Remote Procedure Call	235
ArnRpcMode		246
ArnSapi	Service API	247
ArnSapiQml	ARN Sapi QML	251
ArnScript		254
ArnScriptJob	Interface class to be normally used, is also Script Job interface	257
ArnScriptJobControl	Is thread-safe (except doSetupJob)	259
ArnScriptJobFactory	Must be thread-safe as subclassed	262
ArnScriptJobs		263
ArnServer	Class for making an Arn Server	264
ArnServerRemote	Class for remote controlling an Arn Server	270
ArnServerRemoteSession		271
ArnServerRemoteSessionKillMode		273
ArnServerSession		273
ArnZeroConfB	Base class for Zero Config	275
ArnZeroConfBrowser	Browsing for ZeroConfig services	279
ArnZeroConfLookup	Lookup a host	286
ArnZeroConfRegister	Registering a ZeroConfig service	292
ArnZeroConfResolve	Resolv a ZeroConfig service	301
Arn::Coding		307
Arn::DataType	Data type of an Arn Data Object	307

Arn::EnumTxt	
Class Enum text	308
ArnZeroConf::Error	
Errors of ZeroConfig, other values are defined in <code>dns_sd.h</code>	317
Arn::ExportCode	
Code used in blob for <code>arnExport()</code> and <code>arnImport()</code>	318
ArnClient::HostAddrPort	319
Arn::InfoType	
Info type for exchange static (meta) info between ArnClient and ArnServer	320
ArnRpc::Invoke	320
Arn::LinkFlags	
Link flags when accessing an <i>Arn Data Object</i>	321
MQArgument< T >	
Similar to <code>QArgument</code> but with added argument label (parameter name)	321
MQBasicTimer	323
MQGenericArgument	
Similar to <code>QGenericArgument</code> but with added argument label (parameter name)	324
Arn::NameF	326
Arn::ObjectMode	327
Arn::ObjectSyncMode	327
ArnRpc::MethodsParam::Params	328
Arn::QmlMFileIO	328
Arn::QmlMQObject	331
Arn::SameValue	
Action when assigning same value to an ArnItem	333
ArnDiscoverInfo::State	
State of Arn discover browse data. Can be tested by relative order	334
ArnZeroConf::State	
States of ZeroConfig, limited valid for each ArnZeroConfB subclass / These values must be synced with: ArnDiscover::State	334
ArnDiscoverAdvertise::State	
States of DiscoverAdvertise / These values must be synced with: ArnZeroConf::State	335
ArnError::StdCode	336
ArnItemValve::SwitchMode	336
ArnScriptJobs::Type	337
ArnDiscover::Type	
Types of Arn discover advertise	338
ArnServer::Type	338
ArnQml::UseFlags	339
Arn::XStringMap	
Container class with string representation for serialized data	339

Chapter 12

File Index

12.1 File List

Here is a list of all files with brief descriptions:

src/Arn.cpp	351
src/ArnBasicItem.cpp	353
src/ArnClient.cpp	353
src/ArnDepend.cpp	354
src/ArnDiscover.cpp	355
src/ArnDiscoverConnect.cpp	355
src/ArnDiscoverRemote.cpp	356
src/ArnEvent.cpp	356
src/ArnItem.cpp	394
src/ArnItemB.cpp	394
src/ArnItemNet.cpp	395
src/ArnItemNet.hpp	395
src/ArnItemValve.cpp	396
src/ArnLib.cpp	396
src/ArnLink.cpp	397
src/ArnLink.hpp	398
src/ArnLinkHandle.cpp	399
src/ArnM.cpp	400
src/ArnMonitor.cpp	400
src/ArnPersist.cpp	401
src/ArnPipe.cpp	402
src/ArnQml.cpp	402
src/ArnQmlMQT.cpp	403
src/ArnQmlMSystem.cpp	403
src/ArnRpc.cpp	404
src/ArnSapi.cpp	404
src/ArnScript.cpp	405
src/ArnScriptJob.cpp	405
src/ArnScriptJobs.cpp	406
src/ArnServer.cpp	406
src/ArnServerRemote.cpp	407
src/ArnSync.cpp	407
src/ArnSync.hpp	408
src/ArnSyncLogin.cpp	409
src/ArnSyncLogin.hpp	409
src/ArnXStringMap.cpp	410
src/ArnZeroConf.cpp	410
src/MQFlags.cpp	411

src/ArnInc/Arn.hpp	357
src/ArnInc/ArnBasicItem.hpp	359
src/ArnInc/ArnClient.hpp	360
src/ArnInc/ArnDepend.hpp	360
src/ArnInc/ArnDiscover.hpp	361
src/ArnInc/ArnDiscoverConnect.hpp	363
src/ArnInc/ArnDiscoverRemote.hpp	363
src/ArnInc/ArnError.hpp	364
src/ArnInc/ArnEvent.hpp	365
src/ArnInc/ArnInterface.hpp	366
src/ArnInc/ArnItem.hpp	367
src/ArnInc/ArnItemB.hpp	368
src/ArnInc/ArnItemValve.hpp	369
src/ArnInc/ArnLib.hpp	370
src/ArnInc/ArnLib_global.hpp	371
src/ArnInc/ArnLinkHandle.hpp	371
src/ArnInc/ArnM.hpp	372
src/ArnInc/ArnMonEvent.hpp	373
src/ArnInc/ArnMonitor.hpp	374
src/ArnInc/ArnPersist.hpp	374
src/ArnInc/ArnPersistSapi.hpp	375
src/ArnInc/ArnPipe.hpp	376
src/ArnInc/ArnQml.hpp	377
src/ArnInc/ArnQmlMQt.hpp	379
src/ArnInc/ArnQmlMSystem.hpp	380
src/ArnInc/ArnRpc.hpp	381
src/ArnInc/ArnSapi.hpp	383
src/ArnInc/ArnScript.hpp	384
src/ArnInc/ArnScriptJob.hpp	385
src/ArnInc/ArnScriptJobs.hpp	386
src/ArnInc/ArnServer.hpp	386
src/ArnInc/ArnServerRemote.hpp	387
src/ArnInc/ArnZeroConf.hpp	388
src/ArnInc/MQFlags.hpp	390
src/ArnInc/XStringMap.hpp	393

Chapter 13

Namespace Documentation

13.1 Arn Namespace Reference

Classes

- struct [_InitEnumTxt](#)
- class [Allow](#)
- 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](#)
- struct [SameValue](#)
Action when assigning same value to an [ArnItem](#).
- class [XStringMap](#)
Container class with string representation for serialized data.

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 [isPower2](#) (uint x)

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 [debugRecInOut](#) = 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
- const QString [resourceArnLib](#) = ":/ArnLib/"
- const QString [resourceArnRoot](#) = ":/ArnLib/ArnRoot/"
- const quint16 [defaultTcpPort](#) = 2022

13.1.1 Function Documentation

13.1.1.1 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 133 of file Arn.cpp.

13.1.1.2 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 110 of file Arn.cpp.

13.1.1.3 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 96 of file Arn.cpp.

13.1.1.4 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 50 of file Arn.cpp.

13.1.1.5 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 144 of file Arn.cpp.

13.1.1.6 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 78 of file Arn.cpp.

13.1.1.7 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 238 of file Arn.cpp.

13.1.1.8 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 206 of file Arn.cpp.

13.1.1.9 bool Arn::isPower2 (uint *x*)

Definition at line 53 of file MQFlags.cpp.

13.1.1.10 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 212 of file Arn.cpp.

13.1.1.11 QString Arn::itemName (const QString & *path*)

The last part of a *path*

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

Parameters

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

Returns

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

Definition at line 86 of file Arn.cpp.

13.1.1.12 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. [ArnDiscover](#)↔
[Browser](#) 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

<i>in</i>	<i>host</i>	the name or address of the host
<i>in</i>	<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 231 of file Arn.cpp.

13.1.1.13 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 124 of file Arn.cpp.

13.1.1.14 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 179 of file Arn.cpp.

13.1.1.15 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>	chooses between provider and requester path. false = requester path, default is true = provider path.

Return values

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

See also

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

Definition at line 200 of file Arn.cpp.

13.1.1.16 QString Arn::twinPath (const QString & *path*)

Get the bidirectional twin to a given *path*

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

Parameters

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

Returns

The twin *path*

See also

[Bidirectional Arn Data Objects](#)

Definition at line 191 of file Arn.cpp.

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

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

Parameters

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

Returns

the unique path

Definition at line 218 of file Arn.cpp.

13.1.2 Variable Documentation**13.1.2.1 bool Arn::debugDepend = false**

Definition at line 46 of file ArnLib.cpp.

13.1.2.2 bool Arn::debugDiscover = false

Definition at line 48 of file ArnLib.cpp.

13.1.2.3 bool Arn::debugLinkDestroy = false

Definition at line 40 of file ArnLib.cpp.

13.1.2.4 bool Arn::debugLinkRef = false

Definition at line 39 of file ArnLib.cpp.

13.1.2.5 bool Arn::debugMDNS = false

Definition at line 50 of file ArnLib.cpp.

13.1.2.6 bool Arn::debugMonitor = false

Definition at line 43 of file ArnLib.cpp.

13.1.2.7 `bool Arn::debugMonitorTest = false`

Definition at line 44 of file ArnLib.cpp.

13.1.2.8 `bool Arn::debugQmlNetwork = false`

Definition at line 47 of file ArnLib.cpp.

13.1.2.9 `bool Arn::debugReclnOut = false`

Definition at line 41 of file ArnLib.cpp.

13.1.2.10 `bool Arn::debugRPC = false`

Definition at line 45 of file ArnLib.cpp.

13.1.2.11 `bool Arn::debugShareObj = false`

Definition at line 42 of file ArnLib.cpp.

13.1.2.12 `bool Arn::debugSizes = false`

Definition at line 37 of file ArnLib.cpp.

13.1.2.13 `bool Arn::debugThreading = false`

Definition at line 38 of file ArnLib.cpp.

13.1.2.14 `bool Arn::debugZeroConf = false`

Definition at line 49 of file ArnLib.cpp.

13.1.2.15 `const quint16 Arn::defaultTcpPort = 2022`

Definition at line 50 of file Arn.hpp.

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

Definition at line 43 of file Arn.cpp.

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

Definition at line 45 of file Arn.cpp.

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

Definition at line 44 of file Arn.cpp.

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

Definition at line 41 of file Arn.cpp.

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

Definition at line 42 of file Arn.cpp.

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

Definition at line 46 of file Arn.cpp.

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

Definition at line 47 of file Arn.cpp.

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

Definition at line 53 of file ArnLib.cpp.

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

Definition at line 54 of file ArnLib.cpp.

13.1.2.25 `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 62 of file MQFlags.hpp.

14.1.2 Member Data Documentation

14.1.2.1 const char* Arn::_InitEnumTxt::enumTxt

Definition at line 65 of file MQFlags.hpp.

14.1.2.2 int Arn::_InitEnumTxt::enumVal

Definition at line 64 of file MQFlags.hpp.

14.1.2.3 int Arn::_InitEnumTxt::ns

Definition at line 63 of file MQFlags.hpp.

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

- [src/ArnInc/MQFlags.hpp \(3.0.0\)](#)

14.2 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 }

14.2.1 Detailed Description

Definition at line 191 of file Arn.hpp.

14.2.2 Member Enumeration Documentation

14.2.2.1 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 195 of file Arn.hpp.

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

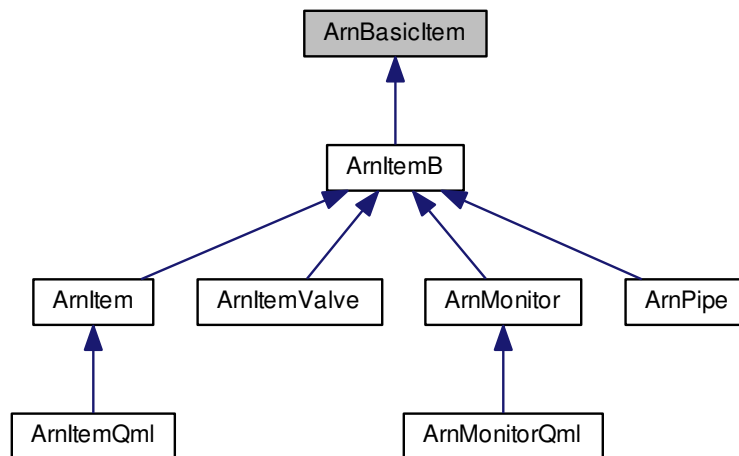
- src/ArnInc/[Arn.hpp \(3.0.0\)](#)

14.3 ArnBasicItem Class Reference

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

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

Inheritance 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
- `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`=0) const
- double `toDouble` (bool *`isOk`=0) const
- `ARNREAL` `toReal` (bool *`isOk`=0) const
- QString `toString` (bool *`isOk`=0) const
- QByteArray `toByteArray` (bool *`isOk`=0) const
- QVariant `toVariant` (bool *`isOk`=0) const
- bool `toBool` (bool *`isOk`=0) const
- uint `toUInt` (bool *`isOk`=0) const
- qint64 `toInt64` (bool *`isOk`=0) const
- quint64 `toUInt64` (bool *`isOk`=0) 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=` (qint64 `val`)
- `ArnBasicItem` & `operator=` (quint64 `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` (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
- 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](#).

Friends

- class [ArnBasicItemEventHandler](#)

14.3.1 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::ValueChange:
    {
        ArnEvValueChange* e = static_cast<ArnEvValueChange*>( 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 ArnEvValueChange: inItemPath=" << item->path()
        << " value=" << val;
    }
    default:
        break;
    }

    ArnBasicItemEventHandler::defaultEvent( ev);
}

```

Definition at line 119 of file ArnBasicItem.hpp.

14.3.2 Constructor & Destructor Documentation

14.3.2.1 ArnBasicItem::ArnBasicItem ()

Standard constructor of a closed handle.

Definition at line 82 of file ArnBasicItem.cpp.

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

Definition at line 96 of file ArnBasicItem.cpp.

14.3.3 Member Function Documentation

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

See also

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

Definition at line 387 of file ArnBasicItem.cpp.

14.3.3.2 QByteArray ArnBasicItem::arnExport () const

Returns

A data blob representing the *Arn Data Object*

See also

[arnImport\(\)](#)

Definition at line 579 of file ArnBasicItem.cpp.

14.3.3.3 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	data	is the data blob
in	ignoreSame	can override default ignoreSameValue setting.

See also

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

Definition at line 478 of file ArnBasicItem.cpp.

14.3.3.4 void ArnBasicItem::close ()

Close the handle.

Definition at line 144 of file ArnBasicItem.cpp.

14.3.3.5 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 always forcing global destroy.

Parameters

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

See also

[destroyLinkLocal\(\)](#)

Definition at line 169 of file ArnBasicItem.cpp.

14.3.3.6 void ArnBasicItem::destroyLinkLocal () [inline]

Destroy the local *Arn Data Object*

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

See also

[destroyLink\(\)](#)

Definition at line 155 of file ArnBasicItem.hpp.

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

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

Returns

the event handler

See also

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

Definition at line 1076 of file ArnBasicItem.cpp.

14.3.3.8 `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 405 of file ArnBasicItem.cpp.

14.3.3.9 `bool ArnBasicItem::isAutoDestroy () const`

Return values

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

See also

[setAutoDestroy\(\)](#)

Definition at line 381 of file ArnBasicItem.cpp.

14.3.3.10 `bool ArnBasicItem::isBiDirMode () const`

Return values

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

See also

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

Definition at line 292 of file ArnBasicItem.cpp.

14.3.3.11 `bool ArnBasicItem::isFolder () const`

Return values

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

Definition at line 181 of file ArnBasicItem.cpp.

14.3.3.12 `bool ArnBasicItem::isIgnoreSameValue () const`

Return values

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

See also

[setIgnoreSameValue\(\)](#)

Definition at line 430 of file ArnBasicItem.cpp.

14.3.3.13 `bool ArnBasicItem::isMaster () const`

Return values

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

See also

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

Definition at line 364 of file ArnBasicItem.cpp.

14.3.3.14 `bool ArnBasicItem::isOpen () const`

State of the handle.

Return values

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

Definition at line 175 of file ArnBasicItem.cpp.

14.3.3.15 `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 321 of file ArnBasicItem.cpp.

14.3.3.16 bool ArnBasicItem::isProvider () const

Return values

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

See also

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

Definition at line 189 of file ArnBasicItem.cpp.

14.3.3.17 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 343 of file ArnBasicItem.cpp.

14.3.3.18 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 470 of file ArnBasicItem.cpp.

14.3.3.19 `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 205 of file ArnBasicItem.cpp.

14.3.3.20 `QString ArnBasicItem::name (Arn::NameF nameF) const`

Name of the [Arn Data Object](#)

Parameters

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

Returns

The object name

Definition at line 446 of file ArnBasicItem.cpp.

14.3.3.21 `bool ArnBasicItem::open (const QString & path)`

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

Parameters

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

Return values

<code>false</code>	if error
--------------------	----------

Definition at line 138 of file ArnBasicItem.cpp.

14.3.3.22 `ArnBasicItem & ArnBasicItem::operator= (const ArnBasicItem & other)`

Definition at line 713 of file ArnBasicItem.cpp.

14.3.3.23 `ArnBasicItem & ArnBasicItem::operator= (int val)`

Definition at line 720 of file ArnBasicItem.cpp.

14.3.3.24 `ArnBasicItem & ArnBasicItem::operator= (ARNREAL val)`

Definition at line 727 of file ArnBasicItem.cpp.

14.3.3.25 ArnBasicItem & ArnBasicItem::operator= (const QString & val)

Definition at line 734 of file ArnBasicItem.cpp.

14.3.3.26 ArnBasicItem & ArnBasicItem::operator= (const QByteArray & val)

Definition at line 741 of file ArnBasicItem.cpp.

14.3.3.27 ArnBasicItem & ArnBasicItem::operator= (const QVariant & val)

Definition at line 776 of file ArnBasicItem.cpp.

14.3.3.28 ArnBasicItem & ArnBasicItem::operator= (const char * val)

Definition at line 748 of file ArnBasicItem.cpp.

14.3.3.29 ArnBasicItem & ArnBasicItem::operator= (uint val)

Definition at line 755 of file ArnBasicItem.cpp.

14.3.3.30 ArnBasicItem & ArnBasicItem::operator= (qint64 val)

Definition at line 762 of file ArnBasicItem.cpp.

14.3.3.31 ArnBasicItem & ArnBasicItem::operator= (quint64 val)

Definition at line 769 of file ArnBasicItem.cpp.

14.3.3.32 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 438 of file ArnBasicItem.cpp.

14.3.3.33 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 213 of file ArnBasicItem.cpp.

14.3.3.34 void * ArnBasicItem::reference () const

Get the stored external reference.

Returns

The associated external reference

See also

[setReference\(\)](#)

Definition at line 462 of file ArnBasicItem.cpp.

14.3.3.35 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 370 of file ArnBasicItem.cpp.

14.3.3.36 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 275 of file ArnBasicItem.cpp.

14.3.3.37 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 1067 of file ArnBasicItem.cpp.

14.3.3.38 void ArnBasicItem::setIgnoreSameValue (bool *isIgnore* = true)

Set skipping of equal value.

Parameters

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

Definition at line 422 of file ArnBasicItem.cpp.

14.3.3.39 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 353 of file ArnBasicItem.cpp.

14.3.3.40 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 302 of file ArnBasicItem.cpp.

14.3.3.41 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

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

See also

[reference\(\)](#)

Definition at line 454 of file ArnBasicItem.cpp.

14.3.3.42 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 331 of file ArnBasicItem.cpp.

14.3.3.43 void ArnBasicItem::setValue (const ArnBasicItem & *other*, int *ignoreSame* = Arn::SameValue::DefaultAction)

Definition at line 783 of file ArnBasicItem.cpp.

14.3.3.44 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 825 of file ArnBasicItem.cpp.

14.3.3.45 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 847 of file ArnBasicItem.cpp.

14.3.3.46 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
----	--------------	----------------

<i>in</i>	<i>ignoreSame</i>	can override default ignoreSameValue setting.
-----------	-------------------	---

See also

[setIgnoreSameValue\(\)](#)

Definition at line 869 of file ArnBasicItem.cpp.

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

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

Parameters

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

See also

[setIgnoreSameValue\(\)](#)

Definition at line 891 of file ArnBasicItem.cpp.

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

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

Parameters

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

See also

[setIgnoreSameValue\(\)](#)

Definition at line 913 of file ArnBasicItem.cpp.

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

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

Parameters

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

See also

[setIgnoreSameValue\(\)](#)

Definition at line 935 of file ArnBasicItem.cpp.

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

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

Parameters

in	value	to be assigned
in	ignoreSame	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 957 of file ArnBasicItem.cpp.

14.3.3.51 void ArnBasicItem::setValue (uint value, int ignoreSame = Arn::SameValue::DefaultAction)

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

Parameters

in	value	to be assigned
in	ignoreSame	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 963 of file ArnBasicItem.cpp.

14.3.3.52 void ArnBasicItem::setValue (quint64 value, int ignoreSame = Arn::SameValue::DefaultAction)

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

Parameters

in	value	to be assigned
in	ignoreSame	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Note

Not native ARN datatype. ByteArray is assigned.

Definition at line 975 of file ArnBasicItem.cpp.

14.3.3.53 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 987 of file ArnBasicItem.cpp.

14.3.3.54 `Arn::ObjectSyncMode ArnBasicItem::syncMode () const`

Returns

The client session *sync mode* of an [Arn Data Object](#)

See also

[Modes](#)

Definition at line 264 of file ArnBasicItem.cpp.

14.3.3.55 `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 999 of file ArnBasicItem.cpp.

14.3.3.56 `bool ArnBasicItem::toBool (bool * isOk = 0) 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 681 of file ArnBasicItem.cpp.

14.3.3.57 QByteArray ArnBasicItem::toByteArray (bool * isOk = 0) 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 641 of file ArnBasicItem.cpp.

14.3.3.58 double ArnBasicItem::toDouble (bool * isOk = 0) 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 665 of file ArnBasicItem.cpp.

14.3.3.59 int ArnBasicItem::toInt (bool * isOk = 0) 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 657 of file ArnBasicItem.cpp.

14.3.3.60 quint64 ArnBasicItem::toInt64 (bool * isOk = 0) 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 697 of file ArnBasicItem.cpp.

14.3.3.61 ARNREAL ArnBasicItem::toReal (bool * isOk = 0) 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 673 of file ArnBasicItem.cpp.

14.3.3.62 QString ArnBasicItem::toString (bool * isOk = 0) 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 633 of file ArnBasicItem.cpp.

14.3.3.63 uint ArnBasicItem::toUInt (bool * isOk = 0) 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 689 of file ArnBasicItem.cpp.

14.3.3.64 quint64 ArnBasicItem::toUInt64 (bool * isOk = 0) 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 705 of file ArnBasicItem.cpp.

14.3.3.65 `QVariant ArnBasicItem::toVariant (bool * isOk = 0) 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 649 of file ArnBasicItem.cpp.

14.3.3.66 `Arn::DataType ArnBasicItem::type () const`

The type stored in the [Arn Data Object](#)

Returns

The type stored

Definition at line 197 of file ArnBasicItem.cpp.

14.3.4 Friends And Related Function Documentation

14.3.4.1 `friend class ArnBasicItemEventHandler [friend]`

Definition at line 122 of file ArnBasicItem.hpp.

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

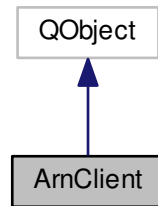
- [src/ArnInc/ArnBasicItem.hpp \(3.0.0\)](#)
- [src/ArnBasicItem.cpp \(3.0.0\)](#)

14.4 ArnClient Class Reference

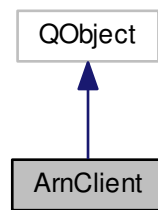
Class for connecting to an [Arn Server](#).

```
#include <ArnClient.hpp>
```

Inheritance diagram for ArnClient:



Collaboration diagram for ArnClient:



Classes

- struct [HostAddrPort](#)

Public Types

- typedef [ArnClientConnectStat](#) [ConnectStat](#)
- typedef [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=0)
- [~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.*

- QStringList [freePaths](#) () const
Returns current list of freePaths.
- void [setWholAm](#) (const [Arn::XStringMap](#) &wholAmXsm)
Set clients human readable identification information.
- [Arn::XStringMap](#) [remoteWholAm](#) () 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.

14.4.1 Detailed Description

Class for connecting to an [Arn](#) Server.

About Sharing Arn Data Objects

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 103 of file ArnClient.hpp.

14.4.2 Member Typedef Documentation

14.4.2.1 typedef ArnClientConnectStat ArnClient::ConnectStat

Definition at line 109 of file ArnClient.hpp.

14.4.2.2 typedef QList<HostAddrPort> ArnClient::HostList

Definition at line 119 of file ArnClient.hpp.

14.4.3 Constructor & Destructor Documentation

14.4.3.1 ArnClient::ArnClient (QObject * *parent* = 0) [explicit]

Definition at line 246 of file ArnClient.cpp.

14.4.3.2 ArnClient::~~ArnClient ()

Definition at line 262 of file ArnClient.cpp.

14.4.4 Member Function Documentation

14.4.4.1 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 593 of file ArnClient.cpp.

14.4.4.2 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 376 of file ArnClient.cpp.

14.4.4.3 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 286 of file ArnClient.cpp.

14.4.4.4 **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 278 of file ArnClient.cpp.

14.4.4.5 **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\(\)](#)

14.4.4.6 **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.
----	-------------	-----------------

<i>in</i>	<i>prioType</i>	is the priority of the message (1=Hi 2=Normal).
-----------	-----------------	---

See also

[chatReceived\(\)](#)

Definition at line 583 of file ArnClient.cpp.

14.4.4.7 void ArnClient::clearArnList (int *prioFilter* = -1)

Clear the [Arn](#) connection list.

Typically used to start making a new [Arn](#) connection list.

Parameters

<i>in</i>	<i>prioFilter</i>	selects hosts in the list with this pri, to be removed. Default -1 removes all.
-----------	-------------------	---

See also

[addToArnList\(\)](#)

Definition at line 270 of file ArnClient.cpp.

14.4.4.8 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 346 of file ArnClient.cpp.

14.4.4.9 void ArnClient::connectionStatusChanged (int *status*, int *curPrio*) [signal]

Signal emitted when the connection status is changed.

Parameters

<i>in</i>	<i>status</i>	is the new connection status ArnClient::ConnectStat .
<i>in</i>	<i>curPrio</i>	is the current priority of the connection in ArnList

See also

[curPrio\(\)](#)

14.4.4.10 ArnClient::ConnectStat ArnClient::connectStatus () const

Return the [Arn](#) connection status.

Return values

<i>the</i>	Arn connection status.
------------	--

Definition at line 355 of file ArnClient.cpp.

14.4.4.11 `void ArnClient::connectToArn (const QString & arnHost, quint16 port = 0)`

Connect to an [Arn Server](#)

Parameters

<i>in</i>	<i>arnHost</i>	is host name or ip address, e.g. "192.168.1.1".
<i>in</i>	<i>port</i>	is the host port, 0 gives Arn::defaultTcpPort .

See also

[Arn::makeHostWithInfo\(\)](#)
[connectToArnList\(\)](#)

Definition at line 307 of file ArnClient.cpp.

14.4.4.12 `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 294 of file ArnClient.cpp.

14.4.4.13 `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 synchronized 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 320 of file ArnClient.cpp.

14.4.4.14 `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 transferred to [ArnServer](#). [ArnServer](#) still decides what's allowed. The list is automatically transferred from [ArnServer](#) to [ArnClient](#) during the negotiation phase.

Returns

the freePath list.

See also

[ArnServer::addFreePath\(\)](#)

Definition at line 535 of file ArnClient.cpp.

14.4.4.15 `ArnClient * ArnClient::getClient (const QString & id) [static]`

Get a client by its id.

Parameters

<i>in</i>	<i>id</i>	if "" will always return 0.
-----------	-----------	-----------------------------

Returns

the found client, 0 = not found or id == ""

See also

[registerClient\(\)](#)

Definition at line 483 of file ArnClient.cpp.

14.4.4.16 `bool ArnClient::getTraffic (quint64 & in, quint64 & out) const`

Get traffic metrics.

Return values

<i>true</i>	if ok.
-------------	--------

Parameters

<i>out</i>	<i>in</i>	is the clients received number of bytes.
<i>out</i>	<i>out</i>	is the clients sent number of bytes.

Definition at line 601 of file ArnClient.cpp.

14.4.4.17 `QString ArnClient::id () const`

Get the id of this client.

Returns

the id, "" = none (local)

See also

[registerClient\(\)](#)

Definition at line 489 of file ArnClient.cpp.

14.4.4.18 `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 513 of file ArnClient.cpp.

14.4.4.19 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 567 of file ArnClient.cpp.

14.4.4.20 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

<i>true</i>	if this is a reContact.
-------------	-------------------------

See also

[isReConnect\(\)](#)
[setAutoConnect\(\)](#)
[connectionStatusChanged\(\)](#)

Definition at line 559 of file ArnClient.cpp.

14.4.4.21 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\(\)](#)

14.4.4.22 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\(\)](#)

14.4.4.23 **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 automatical 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 330 of file ArnClient.cpp.

14.4.4.24 **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 337 of file ArnClient.cpp.

14.4.4.25 **QString** ArnClient::passwordHash (const QString & *password*) [static]

Generate a hashed password from clear text password.

Parameters

<i>in</i>	<i>password</i>	is the clear text password.
-----------	-----------------	-----------------------------

Returns

the hashed password, e.g. "{A5ha62Aug}"

Definition at line 529 of file ArnClient.cpp.

14.4.4.26 int ArnClient::receiveTimeout () const

Get receive data timeout (base time)

Returns

the timeout in seconds

See also

[setReceiveTimeout\(\)](#)

Definition at line 497 of file ArnClient.cpp.

14.4.4.27 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

<i>in</i>	<i>id</i>	must not be "".
-----------	-----------	-----------------

See also

[getClient\(\)](#)
[id\(\)](#)

Definition at line 472 of file ArnClient.cpp.

14.4.4.28 Arn::XStringMap ArnClient::remoteWholAm () const

Returns remote side (server) readable identification information.

This is used to identify the server side in session.

Returns

the inforamtion.

See also

[setWholAm\(\)](#)

Definition at line 551 of file ArnClient.cpp.

14.4.4.29 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 431 of file ArnClient.cpp.

14.4.4.30 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 463 of file ArnClient.cpp.

14.4.4.31 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 521 of file ArnClient.cpp.

14.4.4.32 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

<i>in</i>	<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 363 of file ArnClient.cpp.

14.4.4.33 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

<i>in</i>	<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 505 of file ArnClient.cpp.

14.4.4.34 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 543 of file ArnClient.cpp.

14.4.4.35 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.

14.4.4.36 void ArnClient::tcpDisConnected () [signal]

Signal emitted when the tcp connection is broken (has been successfull).

14.4.4.37 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 \(3.0.0\)](#)
- [src/ArnClient.cpp \(3.0.0\)](#)

14.5 ArnClientConnectStat Class Reference

```
#include <ArnClient.hpp>
```

Public Types

- enum [E](#) {
[Init](#) = 0, [Connecting](#), [Negotiating](#), [Connected](#),
[Stopped](#), [Error](#), [Disconnected](#), [TriedAll](#) }
- enum [NS](#) { [NsEnum](#), [NsHuman](#) }

14.5.1 Detailed Description

Definition at line 49 of file ArnClient.hpp.

14.5.2 Member Enumeration Documentation

14.5.2.1 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.

14.5.2.2 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 \(3.0.0\)](#)

14.6 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](#) ()

14.6.1 Detailed Description

Definition at line 48 of file ArnClient.cpp.

14.6.2 Member Function Documentation

14.6.2.1 [ArnClient](#) * ArnClientReg::get (const QString & id)

Definition at line 79 of file ArnClient.cpp.

14.6.2.2 **ArnClientReg** & **ArnClientReg::instance** () [static]

Definition at line 114 of file ArnClient.cpp.

14.6.2.3 **int** **ArnClientReg::remove** (**const** **QString** & *id*)

Definition at line 87 of file ArnClient.cpp.

14.6.2.4 **int** **ArnClientReg::remove** (**const** **ArnClient** * *client*)

Definition at line 95 of file ArnClient.cpp.

14.6.2.5 **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:

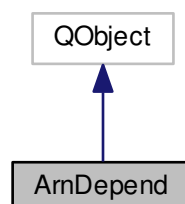
- [src/ArnClient.cpp \(3.0.0\)](#)

14.7 **ArnDepend** Class Reference

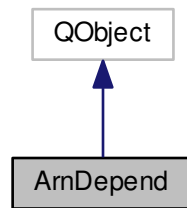
Class for setting up dependencis to needed services.

```
#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=0)
- [~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.

14.7.1 Detailed Description

Class for setting up dependencies 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.

14.7.2 Member Typedef Documentation

14.7.2.1 typedef ArnDependSlot ArnDepend::DepSlot

Definition at line 138 of file ArnDepend.hpp.

14.7.3 Constructor & Destructor Documentation

14.7.3.1 ArnDepend::ArnDepend (QObject * *parent* = 0) [explicit]

Definition at line 169 of file ArnDepend.cpp.

14.7.3.2 ArnDepend::~~ArnDepend ()

Definition at line 185 of file ArnDepend.cpp.

14.7.4 Member Function Documentation

14.7.4.1 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.

14.7.4.2 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.

14.7.4.3 void ArnDepend::completed () [signal]

Signal emitted when all dependent services are available.

14.7.4.4 void ArnDepend::setMonitorName (const QString & *name*)

Set an optional monitor name for debugging.

Parameters

<code>in</code>	<code>name</code>	is the monitor name.
-----------------	-------------------	----------------------

Definition at line 229 of file ArnDepend.cpp.

14.7.4.5 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:

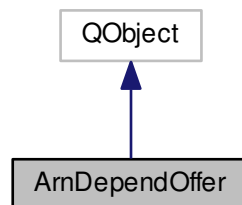
- [src/ArnInc/ArnDepend.hpp \(3.0.0\)](#)
- [src/ArnDepend.cpp \(3.0.0\)](#)

14.8 ArnDependOffer Class Reference

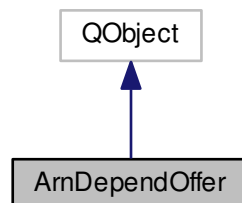
Class for advertising that a *service* is available.

```
#include <ArnDepend.hpp>
```

Inheritance diagram for ArnDependOffer:



Collaboration diagram for ArnDependOffer:



Public Member Functions

- [ArnDependOffer](#) (QObject *parent=0)
- [~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

14.8.1 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.

14.8.2 Constructor & Destructor Documentation

14.8.2.1 ArnDependOffer::ArnDependOffer (QObject * parent = 0) [explicit]

Definition at line 56 of file ArnDepend.cpp.

14.8.2.2 ArnDependOffer::~~ArnDependOffer ()

Definition at line 70 of file ArnDepend.cpp.

14.8.3 Member Function Documentation

14.8.3.1 void ArnDependOffer::advertise (const QString & serviceName)

Advertise an available *service*

Parameters

<code>in</code>	<code>serviceName</code>	is the name of the <i>service</i> .
-----------------	--------------------------	-------------------------------------

Definition at line 76 of file ArnDepend.cpp.

14.8.3.2 void ArnDependOffer::setStateId (int id)

Set the *state* of the *service* by an id number.

The *state* starts of by 0 as default.

Parameters

<code>in</code>	<code>id</code>	is the <i>state</i> id number.
-----------------	-----------------	--------------------------------

Definition at line 114 of file ArnDepend.cpp.

14.8.3.3 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

<code>in</code>	<code>name</code>	is the <i>state</i> name.
-----------------	-------------------	---------------------------

Definition at line 98 of file ArnDepend.cpp.

14.8.3.4 int ArnDependOffer::stateld () const

Returns

The *state* id number.

See also

[setStateId\(\)](#)

Definition at line 122 of file ArnDepend.cpp.

14.8.3.5 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:

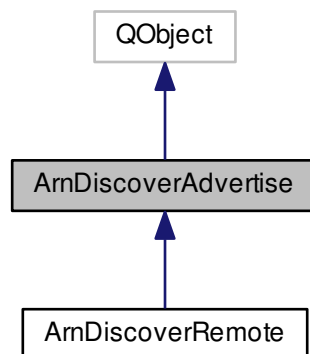
- [src/ArnInc/ArnDepend.hpp \(3.0.0\)](#)
- [src/ArnDepend.cpp \(3.0.0\)](#)

14.9 ArnDiscoverAdvertise Class Reference

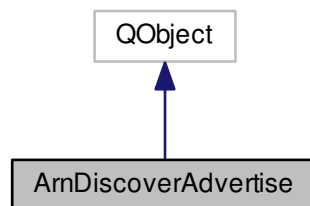
Advertise an [Arn](#) service.

```
#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 unsuccessfull advertise of service.

Public Member Functions

- [ArnDiscoverAdvertise](#) (QObject *parent=0)
- [~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.

14.9.1 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 628 of file ArnDiscover.hpp.

14.9.2 Constructor & Destructor Documentation

14.9.2.1 ArnDiscoverAdvertise::ArnDiscoverAdvertise (QObject * parent = 0) [explicit]

Definition at line 774 of file ArnDiscover.cpp.

14.9.2.2 ArnDiscoverAdvertise::~~ArnDiscoverAdvertise ()

Definition at line 790 of file ArnDiscover.cpp.

14.9.3 Member Function Documentation

14.9.3.1 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 872 of file ArnDiscover.cpp.

14.9.3.2 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 946 of file ArnDiscover.cpp.

14.9.3.3 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 797 of file ArnDiscover.cpp.

14.9.3.4 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 896 of file ArnDiscover.cpp.

14.9.3.5 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 856 of file ArnDiscover.cpp.

14.9.3.6 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 930 of file ArnDiscover.cpp.

14.9.3.7 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 888 of file ArnDiscover.cpp.

14.9.3.8 void ArnDiscoverAdvertise::serviceChanged (const QString & *serviceName*) [signal]

Indicate successfull advertise of service.

Parameters

<i>in</i>	<i>serviceName</i>	is the really advertised name e.g. "My House Registry (2)"
-----------	--------------------	--

See also

[advertiseService\(\)](#)
[setService\(\)](#)

14.9.3.9 void ArnDiscoverAdvertise::serviceChangeError (int *code*) [signal]

Indicate unsuccessful advertise of service.

Parameters

<i>in</i>	<i>code</i>	error code.
-----------	-------------	-------------

See also

[advertiseService\(\)](#)

14.9.3.10 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

<code>in</code>	<i>custom\leftrightarrow Properties</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 864 of file ArnDiscover.cpp.

14.9.3.11 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

<code>in</code>	<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 938 of file ArnDiscover.cpp.

14.9.3.12 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

<code>in</code>	<i>service</i>	is the requested service name e.g. "My House Registry"
-----------------	----------------	--

See also

[service\(\)](#)
[currentService\(\)](#)
[advertiseService\(\)](#)
[serviceChanged\(\)](#)
[serviceChangeError\(\)](#)

Definition at line 912 of file ArnDiscover.cpp.

14.9.3.13 ArnDiscoverAdvertise::State ArnDiscoverAdvertise::state () const

Returns the state for this Advertise.

Returns

current state

See also

[State](#)

Definition at line 904 of file ArnDiscover.cpp.

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

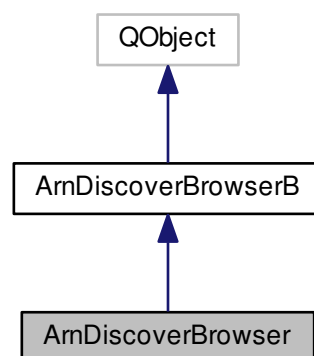
- [src/ArnInc/ArnDiscover.hpp \(3.0.0\)](#)
- [src/ArnDiscover.cpp \(3.0.0\)](#)

14.10 ArnDiscoverBrowser Class Reference

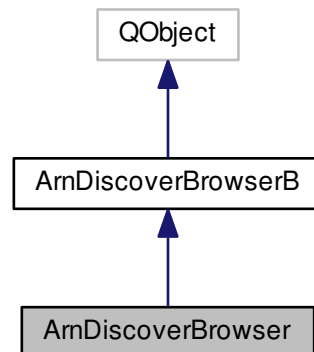
Browsing for [Arn](#) services.

```
#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=0)
- 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

14.10.1 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 476 of file ArnDiscover.hpp.

14.10.2 Constructor & Destructor Documentation

14.10.2.1 ArnDiscoverBrowser::ArnDiscoverBrowser (QObject * *parent* = 0) [explicit]

Definition at line 236 of file ArnDiscover.cpp.

14.10.3 Member Function Documentation

14.10.3.1 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 515 of file ArnDiscover.hpp.

14.10.3.2 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 486 of file ArnDiscover.hpp.

14.10.3.3 `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

<i>in</i>	<i>typeFilter</i>	
-----------	-------------------	--

See also

[ArnDiscoverAdvertise::advertiseService\(\)](#)

Definition at line 495 of file ArnDiscover.hpp.

14.10.3.4 `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

<i>in</i>	<i>group</i>	the filter group name, e.g. "myGroup1"
-----------	--------------	--

See also

[ArnDiscoverAdvertise::setGroups\(\)](#)

Definition at line 505 of file ArnDiscover.hpp.

14.10.3.5 `void ArnDiscoverBrowser::stopBrowse () [inline],[slot]`

Stop browsing.

See also

[browse\(\)](#)

Definition at line 521 of file ArnDiscover.hpp.

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

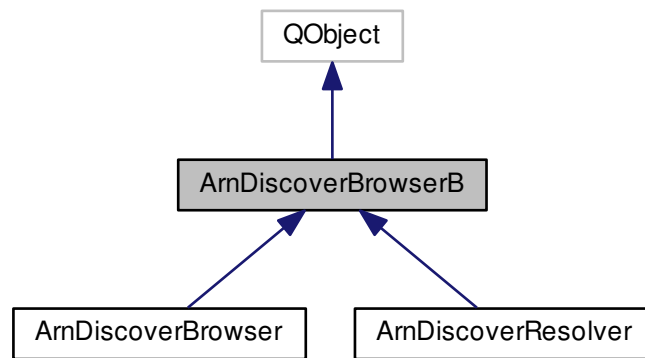
- [src/ArnInc/ArnDiscover.hpp \(3.0.0\)](#)
- [src/ArnDiscover.cpp \(3.0.0\)](#)

14.11 ArnDiscoverBrowserB Class Reference

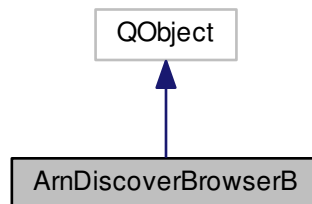
Browse() and resolve() together, may never be used to the same instance.

```
#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=0)

- [~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.

14.11.1 Detailed Description

Browse() and resolve() together, may never be used to the same instance.

Definition at line 224 of file ArnDiscover.hpp.

14.11.2 Constructor & Destructor Documentation

14.11.2.1 [ArnDiscoverBrowserB::ArnDiscoverBrowserB \(QObject * parent = 0 \)](#) [explicit]

Definition at line 320 of file ArnDiscover.cpp.

14.11.2.2 [ArnDiscoverBrowserB::~~ArnDiscoverBrowserB \(\)](#)

Definition at line 328 of file ArnDiscover.cpp.

14.11.3 Member Function Documentation

14.11.3.1 [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 437 of file ArnDiscover.cpp.

14.11.3.2 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

<i>in</i>	<i>index</i>	for the service
<i>in</i>	<i>state</i>	is the wanted final state

See also

[defaultStopState\(\)](#)
[infoUpdated\(\)](#)
[ArnDiscoverInfo::stopState\(\)](#)
State

Definition at line 453 of file ArnDiscover.cpp.

14.11.3.3 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

<i>in</i>	<i>id</i>	
-----------	-----------	--

Returns

selected service discover index

See also

[indexTold\(\)](#)
[infoByIndex\(\)](#)

Definition at line 387 of file ArnDiscover.cpp.

14.11.3.4 int ArnDiscoverBrowserB::indexTold (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

<i>in</i>	<i>index</i>	
-----------	--------------	--

Returns

selected service discover id

See also

[IdToIndex\(\)](#)
[infoById\(\)](#)

Definition at line 377 of file ArnDiscover.cpp.

14.11.3.5 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

<i>in</i>	<i>id</i>	
-----------	-----------	--

Returns

selected service discover info

See also

[infoByIndex\(\)](#)

Definition at line 362 of file ArnDiscover.cpp.

14.11.3.6 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

<i>in</i>	<i>index</i>	
-----------	--------------	--

Returns

selected service discover info

See also

[infoById\(\)](#)
[infoByName\(\)](#)
[indexTold\(\)](#)

Definition at line 350 of file ArnDiscover.cpp.

14.11.3.7 `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

<i>in</i>	<i>serviceName</i>	
-----------	--------------------	--

Returns

selected service discover info

See also

[serviceNameTold\(\)](#)

Definition at line 371 of file ArnDiscover.cpp.

14.11.3.8 `void ArnDiscoverBrowserB::infoUpdated (int index, ArnDiscoverInfo::State state) [signal]`

Indicate service has been updated.

Parameters

<i>in</i>	<i>index</i>	for the service
<i>in</i>	<i>state</i>	is the current state of the service info

See also

[goTowardState\(\)](#)
[serviceAdded\(\)](#)

14.11.3.9 `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

<i>in</i>	<i>index</i>	for the service
<i>in</i>	<i>name</i>	is the service name e.g. "My House Registry"

See also

[serviceRemoved\(\)](#)
[infoUpdated\(\)](#)

14.11.3.10 `int ArnDiscoverBrowserB::serviceCount () const`

Return the number of active discover services.

Returns

number of services

Definition at line 342 of file ArnDiscover.cpp.

14.11.3.11 `int ArnDiscoverBrowserB::serviceNameTold (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

<i>in</i>	<i>name</i>	
-----------	-------------	--

Returns

selected service discover id

See also

[IdToIndex\(\)](#)
[infoByName\(\)](#)

Definition at line 395 of file ArnDiscover.cpp.

14.11.3.12 `void ArnDiscoverBrowserB::serviceRemoved (int index) [signal]`

Indicate service has been removed.

Parameters

<i>in</i>	<i>index</i>	for the service
-----------	--------------	-----------------

See also

[serviceAdded\(\)](#)

14.11.3.13 `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

<i>in</i>	<i>defaultStopState</i>	
-----------	-------------------------	--

See also

[defaultStopState\(\)](#)
[goTowardState\(\)](#)
[ArnDiscoverInfo::stopState\(\)](#)
[State](#)

Definition at line 445 of file ArnDiscover.cpp.

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

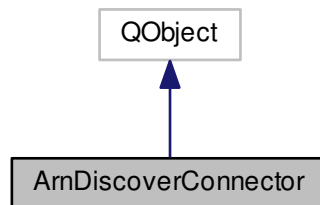
- [src/ArnInc/ArnDiscover.hpp \(3.0.0\)](#)
- [src/ArnDiscover.cpp \(3.0.0\)](#)

14.12 ArnDiscoverConnector Class Reference

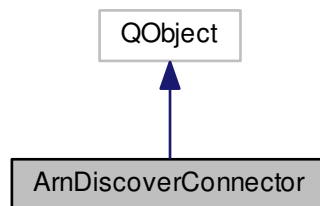
An automatic client discover connector.

```
#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.*

14.12.1 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.

14.12.2 Constructor & Destructor Documentation

14.12.2.1 ArnDiscoverConnector::ArnDiscoverConnector (ArnClient & *client*, const QString & *id*)

Definition at line 70 of file ArnDiscoverConnect.cpp.

14.12.2.2 ArnDiscoverConnector::~ArnDiscoverConnector ()

Definition at line 93 of file ArnDiscoverConnect.cpp.

14.12.3 Member Function Documentation

14.12.3.1 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.

14.12.3.2 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.

14.12.3.3 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 "Weather↵Data-XYZ"

See also

[ArnDiscoverRemote::newConnector\(\)](#)
[setExternalClientConnect\(\)](#)

14.12.3.4 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.

14.12.3.5 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.

14.12.3.6 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.

14.12.3.7 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.

14.12.3.8 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.

14.12.3.9 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.

14.12.3.10 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

<i>in</i>	<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.

14.12.3.11 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.

14.12.3.12 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>externalClient↔ Connect</i>	true to activate.
----	------------------------------------	-------------------

See also

[externalClientConnect\(\)](#)

Definition at line 201 of file ArnDiscoverConnect.cpp.

14.12.3.13 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 previos 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.

14.12.3.14 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.

14.12.3.15 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.

14.12.3.16 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 \(3.0.0\)](#)
- [src/ArnDiscoverConnect.cpp \(3.0.0\)](#)

14.13 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 [resolveCode](#) () const
Return the latest resolv error code for this service.

Friends

- class [ArnDiscoverBrowserB](#)

14.13.1 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.

14.13.2 Constructor & Destructor Documentation

14.13.2.1 ArnDiscoverInfo::ArnDiscoverInfo ()

Definition at line 59 of file ArnDiscover.cpp.

14.13.2.2 ArnDiscoverInfo::ArnDiscoverInfo (const ArnDiscoverInfo & other)

Definition at line 71 of file ArnDiscover.cpp.

14.13.2.3 ArnDiscoverInfo::~ArnDiscoverInfo ()

Definition at line 85 of file ArnDiscover.cpp.

14.13.3 Member Function Documentation

14.13.3.1 QString ArnDiscoverInfo::domain () const

Return the domain for this service.

Returns

domain, e.g. "local."

Definition at line 148 of file ArnDiscover.cpp.

14.13.3.2 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 132 of file ArnDiscover.cpp.

14.13.3.3 QHostAddress ArnDiscoverInfo::hostIp () const

Return the host ip-address for this service.

Returns

host ip-address

Definition at line 172 of file ArnDiscover.cpp.

14.13.3.4 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 212 of file ArnDiscover.cpp.

14.13.3.5 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 156 of file ArnDiscover.cpp.

14.13.3.6 quint16 ArnDiscoverInfo::hostPort () const

Return the port for this service.

Returns

port

See also

[ArnDiscoverAdvertise::advertiseService\(\)](#)

Definition at line 164 of file ArnDiscover.cpp.

14.13.3.7 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 204 of file ArnDiscover.cpp.

14.13.3.8 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 220 of file ArnDiscover.cpp.

14.13.3.9 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 92 of file ArnDiscover.cpp.

14.13.3.10 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 100 of file ArnDiscover.cpp.

14.13.3.11 ArnDiscoverInfo & ArnDiscoverInfo::operator= (const ArnDiscoverInfo & other)

Definition at line 77 of file ArnDiscover.cpp.

14.13.3.12 **XStringMap ArnDiscoverInfo::properties () const**

Return the properties for this service.

Will return booth [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 180 of file ArnDiscover.cpp.

14.13.3.13 **int ArnDiscoverInfo::resolveCode () const**

Return the latest resolve error code for this service.

This code can come from booth resolving a service and lookup ip-address.

Returns

error code

See also

[ArnZeroConf::Error](#)

Definition at line 226 of file ArnDiscover.cpp.

14.13.3.14 **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 140 of file ArnDiscover.cpp.

14.13.3.15 **ArnDiscoverInfo::State ArnDiscoverInfo::state () const**

Return the state for this service.

Returns

state

See also

[State](#)

Definition at line 108 of file ArnDiscover.cpp.

14.13.3.16 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 116 of file ArnDiscover.cpp.

14.13.3.17 ArnDiscover::Type ArnDiscoverInfo::type () const

Return the discover type for this service.

Returns

discover type

See also

[Type](#)
[ArnDiscoverAdvertise::advertiseService\(\)](#)

Definition at line 124 of file ArnDiscover.cpp.

14.13.3.18 QString ArnDiscoverInfo::typeString () const

Return the printable type for this service.

Returns

type, e.g. "Client"

Definition at line 188 of file ArnDiscover.cpp.

14.13.4 Friends And Related Function Documentation

14.13.4.1 friend class ArnDiscoverBrowserB [friend]

Definition at line 75 of file ArnDiscover.hpp.

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

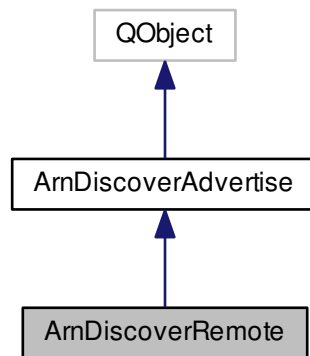
- [src/ArnInc/ArnDiscover.hpp \(3.0.0\)](#)
- [src/ArnDiscover.cpp \(3.0.0\)](#)

14.14 ArnDiscoverRemote Class Reference

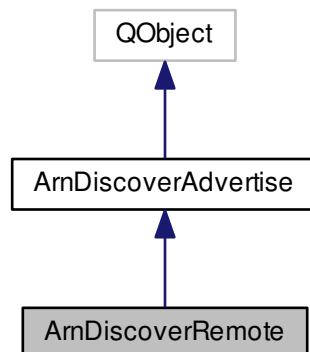
Discover with remote setting.

```
#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=0)
- [~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.

14.14.1 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](#).

14.14.2 Constructor & Destructor Documentation

14.14.2.1 ArnDiscoverRemote::ArnDiscoverRemote (QObject *parent = 0) [explicit]

Definition at line 63 of file [ArnDiscoverRemote.cpp](#).

14.14.2.2 ArnDiscoverRemote::~ArnDiscoverRemote ()

Definition at line 75 of file [ArnDiscoverRemote.cpp](#).

14.14.3 Member Function Documentation

14.14.3.1 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\(\)](#)

14.14.3.2 QString ArnDiscoverRemote::defaultService () const

Return the default service name.

Returns

default service name, e.g. "Arn Default Service"

See also

[setDefaultService\(\)](#)

Definition at line 232 of file ArnDiscoverRemote.cpp.

14.14.3.3 int ArnDiscoverRemote::initialServiceTimeout () const

Return the time for initial timeout processing.

Returns

time in seconds

See also

[setInitialServiceTimeout\(\)](#)

Definition at line 249 of file ArnDiscoverRemote.cpp.

14.14.3.4 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 135 of file ArnDiscoverRemote.cpp.

14.14.3.5 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 240 of file ArnDiscoverRemote.cpp.

14.14.3.6 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>initialService↔ Timeout</i>	in seconds
----	------------------------------------	------------

See also

[initialServiceTimeout\(\)](#)

Definition at line 257 of file ArnDiscoverRemote.cpp.

14.14.3.7 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 215 of file ArnDiscoverRemote.cpp.

14.14.3.8 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 an 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 122 of file ArnDiscoverRemote.cpp.

14.14.3.9 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:

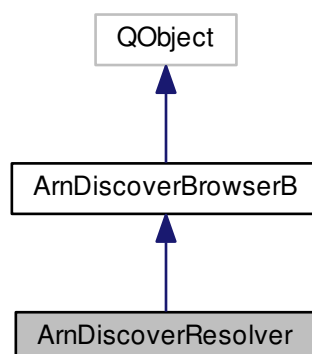
- [src/ArnInc/ArnDiscoverRemote.hpp \(3.0.0\)](#)
- [src/ArnDiscoverRemote.cpp \(3.0.0\)](#)

14.15 ArnDiscoverResolver Class Reference

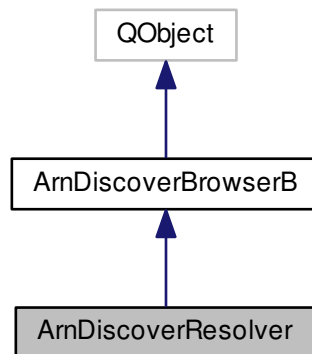
Resolv an [Arn](#) service.

```
#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=0)
- QString [defaultService](#) () const
Return the default service name.
- void [setDefaultService](#) (const QString &defaultService)
Set the default service name.

Additional Inherited Members

14.15.1 Detailed Description

Resolve 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 555 of file ArnDiscover.hpp.

14.15.2 Constructor & Destructor Documentation

14.15.2.1 `ArnDiscoverResolver::ArnDiscoverResolver (QObject * parent = 0)` `[explicit]`

Definition at line 255 of file ArnDiscover.cpp.

14.15.3 Member Function Documentation

14.15.3.1 `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 275 of file ArnDiscover.cpp.

14.15.3.2 `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

<i>in</i>	<i>serviceName</i>	is the service to be resolved
<i>in</i>	<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 267 of file ArnDiscover.cpp.

14.15.3.3 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

<i>in</i>	<i>defaultService</i>	e.g. "My Default Service"
-----------	-----------------------	---------------------------

See also

[defaultService\(\)](#)
[resolve\(\)](#)

Definition at line 283 of file ArnDiscover.cpp.

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

- src/ArnInc/ArnDiscover.hpp (3.0.0)
- src/ArnDiscover.cpp (3.0.0)

14.16 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](#) }

14.16.1 Detailed Description

Definition at line 38 of file ArnError.hpp.

14.16.2 Member Enumeration Documentation

14.16.2.1 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.

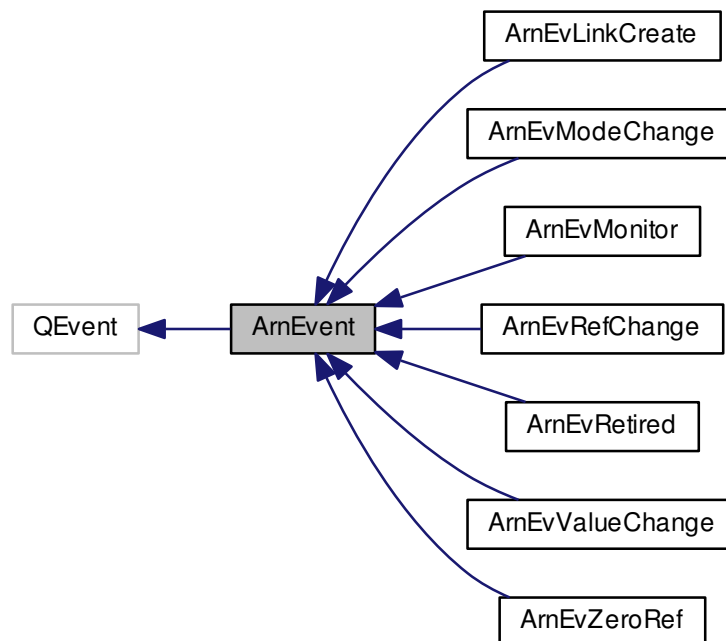
The documentation for this class was generated from the following file:

- [src/ArnInc/ArnError.hpp \(3.0.0\)](#)

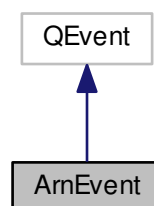
14.17 ArnEvent Class Reference

```
#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=0)
- 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)

14.17.1 Detailed Description

Definition at line 66 of file ArnEvent.hpp.

14.17.2 Member Typedef Documentation

14.17.2.1 typedef ArnEventIdx ArnEvent::Idx

Definition at line 76 of file ArnEvent.hpp.

14.17.3 Constructor & Destructor Documentation

14.17.3.1 ArnEvent::ArnEvent (QEvent::Type *type*)

Definition at line 37 of file ArnEvent.cpp.

14.17.3.2 ArnEvent::~~ArnEvent () [virtual]

Definition at line 47 of file ArnEvent.cpp.

14.17.4 Member Function Documentation

14.17.4.1 int ArnEvent::baseType (int *setVal* = -1) [static]

Definition at line 61 of file ArnEvent.cpp.

14.17.4.2 ArnEvent * ArnEvent::copyOpt (const ArnEvent * *other*) [protected]

Definition at line 127 of file ArnEvent.cpp.

14.17.4.3 void ArnEvent::inhibitPendingChain ()

Definition at line 165 of file ArnEvent.cpp.

14.17.4.4 bool ArnEvent::isArnEvent (int *evType*) [static]

Definition at line 87 of file ArnEvent.cpp.

14.17.4.5 virtual ArnEvent* ArnEvent::makeHeapClone () [pure virtual]

Implemented in [ArnEvRefChange](#), [ArnEvValueChange](#), [ArnEvZeroRef](#), [ArnEvRetired](#), [ArnEvMonitor](#), [ArnEvModeChange](#), and [ArnEvLinkCreate](#).

14.17.4.6 void ArnEvent::setTarget (void * *target*)

Definition at line 136 of file ArnEvent.cpp.

14.17.4.7 void ArnEvent::setTargetMutex (QMutex * *targetMutex*)

Definition at line 159 of file ArnEvent.cpp.

14.17.4.8 void ArnEvent::setTargetPendingChain (ArnEvent ** *targetPendingChain* = 0)

Definition at line 142 of file ArnEvent.cpp.

14.17.4.9 void* ArnEvent::target () const [inline]

Definition at line 91 of file ArnEvent.hpp.

14.17.4.10 int ArnEvent::toldx (QEvent::Type *type*) [static]

Definition at line 100 of file ArnEvent.cpp.

14.17.4.11 int ArnEvent::toldx () const

Definition at line 107 of file ArnEvent.cpp.

14.17.4.12 QString ArnEvent::toString (QEvent::Type *type*) [static]

Definition at line 114 of file ArnEvent.cpp.

14.17.4.13 QString ArnEvent::toString () const

Definition at line 121 of file ArnEvent.cpp.

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

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

14.18 ArnEventIdx Class Reference

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

Public Types

- enum [E](#) {
 [QtEvent](#) = -1, [ValueChange](#) = 0, [LinkCreate](#), [ModeChange](#),
 [Monitor](#), [Retired](#), [ZeroRef](#), [RefChange](#),
 [N](#) }

14.18.1 Detailed Description

Definition at line 46 of file ArnEvent.hpp.

14.18.2 Member Enumeration Documentation

14.18.2.1 enum ArnEventIdx::E

Enumerator

QtEvent

ValueChange

LinkCreate

ModeChange

Monitor

Retired

ZeroRef

RefChange

N Max index.

Definition at line 50 of file ArnEvent.hpp.

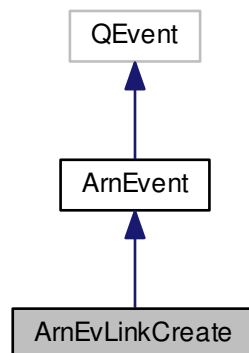
The documentation for this class was generated from the following file:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)

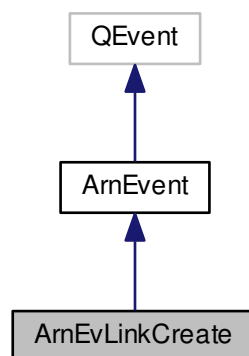
14.19 ArnEvLinkCreate Class Reference

```
#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

14.19.1 Detailed Description

Definition at line 104 of file ArnEvent.hpp.

14.19.2 Constructor & Destructor Documentation

14.19.2.1 ArnEvLinkCreate::ArnEvLinkCreate (const QString & *path*, ArnLink * *arnLink*, bool *isLastLink*)

Definition at line 226 of file ArnEvent.cpp.

14.19.3 Member Function Documentation

14.19.3.1 ArnLink* ArnEvLinkCreate::arnLink () const [inline]

Definition at line 118 of file ArnEvent.hpp.

14.19.3.2 bool ArnEvLinkCreate::isLastLink () const [inline]

Definition at line 121 of file ArnEvent.hpp.

14.19.3.3 ArnEvent* ArnEvLinkCreate::makeHeapClone () [virtual]

Implements [ArnEvent](#).

Definition at line 243 of file ArnEvent.cpp.

14.19.3.4 const QString& ArnEvLinkCreate::path () const [inline]

Definition at line 115 of file ArnEvent.hpp.

14.19.3.5 QEvent::Type ArnEvLinkCreate::type () [static]

Definition at line 235 of file ArnEvent.cpp.

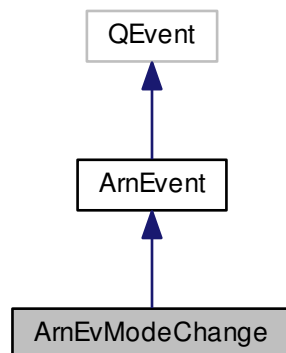
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

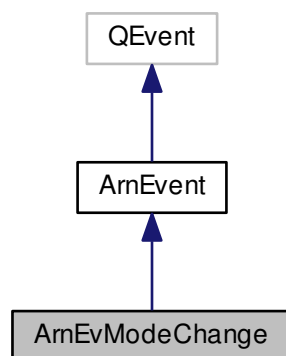
14.20 ArnEvModeChange Class Reference

```
#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

14.20.1 Detailed Description

Definition at line 126 of file ArnEvent.hpp.

14.20.2 Constructor & Destructor Documentation

14.20.2.1 `ArnEvModeChange::ArnEvModeChange (const QString & path, uint linkId, Arn::ObjectMode mode)`

Definition at line 250 of file ArnEvent.cpp.

14.20.3 Member Function Documentation

14.20.3.1 `uint ArnEvModeChange::linkId () const` `[inline]`

Definition at line 140 of file ArnEvent.hpp.

14.20.3.2 `ArnEvent * ArnEvModeChange::makeHeapClone ()` `[virtual]`

Implements [ArnEvent](#).

Definition at line 267 of file ArnEvent.cpp.

14.20.3.3 `Arn::ObjectMode ArnEvModeChange::mode () const` `[inline]`

Definition at line 143 of file ArnEvent.hpp.

14.20.3.4 `const QString& ArnEvModeChange::path () const` `[inline]`

Definition at line 137 of file ArnEvent.hpp.

14.20.3.5 `QEvent::Type ArnEvModeChange::type ()` `[static]`

Definition at line 259 of file ArnEvent.cpp.

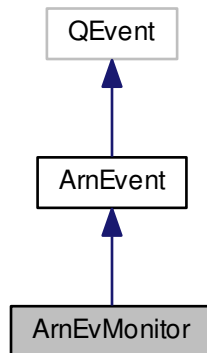
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

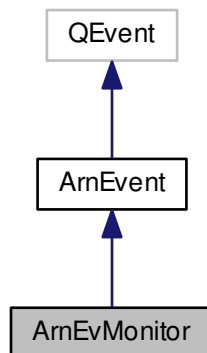
14.21 ArnEvMonitor Class Reference

```
#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

14.21.1 Detailed Description

Definition at line 148 of file ArnEvent.hpp.

14.21.2 Constructor & Destructor Documentation

14.21.2.1 `ArnEvMonitor::ArnEvMonitor (int monEvType, const QByteArray & data, bool isLocal, void * sessionHandler)`

Definition at line 274 of file ArnEvent.cpp.

14.21.3 Member Function Documentation

14.21.3.1 `const QByteArray& ArnEvMonitor::data () const` `[inline]`

Definition at line 163 of file ArnEvent.hpp.

14.21.3.2 `bool ArnEvMonitor::isLocal () const` `[inline]`

Definition at line 166 of file ArnEvent.hpp.

14.21.3.3 `ArnEvent * ArnEvMonitor::makeHeapClone ()` `[virtual]`

Implements [ArnEvent](#).

Definition at line 292 of file ArnEvent.cpp.

14.21.3.4 `int ArnEvMonitor::monEvType () const` `[inline]`

Definition at line 160 of file ArnEvent.hpp.

14.21.3.5 `void* ArnEvMonitor::sessionHandler () const` `[inline]`

Definition at line 169 of file ArnEvent.hpp.

14.21.3.6 `QEvent::Type ArnEvMonitor::type ()` `[static]`

Definition at line 284 of file ArnEvent.cpp.

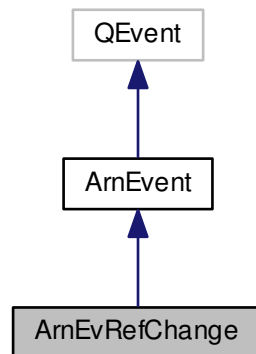
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

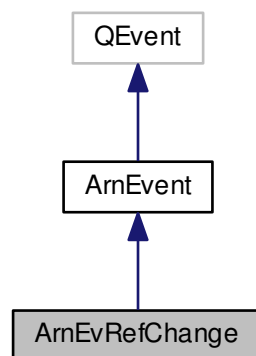
14.22 ArnEvRefChange Class Reference

```
#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

14.22.1 Detailed Description

Definition at line 233 of file ArnEvent.hpp.

14.22.2 Constructor & Destructor Documentation

14.22.2.1 ArnEvRefChange::ArnEvRefChange (int refStep)

Definition at line 345 of file ArnEvent.cpp.

14.22.2.2 ArnEvRefChange::~ArnEvRefChange () [virtual]

Definition at line 352 of file ArnEvent.cpp.

14.22.3 Member Function Documentation

14.22.3.1 ArnEvent * ArnEvRefChange::makeHeapClone () [virtual]

Implements [ArnEvent](#).

Definition at line 365 of file ArnEvent.cpp.

14.22.3.2 int ArnEvRefChange::refStep () const [inline]

Definition at line 243 of file ArnEvent.hpp.

14.22.3.3 QEvent::Type ArnEvRefChange::type () [static]

Definition at line 357 of file ArnEvent.cpp.

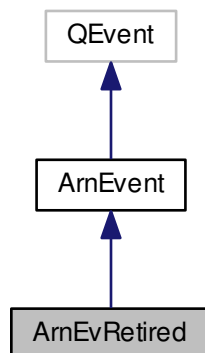
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

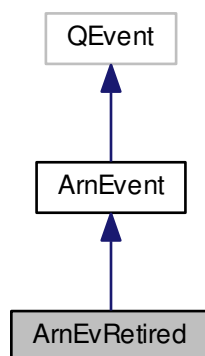
14.23 ArnEvRetired Class Reference

```
#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

14.23.1 Detailed Description

Definition at line 174 of file ArnEvent.hpp.

14.23.2 Constructor & Destructor Documentation

14.23.2.1 ArnEvRetired::ArnEvRetired (ArnLink * *startLink*, bool *isBelow*, bool *isGlobal*)

Definition at line 299 of file ArnEvent.cpp.

14.23.3 Member Function Documentation

14.23.3.1 bool ArnEvRetired::isBelow () const [inline]

Definition at line 188 of file ArnEvent.hpp.

14.23.3.2 bool ArnEvRetired::isGlobal () const [inline]

Definition at line 191 of file ArnEvent.hpp.

14.23.3.3 ArnEvent * ArnEvRetired::makeHeapClone () [virtual]

Implements [ArnEvent](#).

Definition at line 316 of file ArnEvent.cpp.

14.23.3.4 ArnLink* ArnEvRetired::startLink () const [inline]

Definition at line 185 of file ArnEvent.hpp.

14.23.3.5 QEvent::Type ArnEvRetired::type () [static]

Definition at line 308 of file ArnEvent.cpp.

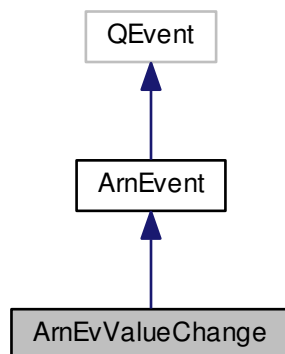
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

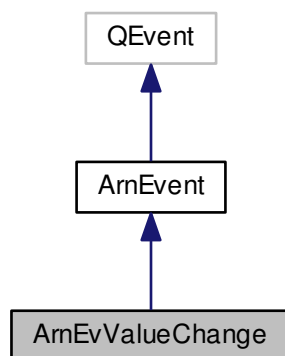
14.24 ArnEvValueChange Class Reference

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

Inheritance diagram for ArnEvValueChanged:



Collaboration diagram for ArnEvValueChanged:



Public Member Functions

- [ArnEvValueChanged](#) (int [sendId](#), const QByteArray *[valueData](#), const ArnLinkHandle &[handleData](#))
- virtual [~ArnEvValueChanged](#) ()
- 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

14.24.1 Detailed Description

Definition at line 210 of file ArnEvent.hpp.

14.24.2 Constructor & Destructor Documentation

14.24.2.1 ArnEvValueChange::ArnEvValueChange (int *sendId*, const QByteArray * *valueData*, const ArnLinkHandle & *handleData*)

Definition at line 187 of file ArnEvent.cpp.

14.24.2.2 ArnEvValueChange::~ArnEvValueChange () [virtual]

Definition at line 202 of file ArnEvent.cpp.

14.24.3 Member Function Documentation

14.24.3.1 const ArnLinkHandle& ArnEvValueChange::handleData () const [inline]

Definition at line 228 of file ArnEvent.hpp.

14.24.3.2 ArnEvent * ArnEvValueChange::makeHeapClone () [virtual]

Implements [ArnEvent](#).

Definition at line 219 of file ArnEvent.cpp.

14.24.3.3 int ArnEvValueChange::sendId () const [inline]

Definition at line 222 of file ArnEvent.hpp.

14.24.3.4 QEvent::Type ArnEvValueChange::type () [static]

Definition at line 211 of file ArnEvent.cpp.

14.24.3.5 const QByteArray* ArnEvValueChange::valueData () const [inline]

Definition at line 225 of file ArnEvent.hpp.

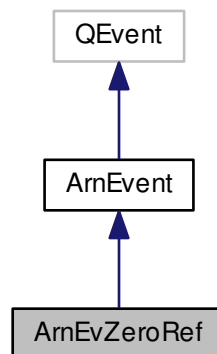
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

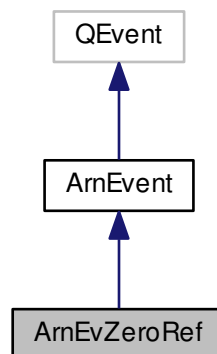
14.25 ArnEvZeroRef Class Reference

```
#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

14.25.1 Detailed Description

Definition at line 196 of file ArnEvent.hpp.

14.25.2 Constructor & Destructor Documentation

14.25.2.1 ArnEvZeroRef::ArnEvZeroRef (ArnLink * *arnLink*)

Definition at line 323 of file ArnEvent.cpp.

14.25.3 Member Function Documentation

14.25.3.1 ArnLink* ArnEvZeroRef::arnLink () const [inline]

Definition at line 205 of file ArnEvent.hpp.

14.25.3.2 ArnEvent* ArnEvZeroRef::makeHeapClone () [virtual]

Implements [ArnEvent](#).

Definition at line 338 of file ArnEvent.cpp.

14.25.3.3 QEvent::Type ArnEvZeroRef::type () [static]

Definition at line 330 of file ArnEvent.cpp.

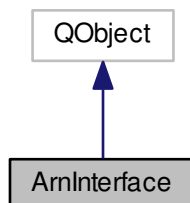
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnEvent.hpp \(3.0.0\)](#)
- [src/ArnEvent.cpp \(3.0.0\)](#)

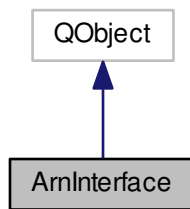
14.26 ArnInterface Class Reference

```
#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\(\)](#)

14.26.1 Detailed Description

Definition at line 39 of file ArnInterface.hpp.

14.26.2 Member Enumeration Documentation

14.26.2.1 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.

14.26.2.2 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.

14.26.2.3 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 *BiDir* 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.

14.26.2.4 enum ArnInterface::SameValue

Action when assigning same value to an [ArnItem](#).

Enumerator

SameValue_Accept Assigning same value generates an update of the [Arn Data Object](#)

SameValue_Ignore Assigning same value is ignored.

SameValue_DefaultAction Assigning same value gives default action set in [ArnM](#) or [ArnItem](#).

Definition at line 46 of file ArnInterface.hpp.

14.26.3 Member Function Documentation

14.26.3.1 QByteArray ArnInterface::bytes (const QString & path) [inline], [slot]

See [ArnM::valueByteArray\(\)](#)

Definition at line 110 of file ArnInterface.hpp.

14.26.3.2 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.

14.26.3.3 `QString ArnInterface::childPath (const QString & parentPath, const QString & posterityPath)` `[inline]`,
`[slot]`

See [Arn::childPath\(\)](#)

Definition at line 181 of file ArnInterface.hpp.

14.26.3.4 `bool ArnInterface::exist (const QString & path)` `[inline]`,`[slot]`

See [ArnM::exist\(\)](#)

Definition at line 126 of file ArnInterface.hpp.

14.26.3.5 `int ArnInterface::intNum (const QString & path)` `[inline]`,`[slot]`

See [ArnM::valueInt\(\)](#)

Definition at line 120 of file ArnInterface.hpp.

14.26.3.6 `bool ArnInterface::isFolder (const QString & path)` `[inline]`,`[slot]`

See [ArnM::isFolder\(\)](#)

Definition at line 129 of file ArnInterface.hpp.

14.26.3.7 `bool ArnInterface::isFolderPath (const QString & path)` `[inline]`,`[slot]`

See [Arn::isFolderPath\(\)](#)

Definition at line 165 of file ArnInterface.hpp.

14.26.3.8 `bool ArnInterface::isLeaf (const QString & path)` `[inline]`,`[slot]`

See [ArnM::isLeaf\(\)](#)

Definition at line 132 of file ArnInterface.hpp.

14.26.3.9 `bool ArnInterface::isProviderPath (const QString & path)` `[inline]`,`[slot]`

See [Arn::isProviderPath\(\)](#)

Definition at line 168 of file ArnInterface.hpp.

14.26.3.10 `QString ArnInterface::itemName (const QString & path)` `[inline]`,`[slot]`

See [Arn::itemName\(\)](#)

Definition at line 171 of file ArnInterface.hpp.

14.26.3.11 `QStringList ArnInterface::items (const QString & path)` `[inline]`,`[slot]`

See [ArnM::items\(\)](#)

Definition at line 123 of file ArnInterface.hpp.

14.26.3.12 `QString ArnInterface::makePath (const QString & parentPath, const QString & itemName)` `[inline]`,
`[slot]`

See [Arn::makePath\(\)](#)

Definition at line 185 of file ArnInterface.hpp.

14.26.3.13 `double ArnInterface::num (const QString & path)` `[inline]`,`[slot]`

See [ArnM::valueDouble\(\)](#)

Definition at line 116 of file ArnInterface.hpp.

14.26.3.14 `QString ArnInterface::providerPath (const QString & path, bool giveProviderPath =true)` `[inline]`,
`[slot]`

See [Arn::providerPath\(\)](#)

Definition at line 189 of file ArnInterface.hpp.

14.26.3.15 `void ArnInterface::setBytes (const QString & path, const QByteArray & value)` `[inline]`,`[slot]`

See [ArnM::setValue\(\)](#)

Definition at line 147 of file ArnInterface.hpp.

14.26.3.16 `void ArnInterface::setIntNum (const QString & path, int value)` `[inline]`,`[slot]`

See [ArnM::setValue\(\)](#)

Definition at line 159 of file ArnInterface.hpp.

14.26.3.17 `void ArnInterface::setNum (const QString & path, double value)` `[inline]`,`[slot]`

See [ArnM::setValue\(\)](#)

Definition at line 154 of file ArnInterface.hpp.

14.26.3.18 `void ArnInterface::setString (const QString & path, const QString & value)` `[inline]`,`[slot]`

See [ArnM::setValue\(\)](#)

Definition at line 143 of file ArnInterface.hpp.

14.26.3.19 `void ArnInterface::setValue (const QString & path, const QVariant & value)` `[inline]`,`[slot]`

See [ArnM::setValue\(\)](#)

Definition at line 135 of file ArnInterface.hpp.

14.26.3.20 `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.

14.26.3.21 `QString ArnInterface::string (const QString & path)` `[inline],[slot]`

See [ArnM::valueString\(\)](#)

Definition at line 107 of file ArnInterface.hpp.

14.26.3.22 `QString ArnInterface::twinPath (const QString & path)` `[inline],[slot]`

See [Arn::twinPath\(\)](#)

Definition at line 174 of file ArnInterface.hpp.

14.26.3.23 `QVariant ArnInterface::value (const QString & path)` `[inline],[slot]`

See [ArnM::valueVariant\(\)](#)

Definition at line 101 of file ArnInterface.hpp.

14.26.3.24 `QVariant ArnInterface::variant (const QString & path)` `[inline],[slot]`

See [ArnM::valueVariant\(\)](#)

Definition at line 104 of file ArnInterface.hpp.

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

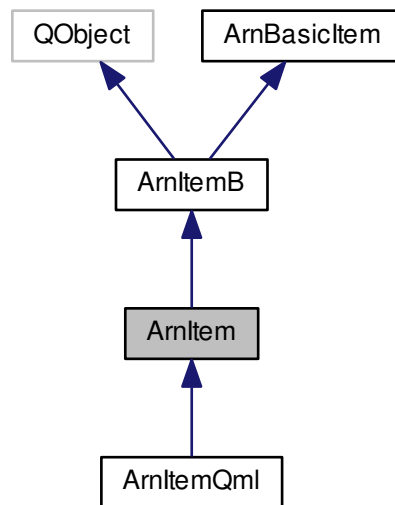
- [src/ArnInc/ArnInterface.hpp \(3.0.0\)](#)

14.27 ArnItem Class Reference

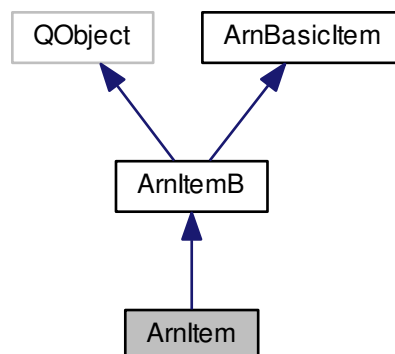
Handle for an [Arn Data Object](#).

```
#include <ArnItem.hpp>
```

Inheritance diagram for ArnItem:



Collaboration diagram for ArnItem:



Public Slots

- void [setValue](#) (int value, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Assign an integer to an [Arn](#) Data Object
- void [setValue](#) (double 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 [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=0)
Standard constructor of a closed handle.
- [ArnItem](#) (const QString &path, QObject *parent=0)
Construction of a handle to a path.
- [ArnItem](#) (const [ArnItem](#) &itemTemplate, const QString &path, QObject *parent=0)
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 `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=0) const
- double `toDouble` (bool *isOk=0) const
- [ARNREAL](#) `toReal` (bool *isOk=0) const
- bool `toBool` (bool *isOk=0) const
- QString `toString` (bool *isOk=0) const
- QByteArray `toByteArray` (bool *isOk=0) const
- QVariant `toVariant` (bool *isOk=0) const
- uint `toUInt` (bool *isOk=0) const
- quint64 `toInt64` (bool *isOk=0) const
- quint64 `toUInt64` (bool *isOk=0) 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=` (quint64 val)
- [ArnItem](#) & `operator=` (quint64 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

14.27.1 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.cpp](#).

Definition at line 72 of file [ArnItem.hpp](#).

14.27.2 Constructor & Destructor Documentation

14.27.2.1 [ArnItem::ArnItem](#) ([QObject](#) * *parent* = 0)

Standard constructor of a closed handle.

Parameters

<i>in</i>	<i>parent</i>	
-----------	---------------	--

Definition at line 109 of file [ArnItem.cpp](#).

14.27.2.2 [ArnItem::ArnItem](#) (const [QString](#) & *path*, [QObject](#) * *parent* = 0)

Construction of a handle to a path.

Parameters

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

See also

[open\(\)](#)

Definition at line 116 of file ArnItem.cpp.

14.27.2.3 ArnItem::ArnItem (const ArnItem & *itemTemplate*, const QString & *path*, QObject * *parent* = 0)

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

Definition at line 124 of file ArnItem.cpp.

14.27.2.4 ArnItem::~ArnItem () [virtual]

Definition at line 531 of file ArnItem.cpp.

14.27.3 Member Function Documentation

14.27.3.1 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.

14.27.3.2 QByteArray ArnItem::arnExport () const [inline]

Returns

A data blob representing the [Arn Data Object](#)

See also

[arnImport\(\)](#)

Definition at line 324 of file ArnItem.hpp.

14.27.3.3 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 318 of file ArnItem.hpp.

14.27.3.4 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.

14.27.3.5 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.

14.27.3.6 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.

14.27.3.7 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\(\)](#)

14.27.3.8 void ArnItem::changed (int value) [signal]

See also

[changed\(\)](#)

14.27.3.9 void ArnItem::changed (double value) [signal]

See also

[changed\(\)](#)

14.27.3.10 void ArnItem::changed (bool value) [signal]

See also

[changed\(\)](#)

14.27.3.11 void ArnItem::changed (const QString & value) [signal]

See also

[changed\(\)](#)

14.27.3.12 void ArnItem::changed (const QByteArray & value) [signal]

See also

[changed\(\)](#)

14.27.3.13 void ArnItem::changed (const QVariant & value) [signal]

See also

[changed\(\)](#)

14.27.3.14 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.

14.27.3.15 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.

14.27.3.16 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.

14.27.3.17 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.

14.27.3.18 bool ArnItem::isDelayPending () const

Delay pending status

Return values

<i>true</i>	if the Arn Data Object is changed, but the changed signal is pending in a delay.
-------------	--

See also

[setDelay\(\)](#)
[bypassDelayPending\(\)](#)

Definition at line 220 of file ArnItem.cpp.

14.27.3.19 `bool ArnItem::isFolder () const [inline]`

Return values

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

Definition at line 122 of file ArnItem.hpp.

14.27.3.20 `bool ArnItem::isIgnoreSameValue () [inline]`

Return values

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

See also

[setIgnoreSameValue\(\)](#)

Definition at line 147 of file ArnItem.hpp.

14.27.3.21 `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.

14.27.3.22 `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.

14.27.3.23 `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.

14.27.3.24 `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.

14.27.3.25 `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.

14.27.3.26 `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](#)

14.27.3.27 `bool ArnItem::openFolder (const QString & path)` `[inline]`

Open a handle to an [Arn](#) folder.

Parameters

<i>in</i>	<i>path</i>	The Arn folder path e.g. <code>"//Measure/Water"</code> (the <code>/</code> is appended)
-----------	-------------	--

Return values

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

Definition at line 117 of file `ArnItem.hpp`.

14.27.3.28 `bool ArnItem::openUuid (const QString & path) [inline]`

Open a handle to an [Arn](#) Object with a unique uuid name.

Parameters

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

Return values

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

Definition at line 103 of file `ArnItem.hpp`.

14.27.3.29 `bool ArnItem::openUuidPipe (const QString & path) [inline]`

Open a handle to an [Arn](#) Pipe Object with a unique uuid name.

Parameters

<i>in</i>	<i>path</i>	The prefix for Arn uuid pipe path e.g. <code>"//Pipes/pipe"</code>
-----------	-------------	--

Return values

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

Definition at line 110 of file `ArnItem.hpp`.

14.27.3.30 `ArnItem & ArnItem::operator= (const ArnItem & other)`

Definition at line 236 of file `ArnItem.cpp`.

14.27.3.31 `ArnItem & ArnItem::operator= (int val)`

Definition at line 243 of file `ArnItem.cpp`.

14.27.3.32 `ArnItem & ArnItem::operator= (ARNREAL other)`

Definition at line 250 of file `ArnItem.cpp`.

14.27.3.33 `ArnItem & ArnItem::operator= (const QString & val)`

Definition at line 257 of file `ArnItem.cpp`.

14.27.3.34 `ArnItem & ArnItem::operator= (const QByteArray & val)`

Definition at line 264 of file `ArnItem.cpp`.

14.27.3.35 ArnItem & ArnItem::operator= (const QVariant & val)

Definition at line 299 of file ArnItem.cpp.

14.27.3.36 ArnItem & ArnItem::operator= (const char * val)

Definition at line 271 of file ArnItem.cpp.

14.27.3.37 ArnItem & ArnItem::operator= (uint val)

Definition at line 278 of file ArnItem.cpp.

14.27.3.38 ArnItem & ArnItem::operator= (qint64 val)

Definition at line 285 of file ArnItem.cpp.

14.27.3.39 ArnItem & ArnItem::operator= (quint64 val)

Definition at line 292 of file ArnItem.cpp.

14.27.3.40 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.

14.27.3.41 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.

14.27.3.42 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 272 of file ArnItem.hpp.

14.27.3.43 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.

14.27.3.44 void ArnItem::setIgnoreSameValue (bool *isIgnore* = true) [inline]

Set skipping assignment of equal value.

Parameters

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

Definition at line 141 of file ArnItem.hpp.

14.27.3.45 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.

14.27.3.46 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.

14.27.3.47 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.

14.27.3.48 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

<i>in</i>	<i>isTemplate</i>	True for template mode.
-----------	-------------------	-------------------------

See also

[open\(\)](#)
[Modes](#)

Definition at line 182 of file ArnItem.cpp.

14.27.3.49 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

<i>in</i>	<i>other</i>	is the ArnItem containing the value to assign
<i>in</i>	<i>ignoreSame</i>	can override default ignoreSameValue setting.

See also

[setIgnoreSameValue\(\)](#)

Definition at line 416 of file ArnItem.hpp.

14.27.3.50 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 425 of file ArnItem.hpp.

14.27.3.51 void ArnItem::setValue (quint64 *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 434 of file ArnItem.hpp.

14.27.3.52 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 443 of file ArnItem.hpp.

14.27.3.53 void ArnItem::setValue (int *value*, int *ignoreSame* = Arn::SameValue::DefaultAction) [inline],
[slot]

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 452 of file ArnItem.hpp.

```
14.27.3.54 void ArnItem::setValue ( double value, int ignoreSame = Arn::SameValue::DefaultAction ) [inline],
        [slot]
```

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 463 of file ArnItem.hpp.

```
14.27.3.55 void ArnItem::setValue ( bool value, int ignoreSame = Arn::SameValue::DefaultAction ) [inline],
        [slot]
```

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 472 of file ArnItem.hpp.

```
14.27.3.56 void ArnItem::setValue ( const QString & value, int ignoreSame = Arn::SameValue::DefaultAction )
        [inline], [slot]
```

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 480 of file ArnItem.hpp.

14.27.3.57 `void ArnItem::setValue (const QByteArray & value, int ignoreSame = Arn::SameValue::DefaultAction)`
[inline],[slot]

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 488 of file ArnItem.hpp.

14.27.3.58 `void ArnItem::setValue (const QVariant & value, int ignoreSame = Arn::SameValue::DefaultAction)`
`[inline], [slot]`

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 496 of file ArnItem.hpp.

14.27.3.59 `void ArnItem::setValue (const char * value, int ignoreSame = Arn::SameValue::DefaultAction)`
`[inline], [slot]`

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 504 of file ArnItem.hpp.

14.27.3.60 `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.

14.27.3.61 `bool ArnItem::toBool (bool * isOk = 0) 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 352 of file ArnItem.hpp.

14.27.3.62 `QByteArray ArnItem::toByteArray (bool * isOk = 0) 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 366 of file ArnItem.hpp.

14.27.3.63 `double ArnItem::toDouble (bool * isOk = 0) 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 338 of file ArnItem.hpp.

14.27.3.64 `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 306 of file ArnItem.cpp.

14.27.3.65 `int ArnItem::toInt (bool * isOk = 0) 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 331 of file ArnItem.hpp.

14.27.3.66 `qint64 ArnItem::toInt64 (bool * isOk = 0) 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 389 of file ArnItem.hpp.

14.27.3.67 `ARNREAL ArnItem::toReal (bool * isOk = 0) 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 345 of file ArnItem.hpp.

14.27.3.68 `QString ArnItem::toString (bool * isOk = 0) const` `[inline]`

Returns

Convert [Arn Data Object](#) to a *QString*

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.

14.27.3.69 `uint ArnItem::toUInt (bool * isOk = 0) const` `[inline]`

Returns

Convert [Arn Data Object](#) to an *unsigned int*

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 381 of file ArnItem.hpp.

14.27.3.70 `quint64 ArnItem::toUInt64 (bool * isOk = 0) 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, *isOk is set to false, otherwise *isOk is set to true.
-----	-------------	---

Note

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

Definition at line 397 of file ArnItem.hpp.

14.27.3.71 `QVariant ArnItem::toVariant (bool * isOk = 0) const` `[inline]`

Returns

Convert [Arn Data Object](#) to a *QVariant*

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.

14.27.3.72 `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:

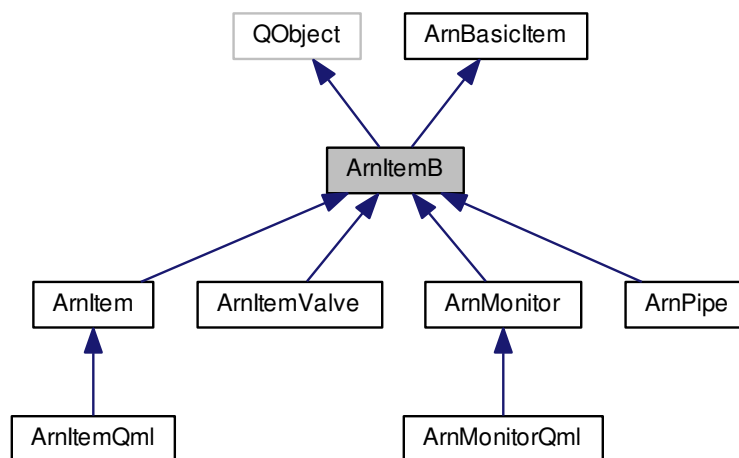
- [src/ArnInc/ArnItem.hpp \(3.0.0\)](#)
- [src/ArnItem.cpp \(3.0.0\)](#)

14.28 ArnItemB Class Reference

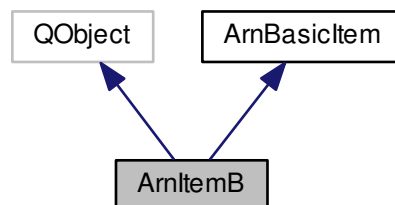
Base class handle for an [Arn Data Object](#).

```
#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=0)
Standard constructor of a closed handle.
- virtual [~ArnItemB](#) ()
- bool [open](#) (const QString &path)
Open a handle to an [Arn Data Object](#)

14.28.1 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.

14.28.2 Constructor & Destructor Documentation

14.28.2.1 ArnItemB::ArnItemB (QObject * parent = 0)

Standard constructor of a closed handle.

Parameters

in	parent	
----	--------	--

Definition at line 59 of file ArnItemB.cpp.

14.28.2.2 ArnItemB::~ArnItemB () [virtual]

Definition at line 75 of file ArnItemB.cpp.

14.28.3 Member Function Documentation

14.28.3.1 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\(\)](#)

14.28.3.2 bool ArnItemB::open (const QString & path)

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

Parameters

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

Return values

false	if error
-------	----------

Definition at line 89 of file ArnItemB.cpp.

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

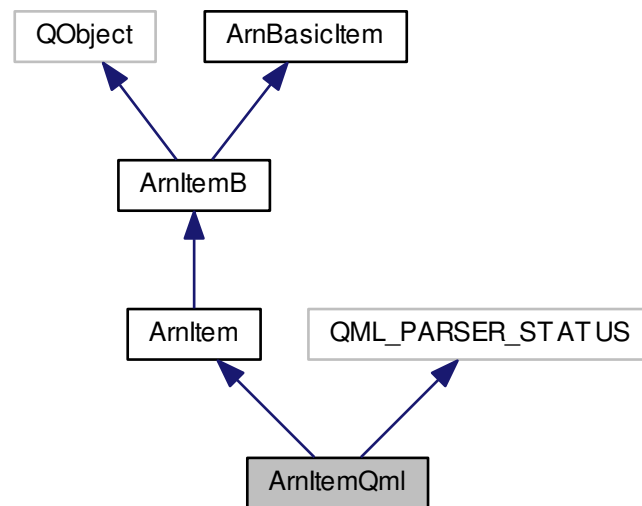
- [src/ArnInc/ArnItemB.hpp \(3.0.0\)](#)
- [src/ArnItemB.cpp \(3.0.0\)](#)

14.29 ArnItemQml Class Reference

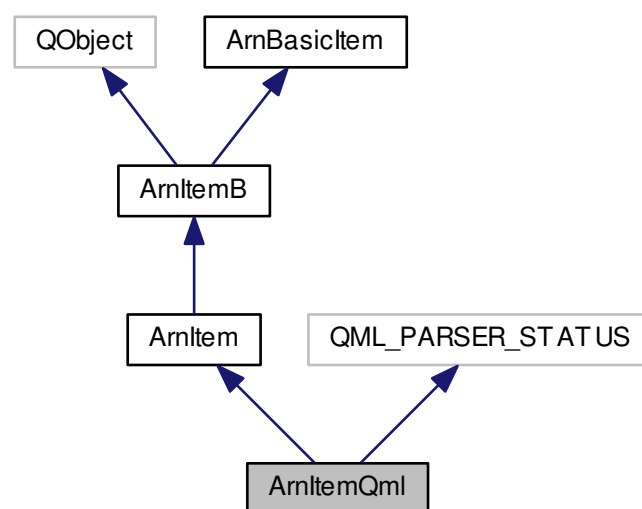
ARN Item QML.

```
#include <ArnQml.hpp>
```

Inheritance diagram for ArnItemQml:



Collaboration diagram for ArnItemQml:



Public Slots

- void [addMode](#) ([ArnInterface::ObjectMode](#) mode)
Add general mode settings for this [Arn](#) Data Object
- [ArnInterface::ObjectMode](#) [getMode](#) () const

Additional Inherited Members

14.29.1 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 arnTl: ArnItem {path: "//El/UpdClock/value"}

    ArnItem {id: arnElUpdClock; path: "//El/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 283 of file [ArnQml.hpp](#).

14.29.2 Member Function Documentation

14.29.2.1 void [ArnItemQml::addMode](#) ([ArnInterface::ObjectMode](#) mode) [inline], [slot]

Add general mode settings for this [Arn](#) Data Object

See also

[ArnItem::addMode\(\)](#)

Definition at line 336 of file [ArnQml.hpp](#).

14.29.2.2 `ArnInterface::ObjectMode ArnItemQml::getMode () const` `[inline],[slot]`

Returns

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

See also

[ArnItem::getMode\(\)](#)

Definition at line 342 of file ArnQml.hpp.

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

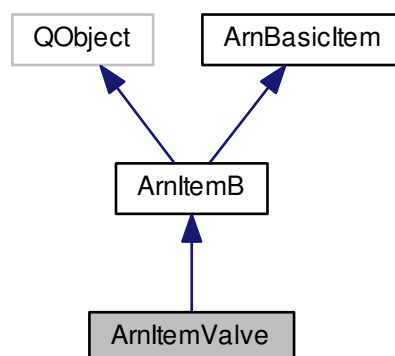
- [src/ArnInc/ArnQml.hpp \(3.0.0\)](#)
- [src/ArnQml.cpp \(3.0.0\)](#)

14.30 ArnItemValve Class Reference

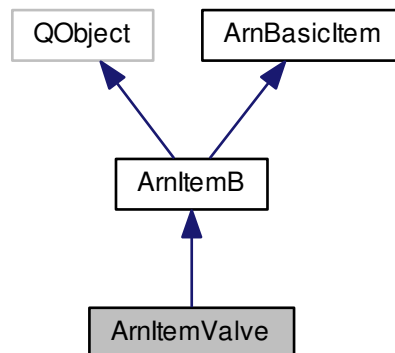
Valve for controlling stream to/from an [ArnItemB](#).

```
#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=0)
- 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)

14.30.1 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 beeing 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`.

14.30.2 Constructor & Destructor Documentation

14.30.2.1 ArnItemValve::ArnItemValve (QObject *parent = 0) [explicit]

Definition at line 48 of file `ArnItemValve.cpp`.

14.30.3 Member Function Documentation

14.30.3.1 void ArnItemValve::changed (int value) [signal]

Signals emitted when data in *Arn Data Object* is changed.

14.30.3.2 bool ArnItemValve::isAutoDestroy () const [inline]

Return values

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

See also

[setAutoDestroy\(\)](#)

Definition at line 143 of file `ArnItemValve.hpp`.

14.30.3.3 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`.

14.30.3.4 `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`.

14.30.3.5 `ArnItemValve & ArnItemValve::operator= (bool value)`

Definition at line 91 of file `ArnItemValve.cpp`.

14.30.3.6 `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`.

14.30.3.7 `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`.

14.30.3.8 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.

14.30.3.9 bool ArnItemValve::setTarget (ArnItemB * targetItem, ArnItemValve::SwitchMode mode = SwitchMode::InOutputStream)

Definition at line 60 of file ArnItemValve.cpp.

14.30.3.10 void ArnItemValve::setValue (bool value) [slot]

Assign a *bool* to an [Arn Data Object](#)

Parameters

in	value	to be assigned
----	-------	----------------

Definition at line 98 of file ArnItemValve.cpp.

14.30.3.11 ArnItemValve::SwitchMode ArnItemValve::switchMode () const

Definition at line 72 of file ArnItemValve.cpp.

14.30.3.12 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 \(3.0.0\)](#)
- [src/ArnItemValve.cpp \(3.0.0\)](#)

14.31 ArnLinkValue Struct Reference**Public Member Functions**

- [ArnLinkValue \(\)](#)

Public Attributes

- QString [valueString](#)
- QByteArray [valueByteArray](#)
- QVariant [valueVariant](#)
- volatile [ARNREAL](#) [valueReal](#)
- volatile int [valueInt](#)

14.31.1 Detailed Description

Definition at line 43 of file ArnLink.cpp.

14.31.2 Constructor & Destructor Documentation

14.31.2.1 ArnLinkValue::ArnLinkValue () `[inline]`

Definition at line 50 of file ArnLink.cpp.

14.31.3 Member Data Documentation

14.31.3.1 QByteArray ArnLinkValue::valueByteArray

Definition at line 45 of file ArnLink.cpp.

14.31.3.2 volatile int ArnLinkValue::valueInt

Definition at line 48 of file ArnLink.cpp.

14.31.3.3 volatile [ARNREAL](#) ArnLinkValue::valueReal

Definition at line 47 of file ArnLink.cpp.

14.31.3.4 QString ArnLinkValue::valueString

Definition at line 44 of file ArnLink.cpp.

14.31.3.5 QVariant ArnLinkValue::valueVariant

Definition at line 46 of file ArnLink.cpp.

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

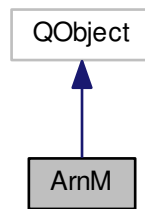
- [src/ArnLink.cpp \(3.0.0\)](#)

14.32 ArnM Class Reference

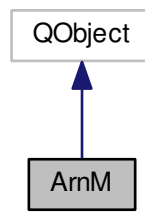
[Arn](#) main class.

```
#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 [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=0)
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=0)
- 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](#)

14.32.1 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.

14.32.2 Member Function Documentation

14.32.2.1 `bool ArnM::defaultIgnoreSameValue () [static]`

Return values

<i>true</i>	if default skipping equal assignment value
-------------	--

See also

[setDefaultIgnoreSameValue\(\)](#)

Definition at line 1036 of file ArnM.cpp.

14.32.2.2 `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

<i>in</i>	<i>path</i>	
<i>in</i>	<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 822 of file ArnM.cpp.

14.32.2.3 `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

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

See also

[destroyLink\(\)](#)

Definition at line 284 of file ArnM.hpp.

14.32.2.4 void ArnM::errorLog (QString *errText*, ArnError *err* = ArnError::Undef, void * *reference* = 0) [static]

Definition at line 993 of file ArnM.cpp.

14.32.2.5 void ArnM::errorLogSig (const QString & *errText*, uint *errCode*, void * *reference*) [signal]

14.32.2.6 QString ArnM::errorSysName () [static]

Definition at line 929 of file ArnM.cpp.

14.32.2.7 bool ArnM::exist (const QString & *path*) [static]

Parameters

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

Return values

<i>true</i>	if Arn Data Object exist at <i>path</i>
-------------	---

Definition at line 397 of file ArnM.cpp.

14.32.2.8 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 935 of file ArnM.cpp.

14.32.2.9 ArnM & ArnM::instance () [static]

Definition at line 1016 of file ArnM.cpp.

14.32.2.10 bool ArnM::isFolder (const QString & *path*) [static]

Parameters

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

Return values

<i>true</i>	if Arn Data Object at <i>path</i> is a folder
-------------	---

Definition at line 408 of file ArnM.cpp.

14.32.2.11 bool ArnM::isLeaf (const QString & *path*) [static]

Parameters

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

Return values

<i>true</i>	if Arn Data Object at <i>path</i> is a leaf (non folder)
-------------	--

Definition at line 419 of file ArnM.cpp.

14.32.2.12 bool ArnM::isMainThread () [static]

Return values

<i>true</i>	if this is the main thread in the application
-------------	---

Definition at line 374 of file ArnM.cpp.

14.32.2.13 bool ArnM::isThreadedApp () [static]

Return values

<i>true</i>	if this is a threaded application
-------------	-----------------------------------

Definition at line 391 of file ArnM.cpp.

14.32.2.14 QStringList ArnM::items (const QString & path) [static]

Get the childrens of the folder at *path*

Example: return list = {"test", "folder/", "@/", "value"}

Parameters

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

Returns

The items (children)

Definition at line 312 of file ArnM.cpp.

14.32.2.15 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

<i>in</i>	<i>path</i>	is the path of the Arn Data Object and also path relative to <i>dirRoot</i>
<i>in</i>	<i>dirRoot</i>	is the file directory to be used as root for the <i>path</i>
<i>in</i>	<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 525 of file ArnM.cpp.

14.32.2.16 `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 Arn Data Object
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 507 of file ArnM.cpp.

14.32.2.17 `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 Arn Data Object
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 534 of file ArnM.cpp.

14.32.2.18 `void ArnM::setConsoleError (bool isConsoleError) [static]`

Definition at line 1024 of file ArnM.cpp.

14.32.2.19 `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 1030 of file ArnM.cpp.

14.32.2.20 `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 1048 of file ArnM.cpp.

14.32.2.21 void ArnM::setErrorlog (QObject * *errLog*) [static],[slot]

Definition at line 941 of file ArnM.cpp.

14.32.2.22 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 442 of file ArnM.cpp.

14.32.2.23 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 453 of file ArnM.cpp.

14.32.2.24 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 431 of file ArnM.cpp.

14.32.2.25 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 464 of file ArnM.cpp.

14.32.2.26 void ArnM::setValue (const QString & *path*, const QVariant & *value*, const char * *typeName* = 0) [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 475 of file ArnM.cpp.

14.32.2.27 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 501 of file ArnM.cpp.

14.32.2.28 bool ArnM::skipLocalSysLoading () const

Return mode skip "/Local/Sys/" loading.

Returns

mode skipLocalSysLoading

See also

[setSkipLocalSysLoading\(\)](#)

Definition at line 1042 of file ArnM.cpp.

14.32.2.29 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 280 of file ArnM.cpp.

14.32.2.30 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 252 of file ArnM.cpp.

14.32.2.31 `int ArnM::valueInt (const QString & path) [static]`

Get the value of *Arn Data Object* at *path*

Parameters

in	path	
----	------	--

Returns

The [Arn Data Object](#) as an *integer*

Definition at line 241 of file ArnM.cpp.

14.32.2.32 ARNREAL ArnM::valueReal (const QString & path) [static]

Get the value of [Arn Data Object](#) at *path*

Parameters

in	path	
----	------	--

Returns

The [Arn Data Object](#) as an *ARNREAL*

Definition at line 258 of file ArnM.cpp.

14.32.2.33 QString ArnM::valueString (const QString & path) [static]

Get the value of [Arn Data Object](#) at *path*

Parameters

in	path	
----	------	--

Returns

The [Arn Data Object](#) as a *QString*

Definition at line 269 of file ArnM.cpp.

14.32.2.34 QVariant ArnM::valueVariant (const QString & path) [static]

Get the value of [Arn Data Object](#) at *path*

Parameters

in	path	
----	------	--

Returns

The [Arn Data Object](#) as a *QVariant*

Definition at line 291 of file ArnM.cpp.

14.32.3 Friends And Related Function Documentation**14.32.3.1 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 \(3.0.0\)](#)
- [src/ArnM.cpp \(3.0.0\)](#)

14.33 ArnMonEventType Class Reference

```
#include <ArnMonEvent.hpp>
```

Public Types

- enum [E](#) {
 [None](#) = 0, [ItemCreated](#), [ItemFound](#), [ItemDeleted](#),
 [ItemModeChg](#), [MonitorStart](#), [MonitorReStart](#) }
- enum [NS](#) { [NsEnum](#), [NsCom](#) }

14.33.1 Detailed Description

Definition at line 39 of file ArnMonEvent.hpp.

14.33.2 Member Enumeration Documentation

14.33.2.1 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.

14.33.2.2 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:

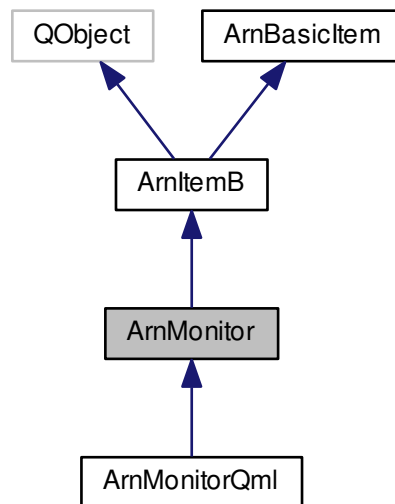
- [src/ArnInc/ArnMonEvent.hpp \(3.0.0\)](#)

14.34 ArnMonitor Class Reference

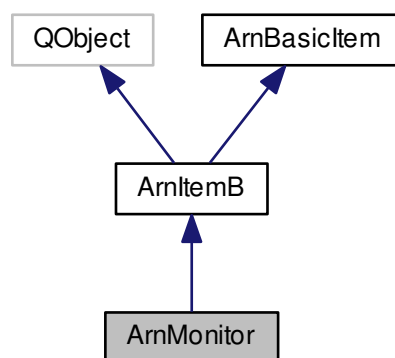
A client remote monitor to detect changes at server.

```
#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=0)
- [ArnMonitor](#) (const QString &path, QObject *parent=0)
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=0)
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.

14.34.1 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.

14.34.2 Constructor & Destructor Documentation

14.34.2.1 ArnMonitor::ArnMonitor (QObject * *parent* = 0) [explicit]

Definition at line 64 of file ArnMonitor.cpp.

14.34.2.2 ArnMonitor::ArnMonitor (const QString & *path*, QObject * *parent* = 0)

Starts local monitoring.

Parameters

in	<i>path</i>	
in	<i>parent</i>	

See also

[start\(\)](#)

Definition at line 72 of file ArnMonitor.cpp.

14.34.2.3 ArnMonitor::~ArnMonitor ()

Definition at line 89 of file ArnMonitor.cpp.

14.34.3 Member Function Documentation

14.34.3.1 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

<i>in</i>	<i>path</i>	to the child
-----------	-------------	--------------

See also

[arnItemDeleted\(\)](#)

14.34.3.2 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

<i>in</i>	<i>path</i>	to the child
-----------	-------------	--------------

See also

[arnItemCreated\(\)](#)

14.34.3.3 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

<i>in</i>	<i>path</i>	to the child
-----------	-------------	--------------

See also

[arnItemCreated\(\)](#)

[arnChildFound\(\)](#)

14.34.3.4 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\(\)](#)

14.34.3.5 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\(\)](#)

14.34.3.6 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 Arn Data Object
----	-------------	--

14.34.3.7 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 Arn Data Object
----	-------------	--

14.34.3.8 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 chnging mode in this folder or its children below will give this signal.

Parameters

in	<i>path</i>	to the mode changing Arn Data Object
in	<i>mode</i>	is the new Arn::ObjectMode

14.34.3.9 **ArnClient * ArnMonitor::client () const**

Get the used *client*

Returns

The *client*

See also

[setClient\(\)](#)

Definition at line 126 of file ArnMonitor.cpp.

14.34.3.10 **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.

14.34.3.11 **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.

14.34.3.12 **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

14.34.3.13 QString ArnMonitor::monitorPath () const

Get the monitored *path*

Returns

The *path*

See also

[start\(\)](#)

Definition at line 214 of file ArnMonitor.cpp.

14.34.3.14 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.

14.34.3.15 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.

14.34.3.16 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.

14.34.3.17 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.

14.34.3.18 void ArnMonitor::setMonitorPath (const QString & *path*, ArnClient * *client* = 0)

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.

14.34.3.19 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.

14.34.3.20 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.

14.34.3.21 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:

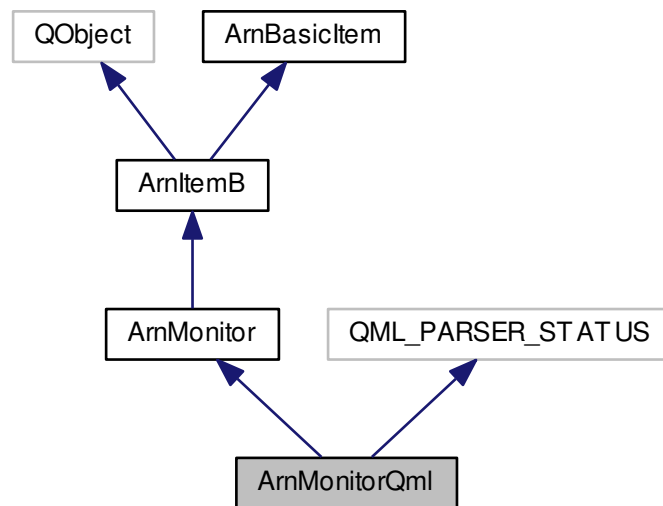
- [src/ArnInc/ArnMonitor.hpp \(3.0.0\)](#)
- [src/ArnMonitor.cpp \(3.0.0\)](#)

14.35 ArnMonitorQml Class Reference

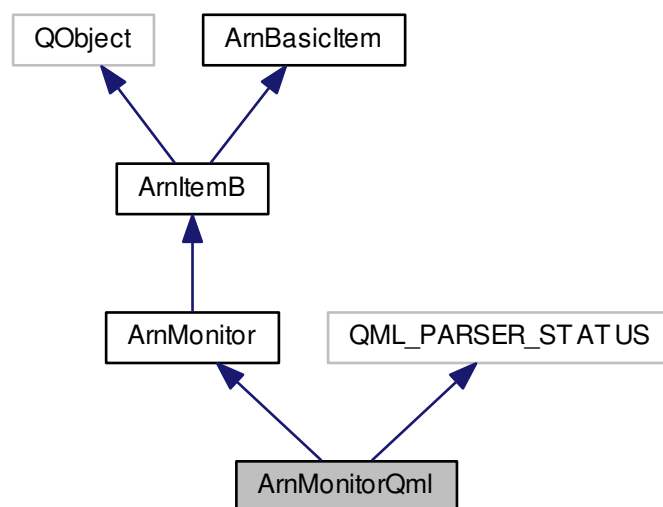
ARN Monitor QML.

```
#include <ArnQml.hpp>
```

Inheritance diagram for ArnMonitorQml:



Collaboration diagram for ArnMonitorQml:



Public Slots

- void [reStart](#) ()

Restart the monitor.

Additional Inherited Members

14.35.1 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 419 of file ArnQml.hpp.

14.35.2 Member Function Documentation

14.35.2.1 void ArnMonitorQml::reStart () [slot]

Restart the monitor.

All signals for found childs will be emitted again.

Definition at line 310 of file ArnQml.cpp.

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

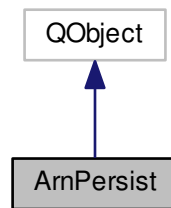
- src/ArnInc/[ArnQml.hpp](#) (3.0.0)
- src/[ArnQml.cpp](#) (3.0.0)

14.36 ArnPersist Class Reference

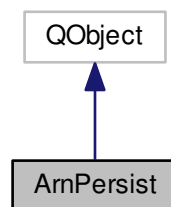
Class for handling persistent [Arn Data object](#).

```
#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=0)
- [~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.

14.36.1 Detailed Description

Class for handling persistent *Arn Data object*.

About Persistent Arn Data Object

This class is used at an *ArnServer* to implemennt 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 149 of file ArnPersist.hpp.

14.36.2 Constructor & Destructor Documentation

14.36.2.1 ArnPersist::ArnPersist (QObject * *parent* = 0) [explicit]

Definition at line 191 of file ArnPersist.cpp.

14.36.2.2 ArnPersist::~~ArnPersist ()

Definition at line 207 of file ArnPersist.cpp.

14.36.3 Member Function Documentation

14.36.3.1 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

<i>in</i>	<i>name</i>	is the file name of the backup. QString() gives default name.
-----------	-------------	---

See also

[setArchiveDir\(\)](#)

Definition at line 824 of file ArnPersist.cpp.

14.36.3.2 bool ArnPersist::flush (const QString & *path* = QString())

Save any pending values now.

Persistent values are normally delayed before saving.

Parameters

<i>in</i>	<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 508 of file ArnPersist.cpp.

14.36.3.3 void ArnPersist::setArchiveDir (const QString & *path*)

Set the persistent database backup directory.

In this directory, all backup files are stored.

Parameters

<i>in</i>	<i>path</i>	is the persistent file directory <i>root</i> .
-----------	-------------	--

See also

[doArchive\(\)](#)

[Persistent Arn Data Objects](#)

Definition at line 221 of file ArnPersist.cpp.

14.36.3.4 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

<i>in</i>	<i>path</i>	is the persistent enabled tree.
-----------	-------------	---------------------------------

Return values

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

See also

[Persistent Arn Data Objects](#)

Definition at line 415 of file ArnPersist.cpp.

14.36.3.5 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

<code>in</code>	<code>path</code>	is the persistent file directory <i>root</i> .
-----------------	-------------------	--

See also

[setVcs\(\)](#)
[Persistent Arn Data Objects](#)

Definition at line 213 of file ArnPersist.cpp.

14.36.3.6 `bool ArnPersist::setUpDataBase (const QString & dbName = "persist.db")`

Setup the persistent database.

Starting a SQLite database to store persistent [Arn Data Object](#) in.

Parameters

<code>in</code>	<code>dbName</code>	is the name (and path) of the SQLite database file.
-----------------	---------------------	---

Return values

<code>false</code>	if error.
--------------------	-----------

See also

[Persistent Arn Data Objects](#)

Definition at line 447 of file ArnPersist.cpp.

14.36.3.7 `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

<code>in</code>	<code>vcs</code>	is the class implementing the VCS. Use 0 (null) to set none.
-----------------	------------------	--

See also

[setPersistDir\(\)](#)
[Persistent Arn Data Objects](#)

Definition at line 229 of file ArnPersist.cpp.

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

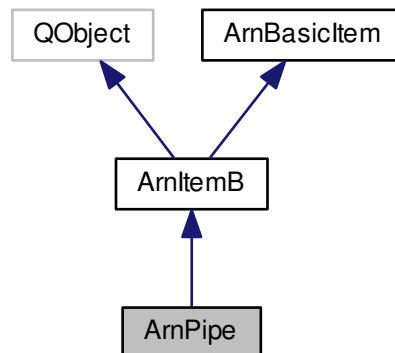
- [src/ArnInc/ArnPersist.hpp \(3.0.0\)](#)
- [src/ArnPersist.cpp \(3.0.0\)](#)

14.37 ArnPipe Class Reference

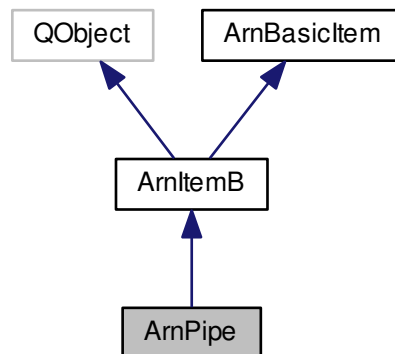
[ArnItem](#) specialized as a pipe.

```
#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=0)
Standard constructor of a closed handle.
- [ArnPipe](#) (const QString &path, QObject *parent=0)
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 QRegExp &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.

14.37.1 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)));

QRegExp rx("^ping\\b");
_arnPipe.setValueOverwrite( "ping new", rx);
```

Definition at line 63 of file ArnPipe.hpp.

14.37.2 Constructor & Destructor Documentation

14.37.2.1 ArnPipe::ArnPipe (QObject * parent = 0)

Standard constructor of a closed handle.

Parameters

<i>in</i>	<i>parent</i>	
-----------	---------------	--

Definition at line 58 of file ArnPipe.cpp.

14.37.2.2 ArnPipe::ArnPipe (const QString & *path*, QObject * *parent* = 0)

Construction of a pipe handle to a *path*

The mode for this handle is set to [Arn::ObjectMode::Pipe](#).

Parameters

<i>in</i>	<i>path</i>	The Arn Data Object path e.g. "//Pipes/myPipe/value"
<i>in</i>	<i>parent</i>	

See also

[open\(\)](#)

Definition at line 65 of file ArnPipe.cpp.

14.37.2.3 ArnPipe::~ArnPipe () [virtual]

Definition at line 79 of file ArnPipe.cpp.

14.37.3 Member Function Documentation

14.37.3.1 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

<i>in</i>	<i>value</i>	is the received bytes
-----------	--------------	-----------------------

14.37.3.2 bool ArnPipe::isAutoDestroy () const [inline]

Return values

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

See also

[setAutoDestroy\(\)](#)

Definition at line 117 of file ArnPipe.hpp.

14.37.3.3 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.

14.37.3.4 bool ArnPipe::isMaster () const [inline]

Return values

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

See also

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

Definition at line 104 of file ArnPipe.hpp.

14.37.3.5 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.

14.37.3.6 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

<i>in</i>	<i>path</i>	The prefix for Arn uuid pipe path e.g. <code>"/Pipes/pipe"</code>
-----------	-------------	---

Return values

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

Definition at line 89 of file ArnPipe.hpp.

14.37.3.7 ArnPipe & ArnPipe::operator= (const QByteArray & value)

Definition at line 98 of file ArnPipe.cpp.

14.37.3.8 void ArnPipe::outOfSequence () [signal]

Signal emitted when the received sequence numbers are "out of sequence".

See also

[setCheckSeq\(\)](#)
[setSendSeq\(\)](#)
[Pipe sequence check](#)

14.37.3.9 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 111 of file ArnPipe.hpp.

14.37.3.10 void ArnPipe::setCheckSeq (bool useCheckSeq)

Change usage of checking received sequence numbers.

Parameters

in	useCheckSeq	is true for activation
--------------------	-----------------------------	------------------------

See also

[isCheckSeq\(\)](#)
[setSendSeq\(\)](#)
[outOfSequence\(\)](#)
[Pipe sequence check](#)

Definition at line 154 of file ArnPipe.cpp.

14.37.3.11 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 97 of file ArnPipe.hpp.

14.37.3.12 void ArnPipe::setSendSeq (bool useSendSeq)

Change usage of sending sequence numbers.

Parameters

<code>in</code>	<code>useSendSeq</code>	is true for activation
-----------------	-------------------------	------------------------

See also

[isSendSeq\(\)](#)
[setCheckSeq\(\)](#)
[outOfSequence\(\)](#)
[Pipe sequence check](#)

Definition at line 138 of file ArnPipe.cpp.

14.37.3.13 `void ArnPipe::setValue (const QByteArray & value) [slot]`

Assign a *QByteArray* to a *Pipe*

Parameters

<code>in</code>	<code>value</code>	to be assigned
-----------------	--------------------	----------------

Definition at line 84 of file ArnPipe.cpp.

14.37.3.14 `void ArnPipe::setValueOverwrite (const QByteArray & value, const QRegExp & 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
QRegExp rx("^" + funcName + "\\b");
_pipe->setValueOverwrite( message, rx);
```

Parameters

<code>in</code>	<code>value</code>	to be assigned
<code>in</code>	<code>rx</code>	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:

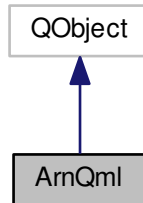
- [src/ArnInc/ArnPipe.hpp \(3.0.0\)](#)
- [src/ArnPipe.cpp \(3.0.0\)](#)

14.38 ArnQml Class Reference

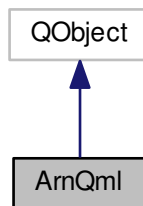
ARN QML.

```
#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.

14.38.1 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

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 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: "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 180 of file ArnQml.hpp.

14.38.2 Member Function Documentation

14.38.2.1 QString ArnQml::arnRootPath () [static]

Gives current ARN root path for all qml instances.

Returns

the root path

See also

[setArnRootPath](#)

Definition at line 58 of file ArnQml.cpp.

14.38.2.2 ArnQml & ArnQml::instance () [static]

Definition at line 116 of file ArnQml.cpp.

14.38.2.3 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 64 of file ArnQml.cpp.

14.38.2.4 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 80 of file ArnQml.cpp.

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

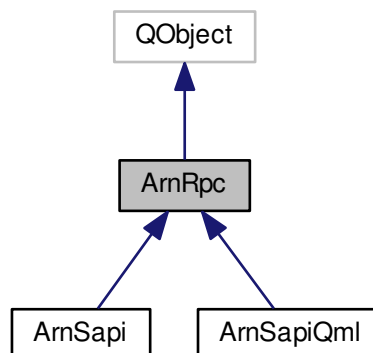
- [src/ArnInc/ArnQml.hpp \(3.0.0\)](#)
- [src/ArnQml.cpp \(3.0.0\)](#)

14.39 ArnRpc Class Reference

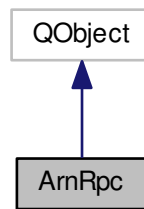
Remote Procedure Call.

```
#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=0)
- [~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 QRegExp &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 QRegExp &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 QRegExp &rgx, const QObject *`receiver`, const QString &replace, `Mode mode`=`Mode`())
Make batch connection from one senders signals to another receivers slots/signals.

14.39.1 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 155 of file ArnRpc.hpp.

14.39.2 Member Typedef Documentation

14.39.2.1 typedef ArnRpcMode ArnRpc::Mode

Definition at line 161 of file ArnRpc.hpp.

14.39.3 Constructor & Destructor Documentation

14.39.3.1 ArnRpc::ArnRpc(QObject * *parent* = 0) [explicit]

Definition at line 204 of file ArnRpc.cpp.

14.39.3.2 ArnRpc::~ArnRpc()

Definition at line 220 of file ArnRpc.cpp.

14.39.4 Member Function Documentation

14.39.4.1 void ArnRpc::addSenderSignals(QObject * *sender*, const QString & *prefix*)

Definition at line 440 of file ArnRpc.cpp.

14.39.4.2 `void ArnRpc::batchConnect (const QObject * sender, const QRegExp & 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 testT() are not allowed.

Example: `batchConnect (_commonSapi, QRegExp("^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 1476 of file ArnRpc.cpp.

14.39.4.3 `void ArnRpc::batchConnect (const QRegExp & 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 testT() are not allowed.

Example: `_commonSapi.batchConnect (QRegExp("^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 QRegExp&, const QObject*, const QString&, [Mode](#))

Definition at line 347 of file ArnRpc.hpp.

14.39.4.4 `void ArnRpc::batchConnect (const QObject * sender, const QRegExp & 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, QRegExp("^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 QRegExp&, const QObject*, const QString&, [Mode](#))

Definition at line 368 of file ArnRpc.hpp.

14.39.4.5 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

in	<i>data</i>	is the received call message in XString format.
----	-------------	---

14.39.4.6 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.

14.39.4.7 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.

14.39.4.8 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.
----	-------------	--

14.39.4.9 void ArnRpc::heartBeatReceived () [signal]

Signal emitted when Heart beat message is received.

14.39.4.10 bool ArnRpc::invoke (const QString & *funcName*, MQGenericArgument *val0* = MQGenericArgument (), 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.

14.39.4.11 bool ArnRpc::invoke (const QString & *funcName*, Invoke *invokeFlags*, MQGenericArgument *val0* = MQGenericArgument (), 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.

14.39.4.12 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.

14.39.4.13 QString ArnRpc::methodPrefix () const

Definition at line 346 of file ArnRpc.cpp.

14.39.4.14 ArnRpc::Mode ArnRpc::mode () const

Get the mode.

Returns

current *mode*

Definition at line 386 of file ArnRpc.cpp.

14.39.4.15 bool ArnRpc::open (const QString & pipePath)

Definition at line 236 of file ArnRpc.cpp.

14.39.4.16 void ArnRpc::outOfSequence () [signal]

Signal emitted when checked sequence order is wrong.

14.39.4.17 ArnPipe * ArnRpc::pipe () const

Get the used *pipe*

Returns

pipe

See also[setPipe\(\)](#)

Definition at line 292 of file ArnRpc.cpp.

14.39.4.18 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.**14.39.4.19** QString ArnRpc::pipePath () constGet the path for the used *pipe***Returns**

path

Definition at line 226 of file ArnRpc.cpp.

14.39.4.20 QObject * ArnRpc::receiver () const

Definition at line 329 of file ArnRpc.cpp.

14.39.4.21 ArnRpc * ArnRpc::rpcSender ()

Definition at line 473 of file ArnRpc.cpp.

14.39.4.22 ArnRpc * ArnRpc::rpcSender (QObject * *receiver*) [static]

Definition at line 483 of file ArnRpc.cpp.

14.39.4.23 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**

<i>in</i>	<i>txt</i>	is the text to be sent
-----------	------------	------------------------

See also[textReceived\(\);](#)

Definition at line 1455 of file ArnRpc.cpp.

14.39.4.24 void ArnRpc::setHeartBeatCheck (int *time*)

Set max time period for receiving heart beat message.

Setting time to zero will turn off checking.

Parameters

<i>in</i>	<i>time</i>	is the time period in seconds
-----------	-------------	-------------------------------

See also

[setHeartBeatSend\(\);](#)

Definition at line 413 of file ArnRpc.cpp.

14.39.4.25 void ArnRpc::setHeartBeatSend (int *time*)

Set period time for sending heart beat message.

Setting time to zero will turn off sending.

Parameters

<i>in</i>	<i>time</i>	is the time period in seconds
-----------	-------------	-------------------------------

See also

[setHeartBeatCheck\(\)](#)

Definition at line 394 of file ArnRpc.cpp.

14.39.4.26 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.

14.39.4.27 void ArnRpc::setMethodPrefix (const QString & *prefix*)

Definition at line 337 of file ArnRpc.cpp.

14.39.4.28 void ArnRpc::setMode (Mode *mode*)

Definition at line 378 of file ArnRpc.cpp.

14.39.4.29 void ArnRpc::setPipe (ArnPipe * *pipe*)

Set pipe for this Rpc.

The Rpc will take ownership of the pip.

Parameters

<i>in</i>	<i>pipe</i>	
-----------	-------------	--

See also

[pipe\(\)](#)
[pipePath\(\)](#)

Definition at line 273 of file ArnRpc.cpp.

14.39.4.30 `bool ArnRpc::setReceiver (QObject * receiver, bool useTrackRpcSender = true)`

Definition at line 300 of file ArnRpc.cpp.

14.39.4.31 `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

<i>in</i>	<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 \(3.0.0\)](#)
- [src/ArnRpc.cpp \(3.0.0\)](#)

14.40 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 }

14.40.1 Detailed Description

Examples:

[ArnDemoChatServer/MainWindow.cpp](#).

Definition at line 84 of file ArnRpc.hpp.

14.40.2 Member Enumeration Documentation

14.40.2.1 enum `ArnRpcMode::E`

Enumerator

Provider Provider side (opposed to requester)

AutoDestroy Use *AutoDestroy* for the pipe, i.e. it is closed when tcp/ip is broken.

UuidPipe Use an unique uuid in the pipe name.

NoDefaultArgs If guarantied 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()`.

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 `defaultCall()` signal instead of error.

Debug Debug mode, dumping info for the batch connections.

UuidAutoDestroy Convenience, combined *UuidPipe* and *AutoDestroy*

AnyNamedArg Convenience, combined *NamedArg* and *NamedTypedArg*

Definition at line 88 of file `ArnRpc.hpp`.

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

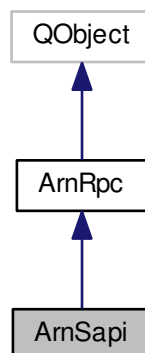
- `src/ArnInc/ArnRpc.hpp` (3.0.0)

14.41 ArnSapi Class Reference

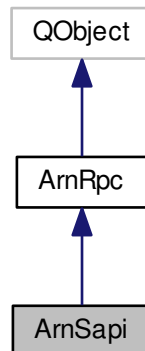
Service API.

```
#include <ArnSapi.hpp>
```

Inheritance diagram for ArnSapi:



Collaboration diagram for ArnSapi:



Public Member Functions

- [ArnSapi](#) (QObject *parent=0)
- bool [open](#) (const QString &[pipePath](#)=QString(), [Mode mode](#)=[Mode](#)(), const char *providerPrefix=0, const char *requesterPrefix=0)
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=0)
- void [setDefaultPath](#) (const QString &[defaultPath](#))
Set default path for the pipe to be used.

Additional Inherited Members

14.41.1 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 = 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);
};

// 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.

14.41.2 Constructor & Destructor Documentation

14.41.2.1 ArnSapi::ArnSapi (QObject * parent = 0) [explicit]

Examples:

[ArnDemoChatServer/ChatSapi.hpp](#).

Definition at line 47 of file ArnSapi.cpp.

14.41.2.2 ArnSapi::ArnSapi (const QString & defaultPath, QObject * parent = 0) [protected]

Definition at line 53 of file ArnSapi.cpp.

14.41.3 Member Function Documentation

14.41.3.1 `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 testT() 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 QRegExp&, const QObject*, const QString&, [Mode](#))

Definition at line 107 of file ArnSapi.cpp.

14.41.3.2 `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 testT() 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 QRegExp&, const QObject*, const QString&, [Mode](#))

Definition at line 89 of file ArnSapi.cpp.

14.41.3.3 `QString ArnSapi::defaultPath () const`

Get default path for the *pipe* to be used.

Returns

default path

Definition at line 118 of file ArnSapi.cpp.

14.41.3.4 `bool ArnSapi::open (const QString & pipePath = QString(), Mode mode = Mode(), const char * providerPrefix = 0, const char * requesterPrefix = 0)`

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.

14.41.3.5 `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:

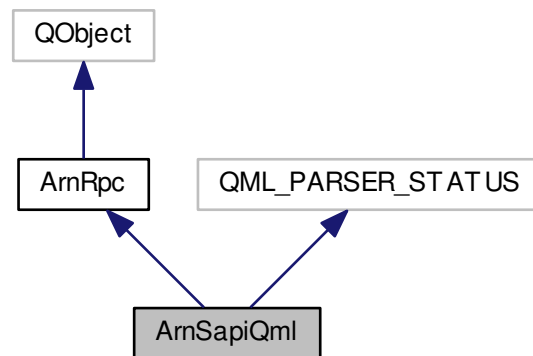
- [src/ArnInc/ArnSapi.hpp \(3.0.0\)](#)
- [src/ArnSapi.cpp \(3.0.0\)](#)

14.42 ArnSapiQml Class Reference

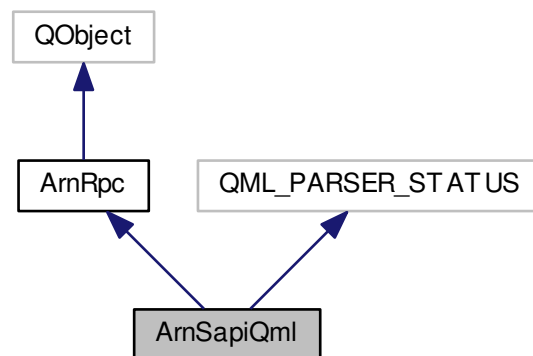
ARN Sapi QML.

```
#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](#) ()

Additional Inherited Members

14.42.1 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 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: "Msg: " + info.testMsg}
            Text {text: Arn.info} // ArnLib version info
        }

        function setTestMsg( msg) {
            info.testMsg = msg;
        }
    }
}
```

Definition at line 522 of file ArnQml.hpp.

14.42.2 Member Enumeration Documentation

14.42.2.1 enum ArnSapiQml::Mode

Enumerator

Provider Provider side (opposed to requester)

AutoDestroy Use *AutoDestroy* for the pipe, i.e. it is closed when tcp/ip is broken.

UuidPipe Use an unique uuid in the pipe name.

NoDefaultArgs If guarantied 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 *UuidPipe* and *AutoDestroy*

Definition at line 549 of file ArnQml.hpp.

14.42.3 Member Function Documentation

14.42.3.1 `bool ArnSapiQml::isHeartBeatOk () [inline],[slot]`

Definition at line 574 of file ArnQml.hpp.

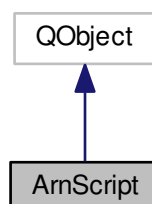
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQml.hpp \(3.0.0\)](#)
- [src/ArnQml.cpp \(3.0.0\)](#)

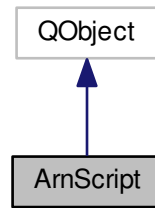
14.43 ArnScript Class Reference

```
#include <ArnScript.hpp>
```

Inheritance diagram for ArnScript:



Collaboration diagram for ArnScript:



Signals

- void `errorText` (QString txt)

Public Member Functions

- `ArnScript` (QObject *parent=0)
- `ArnScript` (QScriptEngine *engine, QObject *parent=0)
- QScriptEngine & `engine` () const
- bool `evaluate` (const QByteArray &script, const QString &idName)
- bool `evaluateFile` (const QString &fileName)
- bool `logUncaughtError` (QScriptValue &scriptValue)
- QString `idName` () const

Protected Member Functions

- void `errorLog` (const QString &errText, ArnError err=ArnError::Undef, void *reference=0)

Static Protected Member Functions

- static QScriptValue `printFunction` (QScriptContext *context, QScriptEngine *engine)

Protected Attributes

- QScriptEngine * `_engine`
- ArnItemProto * `_itemProto`
- ArnMonitorProto * `_monitorProto`
- ArnDepOfferProto * `_depOfferProto`
- ArnDepProto * `_depProto`

14.43.1 Detailed Description

Definition at line 160 of file ArnScript.hpp.

14.43.2 Constructor & Destructor Documentation

14.43.2.1 `ArnScript::ArnScript (QObject * parent = 0)` `[explicit]`

Definition at line 79 of file `ArnScript.cpp`.

14.43.2.2 `ArnScript::ArnScript (QScriptEngine * engine, QObject * parent = 0)`

Definition at line 86 of file `ArnScript.cpp`.

14.43.3 Member Function Documentation

14.43.3.1 `QScriptEngine & ArnScript::engine () const`

Definition at line 93 of file `ArnScript.cpp`.

14.43.3.2 `void ArnScript::errorLog (const QString & errText, ArnError err = ArnError::Undef, void * reference = 0)`
`[protected]`

Definition at line 217 of file `ArnScript.cpp`.

14.43.3.3 `void ArnScript::errorText (QString txt)` `[signal]`

14.43.3.4 `bool ArnScript::evaluate (const QByteArray & script, const QString & idName)`

Definition at line 99 of file `ArnScript.cpp`.

14.43.3.5 `bool ArnScript::evaluateFile (const QString & fileName)`

Definition at line 110 of file `ArnScript.cpp`.

14.43.3.6 `QString ArnScript::idName () const`

Definition at line 135 of file `ArnScript.cpp`.

14.43.3.7 `bool ArnScript::logUncaughtError (QScriptValue & scriptValue)`

Definition at line 119 of file `ArnScript.cpp`.

14.43.3.8 `QScriptValue ArnScript::printFunction (QScriptContext * context, QScriptEngine * engine)` `[static]`,
`[protected]`

Definition at line 149 of file `ArnScript.cpp`.

14.43.4 Member Data Documentation

14.43.4.1 `ArnDepOfferProto* ArnScript::_depOfferProto` `[protected]`

Definition at line 187 of file `ArnScript.hpp`.

14.43.4.2 ArnDepProto* ArnScript::_depProto [protected]

Definition at line 188 of file ArnScript.hpp.

14.43.4.3 QScriptEngine* ArnScript::_engine [protected]

Definition at line 184 of file ArnScript.hpp.

14.43.4.4 ArnItemProto* ArnScript::_itemProto [protected]

Definition at line 185 of file ArnScript.hpp.

14.43.4.5 ArnMonitorProto* ArnScript::_monitorProto [protected]

Definition at line 186 of file ArnScript.hpp.

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

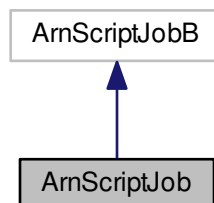
- [src/ArnInc/ArnScript.hpp \(3.0.0\)](#)
- [src/ArnScript.cpp \(3.0.0\)](#)

14.44 ArnScriptJob Class Reference

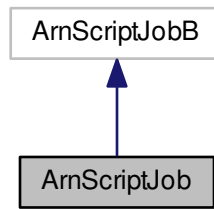
Interface class to be normally used, is also Script Job interface.

```
#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=0)

14.44.1 Detailed Description

Interface class to be normally used, is also Script Job interface.

Definition at line 123 of file ArnScriptJob.hpp.

14.44.2 Constructor & Destructor Documentation

14.44.2.1 [ArnScriptJob::ArnScriptJob](#) (int *id*, QObject * *parent* = 0) [explicit]

Definition at line 348 of file ArnScriptJob.cpp.

14.44.3 Member Function Documentation

14.44.3.1 void [ArnScriptJob::errorLog](#) (const QString & *txt*) [inline],[slot]

Definition at line 140 of file ArnScriptJob.hpp.

14.44.3.2 void [ArnScriptJob::quit](#) () [inline],[slot]

Definition at line 139 of file ArnScriptJob.hpp.

14.44.3.3 void ArnScriptJob::setWatchDogTime (int *time*) [inline],[slot]

Definition at line 137 of file ArnScriptJob.hpp.

14.44.3.4 void ArnScriptJob::sigQuit () [signal]

14.44.3.5 void ArnScriptJob::yield () [inline],[slot]

Definition at line 138 of file ArnScriptJob.hpp.

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

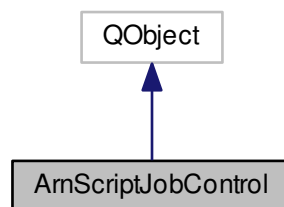
- src/ArnInc/ArnScriptJob.hpp (3.0.0)
- src/ArnScriptJob.cpp (3.0.0)

14.45 ArnScriptJobControl Class Reference

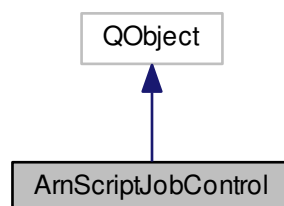
Is thread-safe (except doSetupJob)

```
#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=0)
- 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.

14.45.1 Detailed Description

Is thread-safe (except doSetupJob)

Definition at line 160 of file ArnScriptJob.hpp.

14.45.2 Constructor & Destructor Documentation

14.45.2.1 [ArnScriptJobControl::ArnScriptJobControl](#) ([QObject](#) * *parent* = 0) [explicit]

Definition at line 359 of file ArnScriptJob.cpp.

14.45.3 Member Function Documentation

14.45.3.1 void [ArnScriptJobControl::addConfig](#) ([QObject](#) * *obj*)

Definition at line 458 of file ArnScriptJob.cpp.

14.45.3.2 void [ArnScriptJobControl::addInterface](#) (const QString & *id*)

Definition at line 396 of file ArnScriptJob.cpp.

14.45.3.3 void [ArnScriptJobControl::addInterfaceList](#) (const QStringList & *interfaceList*)

Definition at line 405 of file ArnScriptJob.cpp.

14.45.3.4 `QVariant ArnScriptJobControl::config (const char * name) const`

Definition at line 492 of file ArnScriptJob.cpp.

14.45.3.5 `void ArnScriptJobControl::doSetupJob (ArnScriptJob * job, ArnScriptJobFactory * jobFactory)`

Not threadsafe, only run in same thread as script.

Definition at line 476 of file ArnScriptJob.cpp.

14.45.3.6 `void ArnScriptJobControl::errorText (const QString & txt) [signal]`

14.45.3.7 `int ArnScriptJobControl::id ()`

Definition at line 376 of file ArnScriptJob.cpp.

14.45.3.8 `void ArnScriptJobControl::loadScriptFile (const QString & fileName)`

Definition at line 434 of file ArnScriptJob.cpp.

14.45.3.9 `QString ArnScriptJobControl::name () const`

Definition at line 386 of file ArnScriptJob.cpp.

14.45.3.10 `QByteArray ArnScriptJobControl::script () const`

Definition at line 424 of file ArnScriptJob.cpp.

14.45.3.11 `void ArnScriptJobControl::scriptChanged (int id) [signal]`

14.45.3.12 `bool ArnScriptJobControl::setConfig (const char * name, const QVariant & value)`

Definition at line 446 of file ArnScriptJob.cpp.

14.45.3.13 `void ArnScriptJobControl::setName (const QString & name)`

Definition at line 368 of file ArnScriptJob.cpp.

14.45.3.14 `void ArnScriptJobControl::setScript (const QByteArray & script) [slot]`

Definition at line 414 of file ArnScriptJob.cpp.

14.45.3.15 `void ArnScriptJobControl::setThreaded (bool isThreaded)`

Definition at line 469 of file ArnScriptJob.cpp.

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

- [src/ArnInc/ArnScriptJob.hpp \(3.0.0\)](#)
- [src/ArnScriptJob.cpp \(3.0.0\)](#)

14.46 ArnScriptJobFactory Class Reference

Must be thread-safe as subclassed.

```
#include <ArnScriptJob.hpp>
```

Public Member Functions

- [ArnScriptJobFactory](#) ()
- virtual [~ArnScriptJobFactory](#) ()
- virtual bool [installExtension](#) (const QString &id, QScriptEngine &engine, const [ArnScriptJobControl](#) *jobControl=0)=0

Static Protected Member Functions

- static void [setupJsObj](#) (const QString &id, const QScriptValue &jsObj, QScriptEngine &engine)
- static bool [setupInterface](#) (const QString &id, QObject *interface, QScriptEngine &engine)

14.46.1 Detailed Description

Must be thread-safe as subclassed.

Definition at line 145 of file ArnScriptJob.hpp.

14.46.2 Constructor & Destructor Documentation

14.46.2.1 [ArnScriptJobFactory::ArnScriptJobFactory](#) () [explicit]

Definition at line 314 of file ArnScriptJob.cpp.

14.46.2.2 [ArnScriptJobFactory::~~ArnScriptJobFactory](#) () [virtual]

Definition at line 319 of file ArnScriptJob.cpp.

14.46.3 Member Function Documentation

14.46.3.1 virtual bool [ArnScriptJobFactory::installExtension](#) (const QString &id, QScriptEngine &engine, const [ArnScriptJobControl](#) *jobControl = 0) [pure virtual]

14.46.3.2 bool [ArnScriptJobFactory::setupInterface](#) (const QString &id, QObject *interface, QScriptEngine &engine) [static], [protected]

Definition at line 330 of file ArnScriptJob.cpp.

14.46.3.3 void [ArnScriptJobFactory::setupJsObj](#) (const QString &id, const QScriptValue &jsObj, QScriptEngine &engine) [static], [protected]

Definition at line 324 of file ArnScriptJob.cpp.

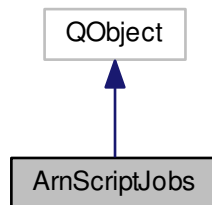
The documentation for this class was generated from the following files:

- src/ArnInc/[ArnScriptJob.hpp](#) (3.0.0)
- src/[ArnScriptJob.cpp](#) (3.0.0)

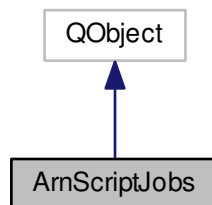
14.47 ArnScriptJobs Class Reference

```
#include <ArnScriptJobs.hpp>
```

Inheritance diagram for ArnScriptJobs:



Collaboration diagram for ArnScriptJobs:



Classes

- struct [Type](#)

Public Member Functions

- [ArnScriptJobs](#) (`QObject *parent=0`)
- void [addJob](#) (`ArnScriptJobControl *jobConfig`, `int prio=1`)
- void [setFactory](#) (`ArnScriptJobFactory *jobFactory`)
- void [start](#) (`Type type=Type::Cooperative`)

14.47.1 Detailed Description

TODO: Add destructor that deletes jobs in `_jobSlots`

Definition at line 88 of file `ArnScriptJobs.hpp`.

14.47.2 Constructor & Destructor Documentation

14.47.2.1 `ArnScriptJobs::ArnScriptJobs (QObject * parent = 0)` `[explicit]`

Definition at line 140 of file `ArnScriptJobs.cpp`.

14.47.3 Member Function Documentation

14.47.3.1 `void ArnScriptJobs::addJob (ArnScriptJobControl * jobConfig, int prio = 1)`

Definition at line 149 of file `ArnScriptJobs.cpp`.

14.47.3.2 `void ArnScriptJobs::setFactory (ArnScriptJobFactory * jobFactory)`

Definition at line 161 of file `ArnScriptJobs.cpp`.

14.47.3.3 `void ArnScriptJobs::start (Type type = Type::Cooperative)`

Definition at line 167 of file `ArnScriptJobs.cpp`.

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

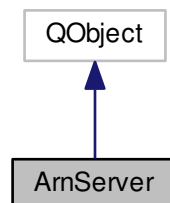
- `src/ArnInc/ArnScriptJobs.hpp` (3.0.0)
- `src/ArnScriptJobs.cpp` (3.0.0)

14.48 ArnServer Class Reference

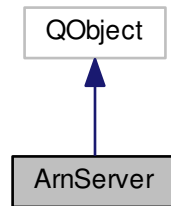
Class for making an [Arn Server](#).

```
#include <ArnServer.hpp>
```

Inheritance diagram for ArnServer:



Collaboration diagram for ArnServer:



Classes

- struct [Type](#)

Public Member Functions

- [ArnServer](#) ([Type](#) serverType, `QObject *parent=0`)
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.

14.48.1 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.

14.48.2 Constructor & Destructor Documentation

14.48.2.1 ArnServer::ArnServer (Type *serverType*, QObject * *parent* = 0)

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 196 of file ArnServer.cpp.

14.48.2.2 ArnServer::~~ArnServer ()

Definition at line 210 of file ArnServer.cpp.

14.48.3 Member Function Documentation

14.48.3.1 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 261 of file ArnServer.cpp.

14.48.3.2 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

<i>in</i>	<i>path</i>	is the freePath, eg "/Local/Sys/Legal/".
-----------	-------------	--

See also

[freePaths\(\)](#)

Definition at line 345 of file ArnServer.cpp.

14.48.3.3 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 354 of file ArnServer.cpp.

14.48.3.4 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 269 of file ArnServer.cpp.

14.48.3.5 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 301 of file ArnServer.cpp.

14.48.3.6 `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 <code>QHostAddress::Any</code>).
-----------	---

See also

[start\(\)](#)

Definition at line 252 of file ArnServer.cpp.

14.48.3.7 `QStringList ArnServer::noLoginNets () const`

Get the nets not demanding login.

Returns

the nets not demanding login.

See also

[setNoLoginNets\(\)](#)

Definition at line 293 of file ArnServer.cpp.

14.48.3.8 `int ArnServer::port ()`

Port number of the [Arn](#) server

Return values

<i>is</i>	the port number.
-----------	------------------

Definition at line 244 of file ArnServer.cpp.

14.48.3.9 `void ArnServer::setDemandLogin (bool isDemandLogin)`

Set servers demand for login.

If any of server or client demand login, it must be used.

Parameters

<i>in</i>	<i>isDemandLogin</i>	true if server demand login.
-----------	----------------------	------------------------------

See also

[isDemandLogin\(\)](#)

Definition at line 277 of file ArnServer.cpp.

14.48.3.10 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

<i>in</i>	<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 285 of file ArnServer.cpp.

14.48.3.11 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

<i>in</i>	<i>whoIamXsm</i>	contains the information.
-----------	------------------	---------------------------

See also

[remoteWhoIam\(\)](#)

Definition at line 386 of file ArnServer.cpp.

14.48.3.12 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 216 of file ArnServer.cpp.

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

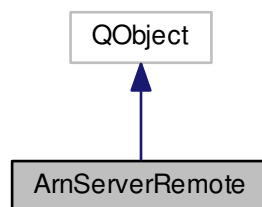
- [src/ArnInc/ArnServer.hpp \(3.0.0\)](#)
- [src/ArnServer.cpp \(3.0.0\)](#)

14.49 ArnServerRemote Class Reference

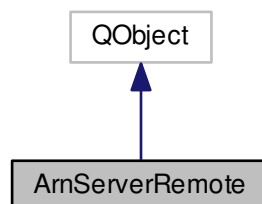
Class for remote controlling an [Arn Server](#).

```
#include <ArnServerRemote.hpp>
```

Inheritance diagram for ArnServerRemote:



Collaboration diagram for ArnServerRemote:

**Public Member Functions**

- [ArnServerRemote](#) (QObject *parent=0)
- [~ArnServerRemote](#) ()
- void [startUseServer](#) ([ArnServer](#) *arnServer)
Start making remote control objects for the [ArnServer](#).

14.49.1 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 121 of file ArnServerRemote.hpp.

14.49.2 Constructor & Destructor Documentation

14.49.2.1 ArnServerRemote::ArnServerRemote (QObject * *parent* = 0) [explicit]

Definition at line 309 of file ArnServerRemote.cpp.

14.49.2.2 ArnServerRemote::~ArnServerRemote ()

Definition at line 325 of file ArnServerRemote.cpp.

14.49.3 Member Function Documentation

14.49.3.1 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

<i>in</i>	<i>arnServer</i>	is the ArnServer to make remote controlled
-----------	------------------	--

See also

[ArnClient](#)

Definition at line 331 of file ArnServerRemote.cpp.

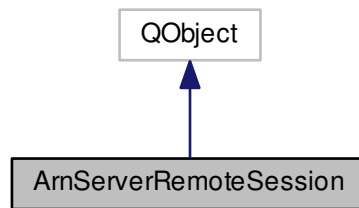
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnServerRemote.hpp \(3.0.0\)](#)
- [src/ArnServerRemote.cpp \(3.0.0\)](#)

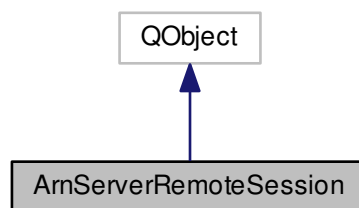
14.50 ArnServerRemoteSession Class Reference

```
#include <ArnServerRemote.hpp>
```

Inheritance diagram for ArnServerRemoteSession:



Collaboration diagram for ArnServerRemoteSession:



Public Types

- typedef
[ArnServerRemoteSessionKillMode](#) KillMode

Public Member Functions

- [ArnServerRemoteSession](#) ([ArnServerSession](#) *arnServerSession, [ArnServerRemote](#) *arnServerRemote)

14.50.1 Detailed Description

Definition at line 60 of file `ArnServerRemote.hpp`.

14.50.2 Member Typedef Documentation

14.50.2.1 typedef ArnServerRemoteSessionKillMode ArnServerRemoteSession::KillMode

Definition at line 64 of file `ArnServerRemote.hpp`.

14.50.3 Constructor & Destructor Documentation

14.50.3.1 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 (3.0.0)
- src/ArnServerRemote.cpp (3.0.0)

14.51 ArnServerRemoteSessionKillMode Class Reference

```
#include <ArnServerRemote.hpp>
```

Public Types

- enum [E](#) { [Off](#), [Delay10Sec](#), [Delay60Sec](#) }

14.51.1 Detailed Description

Definition at line 47 of file ArnServerRemote.hpp.

14.51.2 Member Enumeration Documentation

14.51.2.1 enum ArnServerRemoteSessionKillMode::E

Enumerator

Off

Delay10Sec

Delay60Sec

Definition at line 51 of file ArnServerRemote.hpp.

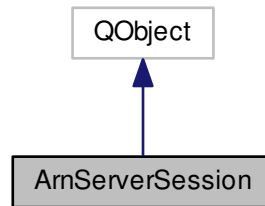
The documentation for this class was generated from the following file:

- src/ArnInc/ArnServerRemote.hpp (3.0.0)

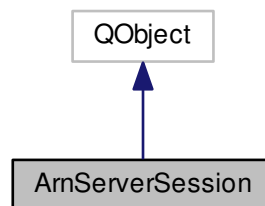
14.52 ArnServerSession Class Reference

```
#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

14.52.1 Detailed Description

Definition at line 53 of file `ArnServer.hpp`.

14.52.2 Constructor & Destructor Documentation

14.52.2.1 ArnServerSession::ArnServerSession (QTcpSocket * *socket*, ArnServer * *arnServer*)

Definition at line 49 of file ArnServer.cpp.

14.52.3 Member Function Documentation

14.52.3.1 Arn::Allow ArnServerSession::getAllow () const

Definition at line 151 of file ArnServer.cpp.

14.52.3.2 bool ArnServerSession::getTraffic (quint64 & *in*, quint64 & *out*) const

Definition at line 167 of file ArnServer.cpp.

14.52.3.3 void ArnServerSession::infoReceived (int *type*) [signal]

14.52.3.4 void ArnServerSession::loginCompleted () [signal]

14.52.3.5 QString ArnServerSession::loginUserName () const

Definition at line 143 of file ArnServer.cpp.

14.52.3.6 void ArnServerSession::messageReceived (int *type*, const QByteArray & *data*) [signal]

14.52.3.7 Arn::XStringMap ArnServerSession::remoteWhoIAm () const

Definition at line 135 of file ArnServer.cpp.

14.52.3.8 void ArnServerSession::sendMessage (int *type*, const QByteArray & *data* = QByteArray())

Definition at line 159 of file ArnServer.cpp.

14.52.3.9 QTcpSocket * ArnServerSession::socket () const

Definition at line 129 of file ArnServer.cpp.

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

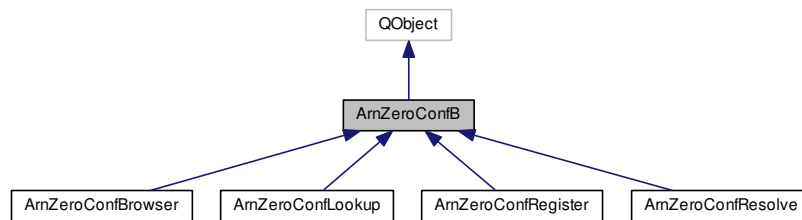
- [src/ArnInc/ArnServer.hpp \(3.0.0\)](#)
- [src/ArnServer.cpp \(3.0.0\)](#)

14.53 ArnZeroConfB Class Reference

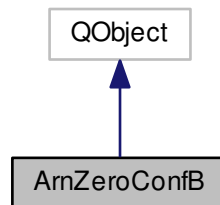
Base class for Zero Config.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfB:



Collaboration diagram for ArnZeroConfB:



Public Member Functions

- [ArnZeroConfB](#) (QObject *parent=0)
- 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.

14.53.1 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.

14.53.2 Constructor & Destructor Documentation

14.53.2.1 ArnZeroConfB::ArnZeroConfB (QObject * *parent* = 0)

Definition at line 83 of file ArnZeroConf.cpp.

14.53.2.2 ArnZeroConfB::~ArnZeroConfB () [virtual]

Definition at line 102 of file ArnZeroConf.cpp.

14.53.3 Member Function Documentation

14.53.3.1 QString ArnZeroConfB::domain () const

Returns the domain for this Zero Config.

Returns

current domain.

See also

[setDomain\(\)](#)

Definition at line 290 of file ArnZeroConf.cpp.

14.53.3.2 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 325 of file ArnZeroConf.cpp.

14.53.3.3 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 261 of file ArnZeroConf.cpp.

14.53.3.4 void ArnZeroConfB::setDomain (const QString & domain)

Sets the domain for this Zero Config.

Default set by this class is "local".

Parameters

<i>in</i>	<i>domain</i>	
-----------	---------------	--

See also

[domain\(\)](#)

Definition at line 296 of file ArnZeroConf.cpp.

14.53.3.5 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

<i>in</i>	<i>type</i>	is the service type (se above).
-----------	-------------	---------------------------------

See also

[serviceType\(\)](#)

Definition at line 267 of file ArnZeroConf.cpp.

14.53.3.6 void ArnZeroConfB::setSocketType (QAbstractSocket::SocketType type)

Sets the socket type for this Zero Config.

Allowed Socket type is: QAbstractSocket::TcpSocket, QAbstractSocket::UdpSocket.

Parameters

<i>in</i>	<i>type</i>	is one of the allowed types.
-----------	-------------	------------------------------

See also

[socketType\(\)](#)

Definition at line 255 of file ArnZeroConf.cpp.

14.53.3.7 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 249 of file ArnZeroConf.cpp.

14.53.3.8 ArnZeroConf::State ArnZeroConfB::state () const

Returns the current state of the service.

Return values

<i>the</i>	state of the service
------------	----------------------

Definition at line 191 of file ArnZeroConf.cpp.

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

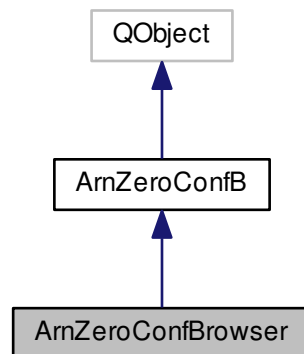
- [src/ArnInc/ArnZeroConf.hpp \(3.0.0\)](#)
- [src/ArnZeroConf.cpp \(3.0.0\)](#)

14.54 ArnZeroConfBrowser Class Reference

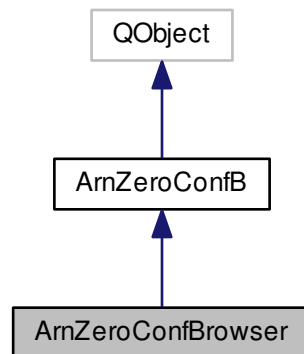
Browsing for ZeroConfig services.

```
#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 unsuccessful browsing.*

Public Member Functions

- [ArnZeroConfBrowser](#) (QObject *parent=0)
 - Standard constructor of an [ArnZeroConfBrowser](#) object.
- [ArnZeroConfBrowser](#) (const QString &serviceType, QObject *parent=0)
 - 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](#)

14.54.1 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.

14.54.2 Constructor & Destructor Documentation

14.54.2.1 ArnZeroConfBrowser::ArnZeroConfBrowser (QObject * *parent* = 0)

Standard constructor of an [ArnZeroConfBrowser](#) object.

All needed for browsing an "arn" service type.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 890 of file ArnZeroConf.cpp.

14.54.2.2 ArnZeroConfBrowser::ArnZeroConfBrowser (const QString & *serviceType*, QObject * *parent* = 0)

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 897 of file ArnZeroConf.cpp.

14.54.2.3 ArnZeroConfBrowser::~ArnZeroConfBrowser () [virtual]

Destructor of an [ArnZeroConfBrowser](#) object.

If browsing is active, it will be stopped.

Definition at line 905 of file ArnZeroConf.cpp.

14.54.3 Member Function Documentation

14.54.3.1 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 915 of file ArnZeroConf.cpp.

14.54.3.2 void ArnZeroConfBrowser::browse (bool *enable* = true) [slot]

Change state of browsing.

When browsing is started, services will be discovered.

Parameters

<i>in</i>	<i>enable</i>	if true browsing is started, otherwise it is stopped
-----------	---------------	--

See also

[stopBrowse\(\)](#)

Definition at line 947 of file ArnZeroConf.cpp.

14.54.3.3 void ArnZeroConfBrowser::browseError (int *errorCode*) [signal]

Indicate unsuccessful browsing.

Parameters

<i>in</i>	<i>errorCode</i>	
-----------	------------------	--

See also

[browse\(\)](#)

14.54.3.4 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.

14.54.3.5 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 927 of file ArnZeroConf.cpp.

14.54.3.6 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\(\)](#)

14.54.3.7 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\(\)](#)

14.54.3.8 int ArnZeroConfBrowser::serviceNameTold (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 921 of file ArnZeroConf.cpp.

14.54.3.9 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\(\)](#)

14.54.3.10 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 933 of file ArnZeroConf.cpp.

14.54.3.11 void ArnZeroConfBrowser::stopBrowse () [slot]

Stop browsing.

See also

[browse\(\)](#)

Definition at line 981 of file ArnZeroConf.cpp.

14.54.3.12 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 939 of file ArnZeroConf.cpp.

14.54.4 Friends And Related Function Documentation

14.54.4.1 friend class ArnZeroConfIntern [friend]

Definition at line 938 of file ArnZeroConf.hpp.

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

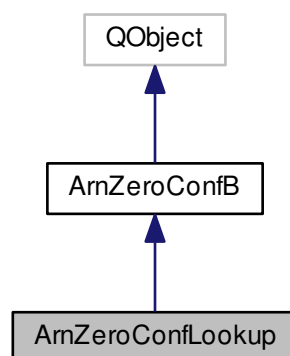
- [src/ArnInc/ArnZeroConf.hpp \(3.0.0\)](#)
- [src/ArnZeroConf.cpp \(3.0.0\)](#)

14.55 ArnZeroConfLookup Class Reference

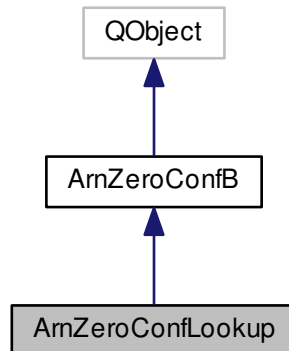
Lookup a host.

```
#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 unsuccessfull lookup of host.

Public Member Functions

- `ArnZeroConfLookup` (QObject *parent=0)
Standard constructor of an `ArnZeroConfLookup` object.
- `ArnZeroConfLookup` (const QString &hostName, QObject *parent=0)
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](#)

14.55.1 Detailed Description

Lookup a host.

[About Zero Config](#)

This class handles lookup of a host. It can be booth 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(lookuped(int)), this, SLOT(onLookuped(int)));
ds->lookup();

void XXX::onLookuped( 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.

14.55.2 Constructor & Destructor Documentation

14.55.2.1 ArnZeroConfLookup::ArnZeroConfLookup (QObject * *parent* = 0)

Standard constructor of an [ArnZeroConfLookup](#) object.

Parameters

<i>in</i>	<i>parent</i>
-----------	---------------

Definition at line 685 of file ArnZeroConf.cpp.

14.55.2.2 ArnZeroConfLookup::ArnZeroConfLookup (const QString & *hostName*, QObject * *parent* = 0)

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 692 of file ArnZeroConf.cpp.

14.55.2.3 ArnZeroConfLookup::~ArnZeroConfLookup () [virtual]

Destructor of an [ArnZeroConfLookup](#) object.

If the lookup is ongoing, it will be released.

Definition at line 701 of file ArnZeroConf.cpp.

14.55.3 Member Function Documentation

14.55.3.1 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.

14.55.3.2 QHostAddress ArnZeroConfLookup::hostAddr () const [inline]

Returns the host address for this Lookup.

Returns

current host adress

Definition at line 838 of file ArnZeroConf.hpp.

14.55.3.3 int ArnZeroConfLookup::id () const

Returns the id number for this lookup.

Return values

<i>the</i>	id number
------------	-----------

See also

[setId\(\)](#)

Definition at line 711 of file ArnZeroConf.cpp.

14.55.3.4 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 868 of file ArnZeroConf.cpp.

14.55.3.5 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 [lookuped\(\)](#) and [lookupError\(\)](#) signals.

Parameters

<i>in</i>	<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

[lookuped\(\)](#)
[lookupError\(\)](#)

Definition at line 723 of file ArnZeroConf.cpp.

14.55.3.6 void ArnZeroConfLookup::lookuped (int *id*) [signal]

Indicate successfull lookup of host.

Parameters

<i>in</i>	<i>id</i>	is the id number for this lookup
-----------	-----------	----------------------------------

See also

[lookup\(\)](#)

14.55.3.7 void ArnZeroConfLookup::lookupError (int *id*, int *code*) [signal]

Indicate unsuccessfull lookup of host.

Parameters

<i>in</i>	<i>id</i>	is the id number for this lookup
<i>in</i>	<i>code</i>	error code.

See also

[lookup\(\)](#)

14.55.3.8 void ArnZeroConfLookup::releaseLookup ()

Release the lookup.

Any lookup attempts in progress will be aborted.

Definition at line 779 of file ArnZeroConf.cpp.

14.55.3.9 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 874 of file ArnZeroConf.cpp.

14.55.3.10 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.

14.55.3.11 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 717 of file ArnZeroConf.cpp.

14.55.4 Friends And Related Function Documentation

14.55.4.1 friend class ArnZeroConfIntern [friend]

Definition at line 785 of file ArnZeroConf.hpp.

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

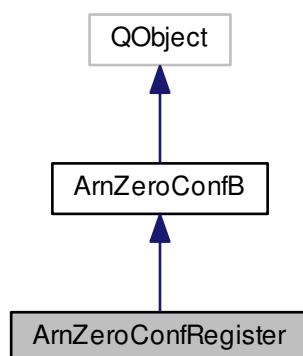
- src/ArnInc/[ArnZeroConf.hpp](#) (3.0.0)
- src/[ArnZeroConf.cpp](#) (3.0.0)

14.56 ArnZeroConfRegister Class Reference

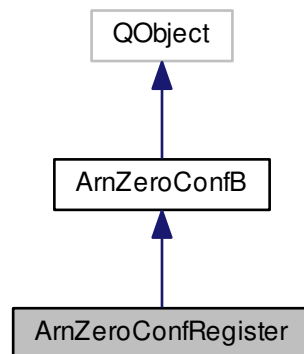
Registering a ZeroConfig service.

```
#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=0)
Standard constructor of an [ArnZeroConfRegister](#) object.
- [ArnZeroConfRegister](#) (const QString &[serviceName](#), QObject *parent=0)
Constructor of an [ArnZeroConfRegister](#) object.
- [ArnZeroConfRegister](#) (const QString &[serviceName](#), const QString &[serviceType](#), quint16 [port](#), QObject *parent=0)
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](#)

14.56.1 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.

14.56.2 Constructor & Destructor Documentation

14.56.2.1 ArnZeroConfRegister::ArnZeroConfRegister (QObject * parent = 0)

Standard constructor of an [ArnZeroConfRegister](#) object.

The service name can be automatically generated based on the system's hostname.

Parameters

<i>in</i>	<i>parent</i>	
-----------	---------------	--

Definition at line 370 of file ArnZeroConf.cpp.

14.56.2.2 ArnZeroConfRegister::ArnZeroConfRegister (const QString & *serviceName*, QObject * *parent* = 0)

Constructor of an [ArnZeroConfRegister](#) object.

All needed parameters for an "arn" service type, using standard arn-port at this computer.

Parameters

<i>in</i>	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
<i>in</i>	<i>parent</i>	

Definition at line 377 of file ArnZeroConf.cpp.

14.56.2.3 ArnZeroConfRegister::ArnZeroConfRegister (const QString & *serviceName*, const QString & *serviceType*, quint16 *port*, QObject * *parent* = 0)

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

<i>in</i>	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
<i>in</i>	<i>serviceType</i>	the service type, e.g. "arn" or "_arn._tcp".
<i>in</i>	<i>port</i>	the service port num
<i>in</i>	<i>parent</i>	

Definition at line 386 of file ArnZeroConf.cpp.

14.56.2.4 ArnZeroConfRegister::~ArnZeroConfRegister () [virtual]

Destructor of an [ArnZeroConfRegister](#) object.

If the service is registered, it will be unregistered.

Definition at line 398 of file ArnZeroConf.cpp.

14.56.3 Member Function Documentation

14.56.3.1 void ArnZeroConfRegister::addSubType (const QString & *subtype*) [inline]

Add a subtype to the list of current subtypes.

Parameters

<i>in</i>	<i>subtype</i>	the subtype to add, e.g. "myGroup1"
-----------	----------------	-------------------------------------

See also

[subTypes\(\)](#)
[setSubTypes\(\)](#)

Definition at line 427 of file ArnZeroConf.hpp.

14.56.3.2 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 409 of file ArnZeroConf.cpp.

14.56.3.3 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::to←XString\(\)](#) 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

[setTxtRecordMap\(\)](#)
[Arn::XStringMap](#)

Definition at line 509 of file ArnZeroConf.hpp.

14.56.3.4 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.

14.56.3.5 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.

14.56.3.6 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

in	<i>serviceName</i>	is the realy registered name e.g. "My House Registry (2)"
----	--------------------	---

See also

[registerService\(\)](#)
[setServiceName\(\)](#)
[serviceName\(\)](#)

14.56.3.7 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 422 of file ArnZeroConf.cpp.

14.56.3.8 void ArnZeroConfRegister::registrationError (int *code*) [signal]

Indicate unsuccessfull registration of service.

Parameters

in	<i>code</i>	error code.
----	-------------	-------------

See also

[registerService\(\)](#)

14.56.3.9 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 467 of file ArnZeroConf.cpp.

14.56.3.10 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.

14.56.3.11 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.

14.56.3.12 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

<i>in</i>	<i>port</i>	the port number
-----------	-------------	-----------------

See also

[port\(\)](#)

Definition at line 443 of file ArnZeroConf.hpp.

14.56.3.13 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 guaranted 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

<i>in</i>	<i>name</i>	is service name, e.g. "My House Registry"
-----------	-------------	---

See also

[serviceName\(\)](#)
[currentServiceName\(\)](#)
[registered\(\)](#)

Definition at line 415 of file ArnZeroConf.cpp.

14.56.3.14 void ArnZeroConfRegister::setSubTypes (const QStringList & *subtypes*) [inline]

Sets the list of current subtypes.

Parameters

<i>in</i>	<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.

14.56.3.15 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

<i>in</i>	<i>txt</i>	is The Txt Record (in binary format)
-----------	------------	--------------------------------------

See also

[txtRecord\(\)](#)
[setTxtRecordMap\(\)](#)

Definition at line 540 of file ArnZeroConf.hpp.

14.56.3.16 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.

14.56.3.17 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.

14.56.3.18 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.

14.56.4 Friends And Related Function Documentation

14.56.4.1 friend class ArnZeroConfIntern [friend]

Definition at line 368 of file ArnZeroConf.hpp.

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

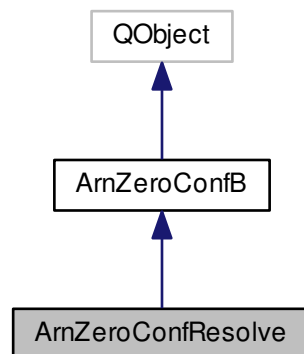
- [src/ArnInc/ArnZeroConf.hpp \(3.0.0\)](#)
- [src/ArnZeroConf.cpp \(3.0.0\)](#)

14.57 ArnZeroConfResolve Class Reference

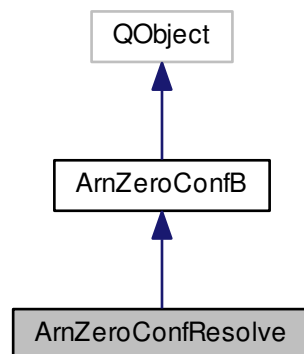
Resolv a ZeroConfig service.

```
#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 unsuccessfull resolve of service.

Public Member Functions

- [ArnZeroConfResolve](#) (QObject *parent=0)
Standard constructor of an ArnZeroConfResolv object.
- [ArnZeroConfResolve](#) (const QString &[serviceName](#), QObject *parent=0)
Constructor of an ArnZeroConfResolv object.
- [ArnZeroConfResolve](#) (const QString &[serviceName](#), const QString &[serviceType](#), QObject *parent=0)
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](#)

14.57.1 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.

14.57.2 Constructor & Destructor Documentation

14.57.2.1 ArnZeroConfResolve::ArnZeroConfResolve (QObject * *parent* = 0)

Standard constructor of an ArnZeroConfResolv object.

Parameters

in	<i>parent</i>	
----	---------------	--

Definition at line 523 of file ArnZeroConf.cpp.

14.57.2.2 ArnZeroConfResolve::ArnZeroConfResolve (const QString & *serviceName*, QObject * *parent* = 0)

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 530 of file ArnZeroConf.cpp.

14.57.2.3 ArnZeroConfResolve::ArnZeroConfResolve (const QString & *serviceName*, const QString & *serviceType*, QObject * *parent* = 0)

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 539 of file ArnZeroConf.cpp.

14.57.2.4 ArnZeroConfResolve::~ArnZeroConfResolve () [virtual]

Destructor of an ArnZeroConfResolv object.

If the service is registered, it will be unregistered.

Definition at line 550 of file ArnZeroConf.cpp.

14.57.3 Member Function Documentation

14.57.3.1 `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

<code>out</code>	<code>xsm</code>	is the loaded XStringMap if successfull, otherwise undefined.
------------------	------------------	---

Return values

<code>true</code>	if successfull.
-------------------	-----------------

See also

[Arn::XStringMap](#)

Definition at line 703 of file ArnZeroConf.hpp.

14.57.3.2 `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.

14.57.3.3 `int ArnZeroConfResolve::id () const`

Returns the id number for this resolv.

Returns

the id number

See also

[setId\(\)](#)

Definition at line 560 of file ArnZeroConf.cpp.

14.57.3.4 `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.

14.57.3.5 void ArnZeroConfResolve::releaseResolve ()

Release the resolving.

Any resolve attempts in progress will be aborted.

Definition at line 610 of file ArnZeroConf.cpp.

14.57.3.6 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 572 of file ArnZeroConf.cpp.

14.57.3.7 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\(\)](#)

14.57.3.8 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\(\)](#)

14.57.3.9 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.

14.57.3.10 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	id	the id number
----	----	---------------

See also

[id\(\)](#)

Definition at line 566 of file ArnZeroConf.cpp.

14.57.3.11 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	name	is service name, e.g. "My House Registry"
----	------	---

See also

[serviceName\(\)](#)

Definition at line 691 of file ArnZeroConf.hpp.

14.57.3.12 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.

14.57.4 Friends And Related Function Documentation

14.57.4.1 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 (3.0.0)
- src/ArnZeroConf.cpp (3.0.0)

14.58 Arn::Coding Struct Reference

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

Public Types

- enum [E](#) { [Binary](#) = 0x0000, [Text](#) = 0x1000 }

14.58.1 Detailed Description

Definition at line 181 of file Arn.hpp.

14.58.2 Member Enumeration Documentation

14.58.2.1 enum Arn::Coding::E

Enumerator

Binary No special coding, can be anything.

Text Text coding, can be any character set.

Definition at line 182 of file Arn.hpp.

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

- src/ArnInc/Arn.hpp (3.0.0)

14.59 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 }

14.59.1 Detailed Description

Data type of an [Arn Data Object](#)

Definition at line 74 of file Arn.hpp.

14.59.2 Member Enumeration Documentation

14.59.2.1 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 \(3.0.0\)](#)

14.60 Arn::EnumTxt Class Reference

Class Enum text.

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

Public Member Functions

- [EnumTxt](#) (const QMetaObject &metaObj, bool isFlag, const [_InitEnumTxt](#) *initTxt, const char *name)
- 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) 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) const
Returns the text for a enum value in a namespace.
- int [getEnumVal](#) (const char *txt, int defaultVal=0, quint16 nameSpace=0)
Returns the enum value for a text in a namespace.
- int [getEnumVal](#) (const QString &txt, int defaultVal=0, quint16 nameSpace=0)
Returns the enum value for a text in a namespace.
- void [addBitSet](#) ([Arn::XStringMap](#) &xsm, quint16 nameSpace=0, bool neverHumanize=false)
Adds bit set for enum flags to a [XStringMap](#).
- QString [getBitSet](#) (quint16 nameSpace=0, bool neverHumanize=false)
returns the bit set string for enum flags

- QString [flagsToString](#) (int val, quint16 nameSpace=0)
returns text string for enum flags
- QStringList [flagsToStringList](#) (int val, quint16 nameSpace=0)
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 [addEnumSet](#) (Arn::XStringMap &xsm, quint16 nameSpace=0, bool neverHumanize=false)
Adds enum set to a [XStringMap](#).
- QString [getEnumSet](#) (quint16 nameSpace=0, bool neverHumanize=false)
returns the enum set string
- const char * [name](#) () const
returns the name of the enum (class)
- void [setMissingTxt](#) (quint16 toNameSpace, quint16 fromNameSpace=0, bool neverHumanize=false)
Copies missing enum texts from one namespace to another.

Static Public Member Functions

- static QString [humanize](#) (const QString &txt)
returns the humanized text

14.60.1 Detailed Description

Class Enum text.

Example usage

```
class AllowClassT {
    Q_GADGET
    Q_ENUMS(E)
public:
    enum E {
        None      = 0x00,
        Read       = 0x01,
        Create      = 0x04,
        Delete      = 0x08,
        All         = 0xff
    };
    MQ_DECLARE_FLAGSTXT( AllowClassT)

    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, TriedAll, "Tried all" },
        { NsHuman, MQ_NSTXT_FILL_MISSING_FROM( NsEnum) }
    )
};

```

Definition at line 118 of file MQFlags.hpp.

14.60.2 Constructor & Destructor Documentation

14.60.2.1 `Arn::EnumTxt::EnumTxt (const QMetaObject & metaObj, bool isFlag, const _InitEnumTxt * initTxt, const char * name)`

Definition at line 59 of file MQFlags.cpp.

14.60.3 Member Function Documentation

14.60.3.1 `void Arn::EnumTxt::addBitSet (Arn::XStringMap & xsm, quint16 nameSpace = 0, bool neverHumanize = false)`

Adds bit set for enum flags to a [XStringMap](#).

Example

```

Arn::XStringMap xsm;
xsm.add("T", "Test");
AllowClassT::txt().addBitSet( xsm);

```

will give xsm containing: T=Test B0=Read B2=Create B3=Delete

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

[humanize\(\)](#)

Definition at line 140 of file MQFlags.cpp.

14.60.3.2 `void Arn::EnumTxt::addEnumSet (Arn::XStringMap & xsm, quint16 nameSpace = 0, bool neverHumanize = false)`

Adds enum set to a [XStringMap](#).

Example

```

Arn::XStringMap xsm;
xsm.add("T", "Test");
ConnectStatT::txt().addEnumSet( 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[humanize\(\)](#)

Definition at line 241 of file MQFlags.cpp.

14.60.3.3 int Arn::EnumTxt::flagsFromString (const QString & *flagString*, quint16 *nameSpace* = 0)

returns enum flags from string

Example

```
QString flagString = "Create | Delete";
int val = AllowClassT::txt().flagsFromString( flagString);
```

will give val containing: 0xc (0x4 + 0x8)

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.

Definition at line 211 of file MQFlags.cpp.

14.60.3.4 int Arn::EnumTxt::flagsFromStringList (const QStringList & *flagStrings*, quint16 *nameSpace* = 0)

returns enum flags from string list

Example

```
QStringList flagStrings;
flagStrings << "Create" << "Delete";
int val = AllowClassT::txt().flagsFromString( flagStrings);
```

will give val containing: 0xc (0x4 + 0x8)

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.

Definition at line 219 of file MQFlags.cpp.

14.60.3.5 QString Arn::EnumTxt::flagsToString (int *val*, quint16 *nameSpace* = 0)

returns text string for enum flags

Example

```
AllowClassT allow;  
allow = allow.Create | allow.Delete;  
QDebug() << AllowClassT::txt().flagsToString( allow);
```

will print: "Create | Delete"

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.

Definition at line 180 of file MQFlags.cpp.

14.60.3.6 QStringList Arn::EnumTxt::flagsToStringList (int *val*, quint16 *nameSpace* = 0)

returns string list for enum flags

Example

```
AllowClassT allow;  
allow = allow.Create | allow.Delete;  
QStringList allowList = AllowClassT::txt().flagsToStringList( allow);
```

will give allowList containing: "Create", "Delete"

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.

Definition at line 188 of file MQFlags.cpp.

14.60.3.7 QString Arn::EnumTxt::getBitSet (quint16 *nameSpace* = 0, bool *neverHumanize* = false)

returns the bit set string for enum flags

Example

```
QDebug() << AllowClassT::txt().getBitSet();
```

will print: "B0=Read B2=Create B3=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 bit set string.

See also

[humanize\(\)](#)

Definition at line 170 of file MQFlags.cpp.

14.60.3.8 QString Arn::EnumTxt::getEnumSet (quint16 *nameSpace* = 0, bool *neverHumanize* = false)

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 265 of file MQFlags.cpp.

14.60.3.9 int Arn::EnumTxt::getEnumVal (const char * *txt*, int *defaultVal* = 0, quint16 *nameSpace* = 0)

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.

Returns

the enum value.

See also

[setTxt\(\);](#)

Definition at line 116 of file MQFlags.cpp.

14.60.3.10 `int Arn::EnumTxt::getEnumVal (const QString & txt, int defaultVal = 0, quint16 nameSpace = 0)`

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.

Returns

the enum value.

See also

[setTxt\(\);](#)
[setTxtString\(\);](#)

Definition at line 134 of file MQFlags.cpp.

14.60.3.11 `const char * Arn::EnumTxt::getTxt (int enumVal, quint16 nameSpace = 0) 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.

Returns

the enum text.

See also

[setTxt\(\);](#)

Definition at line 98 of file MQFlags.cpp.

14.60.3.12 `QString Arn::EnumTxt::getTxtString (int enumVal, quint16 nameSpace = 0) 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.

Returns

the enum text.

See also

[setTxt\(\);](#)
[setTxtString\(\);](#)

Definition at line 110 of file MQFlags.cpp.

14.60.3.13 QString Arn::EnumTxt::humanize (const QString & txt) [static]

returns the humanized text

The input text can be Chamel-case or '_' word separeted. 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

in	txt	is the text to be humanized.
----	-----	------------------------------

Returns

the humanized text.

Definition at line 301 of file MQFlags.cpp.

14.60.3.14 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 362 of file MQFlags.cpp.

14.60.3.15 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	toNameSpace	is the altered one. Can not be 0.
in	fromNameSpace	is the one to copy from.
in	neverHumanize	if true never applies the enum text humanize algorithm.

See also

[humanize\(\)](#)

Definition at line 273 of file MQFlags.cpp.

14.60.3.16 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)
```

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 84 of file MQFlags.cpp.

14.60.3.17 void Arn::EnumTxt::setTxtRef (const char * *txt*, int *enumVal*, quint16 *nameSpace*)

Definition at line 77 of file MQFlags.cpp.

14.60.3.18 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 104 of file MQFlags.cpp.

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

- [src/ArnInc/MQFlags.hpp \(3.0.0\)](#)
- [src/MQFlags.cpp \(3.0.0\)](#)

14.61 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 }

14.61.1 Detailed Description

Errors of ZeroConfig, other values are defined in `dns_sd.h`.

Definition at line 53 of file `ArnZeroConf.hpp`.

14.61.2 Member Enumeration Documentation

14.61.2.1 enum `ArnZeroConf::Error::E`

Enumerator

Ok *Ok*, defined as `kDNSServiceErr_NoError` in `dns_sd.h`.

Running *Operation in progress.*

BadReqSeq *Bad request sequence.*

Timeout *Operation timeout.*

UDnsFail *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` (3.0.0)

14.62 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 }

14.62.1 Detailed Description

Code used in blob for `arnExport()` and `arnImport()`

Definition at line 92 of file `Arn.hpp`.

14.62.2 Member Enumeration Documentation

14.62.2.1 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 \(3.0.0\)](#)

14.63 ArnClient::HostAddrPort Struct Reference

```
#include <ArnClient.hpp>
```

Public Member Functions

- [HostAddrPort \(\)](#)

Public Attributes

- [QString addr](#)
- [quint16 port](#)

14.63.1 Detailed Description

Definition at line 111 of file ArnClient.hpp.

14.63.2 Constructor & Destructor Documentation

14.63.2.1 [ArnClient::HostAddrPort::HostAddrPort \(\)](#) `[inline]`

Definition at line 115 of file ArnClient.hpp.

14.63.3 Member Data Documentation

14.63.3.1 [QString ArnClient::HostAddrPort::addr](#)

Definition at line 112 of file ArnClient.hpp.

14.63.3.2 [quint16 ArnClient::HostAddrPort::port](#)

Definition at line 113 of file ArnClient.hpp.

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

- [src/ArnInc/ArnClient.hpp \(3.0.0\)](#)

14.64 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](#) }

14.64.1 Detailed Description

Info type for exchange static (meta) info between [ArnClient](#) and [ArnServer](#).

Definition at line 107 of file Arn.hpp.

14.64.2 Member Enumeration Documentation

14.64.2.1 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 \(3.0.0\)](#)

14.65 ArnRpc::Invoke Struct Reference

```
#include <ArnRpc.hpp>
```

Public Types

- enum [E](#) { [NoQueue](#) = 0x01 }

14.65.1 Detailed Description

Definition at line 163 of file ArnRpc.hpp.

14.65.2 Member Enumeration Documentation

14.65.2.1 enum ArnRpc::Invoke::E

Enumerator

NoQueue This invoke is not queued, multiple calls to same method might overwrite.

Definition at line 164 of file ArnRpc.hpp.

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

- [src/ArnInc/ArnRpc.hpp \(3.0.0\)](#)

14.66 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 }

14.66.1 Detailed Description

Link flags when accessing an [Arn Data Object](#)

Definition at line 155 of file Arn.hpp.

14.66.2 Member Enumeration Documentation

14.66.2.1 enum Arn::LinkFlags::E

Enumerator

Folder

CreateAllowed

SilentError

LastLink

Threaded

Definition at line 156 of file Arn.hpp.

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

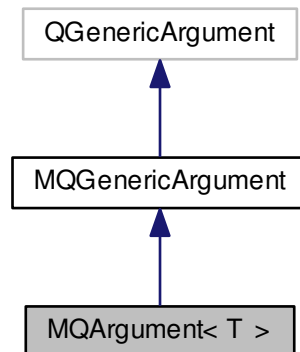
- [src/ArnInc/Arn.hpp \(3.0.0\)](#)

14.67 MQArgument< T > Class Template Reference

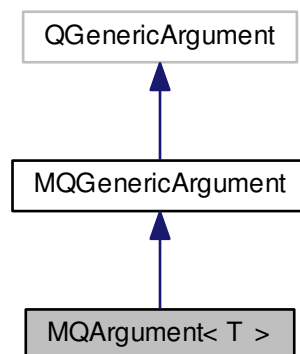
Similar to QArgument but with added argument label (parameter name)

```
#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)

14.67.1 Detailed Description

```
template<class T>class MQArgument< T >
```

Similar to QArgument but with added argument label (parameter name)

Definition at line 75 of file ArnRpc.hpp.

14.67.2 Constructor & Destructor Documentation

14.67.2.1 `template<class T> MQArgument< T >::MQArgument (const char * aName, const char * aLabel, const T & aData) [inline]`

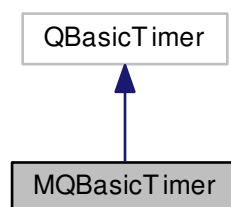
Definition at line 78 of file ArnRpc.hpp.

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

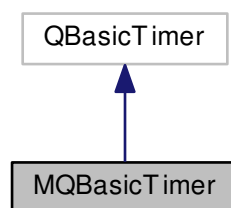
- [src/ArnInc/ArnRpc.hpp \(3.0.0\)](#)

14.68 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)

14.68.1 Detailed Description

Definition at line 61 of file ArnItem.cpp.

14.68.2 Constructor & Destructor Documentation

14.68.2.1 MQBasicTimer::MQBasicTimer () [inline]

Definition at line 64 of file ArnItem.cpp.

14.68.3 Member Function Documentation

14.68.3.1 int MQBasicTimer::interval () const [inline]

Definition at line 69 of file ArnItem.cpp.

14.68.3.2 void MQBasicTimer::setInterval (int *interval*) [inline]

Definition at line 70 of file ArnItem.cpp.

14.68.3.3 void MQBasicTimer::start (QObject * *obj*) [inline]

Definition at line 71 of file ArnItem.cpp.

14.68.3.4 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:

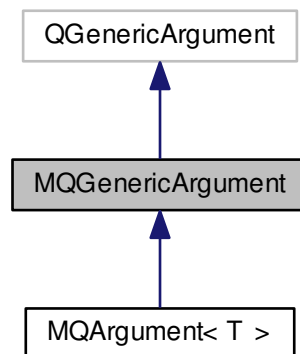
- [src/ArnItem.cpp \(3.0.0\)](#)

14.69 MQGenericArgument Class Reference

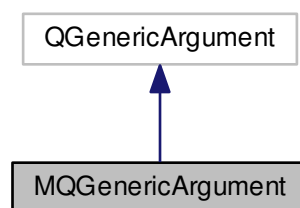
Similar to QGenericArgument but with added argument label (parameter name)

```
#include <ArnRpc.hpp>
```

Inheritance diagram for MQGenericArgument:



Collaboration diagram for MQGenericArgument:



Public Member Functions

- [MQGenericArgument](#) (const char *aName=0, const char *aLabel=0, const void *aData=0)
- [MQGenericArgument](#) (const QGenericArgument &qgenArg)
- const char * [label](#) () const

14.69.1 Detailed Description

Similar to QGenericArgument but with added argument label (parameter name)

Definition at line 59 of file ArnRpc.hpp.

14.69.2 Constructor & Destructor Documentation

14.69.2.1 `MQGenericArgument::MQGenericArgument (const char * aName = 0, const char * aLabel = 0, const void * aData = 0) [inline]`

Definition at line 62 of file ArnRpc.hpp.

14.69.2.2 `MQGenericArgument::MQGenericArgument (const QGenericArgument & qgenArg) [inline]`

Definition at line 64 of file ArnRpc.hpp.

14.69.3 Member Function Documentation

14.69.3.1 `const char* MQGenericArgument::label () const [inline]`

Definition at line 66 of file ArnRpc.hpp.

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

- [src/ArnInc/ArnRpc.hpp \(3.0.0\)](#)

14.70 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.

14.70.1 Detailed Description

Definition at line 166 of file Arn.hpp.

14.70.2 Member Enumeration Documentation

14.70.2.1 `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 168 of file Arn.hpp.

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

- [src/ArnInc/Arn.hpp \(3.0.0\)](#)

14.71 Arn::ObjectMode Class Reference

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

Public Types

- enum **E** { **Normal** = 0x00, **BiDir** = 0x01, **Pipe** = 0x02, **Save** = 0x04 }

14.71.1 Detailed Description

General global mode of an *Arn Data Object* Max 8 bit

Definition at line 118 of file Arn.hpp.

14.71.2 Member Enumeration Documentation

14.71.2.1 enum Arn::ObjectMode::E

Enumerator

Normal default

BiDir A two way object, typically for validation or pipe.

Pipe Implies *BiDir* 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 (3.0.0)

14.72 Arn::ObjectSyncMode Class Reference

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

Public Types

- enum **E** { **Normal** = 0x00, **Monitor** = 0x01, **Master** = 0x02, **AutoDestroy** = 0x04 }

14.72.1 Detailed Description

The client session sync mode of an *Arn Data Object* Max 8 bit

Definition at line 137 of file Arn.hpp.

14.72.2 Member Enumeration Documentation

14.72.2.1 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 [Arn Data Object](#) 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 \(3.0.0\)](#)

14.73 ArnRpc::MethodsParam::Params Struct Reference

```
#include <ArnRpc.hpp>
```

Public Attributes

- `QList< QByteArray >` [paramNames](#)
- `QList< QList< int > >` [methodIdsTab](#)
- `QList< int >` [allMethodIds](#)

14.73.1 Detailed Description

Definition at line 468 of file ArnRpc.hpp.

14.73.2 Member Data Documentation

14.73.2.1 `QList<int>` ArnRpc::MethodsParam::Params::allMethodIds

Definition at line 471 of file ArnRpc.hpp.

14.73.2.2 `QList< QList<int> >` ArnRpc::MethodsParam::Params::methodIdsTab

Definition at line 470 of file ArnRpc.hpp.

14.73.2.3 `QList<QByteArray>` ArnRpc::MethodsParam::Params::paramNames

Definition at line 469 of file ArnRpc.hpp.

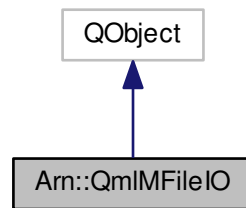
The documentation for this struct was generated from the following file:

- src/ArnInc/[ArnRpc.hpp \(3.0.0\)](#)

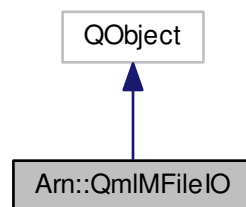
14.74 Arn::QmlMFileIO Class Reference

```
#include <ArnQmlMSystem.hpp>
```


Inheritance diagram for Arn::QmlMFileIO:



Collaboration diagram for Arn::QmlMFileIO:



Public Slots

- void [setPath](#) (const QString &[path](#))

Signals

- void [pathChanged](#) (const QString &[path](#))
- void [error](#) (const QString &msg)

Public Member Functions

- [QmlMFileIO](#) (QObject *parent=0)
- 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](#)

14.74.1 Detailed Description

Definition at line 40 of file ArnQmlMSystem.hpp.

14.74.2 Constructor & Destructor Documentation

14.74.2.1 `Arn::QmlMFileIO::QmlMFileIO (QObject * parent = 0)` `[explicit]`

Definition at line 41 of file ArnQmlMSystem.cpp.

14.74.3 Member Function Documentation

14.74.3.1 `void Arn::QmlMFileIO::error (const QString & msg)` `[signal]`

14.74.3.2 `QString Arn::QmlMFileIO::path ()`

14.74.3.3 `void Arn::QmlMFileIO::pathChanged (const QString & path)` `[signal]`

14.74.3.4 `QString Arn::QmlMFileIO::read ()`

Definition at line 47 of file ArnQmlMSystem.cpp.

14.74.3.5 `QByteArray Arn::QmlMFileIO::readBytes ()`

Definition at line 95 of file ArnQmlMSystem.cpp.

14.74.3.6 `void Arn::QmlMFileIO::setPath (const QString & path)` `[slot]`

Definition at line 141 of file ArnQmlMSystem.cpp.

14.74.3.7 `bool Arn::QmlMFileIO::write (const QString & data)`

Definition at line 77 of file ArnQmlMSystem.cpp.

14.74.3.8 `bool Arn::QmlMFileIO::writeBytes (const QByteArray & data)`

Definition at line 118 of file ArnQmlMSystem.cpp.

14.74.4 Property Documentation

14.74.4.1 `QString Arn::QmlMFileIO::path` `[read],[write]`

Definition at line 45 of file ArnQmlMSystem.hpp.

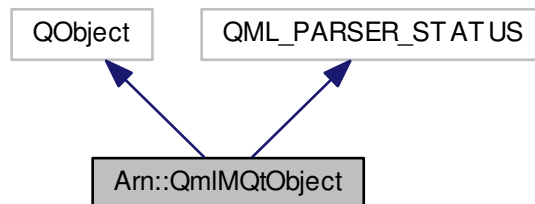
The documentation for this class was generated from the following files:

- [src/ArnInc/ArnQmlMSystem.hpp \(3.0.0\)](#)
- [src/ArnQmlMSystem.cpp \(3.0.0\)](#)

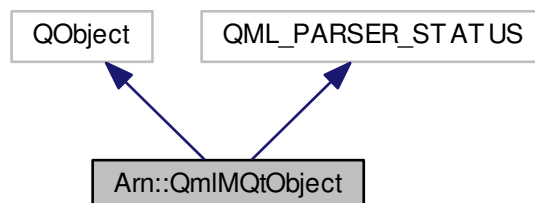
14.75 Arn::QmlMQtObject Class Reference

```
#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=0`)
- 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 int [data_count](#) ([QML_LIST_PROPERTY](#)< [QObject](#) > *prop)
- static [QObject](#) * [data_at](#) ([QML_LIST_PROPERTY](#)< [QObject](#) > *prop, int index)
- static void [data_clear](#) ([QML_LIST_PROPERTY](#)< [QObject](#) > *prop)

14.75.1 Detailed Description

Definition at line 51 of file ArnQmlMQt.hpp.

14.75.2 Constructor & Destructor Documentation

14.75.2.1 `Arn::QmlMQtObject::QmlMQtObject (QmlMQtObject * parent = 0)`

Definition at line 47 of file ArnQmlMQt.cpp.

14.75.2.2 `Arn::QmlMQtObject::~~QmlMQtObject ()` [virtual]

Definition at line 53 of file ArnQmlMQt.cpp.

14.75.3 Member Function Documentation

14.75.3.1 `void Arn::QmlMQtObject::classBegin ()` [virtual]

Definition at line 113 of file ArnQmlMQt.cpp.

14.75.3.2 `void Arn::QmlMQtObject::completed ()` [signal]

14.75.3.3 `void Arn::QmlMQtObject::componentComplete ()` [virtual]

Definition at line 118 of file ArnQmlMQt.cpp.

14.75.3.4 `QML_LIST_PROPERTY< QObject > Arn::QmlMQtObject::data ()`

14.75.3.5 `void Arn::QmlMQtObject::data_append (QML_LIST_PROPERTY< QObject > * prop, QObject * obj)`
[static]

Definition at line 77 of file ArnQmlMQt.cpp.

14.75.3.6 `QObject * Arn::QmlMQtObject::data_at (QML_LIST_PROPERTY< QObject > * prop, int index)`
[static]

Definition at line 93 of file ArnQmlMQt.cpp.

14.75.3.7 `void Arn::QmlMQtObject::data_clear (QML_LIST_PROPERTY< QObject > * prop)` [static]

Definition at line 103 of file ArnQmlMQt.cpp.

14.75.3.8 int Arn::QmlMQtObject::data_count (QML_LIST_PROPERTY< QObject > * *prop*) [static]

Definition at line 86 of file ArnQmlMQt.cpp.

14.75.3.9 void Arn::QmlMQtObject::parentChanged (QmlMQtObject * *obj*) [signal]

14.75.3.10 QmlMQtObject * Arn::QmlMQtObject::parentItem () const

Definition at line 58 of file ArnQmlMQt.cpp.

14.75.3.11 void Arn::QmlMQtObject::setParentItem (QmlMQtObject * *parent*)

Definition at line 64 of file ArnQmlMQt.cpp.

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

- src/ArnInc/ArnQmlMQt.hpp (3.0.0)
- src/ArnQmlMQt.cpp (3.0.0)

14.76 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 }

14.76.1 Detailed Description

Action when assigning same value to an [ArnItem](#).

Definition at line 61 of file Arn.hpp.

14.76.2 Member Enumeration Documentation

14.76.2.1 enum Arn::SameValue::E

Enumerator

Accept Assigning same value generates an update of the [Arn Data Object](#)

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 (3.0.0)

14.77 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](#) }

14.77.1 Detailed Description

State of [Arn](#) discover browse data. Can be tested by relative order.

Definition at line 79 of file [ArnDiscover.hpp](#).

14.77.2 Member Enumeration Documentation

14.77.2.1 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](#) (3.0.0)

14.78 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 }

14.78.1 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.

14.78.2 Member Enumeration Documentation

14.78.2.1 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 \(3.0.0\)](#)

14.79 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 }

14.79.1 Detailed Description

States of DiscoverAdvertise / These values must be synced with: [ArnZeroConf::State](#).

Definition at line 636 of file ArnDiscover.hpp.

14.79.2 Member Enumeration Documentation

14.79.2.1 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 637 of file ArnDiscover.hpp.

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

- src/ArnInc/[ArnDiscover.hpp \(3.0.0\)](#)

14.80 ArnError::StdCode Struct Reference

```
#include <ArnError.hpp>
```

Public Types

- enum E {
 Ok = 0, **Info** = 1, **Warning** = 2, **Err_Undef** = 15,
 Err_Custom = 16 }

14.80.1 Detailed Description

Definition at line 72 of file ArnError.hpp.

14.80.2 Member Enumeration Documentation

14.80.2.1 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 \(3.0.0\)](#)

14.81 ArnItemValve::SwitchMode Struct Reference

```
#include <ArnItemValve.hpp>
```


Public Types

- enum [E](#) { [InStream](#) = 0x01, [OutStream](#) = 0x02, [InOutStream](#) = InStream | OutStream }

14.81.1 Detailed Description

Definition at line 83 of file ArnItemValve.hpp.

14.81.2 Member Enumeration Documentation

14.81.2.1 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 *InStream* and *OutStream*

Definition at line 84 of file ArnItemValve.hpp.

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

- src/ArnInc/[ArnItemValve.hpp](#) (3.0.0)

14.82 ArnScriptJobs::Type Struct Reference

```
#include <ArnScriptJobs.hpp>
```

Public Types

- enum [E](#) { [Null](#), [Cooperative](#), [Preemptive](#) }

14.82.1 Detailed Description

Definition at line 92 of file ArnScriptJobs.hpp.

14.82.2 Member Enumeration Documentation

14.82.2.1 enum ArnScriptJobs::Type::E

Enumerator

Null

Cooperative

Preemptive

Definition at line 93 of file ArnScriptJobs.hpp.

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

- src/ArnInc/[ArnScriptJobs.hpp](#) (3.0.0)

14.83 ArnDiscover::Type Struct Reference

Types of [Arn](#) discover advertise.

```
#include <ArnDiscover.hpp>
```

Public Types

- enum [E](#) { [None](#), [Server](#), [Client](#) }

14.83.1 Detailed Description

Types of [Arn](#) discover advertise.

Definition at line 52 of file ArnDiscover.hpp.

14.83.2 Member Enumeration Documentation

14.83.2.1 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](#) (3.0.0)

14.84 ArnServer::Type Struct Reference

```
#include <ArnServer.hpp>
```

Public Types

- enum [E](#) { [NetSync](#) }

14.84.1 Detailed Description

Definition at line 104 of file ArnServer.hpp.

14.84.2 Member Enumeration Documentation

14.84.2.1 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 \(3.0.0\)](#)

14.85 ArnQml::UseFlags Struct Reference

```
#include <ArnQml.hpp>
```

Public Types

- enum [E](#) { [ArnLib](#) = 0x01, [MSystem](#) = 0x02, [MQt](#) = 0x04, [All](#) = 0xff }

14.85.1 Detailed Description

Definition at line 184 of file ArnQml.hpp.

14.85.2 Member Enumeration Documentation

14.85.2.1 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 185 of file ArnQml.hpp.

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

- [src/ArnInc/ArnQml.hpp \(3.0.0\)](#)

14.86 Arn::XStringMap Class Reference

Container class with string representation for serialized data.

```
#include <XStringMap.hpp>
```

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 ()`
- `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 (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)`
- `const QByteArray & keyRef (int i) const`
- `QByteArray key (int i, const char *def=0) const`
- `QByteArray key (const QByteArray &value, const char *def=0) const`
- `QByteArray key (const QString &value, const char *def=0) 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=0) const`
- `QByteArray value (const char *key, const char *def=0) const`
- `QByteArray value (const char *keyPrefix, uint eNum, const char *def=0) const`
- `QByteArray value (const QByteArray &key, const char *def=0) 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](#))
- QByteArray [toXString](#) () const
- bool [fromXString](#) (const QByteArray &inXString, int [size](#)==1)
- void [setEmptyKeysToValue](#) ()
- QStringList [keys](#) () const
- QStringList [values](#) (const char *keyPrefix=0) const
- QVariantMap [toVariantMap](#) () 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](#) ()

Static Public Member Functions

- static void [stringCode](#) (QByteArray &dst, const QByteArray &src)
- static void [stringDecode](#) (QByteArray &dst, const QByteArray &src)

14.86.1 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 guarantied 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) -> "_", "_" -> "\\_", "^" -> "\\^", "\" -> "\\\""
```

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.add("val", QByteArray::number(12));
qDebug() << "XString: " << xsm.toXString();
```

This will print "XString: put id=level val=12"

Definition at line 83 of file XStringMap.hpp.

14.86.2 Constructor & Destructor Documentation

14.86.2.1 Arn::XStringMap::XStringMap ()

Definition at line 53 of file ArnXStringMap.cpp.

14.86.2.2 Arn::XStringMap::XStringMap (const XStringMap & *other*)

Make shallow copy (Qt style)

Definition at line 59 of file ArnXStringMap.cpp.

14.86.2.3 Arn::XStringMap::XStringMap (const QByteArray & *xString*) [explicit]

Definition at line 67 of file ArnXStringMap.cpp.

14.86.2.4 Arn::XStringMap::XStringMap (const QVariantMap & *variantMap*) [explicit]

Definition at line 74 of file ArnXStringMap.cpp.

14.86.2.5 Arn::XStringMap::~XStringMap ()

Definition at line 81 of file ArnXStringMap.cpp.

14.86.3 Member Function Documentation

14.86.3.1 XStringMap & Arn::XStringMap::add (const char * *key*, const QByteArray & *val*)

Definition at line 185 of file ArnXStringMap.cpp.

14.86.3.2 XStringMap & Arn::XStringMap::add (const char * *key*, const char * *val*)

Definition at line 200 of file ArnXStringMap.cpp.

14.86.3.3 XStringMap & Arn::XStringMap::add (const char * *keyPrefix*, uint *eNum*, const QByteArray & *val*)

Definition at line 206 of file ArnXStringMap.cpp.

14.86.3.4 XStringMap & Arn::XStringMap::add (const QByteArray & *key*, const QByteArray & *val*)

Definition at line 216 of file ArnXStringMap.cpp.

14.86.3.5 XStringMap & Arn::XStringMap::add (const char * *key*, const QString & *val*)

Definition at line 222 of file ArnXStringMap.cpp.

14.86.3.6 XStringMap & Arn::XStringMap::add (const char * *keyPrefix*, uint *eNum*, const QString & *val*)

Definition at line 228 of file ArnXStringMap.cpp.

14.86.3.7 XStringMap & Arn::XStringMap::add (const QByteArray & key, const QString & val)

Definition at line 234 of file ArnXStringMap.cpp.

14.86.3.8 XStringMap & Arn::XStringMap::add (const QString & key, const QString & val)

Definition at line 240 of file ArnXStringMap.cpp.

14.86.3.9 XStringMap & Arn::XStringMap::add (const XStringMap & other)

Definition at line 246 of file ArnXStringMap.cpp.

14.86.3.10 XStringMap & Arn::XStringMap::add (const QVariantMap & variantMap)

Definition at line 256 of file ArnXStringMap.cpp.

14.86.3.11 XStringMap & Arn::XStringMap::addNum (const char * key, int val)

Definition at line 272 of file ArnXStringMap.cpp.

14.86.3.12 XStringMap & Arn::XStringMap::addNum (const QByteArray & key, int val)

Definition at line 278 of file ArnXStringMap.cpp.

14.86.3.13 XStringMap & Arn::XStringMap::addNum (const QString & key, int val)

Definition at line 284 of file ArnXStringMap.cpp.

14.86.3.14 XStringMap & Arn::XStringMap::addNum (const char * key, uint val)

Definition at line 290 of file ArnXStringMap.cpp.

14.86.3.15 XStringMap & Arn::XStringMap::addNum (const QByteArray & key, uint val)

Definition at line 296 of file ArnXStringMap.cpp.

14.86.3.16 XStringMap & Arn::XStringMap::addNum (const QString & key, uint val)

Definition at line 302 of file ArnXStringMap.cpp.

14.86.3.17 XStringMap & Arn::XStringMap::addNum (const char * key, double val, int precision = -1)

Definition at line 308 of file ArnXStringMap.cpp.

14.86.3.18 XStringMap & Arn::XStringMap::addNum (const QByteArray & key, double val, int precision = -1)

Definition at line 315 of file ArnXStringMap.cpp.

14.86.3.19 **XStringMap** & Arn::XStringMap::addNum (const QString & *key*, double *val*, int *precision* = -1)

Definition at line 322 of file ArnXStringMap.cpp.

14.86.3.20 **XStringMap** & Arn::XStringMap::addValues (const QStringList & *stringList*)

Definition at line 329 of file ArnXStringMap.cpp.

14.86.3.21 void Arn::XStringMap::append (const char * *key*, const QByteArray & *val*) [inline]

Definition at line 172 of file XStringMap.hpp.

14.86.3.22 void Arn::XStringMap::append (const char * *key*, const char * *val*) [inline]

Definition at line 174 of file XStringMap.hpp.

14.86.3.23 void Arn::XStringMap::append (const char * *keyPrefix*, uint *eNum*, const QByteArray & *val*) [inline]

Definition at line 176 of file XStringMap.hpp.

14.86.3.24 void Arn::XStringMap::append (const QByteArray & *key*, const QByteArray & *val*) [inline]

Definition at line 178 of file XStringMap.hpp.

14.86.3.25 void Arn::XStringMap::append (const char * *key*, const QString & *val*) [inline]

Definition at line 180 of file XStringMap.hpp.

14.86.3.26 void Arn::XStringMap::append (const char * *keyPrefix*, uint *eNum*, const QString & *val*) [inline]

Definition at line 182 of file XStringMap.hpp.

14.86.3.27 void Arn::XStringMap::append (const QByteArray & *key*, const QString & *val*) [inline]

Definition at line 184 of file XStringMap.hpp.

14.86.3.28 void Arn::XStringMap::append (const QString & *key*, const QString & *val*) [inline]

Definition at line 186 of file XStringMap.hpp.

14.86.3.29 void Arn::XStringMap::append (const XStringMap & *other*) [inline]

Definition at line 188 of file XStringMap.hpp.

14.86.3.30 void Arn::XStringMap::append (const QVariantMap & *other*) [inline]

Definition at line 190 of file XStringMap.hpp.

14.86.3.31 void Arn::XStringMap::clear (bool *freeMem* = false)

Definition at line 102 of file ArnXStringMap.cpp.

14.86.3.32 bool Arn::XStringMap::fromXString (const QByteArray & *inXString*, int *size* = -1)

Definition at line 643 of file ArnXStringMap.cpp.

14.86.3.33 int Arn::XStringMap::indexOf (const char * *key*, int *from* = 0) const

Definition at line 122 of file ArnXStringMap.cpp.

14.86.3.34 int Arn::XStringMap::indexOf (const QByteArray & *key*, int *from* = 0) const

Definition at line 135 of file ArnXStringMap.cpp.

14.86.3.35 int Arn::XStringMap::indexOf (const QString & *key*, int *from* = 0) const

Definition at line 146 of file ArnXStringMap.cpp.

14.86.3.36 int Arn::XStringMap::indexOfValue (const QByteArray & *value*, int *from* = 0) const

Definition at line 152 of file ArnXStringMap.cpp.

14.86.3.37 int Arn::XStringMap::indexOfValue (const QString & *value*, int *from* = 0) const

Definition at line 163 of file ArnXStringMap.cpp.

14.86.3.38 QByteArray Arn::XStringMap::info ()

Definition at line 808 of file ArnXStringMap.cpp.

14.86.3.39 QByteArray Arn::XStringMap::key (int *i*, const char * *def* = 0) const

Definition at line 400 of file ArnXStringMap.cpp.

14.86.3.40 QByteArray Arn::XStringMap::key (const QByteArray & *value*, const char * *def* = 0) const

Definition at line 408 of file ArnXStringMap.cpp.

14.86.3.41 QByteArray Arn::XStringMap::key (const QString & *value*, const char * *def* = 0) const

Definition at line 417 of file ArnXStringMap.cpp.

14.86.3.42 const QByteArray & Arn::XStringMap::keyRef (int *i*) const

Definition at line 392 of file ArnXStringMap.cpp.

14.86.3.43 `QStringList Arn::XStringMap::keys () const`

Definition at line 583 of file ArnXStringMap.cpp.

14.86.3.44 `QString Arn::XStringMap::keyString (int i, const QString & def = QString()) const`

Definition at line 423 of file ArnXStringMap.cpp.

14.86.3.45 `QString Arn::XStringMap::keyString (const QString & value, const QString & def = QString()) const`

Definition at line 432 of file ArnXStringMap.cpp.

14.86.3.46 `int Arn::XStringMap::maxEnumOf (const char * keyPrefix) const`

Definition at line 169 of file ArnXStringMap.cpp.

14.86.3.47 `XStringMap & Arn::XStringMap::operator+= (const XStringMap & other)`

Definition at line 802 of file ArnXStringMap.cpp.

14.86.3.48 `XStringMap & Arn::XStringMap::operator+= (const QVariantMap & other)`

Definition at line 796 of file ArnXStringMap.cpp.

14.86.3.49 `XStringMap & Arn::XStringMap::operator= (const XStringMap & other)`

Make shallow copy (Qt style)

Definition at line 86 of file ArnXStringMap.cpp.

14.86.3.50 `XStringMap & Arn::XStringMap::remove (int index)`

Definition at line 538 of file ArnXStringMap.cpp.

14.86.3.51 `XStringMap & Arn::XStringMap::remove (const char * key)`

Definition at line 554 of file ArnXStringMap.cpp.

14.86.3.52 `XStringMap & Arn::XStringMap::remove (const QByteArray & key)`

Definition at line 560 of file ArnXStringMap.cpp.

14.86.3.53 `XStringMap & Arn::XStringMap::remove (const QString & key)`

Definition at line 566 of file ArnXStringMap.cpp.

14.86.3.54 `XStringMap & Arn::XStringMap::set (int i, const QByteArray & val)`

Definition at line 339 of file ArnXStringMap.cpp.

14.86.3.55 `XStringMap & Arn::XStringMap::set (const char * key, const QByteArray & val)`

Definition at line 350 of file ArnXStringMap.cpp.

14.86.3.56 `XStringMap & Arn::XStringMap::set (const char * key, const char * val)`

Definition at line 362 of file ArnXStringMap.cpp.

14.86.3.57 `XStringMap & Arn::XStringMap::set (const QByteArray & key, const QByteArray & val)`

Definition at line 368 of file ArnXStringMap.cpp.

14.86.3.58 `XStringMap & Arn::XStringMap::set (const char * key, const QString & val)`

Definition at line 374 of file ArnXStringMap.cpp.

14.86.3.59 `XStringMap & Arn::XStringMap::set (const QByteArray & key, const QString & val)`

Definition at line 380 of file ArnXStringMap.cpp.

14.86.3.60 `XStringMap & Arn::XStringMap::set (const QString & key, const QString & val)`

Definition at line 386 of file ArnXStringMap.cpp.

14.86.3.61 `void Arn::XStringMap::setEmptyKeysToValue ()`

Definition at line 572 of file ArnXStringMap.cpp.

14.86.3.62 `int Arn::XStringMap::size () const [inline]`

Definition at line 95 of file XStringMap.hpp.

14.86.3.63 `void Arn::XStringMap::squeeze ()`

Definition at line 113 of file ArnXStringMap.cpp.

14.86.3.64 `void Arn::XStringMap::stringCode (QByteArray & dst, const QByteArray & src) [static]`

Definition at line 688 of file ArnXStringMap.cpp.

14.86.3.65 `void Arn::XStringMap::stringDecode (QByteArray & dst, const QByteArray & src) [static]`

Definition at line 742 of file ArnXStringMap.cpp.

14.86.3.66 `QVariantMap Arn::XStringMap::toVariantMap () const`

Definition at line 609 of file ArnXStringMap.cpp.

14.86.3.67 `QByteArray Arn::XStringMap::toString () const`

Definition at line 623 of file ArnXStringMap.cpp.

14.86.3.68 `QByteArray Arn::XStringMap::value (int i, const char * def = 0) const`

Definition at line 447 of file ArnXStringMap.cpp.

14.86.3.69 `QByteArray Arn::XStringMap::value (const char * key, const char * def = 0) const`

Definition at line 455 of file ArnXStringMap.cpp.

14.86.3.70 `QByteArray Arn::XStringMap::value (const char * keyPrefix, uint eNum, const char * def = 0) const`

Definition at line 464 of file ArnXStringMap.cpp.

14.86.3.71 `QByteArray Arn::XStringMap::value (const QByteArray & key, const char * def = 0) const`

Definition at line 477 of file ArnXStringMap.cpp.

14.86.3.72 `QByteArray Arn::XStringMap::value (const QByteArray & key, const QByteArray & def) const`

Definition at line 486 of file ArnXStringMap.cpp.

14.86.3.73 `const QByteArray & Arn::XStringMap::valueRef (int i) const`

Definition at line 439 of file ArnXStringMap.cpp.

14.86.3.74 `QStringList Arn::XStringMap::values (const char * keyPrefix = 0) const`

Definition at line 594 of file ArnXStringMap.cpp.

14.86.3.75 `QString Arn::XStringMap::valueString (int i, const QString & def = QString()) const`

Definition at line 496 of file ArnXStringMap.cpp.

14.86.3.76 `QString Arn::XStringMap::valueString (const char * key, const QString & def = QString()) const`

Definition at line 505 of file ArnXStringMap.cpp.

14.86.3.77 `QString Arn::XStringMap::valueString (const char * keyPrefix, uint eNum, const QString & def = QString()) const`

Definition at line 512 of file ArnXStringMap.cpp.

14.86.3.78 `QString Arn::XStringMap::valueString (const QByteArray & key, const QString & def = QString()) const`

Definition at line 524 of file ArnXStringMap.cpp.

14.86.3.79 `QString Arn::XStringMap::valueString (const QString & key, const QString & def = QString()) const`

Definition at line 531 of file ArnXStringMap.cpp.

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

- [src/ArnInc/XStringMap.hpp \(3.0.0\)](#)
- [src/ArnXStringMap.cpp \(3.0.0\)](#)

Chapter 15

File Documentation

15.1 doc/Description.md File Reference

15.2 doc/HelpIndex.txt File Reference

15.3 doc/Install.md File Reference

15.4 doc/Internals.md File Reference

15.5 doc/ToDo.md File Reference

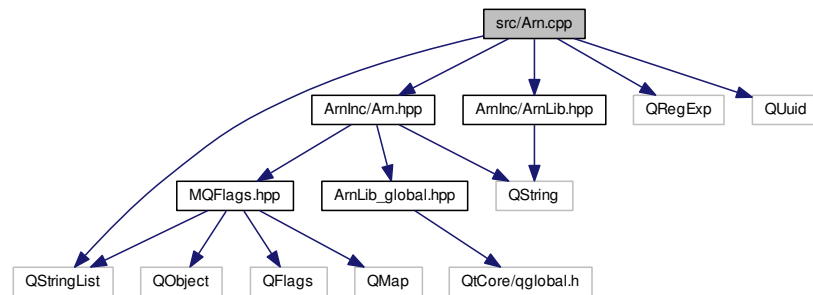
15.6 examples/Examples.txt File Reference

15.7 README.md File Reference

15.8 src/Arn.cpp File Reference

```
#include "ArnInc/Arn.hpp"  
#include "ArnInc/ArnLib.hpp"  
#include <QRegExp>  
#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.

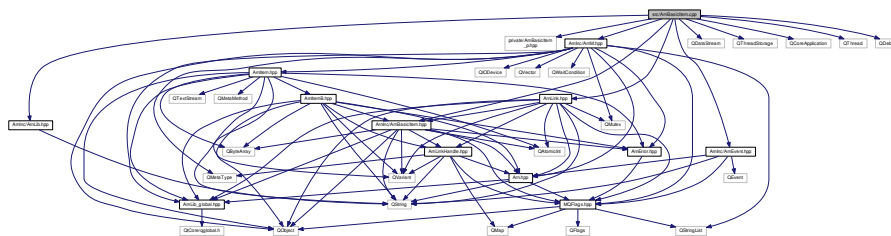
Variables

- 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/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:

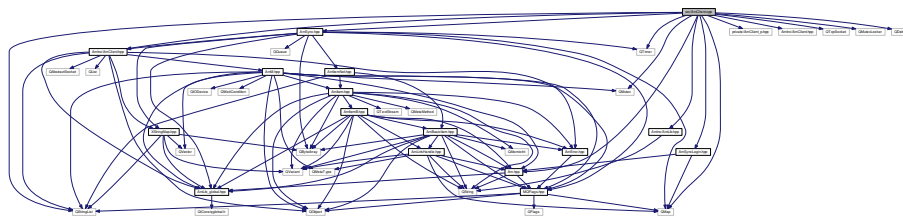


15.10 src/ArnClient.cpp File Reference

```
#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:



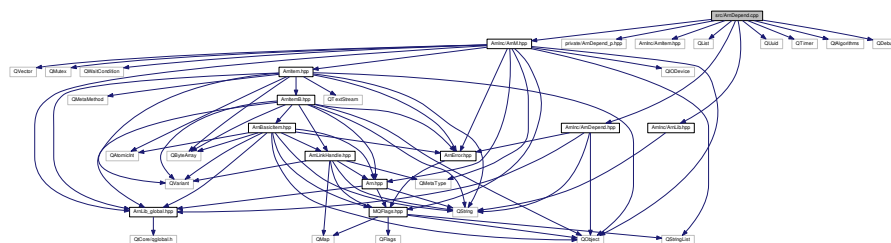
Classes

- class [ArnClientReg](#)

15.11 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/"

15.11.1 Variable Documentation

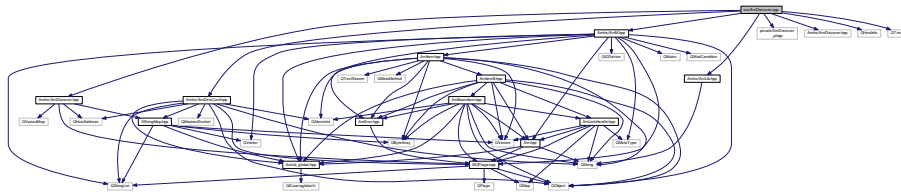
15.11.1.1 `const char* ArnDependPath = "//.sys/Depend/"`

Definition at line 41 of file ArnDepend.cpp.

15.12 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 <QTimer>
```

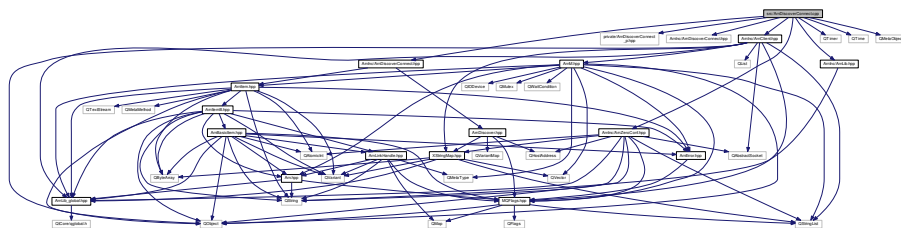
Include dependency graph for ArnDiscover.cpp:



15.13 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 <QTimer>
#include <QTime>
#include <QMetaObject>
```

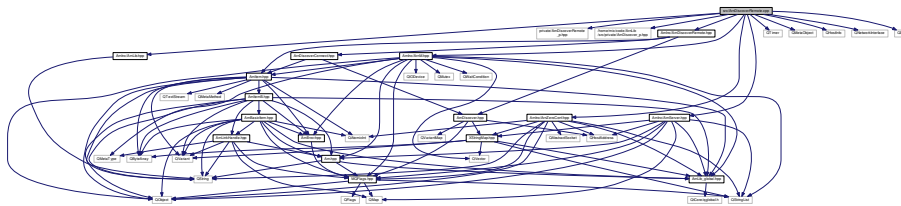
Include dependency graph for ArnDiscoverConnect.cpp:



15.14 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>
```

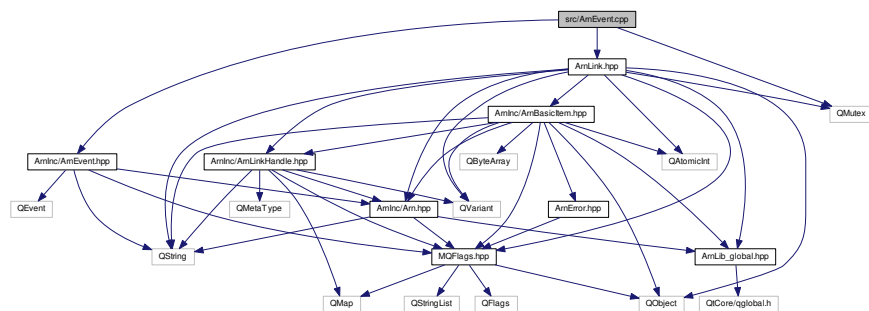
Include dependency graph for ArnDiscoverRemote.cpp:



15.15 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)`

15.15.1 Macro Definition Documentation

15.15.1.1 `#define TO_IDX_RETVAL(evType)`

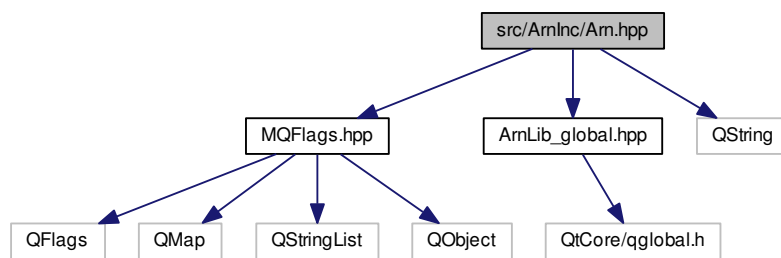
Value:

```
int retVal = (evType) - baseType(); \
retVal = ((retVal >= 0) && (retVal < Idx::N)) ? retVal : Idx::QtEvent;
```

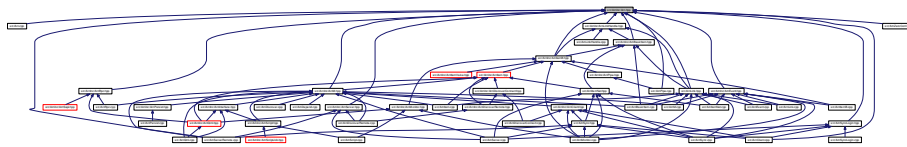
Definition at line 95 of file ArnEvent.cpp.

15.16 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::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.

Variables

- `const quint16 Arn::defaultTcpPort = 2022`

15.16.1 Macro Definition Documentation

15.16.1.1 `#define ARNREAL double`

Definition at line 44 of file Arn.hpp.

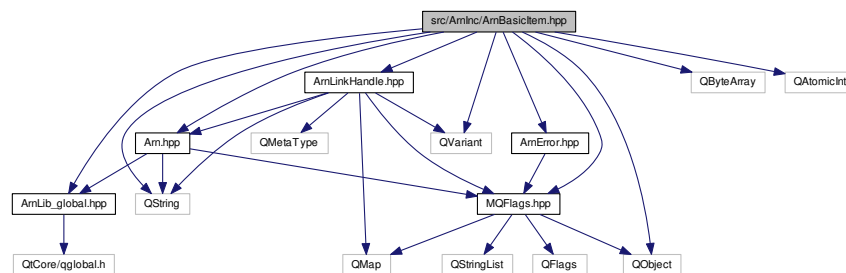
15.16.1.2 `#define DATASTREAM_VER QDataStream::Qt_4_6`

Definition at line 39 of file Arn.hpp.

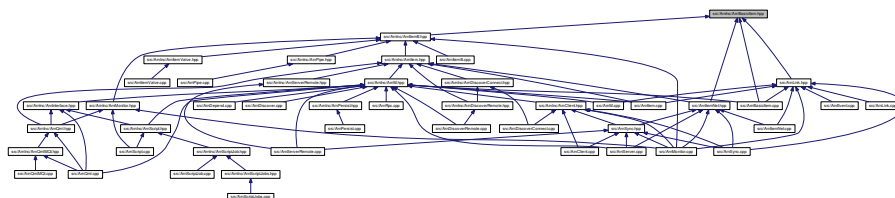
15.17 src/ArnInc/ArnBasicItem.hpp File Reference

```
#include "ArnLib_global.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

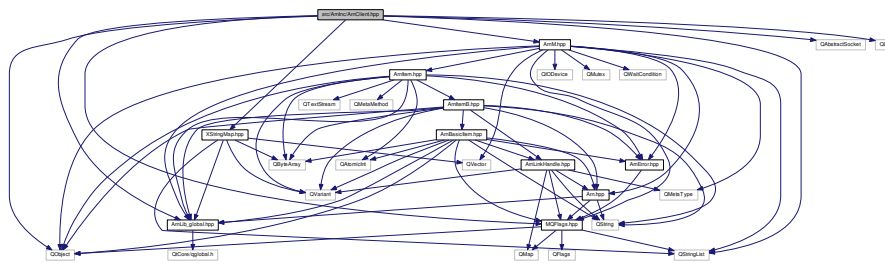
- class [ArnBasicItem](#)

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

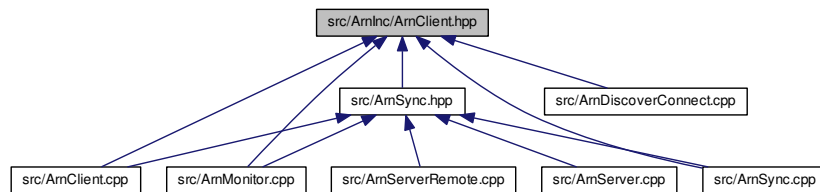
15.18 src/ArnInc/ArnClient.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnM.hpp"
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QObject>
#include <QAbstractSocket>
#include <QStringList>
#include <QList>
```

Include dependency graph for ArnClient.hpp:



This graph shows which files directly or indirectly include this file:



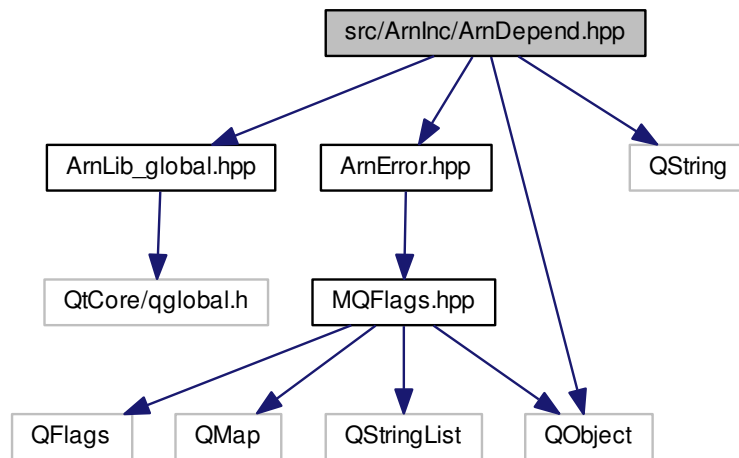
Classes

- class [ArnClientConnectStat](#)
- class [ArnClient](#)
 - Class for connecting to an [Arn](#) Server.
- struct [ArnClient::HostAddrPort](#)

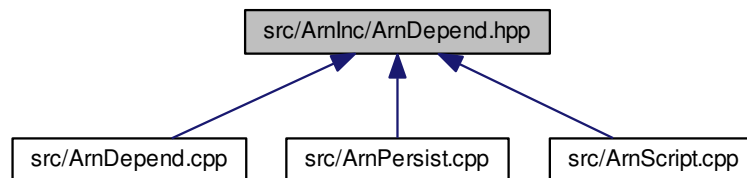
15.19 src/ArnInc/ArnDepend.hpp File Reference

```
#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.

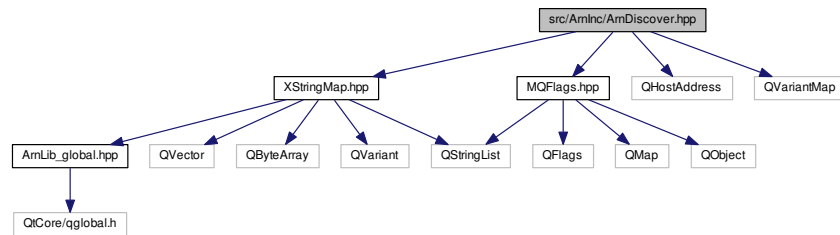
15.20 src/ArnInc/ArnDiscover.hpp File Reference

```

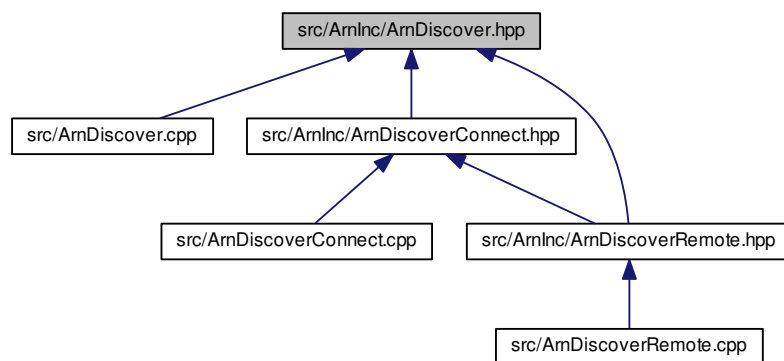
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QHostAddress>
#include <QVariantMap>

```

Include dependency graph for ArnDiscover.hpp:



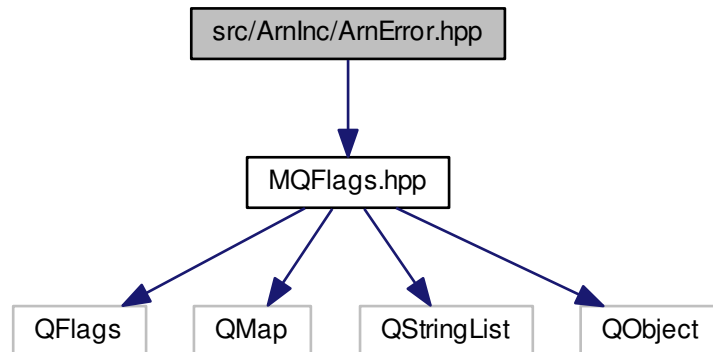
This graph shows which files directly or indirectly include this file:



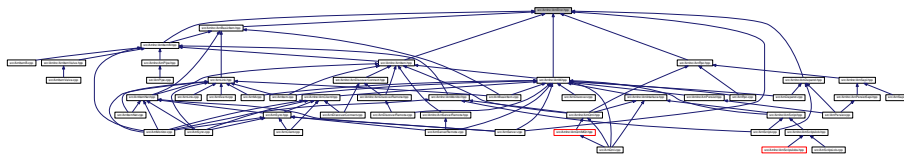
Classes

- 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](#)

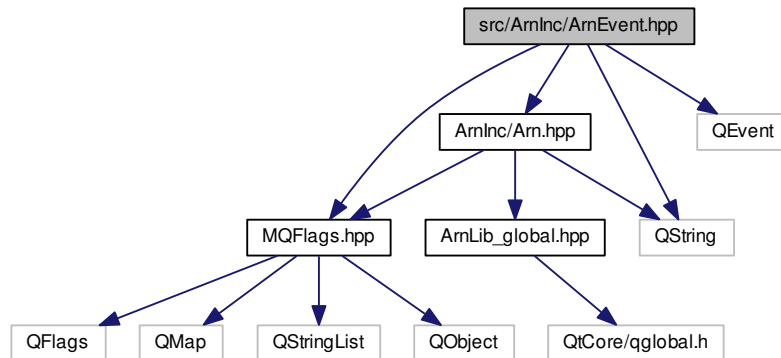
15.24 src/ArnInc/ArnEvent.hpp File Reference

```

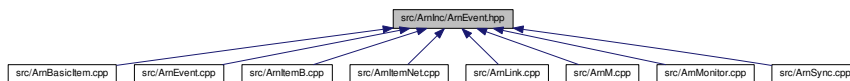
#include "ArnInc/Arn.hpp"
#include "ArnInc/MQFlags.hpp"
#include <QEvent>
#include <QString>

```

Include dependency graph for ArnEvent.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnEventIdx](#)
- class [ArnEvent](#)
- class [ArnEvLinkCreate](#)
- class [ArnEvModeChange](#)
- class [ArnEvMonitor](#)
- class [ArnEvRetired](#)
- class [ArnEvZeroRef](#)
- class [ArnEvValueChange](#)
- class [ArnEvRefChange](#)

15.25 src/ArnInc/ArnInterface.hpp File Reference

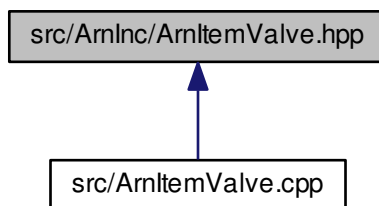
```
#include "ArnLib_global.hpp"
#include "ArnM.hpp"
```


The graph illustrates a complex network of relationships between various entities, including 'The American Revolution', 'The American Civil War', and 'The American West'. The nodes are interconnected by a dense web of blue lines, representing a complex network of relationships. The graph is a directed graph, with arrows indicating the direction of the relationships. The nodes are arranged in a hierarchical manner, with 'The American Revolution' at the top and 'The American West' at the bottom. The graph is a complex network of relationships, with many nodes and edges, representing a complex network of relationships.

- class **ArnlItemB**

15.28 src/ArnInc/ArnItemValve.hpp File Reference

This graph shows which files directly or indirectly include this file:



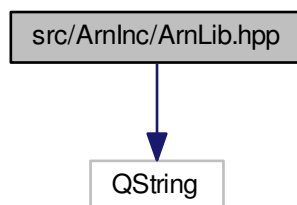
Classes

- class [ArnItemValve](#)
Valve for controlling stream to/from an [ArnItemB](#).
- struct [ArnItemValve::SwitchMode](#)

15.29 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:



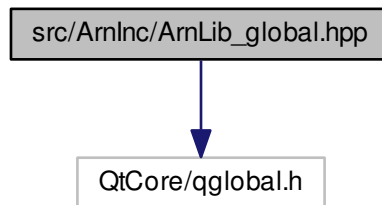
Namespaces

- [Arn](#)

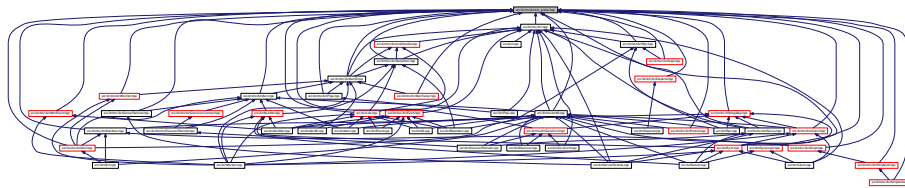
15.30 src/ArnInc/ArnLib_global.hpp File Reference

```
#include <QtCore/qglobal.h>
```

Include dependency graph for ArnLib_global.hpp:



This graph shows which files directly or indirectly include this file:



Macros

- `#define ARNLIBSHARED_EXPORT Q_DECL_IMPORT`

15.30.1 Macro Definition Documentation

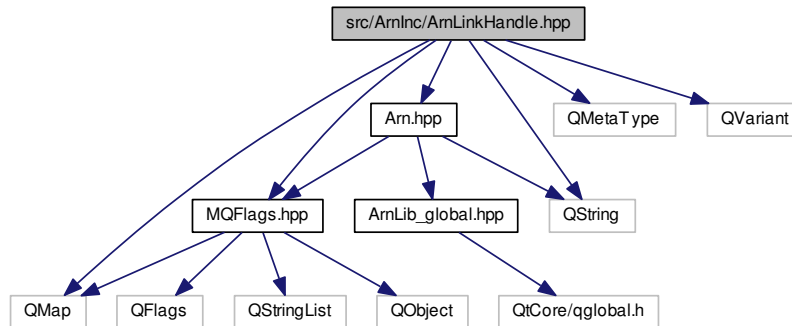
15.30.1.1 `#define ARNLIBSHARED_EXPORT Q_DECL_IMPORT`

Definition at line 11 of file ArnLib_global.hpp.

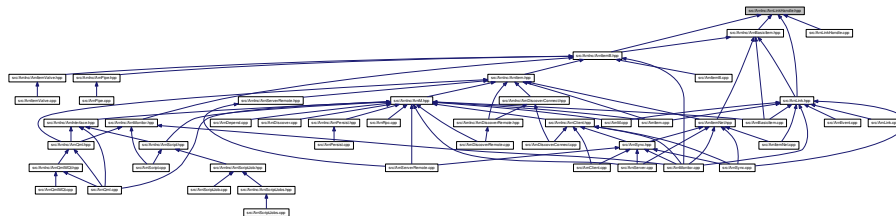
15.31 src/ArnInc/ArnLinkHandle.hpp File Reference

```
#include "Arn.hpp"
#include "MQFlags.hpp"
#include <QMetaType>
#include <QString>
#include <QVariant>
#include <QMap>
```

Include dependency graph for ArnLinkHandle.hpp:



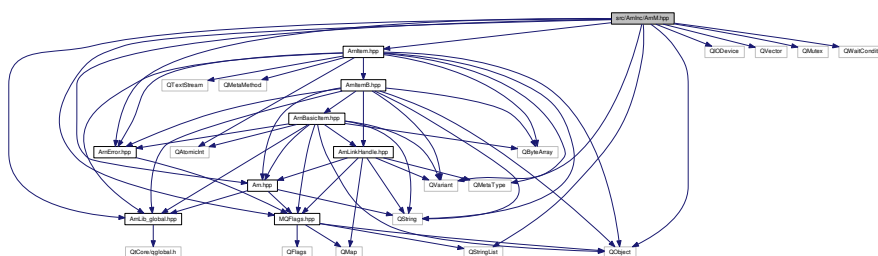
This graph shows which files directly or indirectly include this file:



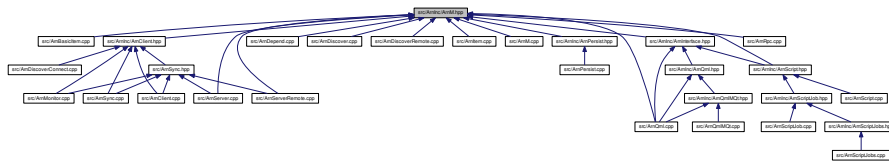
15.32 src/ArnInc/ArnM.hpp File Reference

```
#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>
#include <QWaitCondition>
```

Include dependency graph for ArnM.hpp:



This graph shows which files directly or indirectly include this file:

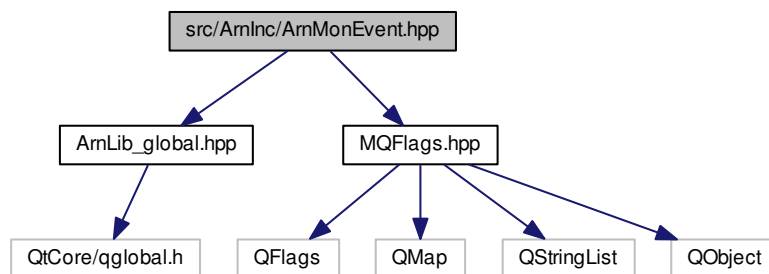


Classes

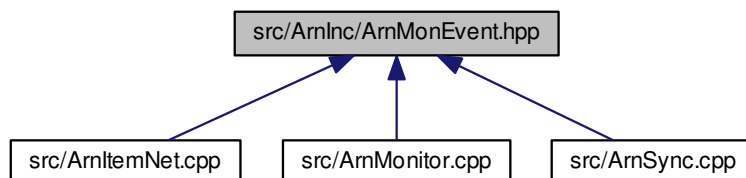
- class [ArnM](#)
Arn main class.

15.33 src/ArnInc/ArnMonEvent.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "MQFlags.hpp"
Include dependency graph for ArnMonEvent.hpp:
```



This graph shows which files directly or indirectly include this file:



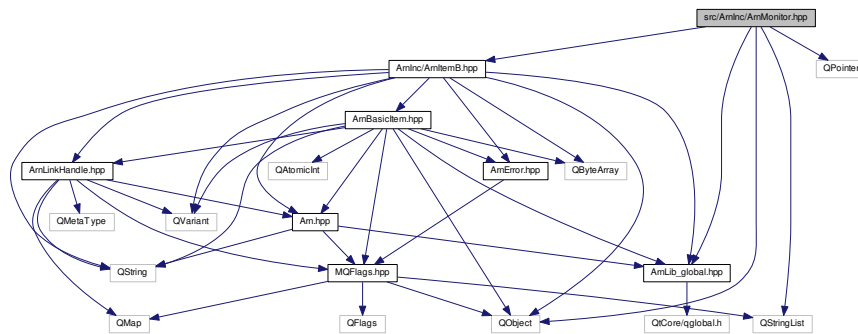
Classes

- class [ArnMonEventType](#)

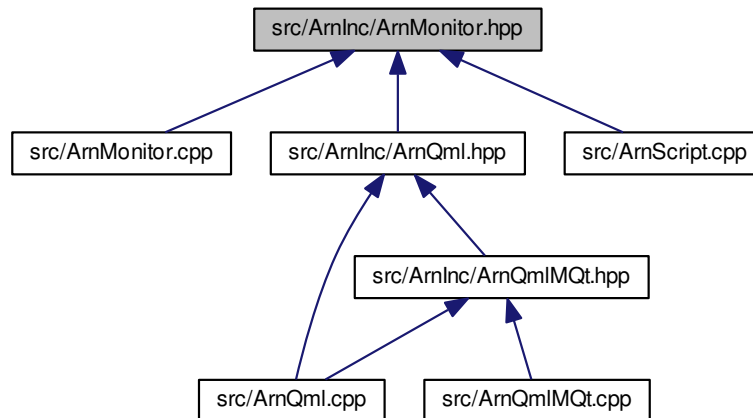
15.34 src/ArnInc/ArnMonitor.hpp File Reference

```
#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

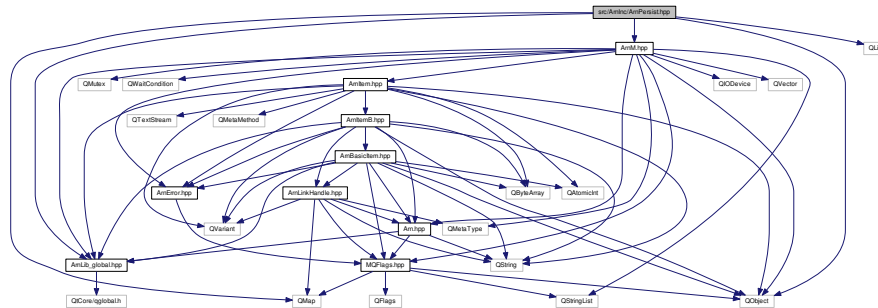
- class [ArnMonitor](#)

A client remote monitor to detect changes at server.

15.35 src/ArnInc/ArnPersist.hpp File Reference

```
#include "ArnLib_global.hpp"
```

Include dependency graph for ArnPersist.hpp:



```
graph BT; A[src/ArnPersist.cpp] --> B[src/ArnInc/ArnPersist.hpp];
```

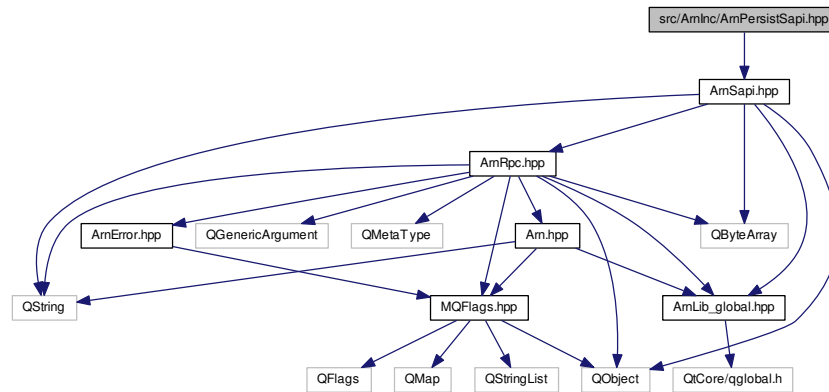
- class **ArnPersist**

Namespaces

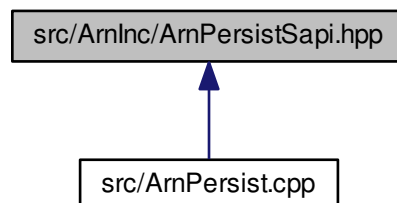
- Arn

```
#include "ArnSapi.hpp"
```

Include dependency graph for ArnPersistSapi.hpp:



This graph shows which files directly or indirectly include this file:



15.37 src/ArnInc/ArnPipe.hpp File Reference

```

#include "ArnLib_global.hpp"
#include "ArnItemB.hpp"

```


15.38.1 Macro Definition Documentation

15.38.1.1 `#define QML_ENGINE QDeclarativeEngine`

Definition at line 60 of file ArnQml.hpp.

15.38.1.2 `#define QML_LIST_PROPERTY QDeclarativeListProperty`

Definition at line 63 of file ArnQml.hpp.

15.38.1.3 `#define QML_NETACC_FACTORY QDeclarativeNetworkAccessManagerFactory`

Definition at line 62 of file ArnQml.hpp.

15.38.1.4 `#define QML_PARSER_STATUS QDeclarativeParserStatus`

Definition at line 61 of file ArnQml.hpp.

15.38.1.5 `#define QML_Qt4`

Definition at line 58 of file ArnQml.hpp.

15.38.1.6 `#define QML_QUICK_TYPE 1`

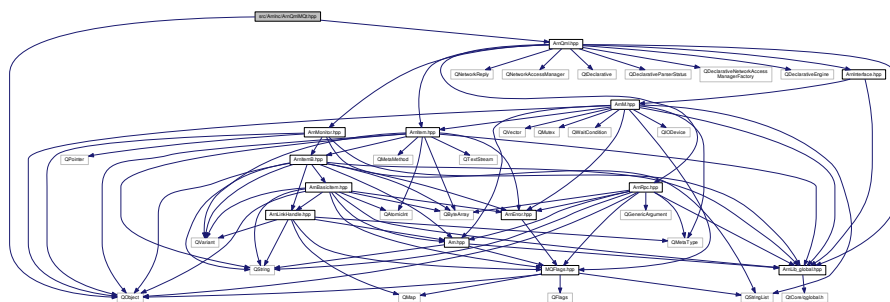
Definition at line 59 of file ArnQml.hpp.

15.39 src/ArnInc/ArnQmlMQt.hpp File Reference

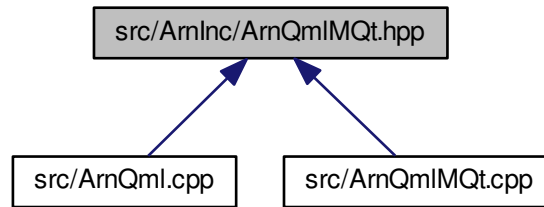
```
#include "ArnQml.hpp"
```

```
#include <QObject>
```

Include dependency graph for ArnQmlMQt.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Arn::QmlMQtObject](#)

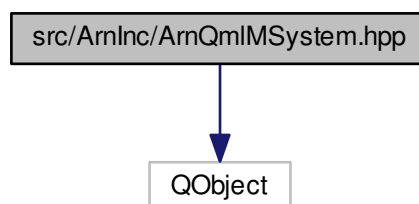
Namespaces

- [Arn](#)

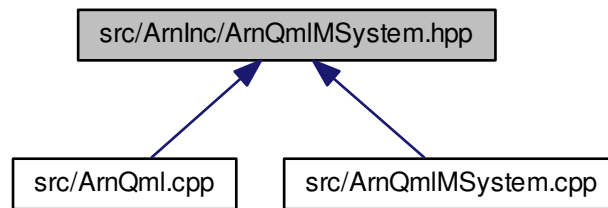
15.40 src/ArnInc/ArnQmlMSystem.hpp File Reference

```
#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](#)

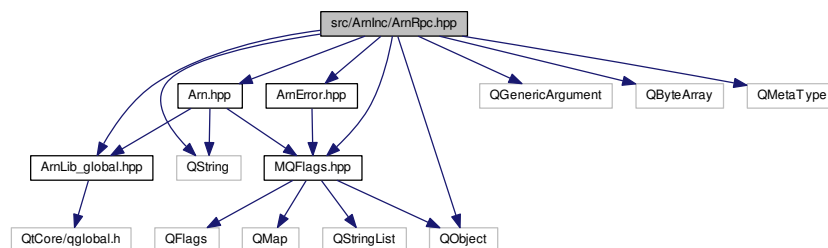
15.41 src/ArnInc/ArnRpc.hpp File Reference

```

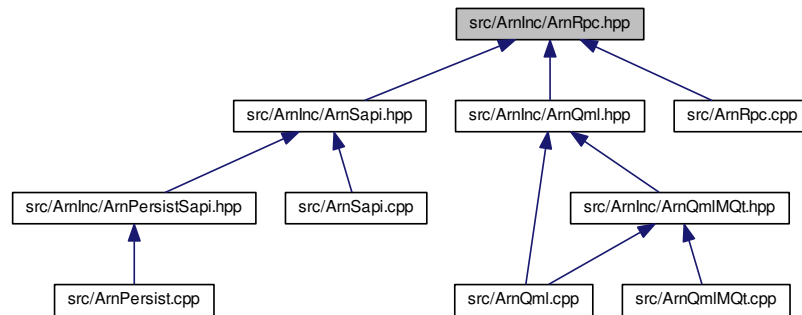
#include "ArnLib_global.hpp"
#include "Arn.hpp"
#include "ArnError.hpp"
#include "MQFlags.hpp"
#include <QGenericArgument>
#include <QString>
#include <QByteArray>
#include <QObject>
#include <QMetaType>

```

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)

15.41.1 Macro Definition Documentation

15.41.1.1 `#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 48 of file ArnRpc.hpp.

15.41.1.2 `#define no_queue`

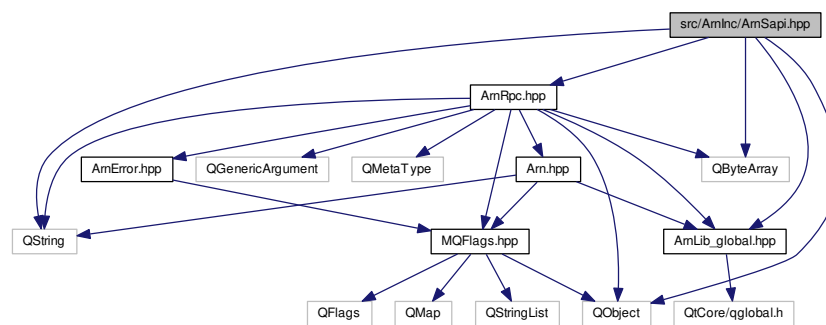
Examples:

[ArnDemoChatServer/ChatSapi.hpp](#).

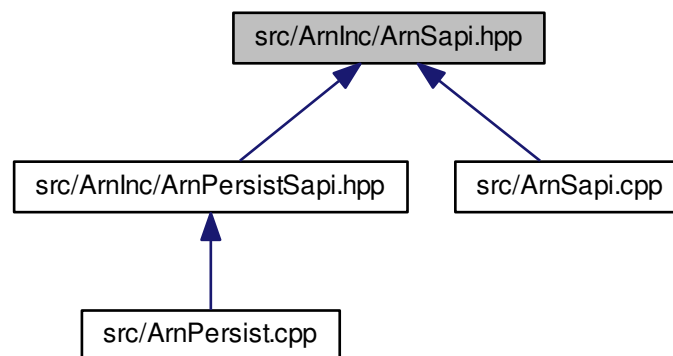
Definition at line 35 of file ArnRpc.hpp.

15.42 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:



Classes

- class [ArnSapi](#)
Service API.

Macros

- `#define` [MQ_PUBLIC_ACCESS](#)

15.42.1 Macro Definition Documentation

15.42.1.1 #define MQ_PUBLIC_ACCESS

Examples:

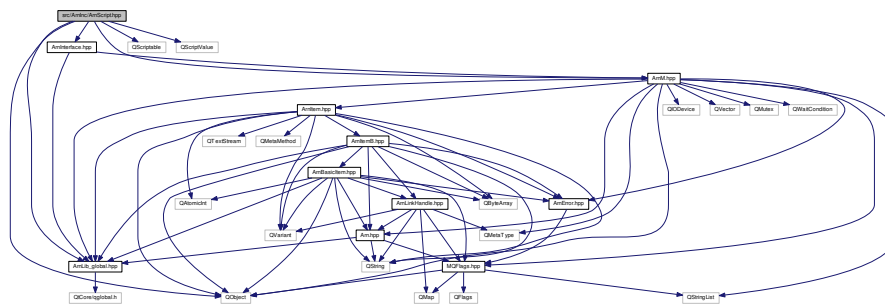
[ArnDemoChatServer/ChatSapi.hpp](#).

Definition at line 44 of file ArnSapi.hpp.

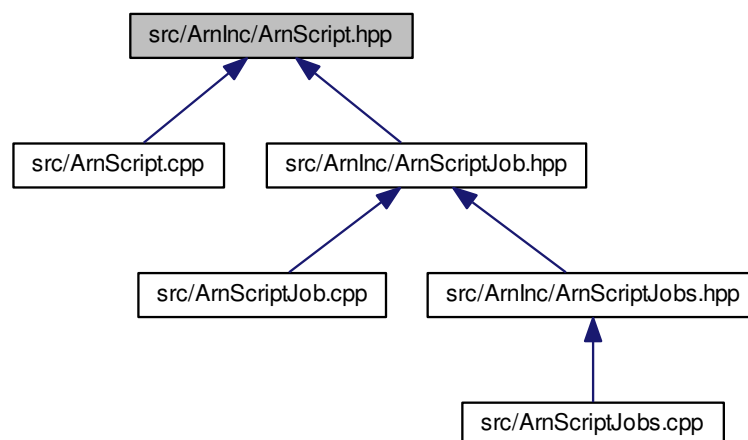
15.43 src/ArnInc/ArnScript.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnInterface.hpp"
#include "ArnM.hpp"
#include <QObject>
#include <QScriptable>
#include <QScriptValue>
```

Include dependency graph for ArnScript.hpp:



This graph shows which files directly or indirectly include this file:



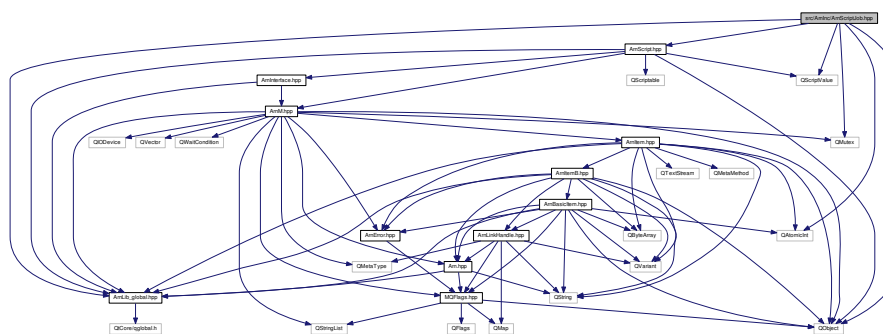
Classes

- class **ArnScript**

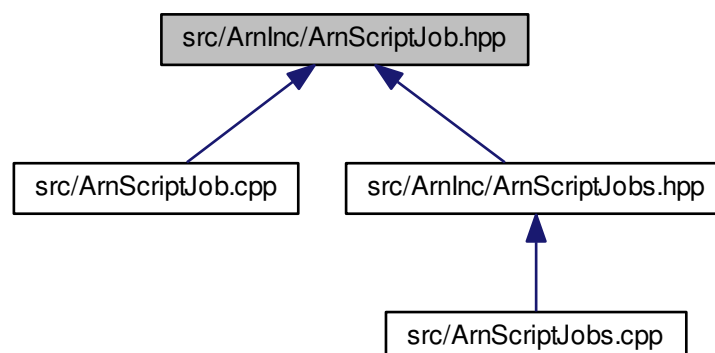
15.44 src/ArnInc/ArnScriptJob.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnScript.hpp"
#include <QScriptValue>
#include <QObject>
#include <QAtomicInt>
#include <QMutex>
```

Include dependency graph for ArnScriptJob.hpp:



This graph shows which files directly or indirectly include this file:

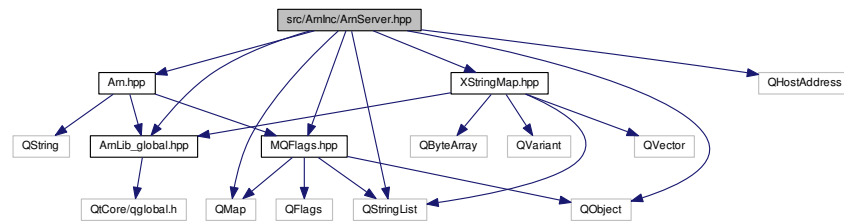


Classes

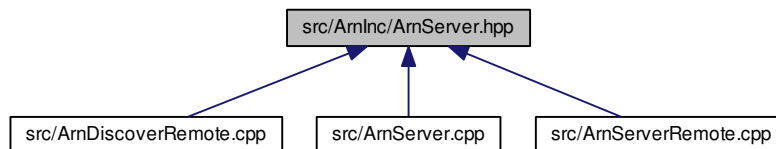
- class [ArnScriptJob](#)
Interface class to be normally used, is also Script Job interface.
- class [ArnScriptJobFactory](#)
Must be thread-safe as subclassed.


```
#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:



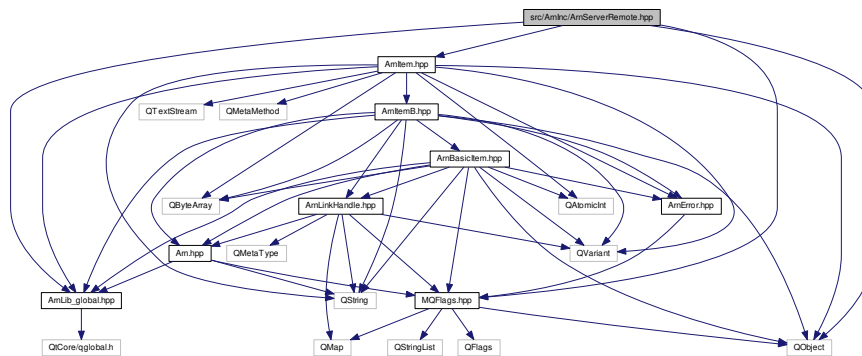
Classes

- class [ArnServerSession](#)
- class [ArnServer](#)
 - Class for making an [Arn](#) Server.*
- struct [ArnServer::Type](#)

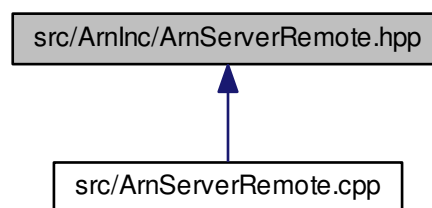
15.47 src/ArnInc/ArnServerRemote.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnItem.hpp"
#include "MQFlags.hpp"
#include <QObject>
```

Include dependency graph for ArnServerRemote.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.

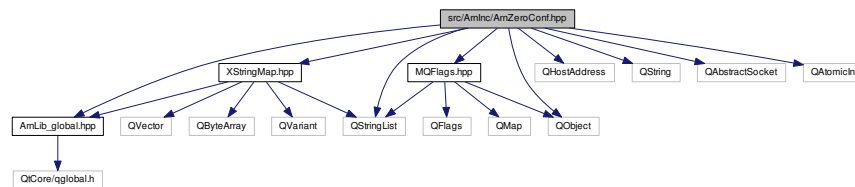
15.48 src/ArnInc/ArnZeroConf.hpp File Reference

```

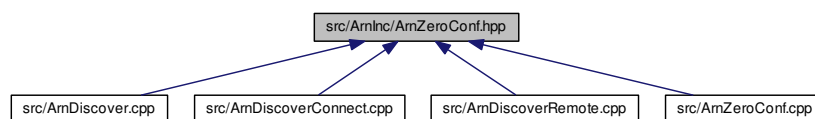
#include "ArnLib_global.hpp"
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QHostAddress>
#include <QObject>
#include <QStringList>
#include <QString>
#include <QAbstractSocket>
#include <QAtomicInt>

```

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](#)

15.48.1 Typedef Documentation

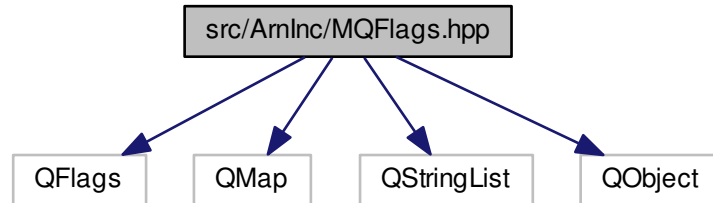
15.48.1.1 typedef struct _DNSServiceRef_t* DNSServiceRef

Definition at line 45 of file ArnZeroConf.hpp.

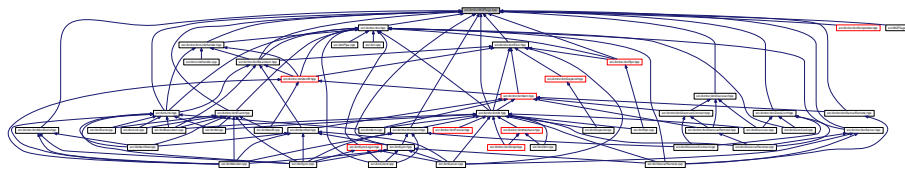
15.49 src/ArnInc/MQFlags.hpp File Reference

```
#include <QFlags>
#include <QMap>
#include <QStringList>
#include <QObject>
```

Include dependency graph for MQFlags.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct [Arn::_InitEnumTxt](#)
- class [Arn::EnumTxt](#)

Class Enum text.

Namespaces

- [Arn](#)

Macros

- `#define MQ_NSTXT_FILL_MISSING 0, 0`
- `#define MQ_NSTXT_FILL_MISSING_FROM(FromNs) FromNs, 0`
- `#define MQ_DECLARE_FLAGS(FEStruct)`
Flags.
- `#define MQ_DECLARE_FLAGSTXT(FEStruct)`
- `#define MQ_DECLARE_FLAGS_NSTXT(...)`
- `#define MQ_DECLARE_OPERATORS_FOR_FLAGS(FEStruct) Q_DECLARE_OPERATORS_FOR_FLAGS(FEStruct::F)`
- `#define MQ_DECLARE_ENUM(EStruct)`

Enums.

- `#define MQ_DECLARE_ENUMTXT(EStruct)`
- `#define MQ_DECLARE_ENUM_NSTXT(...)`

Functions

- `bool Arn::isPower2 (uint x)`

15.49.1 Macro Definition Documentation

15.49.1.1 `#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 395 of file MQFlags.hpp.

15.49.1.2 `#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 415 of file MQFlags.hpp.

15.49.1.3 `#define MQ_DECLARE_ENUMTXT(EStruct)`

Value:

```
MQ_DECLARE_ENUM( EStruct) \
static Arn::EnumTxt& txt() {static Arn::EnumTxt in( staticMetaObject, false, \
    _setNs(0), \
    #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)));}
```

Definition at line 404 of file MQFlags.hpp.

15.49.1.4 `#define MQ_DECLARE_FLAGS(FEStruct)`

Value:

```
Q_DECLARE_FLAGS(F, E) \
    F f; \
    inline FStruct(F v_ = F(0)) : f( v_ ) {setup(0);} \
    inline FStruct(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 FStruct& set(E e, bool v_ = true) {f = v_ ? (f | e) : (f & ~e); return *this;} \
    inline static FStruct fromInt( int v_) {return FStruct( 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 356 of file MQFlags.hpp.

15.49.1.5 #define MQ_DECLARE_FLAGS_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 382 of file MQFlags.hpp.

15.49.1.6 #define MQ_DECLARE_FLAGSTXT(FStruct)

Value:

```
MQ_DECLARE_FLAGS( FStruct) \
    static Arn::EnumTxt& txt() {static Arn::EnumTxt in( staticMetaObject, true, \
        _setNs(0), \
        #FStruct); 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().flagsToString( f, nameSpace);} \
    inline static FStruct fromString( const QString& text, quint16 nameSpace = 0) \
        {return FStruct( F( txt().flagsFromString( text, nameSpace)));}
```

Definition at line 371 of file MQFlags.hpp.

15.49.1.7 #define MQ_DECLARE_OPERATORS_FOR_FLAGS(FStruct) Q_DECLARE_OPERATORS_FOR_FLAGS(FStruct::F)

Definition at line 390 of file MQFlags.hpp.

15.49.1.8 #define MQ_NSTXT_FILL_MISSING 0, 0

Definition at line 351 of file MQFlags.hpp.

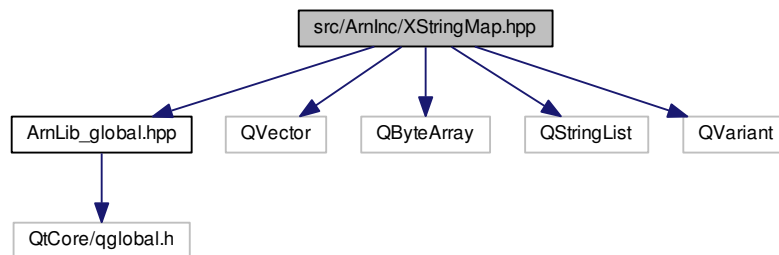
15.49.1.9 #define MQ_NSTXT_FILL_MISSING_FROM(FromNs) FromNs, 0

Definition at line 352 of file MQFlags.hpp.

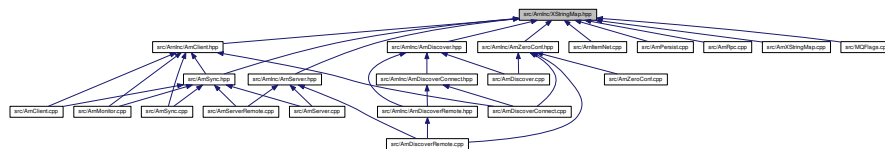
15.50 src/ArnInc/XStringMap.hpp File Reference

```
#include "ArnLib_global.hpp"
#include <QVector>
#include <QByteArray>
#include <QStringList>
#include <QVariant>
```

Include dependency graph for XStringMap.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `Arn::XStringMap`
Container class with string representation for serialized data.

Namespaces

- Arn

Macros

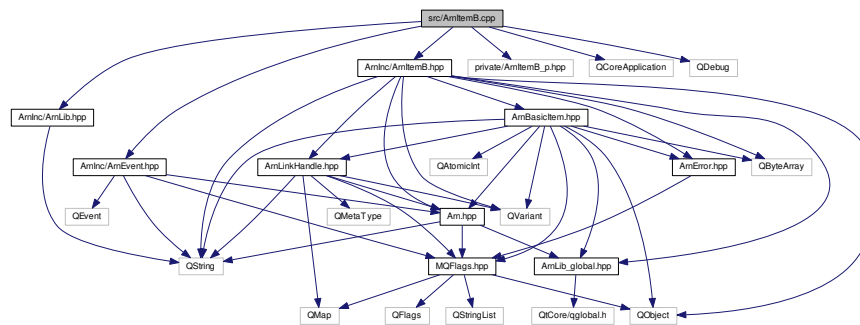
- #define ARNXSTRINGMAP VER "3.0"

15.50.1 Macro Definition Documentation

```
15.50.1.1 #define ARNXSTRINGMAP VER "3.0"
```

Definition at line 51 of file XStringMap.hpp.

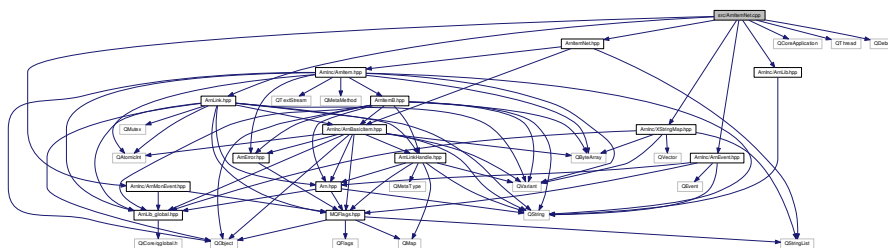
Include dependency graph for ArnItemB.cpp:



15.53 src/ArnItemNet.cpp File Reference

```
#include "ArnItemNet.hpp"
#include "ArnLink.hpp"
#include "ArnInc/ArnMonEvent.hpp"
#include "ArnInc/ArnEvent.hpp"
#include "ArnInc/XStringMap.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QCoreApplication>
#include <QThread>
#include <QDebug>
```

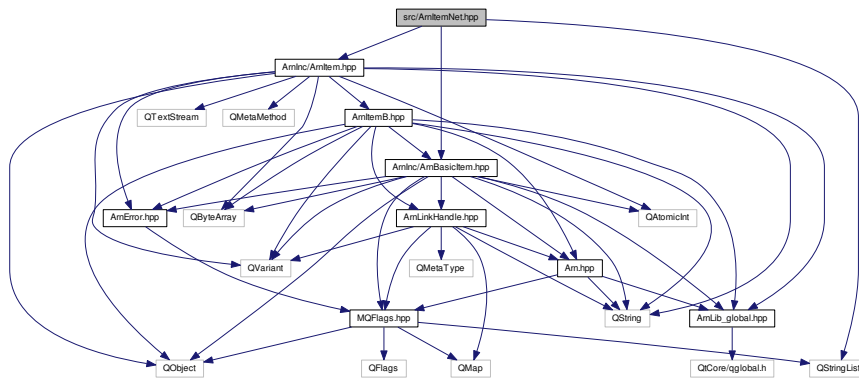
Include dependency graph for ArnItemNet.cpp:



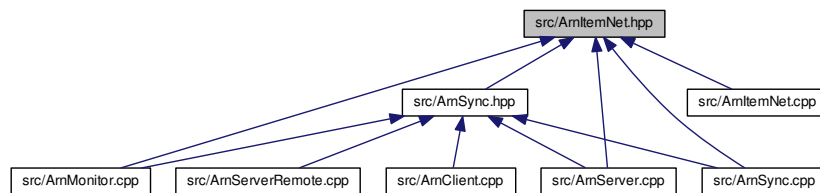
15.54 src/ArnItemNet.hpp File Reference

```
#include "ArnInc/ArnBasicItem.hpp"
#include "ArnInc/ArnItem.hpp"
#include <QStringList>
```

Include dependency graph for ArnItemNet.hpp:

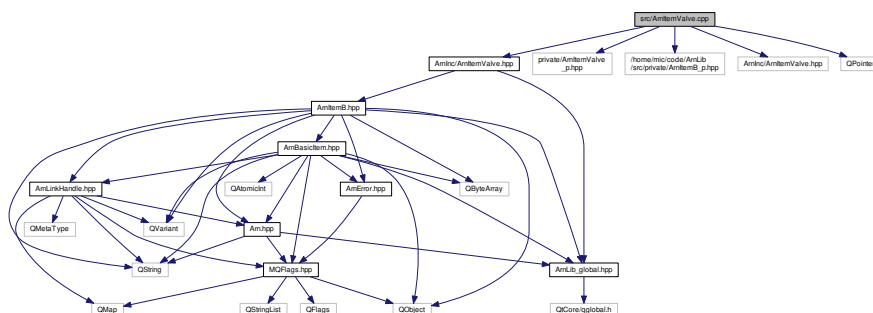


This graph shows which files directly or indirectly include this file:



15.55 src/ArnItemValve.cpp File Reference

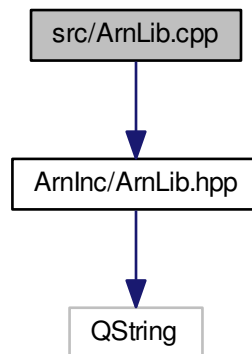
```
#include "ArnInc/ArnItemValve.hpp"
#include "private/ArnItemValve_p.hpp"
Include dependency graph for ArnItemValve.cpp:
```



15.56 src/ArnLib.cpp File Reference

```
#include "ArnInc/ArnLib.hpp"
```

Include dependency graph for ArnLib.cpp:



Namespaces

- [Arn](#)

Variables

- bool [Arn::debugSizes](#) = false
- bool [Arn::debugThreading](#) = false
- bool [Arn::debugLinkRef](#) = false
- bool [Arn::debugLinkDestroy](#) = false
- bool [Arn::debugRecInOut](#) = false
- 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
- const QString [Arn::resourceArnLib](#) = ":/ArnLib/"
- const QString [Arn::resourceArnRoot](#) = ":/ArnLib/ArnRoot/"

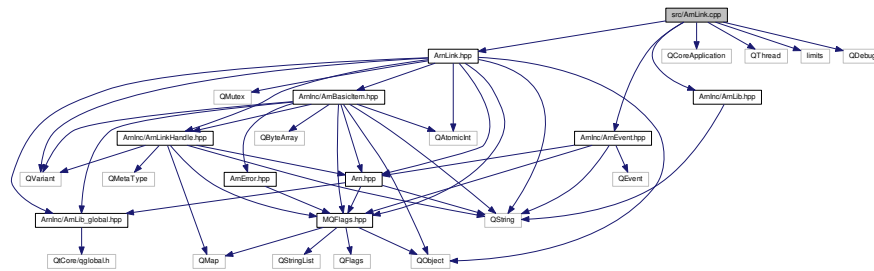
15.57 src/ArnLink.cpp File Reference

```

#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](#)

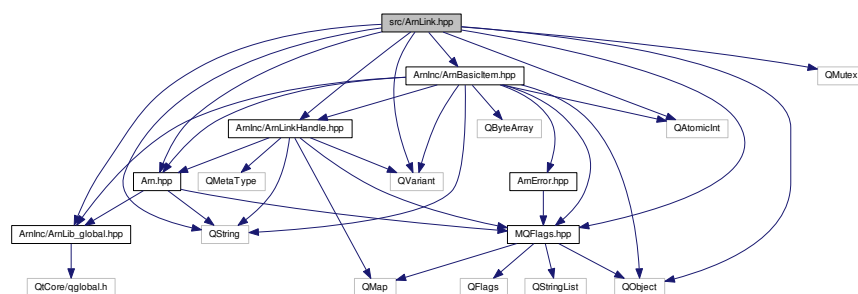
15.58 src/ArnLink.hpp File Reference

```

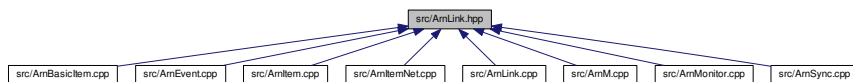
#include "ArnInc/ArnLib_global.hpp"
#include "ArnInc/ArnLinkHandle.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnInc/ArnBasicItem.hpp"
#include "ArnInc/MQFlags.hpp"
#include <QObject>
#include <QString>
#include <QVariant>
#include <QAtomicInt>
#include <QMutex>

```

Include dependency graph for ArnLink.hpp:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef QList< ArnLink * > [ArnLinkList](#)
- typedef QList< [ArnBasicItem](#) * > [ArnBasicItemList](#)

15.58.1 Typedef Documentation

15.58.1.1 typedef QList<ArnBasicItem*> ArnBasicItemList

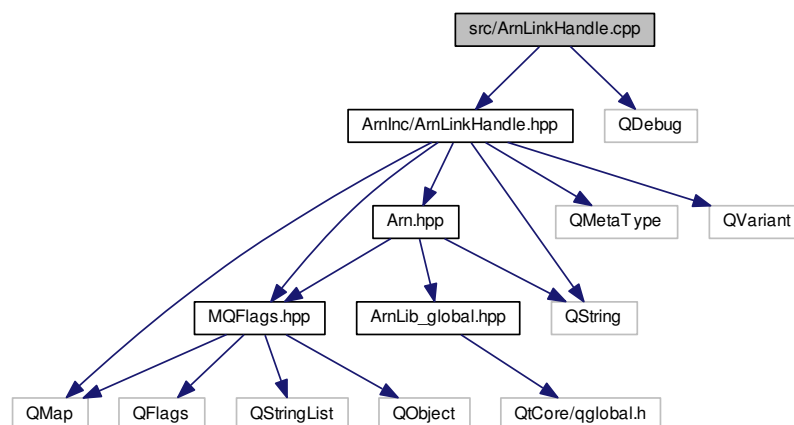
Definition at line 51 of file ArnLink.hpp.

15.58.1.2 typedef QList<ArnLink*> ArnLinkList

Definition at line 48 of file ArnLink.hpp.

15.59 src/ArnLinkHandle.cpp File Reference

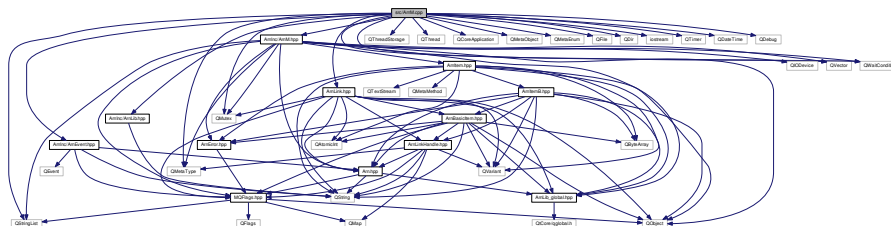
```
#include "ArnInc/ArnLinkHandle.hpp"
#include <QDebug>
Include dependency graph for ArnLinkHandle.cpp:
```



15.60 src/ArnM.cpp File Reference

```
#include "ArnInc/ArnM.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnInc/ArnEvent.hpp"
#include "ArnLink.hpp"
#include <QMutex>
#include <QWaitCondition>
#include <QThreadStorage>
#include <QThread>
#include <QCoreApplication>
#include <QMetaType>
#include <QMetaObject>
#include <QMetaEnum>
#include <QFile>
#include <QDir>
#include <iostream>
#include <QTimer>
#include <QDateTime>
#include <QStringList>
#include <QVector>
#include <QDebug>
```

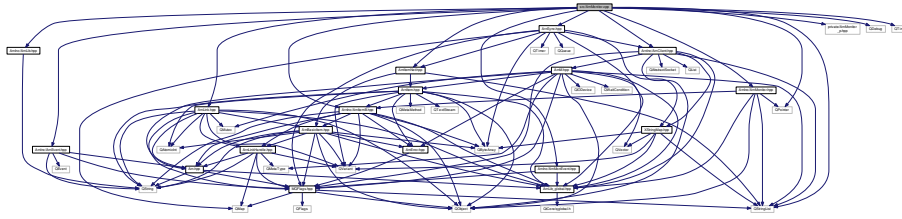
Include dependency graph for ArnM.cpp:



15.61 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>
```

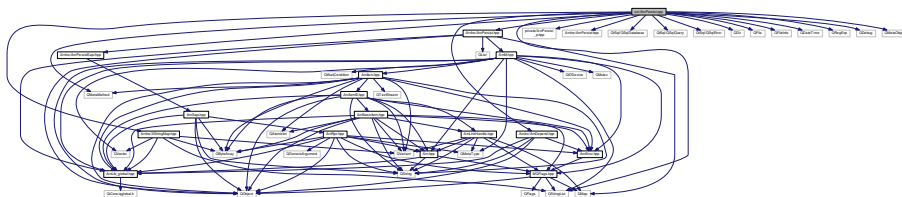

Include dependency graph for ArnMonitor.cpp:



15.62 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 <QtSql/QtSqlDatabase>
#include <QtSql/QtSqlQuery>
#include <QtSql/QtSqlError>
#include <QDir>
#include <QFile>
#include <QFileInfo>
#include <QDateTime>
#include <QRegExp>
#include <QStringList>
#include <QDebug>
#include <QMetaObject>
#include <QMetaMethod>
```

Include dependency graph for ArnPersist.cpp:



Variables

- const int [arnDbSaveVer](#) = 200

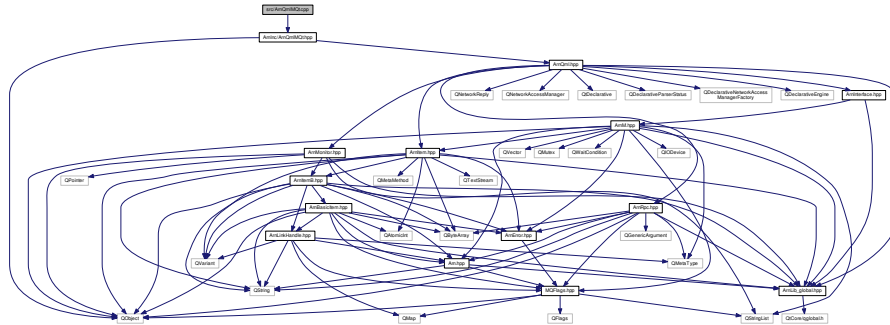
15.62.1 Variable Documentation

15.62.1.1 const int arnDbSaveVer = 200

Definition at line 53 of file ArnPersist.cpp.

15.65 src/ArnQmlMQt.cpp File Reference

```
#include "ArnInc/ArnQmlMQt.hpp"
Include dependency graph for ArnQmlMQt.cpp:
```



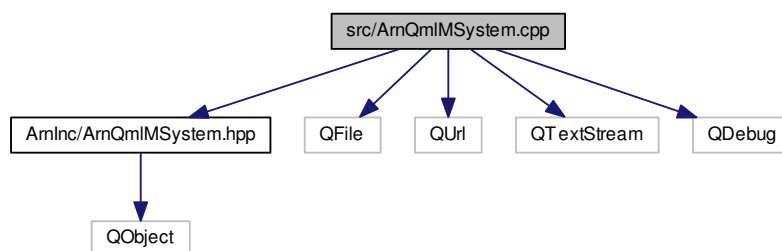
Namespaces

- Arn

15.66 src/ArnQmlMSystem.cpp File Reference

```
#include "ArnInc/ArnQmlMSystem.hpp"
#include <QFile>
#include <QUrl>
#include <QTextStream>
#include <QDebug>
```

Include dependency graph for ArnQmlMSystem.cpp:



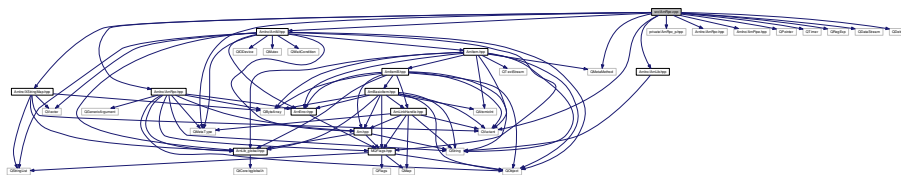
Namespaces

- Arn

15.67 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 <QMetaType>
#include <QMetaMethod>
#include <QTimer>
#include <QRegExp>
#include <QDataStream>
#include <QVariant>
#include <QDebug>
```

Include dependency graph for ArnRpc.cpp:



Macros

- `#define RPC_STORAGE_NAME "_ArnRpcStorage"`

15.67.1 Macro Definition Documentation

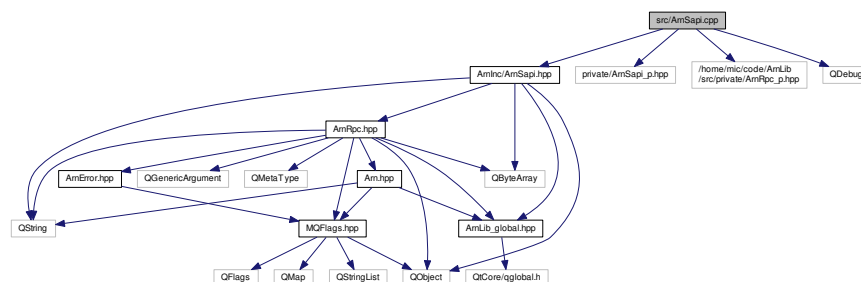
15.67.1.1 `#define RPC_STORAGE_NAME "_ArnRpcStorage"`

Definition at line 47 of file ArnRpc.cpp.

15.68 src/ArnSapi.cpp File Reference

```
#include "ArnInc/ArnSapi.hpp"
#include "private/ArnSapi_p.hpp"
#include <QDebug>
```

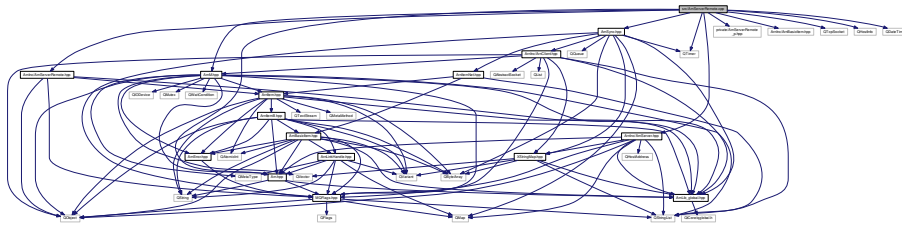
Include dependency graph for ArnSapi.cpp:



15.73 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 <QDateTime>
```

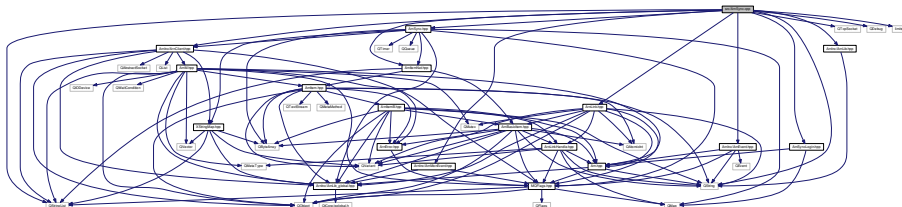
Include dependency graph for ArnServerRemote.cpp:



15.74 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/ArnLib.hpp"
#include <QTcpSocket>
#include <QString>
#include <QStringList>
#include <QDebug>
#include <limits.h>
```

Include dependency graph for ArnSync.cpp:



Macros

- `#define ARNSYNCOVER "2.0"`

15.74.1 Macro Definition Documentation

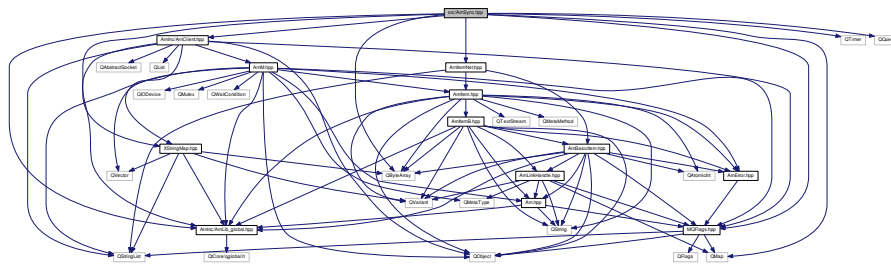
15.74.1.1 #define ARNSYNCVER "2.0"

Definition at line 46 of file ArnSync.cpp.

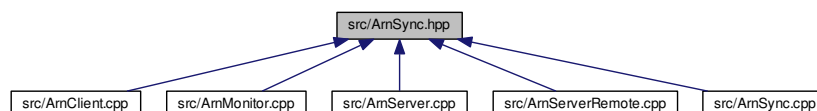
15.75 src/ArnSync.hpp File Reference

```
#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](#) ""

15.75.1 Macro Definition Documentation

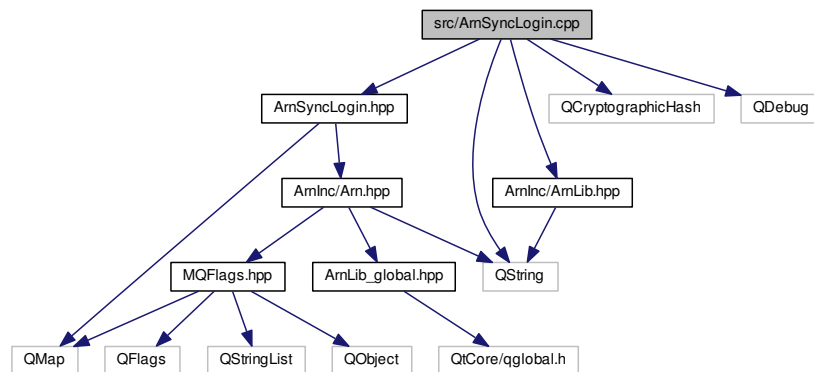
15.75.1.1 #define ARNRECNAME ""

Definition at line 45 of file ArnSync.hpp.

15.76 src/ArnSyncLogin.cpp File Reference

```
#include "ArnSyncLogin.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QCryptographicHash>
#include <QString>
#include <QDebug>
```

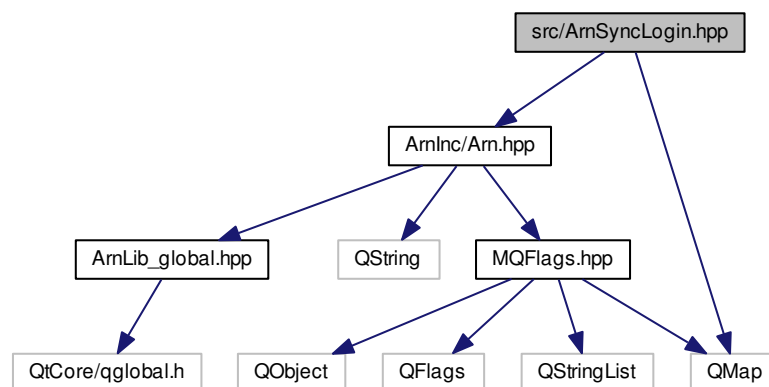
Include dependency graph for ArnSyncLogin.cpp:



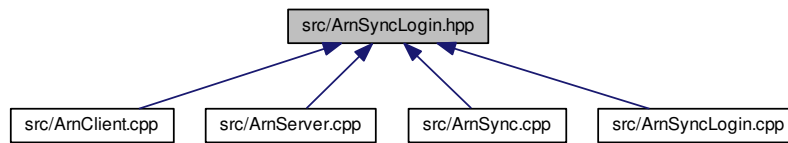
15.77 src/ArnSyncLogin.hpp File Reference

```
#include "ArnInc/Arn.hpp"
#include <QMap>
```

Include dependency graph for ArnSyncLogin.hpp:



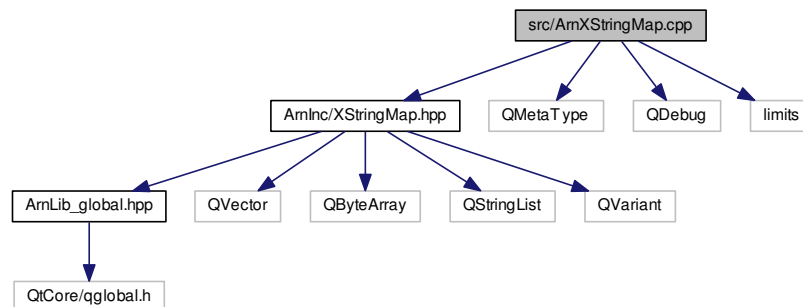
This graph shows which files directly or indirectly include this file:



15.78 src/ArnXStringMap.cpp File Reference

```
#include "ArnInc/XStringMap.hpp"
#include <QMetaType>
#include <QDebug>
#include <limits>
```

Include dependency graph for `ArnXStringMap.cpp`:



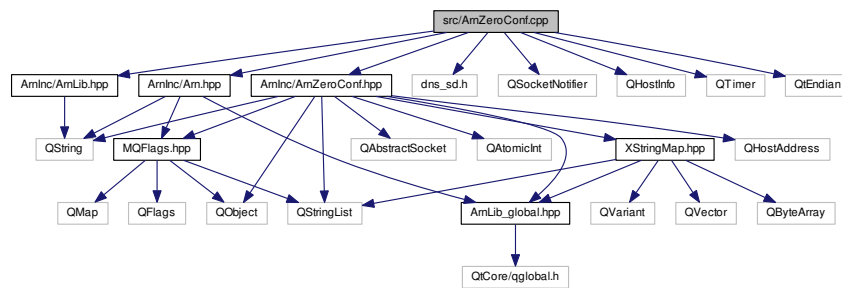
Namespaces

- [Arn](#)

15.79 src/ArnZeroConf.cpp File Reference

```
#include "ArnInc/ArnZeroConf.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnInc/ArnLib.hpp"
#include <dns_sd.h>
#include <QSocketNotifier>
#include <QHostInfo>
#include <QTimer>
#include <QtEndian>
```

Include dependency graph for ArnZeroConf.cpp:



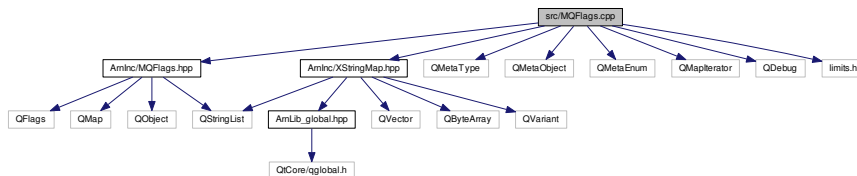
15.80 src/MQFlags.cpp File Reference

```

#include "ArnInc/MQFlags.hpp"
#include <ArnInc/XStringMap.hpp>
#include <QMetaType>
#include <QMetaObject>
#include <QMetaEnum>
#include <QMapIterator>
#include <QDebug>
#include <limits.h>

```

Include dependency graph for MQFlags.cpp:



Namespaces

- [Arn](#)

Functions

- bool [Arn::isPower2](#) (uint x)

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@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.
//
#ifdef 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

- Accept
 - Arn::SameValue, 333
- Advertise
 - ArnDiscoverAdvertise::State, 336
- Advertising
 - ArnDiscoverAdvertise::State, 336
- All
 - Arn::Allow, 60
 - ArnQml::UseFlags, 339
- AlreadyExist
 - ArnError, 142
- AlreadyOpen
 - ArnError, 142
- AnyNamedArg
 - ArnRpcMode, 247
- Arn, 49
- Arn::Allow
 - All, 60
 - Create, 60
 - Delete, 60
 - ModeChange, 60
 - None, 60
 - Read, 60
 - ReadWrite, 60
 - Write, 60
- Arn::Coding
 - Binary, 307
 - Text, 307
- Arn::DataType
 - ByteArray, 308
 - Double, 308
 - Int, 308
 - Null, 308
 - Real, 308
 - String, 308
 - Variant, 308
- Arn::ExportCode
 - ByteArray, 318
 - String, 318
 - Variant, 319
 - VariantBin, 319
 - VariantTxt, 319
- Arn::InfoType
 - Custom, 320
 - N, 320
- Arn::LinkFlags
 - CreateAllowed, 321
 - Folder, 321
 - LastLink, 321
 - SilentError, 321
 - Threaded, 321
- Arn::NameF
 - Default, 326
 - EmptyOk, 326
 - NoFolderMark, 326
 - Relative, 326
- Arn::ObjectMode
 - BiDir, 327
 - Normal, 327
 - Pipe, 327
 - Save, 327
- Arn::ObjectSyncMode
 - AutoDestroy, 328
 - Master, 328
 - Monitor, 327
 - Normal, 327
- Arn::SameValue
 - Accept, 333
 - DefaultAction, 333
 - Ignore, 333
- ArnClientConnectStat
 - Connected, 95
 - Connecting, 95
 - Disconnected, 95
 - Error, 95
 - Init, 95
 - Negotiating, 95
 - NsEnum, 95
 - NsHuman, 95
 - Stopped, 95
 - TriedAll, 95
- ArnDiscover::Type
 - Client, 338
 - None, 338
 - Server, 338
- ArnDiscoverAdvertise::State
 - Advertise, 336
 - Advertising, 336
 - None, 336
 - StartupAdvertise, 336
- ArnDiscoverInfo::State
 - HostInfo, 334
 - HostInfoErr, 334
 - HostIp, 334
 - HostIpErr, 334
 - Init, 334
 - ServiceName, 334
- ArnError

- AlreadyExist, [142](#)
- AlreadyOpen, [142](#)
- ConnectionError, [142](#)
- CreateError, [142](#)
- Err_Custom, [142](#)
- Err_N, [142](#)
- Err_Undef, [142](#)
- FolderNotOpen, [142](#)
- Info, [141](#)
- ItemNotOpen, [142](#)
- ItemNotSet, [142](#)
- LoginBad, [142](#)
- NotFound, [142](#)
- NotMainThread, [142](#)
- NotOpen, [142](#)
- Ok, [141](#)
- OpNotAllowed, [142](#)
- RecNotExpected, [142](#)
- RecUnknown, [142](#)
- Retired, [142](#)
- RpcInvokeError, [142](#)
- RpcReceiveError, [142](#)
- ScriptError, [142](#)
- Undef, [141](#)
- Warning, [141](#)
- ArnError::StdCode
 - Err_Custom, [336](#)
 - Err_Undef, [336](#)
 - Info, [336](#)
 - Ok, [336](#)
 - Warning, [336](#)
- ArnEventIdx
 - LinkCreate, [146](#)
 - ModeChange, [146](#)
 - Monitor, [146](#)
 - N, [146](#)
 - QtEvent, [146](#)
 - RefChange, [146](#)
 - Retired, [146](#)
 - ValueChange, [146](#)
 - ZeroRef, [146](#)
- ArnInterface
 - DataType_ByteArray, [162](#)
 - DataType_Double, [162](#)
 - DataType_Int, [162](#)
 - DataType_Null, [162](#)
 - DataType_Real, [162](#)
 - DataType_String, [162](#)
 - DataType_Variant, [162](#)
 - NameF_Default, [163](#)
 - NameF_EmptyOk, [163](#)
 - NameF_NoFolderMark, [163](#)
 - NameF_Relative, [163](#)
 - ObjectMode_BiDir, [163](#)
 - ObjectMode_Pipe, [163](#)
 - ObjectMode_Save, [163](#)
 - SameValue_Accept, [163](#)
 - SameValue_DefaultAction, [163](#)
 - SameValue_Ignore, [163](#)
- ArnItemValve::SwitchMode
 - InOutputStream, [337](#)
 - InStream, [337](#)
 - OutputStream, [337](#)
- ArnLib
 - ArnQml::UseFlags, [339](#)
- ArnMonEventType
 - ItemCreated, [210](#)
 - ItemDeleted, [210](#)
 - ItemFound, [210](#)
 - ItemModeChg, [210](#)
 - MonitorReStart, [210](#)
 - MonitorStart, [210](#)
 - None, [210](#)
 - NsCom, [210](#)
 - NsEnum, [210](#)
- ArnQml::UseFlags
 - All, [339](#)
 - ArnLib, [339](#)
 - MQt, [339](#)
 - MSystem, [339](#)
- ArnRpc::Invoke
 - NoQueue, [320](#)
- ArnRpcMode
 - AnyNamedArg, [247](#)
 - AutoDestroy, [246](#)
 - CheckSequence, [247](#)
 - Debug, [247](#)
 - NamedArg, [247](#)
 - NamedTypedArg, [247](#)
 - NoDefaultArgs, [246](#)
 - OnlyPosArgIn, [247](#)
 - Provider, [246](#)
 - SendSequence, [246](#)
 - UseDefaultCall, [247](#)
 - UuidAutoDestroy, [247](#)
 - UuidPipe, [246](#)
- ArnSapiQml
 - AutoDestroy, [253](#)
 - CheckSequence, [254](#)
 - NamedArg, [254](#)
 - NamedTypedArg, [254](#)
 - NoDefaultArgs, [253](#)
 - Provider, [253](#)
 - SendSequence, [254](#)
 - UseDefaultCall, [254](#)
 - UuidAutoDestroy, [254](#)
 - UuidPipe, [253](#)
- ArnScriptJobs::Type
 - Cooperative, [337](#)
 - Null, [337](#)
 - Preemptive, [337](#)
- ArnServer::Type
 - NetSync, [338](#)
- ArnServerRemoteSessionKillMode
 - Delay10Sec, [273](#)
 - Delay60Sec, [273](#)

- Off, [273](#)
- ArnZeroConf::Error
 - BadReqSeq, [318](#)
 - Ok, [318](#)
 - Running, [318](#)
 - Timeout, [318](#)
 - UDnsFail, [318](#)
- ArnZeroConf::State
 - Browsing, [335](#)
 - InProgress, [335](#)
 - LookingUp, [335](#)
 - Lookup, [335](#)
 - Lookuped, [335](#)
 - None, [335](#)
 - Register, [335](#)
 - Registered, [335](#)
 - Registering, [335](#)
 - Resolve, [335](#)
 - Resolved, [335](#)
 - Resolving, [335](#)
- AutoDestroy
 - Arn::ObjectSyncMode, [328](#)
 - ArnRpcMode, [246](#)
 - ArnSapiQml, [253](#)
- BadReqSeq
 - ArnZeroConf::Error, [318](#)
- BiDir
 - Arn::ObjectMode, [327](#)
- Binary
 - Arn::Coding, [307](#)
- Browsing
 - ArnZeroConf::State, [335](#)
- ByteArray
 - Arn::DataType, [308](#)
 - Arn::ExportCode, [318](#)
- CheckSequence
 - ArnRpcMode, [247](#)
 - ArnSapiQml, [254](#)
- Client
 - ArnDiscover::Type, [338](#)
- Connected
 - ArnClientConnectStat, [95](#)
- Connecting
 - ArnClientConnectStat, [95](#)
- ConnectionError
 - ArnError, [142](#)
- Cooperative
 - ArnScriptJobs::Type, [337](#)
- Create
 - Arn::Allow, [60](#)
- CreateAllowed
 - Arn::LinkFlags, [321](#)
- CreateError
 - ArnError, [142](#)
- Custom
 - Arn::InfoType, [320](#)
- DataType_ByteArray
 - ArnInterface, [162](#)
- DataType_Double
 - ArnInterface, [162](#)
- DataType_Int
 - ArnInterface, [162](#)
- DataType_Null
 - ArnInterface, [162](#)
- DataType_Real
 - ArnInterface, [162](#)
- DataType_String
 - ArnInterface, [162](#)
- DataType_Variant
 - ArnInterface, [162](#)
- Debug
 - ArnRpcMode, [247](#)
- Default
 - Arn::NameF, [326](#)
- DefaultAction
 - Arn::SameValue, [333](#)
- Delay10Sec
 - ArnServerRemoteSessionKillMode, [273](#)
- Delay60Sec
 - ArnServerRemoteSessionKillMode, [273](#)
- Delete
 - Arn::Allow, [60](#)
- Disconnected
 - ArnClientConnectStat, [95](#)
- Double
 - Arn::DataType, [308](#)
- EmptyOk
 - Arn::NameF, [326](#)
- Err_Custom
 - ArnError, [142](#)
 - ArnError::StdCode, [336](#)
- Err_N
 - ArnError, [142](#)
- Err_Undef
 - ArnError, [142](#)
 - ArnError::StdCode, [336](#)
- Error
 - ArnClientConnectStat, [95](#)
- Folder
 - Arn::LinkFlags, [321](#)
- FolderNotOpen
 - ArnError, [142](#)
- HostInfo
 - ArnDiscoverInfo::State, [334](#)
- HostInfoErr
 - ArnDiscoverInfo::State, [334](#)
- HostIp
 - ArnDiscoverInfo::State, [334](#)
- HostIpErr
 - ArnDiscoverInfo::State, [334](#)
- Ignore

- Arn::SameValue, [333](#)
- InOutputStream
 - ArnItemValve::SwitchMode, [337](#)
- InProgress
 - ArnZeroConf::State, [335](#)
- InStream
 - ArnItemValve::SwitchMode, [337](#)
- Info
 - ArnError, [141](#)
 - ArnError::StdCode, [336](#)
- Init
 - ArnClientConnectStat, [95](#)
 - ArnDiscoverInfo::State, [334](#)
- Int
 - Arn::DataType, [308](#)
- ItemCreated
 - ArnMonEventType, [210](#)
- ItemDeleted
 - ArnMonEventType, [210](#)
- ItemFound
 - ArnMonEventType, [210](#)
- ItemModeChg
 - ArnMonEventType, [210](#)
- ItemNotOpen
 - ArnError, [142](#)
- ItemNotSet
 - ArnError, [142](#)
- LastLink
 - Arn::LinkFlags, [321](#)
- LinkCreate
 - ArnEventIdx, [146](#)
- LoginBad
 - ArnError, [142](#)
- LookingUp
 - ArnZeroConf::State, [335](#)
- Lookup
 - ArnZeroConf::State, [335](#)
- Lookuped
 - ArnZeroConf::State, [335](#)
- MQT
 - ArnQml::UseFlags, [339](#)
- MSystem
 - ArnQml::UseFlags, [339](#)
- Master
 - Arn::ObjectSyncMode, [328](#)
- ModeChange
 - Arn::Allow, [60](#)
 - ArnEventIdx, [146](#)
- Monitor
 - Arn::ObjectSyncMode, [327](#)
 - ArnEventIdx, [146](#)
- MonitorReStart
 - ArnMonEventType, [210](#)
- MonitorStart
 - ArnMonEventType, [210](#)
- N
 - Arn::InfoType, [320](#)
 - ArnEventIdx, [146](#)
 - NameF_Default
 - ArnInterface, [163](#)
 - NameF_EmptyOk
 - ArnInterface, [163](#)
 - NameF_NoFolderMark
 - ArnInterface, [163](#)
 - NameF_Relative
 - ArnInterface, [163](#)
 - NamedArg
 - ArnRpcMode, [247](#)
 - ArnSapiQml, [254](#)
 - NamedTypedArg
 - ArnRpcMode, [247](#)
 - ArnSapiQml, [254](#)
 - Negotiating
 - ArnClientConnectStat, [95](#)
 - NetSync
 - ArnServer::Type, [338](#)
 - NoDefaultArgs
 - ArnRpcMode, [246](#)
 - ArnSapiQml, [253](#)
 - NoFolderMark
 - Arn::NameF, [326](#)
 - NoQueue
 - ArnRpc::Invoke, [320](#)
 - None
 - Arn::Allow, [60](#)
 - ArnDiscover::Type, [338](#)
 - ArnDiscoverAdvertise::State, [336](#)
 - ArnMonEventType, [210](#)
 - ArnZeroConf::State, [335](#)
 - Normal
 - Arn::ObjectMode, [327](#)
 - Arn::ObjectSyncMode, [327](#)
 - NotFound
 - ArnError, [142](#)
 - NotMainThread
 - ArnError, [142](#)
 - NotOpen
 - ArnError, [142](#)
 - NsCom
 - ArnMonEventType, [210](#)
 - NsEnum
 - ArnClientConnectStat, [95](#)
 - ArnMonEventType, [210](#)
 - NsHuman
 - ArnClientConnectStat, [95](#)
 - Null
 - Arn::DataType, [308](#)
 - ArnScriptJobs::Type, [337](#)
 - ObjectMode_BiDir
 - ArnInterface, [163](#)
 - ObjectMode_Pipe
 - ArnInterface, [163](#)
 - ObjectMode_Save
 - ArnInterface, [163](#)

- Off
 - ArnServerRemoteSessionKillMode, 273
- Ok
 - ArnError, 141
 - ArnError::StdCode, 336
 - ArnZeroConf::Error, 318
- OnlyPosArgIn
 - ArnRpcMode, 247
- OpNotAllowed
 - ArnError, 142
- OutStream
 - ArnItemValve::SwitchMode, 337
- Pipe
 - Arn::ObjectMode, 327
- Preemptive
 - ArnScriptJobs::Type, 337
- Provider
 - ArnRpcMode, 246
 - ArnSapiQml, 253
- QtEvent
 - ArnEventIdx, 146
- Read
 - Arn::Allow, 60
- ReadWrite
 - Arn::Allow, 60
- Real
 - Arn::DataType, 308
- RecNotExpected
 - ArnError, 142
- RecUnknown
 - ArnError, 142
- RefChange
 - ArnEventIdx, 146
- Register
 - ArnZeroConf::State, 335
- Registered
 - ArnZeroConf::State, 335
- Registering
 - ArnZeroConf::State, 335
- Relative
 - Arn::NameF, 326
- Resolve
 - ArnZeroConf::State, 335
- Resolved
 - ArnZeroConf::State, 335
- Resolving
 - ArnZeroConf::State, 335
- Retired
 - ArnError, 142
 - ArnEventIdx, 146
- RpcInvokeError
 - ArnError, 142
- RpcReceiveError
 - ArnError, 142
- Running
 - ArnZeroConf::Error, 318
- SameValue_Accept
 - ArnInterface, 163
- SameValue_DefaultAction
 - ArnInterface, 163
- SameValue_Ignore
 - ArnInterface, 163
- Save
 - Arn::ObjectMode, 327
- ScriptError
 - ArnError, 142
- SendSequence
 - ArnRpcMode, 246
 - ArnSapiQml, 254
- Server
 - ArnDiscover::Type, 338
- ServiceName
 - ArnDiscoverInfo::State, 334
- SilentError
 - Arn::LinkFlags, 321
- StartupAdvertise
 - ArnDiscoverAdvertise::State, 336
- Stopped
 - ArnClientConnectStat, 95
- String
 - Arn::DataType, 308
 - Arn::ExportCode, 318
- Text
 - Arn::Coding, 307
- Threaded
 - Arn::LinkFlags, 321
- Timeout
 - ArnZeroConf::Error, 318
- TriedAll
 - ArnClientConnectStat, 95
- UDnsFail
 - ArnZeroConf::Error, 318
- Undef
 - ArnError, 141
- UseDefaultCall
 - ArnRpcMode, 247
 - ArnSapiQml, 254
- UuidAutoDestroy
 - ArnRpcMode, 247
 - ArnSapiQml, 254
- UuidPipe
 - ArnRpcMode, 246
 - ArnSapiQml, 253
- ValueChange
 - ArnEventIdx, 146
- Variant
 - Arn::DataType, 308
 - Arn::ExportCode, 319
- VariantBin
 - Arn::ExportCode, 319
- VariantTxt
 - Arn::ExportCode, 319

Warning

ArnError, [141](#)ArnError::StdCode, [336](#)

Write

Arn::Allow, [60](#)

ZeroRef

ArnEventIdx, [146](#)