

ArnLib

2.2.x

Generated by Doxygen 1.8.1

Tue Jul 29 2014 23:17:17

Contents

1	README	1
2	General Description	5
2.1	Arn Data Objects	5
2.1.1	Modes	5
2.1.2	Local path	6
2.1.3	Naming conventions	6
2.2	Bidirectional Arn Data Objects	6
2.3	Pipe Arn Data Objects	7
2.3.1	Pipe sequence check	7
2.3.2	Pipe anti congest	7
2.4	Persistent Arn Data Objects	8
2.4.1	Saving objects in files	8
2.5	Sharing Arn Data Objects	8
2.5.1	Dynamic port	9
2.6	RPC and SAPI	9
2.6.1	RPC and SAPI method name overload	9
2.6.2	RPC and SAPI communication format	10
2.7	ZeroConfig	12
2.7.1	Service name	12
2.7.2	Sub types	12
2.7.3	Text record	12
2.8	Discover	13
2.9	Discover remote	13
2.10	Application notations	14
3	Installation and usage	15
3.1	Introduction	15
3.2	Documentation	15
3.3	Building ArnLib	15
3.4	Using ArnLib	17

4 ArnLib Internals	19
4.1 ScriptJobs	19
4.2 ArnMonitor	20
4.3 Destroy	21
5 ArnLib Todo	23
6 Example Collection	25
6.1 Chat Demo	25
6.1.1 Chat Server	25
6.1.1.1 ChatSapi.hpp	25
6.1.1.2 MainWindow.hpp	25
6.1.1.3 MainWindow.cpp	26
6.1.1.4 main.cpp	28
6.1.2 Chat Client	28
6.1.2.1 MainWindow.hpp	28
6.1.2.2 MainWindow.cpp	29
6.1.2.3 main.cpp	30
6.1.3 Pictures	30
7 Help descriptions	31
7.1 Discover	31
7.1.1 Description	31
8 Deprecated List	33
9 Namespace Index	35
9.1 Namespace List	35
10 Class Index	37
10.1 Class Hierarchy	37
11 Class Index	39
11.1 Class List	39
12 File Index	41
12.1 File List	41
13 Namespace Documentation	43
13.1 Arn Namespace Reference	43
13.1.1 Function Documentation	44
13.1.1.1 addPath	44
13.1.1.2 changeBasePath	45
13.1.1.3 childPath	45

13.1.1.4	convertName	45
13.1.1.5	convertPath	46
13.1.1.6	fullPath	46
13.1.1.7	hostFromHostWithInfo	46
13.1.1.8	isFolderPath	47
13.1.1.9	isProviderPath	47
13.1.1.10	itemName	47
13.1.1.11	makeHostWithInfo	48
13.1.1.12	makePath	48
13.1.1.13	providerPath	49
13.1.1.14	twinPath	49
13.1.2	Variable Documentation	49
13.1.2.1	debugDepend	49
13.1.2.2	debugDiscover	49
13.1.2.3	debugLinkDestroy	49
13.1.2.4	debugLinkRef	50
13.1.2.5	debugMDNS	50
13.1.2.6	debugMonitor	50
13.1.2.7	debugMonitorTest	50
13.1.2.8	debugReclnOut	50
13.1.2.9	debugRPC	50
13.1.2.10	debugShareObj	50
13.1.2.11	debugThreading	50
13.1.2.12	debugZeroConf	50
13.1.2.13	defaultTcpPort	50
13.1.2.14	pathDiscover	50
13.1.2.15	pathDiscoverConnect	50
13.1.2.16	pathDiscoverThis	51
13.1.2.17	pathLocal	51
13.1.2.18	pathLocalSys	51
13.1.2.19	resourceArnLib	51
13.1.2.20	resourceArnRoot	51
13.1.2.21	warningMDNS	51
13.2	ArnDiscover Namespace Reference	51
13.3	ArnZeroConf Namespace Reference	51
14	Class Documentation	53
14.1	ArnClient Class Reference	53
14.1.1	Detailed Description	54
14.1.2	Member Typedef Documentation	54

14.1.2.1	HostList	54
14.1.3	Constructor & Destructor Documentation	54
14.1.3.1	ArnClient	54
14.1.4	Member Function Documentation	55
14.1.4.1	addMountPoint	55
14.1.4.2	addToArnList	55
14.1.4.3	arnList	55
14.1.4.4	clearArnList	56
14.1.4.5	connectionStatusChanged	56
14.1.4.6	connectStatus	56
14.1.4.7	connectToArn	56
14.1.4.8	connectToArnList	57
14.1.4.9	removeMountPoint	57
14.1.4.10	setAutoConnect	57
14.1.4.11	setMountPoint	57
14.1.4.12	tcpConnected	58
14.1.4.13	tcpDisConnected	58
14.1.4.14	tcpError	58
14.2	ArnDepend Class Reference	58
14.2.1	Detailed Description	59
14.2.2	Member Typedef Documentation	59
14.2.2.1	DepSlot	59
14.2.3	Constructor & Destructor Documentation	59
14.2.3.1	ArnDepend	59
14.2.3.2	~ArnDepend	60
14.2.4	Member Function Documentation	60
14.2.4.1	add	60
14.2.4.2	add	60
14.2.4.3	completed	60
14.2.4.4	setMonitorName	60
14.2.4.5	startMonitor	60
14.3	ArnDependOffer Class Reference	61
14.3.1	Detailed Description	61
14.3.2	Constructor & Destructor Documentation	61
14.3.2.1	ArnDependOffer	61
14.3.3	Member Function Documentation	61
14.3.3.1	advertise	61
14.3.3.2	setStateId	62
14.3.3.3	setStateName	62
14.3.3.4	stateId	62

14.3.3.5	stateName	62
14.4	ArnDiscoverAdvertise Class Reference	62
14.4.1	Detailed Description	64
14.4.2	Constructor & Destructor Documentation	64
14.4.2.1	ArnDiscoverAdvertise	64
14.4.3	Member Function Documentation	64
14.4.3.1	addCustomProperty	64
14.4.3.2	addGroup	65
14.4.3.3	advertiseService	65
14.4.3.4	currentService	65
14.4.3.5	customProperties	66
14.4.3.6	groups	66
14.4.3.7	service	66
14.4.3.8	serviceChanged	67
14.4.3.9	serviceChangeError	67
14.4.3.10	setCustomProperties	67
14.4.3.11	setGroups	67
14.4.3.12	setService	68
14.4.3.13	state	68
14.5	ArnDiscoverBrowser Class Reference	69
14.5.1	Detailed Description	70
14.5.2	Constructor & Destructor Documentation	70
14.5.2.1	ArnDiscoverBrowser	70
14.5.3	Member Function Documentation	70
14.5.3.1	browse	70
14.5.3.2	isBrowsing	71
14.5.3.3	setFilter	71
14.5.3.4	setFilter	71
14.5.3.5	stopBrowse	71
14.6	ArnDiscoverBrowserB Class Reference	72
14.6.1	Detailed Description	73
14.6.2	Constructor & Destructor Documentation	73
14.6.2.1	ArnDiscoverBrowserB	73
14.6.3	Member Function Documentation	73
14.6.3.1	defaultStopState	73
14.6.3.2	goTowardState	73
14.6.3.3	IdToIndex	74
14.6.3.4	indexTold	74
14.6.3.5	infoById	74
14.6.3.6	infoByIndex	75

14.6.3.7	infoByName	75
14.6.3.8	infoUpdated	76
14.6.3.9	serviceAdded	76
14.6.3.10	serviceCount	76
14.6.3.11	serviceNameTold	76
14.6.3.12	serviceRemoved	77
14.6.3.13	setDefaultStopState	77
14.7	ArnDiscoverConnector Class Reference	77
14.7.1	Detailed Description	78
14.7.2	Constructor & Destructor Documentation	79
14.7.2.1	ArnDiscoverConnector	79
14.7.3	Member Function Documentation	79
14.7.3.1	addToDirectHosts	79
14.7.3.2	clearDirectHosts	79
14.7.3.3	clientReadyToConnect	79
14.7.3.4	directHostPrio	80
14.7.3.5	discoverHostPrio	80
14.7.3.6	externalClientConnect	80
14.7.3.7	id	81
14.7.3.8	resolveRefreshTimeout	81
14.7.3.9	service	81
14.7.3.10	setDirectHostPrio	81
14.7.3.11	setDiscoverHostPrio	82
14.7.3.12	setExternalClientConnect	82
14.7.3.13	setResolver	82
14.7.3.14	setResolveRefreshTimeout	83
14.7.3.15	setService	83
14.7.3.16	start	83
14.8	ArnDiscoverInfo Class Reference	84
14.8.1	Detailed Description	85
14.8.2	Constructor & Destructor Documentation	85
14.8.2.1	ArnDiscoverInfo	85
14.8.3	Member Function Documentation	85
14.8.3.1	domain	85
14.8.3.2	groups	85
14.8.3.3	hostIp	86
14.8.3.4	hostIpString	86
14.8.3.5	hostName	86
14.8.3.6	hostPort	86
14.8.3.7	hostPortString	87

14.8.3.8	hostWithInfo	87
14.8.3.9	inProgress	87
14.8.3.10	isError	87
14.8.3.11	properties	88
14.8.3.12	resolveCode	88
14.8.3.13	serviceName	88
14.8.3.14	state	88
14.8.3.15	stopState	89
14.8.3.16	type	89
14.8.3.17	typeString	89
14.8.4	Friends And Related Function Documentation	89
14.8.4.1	ArnDiscoverBrowserB	89
14.9	ArnDiscoverRemote Class Reference	90
14.9.1	Detailed Description	91
14.9.2	Constructor & Destructor Documentation	92
14.9.2.1	ArnDiscoverRemote	92
14.9.3	Member Function Documentation	92
14.9.3.1	clientReadyToConnect	92
14.9.3.2	defaultService	92
14.9.3.3	initialServiceTimeout	92
14.9.3.4	newConnector	93
14.9.3.5	setDefaultService	93
14.9.3.6	setInitialServiceTimeout	93
14.9.3.7	setService	94
14.9.3.8	startUseNewServer	94
14.9.3.9	startUseServer	94
14.10	ArnDiscoverResolver Class Reference	95
14.10.1	Detailed Description	96
14.10.2	Constructor & Destructor Documentation	97
14.10.2.1	ArnDiscoverResolver	97
14.10.3	Member Function Documentation	97
14.10.3.1	defaultService	97
14.10.3.2	resolve	97
14.10.3.3	setDefaultService	98
14.11	ArnError Struct Reference	98
14.11.1	Detailed Description	98
14.11.2	Member Enumeration Documentation	98
14.11.2.1	E	98
14.12	ArnItem Class Reference	99
14.12.1	Detailed Description	102

14.12.2 Constructor & Destructor Documentation	102
14.12.2.1 ArnItem	102
14.12.2.2 ArnItem	102
14.12.2.3 ArnItem	103
14.12.2.4 ~ArnItem	103
14.12.3 Member Function Documentation	103
14.12.3.1 addMode	103
14.12.3.2 arnExport	103
14.12.3.3 arnImport	104
14.12.3.4 arnItemCreated	104
14.12.3.5 arnModeChanged	104
14.12.3.6 changed	104
14.12.3.7 changed	105
14.12.3.8 changed	105
14.12.3.9 changed	105
14.12.3.10changed	105
14.12.3.11changed	105
14.12.3.12changed	105
14.12.3.13getMode	105
14.12.3.14isAutoDestroy	106
14.12.3.15sBiDir	106
14.12.3.16sBiDirMode	106
14.12.3.17sFolder	106
14.12.3.18sIgnoreSameValue	106
14.12.3.19sMaster	107
14.12.3.20sPipeMode	107
14.12.3.21isSaveMode	107
14.12.3.22sTemplate	107
14.12.3.23modeChanged	108
14.12.3.24openFolder	108
14.12.3.25openUuid	108
14.12.3.26openUuidPipe	108
14.12.3.27operator=	109
14.12.3.28operator=	109
14.12.3.29operator=	109
14.12.3.30operator=	109
14.12.3.31operator=	109
14.12.3.32operator=	109
14.12.3.33operator=	109
14.12.3.34setAutoDestroy	109

14.12.3.35	setBiDirMode	109
14.12.3.36	setDelay	110
14.12.3.37	setIgnoreSameValue	110
14.12.3.38	setMaster	110
14.12.3.39	setPipeMode	110
14.12.3.40	setSaveMode	111
14.12.3.41	setTemplate	111
14.12.3.42	setValue	111
14.12.3.43	setValue	111
14.12.3.44	setValue	112
14.12.3.45	setValue	112
14.12.3.46	setValue	112
14.12.3.47	setValue	113
14.12.3.48	setValue	113
14.12.3.49	setValue	113
14.12.3.50	syncMode	113
14.12.3.51	toBool	114
14.12.3.52	toByteArray	114
14.12.3.53	toDouble	114
14.12.3.54	toggleBool	114
14.12.3.55	toInt	114
14.12.3.56	toString	114
14.12.3.57	toVariant	115
14.12.3.58	type	115
14.13	ArnItemB Class Reference	115
14.13.1	Detailed Description	116
14.13.2	Constructor & Destructor Documentation	116
14.13.2.1	ArnItemB	116
14.13.2.2	~ArnItemB	116
14.13.3	Member Function Documentation	117
14.13.3.1	arnLinkDestroyed	117
14.13.3.2	close	117
14.13.3.3	destroyLink	117
14.13.3.4	isOpen	117
14.13.3.5	itemId	117
14.13.3.6	linkId	117
14.13.3.7	name	118
14.13.3.8	open	118
14.13.3.9	path	118
14.13.3.10	reference	119

14.13.3.1	setReference	119
14.14	ArnItemValve Class Reference	119
14.14.1	Detailed Description	120
14.14.2	Constructor & Destructor Documentation	121
14.14.2.1	ArnItemValve	121
14.14.3	Member Function Documentation	121
14.14.3.1	changed	121
14.14.3.2	isAutoDestroy	121
14.14.3.3	isMaster	121
14.14.3.4	isSaveMode	122
14.14.3.5	operator=	122
14.14.3.6	setAutoDestroy	122
14.14.3.7	setMaster	122
14.14.3.8	setSaveMode	122
14.14.3.9	setTarget	123
14.14.3.10	setValue	123
14.14.3.11	switchMode	123
14.14.3.12	toBool	123
14.15	ArnM Class Reference	123
14.15.1	Detailed Description	125
14.15.2	Member Function Documentation	125
14.15.2.1	defaultIgnoreSameValue	125
14.15.2.2	destroyLink	125
14.15.2.3	errorLog	125
14.15.2.4	errorLogSig	125
14.15.2.5	errorSysName	126
14.15.2.6	exist	126
14.15.2.7	info	126
14.15.2.8	instance	126
14.15.2.9	isFolder	126
14.15.2.10	isLeaf	126
14.15.2.11	isMainThread	127
14.15.2.12	isThreadedApp	127
14.15.2.13	items	127
14.15.2.14	loadFromDirRoot	127
14.15.2.15	loadFromFile	127
14.15.2.16	saveToFile	128
14.15.2.17	setConsoleError	128
14.15.2.18	setDefaultIgnoreSameValue	128
14.15.2.19	setSkipLocalSysLoading	128

14.15.2.20	setupErrorlog	129
14.15.2.21	setValue	129
14.15.2.22	setValue	129
14.15.2.23	setValue	129
14.15.2.24	setValue	129
14.15.2.25	setValue	130
14.15.2.26	setValue	130
14.15.2.27	skipLocalSysLoading	130
14.15.2.28	valueByteArray	130
14.15.2.29	valueDouble	131
14.15.2.30	valueInt	131
14.15.2.31	valueString	131
14.15.2.32	valueVariant	131
14.15.3	Friends And Related Function Documentation	132
14.15.3.1	ArnItemB	132
14.16	ArnMonitor Class Reference	132
14.16.1	Detailed Description	133
14.16.2	Constructor & Destructor Documentation	133
14.16.2.1	ArnMonitor	133
14.16.3	Member Function Documentation	133
14.16.3.1	arnChildFound	133
14.16.3.2	arnChildFoundFolder	134
14.16.3.3	arnChildFoundLeaf	134
14.16.3.4	arnItemCreated	134
14.16.3.5	client	134
14.16.3.6	clientId	135
14.16.3.7	foundChildDeleted	135
14.16.3.8	monitorPath	135
14.16.3.9	reference	135
14.16.3.10	reStart	136
14.16.3.11	setClient	136
14.16.3.12	setMonitorPath	136
14.16.3.13	setReference	136
14.16.3.14	start	137
14.16.4	Member Data Documentation	137
14.16.4.1	_arnClient	137
14.16.4.2	_monitorPath	137
14.17	ArnPersist Class Reference	137
14.17.1	Detailed Description	138
14.17.2	Constructor & Destructor Documentation	138

14.17.2.1 ArnPersist	138
14.17.2.2 ~ArnPersist	138
14.17.3 Member Function Documentation	138
14.17.3.1 doArchive	138
14.17.3.2 setArchiveDir	139
14.17.3.3 setMountPoint	139
14.17.3.4 setPersistDir	139
14.17.3.5 setupDataBase	140
14.17.3.6 setVcs	140
14.18 ArnPipe Class Reference	140
14.18.1 Detailed Description	142
14.18.2 Constructor & Destructor Documentation	142
14.18.2.1 ArnPipe	142
14.18.2.2 ArnPipe	143
14.18.2.3 ~ArnPipe	143
14.18.3 Member Function Documentation	143
14.18.3.1 changed	143
14.18.3.2 isAutoDestroy	143
14.18.3.3 isCheckSeq	143
14.18.3.4 isMaster	144
14.18.3.5 isSendSeq	144
14.18.3.6 openUuid	144
14.18.3.7 operator=	144
14.18.3.8 outOfSequence	144
14.18.3.9 setAutoDestroy	145
14.18.3.10 setCheckSeq	145
14.18.3.11 setMaster	145
14.18.3.12 setSendSeq	145
14.18.3.13 setValue	146
14.18.3.14 setValueOverwrite	146
14.19 ArnRpc Class Reference	146
14.19.1 Detailed Description	148
14.19.2 Constructor & Destructor Documentation	149
14.19.2.1 ArnRpc	149
14.19.3 Member Function Documentation	149
14.19.3.1 addSenderSignals	149
14.19.3.2 batchConnect	149
14.19.3.3 batchConnect	150
14.19.3.4 batchConnect	150
14.19.3.5 defaultCall	150

14.19.3.6 heartBeatChanged	151
14.19.3.7 heartBeatReceived	151
14.19.3.8 invoke	151
14.19.3.9 invoke	151
14.19.3.10sHeartBeatOk	152
14.19.3.11mode	152
14.19.3.12open	152
14.19.3.13outOfSequence	152
14.19.3.14pipe	152
14.19.3.15pipeClosed	153
14.19.3.16pipePath	153
14.19.3.17pcSender	153
14.19.3.18pcSender	153
14.19.3.19sendText	153
14.19.3.20setHeartBeatCheck	153
14.19.3.21setHeartBeatSend	154
14.19.3.22setIncludeSender	154
14.19.3.23setMethodPrefix	154
14.19.3.24setMode	154
14.19.3.25setPipe	154
14.19.3.26setReceiver	154
14.19.3.27textReceived	154
14.20ArnSapi Class Reference	155
14.20.1 Detailed Description	156
14.20.2 Constructor & Destructor Documentation	157
14.20.2.1 ArnSapi	157
14.20.3 Member Function Documentation	157
14.20.3.1 batchConnectFrom	157
14.20.3.2 batchConnectTo	157
14.20.3.3 open	158
14.21ArnScript Class Reference	158
14.21.1 Detailed Description	159
14.21.2 Constructor & Destructor Documentation	159
14.21.2.1 ArnScript	159
14.21.3 Member Function Documentation	159
14.21.3.1 engine	159
14.21.3.2 errorLog	159
14.21.3.3 errorText	159
14.21.3.4 evaluate	159
14.21.3.5 evaluateFile	159

14.21.3.6 getClient	159
14.21.3.7 idName	159
14.21.3.8 logUncaughtError	160
14.21.3.9 printFunction	160
14.21.4 Member Data Documentation	160
14.21.4.1 _depOfferProto	160
14.21.4.2 _depProto	160
14.21.4.3 _engine	160
14.21.4.4 _itemProto	160
14.21.4.5 _monitorProto	160
14.22 ArnScriptJob Class Reference	160
14.22.1 Detailed Description	161
14.22.2 Constructor & Destructor Documentation	161
14.22.2.1 ArnScriptJob	161
14.22.3 Member Function Documentation	161
14.22.3.1 errorLog	161
14.22.3.2 quit	161
14.22.3.3 setWatchDogTime	161
14.22.3.4 sigQuit	161
14.22.3.5 yield	161
14.22.4 Property Documentation	161
14.22.4.1 name	161
14.22.4.2 poll	162
14.22.4.3 sleepState	162
14.22.4.4 watchDog	162
14.23 ArnScriptJobControl Class Reference	162
14.23.1 Detailed Description	163
14.23.2 Constructor & Destructor Documentation	163
14.23.2.1 ArnScriptJobControl	163
14.23.3 Member Function Documentation	163
14.23.3.1 addConfig	163
14.23.3.2 addInterface	163
14.23.3.3 addInterfaceList	163
14.23.3.4 config	163
14.23.3.5 doSetupJob	163
14.23.3.6 errorText	163
14.23.3.7 id	163
14.23.3.8 loadScriptFile	163
14.23.3.9 name	163
14.23.3.10 script	164

14.23.3.11scriptChanged	164
14.23.3.12setConfig	164
14.23.3.13setName	164
14.23.3.14setScript	164
14.23.3.15setThreaded	164
14.24ArnScriptJobFactory Class Reference	164
14.24.1 Detailed Description	164
14.24.2 Constructor & Destructor Documentation	165
14.24.2.1 ArnScriptJobFactory	165
14.24.2.2 ~ArnScriptJobFactory	165
14.24.3 Member Function Documentation	165
14.24.3.1 getClient	165
14.24.3.2 installExtension	165
14.24.3.3 setupInterface	165
14.24.3.4 setupJsObj	165
14.25ArnScriptJobs Class Reference	165
14.25.1 Detailed Description	166
14.25.2 Constructor & Destructor Documentation	166
14.25.2.1 ArnScriptJobs	166
14.25.3 Member Function Documentation	166
14.25.3.1 addJob	166
14.25.3.2 setFactory	166
14.25.3.3 start	166
14.26ArnServer Class Reference	166
14.26.1 Detailed Description	167
14.26.2 Constructor & Destructor Documentation	167
14.26.2.1 ArnServer	167
14.26.3 Member Function Documentation	167
14.26.3.1 listenAddress	167
14.26.3.2 port	167
14.26.3.3 start	168
14.27ArnZeroConfB Class Reference	168
14.27.1 Detailed Description	169
14.27.2 Constructor & Destructor Documentation	169
14.27.2.1 ArnZeroConfB	169
14.27.2.2 ~ArnZeroConfB	169
14.27.3 Member Function Documentation	169
14.27.3.1 domain	169
14.27.3.2 fullServiceType	169
14.27.3.3 serviceType	170

14.27.3.4 setDomain	170
14.27.3.5 setServiceType	170
14.27.3.6 setSocketType	170
14.27.3.7 socketType	171
14.27.3.8 state	171
14.28 ArnZeroConfBrowser Class Reference	171
14.28.1 Detailed Description	173
14.28.2 Constructor & Destructor Documentation	174
14.28.2.1 ArnZeroConfBrowser	174
14.28.2.2 ArnZeroConfBrowser	174
14.28.2.3 ~ArnZeroConfBrowser	174
14.28.3 Member Function Documentation	174
14.28.3.1 activeServiceNames	174
14.28.3.2 browse	175
14.28.3.3 browseError	175
14.28.3.4 getNextId	175
14.28.3.5 isBrowsing	175
14.28.3.6 serviceAdded	175
14.28.3.7 serviceChanged	176
14.28.3.8 serviceNameTold	176
14.28.3.9 serviceRemoved	176
14.28.3.10 setSubType	177
14.28.3.11 stopBrowse	177
14.28.3.12 subType	177
14.28.4 Friends And Related Function Documentation	178
14.28.4.1 ArnZeroConfIntern	178
14.29 ArnZeroConfLookup Class Reference	178
14.29.1 Detailed Description	179
14.29.2 Constructor & Destructor Documentation	180
14.29.2.1 ArnZeroConfLookup	180
14.29.2.2 ArnZeroConfLookup	180
14.29.2.3 ~ArnZeroConfLookup	180
14.29.3 Member Function Documentation	180
14.29.3.1 host	180
14.29.3.2 hostAddr	180
14.29.3.3 id	181
14.29.3.4 isForceQtDnsLookup	181
14.29.3.5 lookup	181
14.29.3.6 lookuped	182
14.29.3.7 lookupError	182

14.29.3.8 releaseLookup	182
14.29.3.9 setForceQtDnsLookup	182
14.29.3.10setHost	182
14.29.3.11setId	183
14.29.4 Friends And Related Function Documentation	183
14.29.4.1 ArnZeroConfIntern	183
14.30 ArnZeroConfRegister Class Reference	183
14.30.1 Detailed Description	185
14.30.2 Constructor & Destructor Documentation	186
14.30.2.1 ArnZeroConfRegister	186
14.30.2.2 ArnZeroConfRegister	186
14.30.2.3 ArnZeroConfRegister	186
14.30.2.4 ~ArnZeroConfRegister	186
14.30.3 Member Function Documentation	187
14.30.3.1 addSubType	187
14.30.3.2 currentServiceName	187
14.30.3.3 getTxtRecordMap	187
14.30.3.4 host	188
14.30.3.5 port	188
14.30.3.6 registered	188
14.30.3.7 registerService	188
14.30.3.8 registrationError	189
14.30.3.9 releaseService	189
14.30.3.10serviceName	189
14.30.3.11setHost	189
14.30.3.12setPort	190
14.30.3.13setServiceName	190
14.30.3.14setSubTypes	190
14.30.3.15setTxtRecord	191
14.30.3.16setTxtRecordMap	191
14.30.3.17subTypes	191
14.30.3.18txtRecord	192
14.30.4 Friends And Related Function Documentation	192
14.30.4.1 ArnZeroConfIntern	192
14.31 ArnZeroConfResolve Class Reference	192
14.31.1 Detailed Description	194
14.31.2 Constructor & Destructor Documentation	194
14.31.2.1 ArnZeroConfResolve	194
14.31.2.2 ArnZeroConfResolve	194
14.31.2.3 ArnZeroConfResolve	195

14.31.2.4 ~ArnZeroConfResolve	195
14.31.3 Member Function Documentation	195
14.31.3.1 getTxtRecordMap	195
14.31.3.2 host	195
14.31.3.3 id	196
14.31.3.4 port	196
14.31.3.5 releaseResolve	196
14.31.3.6 resolve	196
14.31.3.7 resolved	197
14.31.3.8 resolveError	197
14.31.3.9 serviceName	197
14.31.3.10 setId	197
14.31.3.11 setServiceName	198
14.31.3.12 txtRecord	198
14.31.4 Friends And Related Function Documentation	198
14.31.4.1 ArnZeroConfIntern	198
14.32 Arn::Coding Struct Reference	198
14.32.1 Detailed Description	198
14.32.2 Member Enumeration Documentation	199
14.32.2.1 E	199
14.33 ArnClient::ConnectStat Struct Reference	199
14.33.1 Detailed Description	199
14.33.2 Member Enumeration Documentation	199
14.33.2.1 E	199
14.34 Arn::DataType Struct Reference	199
14.34.1 Detailed Description	200
14.34.2 Member Enumeration Documentation	200
14.34.2.1 E	200
14.35 ArnZeroConf::Error Struct Reference	200
14.35.1 Detailed Description	200
14.35.2 Member Enumeration Documentation	201
14.35.2.1 E	201
14.36 ArnItemB::ExportCode Struct Reference	201
14.36.1 Detailed Description	201
14.36.2 Member Enumeration Documentation	201
14.36.2.1 E	201
14.37 ArnClient::HostAddrPort Struct Reference	202
14.37.1 Detailed Description	202
14.37.2 Constructor & Destructor Documentation	202
14.37.2.1 HostAddrPort	202

14.37.3 Member Data Documentation	202
14.37.3.1 addr	202
14.37.3.2 port	202
14.38Arnrpc::Invoke Struct Reference	202
14.38.1 Detailed Description	202
14.38.2 Member Enumeration Documentation	203
14.38.2.1 E	203
14.39Arnrpc::LinkFlags Struct Reference	203
14.39.1 Detailed Description	203
14.39.2 Member Enumeration Documentation	203
14.39.2.1 E	203
14.40Arnrpc::Mode Struct Reference	203
14.40.1 Detailed Description	204
14.40.2 Member Enumeration Documentation	204
14.40.2.1 E	204
14.41MQArgument< T > Class Template Reference	204
14.41.1 Detailed Description	205
14.41.2 Constructor & Destructor Documentation	205
14.41.2.1 MQArgument	205
14.42MQGenericArgument Class Reference	206
14.42.1 Detailed Description	206
14.42.2 Constructor & Destructor Documentation	206
14.42.2.1 MQGenericArgument	206
14.42.2.2 MQGenericArgument	206
14.42.3 Member Function Documentation	206
14.42.3.1 label	206
14.43Arnrpc::NameF Struct Reference	207
14.43.1 Detailed Description	207
14.43.2 Member Enumeration Documentation	207
14.43.2.1 E	207
14.44Arnrpc::ObjectMode Struct Reference	207
14.44.1 Detailed Description	207
14.44.2 Member Enumeration Documentation	207
14.44.2.1 E	207
14.45Arnrpc::ObjectSyncMode Struct Reference	208
14.45.1 Detailed Description	208
14.45.2 Member Enumeration Documentation	208
14.45.2.1 E	208
14.46Arnrpc::MethodsParam::Params Struct Reference	208
14.46.1 Detailed Description	209

14.46.2 Member Data Documentation	209
14.46.2.1 allMethodIds	209
14.46.2.2 methodIdsTab	209
14.46.2.3 paramNames	209
14.47 Arn::SameValue Struct Reference	209
14.47.1 Detailed Description	209
14.47.2 Member Enumeration Documentation	209
14.47.2.1 E	209
14.48 ArnZeroConf::State Struct Reference	210
14.48.1 Detailed Description	210
14.48.2 Member Enumeration Documentation	210
14.48.2.1 E	210
14.49 ArnDiscoverAdvertise::State Struct Reference	210
14.49.1 Detailed Description	211
14.49.2 Member Enumeration Documentation	211
14.49.2.1 E	211
14.50 ArnDiscoverInfo::State Struct Reference	211
14.50.1 Detailed Description	211
14.50.2 Member Enumeration Documentation	211
14.50.2.1 E	211
14.51 ArnError::StdCode Struct Reference	212
14.51.1 Detailed Description	212
14.51.2 Member Enumeration Documentation	212
14.51.2.1 E	212
14.52 ArnItemValve::SwitchMode Struct Reference	212
14.52.1 Detailed Description	213
14.52.2 Member Enumeration Documentation	213
14.52.2.1 E	213
14.53 ArnServer::Type Struct Reference	213
14.53.1 Detailed Description	213
14.53.2 Member Enumeration Documentation	213
14.53.2.1 E	213
14.54 ArnDiscover::Type Struct Reference	213
14.54.1 Detailed Description	214
14.54.2 Member Enumeration Documentation	214
14.54.2.1 E	214
14.55 ArnScriptJobs::Type Struct Reference	214
14.55.1 Detailed Description	214
14.55.2 Member Enumeration Documentation	214
14.55.2.1 E	214

14.56Arn::XStringMap Class Reference	215
14.56.1 Detailed Description	216
14.56.2 Constructor & Destructor Documentation	217
14.56.2.1 XStringMap	217
14.56.2.2 XStringMap	217
14.56.2.3 XStringMap	217
14.56.2.4 ~XStringMap	217
14.56.3 Member Function Documentation	217
14.56.3.1 add	217
14.56.3.2 add	217
14.56.3.3 add	217
14.56.3.4 add	217
14.56.3.5 add	217
14.56.3.6 add	217
14.56.3.7 add	217
14.56.3.8 add	218
14.56.3.9 add	218
14.56.3.10add	218
14.56.3.11append	218
14.56.3.12append	218
14.56.3.13append	218
14.56.3.14append	218
14.56.3.15append	218
14.56.3.16append	218
14.56.3.17append	218
14.56.3.18append	218
14.56.3.19append	218
14.56.3.20append	219
14.56.3.21clear	219
14.56.3.22fromXString	219
14.56.3.23indexOf	219
14.56.3.24indexOf	219
14.56.3.25indexOf	219
14.56.3.26indexOfValue	219
14.56.3.27indexOfValue	219
14.56.3.28key	219
14.56.3.29key	219
14.56.3.30key	219
14.56.3.31keyRef	219
14.56.3.32keys	220

14.56.3.33	keyString	220
14.56.3.34	keyString	220
14.56.3.35	maxEnumOf	220
14.56.3.36	operator+=	220
14.56.3.37	operator+=	220
14.56.3.38	remove	220
14.56.3.39	remove	220
14.56.3.40	remove	220
14.56.3.41	remove	220
14.56.3.42	set	220
14.56.3.43	set	220
14.56.3.44	set	221
14.56.3.45	set	221
14.56.3.46	set	221
14.56.3.47	set	221
14.56.3.48	set	221
14.56.3.49	setEmptyKeysToValue	221
14.56.3.50	size	221
14.56.3.51	squeeze	221
14.56.3.52	stringCode	221
14.56.3.53	stringDecode	221
14.56.3.54	toVariantMap	221
14.56.3.55	toXString	221
14.56.3.56	value	222
14.56.3.57	value	222
14.56.3.58	value	222
14.56.3.59	value	222
14.56.3.60	value	222
14.56.3.61	valueRef	222
14.56.3.62	values	222
14.56.3.63	valueString	222
14.56.3.64	valueString	222
14.56.3.65	valueString	222
14.56.3.66	valueString	222
14.56.3.67	valueString	222
15	File Documentation	225
15.1	doc/Description.md File Reference	225
15.2	doc/HelpIndex.txt File Reference	225
15.3	doc/Install.md File Reference	225

15.4 doc/Internals.md File Reference	225
15.5 doc/ToDo.md File Reference	225
15.6 examples/Examples.txt File Reference	225
15.7 README.md File Reference	225
15.8 src/Arn.cpp File Reference	225
15.9 src/ArnClient.cpp File Reference	227
15.10src/ArnDepend.cpp File Reference	227
15.10.1 Variable Documentation	228
15.10.1.1 ArnDependPath	228
15.11src/ArnDiscover.cpp File Reference	228
15.12src/ArnDiscoverConnect.cpp File Reference	228
15.13src/ArnDiscoverRemote.cpp File Reference	229
15.14src/ArnInc/Arn.hpp File Reference	229
15.14.1 Macro Definition Documentation	231
15.14.1.1 DATASTREAM_VER	231
15.15src/ArnInc/ArnClient.hpp File Reference	231
15.16src/ArnInc/ArnDepend.hpp File Reference	232
15.17src/ArnInc/ArnDiscover.hpp File Reference	233
15.18src/ArnInc/ArnDiscoverConnect.hpp File Reference	235
15.19src/ArnInc/ArnDiscoverRemote.hpp File Reference	235
15.20src/ArnInc/ArnError.hpp File Reference	236
15.21src/ArnInc/ArnItem.hpp File Reference	237
15.21.1 Function Documentation	238
15.21.1.1 operator<<	238
15.22src/ArnInc/ArnItemB.hpp File Reference	239
15.23src/ArnInc/ArnItemValve.hpp File Reference	239
15.24src/ArnInc/ArnLib.hpp File Reference	240
15.25src/ArnInc/ArnLib_global.hpp File Reference	241
15.25.1 Macro Definition Documentation	242
15.25.1.1 ARNLIBSHARED_EXPORT	242
15.26src/ArnInc/ArnLinkHandle.hpp File Reference	242
15.27src/ArnInc/ArnM.hpp File Reference	243
15.28src/ArnInc/ArnMonitor.hpp File Reference	243
15.29src/ArnInc/ArnPersist.hpp File Reference	244
15.30src/ArnInc/ArnPersistSapi.hpp File Reference	245
15.31src/ArnInc/ArnPipe.hpp File Reference	246
15.32src/ArnInc/ArnRpc.hpp File Reference	247
15.32.1 Macro Definition Documentation	248
15.32.1.1 MQ_ARG	248
15.32.1.2 no_queue	249

15.33src/ArnInc/ArnSapi.hpp File Reference	249
15.33.1 Macro Definition Documentation	250
15.33.1.1 MQ_PUBLIC_ACCESS	250
15.34src/ArnInc/ArnScript.hpp File Reference	250
15.35src/ArnInc/ArnScriptJob.hpp File Reference	251
15.36src/ArnInc/ArnScriptJobs.hpp File Reference	252
15.37src/ArnInc/ArnServer.hpp File Reference	253
15.38src/ArnInc/ArnZeroConf.hpp File Reference	254
15.38.1 Typedef Documentation	255
15.38.1.1 DNSServiceRef	255
15.39src/ArnInc/MQFlags.hpp File Reference	256
15.39.1 Macro Definition Documentation	256
15.39.1.1 MQ_DECLARE_ENUM	256
15.39.1.2 MQ_DECLARE_FLAGS	257
15.39.1.3 MQ_DECLARE_OPERATORS_FOR_FLAGS	257
15.40src/ArnInc/XStringMap.hpp File Reference	257
15.41src/ArnItem.cpp File Reference	258
15.41.1 Function Documentation	258
15.41.1.1 operator<<	258
15.42src/ArnItemB.cpp File Reference	258
15.43src/ArnItemNet.cpp File Reference	259
15.44src/ArnItemNet.hpp File Reference	259
15.45src/ArnItemValve.cpp File Reference	260
15.46src/ArnLib.cpp File Reference	260
15.47src/ArnLink.cpp File Reference	261
15.48src/ArnLink.hpp File Reference	262
15.49src/ArnLinkHandle.cpp File Reference	263
15.50src/ArnM.cpp File Reference	263
15.51src/ArnMonitor.cpp File Reference	264
15.52src/ArnPersist.cpp File Reference	264
15.52.1 Variable Documentation	265
15.52.1.1 arnDbSaveVer	265
15.53src/ArnPipe.cpp File Reference	265
15.54src/ArnRpc.cpp File Reference	265
15.54.1 Macro Definition Documentation	266
15.54.1.1 RPC_STORAGE_NAME	266
15.55src/ArnSapi.cpp File Reference	266
15.56src/ArnScript.cpp File Reference	266
15.57src/ArnScriptJob.cpp File Reference	267
15.57.1 Variable Documentation	267

15.57.1.1 EventQuit	267
15.58src/ArnScriptJobs.cpp File Reference	267
15.59src/ArnServer.cpp File Reference	268
15.60src/ArnSync.cpp File Reference	268
15.61src/ArnSync.hpp File Reference	269
15.61.1 Macro Definition Documentation	270
15.61.1.1 ARNRECNAME	270
15.62src/ArnXStringMap.cpp File Reference	270
15.63src/ArnZeroConf.cpp File Reference	270
16 Example Documentation	273
16.1 ArnDemoChat/main.cpp	273
16.2 ArnDemoChat/MainWindow.cpp	273
16.3 ArnDemoChat/MainWindow.hpp	275
16.4 ArnDemoChatServer/ChatSapi.hpp	275
16.5 ArnDemoChatServer/main.cpp	276
16.6 ArnDemoChatServer/MainWindow.cpp	276
16.7 ArnDemoChatServer/MainWindow.hpp	278

Chapter 1

README

Copyright (C) 2010-2014 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 Registry & AD:** 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).

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.

Persistent storage

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

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.
- Hot swap of changed Java Script in jobstack.

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*.
- Simple integration together with a custom GUI for browsing, etc.
- Optional internal DNS_SD/mDNS routines for no dependency to any extra library.

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 ArnBrowser.

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 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*. This implies that `ArnItems` can benefit from *mode* settings before being opened.

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\(\)](#).

2.1.2 Local path

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".

2.1.3 Naming conventions

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

- First level folder empty, e.g. "//MyGlobalFolder/Date/value", is a global path and is [shared](#) to server and clients.
- First level folder starts with "@", e.g. "@SomeServer/MyFolder/Date/value", is a shared path and is [shared](#) to a 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".
 - **set** allowed values and conversion to a more descriptive form, e.g. "0=Off 1=On".
 - **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.

2.2 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.

2.3 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.

2.3.1 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\(\)](#).

2.3.2 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.

2.4 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. It's just to 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](#).

2.4.1 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.

2.5 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.

```

2.5.1 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](#).

2.6 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.

2.6.1 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.

2.6.2 RPC and SAPI communication format

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

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

The following RPC data types are available:

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

Also generic RPC data types can be formed as:

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

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

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

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

Alternatives in positional argument format:

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

- Argument names are optional and only for human debugging.

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

Alternatives in named argument format:

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

- Only Argument names are used.
- Any order of arguments can be used.
- Extra arguments are discarded.
- If to 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 to 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".

2.7 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 dependences to the rest of the ArnLib.

ARN ZeroConfig can use a built in implementation of Apple (R) *mDns* / *DNS_SD* that has no further dependences 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.

A 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*.

2.7.1 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".

2.7.2 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.

2.7.3 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](#).

2.8 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.

2.9 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\(\)](#).

2.10 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.

A) Unix

```
qmake
make
make install
```

The easiest way of installing this library, is to let it be placed in a standard location for librarys 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
```

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.

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.

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++
```

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.

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

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

4.2 ArnMonitor

- Monitor starts its actual connection job when monitorPath is set.
- Monitor (at client-side) 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), connections are made to send arn-events when new childs are created, and present childs are directly sent as arn-event.

4.3 Destroy

- Command arrives with a netId.
- Corresponding ItemNet is disabled (set as defunct).
- All link-leaves for the ItemNet:s tree is set as retired and each leaf is emitting a retired signal.
- The retired signal is handled by each connected Item. Each Item is 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 signal is sent.
- The signal is handled by doZerRefLink(), in Main thread. It will set the link ref counter to -1 to mark the link as fully de-referenced. The link and parent (and grand parants ...) are deleted if they don't have any children and ref = -1 and they are retired.
- When the ItemNet is sending the linkDestroyed signal, it will be deleted from sync map and all queues. Finally a destroy command is sent with its netId, to spread the destruction to server and other clients.

Chapter 5

ArnLib Todo

Major

- Script support for Sapi.
- QML with "files" as ArnObject and other integration with [Arn](#).
- Convert to d-pointer for making binary compatible library in the future.
- Unit tests

Minor

- In Signal Slot use "const Type&".
- Optimize data transfer with minimal copying.
- Optimize memory consumption with pointers to different data in ArnLink.
- Add tranfer classes for copying values.
- Add more examples
- [ArnClient](#) stored centrally with an id. Also accessible by the id.

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( &_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 [ArnMonitor::setMonitorPath](#) (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	43
ArnDiscover	51
ArnZeroConf	51

Chapter 10

Class Index

10.1 Class Hierarchy

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

ArnClient	53
ArnDepend	58
ArnDependOffer	61
ArnDiscoverAdvertise	62
ArnDiscoverRemote	90
ArnDiscoverBrowserB	72
ArnDiscoverBrowser	69
ArnDiscoverResolver	95
ArnDiscoverConnector	77
ArnDiscoverInfo	84
ArnError	98
ArnItemB	115
ArnItem	99
ArnItemValve	119
ArnPipe	140
ArnM	123
ArnMonitor	132
ArnPersist	137
ArnRpc	146
ArnSapi	155
ArnScript	158
ArnScriptJob	160
ArnScriptJobControl	162
ArnScriptJobFactory	164
ArnScriptJobs	165
ArnServer	166
ArnZeroConfB	168
ArnZeroConfBrowser	171
ArnZeroConfLookup	178
ArnZeroConfRegister	183
ArnZeroConfResolve	192
Arn::Coding	198
ArnClient::ConnectStat	199
Arn::DataType	199
ArnZeroConf::Error	200
ArnItemB::ExportCode	201

ArnClient::HostAddrPort	202
ArnRpc::Invoke	202
Arn::LinkFlags	203
ArnRpc::Mode	203
MQGenericArgument	206
MQArgument< T >	204
Arn::NameF	207
Arn::ObjectMode	207
Arn::ObjectSyncMode	208
ArnRpc::MethodsParam::Params	208
Arn::SameValue	209
ArnZeroConf::State	210
ArnDiscoverAdvertise::State	210
ArnDiscoverInfo::State	211
ArnError::StdCode	212
ArnItemValve::SwitchMode	212
ArnServer::Type	213
ArnDiscover::Type	213
ArnScriptJobs::Type	214
Arn::XStringMap	215

Chapter 11

Class Index

11.1 Class List

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

ArnClient	Class for connecting to an Arn Server	53
ArnDepend	Class for setting up dependencis to needed services	58
ArnDependOffer	Class for advertising that a <i>service</i> is available	61
ArnDiscoverAdvertise	Advertise an Arn service	62
ArnDiscoverBrowser	Browsing for Arn services	69
ArnDiscoverBrowserB	Browse() and resolve() together, may never be used to the same instance	72
ArnDiscoverConnector	An automatic client discover connector	77
ArnDiscoverInfo	Class for holding current discover info of one service	84
ArnDiscoverRemote	Discover with remote setting	90
ArnDiscoverResolver	Resolv an Arn service	95
ArnError	98
ArnItem	Handle for an Arn Data Object	99
ArnItemB	Base class handle for an Arn Data Object	115
ArnItemValve	Valve for controlling stream to/from an ArnItemB	119
ArnM	123
ArnMonitor	A client remote monitor to detect changes at server	132
ArnPersist	137
ArnPipe	ArnItem specialized as a pipe	140
ArnRpc	Remote Procedure Call	146
ArnSapi	Service API	155
ArnScript	158

ArnScriptJob	160
ArnScriptJobControl	
Is thread-safe (except doSetupJob)	162
ArnScriptJobFactory	
Must be thread-safe as subclassed	164
ArnScriptJobs	165
ArnServer	
Class for making an <i>Arn</i> Server	166
ArnZeroConfB	
Base class for Zero Config	168
ArnZeroConfBrowser	
Browsing for ZeroConfig services	171
ArnZeroConfLookup	
Lookup a host	178
ArnZeroConfRegister	
Registering a ZeroConfig service	183
ArnZeroConfResolve	
Resolv a ZeroConfig service	192
Arn::Coding	198
ArnClient::ConnectStat	199
Arn::DataType	
Data type of an <i>Arn Data Object</i>	199
ArnZeroConf::Error	
Errors of ZeroConfig, other values are defined in dns_sd.h	200
ArnItemB::ExportCode	
Code used in blob for arnExport() and arnImport()	201
ArnClient::HostAddrPort	202
ArnRpc::Invoke	202
Arn::LinkFlags	
Link flags when accessing an <i>Arn Data Object</i>	203
ArnRpc::Mode	203
MQArgument< T >	
Similar to QArgument but with added argument label (parameter name)	204
MQGenericArgument	
Similar to QGenericArgument but with added argument label (parameter name)	206
Arn::NameF	207
Arn::ObjectMode	
General global mode of an <i>Arn Data Object</i>	207
Arn::ObjectSyncMode	
The client session sync mode of an <i>Arn Data Object</i>	208
ArnRpc::MethodsParam::Params	208
Arn::SameValue	
Action when assigning same value to an <i>ArnItem</i>	209
ArnZeroConf::State	
States of ZeroConfig, limited valid for each <i>ArnZeroConfB</i> subclass / These values must be synced with: <i>ArnDiscover::State</i>	210
ArnDiscoverAdvertise::State	
States of DiscoverAdvertise / These values must be synced with: <i>ArnZeroConf::State</i>	210
ArnDiscoverInfo::State	
State of <i>Arn</i> discover browse data. Can be tested by relative order	211
ArnError::StdCode	212
ArnItemValve::SwitchMode	212
ArnServer::Type	213
ArnDiscover::Type	
Types of <i>Arn</i> discover advertise	213
ArnScriptJobs::Type	214
Arn::XStringMap	
Container class with string representation for serialized data	215

Chapter 12

File Index

12.1 File List

Here is a list of all files with brief descriptions:

src/Arn.cpp	225
src/ArnClient.cpp	227
src/ArnDepend.cpp	227
src/ArnDiscover.cpp	228
src/ArnDiscoverConnect.cpp	228
src/ArnDiscoverRemote.cpp	229
src/ArnItem.cpp	258
src/ArnItemB.cpp	258
src/ArnItemNet.cpp	259
src/ArnItemNet.hpp	259
src/ArnItemValve.cpp	260
src/ArnLib.cpp	260
src/ArnLink.cpp	261
src/ArnLink.hpp	262
src/ArnLinkHandle.cpp	263
src/ArnM.cpp	263
src/ArnMonitor.cpp	264
src/ArnPersist.cpp	264
src/ArnPipe.cpp	265
src/ArnRpc.cpp	265
src/ArnSapi.cpp	266
src/ArnScript.cpp	266
src/ArnScriptJob.cpp	267
src/ArnScriptJobs.cpp	267
src/ArnServer.cpp	268
src/ArnSync.cpp	268
src/ArnSync.hpp	269
src/ArnXStringMap.cpp	270
src/ArnZeroConf.cpp	270
src/ArnInc/Arn.hpp	229
src/ArnInc/ArnClient.hpp	231
src/ArnInc/ArnDepend.hpp	232
src/ArnInc/ArnDiscover.hpp	233
src/ArnInc/ArnDiscoverConnect.hpp	235
src/ArnInc/ArnDiscoverRemote.hpp	235
src/ArnInc/ArnError.hpp	236
src/ArnInc/ArnItem.hpp	237
src/ArnInc/ArnItemB.hpp	239

src/ArnInc/ArnItemValve.hpp	239
src/ArnInc/ArnLib.hpp	240
src/ArnInc/ArnLib_global.hpp	241
src/ArnInc/ArnLinkHandle.hpp	242
src/ArnInc/ArnM.hpp	243
src/ArnInc/ArnMonitor.hpp	243
src/ArnInc/ArnPersist.hpp	244
src/ArnInc/ArnPersistSapi.hpp	245
src/ArnInc/ArnPipe.hpp	246
src/ArnInc/ArnRpc.hpp	247
src/ArnInc/ArnSapi.hpp	249
src/ArnInc/ArnScript.hpp	250
src/ArnInc/ArnScriptJob.hpp	251
src/ArnInc/ArnScriptJobs.hpp	252
src/ArnInc/ArnServer.hpp	253
src/ArnInc/ArnZeroConf.hpp	254
src/ArnInc/MQFlags.hpp	256
src/ArnInc/XStringMap.hpp	257

Chapter 13

Namespace Documentation

13.1 Arn Namespace Reference

Classes

- struct [SameValue](#)
Action when assigning same value to an [ArnItem](#).
- struct [DataType](#)
Data type of an [Arn](#) Data Object
- struct [ObjectMode](#)
General global mode of an [Arn](#) Data Object
- struct [ObjectSyncMode](#)
The client session sync mode of an [Arn](#) Data Object
- struct [LinkFlags](#)
Link flags when accessing an [Arn](#) Data Object
- struct [NameF](#)
- struct [Coding](#)
- 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 `twinPath` (const QString &path)
 - Get the bidirectional twin to a given path*
- QString `providerPath` (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 `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.*

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/"
- 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 `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 130 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 107 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 93 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 47 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 141 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 75 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 210 of file Arn.cpp.

13.1.1.8 bool Arn::isFolderPath (const QString & *path*)

Test if *path* is a *folder path*

Parameters

<i>in</i>	<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 191 of file Arn.cpp.

13.1.1.9 bool Arn::isProviderPath (const QString & *path*)

Test if *path* is a *provider path*

[About Bidirectional Arn Data Objects](#)

Parameters

<i>in</i>	<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 197 of file Arn.cpp.

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

The last part of a *path*

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

Parameters

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

Returns

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

Definition at line 83 of file Arn.cpp.

13.1.1.11 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

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

Returns

The *HostWithInfo* string

See also

[hostFromHostWithInfo\(\)](#)

Note

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

Definition at line 203 of file Arn.cpp.

13.1.1.12 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 121 of file Arn.cpp.

13.1.1.13 `QString Arn::providerPath (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 185 of file Arn.cpp.

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

Get the bidirectional twin to a given *path*

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

Parameters

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

Returns

The twin *path*

See also

[Bidirectional Arn Data Objects](#)

Definition at line 176 of file Arn.cpp.

13.1.2 Variable Documentation

13.1.2.1 `bool Arn::debugDepend = false`

Definition at line 45 of file ArnLib.cpp.

13.1.2.2 `bool Arn::debugDiscover = false`

Definition at line 46 of file ArnLib.cpp.

13.1.2.3 `bool Arn::debugLinkDestroy = false`

Definition at line 39 of file ArnLib.cpp.

13.1.2.4 `bool Arn::debugLinkRef = false`

Definition at line 38 of file ArnLib.cpp.

13.1.2.5 `bool Arn::debugMDNS = false`

Definition at line 48 of file ArnLib.cpp.

13.1.2.6 `bool Arn::debugMonitor = false`

Definition at line 42 of file ArnLib.cpp.

13.1.2.7 `bool Arn::debugMonitorTest = false`

Definition at line 43 of file ArnLib.cpp.

13.1.2.8 `bool Arn::debugReclnOut = false`

Definition at line 40 of file ArnLib.cpp.

13.1.2.9 `bool Arn::debugRPC = false`

Definition at line 44 of file ArnLib.cpp.

13.1.2.10 `bool Arn::debugShareObj = false`

Definition at line 41 of file ArnLib.cpp.

13.1.2.11 `bool Arn::debugThreading = false`

Definition at line 37 of file ArnLib.cpp.

13.1.2.12 `bool Arn::debugZeroConf = false`

Definition at line 47 of file ArnLib.cpp.

13.1.2.13 `const quint16 Arn::defaultTcpPort = 2022`

Definition at line 43 of file Arn.hpp.

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

Definition at line 42 of file Arn.cpp.

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

Definition at line 44 of file Arn.cpp.

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

Definition at line 43 of file Arn.cpp.

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

Definition at line 40 of file Arn.cpp.

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

Definition at line 41 of file Arn.cpp.

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

Definition at line 51 of file ArnLib.cpp.

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

Definition at line 52 of file ArnLib.cpp.

13.1.2.21 `bool Arn::warningMDNS = false`

Definition at line 49 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 ArnClient Class Reference

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

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

Classes

- struct [ConnectStat](#)
- struct [HostAddrPort](#)
- struct **MountPointSlot**

Public Types

- typedef QList< [HostAddrPort](#) > [HostList](#)

Signals

- void [tcpError](#) (QString errorText, QAbstractSocket::SocketError socketError)
- void [tcpConnected](#) (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.

Public Member Functions

- [ArnClient](#) (QObject *parent=0)
- 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

- 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.

14.1.1 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 71 of file ArnClient.hpp.

14.1.2 Member Typedef Documentation

14.1.2.1 typedef QList<HostAddrPort> ArnClient::HostList

Definition at line 101 of file ArnClient.hpp.

14.1.3 Constructor & Destructor Documentation

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

Definition at line 43 of file ArnClient.cpp.

14.1.4 Member Function Documentation

14.1.4.1 `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 155 of file ArnClient.cpp.

14.1.4.2 `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 105 of file ArnClient.cpp.

14.1.4.3 `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 89 of file ArnClient.cpp.

14.1.4.4 `void ArnClient::clearArnList (int prioFilter = -1)`

Clear the [Arn](#) connection list.

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

Parameters

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

See also

[addToArnList\(\)](#)

Definition at line 68 of file ArnClient.cpp.

14.1.4.5 `void ArnClient::connectionStatusChanged (int status, int curPrio)` `[signal]`

Signal emitted when the connection status is changed.

Parameters

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

See also

`curPrio()`

14.1.4.6 `ArnClient::ConnectStat ArnClient::connectStatus () const`

Return the [Arn](#) connection status.

Return values

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

Definition at line 140 of file ArnClient.cpp.

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

Connect to an [Arn](#) Server

Parameters

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

See also

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

Definition at line 131 of file ArnClient.cpp.

14.1.4.8 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

Definition at line 122 of file ArnClient.cpp.

14.1.4.9 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 192 of file ArnClient.cpp.

14.1.4.10 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 210 of file ArnClient.cpp.

14.1.4.11 bool ArnClient::setMountPoint (const QString & path)

Set the sharing tree path.

For compatability, 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 146 of file ArnClient.cpp.

14.1.4.12 void ArnClient::tcpConnected (QString *arnHost*, quint16 *port*) [signal]

Signal emitted when the tcp connection is successfull.

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, e.g. 2022.

14.1.4.13 void ArnClient::tcpDisconnected () [signal]

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

14.1.4.14 void ArnClient::tcpError (QString *errorText*, QAbstractSocket::SocketError *socketError*) [signal]

Signal emitted when a connection tcp error occur.

Parameters

<i>in</i>	<i>errorText</i>	is the human readable description of the error.
<i>in</i>	<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 \(2.2.0\)](#)
- [src/ArnClient.cpp \(2.2.0\)](#)

14.2 ArnDepend Class Reference

Class for setting up dependencis to needed services.

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

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` (QString serviceName, int stateId=-1)
Add a dependency for a service
- void `add` (QString serviceName, QString stateName)
Add a dependency for a service
- void `setMonitorName` (QString name)
Set an optional monitor name for debugging.
- void `startMonitor` ()
Starting the dependency monitor.

14.2.1 Detailed Description

Class for setting up dependencis to needed services.

The services can be both system types available by internal `Arn`, and custom application types. The system types have a service name starting with "\$".

This is typically used when an application needs a service to continue. When using persistent values, a client will need to know when they have been synced from the server. Then it's convenient to setup a dependency for the system service "\$Persist".

When all dependent services are available, the `completed()` signal is emitted.

Example usage

```
// In class declare
ArnDepend* _arnDepend;

// In class code
_arnDepend = new ArnDepend( this);
_arnDepend->setMonitorName("MyApp_Monitor"); // Optional for
debug
_arnDepend->add("$Persist");
_arnDepend->add("MyService");
_arnDepend->startMonitor();
connect( _arnDepend, SIGNAL(completed()), this, SLOT(arnDependOk()) );
```

Definition at line 128 of file ArnDepend.hpp.

14.2.2 Member Typedef Documentation

14.2.2.1 typedef ArnDependSlot ArnDepend::DepSlot

Definition at line 132 of file ArnDepend.hpp.

14.2.3 Constructor & Destructor Documentation

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

Definition at line 126 of file ArnDepend.cpp.

14.2.3.2 ArnDepend::~~ArnDepend ()

Definition at line 138 of file ArnDepend.cpp.

14.2.4 Member Function Documentation

14.2.4.1 void ArnDepend::add (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 172 of file ArnDepend.cpp.

14.2.4.2 void ArnDepend::add (QString *serviceName*, 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 164 of file ArnDepend.cpp.

14.2.4.3 void ArnDepend::completed () [signal]

Signal emitted when all dependent services are available.

14.2.4.4 void ArnDepend::setMonitorName (QString *name*)

Set an optional monitor name for debugging.

Parameters

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

Definition at line 180 of file ArnDepend.cpp.

14.2.4.5 void ArnDepend::startMonitor ()

Starting the dependency monitor.

Definition at line 186 of file ArnDepend.cpp.

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

- [src/ArnInc/ArnDepend.hpp \(2.2.0\)](#)
- [src/ArnDepend.cpp \(2.2.0\)](#)

14.3 ArnDependOffer Class Reference

Class for advertising that a *service* is available.

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

Public Member Functions

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

14.3.1 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 60 of file ArnDepend.hpp.

14.3.2 Constructor & Destructor Documentation

14.3.2.1 [ArnDependOffer::ArnDependOffer \(QObject * parent = 0 \)](#) [explicit]

Definition at line 44 of file ArnDepend.cpp.

14.3.3 Member Function Documentation

14.3.3.1 void [ArnDependOffer::advertise \(QString serviceName \)](#)

Advertise an available *service*

Parameters

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

Definition at line 50 of file ArnDepend.cpp.

14.3.3.2 void ArnDependOffer::setStateId (int *id*)

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

The *state* starts of by 0 as default.

Parameters

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

Definition at line 82 of file ArnDepend.cpp.

14.3.3.3 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

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

Definition at line 70 of file ArnDepend.cpp.

14.3.3.4 int ArnDependOffer::stateId () const

Returns

The *state* id number.

See also

[setStateId\(\)](#)

Definition at line 88 of file ArnDepend.cpp.

14.3.3.5 QString ArnDependOffer::stateName () const

Returns

The logic *state* name, e.g. the default "Start"

See also

[setStateName\(\)](#)

Definition at line 76 of file ArnDepend.cpp.

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

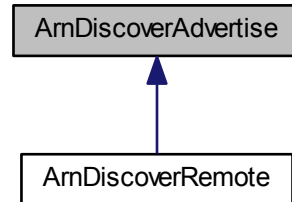
- [src/ArnInc/ArnDepend.hpp \(2.2.0\)](#)
- [src/ArnDepend.cpp \(2.2.0\)](#)

14.4 ArnDiscoverAdvertise Class Reference

Advertise an [Arn](#) service.

```
#include <ArnDiscover.hpp>
```

Inheritance diagram for ArnDiscoverAdvertise:



Classes

- struct [State](#)

States of DiscoverAdvertise / These values must be synced with: [ArnZeroConf::State](#).

Public Slots

- virtual void [setService](#) (QString [service](#))

Set the service name.

Signals

- void [serviceChanged](#) (QString serviceName)

Indicate successfull advertise of service.

- void [serviceChangeError](#) (int code)

Indicate unsuccessful advertise of service.

Public Member Functions

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

- 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, 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.

14.4.1 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 617 of file ArnDiscover.hpp.

14.4.2 Constructor & Destructor Documentation

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

Definition at line 585 of file ArnDiscover.cpp.

14.4.3 Member Function Documentation

14.4.3.1 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 660 of file ArnDiscover.cpp.

14.4.3.2 void ArnDiscoverAdvertise::addGroup (const QString & group)

Add a service discover group.

Parameters

<i>in</i>	<i>group</i>	e.g. "Any Group ID"
-----------	--------------	---------------------

Note

Groups must be set before calling [advertiseService\(\)](#).

See also

[setGroups\(\)](#)

Definition at line 718 of file ArnDiscover.cpp.

14.4.3.3 void ArnDiscoverAdvertise::advertiseService (ArnDiscover::Type discoverType, 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

<i>in</i>	<i>discoverType</i>	is used for discover filtering
<i>in</i>	<i>serviceName</i>	is requested name e.g. "My House Registry"
<i>in</i>	<i>port</i>	is the port of the service, -1 gives default Arn port number
<i>in</i>	<i>hostName</i>	is the host doing the service, empty gives this advertising host

See also

[setService\(\)](#)
[serviceChanged\(\)](#)
[serviceChangeError\(\)](#)

Definition at line 593 of file ArnDiscover.cpp.

14.4.3.4 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 678 of file ArnDiscover.cpp.

14.4.3.5 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 648 of file ArnDiscover.cpp.

14.4.3.6 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 706 of file ArnDiscover.cpp.

14.4.3.7 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 672 of file ArnDiscover.cpp.

14.4.3.8 void ArnDiscoverAdvertise::serviceChanged (QString *serviceName*) [signal]

Indicate successfull advertise of service.

Parameters

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

See also

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

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

Indicate unsuccessfull advertise of service.

Parameters

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

See also

[advertiseService\(\)](#)

14.4.3.10 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

in	<i>custom-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 654 of file ArnDiscover.cpp.

14.4.3.11 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

<i>in</i>	<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 712 of file ArnDiscover.cpp.

14.4.3.12 `void ArnDiscoverAdvertise::setService (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 guaranted 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

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

See also

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

Reimplemented in [ArnDiscoverRemote](#).

Definition at line 690 of file ArnDiscover.cpp.

14.4.3.13 `ArnDiscoverAdvertise::State ArnDiscoverAdvertise::state () const`

Returns the state for this Advertise.

Returns

current state

See also

[State](#)

Definition at line 684 of file ArnDiscover.cpp.

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

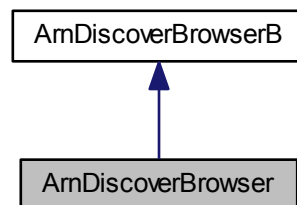
- [src/ArnInc/ArnDiscover.hpp \(2.2.0\)](#)
- [src/ArnDiscover.cpp \(2.2.0\)](#)

14.5 ArnDiscoverBrowser Class Reference

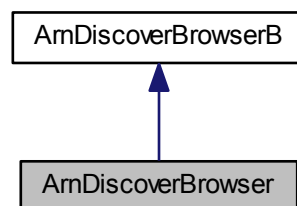
Browsing for [Arn](#) services.

```
#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](#) (QString group)
Set service discover filter using group name.

Additional Inherited Members

14.5.1 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 471 of file ArnDiscover.hpp.

14.5.2 Constructor & Destructor Documentation

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

Definition at line 165 of file ArnDiscover.cpp.

14.5.3 Member Function Documentation

14.5.3.1 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 510 of file ArnDiscover.hpp.

14.5.3.2 `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 481 of file ArnDiscover.hpp.

14.5.3.3 `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

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

See also

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

Definition at line 490 of file ArnDiscover.hpp.

14.5.3.4 `void ArnDiscoverBrowser::setFilter (QString group)` `[inline]`

Set service discover filter using group name.

If passing empty group, this is taken as subtype (filter) disabled. When subtype (filter) is enabled, only services that have the same group is discovered.

Parameters

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

See also

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

Definition at line 500 of file ArnDiscover.hpp.

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

Stop browsing.

See also

[browse\(\)](#)

Definition at line 516 of file ArnDiscover.hpp.

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

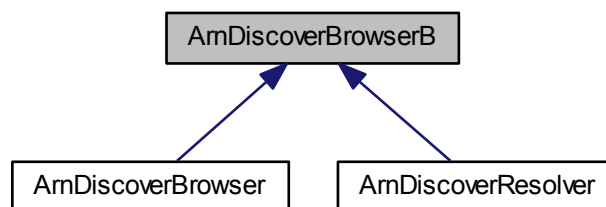
- [src/ArnInc/ArnDiscover.hpp \(2.2.0\)](#)
- [src/ArnDiscover.cpp \(2.2.0\)](#)

14.6 ArnDiscoverBrowserB Class Reference

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

```
#include <ArnDiscover.hpp>
```

Inheritance diagram for ArnDiscoverBrowserB:



Signals

- void [serviceAdded](#) (int index, 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)
- 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](#) (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.

14.6.1 Detailed Description

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

Definition at line 220 of file ArnDiscover.hpp.

14.6.2 Constructor & Destructor Documentation

14.6.2.1 ArnDiscoverBrowserB::ArnDiscoverBrowserB (QObject * *parent* = 0) [explicit]

Definition at line 201 of file ArnDiscover.cpp.

14.6.3 Member Function Documentation

14.6.3.1 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 293 of file ArnDiscover.cpp.

14.6.3.2 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

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

See also

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

Definition at line 305 of file ArnDiscover.cpp.

14.6.3.3 `int ArnDiscoverBrowserB::IdToIndex (int id)`

Return the discover service index by its id.

The index for a service info is only 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 253 of file ArnDiscover.cpp.

14.6.3.4 `int ArnDiscoverBrowserB::indexTold (int index)`

Return the discover service id by its index.

The index for a service info is only 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 245 of file ArnDiscover.cpp.

14.6.3.5 `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 232 of file ArnDiscover.cpp.

14.6.3.6 `const ArnDiscoverInfo & ArnDiscoverBrowserB::infoByIndex (int index)`

Return the discover service info by its index.

The index for a service info is only valid 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 222 of file ArnDiscover.cpp.

14.6.3.7 `const ArnDiscoverInfo & ArnDiscoverBrowserB::infoByName (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 239 of file ArnDiscover.cpp.

14.6.3.8 void ArnDiscoverBrowserB::infoUpdated (int *index*, ArnDiscoverInfo::State *state*) [signal]

Indicate service has been updated.

Parameters

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

See also

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

14.6.3.9 void ArnDiscoverBrowserB::serviceAdded (int *index*, QString *name*) [signal]

Indicate service has been added (discovered)

The service has been added to a list sorted by ascending service names. The index is a reference to this sorted list.

Parameters

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

See also

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

14.6.3.10 int ArnDiscoverBrowserB::serviceCount () const

Return the number of active discover services.

Returns

number of services

Definition at line 216 of file ArnDiscover.cpp.

14.6.3.11 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

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

Returns

selected service discover id

See also

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

Definition at line 259 of file ArnDiscover.cpp.

14.6.3.12 void ArnDiscoverBrowserB::serviceRemoved (int *index*) [signal]

Indicate service has been removed.

Parameters

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

See also

[serviceAdded\(\)](#)

14.6.3.13 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

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

See also

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

Definition at line 299 of file ArnDiscover.cpp.

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

- [src/ArnInc/ArnDiscover.hpp \(2.2.0\)](#)
- [src/ArnDiscover.cpp \(2.2.0\)](#)

14.7 ArnDiscoverConnector Class Reference

An automatic client discover connector.

```
#include <ArnDiscoverConnect.hpp>
```

Public Slots

- void [setService](#) (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)
- 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.

14.7.1 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 74 of file ArnDiscoverConnect.hpp.

14.7.2 Constructor & Destructor Documentation**14.7.2.1 ArnDiscoverConnector::ArnDiscoverConnector (ArnClient & client, const QString & id)**

Definition at line 43 of file ArnDiscoverConnect.cpp.

14.7.3 Member Function Documentation**14.7.3.1 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 72 of file ArnDiscoverConnect.cpp.

14.7.3.2 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 66 of file ArnDiscoverConnect.cpp.

14.7.3.3 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\(\)](#)

14.7.3.4 int ArnDiscoverConnector::directHostPrio () const

Return the priority for *direct hosts*

Returns

direct host priority

See also

[setDirectHostPrio\(\)](#)

Definition at line 128 of file ArnDiscoverConnect.cpp.

14.7.3.5 int ArnDiscoverConnector::discoverHostPrio () const

Return the priority for *discovered hosts*

Returns

discoverHostPrio is the priority.

See also

[setDiscoverHostPrio\(\)](#)

Definition at line 116 of file ArnDiscoverConnect.cpp.

14.7.3.6 bool ArnDiscoverConnector::externalClientConnect () const

Return the *external client connect* mode.

Returns

true when active.

See also

[setExternalClientConnect\(\)](#)

Definition at line 140 of file ArnDiscoverConnect.cpp.

14.7.3.7 QString ArnDiscoverConnector::id () const

Return the identifier for this connector.

Returns

the identifier, e.g "WeatherData-XYZ"

See also

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

Definition at line 98 of file ArnDiscoverConnect.cpp.

14.7.3.8 int ArnDiscoverConnector::resolveRefreshTimeout () const

Return the resolv refresh period.

Returns

resolve refresh timeout in seconds.

See also

[setResolveRefreshTimeout\(\)](#)

Definition at line 104 of file ArnDiscoverConnect.cpp.

14.7.3.9 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 152 of file ArnDiscoverConnect.cpp.

14.7.3.10 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 134 of file ArnDiscoverConnect.cpp.

14.7.3.11 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 122 of file ArnDiscoverConnect.cpp.

14.7.3.12 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>externalClientConnect</i>	true to activate.
----	------------------------------	-------------------

See also

[externalClientConnect\(\)](#)

Definition at line 146 of file ArnDiscoverConnect.cpp.

14.7.3.13 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 78 of file ArnDiscoverConnect.cpp.

14.7.3.14 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 intervall 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 110 of file ArnDiscoverConnect.cpp.

14.7.3.15 void ArnDiscoverConnector::setService (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 158 of file ArnDiscoverConnect.cpp.

14.7.3.16 void ArnDiscoverConnector::start ()

Start connector.

See also

[addToDirectHosts\(\)](#)
[setResolver\(\)](#)

Examples:

[ArnDemoChat/MainWindow.cpp](#).

Definition at line 167 of file [ArnDiscoverConnect.cpp](#).

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

- [src/ArnInc/ArnDiscoverConnect.hpp \(2.2.0\)](#)
- [src/ArnDiscoverConnect.cpp \(2.2.0\)](#)

14.8 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](#) ()
- 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](#)

14.8.1 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 68 of file ArnDiscover.hpp.

14.8.2 Constructor & Destructor Documentation

14.8.2.1 ArnDiscoverInfo::ArnDiscoverInfo ()

Definition at line 44 of file ArnDiscover.cpp.

14.8.3 Member Function Documentation

14.8.3.1 QString ArnDiscoverInfo::domain () const

Return the domain for this service.

Returns

domain, e.g. "local."

Definition at line 95 of file ArnDiscover.cpp.

14.8.3.2 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 83 of file ArnDiscover.cpp.

14.8.3.3 QHostAddress ArnDiscoverInfo::hostIp () const

Return the host ip-address for this service.

Returns

host ip-address

Definition at line 113 of file ArnDiscover.cpp.

14.8.3.4 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 145 of file ArnDiscover.cpp.

14.8.3.5 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 101 of file ArnDiscover.cpp.

14.8.3.6 quint16 ArnDiscoverInfo::hostPort () const

Return the port for this service.

Returns

port

See also

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

Definition at line 107 of file ArnDiscover.cpp.

14.8.3.7 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 139 of file ArnDiscover.cpp.

14.8.3.8 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 151 of file ArnDiscover.cpp.

14.8.3.9 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 53 of file ArnDiscover.cpp.

14.8.3.10 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 59 of file ArnDiscover.cpp.

14.8.3.11 `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 119 of file ArnDiscover.cpp.

14.8.3.12 `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 157 of file ArnDiscover.cpp.

14.8.3.13 `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 89 of file ArnDiscover.cpp.

14.8.3.14 `ArnDiscoverInfo::State ArnDiscoverInfo::state () const`

Return the state for this service.

Returns

state

See also

[State](#)

Definition at line 65 of file ArnDiscover.cpp.

14.8.3.15 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 71 of file ArnDiscover.cpp.

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

Return the discover type for this service.

Returns

discover type

See also

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

Definition at line 77 of file ArnDiscover.cpp.

14.8.3.17 QString ArnDiscoverInfo::typeString () const

Return the printable type for this service.

Returns

type, e.g. "Client"

Definition at line 125 of file ArnDiscover.cpp.

14.8.4 Friends And Related Function Documentation

14.8.4.1 friend class ArnDiscoverBrowserB [friend]

Definition at line 70 of file ArnDiscover.hpp.

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

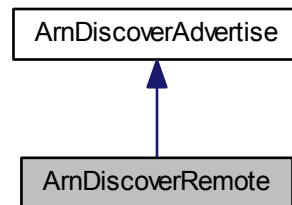
- [src/ArnInc/ArnDiscover.hpp \(2.2.0\)](#)
- [src/ArnDiscover.cpp \(2.2.0\)](#)

14.9 ArnDiscoverRemote Class Reference

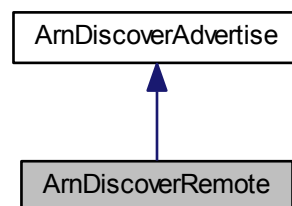
Discover with remote setting.

```
#include <ArnDiscoverRemote.hpp>
```

Inheritance diagram for ArnDiscoverRemote:



Collaboration diagram for ArnDiscoverRemote:



Public Slots

- virtual void [setService](#) (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)
- 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.

14.9.1 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 93 of file [ArnDiscoverRemote.hpp](#).

14.9.2 Constructor & Destructor Documentation

14.9.2.1 [ArnDiscoverRemote::ArnDiscoverRemote \(QObject * *parent* = 0 \)](#) `[explicit]`

Definition at line 46 of file [ArnDiscoverRemote.cpp](#).

14.9.3 Member Function Documentation

14.9.3.1 [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\(\)](#)

14.9.3.2 [QString ArnDiscoverRemote::defaultService \(\)](#) const

Return the default service name.

Returns

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

See also

[setDefaultService\(\)](#)

Definition at line 195 of file [ArnDiscoverRemote.cpp](#).

14.9.3.3 [int ArnDiscoverRemote::initialServiceTimeout \(\)](#) const

Return the time for initial timeout processing.

Returns

time in seconds

See also

[setInitialServiceTimeout\(\)](#)

Definition at line 208 of file ArnDiscoverRemote.cpp.

14.9.3.4 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 108 of file ArnDiscoverRemote.cpp.

14.9.3.5 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 201 of file ArnDiscoverRemote.cpp.

14.9.3.6 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 214 of file ArnDiscoverRemote.cpp.

14.9.3.7 void ArnDiscoverRemote::setService (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\(\)](#)

Reimplemented from [ArnDiscoverAdvertise](#).

Definition at line 180 of file ArnDiscoverRemote.cpp.

14.9.3.8 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 97 of file ArnDiscoverRemote.cpp.

14.9.3.9 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 57 of file ArnDiscoverRemote.cpp.

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

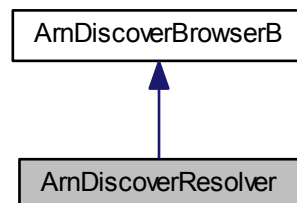
- [src/ArnInc/ArnDiscoverRemote.hpp \(2.2.0\)](#)
- [src/ArnDiscoverRemote.cpp \(2.2.0\)](#)

14.10 ArnDiscoverResolver Class Reference

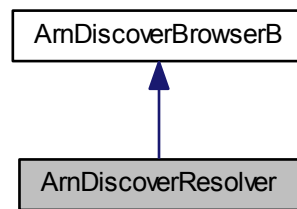
Resolv an [Arn](#) service.

```
#include <ArnDiscover.hpp>
```

Inheritance diagram for ArnDiscoverResolver:



Collaboration diagram for ArnDiscoverResolver:



Public Slots

- `int resolve (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

14.10.1 Detailed Description

Resolv an [Arn](#) service.

[About Arn Discover](#)

Example usage

```

// In class declare
ArnDiscoverResolver* _resolver;

// In class code
_resolver = new ArnDiscoverResolver( this);
connect( _resolver, SIGNAL(infoUpdated(int,
    ArnDiscoverInfo::State)),
    this, SLOT(doClientResolvChanged(int,ArnDiscoverInfo::State
    )));
_resolver->resolve("My service");

void XXX::doClientResolvChanged( int index, ArnDiscoverInfo::State
    state)
{
    const ArnDiscoverInfo& info = _resolver->infoByIndex
        ( index);

    if (state == state.HostIp) {
        qDebug() << "Resolved service:" << info.serviceName()
            << " into host:" << info.hostWithInfo();
    }
    else if (info.isError()) {
        qDebug() << "Error resolving service:" << info.serviceName()
  
```

```

        << " code:" << info.resolveCode();
    }
}

```

Examples:

[ArnDemoChat/MainWindow.cpp](#).

Definition at line 550 of file ArnDiscover.hpp.

14.10.2 Constructor & Destructor Documentation

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

Definition at line 173 of file ArnDiscover.cpp.

14.10.3 Member Function Documentation

14.10.3.1 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 186 of file ArnDiscover.cpp.

14.10.3.2 int ArnDiscoverResolver::resolve (QString *serviceName*, bool *forceUpdate* = true) [slot]

Resolve a specific service name.

Only the specified service will be resolved, but there can be many ongoing resolves by calling this method multiple times with different service names. The [infoUpdated\(\)](#) signal will always be emitted when calling this method. The signal can also be emitted multiple times later regarding the same service.

Parameters

in	<i>serviceName</i>	is the service to be resolved
in	<i>forceUpdate</i>	when true, a new resolve is always done, otherwise a service name that already is resolved will not be resolved again.

Returns

index to service info

See also

[indexTold\(\)](#)
[infoUpdated\(\)](#)

Definition at line 180 of file ArnDiscover.cpp.

14.10.3.3 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

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

See also

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

Definition at line 192 of file ArnDiscover.cpp.

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

- [src/ArnInc/ArnDiscover.hpp \(2.2.0\)](#)
- [src/ArnDiscover.cpp \(2.2.0\)](#)

14.11 ArnError Struct Reference

```
#include <ArnError.hpp>
```

Classes

- struct [StdCode](#)

Public Types

- enum [E](#) {
[Ok](#) = 0, [Info](#) = StdCode::Info, [Warning](#) = StdCode::Warning, [Undef](#) = StdCode::Err_Undef,
[CreateError](#) = StdCode::Err_Custom, [NotFound](#), [NotOpen](#), [AlreadyExist](#),
[AlreadyOpen](#), [Retired](#), [NotMainThread](#), [FolderNotOpen](#),
[ItemNotOpen](#), [ItemNotSet](#), [ConnectionError](#), [RecUnknown](#),
[ScriptError](#), [RpcInvokeError](#), [RpcReceiveError](#), [Err_N](#) }

14.11.1 Detailed Description

Definition at line 38 of file ArnError.hpp.

14.11.2 Member Enumeration Documentation

14.11.2.1 enum ArnError::E

Enumerator:

Ok
Info

Warning
Undef
CreateError
NotFound
NotOpen
AlreadyExist
AlreadyOpen
Retired
NotMainThread
FolderNotOpen
ItemNotOpen
ItemNotSet
ConnectionError
RecUnknown
ScriptError
RpcInvokeError
RpcReceiveError
Err_N

Definition at line 51 of file ArnError.hpp.

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

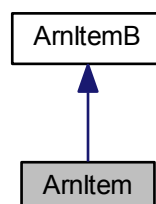
- [src/ArnInc/ArnError.hpp \(2.2.0\)](#)

14.12 ArnItem Class Reference

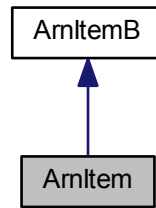
Handle for an *Arn Data Object*.

```
#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 a double 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` (QString value)
- void `changed` (QByteArray value)
- void `changed` (QVariant value)
- void `modeChanged` (`Arn::ObjectMode` mode)
Signal emitted when mode in *Arn* Data Object is changed.
- void `arnItemCreated` (QString path)
Signal emitted when an *Arn* Data Object is created in the tree below.
- void `arnModeChanged` (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 [isBiDir](#) () 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 [setDelay](#) (int delay)
Set delay of data changed signal.
- void [arnImport](#) (const QByteArray &data, int ignoreSame=[Arn::SameValue::DefaultAction](#))
Import data to an [Arn](#) Data Object
- QByteArray [arnExport](#) () const
- int [toInt](#) () const
- double [toDouble](#) () const
- bool [toBool](#) () const

- QString [toString](#) () const
- QByteArray [toByteArray](#) () const
- QVariant [toVariant](#) () const
- [ArnItem](#) & [operator=](#) (const [ArnItem](#) &other)
- [ArnItem](#) & [operator=](#) (int other)
- [ArnItem](#) & [operator=](#) (double other)
- [ArnItem](#) & [operator=](#) (const QString &other)
- [ArnItem](#) & [operator=](#) (const QByteArray &other)
- [ArnItem](#) & [operator=](#) (const QVariant &other)
- [ArnItem](#) & [operator=](#) (const char *other)
- void [setValue](#) (const [ArnItem](#) &other, int ignoreSame=[Arn::SameValue::DefaultAction](#))

Assign the value of an other [ArnItem](#) to an [Arn](#) Data Object

14.12.1 Detailed Description

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

About Arn Data Object

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/Main-Window.hpp](#).

Definition at line 70 of file [ArnItem.hpp](#).

14.12.2 Constructor & Destructor Documentation

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

Standard constructor of a closed handle.

Parameters

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

Definition at line 73 of file [ArnItem.cpp](#).

14.12.2.2 [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. <code>"//Measure/Water/Level/value"</code>
in	<i>parent</i>	

See also

[open\(\)](#)

Definition at line 80 of file ArnItem.cpp.

14.12.2.3 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. <code>"//Measure/Water/Level/value"</code>
in	<i>parent</i>	

Definition at line 88 of file ArnItem.cpp.

14.12.2.4 ArnItem::~~ArnItem () [virtual]

Definition at line 400 of file ArnItem.cpp.

14.12.3 Member Function Documentation

14.12.3.1 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 156 of file ArnItem.hpp.

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

Returns

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

See also

[arnImport\(\)](#)

Definition at line 288 of file ArnItem.hpp.

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

See also

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

Definition at line 282 of file ArnItem.hpp.

14.12.3.4 void ArnItem::arnItemCreated (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	path	to the created Arn Data Object
----	------	--

14.12.3.5 void ArnItem::arnModeChanged (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	path	to the <i>general mode</i> changing Arn Data Object
in	linkId	for the <i>general mode</i> changing Arn Data Object
in	mode	is the new <i>general mode</i>

See also

[linkId\(\)](#)
[Modes](#)

14.12.3.6 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\(\)](#)

14.12.3.7 void ArnItem::changed (int *value*) [signal]

See also

[changed\(\)](#)

14.12.3.8 void ArnItem::changed (double *value*) [signal]

See also

[changed\(\)](#)

14.12.3.9 void ArnItem::changed (bool *value*) [signal]

See also

[changed\(\)](#)

14.12.3.10 void ArnItem::changed (QString *value*) [signal]

See also

[changed\(\)](#)

14.12.3.11 void ArnItem::changed (QByteArray *value*) [signal]

See also

[changed\(\)](#)

14.12.3.12 void ArnItem::changed (QVariant *value*) [signal]

See also

[changed\(\)](#)

14.12.3.13 Arn::ObjectMode ArnItem::getMode () const [inline]

Returns

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

See also

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

Definition at line 163 of file ArnItem.hpp.

14.12.3.14 `bool ArnItem::isAutoDestroy () const` `[inline]`

Return values

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

See also

[setAutoDestroy\(\)](#)

Definition at line 262 of file ArnItem.hpp.

14.12.3.15 `bool ArnItem::isBiDir () const` `[inline]`

Return values

<i>true</i>	if this ArnItem is bi-directional
-------------	---

See also

[setBiDirMode\(\)](#)

[Modes](#)

Definition at line 126 of file ArnItem.hpp.

14.12.3.16 `bool ArnItem::isBiDirMode () const` `[inline]`

Return values

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

See also

[setBiDirMode\(\)](#)

[Modes](#)

[Bidirectional Arn Data Objects](#)

Definition at line 201 of file ArnItem.hpp.

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

Return values

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

Definition at line 119 of file ArnItem.hpp.

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

Return values

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

See also

[setIgnoreSameValue\(\)](#)

Definition at line 144 of file ArnItem.hpp.

14.12.3.19 `bool ArnItem::isMaster () const` `[inline]`

Return values

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

See also

[setMaster\(\)](#)

[Modes](#)

Definition at line 249 of file ArnItem.hpp.

14.12.3.20 `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 217 of file ArnItem.hpp.

14.12.3.21 `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 234 of file ArnItem.hpp.

14.12.3.22 `bool ArnItem::isTemplate () const`

Return values

<i>true</i>	if this is a template
-------------	-----------------------

See also

[setTemplate\(\)](#)

Definition at line 123 of file ArnItem.cpp.

14.12.3.23 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

in	<i>mode</i>	is the new <i>general mode</i>
----	-------------	--------------------------------

See also

[Modes](#)

14.12.3.24 bool ArnItem::openFolder (const QString & *path*) [inline]

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

Parameters

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

Return values

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

Definition at line 114 of file ArnItem.hpp.

14.12.3.25 bool ArnItem::openUuid (const QString & *path*) [inline]

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

Parameters

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

Return values

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

Definition at line 100 of file ArnItem.hpp.

14.12.3.26 bool ArnItem::openUuidPipe (const QString & *path*) [inline]

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

Parameters

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

Return values

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

Definition at line 107 of file ArnItem.hpp.

14.12.3.27 ArnItem & ArnItem::operator= (const ArnItem & *other*)

Definition at line 139 of file ArnItem.cpp.

14.12.3.28 ArnItem & ArnItem::operator= (int *other*)

Definition at line 146 of file ArnItem.cpp.

14.12.3.29 ArnItem & ArnItem::operator= (double *other*)

Definition at line 153 of file ArnItem.cpp.

14.12.3.30 ArnItem & ArnItem::operator= (const QString & *other*)

Definition at line 160 of file ArnItem.cpp.

14.12.3.31 ArnItem & ArnItem::operator= (const QByteArray & *other*)

Definition at line 167 of file ArnItem.cpp.

14.12.3.32 ArnItem & ArnItem::operator= (const QVariant & *other*)

Definition at line 181 of file ArnItem.cpp.

14.12.3.33 ArnItem & ArnItem::operator= (const char * *other*)

Definition at line 174 of file ArnItem.cpp.

14.12.3.34 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 256 of file ArnItem.hpp.

14.12.3.35 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 193 of file ArnItem.hpp.

14.12.3.36 `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

<i>in</i>	<i>delay</i>	in ms.
-----------	--------------	--------

Definition at line 129 of file ArnItem.cpp.

14.12.3.37 `void ArnItem::setIgnoreSameValue (bool isIgnore = true) [inline]`

Set skipping assignment 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 138 of file ArnItem.hpp.

14.12.3.38 `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 242 of file ArnItem.hpp.

14.12.3.39 `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 209 of file ArnItem.hpp.

14.12.3.40 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 226 of file ArnItem.hpp.

14.12.3.41 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 116 of file ArnItem.cpp.

14.12.3.42 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 334 of file ArnItem.hpp.

14.12.3.43 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 343 of file ArnItem.hpp.

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

Assign a *double* 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 351 of file ArnItem.hpp.

14.12.3.45 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 359 of file ArnItem.hpp.

14.12.3.46 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 367 of file ArnItem.hpp.

14.12.3.47 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 375 of file ArnItem.hpp.

14.12.3.48 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 383 of file ArnItem.hpp.

14.12.3.49 void ArnItem::setValue (const char * *value*, int *ignoreSame* = Arn::SameValue::DefaultAction) [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 188 of file ArnItem.cpp.

14.12.3.50 Arn::ObjectSyncMode ArnItem::syncMode () const [inline]

Returns

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

See also

[addSyncMode\(\)](#)

[Modes](#)

Definition at line 170 of file ArnItem.hpp.

```
14.12.3.51 bool ArnItem::toBool ( ) const [inline]
```

Returns

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

Definition at line 303 of file ArnItem.hpp.

```
14.12.3.52 QByteArray ArnItem::toByteArray ( ) const [inline]
```

Returns

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

Definition at line 313 of file ArnItem.hpp.

```
14.12.3.53 double ArnItem::toDouble ( ) const [inline]
```

Returns

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

Definition at line 298 of file ArnItem.hpp.

```
14.12.3.54 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 194 of file ArnItem.cpp.

```
14.12.3.55 int ArnItem::toInt ( ) const [inline]
```

Returns

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

Definition at line 293 of file ArnItem.hpp.

```
14.12.3.56 QString ArnItem::toString ( ) const [inline]
```

Returns

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

Definition at line 308 of file ArnItem.hpp.

14.12.3.57 `QVariant ArnItem::toVariant () const [inline]`

Returns

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

Definition at line 318 of file `ArnItem.hpp`.

14.12.3.58 `Arn::DataType ArnItem::type () const [inline]`

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

Returns

The type stored

Definition at line 132 of file `ArnItem.hpp`.

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

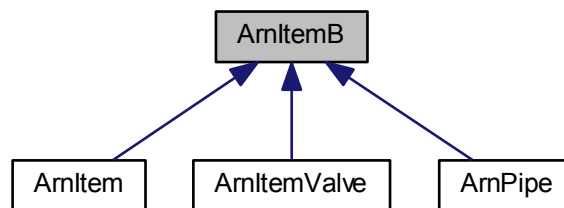
- `src/ArnInc/ArnItem.hpp (2.2.0)`
- `src/ArnItem.cpp (2.2.0)`

14.13 ArnItemB Class Reference

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

```
#include <ArnItemB.hpp>
```

Inheritance diagram for ArnItemB:



Classes

- struct [ExportCode](#)
Code used in blob for `arnExport()` and `arnImport()`

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
- void [close](#) ()
Close the handle.
- void [destroyLink](#) ()
Destroy the [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

14.13.1 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 63 of file ArnItemB.hpp.

14.13.2 Constructor & Destructor Documentation

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

Standard constructor of a closed handle.

Parameters

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

Definition at line 64 of file ArnItemB.cpp.

14.13.2.2 ArnItemB::~ArnItemB () [virtual]

Definition at line 964 of file ArnItemB.cpp.

14.13.3 Member Function Documentation

14.13.3.1 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\(\)](#)

14.13.3.2 void ArnItemB::close ()

Close the handle.

Definition at line 139 of file ArnItemB.cpp.

14.13.3.3 void ArnItemB::destroyLink ()

Destroy the [Arn Data Object](#)

The link ([Arn Data Object](#)) will be removed locally, from server and all connected clients.

Definition at line 150 of file ArnItemB.cpp.

14.13.3.4 bool ArnItemB::isOpen () const

State of the handle.

Return values

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

Definition at line 156 of file ArnItemB.cpp.

14.13.3.5 uint ArnItemB::itemId () const [inline]

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 141 of file ArnItemB.hpp.

14.13.3.6 uint ArnItemB::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 186 of file *ArnItemB.cpp*.

14.13.3.7 QString ArnItemB::name (Arn::NameF nameF) const

Name of the *Arn Data Object*

Parameters

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

Returns

The object name

Definition at line 409 of file *ArnItemB.cpp*.

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

Open a handle to an *Arn Data Object*

Parameters

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

Return values

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

Definition at line 94 of file *ArnItemB.cpp*.

14.13.3.9 QString ArnItemB::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 401 of file *ArnItemB.cpp*.

14.13.3.10 `void* ArnItemB::reference () const` `[inline]`

Get the stored external reference.

Returns

The associated external reference

See also

[setReference\(\)](#)

Definition at line 133 of file ArnItemB.hpp.

14.13.3.11 `void ArnItemB::setReference (void * reference)` `[inline]`

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 127 of file ArnItemB.hpp.

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

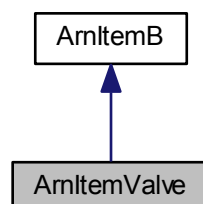
- [src/ArnInc/ArnItemB.hpp \(2.2.0\)](#)
- [src/ArnItemB.cpp \(2.2.0\)](#)

14.14 ArnItemValve Class Reference

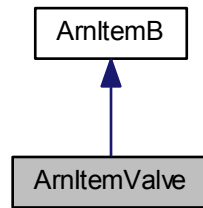
Valve for controlling stream to/from an [ArnItemB](#).

```
#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)

14.14.1 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 76 of file `ArnItemValve.hpp`.

14.14.2 Constructor & Destructor Documentation

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

Definition at line 35 of file `ArnItemValve.cpp`.

14.14.3 Member Function Documentation

14.14.3.1 `void ArnItemValve::changed (int value)` `[signal]`

Signals emitted when data in [Arn Data Object](#) is changed.

14.14.3.2 `bool ArnItemValve::isAutoDestroy () const` `[inline]`

Return values

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

See also

[setAutoDestroy\(\)](#)

Definition at line 140 of file `ArnItemValve.hpp`.

14.14.3.3 `bool ArnItemValve::isMaster () const` `[inline]`

Return values

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

See also

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

Definition at line 127 of file ArnItemValve.hpp.

14.14.3.4 **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 112 of file ArnItemValve.hpp.

14.14.3.5 **ArnItemValve & ArnItemValve::operator= (bool value)**

Definition at line 68 of file ArnItemValve.cpp.

14.14.3.6 **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 134 of file ArnItemValve.hpp.

14.14.3.7 **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 120 of file ArnItemValve.hpp.

14.14.3.8 **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 104 of file ArnItemValve.hpp.

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

Definition at line 43 of file ArnItemValve.cpp.

14.14.3.10 `void ArnItemValve::setValue (bool value) [slot]`

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

Parameters

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

Definition at line 75 of file ArnItemValve.cpp.

14.14.3.11 `ArnItemValve::SwitchMode ArnItemValve::switchMode () const`

Definition at line 53 of file ArnItemValve.cpp.

14.14.3.12 `bool ArnItemValve::toBool () const`

Returns

state of this valve 1 = Enabled selected stream(s)

Definition at line 59 of file ArnItemValve.cpp.

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

- [src/ArnInc/ArnItemValve.hpp \(2.2.0\)](#)
- [src/ArnItemValve.cpp \(2.2.0\)](#)

14.15 ArnM Class Reference

```
#include <ArnM.hpp>
```

Public Slots

- static void [destroyLink](#) (const QString &path)
Destroy the [Arn Data Object](#) at path
- static void [setupErrorlog](#) (QObject *errLog)

Signals

- void [errorLogSig](#) (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 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, double value)
Assign a double 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)
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.

Friends

- class [ArnItemB](#)

14.15.1 Detailed Description

[Arn](#) main class

[About Arn Data Object](#)

This singleton class is the main reference to the Active Registry Network.

Definition at line 104 of file ArnM.hpp.

14.15.2 Member Function Documentation

14.15.2.1 bool [ArnM::defaultIgnoreSameValue](#) () [static]

Return values

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

See also

[setDefaultIgnoreSameValue\(\)](#)

Definition at line 867 of file ArnM.cpp.

14.15.2.2 void [ArnM::destroyLink](#) (const QString & *path*) [static],[slot]

Destroy the [Arn](#) Data Object at *path*

The link ([Arn](#) Data Object) will be removed locally, from server and all connected clients.

Parameters

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

Threaded version of [destroyLink](#)

Definition at line 645 of file ArnM.cpp.

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

Definition at line 763 of file ArnM.cpp.

14.15.2.4 void [ArnM::errorLogSig](#) (QString *errText*, uint *errCode*, void * *reference*) [signal]

14.15.2.5 QString ArnM::errorSysName () [static]

Definition at line 723 of file ArnM.cpp.

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

Parameters

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

Return values

true	if Arn Data Object exist at path
------	--

Definition at line 261 of file ArnM.cpp.

14.15.2.7 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 729 of file ArnM.cpp.

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

Definition at line 847 of file ArnM.cpp.

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

Parameters

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

Return values

true	if Arn Data Object at path is a folder
------	--

Definition at line 272 of file ArnM.cpp.

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

Parameters

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

Return values

true	if Arn Data Object at path is a leaf (non folder)
------	---

Definition at line 283 of file ArnM.cpp.

14.15.2.11 `bool ArnM::isMainThread () [static]`

Return values

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

Definition at line 239 of file ArnM.cpp.

14.15.2.12 `bool ArnM::isThreadedApp () [static]`

Return values

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

Definition at line 255 of file ArnM.cpp.

14.15.2.13 `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 179 of file ArnM.cpp.

14.15.2.14 `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 374 of file ArnM.cpp.

14.15.2.15 `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 356 of file ArnM.cpp.

14.15.2.16 `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 383 of file ArnM.cpp.

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

Definition at line 855 of file ArnM.cpp.

14.15.2.18 `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 861 of file ArnM.cpp.

14.15.2.19 `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>skipLocalSys-Loading</i>	
----	-----------------------------	--

Note

Must be called before entering the Qt event loop
 Check the rules for [Local path](#)

See also

[skipLocalSysLoading\(\)](#)

Definition at line 879 of file ArnM.cpp.

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

Definition at line 735 of file ArnM.cpp.

14.15.2.21 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 306 of file ArnM.cpp.

14.15.2.22 void ArnM::setValue (const QString & *path*, double *value*) [static]

Assign a *double* to an [Arn Data Object](#) at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned

Definition at line 317 of file ArnM.cpp.

14.15.2.23 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 295 of file ArnM.cpp.

14.15.2.24 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 328 of file ArnM.cpp.

14.15.2.25 void ArnM::setValue (const QString & *path*, const QVariant & *value*) [static]

Assign a *QVariant* to an [Arn Data Object](#) at *path*

Parameters

in	<i>path</i>	
in	<i>value</i>	to be assigned

Definition at line 339 of file ArnM.cpp.

14.15.2.26 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 350 of file ArnM.cpp.

14.15.2.27 bool ArnM::skipLocalSysLoading () const

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

Returns

mode skipLocalSysLoading

See also

[setSkipLocalSysLoading\(\)](#)

Definition at line 873 of file ArnM.cpp.

14.15.2.28 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 147 of file ArnM.cpp.

14.15.2.29 `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 125 of file ArnM.cpp.

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

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

Parameters

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

Returns

The *Arn Data Object* as an *integer*

Definition at line 114 of file ArnM.cpp.

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

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

Parameters

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

Returns

The *Arn Data Object* as a *QString*

Definition at line 136 of file ArnM.cpp.

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

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

Parameters

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

Returns

The *Arn Data Object* as a *QVariant*

Definition at line 158 of file ArnM.cpp.

14.15.3 Friends And Related Function Documentation

14.15.3.1 friend class ArnItemB [friend]

Definition at line 107 of file ArnM.hpp.

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

- src/ArnInc/ArnM.hpp (2.2.0)
- src/ArnM.cpp (2.2.0)

14.16 ArnMonitor Class Reference

A client remote monitor to detect changes at server.

```
#include <ArnMonitor.hpp>
```

Public Slots

- void [foundChildDeleted](#) (QString path)
Help telling the monitor about deletion of a previous found child.

Signals

- void [arnItemCreated](#) (QString path)
Signal emitted when an Arn Data Object is created in the tree below.
- void [arnChildFound](#) (QString path)
Signal emitted for present and newly created childs in the monitor folder.
- void [arnChildFoundFolder](#) (QString path)
Signal emitted for present and newly created folder childs in the monitor folder.
- void [arnChildFoundLeaf](#) (QString path)
Signal emitted for present and newly created leaf childs in the monitor folder.

Public Member Functions

- [ArnMonitor](#) (QObject *parent=0)
- void [setClient](#) ([ArnClient](#) *client, QString id=QString())
Set the client to be used.
- QString [clientId](#) () const
Get the id name of the used client
- [ArnClient](#) * [client](#) () const
Get the used client
- void [setMonitorPath](#) (QString path, [ArnClient](#) *client=0)
Set the path to be monitored.
- bool [start](#) (const QString &path, [ArnClient](#) *client)
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.

Protected Attributes

- QPointer< [ArnClient](#) > [_arnClient](#)
- QString [_monitorPath](#)

14.16.1 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 64 of file ArnMonitor.hpp.

14.16.2 Constructor & Destructor Documentation

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

Definition at line 39 of file ArnMonitor.cpp.

14.16.3 Member Function Documentation

14.16.3.1 void ArnMonitor::arnChildFound (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, so only non folder objects will then give this signal.

Example 1: monitorPath = "//Sensors/", created object = "//Sensors/Temp1/value" ==> path to child = "//Sensors/-Temp1/"

Example 2: monitorPath = "//Sensors/", created object = "//Sensors/Temp2/folder/" ==> will not give this signal as the created object is a folder.

Parameters

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

See also

[arnItemCreated\(\)](#)

14.16.3.2 void ArnMonitor::arnChildFoundFolder (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, so only non folder objects will then give this signal.

Example: monitorPath = "//Sensors/", created object = "//Sensors/Temp1/value" ==> path to child = "//Sensors/-Temp1/"

Parameters

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

See also

[arnItemCreated\(\)](#)

[arnChildFound\(\)](#)

14.16.3.3 void ArnMonitor::arnChildFoundLeaf (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.

Example: monitorPath = "//Sensors/", created object = "//Sensors/count" ==> path to child = "//Sensors/count"

Parameters

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

See also

[arnChildFound\(\)](#)

14.16.3.4 void ArnMonitor::arnItemCreated (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. Only created non folder objects will give this signal.

Parameters

in	<i>path</i>	to the created Arn Data Object
----	-------------	--

14.16.3.5 ArnClient * ArnMonitor::client () const

Get the used *client*

Returns

The *client*

See also

[setClient\(\)](#)

Definition at line 62 of file ArnMonitor.cpp.

14.16.3.6 QString ArnMonitor::clientId () const

Get the id name of the used *client*

Returns

The *client* id name

See also

[setClient\(\)](#)

Definition at line 55 of file ArnMonitor.cpp.

14.16.3.7 void ArnMonitor::foundChildDeleted (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.

Parameters

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

Definition at line 212 of file ArnMonitor.cpp.

14.16.3.8 QString ArnMonitor::monitorPath () const [inline]

Get the monitored *path*

Returns

The *path*

See also

[start\(\)](#)

Definition at line 113 of file ArnMonitor.hpp.

14.16.3.9 void* ArnMonitor::reference () const [inline]

Get the stored external reference.

Returns

The associated external reference

See also

[setReference\(\)](#)

Definition at line 134 of file ArnMonitor.hpp.

14.16.3.10 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 135 of file ArnMonitor.cpp.

14.16.3.11 void ArnMonitor::setClient (ArnClient * client, QString id = QString())

Set the *client* to be used.

Parameters

in	<i>client</i>	to be used. If 0, local monitoring is done.
in	<i>id</i>	is an optional name to assign to the client.

Definition at line 47 of file ArnMonitor.cpp.

14.16.3.12 void ArnMonitor::setMonitorPath (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 68 of file ArnMonitor.cpp.

14.16.3.13 void ArnMonitor::setReference (void * reference) [inline]

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

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

See also

[reference\(\)](#)

Definition at line 128 of file ArnMonitor.hpp.

14.16.3.14 `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

<i>in</i>	<i>path</i>	
<i>in</i>	<i>client</i>	to be used. If 0, local monitoring is done.

Definition at line 74 of file ArnMonitor.cpp.

14.16.4 Member Data Documentation

14.16.4.1 `QPointer<ArnClient> ArnMonitor::_arnClient` `[protected]`

Definition at line 195 of file ArnMonitor.hpp.

14.16.4.2 `QString ArnMonitor::_monitorPath` `[protected]`

Definition at line 196 of file ArnMonitor.hpp.

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

- [src/ArnInc/ArnMonitor.hpp \(2.2.0\)](#)
- [src/ArnMonitor.cpp \(2.2.0\)](#)

14.17 ArnPersist Class Reference

```
#include <ArnPersist.hpp>
```

Public Slots

- `bool doArchive (QString name=QString())`

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](#) (QString dbName="persist.db")
Setup the persistent database.

14.17.1 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 152 of file ArnPersist.hpp.

14.17.2 Constructor & Destructor Documentation

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

Definition at line 154 of file ArnPersist.cpp.

14.17.2.2 ArnPersist::~~ArnPersist ()

Definition at line 171 of file ArnPersist.cpp.

14.17.3 Member Function Documentation

14.17.3.1 bool ArnPersist::doArchive (QString *name* = QString()) [slot]

Do a persistent database backup

By default the backup file will be marked by date and clock. Optionally a custom name can be set for the backup file.

Parameters

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

See also

[setArchiveDir\(\)](#)

Definition at line 726 of file ArnPersist.cpp.

14.17.3.2 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 189 of file ArnPersist.cpp.

14.17.3.3 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 366 of file ArnPersist.cpp.

14.17.3.4 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

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

See also

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

Definition at line 183 of file ArnPersist.cpp.

14.17.3.5 `bool ArnPersist::setupDataBase (QString dbName = "persist.db")`

Setup the persistent database.

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

Parameters

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

See also

[Persistent Arn Data Objects](#)

Definition at line 396 of file ArnPersist.cpp.

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

See also

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

Definition at line 195 of file ArnPersist.cpp.

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

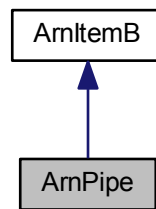
- [src/ArnInc/ArnPersist.hpp \(2.2.0\)](#)
- [src/ArnPersist.cpp \(2.2.0\)](#)

14.18 ArnPipe Class Reference

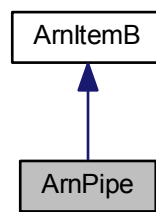
[ArnItem](#) specialized as a pipe.

```
#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](#) (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.

14.18.1 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 61 of file ArnPipe.hpp.

14.18.2 Constructor & Destructor Documentation

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

Standard constructor of a closed handle.

Parameters

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

Definition at line 47 of file ArnPipe.cpp.

14.18.2.2 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

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

See also

[open\(\)](#)

Definition at line 54 of file ArnPipe.cpp.

14.18.2.3 ArnPipe::~ArnPipe () [virtual]

Definition at line 62 of file ArnPipe.cpp.

14.18.3 Member Function Documentation

14.18.3.1 void ArnPipe::changed (QByteArray *value*) [signal]

Signal emitted when *Pipe* has received data.

This is implied by the [Arn Data Object](#) is changed.

Parameters

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

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

Return values

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

See also

[setAutoDestroy\(\)](#)

Definition at line 114 of file ArnPipe.hpp.

14.18.3.3 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 123 of file ArnPipe.cpp.

14.18.3.4 `bool ArnPipe::isMaster () const` `[inline]`

Return values

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

See also

[setMaster\(\)](#)

[Modes](#)

Definition at line 101 of file ArnPipe.hpp.

14.18.3.5 `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 111 of file ArnPipe.cpp.

14.18.3.6 `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 86 of file ArnPipe.hpp.

14.18.3.7 `ArnPipe & ArnPipe::operator= (const QByteArray & value)`

Definition at line 81 of file ArnPipe.cpp.

14.18.3.8 `void ArnPipe::outOfSequence ()` `[signal]`

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

See also

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

14.18.3.9 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 108 of file ArnPipe.hpp.

14.18.3.10 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 129 of file ArnPipe.cpp.

14.18.3.11 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 94 of file ArnPipe.hpp.

14.18.3.12 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 117 of file ArnPipe.cpp.

14.18.3.13 `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 67 of file ArnPipe.cpp.

14.18.3.14 `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 88 of file ArnPipe.cpp.

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

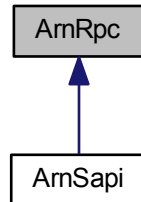
- [src/ArnInc/ArnPipe.hpp \(2.2.0\)](#)
- [src/ArnPipe.cpp \(2.2.0\)](#)

14.19 ArnRpc Class Reference

Remote Procedure Call.

```
#include <ArnRpc.hpp>
```

Inheritance diagram for ArnRpc:



Classes

- struct **ArgInfo**
- struct **Invoke**
- struct **MethodsParam**
- struct **Mode**
- struct **RpcTypeInfo**

Public Slots

- void **sendText** (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** (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)
- QString **pipePath** () const
Get the path for the used pipe
- bool **open** (QString pipePath)

- void [setPipe](#) ([ArnPipe](#) *pipe)
- [ArnPipe](#) * [pipe](#) () const
Get the used pipe
- bool [setReceiver](#) (QObject *receiver, bool useTrackRpcSender=true)
- void [setMethodPrefix](#) (QString prefix)
- 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.
- void [setHeartBeatCheck](#) (int time)
Set max time period for receiving heart beat message.
- bool [isHeartBeatOk](#) () const
Get the state of heart beat.
- void [addSenderSignals](#) (QObject *sender, 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.

14.19.1 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)
<em>rpcCommon = new ArnRpc( this);
_rpcCommon->setMethodPrefix("rpc</em>");
_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 118 of file ArnRpc.hpp.

14.19.2 Constructor & Destructor Documentation

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

Definition at line 160 of file ArnRpc.cpp.

14.19.3 Member Function Documentation

14.19.3.1 void ArnRpc::addSenderSignals (QObject * *sender*, QString *prefix*)

Definition at line 320 of file ArnRpc.cpp.

14.19.3.2 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 1252 of file ArnRpc.cpp.

14.19.3.3 `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 311 of file ArnRpc.hpp.

14.19.3.4 `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 332 of file ArnRpc.hpp.

14.19.3.5 `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.
----	-------------	---

14.19.3.6 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.
----	-------------	--

14.19.3.7 void ArnRpc::heartBeatReceived () [signal]

Signal emitted when Heart beat message is received.

14.19.3.8 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 370 of file ArnRpc.cpp.

14.19.3.9 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 407 of file ArnRpc.cpp.

14.19.3.10 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 314 of file ArnRpc.cpp.

14.19.3.11 ArnRpc::Mode ArnRpc::mode () const

Get the mode.

Returns

current *mode*

Definition at line 290 of file ArnRpc.cpp.

14.19.3.12 bool ArnRpc::open (QString pipePath)

Definition at line 186 of file ArnRpc.cpp.

14.19.3.13 void ArnRpc::outOfSequence () [signal]

Signal emitted when checked sequence order is wrong.

14.19.3.14 ArnPipe * ArnRpc::pipe () const

Get the used *pipe*

Returns

pipe

Definition at line 238 of file ArnRpc.cpp.

14.19.3.15 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.

14.19.3.16 QString ArnRpc::pipePath () const

Get the path for the used *pipe*

Returns

path

Definition at line 178 of file ArnRpc.cpp.

14.19.3.17 ArnRpc * ArnRpc::rpcSender ()

Definition at line 351 of file ArnRpc.cpp.

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

Definition at line 359 of file ArnRpc.cpp.

14.19.3.19 void ArnRpc::sendText (QString txt) [slot]

Send a general text message to the other end of the used *pipe*

Is used by [ArnRpc](#) to give errors and help messages, mostly for debugging.

Parameters

in	txt	is the text to be sent
----	-----	------------------------

See also

[textReceived\(\)](#);

Definition at line 1235 of file ArnRpc.cpp.

14.19.3.20 void ArnRpc::setHeartBeatCheck (int time)

Set max time period for receiving heart beat message.

Setting time to zero will turn off checking.

Parameters

in	time	is the time period in seconds
----	------	-------------------------------

See also

[setHeartBeatSend\(\);](#)

Definition at line 305 of file ArnRpc.cpp.

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

Set period time for sending heart beat message.

Setting time to zero will turn off sending.

Parameters

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

See also

[setHeartBeatCheck\(\);](#)

Definition at line 296 of file ArnRpc.cpp.

14.19.3.22 void ArnRpc::setIncludeSender (bool *v*)

Add sender as argument when calling a rpc method.

Deprecated Use [rpcSender\(\)](#)

Definition at line 278 of file ArnRpc.cpp.

14.19.3.23 void ArnRpc::setMethodPrefix (QString *prefix*)

Definition at line 271 of file ArnRpc.cpp.

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

Definition at line 284 of file ArnRpc.cpp.

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

Definition at line 221 of file ArnRpc.cpp.

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

Definition at line 244 of file ArnRpc.cpp.

14.19.3.27 void ArnRpc::textReceived (QString *text*) [signal]

Signal emitted when a general text message is received.

The text message is received from the other end of the used *pipe*.

Parameters

in	<i>text</i>	is the received text
----	-------------	----------------------

See also

[sendText\(\);](#)

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

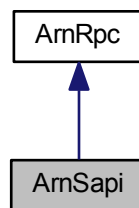
- [src/ArnInc/ArnRpc.hpp \(2.2.0\)](#)
- [src/ArnRpc.cpp \(2.2.0\)](#)

14.20 ArnSapi Class Reference

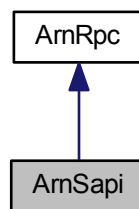
Service API.

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

Inheritance diagram for ArnSapi:



Collaboration diagram for ArnSapi:



Public Member Functions

- [ArnSapi](#) (QObject *parent=0)
- bool [open](#) (QString [pipePath](#), [Mode mode=Mode\(\)](#), const char *providerPrefix=0, const char *requester-Prefix=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.

Additional Inherited Members

14.20.1 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 113 of file ArnSapi.hpp.

14.20.2 Constructor & Destructor Documentation

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

Examples:

[ArnDemoChatServer/ChatSapi.hpp](#).

Definition at line 36 of file ArnSapi.cpp.

14.20.3 Member Function Documentation

14.20.3.1 `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 79 of file ArnSapi.cpp.

14.20.3.2 `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 63 of file ArnSapi.cpp.

14.20.3.3 `bool ArnSapi::open (QString pipePath, 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
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](#)

Definition at line 42 of file ArnSapi.cpp.

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

- src/ArnInc/ArnSapi.hpp (2.2.0)
- src/ArnSapi.cpp (2.2.0)

14.21 ArnScript Class Reference

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

Signals

- void [errorText](#) (QString txt)

Public Member Functions

- [ArnScript](#) (QObject *parent=0)
- QScriptEngine & [engine](#) () const
- bool [evaluate](#) (QByteArray script, QString [idName](#))
- bool [evaluateFile](#) (QString fileName)
- bool [logUncaughtError](#) (QScriptValue &scriptValue)
- QString [idName](#) () const
- virtual [ArnClient](#) * [getClient](#) (QString clientId)

Protected Member Functions

- void [errorLog](#) (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](#)

14.21.1 Detailed Description

Definition at line 197 of file ArnScript.hpp.

14.21.2 Constructor & Destructor Documentation

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

Definition at line 79 of file ArnScript.cpp.

14.21.3 Member Function Documentation

14.21.3.1 **QScriptEngine & ArnScript::engine** () const

Definition at line 135 of file ArnScript.cpp.

14.21.3.2 **void ArnScript::errorLog** (QString *errText*, ArnError *err* = ArnError::Undef, void * *reference* = 0)
[protected]

Definition at line 209 of file ArnScript.cpp.

14.21.3.3 **void ArnScript::errorText** (QString *txt*) [signal]

14.21.3.4 **bool ArnScript::evaluate** (QByteArray *script*, QString *idName*)

Definition at line 141 of file ArnScript.cpp.

14.21.3.5 **bool ArnScript::evaluateFile** (QString *fileName*)

Definition at line 152 of file ArnScript.cpp.

14.21.3.6 **ArnClient * ArnScript::getClient** (QString *clientId*) [virtual]

Definition at line 218 of file ArnScript.cpp.

14.21.3.7 **QString ArnScript::idName** () const

Definition at line 177 of file ArnScript.cpp.

14.21.3.8 bool ArnScript::logUncaughtError (QScriptValue & *scriptValue*)

Definition at line 161 of file ArnScript.cpp.

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

Definition at line 191 of file ArnScript.cpp.

14.21.4 Member Data Documentation

14.21.4.1 ArnDepOfferProto* ArnScript::_depOfferProto [protected]

Definition at line 226 of file ArnScript.hpp.

14.21.4.2 ArnDepProto* ArnScript::_depProto [protected]

Definition at line 227 of file ArnScript.hpp.

14.21.4.3 QScriptEngine* ArnScript::_engine [protected]

Definition at line 223 of file ArnScript.hpp.

14.21.4.4 ArnItemProto* ArnScript::_itemProto [protected]

Definition at line 224 of file ArnScript.hpp.

14.21.4.5 ArnMonitorProto* ArnScript::_monitorProto [protected]

Definition at line 225 of file ArnScript.hpp.

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

- src/ArnInc/ArnScript.hpp (2.2.0)
- src/ArnScript.cpp (2.2.0)

14.22 ArnScriptJob Class Reference

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

Public Slots

- void [setWatchDogTime](#) (int time)
- void [yield](#) ()
- void [quit](#) ()
- void [errorLog](#) (QString txt)

Signals

- void [sigQuit](#) ()

Public Member Functions

- [ArnScriptJob](#) (int id, QObject *parent=0)

Properties

- bool [sleepState](#)
- int [watchDog](#)
- int [poll](#)
- QString [name](#)

14.22.1 Detailed Description

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

Definition at line 134 of file ArnScriptJob.hpp.

14.22.2 Constructor & Destructor Documentation

14.22.2.1 `ArnScriptJob::ArnScriptJob (int id, QObject * parent = 0)` `[explicit]`

Definition at line 373 of file ArnScriptJob.cpp.

14.22.3 Member Function Documentation

14.22.3.1 `void ArnScriptJob::errorLog (QString txt)` `[inline],[slot]`

Definition at line 151 of file ArnScriptJob.hpp.

14.22.3.2 `void ArnScriptJob::quit ()` `[inline],[slot]`

Definition at line 150 of file ArnScriptJob.hpp.

14.22.3.3 `void ArnScriptJob::setWatchDogTime (int time)` `[inline],[slot]`

Definition at line 148 of file ArnScriptJob.hpp.

14.22.3.4 `void ArnScriptJob::sigQuit ()` `[signal]`

14.22.3.5 `void ArnScriptJob::yield ()` `[inline],[slot]`

Definition at line 149 of file ArnScriptJob.hpp.

14.22.4 Property Documentation

14.22.4.1 `QString ArnScriptJob::name` `[read]`

Definition at line 140 of file ArnScriptJob.hpp.

14.22.4.2 `int ArnScriptJob::poll [read],[write]`

Definition at line 139 of file ArnScriptJob.hpp.

14.22.4.3 `bool ArnScriptJob::sleepState [read],[write]`

Definition at line 137 of file ArnScriptJob.hpp.

14.22.4.4 `int ArnScriptJob::watchDog [read],[write]`

Definition at line 138 of file ArnScriptJob.hpp.

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

- [src/ArnInc/ArnScriptJob.hpp \(2.2.0\)](#)
- [src/ArnScriptJob.cpp \(2.2.0\)](#)

14.23 ArnScriptJobControl Class Reference

Is thread-safe (except doSetupJob)

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

Public Slots

- void [setScript](#) (QByteArray [script](#))

Signals

- void [scriptChanged](#) (int [id](#))
- void [errorText](#) (QString [txt](#))

Public Member Functions

- [ArnScriptJobControl](#) (QObject *parent=0)
- int [id](#) ()
- QString [name](#) () const
- void [setName](#) (QString [name](#))
- void [addInterface](#) (QString [id](#))
- void [addInterfaceList](#) (QStringList [interfaceList](#))
- QByteArray [script](#) () const
- void [loadScriptFile](#) (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.

14.23.1 Detailed Description

Is thread-safe (except doSetupJob)

Definition at line 172 of file ArnScriptJob.hpp.

14.23.2 Constructor & Destructor Documentation

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

Definition at line 384 of file ArnScriptJob.cpp.

14.23.3 Member Function Documentation

14.23.3.1 **void ArnScriptJobControl::addConfig (QObject * *obj*)**

Definition at line 483 of file ArnScriptJob.cpp.

14.23.3.2 **void ArnScriptJobControl::addInterface (QString *id*)**

Definition at line 421 of file ArnScriptJob.cpp.

14.23.3.3 **void ArnScriptJobControl::addInterfaceList (QStringList *interfaceList*)**

Definition at line 430 of file ArnScriptJob.cpp.

14.23.3.4 **QVariant ArnScriptJobControl::config (const char * *name*) const**

Definition at line 517 of file ArnScriptJob.cpp.

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

Not threadsafe, only run in same thread as script.

Definition at line 501 of file ArnScriptJob.cpp.

14.23.3.6 **void ArnScriptJobControl::errorText (QString *txt*) [signal]**

14.23.3.7 **int ArnScriptJobControl::id ()**

Definition at line 401 of file ArnScriptJob.cpp.

14.23.3.8 **void ArnScriptJobControl::loadScriptFile (QString *fileName*)**

Definition at line 459 of file ArnScriptJob.cpp.

14.23.3.9 **QString ArnScriptJobControl::name () const**

Definition at line 411 of file ArnScriptJob.cpp.

14.23.3.10 `QByteArray ArnScriptJobControl::script () const`

Definition at line 449 of file `ArnScriptJob.cpp`.

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

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

Definition at line 471 of file `ArnScriptJob.cpp`.

14.23.3.13 `void ArnScriptJobControl::setName (QString name)`

Definition at line 393 of file `ArnScriptJob.cpp`.

14.23.3.14 `void ArnScriptJobControl::setScript (QByteArray script) [slot]`

Definition at line 439 of file `ArnScriptJob.cpp`.

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

Definition at line 494 of file `ArnScriptJob.cpp`.

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

- `src/ArnInc/ArnScriptJob.hpp` (2.2.0)
- `src/ArnScriptJob.cpp` (2.2.0)

14.24 ArnScriptJobFactory Class Reference

Must be thread-safe as subclassed.

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

Public Member Functions

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

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)

14.24.1 Detailed Description

Must be thread-safe as subclassed.

Definition at line 156 of file `ArnScriptJob.hpp`.

14.24.2 Constructor & Destructor Documentation

14.24.2.1 ArnScriptJobFactory::ArnScriptJobFactory () [explicit]

Definition at line 333 of file ArnScriptJob.cpp.

14.24.2.2 ArnScriptJobFactory::~~ArnScriptJobFactory () [virtual]

Definition at line 338 of file ArnScriptJob.cpp.

14.24.3 Member Function Documentation

14.24.3.1 ArnClient * ArnScriptJobFactory::getClient (QString *id*) [virtual]

Definition at line 343 of file ArnScriptJob.cpp.

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

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

Definition at line 355 of file ArnScriptJob.cpp.

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

Definition at line 349 of file ArnScriptJob.cpp.

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

- src/ArnInc/ArnScriptJob.hpp (2.2.0)
- src/ArnScriptJob.cpp (2.2.0)

14.25 ArnScriptJobs Class Reference

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

Classes

- struct **JobSlot**
- 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)

14.25.1 Detailed Description

TODO: Add destructor that deletes jobs in `_jobSlots`

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

14.25.2 Constructor & Destructor Documentation

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

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

14.25.3 Member Function Documentation

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

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

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

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

14.25.3.3 `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 \(2.2.0\)](#)
- [src/ArnScriptJobs.cpp \(2.2.0\)](#)

14.26 ArnServer Class Reference

Class for making an *Arn Server*.

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

Classes

- struct [Type](#)

Public Member Functions

- [ArnServer](#) ([Type](#) *serverType*, QObject **parent*=0)
Create an Arn server object.
- 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

14.26.1 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 57 of file ArnServer.hpp.

14.26.2 Constructor & Destructor Documentation

14.26.2.1 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 43 of file ArnServer.cpp.

14.26.3 Member Function Documentation

14.26.3.1 QHostAddress ArnServer::listenAddress ()

Address of the interface used to listening for connections to the *Arn server*

Return values

<i>is</i>	the address (which usually is QHostAddress::Any).
-----------	---

See also

[start\(\)](#)

Definition at line 84 of file ArnServer.cpp.

14.26.3.2 int ArnServer::port ()

Port number of the *Arn server*

Return values

<i>is</i>	the port number.
-----------	------------------

Definition at line 78 of file ArnServer.cpp.

14.26.3.3 `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 52 of file ArnServer.cpp.

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

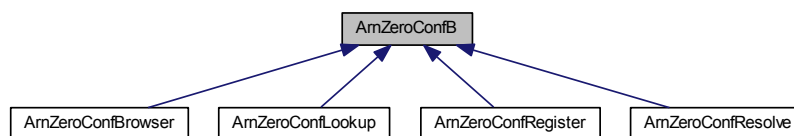
- [src/ArnInc/ArnServer.hpp \(2.2.0\)](#)
- [src/ArnServer.cpp \(2.2.0\)](#)

14.27 ArnZeroConfB Class Reference

Base class for Zero Config.

```
#include <ArnZeroConf.hpp>
```

Inheritance 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.

14.27.1 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.

14.27.2 Constructor & Destructor Documentation

14.27.2.1 ArnZeroConfB::ArnZeroConfB (QObject * *parent* = 0)

Definition at line 83 of file ArnZeroConf.cpp.

14.27.2.2 ArnZeroConfB::~ArnZeroConfB () [virtual]

Definition at line 102 of file ArnZeroConf.cpp.

14.27.3 Member Function Documentation

14.27.3.1 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.

14.27.3.2 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.

14.27.3.3 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.

14.27.3.4 void ArnZeroConfB::setDomain (const QString & domain)

Sets the domain for this Zero Config.

Default set by this class is "local.".

Parameters

in	<i>domain</i>	
----	---------------	--

See also

[domain\(\)](#)

Definition at line 296 of file ArnZeroConf.cpp.

14.27.3.5 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

in	<i>type</i>	is the service type (se above).
----	-------------	---------------------------------

See also

[serviceType\(\)](#)

Definition at line 267 of file ArnZeroConf.cpp.

14.27.3.6 void ArnZeroConfB::setSocketType (QAbstractSocket::SocketType type)

Sets the socket type for this Zero Config.

Allowed Socket type is: QAbstractSocket::TcpSocket, QAbstractSocket::UdpSocket.

Parameters

in	<i>type</i>	is one of the allowed types.
----	-------------	------------------------------

See also

[socketType\(\)](#)

Definition at line 255 of file ArnZeroConf.cpp.

14.27.3.7 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.

14.27.3.8 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:

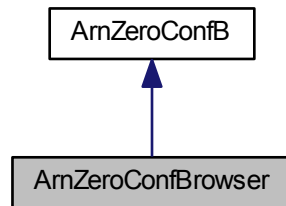
- src/ArnInc/[ArnZeroConf.hpp](#) (2.2.0)
- src/[ArnZeroConf.cpp](#) (2.2.0)

14.28 ArnZeroConfBrowser Class Reference

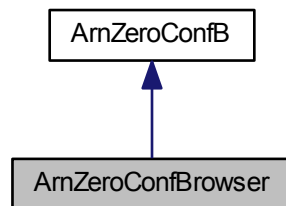
Browsing for ZeroConfig services.

```
#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](#)

14.28.1 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.

14.28.2 Constructor & Destructor Documentation

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

Standard constructor of an [ArnZeroConfBrowser](#) object.

All needed for browsing an "arn" service type.

Parameters

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

Definition at line 889 of file ArnZeroConf.cpp.

14.28.2.2 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

<i>in</i>	<i>serviceType</i>	the service type, e.g. "arn" or "_arn._tcp".
<i>in</i>	<i>parent</i>	

Definition at line 896 of file ArnZeroConf.cpp.

14.28.2.3 ArnZeroConfBrowser::~ArnZeroConfBrowser () [virtual]

Destructor of an [ArnZeroConfBrowser](#) object.

If browsing is active, it will be stopped.

Definition at line 904 of file ArnZeroConf.cpp.

14.28.3 Member Function Documentation

14.28.3.1 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 914 of file ArnZeroConf.cpp.

14.28.3.2 `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 946 of file ArnZeroConf.cpp.

14.28.3.3 `void ArnZeroConfBrowser::browseError (int errorCode) [signal]`

Indicate unsuccessful browsing.

Parameters

<i>in</i>	<i>errorCode</i>	
-----------	------------------	--

See also

[browse\(\)](#)

14.28.3.4 `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.

14.28.3.5 `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 926 of file ArnZeroConf.cpp.

14.28.3.6 `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\(\)](#)

14.28.3.7 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\(\)](#)

14.28.3.8 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 920 of file ArnZeroConf.cpp.

14.28.3.9 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\(\)](#)

14.28.3.10 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 932 of file ArnZeroConf.cpp.

14.28.3.11 void ArnZeroConfBrowser::stopBrowse () [slot]

Stop browsing.

See also

[browse\(\)](#)

Definition at line 980 of file ArnZeroConf.cpp.

14.28.3.12 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 938 of file ArnZeroConf.cpp.

14.28.4 Friends And Related Function Documentation

14.28.4.1 friend class ArnZeroConfIntern [friend]

Definition at line 938 of file ArnZeroConf.hpp.

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

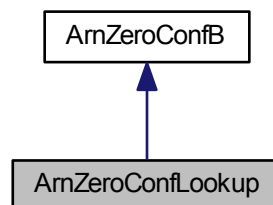
- src/ArnInc/[ArnZeroConf.hpp](#) (2.2.0)
- src/[ArnZeroConf.cpp](#) (2.2.0)

14.29 ArnZeroConfLookup Class Reference

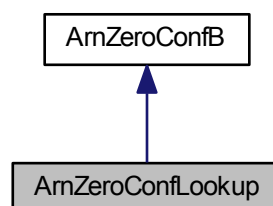
Lookup a host.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfLookup:



Collaboration diagram for ArnZeroConfLookup:



Signals

- void [lookuped](#) (int [id](#))
Indicate successfull lookup of host.
- void [lookupError](#) (int [id](#), int code)
Indicate unsuccessful lookup of host.

Public Member Functions

- [ArnZeroConfLookup](#) (QObject *parent=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](#)

14.29.1 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(lookuper(int)), this, SLOT(onLookuper(int)));
ds->lookup();

void XXX::onLookuper( int id)
{
    ArnZeroConfLookup* ds = qobject_cast<ArnZeroConfLookup
*>( sender());
    QString hostName = ds->host();
    QHostAddress hostIp = ds->hostAddr();
    ds->releaseLookup();
    ds->deleteLater();
}
```

Definition at line 783 of file ArnZeroConf.hpp.

14.29.2 Constructor & Destructor Documentation

14.29.2.1 ArnZeroConfLookup::ArnZeroConfLookup (QObject * *parent* = 0)

Standard constructor of an [ArnZeroConfLookup](#) object.

Parameters

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

Definition at line 685 of file ArnZeroConf.cpp.

14.29.2.2 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.

14.29.2.3 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.

14.29.3 Member Function Documentation

14.29.3.1 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.

14.29.3.2 QHostAddress ArnZeroConfLookup::hostAddr () const [inline]

Returns the host address for this Lookup.

Returns

current host address

Definition at line 838 of file ArnZeroConf.hpp.

14.29.3.3 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.

14.29.3.4 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 867 of file ArnZeroConf.cpp.

14.29.3.5 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 [lookuper\(\)](#) 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

[lookuper\(\)](#)
[lookupError\(\)](#)

Definition at line 723 of file ArnZeroConf.cpp.

14.29.3.6 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\(\)](#)

14.29.3.7 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\(\)](#)

14.29.3.8 void ArnZeroConfLookup::releaseLookup ()

Release the lookup.

Any lookup attempts in progress will be aborted.

Definition at line 779 of file ArnZeroConf.cpp.

14.29.3.9 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

<i>in</i>	<i>isForceQtDns- Lookup</i>	
-----------	---------------------------------	--

See also

[isForceQtDnsLookup\(\)](#)

Definition at line 873 of file ArnZeroConf.cpp.

14.29.3.10 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

<i>in</i>	<i>host</i>	is the current host name (se above)
-----------	-------------	-------------------------------------

See also

[host\(\)](#)

Definition at line 832 of file ArnZeroConf.hpp.

14.29.3.11 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

<i>in</i>	<i>id</i>	the id number
-----------	-----------	---------------

See also

[id\(\)](#)

Definition at line 717 of file ArnZeroConf.cpp.

14.29.4 Friends And Related Function Documentation

14.29.4.1 friend class ArnZeroConfIntern [friend]

Definition at line 785 of file ArnZeroConf.hpp.

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

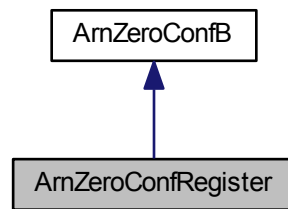
- [src/ArnInc/ArnZeroConf.hpp \(2.2.0\)](#)
- [src/ArnZeroConf.cpp \(2.2.0\)](#)

14.30 ArnZeroConfRegister Class Reference

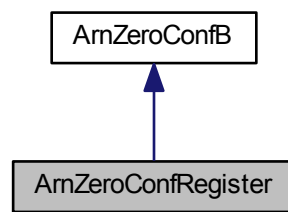
Registering a ZeroConfig service.

```
#include <ArnZeroConf.hpp>
```

Inheritance diagram for ArnZeroConfRegister:



Collaboration diagram for ArnZeroConfRegister:



Signals

- void `registered` (QString `serviceName`)
Indicate successfull registration of service.
- void `registrationError` (int code)
Indicate unsuccessfull 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](#)

14.30.1 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.

14.30.2 Constructor & Destructor Documentation

14.30.2.1 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

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

Definition at line 370 of file ArnZeroConf.cpp.

14.30.2.2 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

in	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
in	<i>parent</i>	

Definition at line 377 of file ArnZeroConf.cpp.

14.30.2.3 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

in	<i>serviceName</i>	the human readable naming of the service, e.g. "My fantastic service".
in	<i>serviceType</i>	the service type, e.g. "arn" or "_arn._tcp".
in	<i>port</i>	the service port num
in	<i>parent</i>	

Definition at line 386 of file ArnZeroConf.cpp.

14.30.2.4 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.

14.30.3 Member Function Documentation

14.30.3.1 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.

14.30.3.2 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.

14.30.3.3 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::toX-String\(\)](#) can return "protovers=1.0 MyParam=xyz".

Parameters

<i>out</i>	<i>xsm</i>	is the loaded XStringMap if successfull, otherwise undefined.
------------	------------	---

Return values

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

See also

[setTxtRecordMap\(\)](#)
[Arn::XStringMap](#)

Definition at line 509 of file ArnZeroConf.hpp.

14.30.3.4 `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.

14.30.3.5 `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.

14.30.3.6 `void ArnZeroConfRegister::registered (QString serviceName)` `[signal]`

Indicate successfull registration of service.

The service name will also be internally updated, it can be accesed via [currentServiceName\(\)](#).

Parameters

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

See also

[registerService\(\)](#)
[setServiceName\(\)](#)
[serviceName\(\)](#)

14.30.3.7 `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

<code>in</code>	<code>noAutoRename</code>	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.

14.30.3.8 `void ArnZeroConfRegister::registrationError (int code)` `[signal]`

Indicate unsuccessful registration of service.

Parameters

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

See also

[registerService\(\)](#)

14.30.3.9 `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.

14.30.3.10 `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.

14.30.3.11 `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.

14.30.3.12 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.

14.30.3.13 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.

14.30.3.14 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.

14.30.3.15 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.

14.30.3.16 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.

14.30.3.17 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.

14.30.3.18 `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.

14.30.4 Friends And Related Function Documentation

14.30.4.1 `friend class ArnZeroConfIntern [friend]`

Definition at line 368 of file ArnZeroConf.hpp.

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

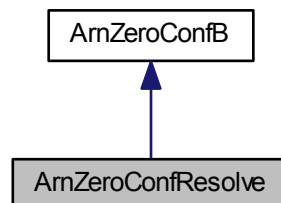
- [src/ArnInc/ArnZeroConf.hpp \(2.2.0\)](#)
- [src/ArnZeroConf.cpp \(2.2.0\)](#)

14.31 ArnZeroConfResolve Class Reference

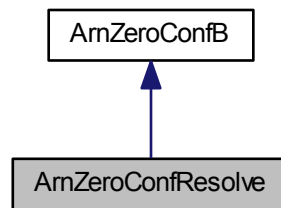
Resolv a ZeroConfig service.

```
#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](#)

14.31.1 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.

14.31.2 Constructor & Destructor Documentation

14.31.2.1 ArnZeroConfResolve::ArnZeroConfResolve (QObject * parent = 0)

Standard constructor of an ArnZeroConfResolv object.

Parameters

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

Definition at line 523 of file ArnZeroConf.cpp.

14.31.2.2 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.

14.31.2.3 `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.

14.31.2.4 `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.

14.31.3 Member Function Documentation

14.31.3.1 `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::toX-String\(\)](#) can return "protovers=1.0 MyParam=xyz".

Parameters

out	<i>xsm</i>	is the loaded XStringMap if successfull, otherwise undefined.
-----	------------	---

Return values

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

See also

[Arn::XStringMap](#)

Definition at line 703 of file ArnZeroConf.hpp.

14.31.3.2 `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.

14.31.3.3 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.

14.31.3.4 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.

14.31.3.5 void ArnZeroConfResolve::releaseResolve ()

Release the resolving.

Any resolve attempts in progress will be aborted.

Definition at line 610 of file ArnZeroConf.cpp.

14.31.3.6 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

<i>in</i>	<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.

14.31.3.7 `void ArnZeroConfResolve::resolved (int id, const QByteArray & escFullDomain) [signal]`

Indicate successfull resolve of service.

Parameters

<code>in</code>	<code><i>id</i></code>	is the id number for this resolve
<code>in</code>	<code><i>escFullDomain</i></code>	is the raw full domain with esc sequences

See also

[resolve\(\)](#)

14.31.3.8 `void ArnZeroConfResolve::resolveError (int id, int code) [signal]`

Indicate unsuccessfull resolve of service.

Parameters

<code>in</code>	<code><i>id</i></code>	is the id number for this resolve
<code>in</code>	<code><i>code</i></code>	is the error code.

See also

[resolve\(\)](#)

14.31.3.9 `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.

14.31.3.10 `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

<code>in</code>	<code><i>id</i></code>	the id number
-----------------	------------------------	---------------

See also

[id\(\)](#)

Definition at line 566 of file ArnZeroConf.cpp.

14.31.3.11 `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

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

See also

[serviceName\(\)](#)

Definition at line 691 of file ArnZeroConf.hpp.

14.31.3.12 `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.

14.31.4 Friends And Related Function Documentation

14.31.4.1 `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 \(2.2.0\)](#)
- [src/ArnZeroConf.cpp \(2.2.0\)](#)

14.32 Arn::Coding Struct Reference

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

Public Types

- enum `E` { `Binary` = 0x0000, `Text` = 0x1000 }

14.32.1 Detailed Description

Definition at line 130 of file Arn.hpp.

14.32.2 Member Enumeration Documentation

14.32.2.1 enum Arn::Coding::E

Enumerator:

Binary No special coding, can be anything.

Text Text coding, can be any character set.

Definition at line 131 of file Arn.hpp.

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

- src/ArnInc/Arn.hpp (2.2.0)

14.33 ArnClient::ConnectStat Struct Reference

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

Public Types

- enum E {
 Init = 0, Connecting, Connected, Error,
 Disconnected, TriedAll }

14.33.1 Detailed Description

Definition at line 75 of file ArnClient.hpp.

14.33.2 Member Enumeration Documentation

14.33.2.1 enum ArnClient::ConnectStat::E

Enumerator:

Init Initialized, not yet any result of trying to connect ...

Connecting Trying to connect to an Arn host.

Connected Successfully connected to an Arn host.

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 76 of file ArnClient.hpp.

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

- src/ArnInc/ArnClient.hpp (2.2.0)

14.34 Arn::DataType Struct Reference

Data type of an Arn Data Object

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

Public Types

- enum [E](#) {
 [Null](#) = 0, [Int](#) = 1, [Double](#) = 2, [ByteArray](#) = 3,
 [String](#) = 4, [Variant](#) = 5 }

14.34.1 Detailed Description

Data type of an [Arn Data Object](#)

Definition at line 65 of file Arn.hpp.

14.34.2 Member Enumeration Documentation

14.34.2.1 enum [Arn::DataType::E](#)

Enumerator:

Null

Int

Double

ByteArray

String

Variant

Definition at line 66 of file Arn.hpp.

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

- src/ArnInc/[Arn.hpp \(2.2.0\)](#)

14.35 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 }

14.35.1 Detailed Description

Errors of ZeroConfig, other values are defined in dns_sd.h.

Definition at line 53 of file ArnZeroConf.hpp.

14.35.2 Member Enumeration Documentation

14.35.2.1 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 \(2.2.0\)](#)

14.36 ArnItemB::ExportCode Struct Reference

Code used in blob for arnExport() and arnImport()

```
#include <ArnItemB.hpp>
```

Public Types

- enum [E](#) {
 [ByteArray](#) = 3, [String](#) = 4, [Variant](#) = 5, [VariantTxt](#) = 16,
 [VariantBin](#) = 17 }

14.36.1 Detailed Description

Code used in blob for arnExport() and arnImport()

Definition at line 69 of file ArnItemB.hpp.

14.36.2 Member Enumeration Documentation

14.36.2.1 enum ArnItemB::ExportCode::E

Enumerator:

ByteArray

String

Variant

VariantTxt

VariantBin

Definition at line 70 of file ArnItemB.hpp.

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

- src/ArnInc/[ArnItemB.hpp \(2.2.0\)](#)

14.37 ArnClient::HostAddrPort Struct Reference

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

Public Member Functions

- [HostAddrPort](#) ()

Public Attributes

- QString [addr](#)
- quint16 [port](#)

14.37.1 Detailed Description

Definition at line 93 of file ArnClient.hpp.

14.37.2 Constructor & Destructor Documentation

14.37.2.1 [ArnClient::HostAddrPort::HostAddrPort \(\)](#) `[inline]`

Definition at line 97 of file ArnClient.hpp.

14.37.3 Member Data Documentation

14.37.3.1 [QString ArnClient::HostAddrPort::addr](#)

Definition at line 94 of file ArnClient.hpp.

14.37.3.2 [quint16 ArnClient::HostAddrPort::port](#)

Definition at line 95 of file ArnClient.hpp.

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

- [src/ArnInc/ArnClient.hpp \(2.2.0\)](#)

14.38 ArnRpc::Invoke Struct Reference

```
#include <ArnRpc.hpp>
```

Public Types

- enum [E](#) { [NoQueue](#) = 0x01 }

14.38.1 Detailed Description

Definition at line 152 of file ArnRpc.hpp.

14.38.2 Member Enumeration Documentation

14.38.2.1 enum ArnRpc::Invoke::E

Enumerator:

NoQueue This invoke is not queued, multiple calls to same method might overwrite.

Definition at line 153 of file ArnRpc.hpp.

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

- src/ArnInc/ArnRpc.hpp (2.2.0)

14.39 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, [Threaded](#) = 0x08 }

14.39.1 Detailed Description

Link flags when accessing an [Arn Data Object](#)

Definition at line 107 of file Arn.hpp.

14.39.2 Member Enumeration Documentation

14.39.2.1 enum Arn::LinkFlags::E

Enumerator:

Folder

CreateAllowed

SilentError

Threaded

Definition at line 108 of file Arn.hpp.

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

- src/ArnInc/Arn.hpp (2.2.0)

14.40 ArnRpc::Mode Struct 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` | `AutoDestroy` }

14.40.1 Detailed Description

Examples:

[ArnDemoChatServer/MainWindow.cpp](#).

Definition at line 122 of file `ArnRpc.hpp`.

14.40.2 Member Enumeration Documentation

14.40.2.1 enum `ArnRpc::Mode::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*

Definition at line 123 of file `ArnRpc.hpp`.

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

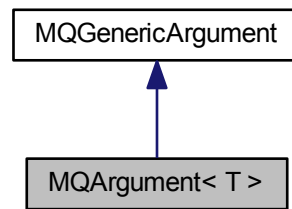
- [src/ArnInc/ArnRpc.hpp \(2.2.0\)](#)

14.41 MQArgument< T > Class Template Reference

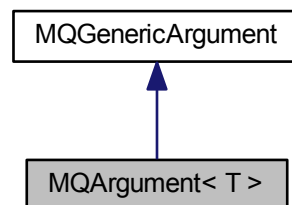
Similar to `QArgument` but with added argument label (parameter name)

```
#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)

14.41.1 Detailed Description

```
template<class T>class MQArgument< T >
```

Similar to QArgument but with added argument label (parameter name)

Definition at line 74 of file ArnRpc.hpp.

14.41.2 Constructor & Destructor Documentation

14.41.2.1 `template<class T> MQArgument< T >::MQArgument (const char * aName, const char * aLabel, const T &aData) [inline]`

Definition at line 77 of file ArnRpc.hpp.

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

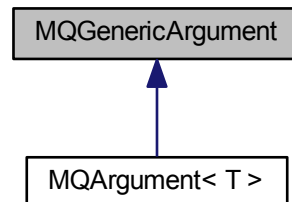
- [src/ArnInc/ArnRpc.hpp \(2.2.0\)](#)

14.42 MQGenericArgument Class Reference

Similar to QGenericArgument but with added argument label (parameter name)

```
#include <ArnRpc.hpp>
```

Inheritance 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

14.42.1 Detailed Description

Similar to QGenericArgument but with added argument label (parameter name)

Definition at line 58 of file ArnRpc.hpp.

14.42.2 Constructor & Destructor Documentation

14.42.2.1 `MQGenericArgument::MQGenericArgument (const char * aName = 0, const char * aLabel = 0, const void * aData = 0)` `[inline]`

Definition at line 61 of file ArnRpc.hpp.

14.42.2.2 `MQGenericArgument::MQGenericArgument (const QGenericArgument & qgenArg)` `[inline]`

Definition at line 63 of file ArnRpc.hpp.

14.42.3 Member Function Documentation

14.42.3.1 `const char* MQGenericArgument::label ()` `const` `[inline]`

Definition at line 65 of file ArnRpc.hpp.

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

- [src/ArnInc/ArnRpc.hpp \(2.2.0\)](#)

14.43 Arn::NameF Struct Reference

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

Public Types

- enum [E](#) { [NoFolderMark](#) = 0x01, [EmptyOk](#) = 0x02, [Relative](#) = 0x04 }
- Selects a format for path or item name.*

14.43.1 Detailed Description

Definition at line 117 of file Arn.hpp.

14.43.2 Member Enumeration Documentation

14.43.2.1 enum Arn::NameF::E

Selects a format for path or item name.

Enumerator:

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 119 of file Arn.hpp.

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

- src/ArnInc/[Arn.hpp \(2.2.0\)](#)

14.44 Arn::ObjectMode Struct Reference

General global mode of an [Arn Data Object](#)

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

Public Types

- enum [E](#) { [BiDir](#) = 0x01, [Pipe](#) = 0x02, [Save](#) = 0x04 }

14.44.1 Detailed Description

General global mode of an [Arn Data Object](#)

Definition at line 79 of file Arn.hpp.

14.44.2 Member Enumeration Documentation

14.44.2.1 enum Arn::ObjectMode::E

Enumerator:

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 80 of file Arn.hpp.

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

- src/ArnInc/[Arn.hpp \(2.2.0\)](#)

14.45 Arn::ObjectSyncMode Struct Reference

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

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

Public Types

- enum **E** { **Normal** = 0x000, **Monitor** = 0x001, **Master** = 0x100, **AutoDestroy** = 0x200 }

14.45.1 Detailed Description

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

Definition at line 92 of file Arn.hpp.

14.45.2 Member Enumeration Documentation

14.45.2.1 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 93 of file Arn.hpp.

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

- src/ArnInc/[Arn.hpp \(2.2.0\)](#)

14.46 ArnRpc::MethodsParam::Params Struct Reference

```
#include <ArnRpc.hpp>
```

Public Attributes

- QList< QByteArray > [paramNames](#)
- QList< QList< int > > [methodIdsTab](#)
- QList< int > [allMethodIds](#)

14.46.1 Detailed Description

Definition at line 424 of file ArnRpc.hpp.

14.46.2 Member Data Documentation

14.46.2.1 QList<int> ArnRpc::MethodsParam::Params::allMethodIds

Definition at line 427 of file ArnRpc.hpp.

14.46.2.2 QList< QList<int> > ArnRpc::MethodsParam::Params::methodIdsTab

Definition at line 426 of file ArnRpc.hpp.

14.46.2.3 QList<QByteArray> ArnRpc::MethodsParam::Params::paramNames

Definition at line 425 of file ArnRpc.hpp.

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

- src/ArnInc/[ArnRpc.hpp \(2.2.0\)](#)

14.47 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 }

14.47.1 Detailed Description

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

Definition at line 52 of file Arn.hpp.

14.47.2 Member Enumeration Documentation

14.47.2.1 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 53 of file Arn.hpp.

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

- src/ArnInc/[Arn.hpp \(2.2.0\)](#)

14.48 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 }

14.48.1 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.

14.48.2 Member Enumeration Documentation

14.48.2.1 enum ArnZeroConf::State::E

Enumerator:

- None*** Inactive state.
- Registering*** Registering service in progress.
- Registered*** Registering service has finished sucessfully.
- Register*** isAny(): Registering service in progress or has finished sucessfully
- Browsing*** Browsing for service in progress.
- Resolving*** Resolving service in progress.
- Resolved*** Resolving service has finished sucessfully.
- Resolve*** isAny(): Resolving service in progress or has finished sucessfully
- LookingUp*** Lookup host in progress.
- Lookuped*** Lookup host has finished sucessfully.
- Lookup*** isAny(): Lookup host in progress or has finished sucessfully
- 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](#) (2.2.0)

14.49 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 }

14.49.1 Detailed Description

States of DiscoverAdvertise / These values must be synced with: [ArnZeroConf::State](#).

Definition at line 623 of file ArnDiscover.hpp.

14.49.2 Member Enumeration Documentation

14.49.2.1 enum ArnDiscoverAdvertise::State::E

Enumerator:

None Inactive state.

StartupAdvertise Startup advertising in progress.

Advertising Is now advertising. Startup has finished sucessfully.

Advertise isAny(): Startup advertising in progress or has finished sucessfully.

Definition at line 624 of file ArnDiscover.hpp.

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

- src/ArnInc/[ArnDiscover.hpp](#) (2.2.0)

14.50 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](#) }

14.50.1 Detailed Description

[State](#) of [Arn](#) discover browse data. Can be tested by relative order.

Definition at line 73 of file ArnDiscover.hpp.

14.50.2 Member Enumeration Documentation

14.50.2.1 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 74 of file ArnDiscover.hpp.

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

- src/ArnInc/[ArnDiscover.hpp \(2.2.0\)](#)

14.51 ArnError::StdCode Struct Reference

```
#include <ArnError.hpp>
```

Public Types

- enum [E](#) {
[Ok](#) = 0, [Info](#) = 1, [Warning](#) = 2, [Err_Undef](#) = 15,
[Err_Custom](#) = 16 }

14.51.1 Detailed Description

Definition at line 40 of file ArnError.hpp.

14.51.2 Member Enumeration Documentation

14.51.2.1 enum ArnError::StdCode::E

Enumerator:

Ok
Info
Warning
Err_Undef
Err_Custom

Definition at line 42 of file ArnError.hpp.

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

- src/ArnInc/[ArnError.hpp \(2.2.0\)](#)

14.52 ArnItemValve::SwitchMode Struct Reference

```
#include <ArnItemValve.hpp>
```

Public Types

- enum [E](#) { [InStream](#) = 0x01, [OutStream](#) = 0x02, [InOutStream](#) = InStream | OutStream }

14.52.1 Detailed Description

Definition at line 80 of file ArnItemValve.hpp.

14.52.2 Member Enumeration Documentation

14.52.2.1 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 81 of file ArnItemValve.hpp.

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

- src/ArnInc/[ArnItemValve.hpp \(2.2.0\)](#)

14.53 ArnServer::Type Struct Reference

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

Public Types

- enum [E](#) { [NetSync](#) }

14.53.1 Detailed Description

Definition at line 61 of file ArnServer.hpp.

14.53.2 Member Enumeration Documentation

14.53.2.1 enum ArnServer::Type::E

Enumerator:

NetSync

Definition at line 62 of file ArnServer.hpp.

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

- src/ArnInc/[ArnServer.hpp \(2.2.0\)](#)

14.54 ArnDiscover::Type Struct Reference

Types of [Arn](#) discover advertise.

```
#include <ArnDiscover.hpp>
```

Public Types

- enum [E](#) { [None](#), [Server](#), [Client](#) }

14.54.1 Detailed Description

Types of [Arn](#) discover advertise.

Definition at line 48 of file ArnDiscover.hpp.

14.54.2 Member Enumeration Documentation

14.54.2.1 enum ArnDiscover::Type::E

Enumerator:

None Undefined [Arn](#) discover.

Server Server [Arn](#) discover.

Client Client [Arn](#) discover.

Definition at line 49 of file ArnDiscover.hpp.

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

- [src/ArnInc/ArnDiscover.hpp \(2.2.0\)](#)

14.55 ArnScriptJobs::Type Struct Reference

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

Public Types

- enum [E](#) { [Null](#), [Cooperative](#), [Preemptive](#) }

14.55.1 Detailed Description

Definition at line 92 of file ArnScriptJobs.hpp.

14.55.2 Member Enumeration Documentation

14.55.2.1 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 \(2.2.0\)](#)

14.56 Arn::XStringMap Class Reference

Container class with string representation for serialized data.

```
#include <XStringMap.hpp>
```

Public Member Functions

- [XStringMap](#) ()
- [XStringMap](#) (const QByteArray &xString)
- [XStringMap](#) (const QVariantMap &variantMap)
- [~XStringMap](#) ()
- 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)
- void [set](#) (int i, const QByteArray &val)
- void [set](#) (const char *key, const QByteArray &val)
- void [set](#) (const char *key, const char *val)
- void [set](#) (const QByteArray &key, const QByteArray &val)
- void [set](#) (const char *key, const QString &val)
- void [set](#) (const QByteArray &key, const QString &val)
- void [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
- void [remove](#) (int index)
- void [remove](#) (const char *[key](#))
- void [remove](#) (const QByteArray &[key](#))
- void [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)

Static Public Member Functions

- static void [stringCode](#) (QByteArray &dst, const QByteArray &src)
- static void [stringDecode](#) (QByteArray &dst, const QByteArray &src)

14.56.1 Detailed Description

Container class with string representation for serialized data.

The primary usage is for creating and parsing serialized data. it's optimized for giving an easy readable representation which never contains char codes below 32 (space).

This class can store data with a key like QMap. There is a guaranteed order of storing, i.e. its not sorted like QMap.

The stored data can be ascii as well as binary.

Following mapping is done when serialized to the XString:

```
Special codes below 32: code 0 -> "\0", code 10 -> "\n", code 13 -> "\r"
General codes below 32: code 1 -> "^A", code 2 -> "^B" and so on to code 31
code 32 (space) -> "_", "_" -> "\_", "^" -> "\^", "\" -> "\\\""
```

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 72 of file XStringMap.hpp.

14.56.2 Constructor & Destructor Documentation

14.56.2.1 Arn::XStringMap::XStringMap () [explicit]

Definition at line 39 of file ArnXStringMap.cpp.

14.56.2.2 Arn::XStringMap::XStringMap (const QByteArray & xString) [explicit]

Definition at line 45 of file ArnXStringMap.cpp.

14.56.2.3 Arn::XStringMap::XStringMap (const QVariantMap & variantMap) [explicit]

Definition at line 52 of file ArnXStringMap.cpp.

14.56.2.4 Arn::XStringMap::~~XStringMap ()

Definition at line 59 of file ArnXStringMap.cpp.

14.56.3 Member Function Documentation

14.56.3.1 XStringMap & Arn::XStringMap::add (const char * key, const QByteArray & val)

Definition at line 153 of file ArnXStringMap.cpp.

14.56.3.2 XStringMap & Arn::XStringMap::add (const char * key, const char * val)

Definition at line 173 of file ArnXStringMap.cpp.

14.56.3.3 XStringMap & Arn::XStringMap::add (const char * keyPrefix, uint eNum, const QByteArray & val)

Definition at line 179 of file ArnXStringMap.cpp.

14.56.3.4 XStringMap & Arn::XStringMap::add (const QByteArray & key, const QByteArray & val)

Definition at line 189 of file ArnXStringMap.cpp.

14.56.3.5 XStringMap & Arn::XStringMap::add (const char * key, const QString & val)

Definition at line 195 of file ArnXStringMap.cpp.

14.56.3.6 XStringMap & Arn::XStringMap::add (const char * keyPrefix, uint eNum, const QString & val)

Definition at line 201 of file ArnXStringMap.cpp.

14.56.3.7 XStringMap & Arn::XStringMap::add (const QByteArray & key, const QString & val)

Definition at line 207 of file ArnXStringMap.cpp.

14.56.3.8 XStringMap & Arn::XStringMap::add (const QString & *key*, const QString & *val*)

Definition at line 213 of file ArnXStringMap.cpp.

14.56.3.9 XStringMap & Arn::XStringMap::add (const XStringMap & *other*)

Definition at line 219 of file ArnXStringMap.cpp.

14.56.3.10 XStringMap & Arn::XStringMap::add (const QVariantMap & *variantMap*)

Definition at line 229 of file ArnXStringMap.cpp.

14.56.3.11 void Arn::XStringMap::append (const char * *key*, const QByteArray & *val*) [inline]

Definition at line 145 of file XStringMap.hpp.

14.56.3.12 void Arn::XStringMap::append (const char * *key*, const char * *val*) [inline]

Definition at line 147 of file XStringMap.hpp.

14.56.3.13 void Arn::XStringMap::append (const char * *keyPrefix*, uint *eNum*, const QByteArray & *val*) [inline]

Definition at line 149 of file XStringMap.hpp.

14.56.3.14 void Arn::XStringMap::append (const QByteArray & *key*, const QByteArray & *val*) [inline]

Definition at line 151 of file XStringMap.hpp.

14.56.3.15 void Arn::XStringMap::append (const char * *key*, const QString & *val*) [inline]

Definition at line 153 of file XStringMap.hpp.

14.56.3.16 void Arn::XStringMap::append (const char * *keyPrefix*, uint *eNum*, const QString & *val*) [inline]

Definition at line 155 of file XStringMap.hpp.

14.56.3.17 void Arn::XStringMap::append (const QByteArray & *key*, const QString & *val*) [inline]

Definition at line 157 of file XStringMap.hpp.

14.56.3.18 void Arn::XStringMap::append (const QString & *key*, const QString & *val*) [inline]

Definition at line 159 of file XStringMap.hpp.

14.56.3.19 void Arn::XStringMap::append (const XStringMap & *other*) [inline]

Definition at line 161 of file XStringMap.hpp.

14.56.3.20 void Arn::XStringMap::append (const QVariantMap & *other*) [inline]

Definition at line 163 of file XStringMap.hpp.

14.56.3.21 void Arn::XStringMap::clear (bool *freeMem* = false)

Definition at line 70 of file ArnXStringMap.cpp.

14.56.3.22 bool Arn::XStringMap::fromXString (const QByteArray & *inXString*, int *size* = -1)

Definition at line 543 of file ArnXStringMap.cpp.

14.56.3.23 int Arn::XStringMap::indexOf (const char * *key*, int *from* = 0) const

Definition at line 90 of file ArnXStringMap.cpp.

14.56.3.24 int Arn::XStringMap::indexOf (const QByteArray & *key*, int *from* = 0) const

Definition at line 103 of file ArnXStringMap.cpp.

14.56.3.25 int Arn::XStringMap::indexOf (const QString & *key*, int *from* = 0) const

Definition at line 114 of file ArnXStringMap.cpp.

14.56.3.26 int Arn::XStringMap::indexOfValue (const QByteArray & *value*, int *from* = 0) const

Definition at line 120 of file ArnXStringMap.cpp.

14.56.3.27 int Arn::XStringMap::indexOfValue (const QString & *value*, int *from* = 0) const

Definition at line 131 of file ArnXStringMap.cpp.

14.56.3.28 QByteArray Arn::XStringMap::key (int *i*, const char * *def* = 0) const

Definition at line 304 of file ArnXStringMap.cpp.

14.56.3.29 QByteArray Arn::XStringMap::key (const QByteArray & *value*, const char * *def* = 0) const

Definition at line 312 of file ArnXStringMap.cpp.

14.56.3.30 QByteArray Arn::XStringMap::key (const QString & *value*, const char * *def* = 0) const

Definition at line 321 of file ArnXStringMap.cpp.

14.56.3.31 const QByteArray & Arn::XStringMap::keyRef (int *i*) const

Definition at line 296 of file ArnXStringMap.cpp.

14.56.3.32 `QStringList Arn::XStringMap::keys () const`

Definition at line 485 of file ArnXStringMap.cpp.

14.56.3.33 `QString Arn::XStringMap::keyString (int i, const QString & def = QString()) const`

Definition at line 327 of file ArnXStringMap.cpp.

14.56.3.34 `QString Arn::XStringMap::keyString (const QString & value, const QString & def = QString()) const`

Definition at line 336 of file ArnXStringMap.cpp.

14.56.3.35 `int Arn::XStringMap::maxEnumOf (const char * keyPrefix) const`

Definition at line 137 of file ArnXStringMap.cpp.

14.56.3.36 `XStringMap & Arn::XStringMap::operator+= (const XStringMap & other)`

Definition at line 702 of file ArnXStringMap.cpp.

14.56.3.37 `XStringMap & Arn::XStringMap::operator+= (const QVariantMap & other)`

Definition at line 696 of file ArnXStringMap.cpp.

14.56.3.38 `void Arn::XStringMap::remove (int index)`

Definition at line 442 of file ArnXStringMap.cpp.

14.56.3.39 `void Arn::XStringMap::remove (const char * key)`

Definition at line 456 of file ArnXStringMap.cpp.

14.56.3.40 `void Arn::XStringMap::remove (const QByteArray & key)`

Definition at line 462 of file ArnXStringMap.cpp.

14.56.3.41 `void Arn::XStringMap::remove (const QString & key)`

Definition at line 468 of file ArnXStringMap.cpp.

14.56.3.42 `void Arn::XStringMap::set (int i, const QByteArray & val)`

Definition at line 245 of file ArnXStringMap.cpp.

14.56.3.43 `void Arn::XStringMap::set (const char * key, const QByteArray & val)`

Definition at line 256 of file ArnXStringMap.cpp.

14.56.3.44 void Arn::XStringMap::set (const char * *key*, const char * *val*)

Definition at line 266 of file ArnXStringMap.cpp.

14.56.3.45 void Arn::XStringMap::set (const QByteArray & *key*, const QByteArray & *val*)

Definition at line 272 of file ArnXStringMap.cpp.

14.56.3.46 void Arn::XStringMap::set (const char * *key*, const QString & *val*)

Definition at line 278 of file ArnXStringMap.cpp.

14.56.3.47 void Arn::XStringMap::set (const QByteArray & *key*, const QString & *val*)

Definition at line 284 of file ArnXStringMap.cpp.

14.56.3.48 void Arn::XStringMap::set (const QString & *key*, const QString & *val*)

Definition at line 290 of file ArnXStringMap.cpp.

14.56.3.49 void Arn::XStringMap::setEmptyKeysToValue ()

Definition at line 474 of file ArnXStringMap.cpp.

14.56.3.50 int Arn::XStringMap::size () const [inline]

Definition at line 80 of file XStringMap.hpp.

14.56.3.51 void Arn::XStringMap::squeeze ()

Definition at line 81 of file ArnXStringMap.cpp.

14.56.3.52 void Arn::XStringMap::stringCode (QByteArray & *dst*, const QByteArray & *src*) [static]

Definition at line 588 of file ArnXStringMap.cpp.

14.56.3.53 void Arn::XStringMap::stringDecode (QByteArray & *dst*, const QByteArray & *src*) [static]

Definition at line 642 of file ArnXStringMap.cpp.

14.56.3.54 QVariantMap Arn::XStringMap::toVariantMap () const

Definition at line 511 of file ArnXStringMap.cpp.

14.56.3.55 QByteArray Arn::XStringMap::toXString () const

Definition at line 525 of file ArnXStringMap.cpp.

14.56.3.56 `QByteArray Arn::XStringMap::value (int i, const char * def = 0) const`

Definition at line 351 of file ArnXStringMap.cpp.

14.56.3.57 `QByteArray Arn::XStringMap::value (const char * key, const char * def = 0) const`

Definition at line 359 of file ArnXStringMap.cpp.

14.56.3.58 `QByteArray Arn::XStringMap::value (const char * keyPrefix, uint eNum, const char * def = 0) const`

Definition at line 368 of file ArnXStringMap.cpp.

14.56.3.59 `QByteArray Arn::XStringMap::value (const QByteArray & key, const char * def = 0) const`

Definition at line 381 of file ArnXStringMap.cpp.

14.56.3.60 `QByteArray Arn::XStringMap::value (const QByteArray & key, const QByteArray & def) const`

Definition at line 390 of file ArnXStringMap.cpp.

14.56.3.61 `const QByteArray & Arn::XStringMap::valueRef (int i) const`

Definition at line 343 of file ArnXStringMap.cpp.

14.56.3.62 `QStringList Arn::XStringMap::values (const char * keyPrefix = 0) const`

Definition at line 496 of file ArnXStringMap.cpp.

14.56.3.63 `QString Arn::XStringMap::valueString (int i, const QString & def = QString()) const`

Definition at line 400 of file ArnXStringMap.cpp.

14.56.3.64 `QString Arn::XStringMap::valueString (const char * key, const QString & def = QString()) const`

Definition at line 409 of file ArnXStringMap.cpp.

14.56.3.65 `QString Arn::XStringMap::valueString (const char * keyPrefix, uint eNum, const QString & def = QString()) const`

Definition at line 416 of file ArnXStringMap.cpp.

14.56.3.66 `QString Arn::XStringMap::valueString (const QByteArray & key, const QString & def = QString()) const`

Definition at line 428 of file ArnXStringMap.cpp.

14.56.3.67 `QString Arn::XStringMap::valueString (const QString & key, const QString & def = QString()) const`

Definition at line 435 of file ArnXStringMap.cpp.

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

- [src/ArnInc/XStringMap.hpp \(2.2.0\)](#)
- [src/ArnXStringMap.cpp \(2.2.0\)](#)

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

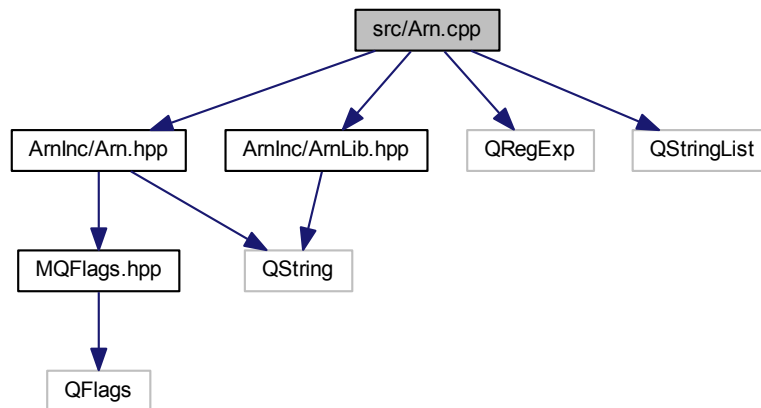
15.6 examples/Examples.txt File Reference

15.7 README.md File Reference

15.8 src/Arn.cpp File Reference

```
#include "ArnInc/Arn.hpp"  
#include "ArnInc/ArnLib.hpp"  
#include <QRegExp>  
#include <QStringList>
```

Include dependency graph for Arn.cpp:



Namespaces

- namespace [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::twinPath (const QString &path)`
Get the bidirectional twin to a given path
- `QString Arn::providerPath (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)`

- QString [Arn::makeHostWithInfo](#) (const QString &host, const QString &info)

- QString [Arn::hostFromHostWithInfo](#) (const QString &hostWithInfo)

Get the host from the HostWithInfo string.

- 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/"

```
#include "ArnInc/ArnClient.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnSync.hpp"
#include <QTcpSocket>
#include <QStringList>
#include <QTimer>
#include <QDebug>
```

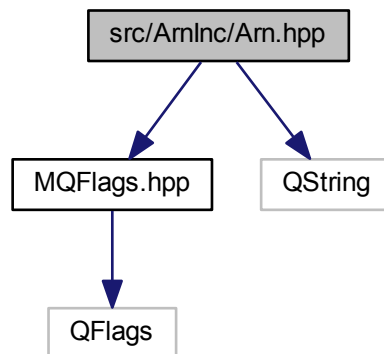
```
#include "ArnInc/ArnDepend.hpp"
#include "ArnInc/ArnM.hpp"
#include <QUuid>
#include <QTimer>
#include <QtAlgorithms>
#include <QDebug>
```



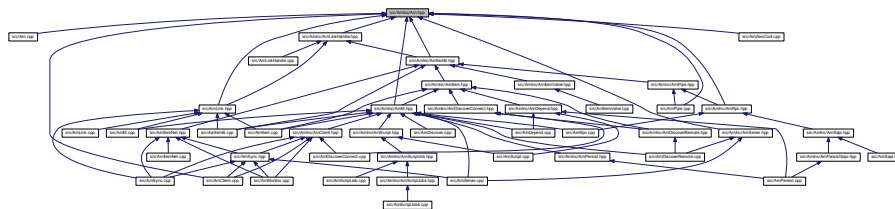
```
#include "ArnInc/ArnDiscoverRemote.hpp"
#include "ArnInc/ArnZeroConf.hpp"
#include "ArnInc/ArnServer.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnInc/ArnLib.hpp"
#include <QTimer>
#include <QMetaObject>
#include <QHostInfo>
#include <QNetworkInterface>
#include <QDir>
```

```
#include "MQFlags.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](#).
- struct [Arn::DataType](#)
Data type of an [Arn](#) Data Object
- struct [Arn::ObjectMode](#)
General global mode of an [Arn](#) Data Object
- struct [Arn::ObjectSyncMode](#)
The client session sync mode of an [Arn](#) Data Object
- struct [Arn::LinkFlags](#)
Link flags when accessing an [Arn](#) Data Object
- struct [Arn::NameF](#)
- struct [Arn::Coding](#)

Namespaces

- namespace [Arn](#)

Macros

- `#define` [DATASTREAM_VER](#) `QDataStream::Qt_4_6`

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::twinPath](#) (const QString &path)
Get the bidirectional twin to a given path
- QString [Arn::providerPath](#) (const QString &path, bool giveProviderPath=true)
Get provider path or requester path
- 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

15.14.1 Macro Definition Documentation

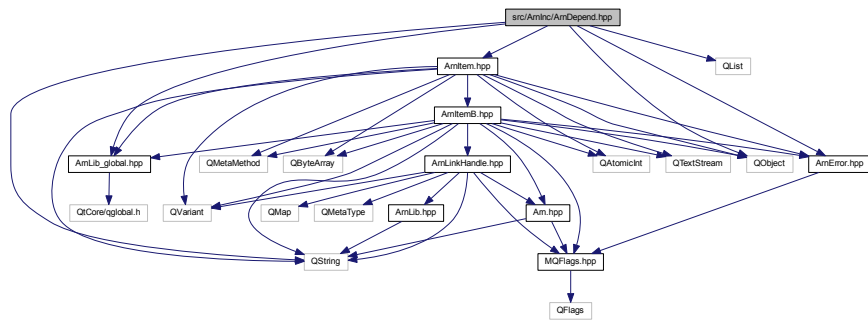
15.14.1.1 `#define DATASTREAM_VER QDataStream::Qt_4_6`

Definition at line 38 of file Arn.hpp.

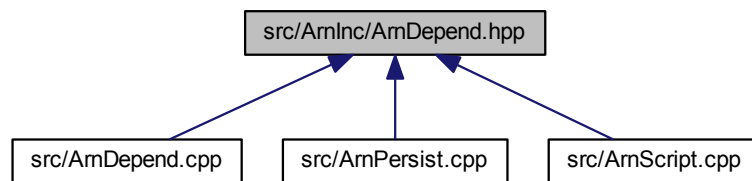
15.15 src/ArnInc/ArnClient.hpp File Reference

```
#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 ArnDepend.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnDependOffer](#)

Class for advertising that a service is available.

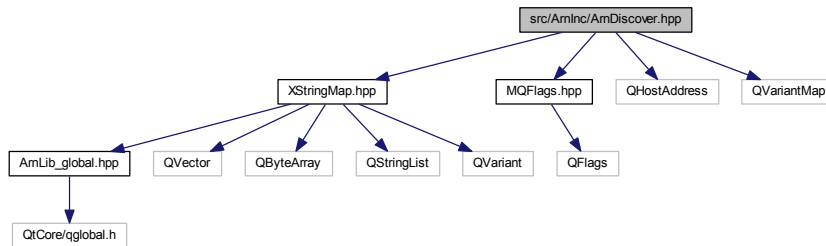
- class [ArnDepend](#)

Class for setting up dependencis to needed services.

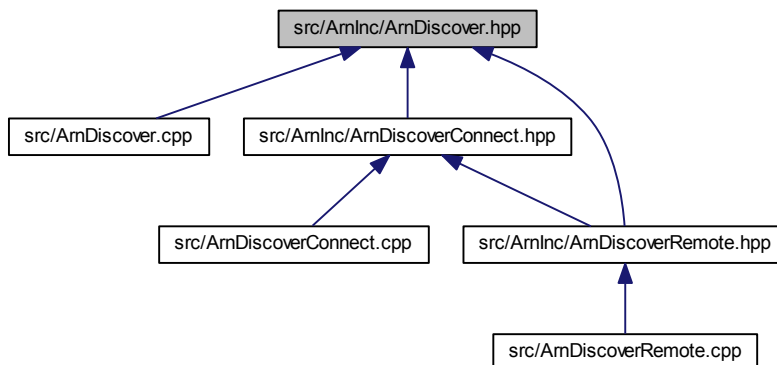
15.17 src/ArnInc/ArnDiscover.hpp File Reference

```
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QHostAddress>
#include <QVariantMap>
```

Include dependency graph for ArnDiscover.hpp:



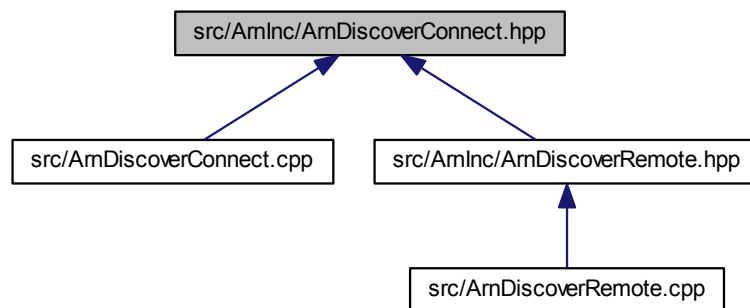
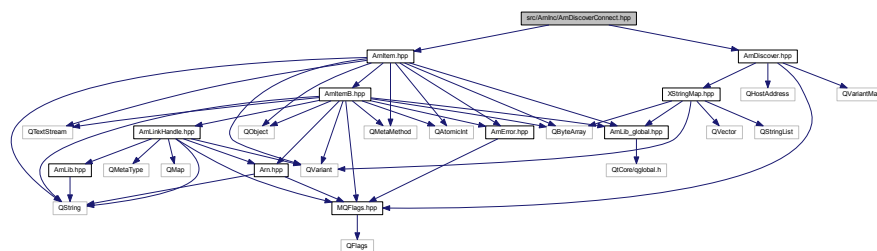
This graph shows which files directly or indirectly include this file:



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](#).

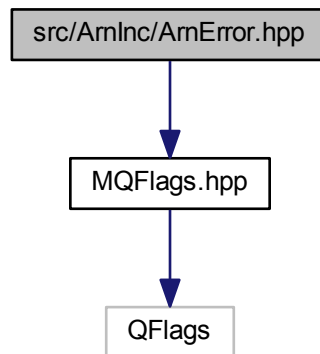
- namespace [ArnDiscover](#)



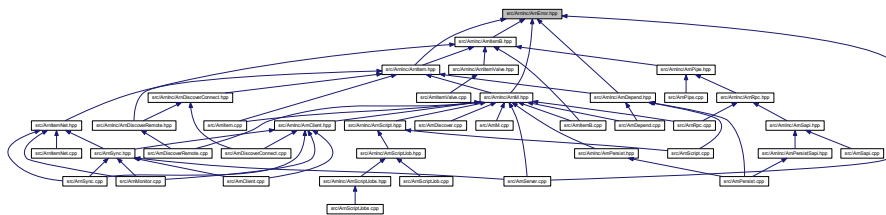
- class `ArnDiscoverConnector`

```
#include "ArnDiscover.hpp"
#include "ArnDiscoverConnect.hpp"
#include "ArnItem.hpp"
```


Include dependency graph for ArnError.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct [ArnError](#)
- struct [ArnError::StdCode](#)

15.21 src/ArnInc/ArnItem.hpp File Reference

```

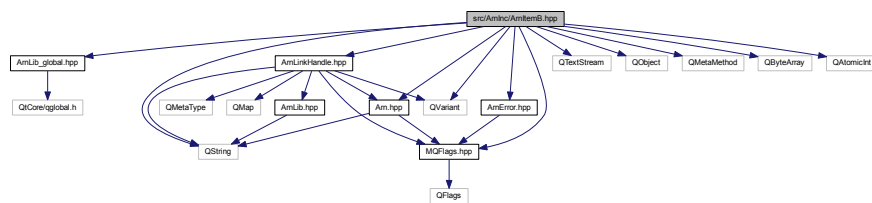
#include "ArnLib_global.hpp"
#include "ArnItemB.hpp"
#include "ArnError.hpp"
#include <QTextStream>
#include <QObject>
#include <QMetaMethod>
#include <QString>
#include <QByteArray>
#include <QVariant>
#include <QAtomicInt>

```

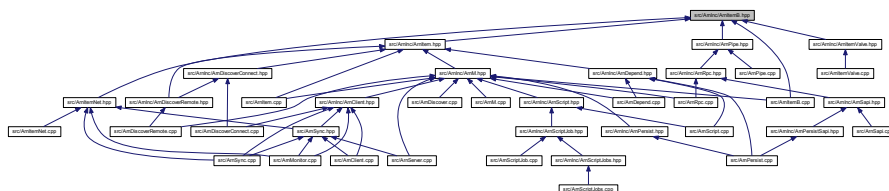

15.22 src/ArnInc/ArnItemB.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnLinkHandle.hpp"
#include "ArnError.hpp"
#include "Arn.hpp"
#include "MQFlags.hpp"
#include <QTextStream>
#include <QObject>
#include <QMetaMethod>
#include <QString>
#include <QByteArray>
#include <QVariant>
#include <QAtomicInt>
```

Include dependency graph for ArnItemB.hpp:



This graph shows which files directly or indirectly include this file:



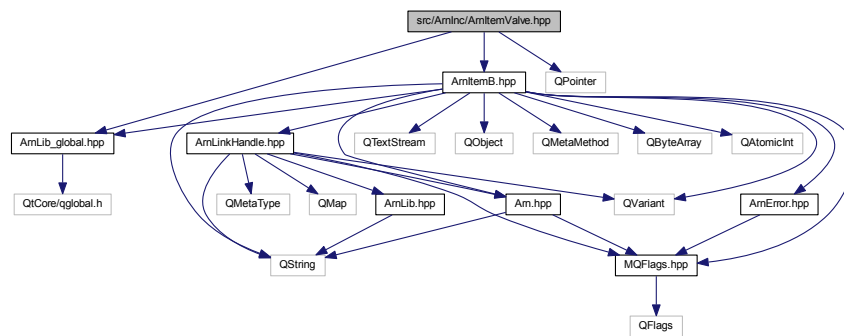
Classes

- class [ArnItemB](#)
Base class handle for an [Arn](#) Data Object.
- struct [ArnItemB::ExportCode](#)
Code used in blob for `arnExport()` and `arnImport()`

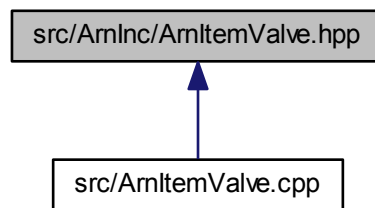
15.23 src/ArnInc/ArnItemValve.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnItemB.hpp"
#include <QPointer>
```

Include dependency graph for ArnItemValve.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnItemValve](#)

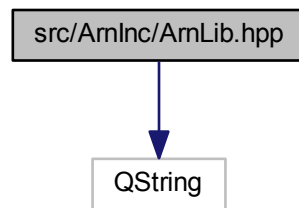
Valve for controlling stream to/from an [ArnItemB](#).

- struct [ArnItemValve::SwitchMode](#)

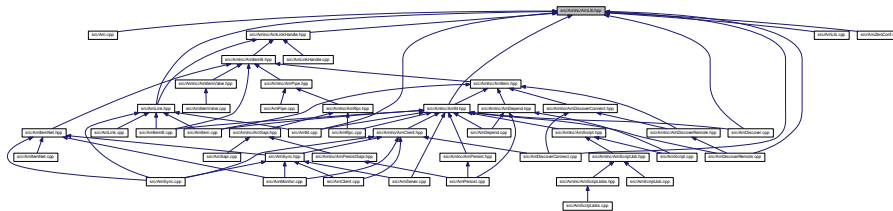
15.24 src/ArnInc/ArnLib.hpp File Reference

```
#include <QString>
```

Include dependency graph for ArnLib.hpp:



This graph shows which files directly or indirectly include this file:



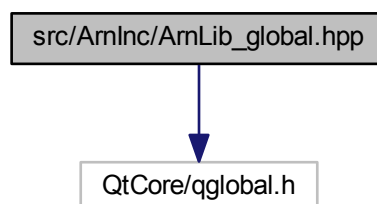
Namespaces

- namespace [Arn](#)

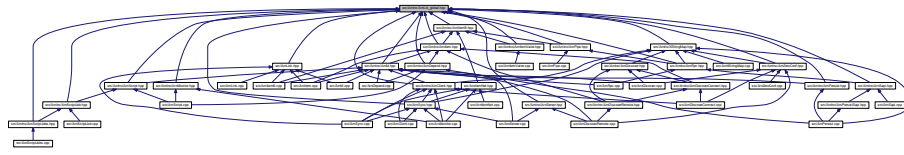
15.25 src/ArnInc/ArnLib_global.hpp File Reference

```
#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`

15.25.1 Macro Definition Documentation

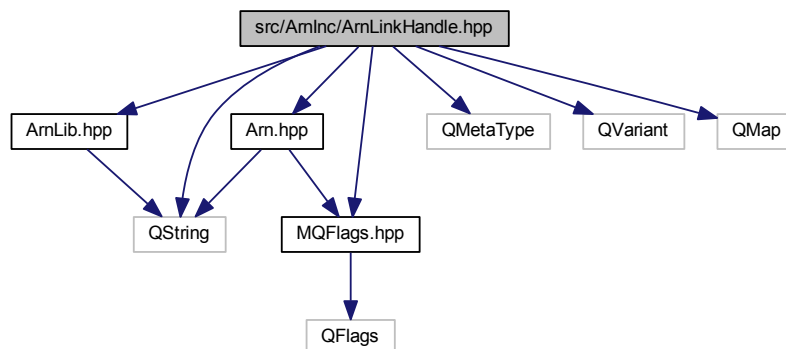
15.25.1.1 `#define ARNLIBSHARED_EXPORT Q_DECL_IMPORT`

Definition at line 11 of file ArnLib_global.hpp.

15.26 src/ArnInc/ArnLinkHandle.hpp File Reference

```
#include "ArnLib.hpp"
#include "Arn.hpp"
#include "MQFlags.hpp"
#include <QMetaType>
#include <QString>
#include <QVariant>
#include <QMap>
```

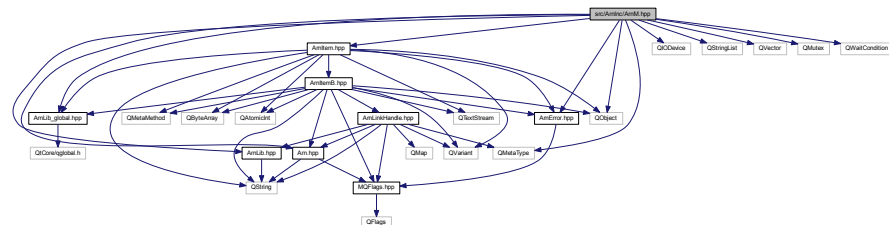
Include dependency graph for ArnLinkHandle.hpp:



[illegible]

```
#include "ArnLib_global.hpp"
#include "ArnLib.hpp"
#include "Arn.hpp"
#include "ArnError.hpp"
#include "ArnItem.hpp"
#include <QIODevice>
#include <QStringList>
#include <QVector>
#include <QMetaType>
#include <QObject>
#include <QMutex>
#include <QWaitCondition>
```

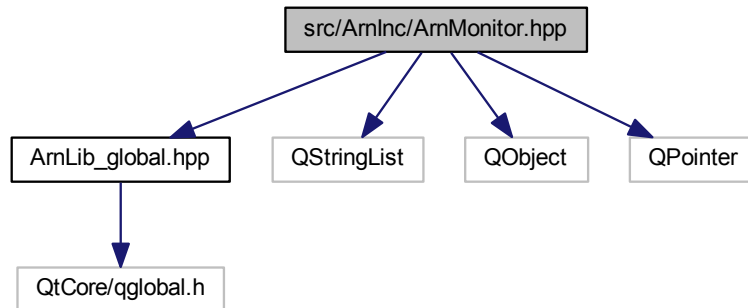
Include dependency graph for ArnM.hpp:



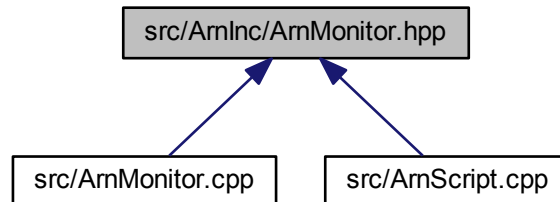
- class ArnM

```
#include "ArnLib_global.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

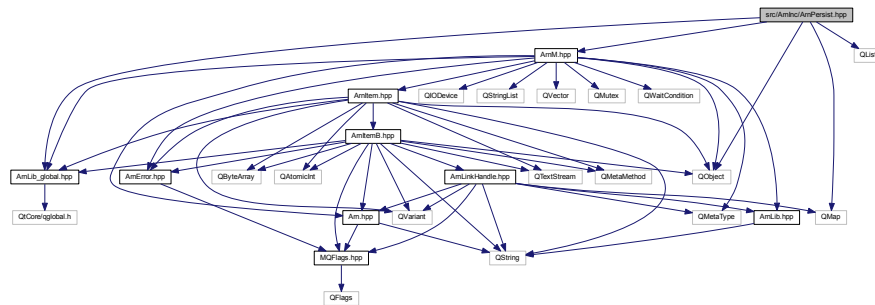
- class [ArnMonitor](#)

A client remote monitor to detect changes at server.

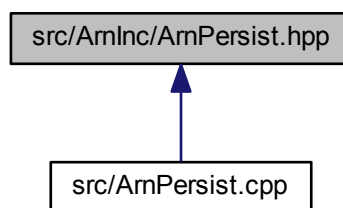
15.29 src/ArnInc/ArnPersist.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnM.hpp"
#include <QMap>
#include <QList>
#include <QObject>
```


Include dependency graph for ArnPersist.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **ArnPersist**

Namespaces

- namespace Arn

15.30 src/ArnInc/ArnPersistSapi.hpp File Reference

```
#include "ArnSapi.hpp"
```

[illegible]

```
graph BT; A[src/AmPersist.cpp] --> B[src/AmInc/AmPersistSapi.hpp];
```

The diagram illustrates a dependency between two source files. At the bottom, a white box with a black border contains the text `src/AmPersist.cpp`. A blue arrow points vertically upwards from this box to a gray box with a black border at the top, which contains the text `src/AmInc/AmPersistSapi.hpp`. This indicates that `src/AmPersist.cpp` depends on or includes `src/AmInc/AmPersistSapi.hpp`.

```
#include "ArnLib_global.hpp"
#include "ArnItemB.hpp"
Include dependency graph for ArnPipe.hpp:
```



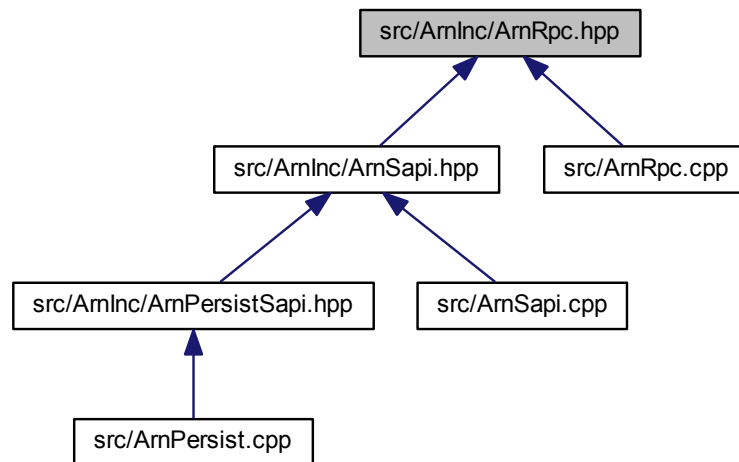
```
graph BT; src/AmInc/AmPipe.hpp[src/AmInc/AmPipe.hpp] --> src/AmInc/AmRpc.hpp[src/AmInc/AmRpc.hpp]; src/AmPipe.cpp[src/AmPipe.cpp] --> src/AmInc/AmPipe.hpp; src/AmInc/AmRpc.hpp --> src/AmInc/AmSapi.hpp[src/AmInc/AmSapi.hpp]; src/AmRpc.cpp[src/AmRpc.cpp] --> src/AmInc/AmRpc.hpp; src/AmInc/AmSapi.hpp --> src/AmInc/AmPersistSapi.hpp[src/AmInc/AmPersistSapi.hpp]; src/AmSapi.cpp[src/AmSapi.cpp] --> src/AmInc/AmSapi.hpp; src/AmPersist.cpp[src/AmPersist.cpp] --> src/AmInc/AmPersistSapi.hpp;
```

- class `ArnPipe`
ArnItem specialized as a pipe.

```
#include "ArnLib_global.hpp"
#include "Arn.hpp"
#include "ArnPipe.hpp"
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QGenericArgument>
#include <QPointer>
#include <QString>
#include <QByteArray>
#include <QObject>
```

[illegible]

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 [ArnRpc](#)
Remote Procedure Call.
- struct [ArnRpc::Mode](#)
- struct [ArnRpc::Invoke](#)
- struct [ArnRpc::RpcTypeInfo](#)
- struct [ArnRpc::ArgInfo](#)
- struct [ArnRpc::MethodsParam](#)
- 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)

15.32.1 Macro Definition Documentation

15.32.1.1 `#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 49 of file ArnRpc.hpp.

15.32.1.2 #define no_queue

Examples:

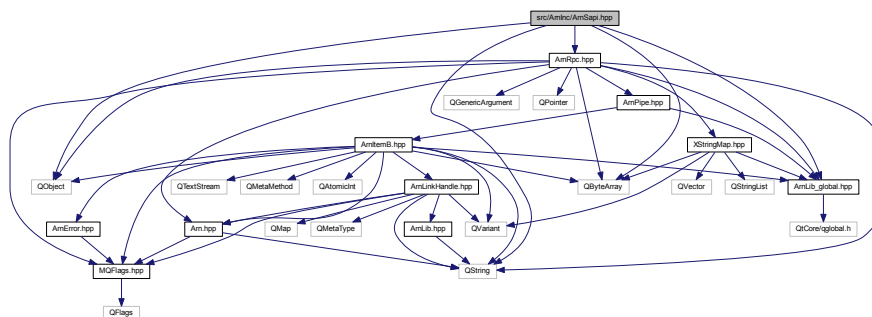
ArnDemoChatServer/ChatSapi.hpp.

Definition at line 35 of file ArnRpc.hpp.

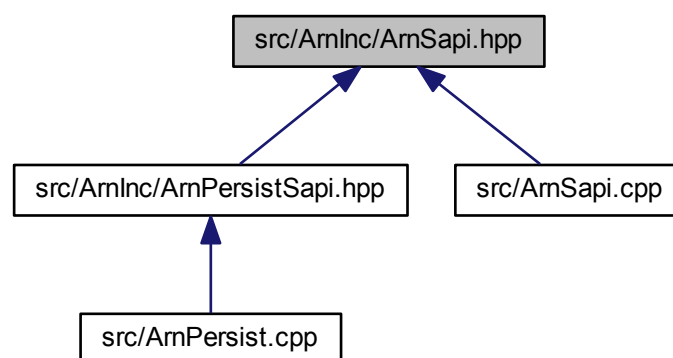
15.33 src/ArnInc/ArnSapi.hpp File Reference

```
#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`

15.33.1 Macro Definition Documentation

15.33.1.1 `#define MQ_PUBLIC_ACCESS`

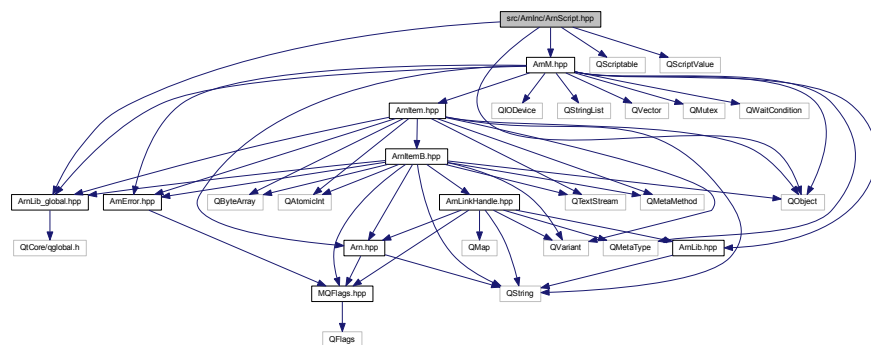
Examples:

[ArnDemoChatServer/ChatSapi.hpp](#).

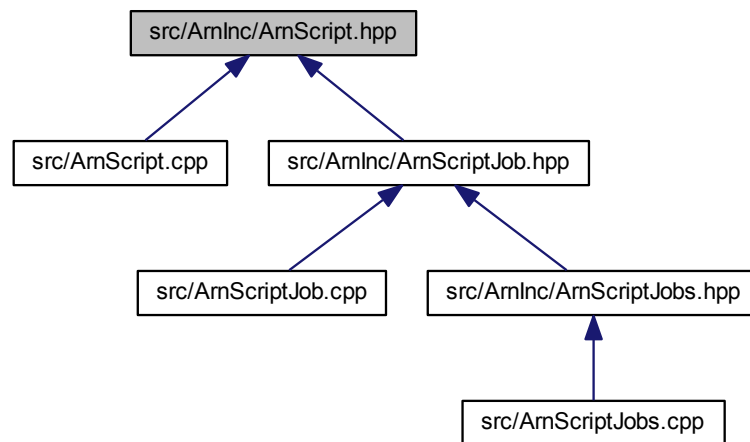
Definition at line 44 of file ArnSapi.hpp.

15.34 src/ArnInc/ArnScript.hpp File Reference

```
#include "ArnLib_global.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:



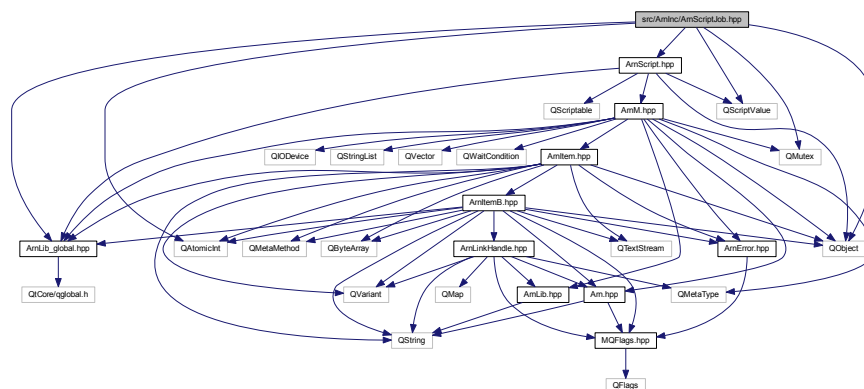
Classes

- class **ArnScript**

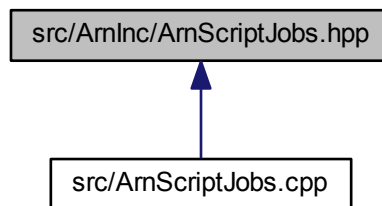
15.35 src/ArnInc/ArnScriptJob.hpp File Reference

```
#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:



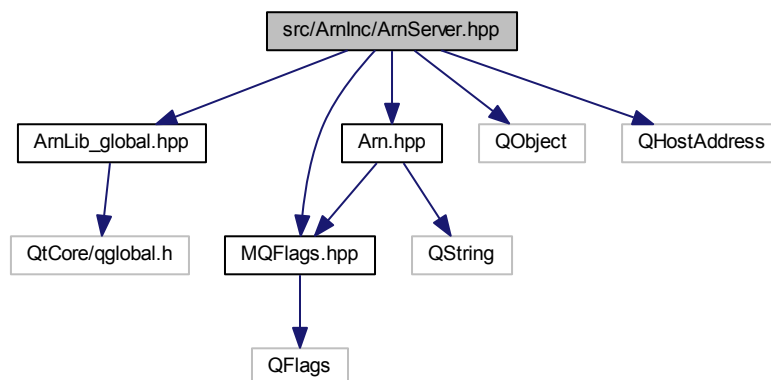
Classes

- class [ArnScriptJobs](#)
- struct [ArnScriptJobs::Type](#)
- struct [ArnScriptJobs::JobSlot](#)

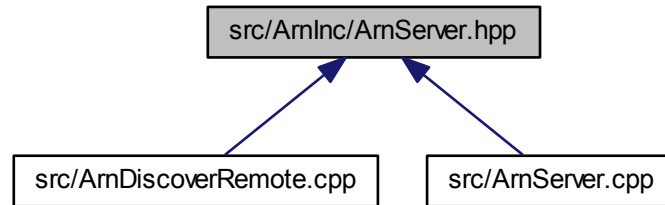
15.37 src/ArnInc/ArnServer.hpp File Reference

```
#include "ArnLib_global.hpp"  
#include "Arn.hpp"  
#include "MQFlags.hpp"  
#include <QObject>  
#include <QHostAddress>
```

Include dependency graph for ArnServer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ArnServer](#)

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

- struct [ArnServer::Type](#)

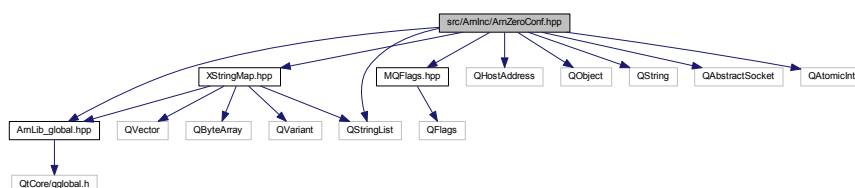
15.38 src/ArnInc/ArnZeroConf.hpp File Reference

```

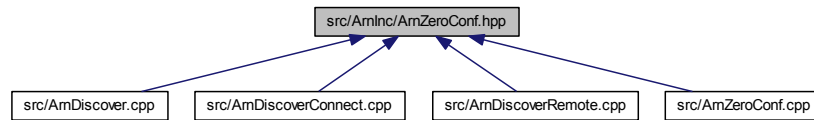
#include "ArnLib_global.hpp"
#include "XStringMap.hpp"
#include "MQFlags.hpp"
#include <QHostAddress>
#include <QObject>
#include <QStringList>
#include <QString>
#include <QAbstractSocket>
#include <QAtomicInt>

```

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

- namespace [ArnZeroConf](#)

Typedefs

- typedef struct _DNSServiceRef_t * [DNSServiceRef](#)

15.38.1 Typedef Documentation

15.38.1.1 typedef struct _DNSServiceRef_t* DNSServiceRef

Definition at line 45 of file ArnZeroConf.hpp.

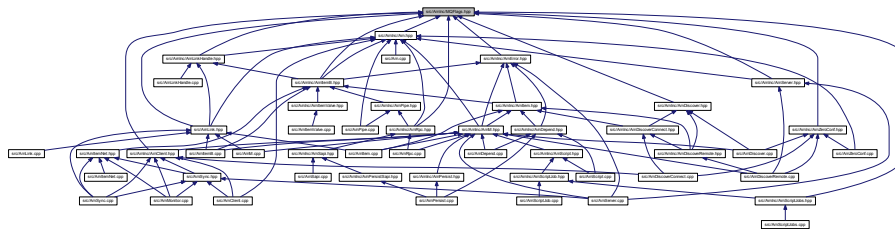
15.39 src/ArnInc/MQFlags.hpp File Reference

```
#include <QFlags>
```

Include dependency graph for MQFlags.hpp:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MQ_DECLARE_FLAGS(FEStruct)`
Flags.
- `#define MQ_DECLARE_OPERATORS_FOR_FLAGS(FEStruct) Q_DECLARE_OPERATORS_FOR_FLAGS(FEStruct::F)`
- `#define MQ_DECLARE_ENUM(EStruct)`
Enums.

15.39.1 Macro Definition Documentation

15.39.1.1 `#define MQ_DECLARE_ENUM(EStruct)`

Value:

```

E e; \
    inline EStruct(E v_ = E(0)) : e( v_) {} \
    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;}
  
```

Enums.

Definition at line 57 of file MQFlags.hpp.

15.39.1.2 #define MQ_DECLARE_FLAGS(*FEStruct*)

Value:

```
Q_DECLARE_FLAGS(F, E) \
    F f; \
    inline FStruct(F v_ = F(0)) : f( v_) {} \
    inline FStruct(E e_) : f( e_) {} \
    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;}
```

Flags.

Definition at line 38 of file MQFlags.hpp.

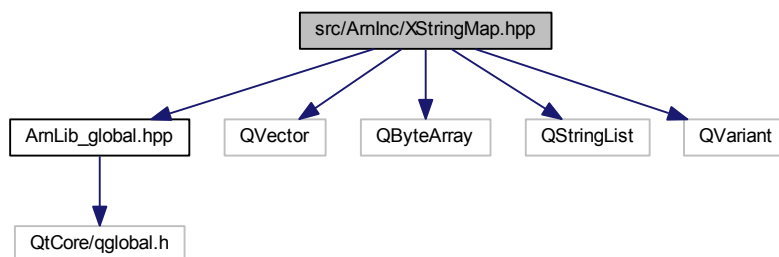
```
15.39.1.3 #define MQ_DECLARE_OPERATORS_FOR_FLAGS( FEStruct ) Q_DECLARE_OPERATORS_FOR_FLAGS( FEStruct::F)
```

Definition at line 52 of file MQFlags.hpp.

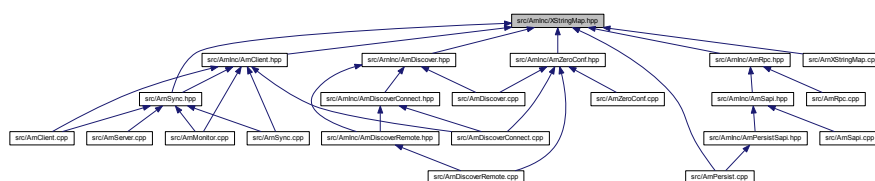
15.40 src/ArnInc/XStringMap.hpp File Reference

```
#include "ArnLib_global.hpp"
#include <QVector>
#include <QByteArray>
#include <QStringList>
#include <QVariant>
```

Include dependency graph for XStringMap.hpp:

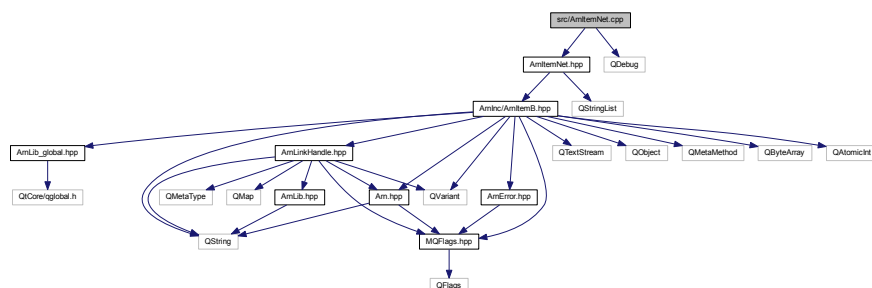


This graph shows which files directly or indirectly include this file:

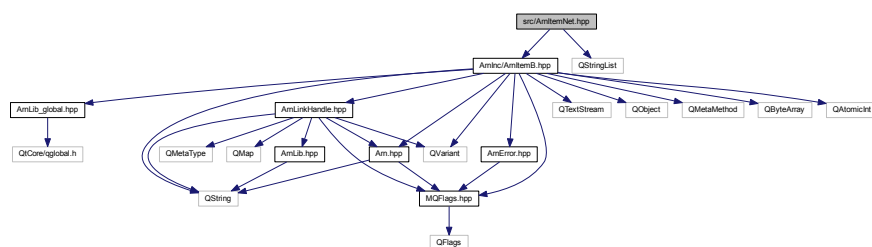


[illegible]

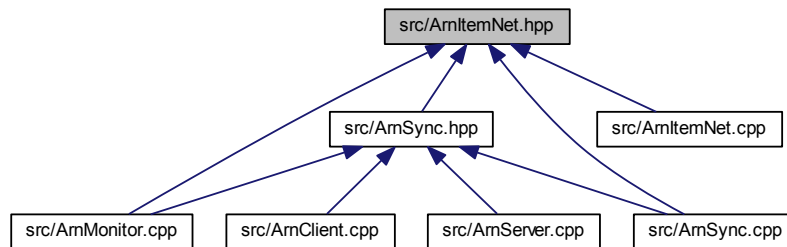
```
#include "ArnItemNet.hpp"
#include <QDebug>
Include dependency graph for ArnItemNet.cpp:
```



```
#include "ArnInc/ArnItemB.hpp"
#include <QStringList>
Include dependency graph for ArnItemNet.hpp:
```



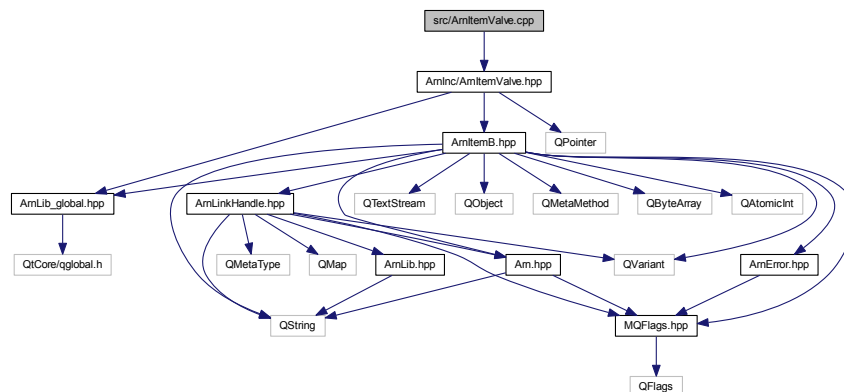
This graph shows which files directly or indirectly include this file:



15.45 src/ArnItemValve.cpp File Reference

```
#include "ArnInc/ArnItemValve.hpp"
```

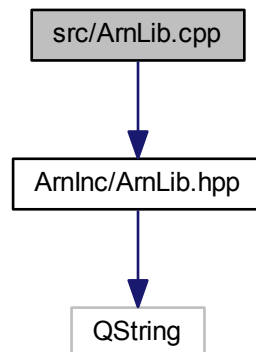
Include dependency graph for ArnItemValve.cpp:



15.46 src/ArnLib.cpp File Reference

```
#include "ArnInc/ArnLib.hpp"
```


Include dependency graph for ArnLib.cpp:



Namespaces

- namespace [Arn](#)

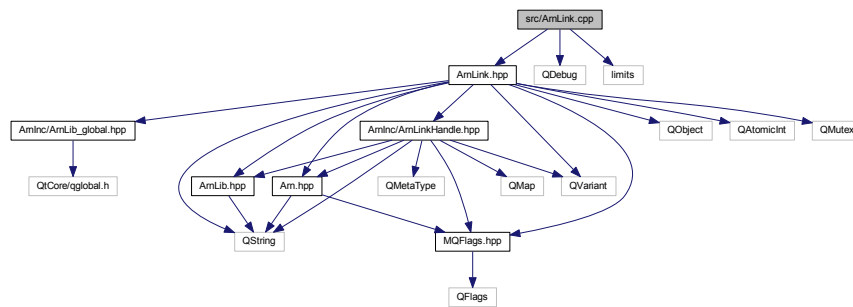
Variables

- bool [Arn::debugThreading](#) = false
- bool [Arn::debugLinkRef](#) = false
- bool [Arn::debugLinkDestroy](#) = false
- bool [Arn::debugReclnOut](#) = false
- bool [Arn::debugShareObj](#) = false
- bool [Arn::debugMonitor](#) = false
- bool [Arn::debugMonitorTest](#) = false
- bool [Arn::debugRPC](#) = false
- bool [Arn::debugDepend](#) = 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/"

15.47 src/ArnLink.cpp File Reference

```
#include "ArnLink.hpp"
#include <QDebug>
#include <limits>
```

Include dependency graph for ArnLink.cpp:



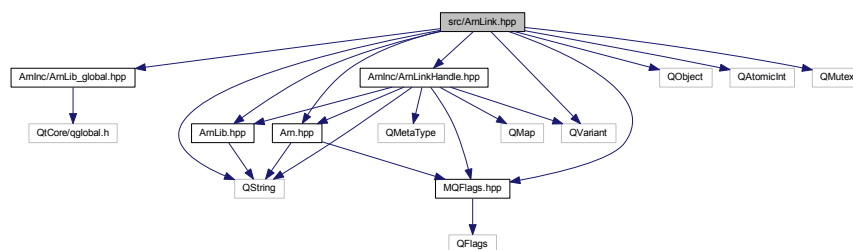
15.48 src/ArnLink.hpp File Reference

```

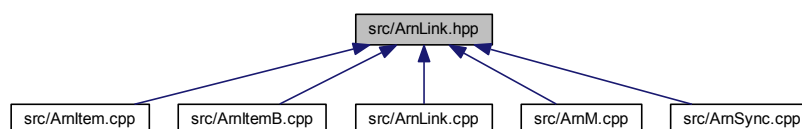
#include "ArnInc/ArnLib_global.hpp"
#include "ArnInc/ArnLinkHandle.hpp"
#include "ArnInc/ArnLib.hpp"
#include "ArnInc/Arn.hpp"
#include "ArnInc/MQFlags.hpp"
#include <QObject>
#include <QString>
#include <QVariant>
#include <QAtomicInt>
#include <QMutex>

```

Include dependency graph for ArnLink.hpp:



This graph shows which files directly or indirectly include this file:



Variables

- `const int arnDbSaveVer = 200`

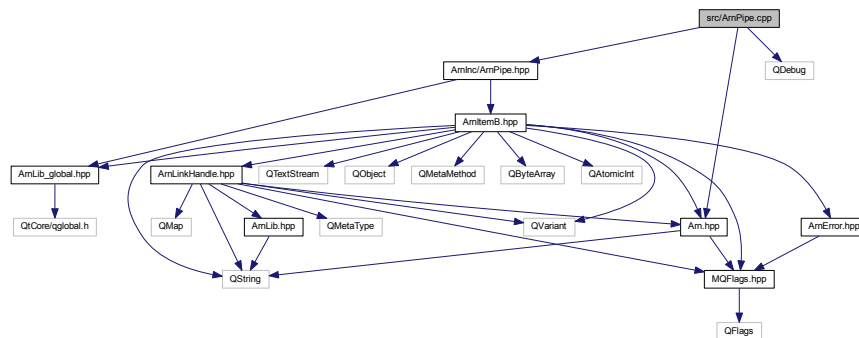
15.52.1 Variable Documentation

15.52.1.1 `const int arnDbSaveVer = 200`

Definition at line 52 of file ArnPersist.cpp.

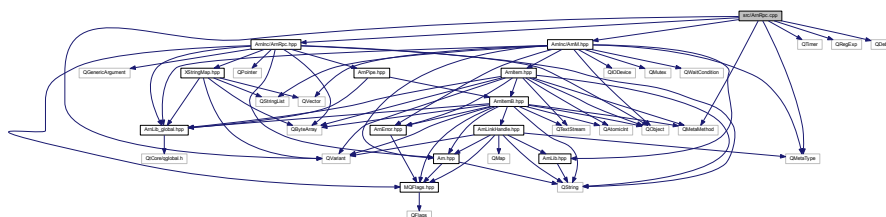
15.53 src/ArnPipe.cpp File Reference

```
#include "ArnInc/ArnPipe.hpp"
#include "ArnInc/Arn.hpp"
#include <QDebug>
Include dependency graph for ArnPipe.cpp:
```



15.54 src/ArnRpc.cpp File Reference

```
#include "ArnInc/ArnRpc.hpp"
#include "ArnInc/ArnM.hpp"
#include <QMetaType>
#include <QMetaMethod>
#include <QTimer>
#include <QRegExp>
#include <QVariant>
#include <QDebug>
Include dependency graph for ArnRpc.cpp:
```



[illegible]

```
#include "ArnInc/ArnServer.hpp"
#include "ArnInc/ArnError.hpp"
#include "ArnInc/ArnM.hpp"
#include "ArnSync.hpp"
#include <QTcpServer>
#include <QTcpSocket>
#include <QHostInfo>
#include <QNetworkInterface>
#include <QDebug>
```

[illegible]

```
#include "ArnSync.hpp"
#include "ArnItemNet.hpp"
#include "ArnLink.hpp"
#include "ArnInc/ArnClient.hpp"
#include <QTcpSocket>
#include <QString>
#include <QStringList>
#include <QDebug>
#include <limits.h>
```



```
#include "ArnInc/ArnLib_global.hpp"
#include "ArnInc/ArnClient.hpp"
#include "ArnInc/XStringMap.hpp"
#include "ArnItemNet.hpp"
#include <QObject>
#include <QByteArray>
#include <QMap>
#include <QQueue>
```

[illegible]

```

graph BT
    AmClient[src/AmClient.cpp] --> AmSync[src/AmSync.hpp]
    AmMonitor[src/AmMonitor.cpp] --> AmSync
    AmServer[src/AmServer.cpp] --> AmSync
    AmSyncC[src/AmSync.cpp] --> AmSync
  
```

Macros

- `#define ARNRECNAME ""`

15.61.1 Macro Definition Documentation

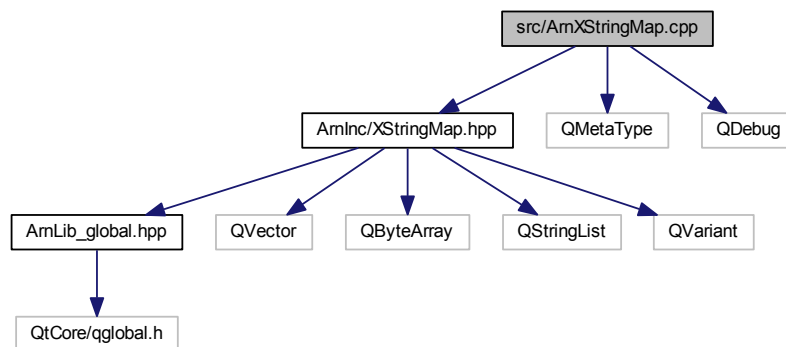
15.61.1.1 `#define ARNRECNAME ""`

Definition at line 44 of file ArnSync.hpp.

15.62 src/ArnXStringMap.cpp File Reference

```
#include "ArnInc/XStringMap.hpp"
#include <QMetaType>
#include <QDebug>
```

Include dependency graph for ArnXStringMap.cpp:



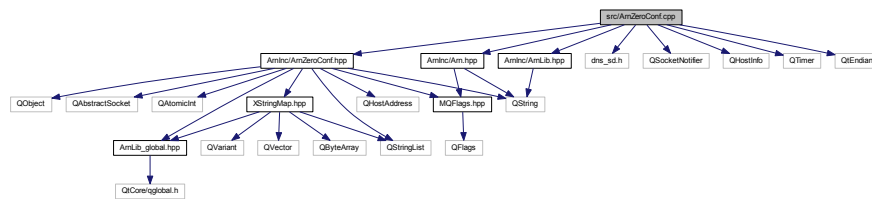
Namespaces

- namespace `Arn`

15.63 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:



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.
//
#ifndef MAINWINDOW_HPP
#define MAINWINDOW_HPP

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

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

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

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

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

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

#endif // MAINWINDOW_HPP
```

16.4 ArnDemoChatServer/ChatSapi.hpp

Demo Chat Server

```
// Copyright (C) 2010-2014 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their
// own
// licenses. ArnDemoChat is independent of these licenses; however, use of
// these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#ifndef CHATSAPI_HPP
#define CHATSAPI_HPP

#include <ArnInc/ArnSapi.hpp>

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

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

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

#endif // CHATSAPI_HPP
```

16.5 ArnDemoChatServer/main.cpp

Demo Chat Server

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

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

    return a.exec();
}
```

16.6 ArnDemoChatServer/MainWindow.cpp

Demo Chat Server


```

// Copyright (C) 2010-2014 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@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/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.
//
#ifdef 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

- ~ArnDepend
 - ArnDepend, [59](#)
- ~ArnItem
 - ArnItem, [103](#)
- ~ArnItemB
 - ArnItemB, [116](#)
- ~ArnPersist
 - ArnPersist, [138](#)
- ~ArnPipe
 - ArnPipe, [143](#)
- ~ArnScriptJobFactory
 - ArnScriptJobFactory, [165](#)
- ~ArnZeroConfB
 - ArnZeroConfB, [169](#)
- ~ArnZeroConfBrowser
 - ArnZeroConfBrowser, [174](#)
- ~ArnZeroConfLookup
 - ArnZeroConfLookup, [180](#)
- ~ArnZeroConfRegister
 - ArnZeroConfRegister, [186](#)
- ~ArnZeroConfResolve
 - ArnZeroConfResolve, [195](#)
- ~XStringMap
 - Arn::XStringMap, [217](#)
- _arnClient
 - ArnMonitor, [137](#)
- _depOfferProto
 - ArnScript, [160](#)
- _depProto
 - ArnScript, [160](#)
- _engine
 - ArnScript, [160](#)
- _itemProto
 - ArnScript, [160](#)
- _monitorPath
 - ArnMonitor, [137](#)
- _monitorProto
 - ArnScript, [160](#)
- ARNLIBSHARED_EXPORT
 - ArnLib_global.hpp, [242](#)
- ARNRECNAM
 - ArnSync.hpp, [270](#)
- Accept
 - Arn::SameValue, [209](#)
- activeServiceNames
 - ArnZeroConfBrowser, [174](#)
- add
 - Arn::XStringMap, [217](#), [218](#)
 - ArnDepend, [60](#)

- addConfig
 - ArnScriptJobControl, [163](#)
- addCustomProperty
 - ArnDiscoverAdvertise, [64](#)
- addGroup
 - ArnDiscoverAdvertise, [65](#)
- addInterface
 - ArnScriptJobControl, [163](#)
- addInterfaceList
 - ArnScriptJobControl, [163](#)
- addJob
 - ArnScriptJobs, [166](#)
- addMode
 - ArnItem, [103](#)
- addMountPoint
 - ArnClient, [55](#)
- addPath
 - Arn, [44](#)
- addSenderSignals
 - ArnRpc, [149](#)
- addSubType
 - ArnZeroConfRegister, [187](#)
- addToArnList
 - ArnClient, [55](#)
- addToDirectHosts
 - ArnDiscoverConnector, [79](#)
- addr
 - ArnClient::HostAddrPort, [202](#)
- Advertise
 - ArnDiscoverAdvertise::State, [211](#)
- advertise
 - ArnDependOffer, [61](#)
- advertiseService
 - ArnDiscoverAdvertise, [65](#)
- Advertising
 - ArnDiscoverAdvertise::State, [211](#)
- allMethodIds
 - ArnRpc::MethodsParam::Params, [209](#)
- AlreadyExist
 - ArnError, [99](#)
- AlreadyOpen
 - ArnError, [99](#)
- append
 - Arn::XStringMap, [218](#)
- Arn, [43](#)
 - addPath, [44](#)
 - changeBasePath, [45](#)
 - childPath, [45](#)
 - convertName, [45](#)

- convertPath, [46](#)
- debugDepend, [49](#)
- debugDiscover, [49](#)
- debugLinkDestroy, [49](#)
- debugLinkRef, [49](#)
- debugMDNS, [50](#)
- debugMonitor, [50](#)
- debugMonitorTest, [50](#)
- debugRPC, [50](#)
- debugRecInOut, [50](#)
- debugShareObj, [50](#)
- debugThreading, [50](#)
- debugZeroConf, [50](#)
- defaultTcpPort, [50](#)
- fullPath, [46](#)
- hostFromHostWithInfo, [46](#)
- isFolderPath, [47](#)
- isProviderPath, [47](#)
- itemName, [47](#)
- makeHostWithInfo, [48](#)
- makePath, [48](#)
- pathDiscover, [50](#)
- pathDiscoverConnect, [50](#)
- pathDiscoverThis, [50](#)
- pathLocal, [51](#)
- pathLocalSys, [51](#)
- providerPath, [48](#)
- resourceArnLib, [51](#)
- resourceArnRoot, [51](#)
- twinPath, [49](#)
- warningMDNS, [51](#)
- Arn.hpp
 - DATASTREAM_VER, [231](#)
- Arn::Coding
 - Binary, [199](#)
 - Text, [199](#)
- Arn::DataType
 - ByteArray, [200](#)
 - Double, [200](#)
 - Int, [200](#)
 - Null, [200](#)
 - String, [200](#)
 - Variant, [200](#)
- Arn::LinkFlags
 - CreateAllowed, [203](#)
 - Folder, [203](#)
 - SilentError, [203](#)
 - Threaded, [203](#)
- Arn::NameF
 - EmptyOk, [207](#)
 - NoFolderMark, [207](#)
 - Relative, [207](#)
- Arn::ObjectMode
 - BiDir, [207](#)
 - Pipe, [207](#)
 - Save, [208](#)
- Arn::ObjectSyncMode
 - AutoDestroy, [208](#)
 - Master, [208](#)
 - Monitor, [208](#)
 - Normal, [208](#)
- Arn::SameValue
 - Accept, [209](#)
 - DefaultAction, [209](#)
 - Ignore, [209](#)
- ArnClient::ConnectStat
 - Connected, [199](#)
 - Connecting, [199](#)
 - Disconnected, [199](#)
 - Error, [199](#)
 - Init, [199](#)
 - TriedAll, [199](#)
- ArnDiscover::Type
 - Client, [214](#)
 - None, [214](#)
 - Server, [214](#)
- ArnDiscoverAdvertise::State
 - Advertise, [211](#)
 - Advertising, [211](#)
 - None, [211](#)
 - StartupAdvertise, [211](#)
- ArnDiscoverInfo::State
 - HostInfo, [211](#)
 - HostInfoErr, [211](#)
 - HostIp, [212](#)
 - HostIpErr, [212](#)
 - Init, [211](#)
 - ServiceName, [211](#)
- ArnError
 - AlreadyExist, [99](#)
 - AlreadyOpen, [99](#)
 - ConnectionError, [99](#)
 - CreateError, [99](#)
 - Err_N, [99](#)
 - FolderNotOpen, [99](#)
 - Info, [98](#)
 - ItemNotOpen, [99](#)
 - ItemNotSet, [99](#)
 - NotFound, [99](#)
 - NotMainThread, [99](#)
 - NotOpen, [99](#)
 - Ok, [98](#)
 - RecUnknown, [99](#)
 - Retired, [99](#)
 - RpcInvokeError, [99](#)
 - RpcReceiveError, [99](#)
 - ScriptError, [99](#)
 - Undef, [99](#)
 - Warning, [98](#)
- ArnError::StdCode
 - Err_Custom, [212](#)
 - Err_Undef, [212](#)
 - Info, [212](#)
 - Ok, [212](#)
 - Warning, [212](#)
- ArnItemB::ExportCode

- ByteArray, [201](#)
 - String, [201](#)
 - Variant, [201](#)
 - VariantBin, [201](#)
 - VariantTxt, [201](#)
- ArnItemValve::SwitchMode
 - InOutputStream, [213](#)
 - InStream, [213](#)
 - OutStream, [213](#)
- ArnRpc::Invoke
 - NoQueue, [203](#)
- ArnRpc::Mode
 - AutoDestroy, [204](#)
 - CheckSequence, [204](#)
 - Debug, [204](#)
 - NamedArg, [204](#)
 - NamedTypedArg, [204](#)
 - NoDefaultArgs, [204](#)
 - OnlyPosArgIn, [204](#)
 - Provider, [204](#)
 - SendSequence, [204](#)
 - UseDefaultCall, [204](#)
 - UuidAutoDestroy, [204](#)
 - UuidPipe, [204](#)
- ArnScriptJobs::Type
 - Cooperative, [214](#)
 - Null, [214](#)
 - Preemptive, [214](#)
- ArnServer::Type
 - NetSync, [213](#)
- ArnZeroConf::Error
 - BadReqSeq, [201](#)
 - Ok, [201](#)
 - Running, [201](#)
 - Timeout, [201](#)
 - UDnsFail, [201](#)
- ArnZeroConf::State
 - Browsing, [210](#)
 - InProgress, [210](#)
 - LookingUp, [210](#)
 - Lookup, [210](#)
 - Lookuped, [210](#)
 - None, [210](#)
 - Register, [210](#)
 - Registered, [210](#)
 - Registering, [210](#)
 - Resolve, [210](#)
 - Resolved, [210](#)
 - Resolving, [210](#)
- Arn::Coding, [198](#)
 - E, [199](#)
- Arn::DataType, [199](#)
 - E, [200](#)
- Arn::LinkFlags, [203](#)
 - E, [203](#)
- Arn::NameF, [207](#)
 - E, [207](#)
- Arn::ObjectMode, [207](#)
 - E, [207](#)
- Arn::ObjectSyncMode, [208](#)
 - E, [208](#)
- Arn::SameValue, [209](#)
 - E, [209](#)
- Arn::XStringMap, [215](#)
 - ~XStringMap, [217](#)
 - add, [217](#), [218](#)
 - append, [218](#)
 - clear, [219](#)
 - fromXString, [219](#)
 - indexOf, [219](#)
 - indexOfValue, [219](#)
 - key, [219](#)
 - keyRef, [219](#)
 - keyString, [220](#)
 - keys, [219](#)
 - maxEnumOf, [220](#)
 - operator+=, [220](#)
 - remove, [220](#)
 - set, [220](#), [221](#)
 - setEmptyKeysToValue, [221](#)
 - size, [221](#)
 - squeeze, [221](#)
 - stringCode, [221](#)
 - stringDecode, [221](#)
 - toVariantMap, [221](#)
 - toXString, [221](#)
 - value, [221](#), [222](#)
 - valueRef, [222](#)
 - valueString, [222](#)
 - values, [222](#)
 - XStringMap, [217](#)
- arnChildFound
 - ArnMonitor, [133](#)
- arnChildFoundFolder
 - ArnMonitor, [134](#)
- arnChildFoundLeaf
 - ArnMonitor, [134](#)
- ArnClient, [53](#)
 - addMountPoint, [55](#)
 - addToArnList, [55](#)
 - ArnClient, [54](#)
 - arnList, [55](#)
 - ArnClient, [54](#)
 - clearArnList, [56](#)
 - connectStatus, [56](#)
 - connectToArn, [56](#)
 - connectToArnList, [57](#)
 - connectionStatusChanged, [56](#)
 - HostList, [54](#)
 - removeMountPoint, [57](#)
 - setAutoConnect, [57](#)
 - setMountPoint, [57](#)
 - tcpConnected, [58](#)
 - tcpDisConnected, [58](#)
 - tcpError, [58](#)
- ArnClient::ConnectStat, [199](#)

- E, 199
- ArnClient::HostAddrPort, 202
 - addr, 202
 - HostAddrPort, 202
 - port, 202
- arnDbSaveVer
 - ArnPersist.cpp, 265
- ArnDepend, 58
 - ~ArnDepend, 59
 - add, 60
 - ArnDepend, 59
 - ArnDepend, 59
 - completed, 60
 - DepSlot, 59
 - setMonitorName, 60
 - startMonitor, 60
- ArnDepend.cpp
 - ArnDependPath, 228
- ArnDependOffer, 61
 - advertise, 61
 - ArnDependOffer, 61
 - ArnDependOffer, 61
 - setStateId, 61
 - setStateName, 62
 - stateId, 62
 - stateName, 62
- ArnDependPath
 - ArnDepend.cpp, 228
- ArnDiscover, 51
- ArnDiscover::Type, 213
 - E, 214
- ArnDiscoverAdvertise, 62
 - addCustomProperty, 64
 - addGroup, 65
 - advertiseService, 65
 - ArnDiscoverAdvertise, 64
 - ArnDiscoverAdvertise, 64
 - currentService, 65
 - customProperties, 66
 - groups, 66
 - service, 66
 - serviceChangeError, 67
 - serviceChanged, 66
 - setCustomProperties, 67
 - setGroups, 67
 - setService, 68
 - state, 68
- ArnDiscoverAdvertise::State, 210
 - E, 211
- ArnDiscoverBrowser, 69
 - ArnDiscoverBrowser, 70
 - ArnDiscoverBrowser, 70
 - browse, 70
 - isBrowsing, 71
 - setFilter, 71
 - stopBrowse, 71
- ArnDiscoverBrowserB, 72
 - ArnDiscoverBrowserB, 73
- ArnDiscoverBrowserB, 73
- ArnDiscoverInfo, 89
 - defaultStopState, 73
 - goTowardState, 73
 - IdToIndex, 74
 - indexTold, 74
 - infoById, 74
 - infoByIndex, 75
 - infoByName, 75
 - infoUpdated, 75
 - serviceAdded, 76
 - serviceCount, 76
 - serviceNameTold, 76
 - serviceRemoved, 77
 - setDefaultStopState, 77
- ArnDiscoverConnector, 77
 - addToDirectHosts, 79
 - ArnDiscoverConnector, 79
 - ArnDiscoverConnector, 79
 - clearDirectHosts, 79
 - clientReadyToConnect, 79
 - directHostPrio, 80
 - discoverHostPrio, 80
 - externalClientConnect, 80
 - id, 80
 - resolveRefreshTimeout, 81
 - service, 81
 - setDirectHostPrio, 81
 - setDiscoverHostPrio, 82
 - setExternalClientConnect, 82
 - setResolveRefreshTimeout, 83
 - setResolver, 82
 - setService, 83
 - start, 83
- ArnDiscoverInfo, 84
 - ArnDiscoverBrowserB, 89
 - ArnDiscoverInfo, 85
 - ArnDiscoverInfo, 85
 - domain, 85
 - groups, 85
 - hostIp, 86
 - hostIpString, 86
 - hostName, 86
 - hostPort, 86
 - hostPortString, 86
 - hostWithInfo, 87
 - inProgress, 87
 - isError, 87
 - properties, 87
 - resolvCode, 88
 - serviceName, 88
 - state, 88
 - stopState, 88
 - type, 89
 - typeString, 89
- ArnDiscoverInfo::State, 211
 - E, 211
- ArnDiscoverRemote, 90

- ArnDiscoverRemote, 92
- ArnDiscoverRemote, 92
- clientReadyToConnect, 92
- defaultService, 92
- initialServiceTimeout, 92
- newConnector, 93
- setDefaultService, 93
- setInitialServiceTimeout, 93
- setService, 94
- startUseNewServer, 94
- startUseServer, 94
- ArnDiscoverResolver, 95
 - ArnDiscoverResolver, 97
 - ArnDiscoverResolver, 97
 - defaultService, 97
 - resolve, 97
 - setDefaultService, 98
- ArnError, 98
 - E, 98
- ArnError::StdCode, 212
 - E, 212
- arnExport
 - ArnItem, 103
- arnImport
 - ArnItem, 103
- ArnItem, 99
 - ~ArnItem, 103
 - addMode, 103
 - arnExport, 103
 - arnImport, 103
 - ArnItem, 102, 103
 - arnItemCreated, 104
 - arnModeChanged, 104
 - ArnItem, 102, 103
 - changed, 104, 105
 - getMode, 105
 - isAutoDestroy, 105
 - isBiDir, 106
 - isBiDirMode, 106
 - isFolder, 106
 - isIgnoreSameValue, 106
 - isMaster, 107
 - isPipeMode, 107
 - isSaveMode, 107
 - isTemplate, 107
 - modeChanged, 108
 - openFolder, 108
 - openUuid, 108
 - openUuidPipe, 108
 - operator=, 109
 - setAutoDestroy, 109
 - setBiDirMode, 109
 - setDelay, 110
 - setIgnoreSameValue, 110
 - setMaster, 110
 - setPipeMode, 110
 - setSaveMode, 110
 - setTemplate, 111
 - setValue, 111–113
 - syncMode, 113
 - toBool, 114
 - toByteArray, 114
 - toDouble, 114
 - toInt, 114
 - toString, 114
 - toVariant, 114
 - toggleBool, 114
 - type, 115
- ArnItem.cpp
 - operator<<, 258
- ArnItem.hpp
 - operator<<, 238
- ArnItemB, 115
 - ~ArnItemB, 116
 - ArnItemB, 116
 - arnLinkDestroyed, 117
 - ArnItemB, 116
 - ArnM, 132
 - close, 117
 - destroyLink, 117
 - isOpen, 117
 - itemId, 117
 - linkId, 117
 - name, 118
 - open, 118
 - path, 118
 - reference, 118
 - setReference, 119
- ArnItemB::ExportCode, 201
 - E, 201
- arnItemCreated
 - ArnItem, 104
 - ArnMonitor, 134
- ArnItemValve, 119
 - ArnItemValve, 121
 - ArnItemValve, 121
 - changed, 121
 - isAutoDestroy, 121
 - isMaster, 121
 - isSaveMode, 122
 - operator=, 122
 - setAutoDestroy, 122
 - setMaster, 122
 - setSaveMode, 122
 - setTarget, 123
 - setValue, 123
 - switchMode, 123
 - toBool, 123
- ArnItemValve::SwitchMode, 212
 - E, 213
- arnLinkDestroyed
 - ArnItemB, 117
- arnList
 - ArnClient, 55
- ArnM, 123
 - ArnItemB, 132

- defaultIgnoreSameValue, [125](#)
- destroyLink, [125](#)
- errorLog, [125](#)
- errorLogSig, [125](#)
- errorSysName, [125](#)
- exist, [126](#)
- info, [126](#)
- instance, [126](#)
- isFolder, [126](#)
- isLeaf, [126](#)
- isMainThread, [126](#)
- isThreadedApp, [127](#)
- items, [127](#)
- loadFromDirRoot, [127](#)
- loadFromFile, [127](#)
- saveToFile, [128](#)
- setConsoleError, [128](#)
- setDefaultIgnoreSameValue, [128](#)
- setSkipLocalSysLoading, [128](#)
- setValue, [129](#), [130](#)
- setErrorlog, [129](#)
- skipLocalSysLoading, [130](#)
- valueByteArray, [130](#)
- valueDouble, [130](#)
- valueInt, [131](#)
- valueString, [131](#)
- valueVariant, [131](#)
- arnModeChanged
 - ArnItem, [104](#)
- ArnMonitor, [132](#)
 - _arnClient, [137](#)
 - _monitorPath, [137](#)
 - arnChildFound, [133](#)
 - arnChildFoundFolder, [134](#)
 - arnChildFoundLeaf, [134](#)
 - arnItemCreated, [134](#)
 - ArnMonitor, [133](#)
 - ArnMonitor, [133](#)
 - client, [134](#)
 - clientId, [135](#)
 - foundChildDeleted, [135](#)
 - monitorPath, [135](#)
 - reStart, [136](#)
 - reference, [135](#)
 - setClient, [136](#)
 - setMonitorPath, [136](#)
 - setReference, [136](#)
 - start, [137](#)
- ArnPersist, [137](#)
 - ~ArnPersist, [138](#)
 - ArnPersist, [138](#)
 - ArnPersist, [138](#)
 - doArchive, [138](#)
 - setArchiveDir, [139](#)
 - setMountPoint, [139](#)
 - setPersistDir, [139](#)
 - setVcs, [140](#)
 - setupDataBase, [140](#)
- ArnPersist.cpp
 - arnDbSaveVer, [265](#)
- ArnPipe, [140](#)
 - ~ArnPipe, [143](#)
 - ArnPipe, [142](#)
 - ArnPipe, [142](#)
 - changed, [143](#)
 - isAutoDestroy, [143](#)
 - isCheckSeq, [143](#)
 - isMaster, [144](#)
 - isSendSeq, [144](#)
 - openUuid, [144](#)
 - operator=, [144](#)
 - outOfSequence, [144](#)
 - setAutoDestroy, [145](#)
 - setCheckSeq, [145](#)
 - setMaster, [145](#)
 - setSendSeq, [145](#)
 - setValue, [146](#)
 - setValueOverwrite, [146](#)
- ArnRpc, [146](#)
 - addSenderSignals, [149](#)
 - ArnRpc, [149](#)
 - ArnRpc, [149](#)
 - batchConnect, [149](#), [150](#)
 - defaultCall, [150](#)
 - heartBeatChanged, [150](#)
 - heartBeatReceived, [151](#)
 - invoke, [151](#)
 - isHeartBeatOk, [152](#)
 - mode, [152](#)
 - open, [152](#)
 - outOfSequence, [152](#)
 - pipe, [152](#)
 - pipeClosed, [153](#)
 - pipePath, [153](#)
 - rpcSender, [153](#)
 - sendText, [153](#)
 - setHeartBeatCheck, [153](#)
 - setHeartBeatSend, [154](#)
 - setIncludeSender, [154](#)
 - setMethodPrefix, [154](#)
 - setMode, [154](#)
 - setPipe, [154](#)
 - setReceiver, [154](#)
 - textReceived, [154](#)
- ArnRpc.cpp
 - RPC_STORAGE_NAME, [266](#)
- ArnRpc.hpp
 - MQ_ARG, [248](#)
 - no_queue, [248](#)
- ArnRpc::Invoke, [202](#)
 - E, [203](#)
- ArnRpc::MethodsParam::Params, [208](#)
 - allMethodIds, [209](#)
 - methodIdsTab, [209](#)
 - paramNames, [209](#)
- ArnRpc::Mode, [203](#)

- E, 204
- ArnSapi, 155
 - ArnSapi, 157
 - ArnSapi, 157
 - batchConnectFrom, 157
 - batchConnectTo, 157
 - open, 157
- ArnSapi.hpp
 - MQ_PUBLIC_ACCESS, 250
- ArnScript, 158
 - _depOfferProto, 160
 - _depProto, 160
 - _engine, 160
 - _itemProto, 160
 - _monitorProto, 160
 - ArnScript, 159
 - ArnScript, 159
 - engine, 159
 - errorLog, 159
 - errorText, 159
 - evaluate, 159
 - evaluateFile, 159
 - getClient, 159
 - idName, 159
 - logUncaughtError, 159
 - printFunction, 160
- ArnScriptJob, 160
 - ArnScriptJob, 161
 - ArnScriptJob, 161
 - errorLog, 161
 - name, 161
 - poll, 161
 - quit, 161
 - setWatchDogTime, 161
 - sigQuit, 161
 - sleepState, 162
 - watchDog, 162
 - yield, 161
- ArnScriptJob.cpp
 - EventQuit, 267
- ArnScriptJobControl, 162
 - addConfig, 163
 - addInterface, 163
 - addInterfaceList, 163
 - ArnScriptJobControl, 163
 - ArnScriptJobControl, 163
 - config, 163
 - doSetupJob, 163
 - errorText, 163
 - id, 163
 - loadScriptFile, 163
 - name, 163
 - script, 163
 - scriptChanged, 164
 - setConfig, 164
 - setName, 164
 - setScript, 164
 - setThreaded, 164
- ArnScriptJobFactory, 164
 - ~ArnScriptJobFactory, 165
 - ArnScriptJobFactory, 165
 - ArnScriptJobFactory, 165
 - getClient, 165
 - installExtension, 165
 - setupInterface, 165
 - setupJsObj, 165
- ArnScriptJobs, 165
 - addJob, 166
 - ArnScriptJobs, 166
 - ArnScriptJobs, 166
 - setFactory, 166
 - start, 166
- ArnScriptJobs::Type, 214
 - E, 214
- ArnServer, 166
 - ArnServer, 167
 - ArnServer, 167
 - listenAddress, 167
 - port, 167
 - start, 168
- ArnServer::Type, 213
 - E, 213
- ArnSync.hpp
 - ARNRECNAME, 270
- ArnZeroConf, 51
- ArnZeroConf.hpp
 - DNSServiceRef, 255
- ArnZeroConf::Error, 200
 - E, 201
- ArnZeroConf::State, 210
 - E, 210
- ArnZeroConfB, 168
 - ~ArnZeroConfB, 169
 - ArnZeroConfB, 169
 - ArnZeroConfB, 169
 - domain, 169
 - fullServiceType, 169
 - serviceType, 169
 - setDomain, 170
 - setServiceType, 170
 - setSocketType, 170
 - socketType, 171
 - state, 171
- ArnZeroConfBrowser, 171
 - ~ArnZeroConfBrowser, 174
 - activeServiceNames, 174
 - ArnZeroConfBrowser, 174
 - ArnZeroConfIntern, 178
 - ArnZeroConfBrowser, 174
 - browse, 174
 - browseError, 175
 - getNextId, 175
 - isBrowsing, 175
 - serviceAdded, 175
 - serviceChanged, 176
 - serviceNameTold, 176

- serviceRemoved, 176
- setSubType, 177
- stopBrowse, 177
- subType, 177
- ArnZeroConfIntern
 - ArnZeroConfBrowser, 178
 - ArnZeroConfLookup, 183
 - ArnZeroConfRegister, 192
 - ArnZeroConfResolve, 198
- ArnZeroConfLookup, 178
 - ~ArnZeroConfLookup, 180
 - ArnZeroConfIntern, 183
 - ArnZeroConfLookup, 180
 - ArnZeroConfLookup, 180
 - host, 180
 - hostAddr, 180
 - id, 181
 - isForceQtDnsLookup, 181
 - lookup, 181
 - lookupError, 182
 - lookupted, 181
 - releaseLookup, 182
 - setForceQtDnsLookup, 182
 - setHost, 182
 - setId, 183
- ArnZeroConfRegister, 183
 - ~ArnZeroConfRegister, 186
 - addSubType, 187
 - ArnZeroConfIntern, 192
 - ArnZeroConfRegister, 186
 - ArnZeroConfRegister, 186
 - currentServiceName, 187
 - getTxtRecordMap, 187
 - host, 188
 - port, 188
 - registerService, 188
 - registered, 188
 - registrationError, 189
 - releaseService, 189
 - serviceName, 189
 - setHost, 189
 - setPort, 190
 - setServiceName, 190
 - setSubTypes, 190
 - setTxtRecord, 191
 - setTxtRecordMap, 191
 - subTypes, 191
 - txtRecord, 191
- ArnZeroConfResolve, 192
 - ~ArnZeroConfResolve, 195
 - ArnZeroConfIntern, 198
 - ArnZeroConfResolve, 194, 195
 - ArnZeroConfResolve, 194, 195
 - getTxtRecordMap, 195
 - host, 195
 - id, 196
 - port, 196
 - releaseResolve, 196
 - resolve, 196
 - resolveError, 197
 - resolved, 196
 - serviceName, 197
 - setId, 197
 - setServiceName, 197
 - txtRecord, 198
- AutoDestroy
 - Arn::ObjectSyncMode, 208
 - ArnRpc::Mode, 204
- BadReqSeq
 - ArnZeroConf::Error, 201
- batchConnect
 - ArnRpc, 149, 150
- batchConnectFrom
 - ArnSapi, 157
- batchConnectTo
 - ArnSapi, 157
- BiDir
 - Arn::ObjectMode, 207
- Binary
 - Arn::Coding, 199
- browse
 - ArnDiscoverBrowser, 70
 - ArnZeroConfBrowser, 174
- browseError
 - ArnZeroConfBrowser, 175
- Browsing
 - ArnZeroConf::State, 210
- ByteArray
 - Arn::DataType, 200
 - ArnItemB::ExportCode, 201
- changeBasePath
 - Arn, 45
- changed
 - ArnItem, 104, 105
 - ArnItemValve, 121
 - ArnPipe, 143
- CheckSequence
 - ArnRpc::Mode, 204
- childPath
 - Arn, 45
- clear
 - Arn::XStringMap, 219
- clearArnList
 - ArnClient, 56
- clearDirectHosts
 - ArnDiscoverConnector, 79
- Client
 - ArnDiscover::Type, 214
- client
 - ArnMonitor, 134
- clientId
 - ArnMonitor, 135
- clientReadyToConnect
 - ArnDiscoverConnector, 79
 - ArnDiscoverRemote, 92

- close
 - ArnItemB, [117](#)
- completed
 - ArnDepend, [60](#)
- config
 - ArnScriptJobControl, [163](#)
- connectStatus
 - ArnClient, [56](#)
- connectToArn
 - ArnClient, [56](#)
- connectToArnList
 - ArnClient, [57](#)
- Connected
 - ArnClient::ConnectStat, [199](#)
- Connecting
 - ArnClient::ConnectStat, [199](#)
- ConnectionError
 - ArnError, [99](#)
- connectionStatusChanged
 - ArnClient, [56](#)
- convertName
 - Arn, [45](#)
- convertPath
 - Arn, [46](#)
- Cooperative
 - ArnScriptJobs::Type, [214](#)
- CreateAllowed
 - Arn::LinkFlags, [203](#)
- CreateError
 - ArnError, [99](#)
- currentService
 - ArnDiscoverAdvertise, [65](#)
- currentServiceName
 - ArnZeroConfRegister, [187](#)
- customProperties
 - ArnDiscoverAdvertise, [66](#)
- DATASTREAM_VER
 - Arn.hpp, [231](#)
- DNSServiceRef
 - ArnZeroConf.hpp, [255](#)
- Debug
 - ArnRpc::Mode, [204](#)
- debugDepend
 - Arn, [49](#)
- debugDiscover
 - Arn, [49](#)
- debugLinkDestroy
 - Arn, [49](#)
- debugLinkRef
 - Arn, [49](#)
- debugMDNS
 - Arn, [50](#)
- debugMonitor
 - Arn, [50](#)
- debugMonitorTest
 - Arn, [50](#)
- debugRPC
 - Arn, [50](#)
- debugReclnOut
 - Arn, [50](#)
- debugShareObj
 - Arn, [50](#)
- debugThreading
 - Arn, [50](#)
- debugZeroConf
 - Arn, [50](#)
- DefaultAction
 - Arn::SameValue, [209](#)
- defaultCall
 - ArnRpc, [150](#)
- defaultIgnoreSameValue
 - ArnM, [125](#)
- defaultService
 - ArnDiscoverRemote, [92](#)
 - ArnDiscoverResolver, [97](#)
- defaultStopState
 - ArnDiscoverBrowserB, [73](#)
- defaultTcpPort
 - Arn, [50](#)
- DepSlot
 - ArnDepend, [59](#)
- destroyLink
 - ArnItemB, [117](#)
 - ArnM, [125](#)
- directHostPrio
 - ArnDiscoverConnector, [80](#)
- Disconnected
 - ArnClient::ConnectStat, [199](#)
- discoverHostPrio
 - ArnDiscoverConnector, [80](#)
- doArchive
 - ArnPersist, [138](#)
- doSetupJob
 - ArnScriptJobControl, [163](#)
- doc/Description.md(2.2.0), [225](#)
- doc/HelpIndex.txt(2.2.0), [225](#)
- doc/Install.md(2.2.0), [225](#)
- doc/Internals.md(2.2.0), [225](#)
- doc/ToDo.md(2.2.0), [225](#)
- domain
 - ArnDiscoverInfo, [85](#)
 - ArnZeroConfB, [169](#)
- Double
 - Arn::DataType, [200](#)
- E
 - Arn::Coding, [199](#)
 - Arn::DataType, [200](#)
 - Arn::LinkFlags, [203](#)
 - Arn::NameF, [207](#)
 - Arn::ObjectMode, [207](#)
 - Arn::ObjectSyncMode, [208](#)
 - Arn::SameValue, [209](#)
 - ArnClient::ConnectStat, [199](#)
 - ArnDiscover::Type, [214](#)
 - ArnDiscoverAdvertise::State, [211](#)
 - ArnDiscoverInfo::State, [211](#)

- ArnError, 98
- ArnError::StdCode, 212
- ArnItemB::ExportCode, 201
- ArnItemValve::SwitchMode, 213
- ArnRpc::Invoke, 203
- ArnRpc::Mode, 204
- ArnScriptJobs::Type, 214
- ArnServer::Type, 213
- ArnZeroConf::Error, 201
- ArnZeroConf::State, 210
- EmptyOk
 - Arn::NameF, 207
- engine
 - ArnScript, 159
- Err_Custom
 - ArError::StdCode, 212
- Err_N
 - ArError, 99
- Err_Undef
 - ArError::StdCode, 212
- Error
 - ArnClient::ConnectStat, 199
- errorLog
 - ArnM, 125
 - ArnScript, 159
 - ArnScriptJob, 161
- errorLogSig
 - ArnM, 125
- errorSysName
 - ArnM, 125
- errorText
 - ArnScript, 159
 - ArnScriptJobControl, 163
- evaluate
 - ArnScript, 159
- evaluateFile
 - ArnScript, 159
- EventQuit
 - ArnScriptJob.cpp, 267
- examples/Examples.txt(2.2.0), 225
- exist
 - ArnM, 126
- externalClientConnect
 - ArnDiscoverConnector, 80
- Folder
 - Arn::LinkFlags, 203
- FolderNotOpen
 - ArError, 99
- foundChildDeleted
 - ArnMonitor, 135
- fromXString
 - Arn::XStringMap, 219
- fullPath
 - Arn, 46
- fullServiceType
 - ArnZeroConfB, 169
- getClient
 - ArnScript, 159
- getMode
 - ArnItem, 105
- getNextId
 - ArnZeroConfBrowser, 175
- getTxtRecordMap
 - ArnZeroConfRegister, 187
 - ArnZeroConfResolve, 195
- goTowardState
 - ArnDiscoverBrowserB, 73
- groups
 - ArnDiscoverAdvertise, 66
 - ArnDiscoverInfo, 85
- heartBeatChanged
 - ArnRpc, 150
- heartBeatReceived
 - ArnRpc, 151
- host
 - ArnZeroConfLookup, 180
 - ArnZeroConfRegister, 188
 - ArnZeroConfResolve, 195
- HostInfo
 - ArnDiscoverInfo::State, 211
- HostInfoErr
 - ArnDiscoverInfo::State, 211
- HostIp
 - ArnDiscoverInfo::State, 212
- HostIpErr
 - ArnDiscoverInfo::State, 212
- hostAddr
 - ArnZeroConfLookup, 180
- HostAddrPort
 - ArnClient::HostAddrPort, 202
- hostFromHostWithInfo
 - Arn, 46
- hostIp
 - ArnDiscoverInfo, 86
- hostIpString
 - ArnDiscoverInfo, 86
- HostList
 - ArnClient, 54
- hostName
 - ArnDiscoverInfo, 86
- hostPort
 - ArnDiscoverInfo, 86
- hostPortString
 - ArnDiscoverInfo, 86
- hostWithInfo
 - ArnDiscoverInfo, 87
- id
 - ArnDiscoverConnector, 80
 - ArnScriptJobControl, 163
 - ArnZeroConfLookup, 181
 - ArnZeroConfResolve, 196
- idName
 - ArnScript, 159

- IdToIndex
 - ArnDiscoverBrowserB, 74
- Ignore
 - Arn::SameValue, 209
- InOutputStream
 - ArnItemValve::SwitchMode, 213
- InProgress
 - ArnZeroConf::State, 210
- InStream
 - ArnItemValve::SwitchMode, 213
- inProgress
 - ArnDiscoverInfo, 87
- indexOf
 - Arn::XStringMap, 219
- indexOfValue
 - Arn::XStringMap, 219
- indexTold
 - ArnDiscoverBrowserB, 74
- Info
 - ArnError, 98
 - ArnError::StdCode, 212
- info
 - ArnM, 126
- infoById
 - ArnDiscoverBrowserB, 74
- infoByIndex
 - ArnDiscoverBrowserB, 75
- infoByName
 - ArnDiscoverBrowserB, 75
- infoUpdated
 - ArnDiscoverBrowserB, 75
- Init
 - ArnClient::ConnectStat, 199
 - ArnDiscoverInfo::State, 211
- initialServiceTimeout
 - ArnDiscoverRemote, 92
- installExtension
 - ArnScriptJobFactory, 165
- instance
 - ArnM, 126
- Int
 - Arn::DataType, 200
- invoke
 - ArnRpc, 151
- isAutoDestroy
 - ArnItem, 105
 - ArnItemValve, 121
 - ArnPipe, 143
- isBiDir
 - ArnItem, 106
- isBiDirMode
 - ArnItem, 106
- isBrowsing
 - ArnDiscoverBrowser, 71
 - ArnZeroConfBrowser, 175
- isCheckSeq
 - ArnPipe, 143
- isError
 - ArnDiscoverInfo, 87
- isFolder
 - ArnItem, 106
 - ArnM, 126
- isFolderPath
 - Arn, 47
- isForceQtDnsLookup
 - ArnZeroConfLookup, 181
- isHeartBeatOk
 - ArnRpc, 152
- isIgnoreSameValue
 - ArnItem, 106
- isLeaf
 - ArnM, 126
- isMainThread
 - ArnM, 126
- isMaster
 - ArnItem, 107
 - ArnItemValve, 121
 - ArnPipe, 144
- isOpen
 - ArnItemB, 117
- isPipeMode
 - ArnItem, 107
- isProviderPath
 - Arn, 47
- isSaveMode
 - ArnItem, 107
 - ArnItemValve, 122
- isSendSeq
 - ArnPipe, 144
- isTemplate
 - ArnItem, 107
- isThreadedApp
 - ArnM, 127
- ItemNotOpen
 - ArnError, 99
- ItemNotSet
 - ArnError, 99
- itemId
 - ArnItemB, 117
- itemName
 - Arn, 47
- items
 - ArnM, 127
- key
 - Arn::XStringMap, 219
- keyRef
 - Arn::XStringMap, 219
- keyString
 - Arn::XStringMap, 220
- keys
 - Arn::XStringMap, 219
- label
 - MQGenericArgument, 206
- linkId
 - ArnItemB, 117

- listenAddress
 - ArnServer, 167
- loadFromDirRoot
 - ArnM, 127
- loadFromFile
 - ArnM, 127
- loadScriptFile
 - ArnScriptJobControl, 163
- logUncaughtError
 - ArnScript, 159
- LookingUp
 - ArnZeroConf::State, 210
- Lookup
 - ArnZeroConf::State, 210
- lookup
 - ArnZeroConfLookup, 181
- lookupError
 - ArnZeroConfLookup, 182
- Lookuped
 - ArnZeroConf::State, 210
- lookuped
 - ArnZeroConfLookup, 181
- MQ_ARG
 - ArnRpc.hpp, 248
- MQ_DECLARE_ENUM
 - MQFlags.hpp, 256
- MQ_DECLARE_FLAGS
 - MQFlags.hpp, 256
- MQ_PUBLIC_ACCESS
 - ArnSapi.hpp, 250
- MQArgument
 - MQArgument, 205
 - MQArgument, 205
- MQArgument< T >, 204
- MQFlags.hpp
 - MQ_DECLARE_ENUM, 256
 - MQ_DECLARE_FLAGS, 256
- MQGenericArgument, 206
 - label, 206
 - MQGenericArgument, 206
 - MQGenericArgument, 206
- makeHostWithInfo
 - Arn, 48
- makePath
 - Arn, 48
- Master
 - Arn::ObjectSyncMode, 208
- maxEnumOf
 - Arn::XStringMap, 220
- methodIdsTab
 - ArnRpc::MethodsParam::Params, 209
- mode
 - ArnRpc, 152
- modeChanged
 - ArnItem, 108
- Monitor
 - Arn::ObjectSyncMode, 208
- monitorPath
 - ArnMonitor, 135
- name
 - ArnItemB, 118
 - ArnScriptJob, 161
 - ArnScriptJobControl, 163
- NamedArg
 - ArnRpc::Mode, 204
- NamedTypedArg
 - ArnRpc::Mode, 204
- NetSync
 - ArnServer::Type, 213
- newConnector
 - ArnDiscoverRemote, 93
- NoDefaultArgs
 - ArnRpc::Mode, 204
- NoFolderMark
 - Arn::NameF, 207
- NoQueue
 - ArnRpc::Invoke, 203
- no_queue
 - ArnRpc.hpp, 248
- None
 - ArnDiscover::Type, 214
 - ArnDiscoverAdvertise::State, 211
 - ArnZeroConf::State, 210
- Normal
 - Arn::ObjectSyncMode, 208
- NotFound
 - ArnError, 99
- NotMainThread
 - ArnError, 99
- NotOpen
 - ArnError, 99
- Null
 - Arn::DataType, 200
 - ArnScriptJobs::Type, 214
- Ok
 - ArnError, 98
 - ArnError::StdCode, 212
 - ArnZeroConf::Error, 201
- OnlyPosArgIn
 - ArnRpc::Mode, 204
- open
 - ArnItemB, 118
 - ArnRpc, 152
 - ArnSapi, 157
- openFolder
 - ArnItem, 108
- openUid
 - ArnItem, 108
 - ArnPipe, 144
- openUidPipe
 - ArnItem, 108
- operator<<
 - ArnItem.cpp, 258
 - ArnItem.hpp, 238
- operator+=

- Arn::XStringMap, 220
- operator=
 - ArnItem, 109
 - ArnItemValve, 122
 - ArnPipe, 144
- OutStream
 - ArnItemValve::SwitchMode, 213
- outOfSequence
 - ArnPipe, 144
 - ArnRpc, 152
- paramNames
 - ArnRpc::MethodsParam::Params, 209
- path
 - ArnItemB, 118
- pathDiscover
 - Arn, 50
- pathDiscoverConnect
 - Arn, 50
- pathDiscoverThis
 - Arn, 50
- pathLocal
 - Arn, 51
- pathLocalSys
 - Arn, 51
- Pipe
 - Arn::ObjectMode, 207
- pipe
 - ArnRpc, 152
- pipeClosed
 - ArnRpc, 153
- pipePath
 - ArnRpc, 153
- poll
 - ArnScriptJob, 161
- port
 - ArnClient::HostAddrPort, 202
 - ArnServer, 167
 - ArnZeroConfRegister, 188
 - ArnZeroConfResolve, 196
- Preemptive
 - ArnScriptJobs::Type, 214
- printFunction
 - ArnScript, 160
- properties
 - ArnDiscoverInfo, 87
- Provider
 - ArnRpc::Mode, 204
- providerPath
 - Arn, 48
- quit
 - ArnScriptJob, 161
- README.md(2.2.0), 225
- RPC_STORAGE_NAME
 - ArnRpc.cpp, 266
- reStart
 - ArnMonitor, 136
- RecUnknown
 - ArnError, 99
- reference
 - ArnItemB, 118
 - ArnMonitor, 135
- Register
 - ArnZeroConf::State, 210
- registerService
 - ArnZeroConfRegister, 188
- Registered
 - ArnZeroConf::State, 210
- registered
 - ArnZeroConfRegister, 188
- Registering
 - ArnZeroConf::State, 210
- registrationError
 - ArnZeroConfRegister, 189
- Relative
 - Arn::NameF, 207
- releaseLookup
 - ArnZeroConfLookup, 182
- releaseResolve
 - ArnZeroConfResolve, 196
- releaseService
 - ArnZeroConfRegister, 189
- remove
 - Arn::XStringMap, 220
- removeMountPoint
 - ArnClient, 57
- resolvCode
 - ArnDiscoverInfo, 88
- Resolve
 - ArnZeroConf::State, 210
- resolve
 - ArnDiscoverResolver, 97
 - ArnZeroConfResolve, 196
- resolveError
 - ArnZeroConfResolve, 197
- resolveRefreshTimeout
 - ArnDiscoverConnector, 81
- Resolved
 - ArnZeroConf::State, 210
- resolved
 - ArnZeroConfResolve, 196
- Resolving
 - ArnZeroConf::State, 210
- resourceArnLib
 - Arn, 51
- resourceArnRoot
 - Arn, 51
- Retired
 - ArnError, 99
- RpcInvokeError
 - ArnError, 99
- RpcReceiveError
 - ArnError, 99
- rpcSender
 - ArnRpc, 153

- Running
 - ArnZeroConf::Error, 201
- Save
 - Arn::ObjectMode, 208
- saveToFile
 - ArnM, 128
- script
 - ArnScriptJobControl, 163
- ScriptError
 - ArnError, 99
- scriptChanged
 - ArnScriptJobControl, 164
- SendSequence
 - ArnRpc::Mode, 204
- sendText
 - ArnRpc, 153
- Server
 - ArnDiscover::Type, 214
- service
 - ArnDiscoverAdvertise, 66
 - ArnDiscoverConnector, 81
- ServiceName
 - ArnDiscoverInfo::State, 211
- serviceAdded
 - ArnDiscoverBrowserB, 76
 - ArnZeroConfBrowser, 175
- serviceChangeError
 - ArnDiscoverAdvertise, 67
- serviceChanged
 - ArnDiscoverAdvertise, 66
 - ArnZeroConfBrowser, 176
- serviceCount
 - ArnDiscoverBrowserB, 76
- serviceName
 - ArnDiscoverInfo, 88
 - ArnZeroConfRegister, 189
 - ArnZeroConfResolve, 197
- serviceNameTold
 - ArnDiscoverBrowserB, 76
 - ArnZeroConfBrowser, 176
- serviceRemoved
 - ArnDiscoverBrowserB, 77
 - ArnZeroConfBrowser, 176
- serviceType
 - ArnZeroConfB, 169
- set
 - Arn::XStringMap, 220, 221
- setArchiveDir
 - ArnPersist, 139
- setAutoConnect
 - ArnClient, 57
- setAutoDestroy
 - ArnItem, 109
 - ArnItemValve, 122
 - ArnPipe, 145
- setBiDirMode
 - ArnItem, 109
- setCheckSeq
 - ArnPipe, 145
- setClient
 - ArnMonitor, 136
- setConfig
 - ArnScriptJobControl, 164
- setConsoleError
 - ArnM, 128
- setCustomProperties
 - ArnDiscoverAdvertise, 67
- setDefaultIgnoreSameValue
 - ArnM, 128
- setDefaultService
 - ArnDiscoverRemote, 93
 - ArnDiscoverResolver, 98
- setDefaultStopState
 - ArnDiscoverBrowserB, 77
- setDelay
 - ArnItem, 110
- setDirectHostPrio
 - ArnDiscoverConnector, 81
- setDiscoverHostPrio
 - ArnDiscoverConnector, 82
- setDomain
 - ArnZeroConfB, 170
- setEmptyKeysToValue
 - Arn::XStringMap, 221
- setExternalClientConnect
 - ArnDiscoverConnector, 82
- setFactory
 - ArnScriptJobs, 166
- setFilter
 - ArnDiscoverBrowser, 71
- setForceQtDnsLookup
 - ArnZeroConfLookup, 182
- setGroups
 - ArnDiscoverAdvertise, 67
- setHeartBeatCheck
 - ArnRpc, 153
- setHeartBeatSend
 - ArnRpc, 154
- setHost
 - ArnZeroConfLookup, 182
 - ArnZeroConfRegister, 189
- setId
 - ArnZeroConfLookup, 183
 - ArnZeroConfResolve, 197
- setIgnoreSameValue
 - ArnItem, 110
- setIncludeSender
 - ArnRpc, 154
- setInitialServiceTimeout
 - ArnDiscoverRemote, 93
- setMaster
 - ArnItem, 110
 - ArnItemValve, 122
 - ArnPipe, 145
- setMethodPrefix
 - ArnRpc, 154

- setMode
 - Arnrpc, 154
- setMonitorName
 - Arndepend, 60
- setMonitorPath
 - Arnmonitor, 136
- setMountPoint
 - Arnclient, 57
 - Arnpersist, 139
- setName
 - ArnScriptJobControl, 164
- setPersistDir
 - Arnpersist, 139
- setPipe
 - Arnrpc, 154
- setPipeMode
 - Arnitem, 110
- setPort
 - ArnZeroConfRegister, 190
- setReceiver
 - Arnrpc, 154
- setReference
 - ArnitemB, 119
 - Arnmonitor, 136
- setResolveRefreshTimeout
 - ArnDiscoverConnector, 83
- setResolver
 - ArnDiscoverConnector, 82
- setSaveMode
 - Arnitem, 110
 - ArnitemValve, 122
- setScript
 - ArnScriptJobControl, 164
- setSendSeq
 - Arnpipe, 145
- setService
 - ArnDiscoverAdvertise, 68
 - ArnDiscoverConnector, 83
 - ArnDiscoverRemote, 94
- setServiceName
 - ArnZeroConfRegister, 190
 - ArnZeroConfResolve, 197
- setServiceType
 - ArnZeroConfB, 170
- setSkipLocalSysLoading
 - ArnM, 128
- setSocketType
 - ArnZeroConfB, 170
- setStateId
 - ArndependOffer, 61
- setStateName
 - ArndependOffer, 62
- setSubType
 - ArnZeroConfBrowser, 177
- setSubTypes
 - ArnZeroConfRegister, 190
- setTarget
 - ArnitemValve, 123
- setTemplate
 - Arnitem, 111
- setThreaded
 - ArnScriptJobControl, 164
- setTxtRecord
 - ArnZeroConfRegister, 191
- setTxtRecordMap
 - ArnZeroConfRegister, 191
- setValue
 - Arnitem, 111–113
 - ArnitemValve, 123
 - ArnM, 129, 130
 - Arnpipe, 146
- setValueOverwrite
 - Arnpipe, 146
- setVcs
 - Arnpersist, 140
- setWatchDogTime
 - ArnScriptJob, 161
- setupDataBase
 - Arnpersist, 140
- setupErrorlog
 - ArnM, 129
- setupInterface
 - ArnScriptJobFactory, 165
- setupJsObj
 - ArnScriptJobFactory, 165
- sigQuit
 - ArnScriptJob, 161
- SilentError
 - Arn::LinkFlags, 203
- size
 - Arn::XStringMap, 221
- skipLocalSysLoading
 - ArnM, 130
- sleepState
 - ArnScriptJob, 162
- socketType
 - ArnZeroConfB, 171
- squeeze
 - Arn::XStringMap, 221
- src/Arn.cpp(2.2.0), 225
- src/ArnClient.cpp(2.2.0), 227
- src/Arndepend.cpp(2.2.0), 227
- src/ArnDiscover.cpp(2.2.0), 228
- src/ArnDiscoverConnect.cpp(2.2.0), 228
- src/ArnDiscoverRemote.cpp(2.2.0), 229
- src/ArnInc/Arn.hpp(2.2.0), 229
- src/ArnInc/ArnClient.hpp(2.2.0), 231
- src/ArnInc/Arndepend.hpp(2.2.0), 232
- src/ArnInc/ArnDiscover.hpp(2.2.0), 233
- src/ArnInc/ArnDiscoverConnect.hpp(2.2.0), 235
- src/ArnInc/ArnDiscoverRemote.hpp(2.2.0), 235
- src/ArnInc/ArnError.hpp(2.2.0), 236
- src/ArnInc/Arnitem.hpp(2.2.0), 237
- src/ArnInc/ArnitemB.hpp(2.2.0), 239
- src/ArnInc/ArnitemValve.hpp(2.2.0), 239
- src/ArnInc/ArnLib.hpp(2.2.0), 240

- src/ArnInc/ArnLib_global.hpp(2.2.0), [241](#)
- src/ArnInc/ArnLinkHandle.hpp(2.2.0), [242](#)
- src/ArnInc/ArnM.hpp(2.2.0), [243](#)
- src/ArnInc/ArnMonitor.hpp(2.2.0), [243](#)
- src/ArnInc/ArnPersist.hpp(2.2.0), [244](#)
- src/ArnInc/ArnPersistSapi.hpp(2.2.0), [245](#)
- src/ArnInc/ArnPipe.hpp(2.2.0), [246](#)
- src/ArnInc/ArnRpc.hpp(2.2.0), [247](#)
- src/ArnInc/ArnSapi.hpp(2.2.0), [249](#)
- src/ArnInc/ArnScript.hpp(2.2.0), [250](#)
- src/ArnInc/ArnScriptJob.hpp(2.2.0), [251](#)
- src/ArnInc/ArnScriptJobs.hpp(2.2.0), [252](#)
- src/ArnInc/ArnServer.hpp(2.2.0), [253](#)
- src/ArnInc/ArnZeroConf.hpp(2.2.0), [254](#)
- src/ArnInc/MQFlags.hpp(2.2.0), [256](#)
- src/ArnInc/XStringMap.hpp(2.2.0), [257](#)
- src/ArnItem.cpp(2.2.0), [258](#)
- src/ArnItemB.cpp(2.2.0), [258](#)
- src/ArnItemNet.cpp(2.2.0), [259](#)
- src/ArnItemNet.hpp(2.2.0), [259](#)
- src/ArnItemValve.cpp(2.2.0), [260](#)
- src/ArnLib.cpp(2.2.0), [260](#)
- src/ArnLink.cpp(2.2.0), [261](#)
- src/ArnLink.hpp(2.2.0), [262](#)
- src/ArnLinkHandle.cpp(2.2.0), [263](#)
- src/ArnM.cpp(2.2.0), [263](#)
- src/ArnMonitor.cpp(2.2.0), [264](#)
- src/ArnPersist.cpp(2.2.0), [264](#)
- src/ArnPipe.cpp(2.2.0), [265](#)
- src/ArnRpc.cpp(2.2.0), [265](#)
- src/ArnSapi.cpp(2.2.0), [266](#)
- src/ArnScript.cpp(2.2.0), [266](#)
- src/ArnScriptJob.cpp(2.2.0), [267](#)
- src/ArnScriptJobs.cpp(2.2.0), [267](#)
- src/ArnServer.cpp(2.2.0), [268](#)
- src/ArnSync.cpp(2.2.0), [268](#)
- src/ArnSync.hpp(2.2.0), [269](#)
- src/ArnXStringMap.cpp(2.2.0), [270](#)
- src/ArnZeroConf.cpp(2.2.0), [270](#)
- start
 - ArnDiscoverConnector, [83](#)
 - ArnMonitor, [137](#)
 - ArnScriptJobs, [166](#)
 - ArnServer, [168](#)
- startMonitor
 - ArnDepend, [60](#)
- startUseNewServer
 - ArnDiscoverRemote, [94](#)
- startUseServer
 - ArnDiscoverRemote, [94](#)
- StartupAdvertise
 - ArnDiscoverAdvertise::State, [211](#)
- state
 - ArnDiscoverAdvertise, [68](#)
 - ArnDiscoverInfo, [88](#)
 - ArnZeroConfB, [171](#)
- stateld
 - ArnDependOffer, [62](#)
- stateName
 - ArnDependOffer, [62](#)
- stopBrowse
 - ArnDiscoverBrowser, [71](#)
 - ArnZeroConfBrowser, [177](#)
- stopState
 - ArnDiscoverInfo, [88](#)
- String
 - Arn::DataType, [200](#)
 - ArnItemB::ExportCode, [201](#)
- stringCode
 - Arn::XStringMap, [221](#)
- stringDecode
 - Arn::XStringMap, [221](#)
- subType
 - ArnZeroConfBrowser, [177](#)
- subTypes
 - ArnZeroConfRegister, [191](#)
- switchMode
 - ArnItemValve, [123](#)
- syncMode
 - ArnItem, [113](#)
- tcpConnected
 - ArnClient, [58](#)
- tcpDisconnected
 - ArnClient, [58](#)
- tcpError
 - ArnClient, [58](#)
- Text
 - Arn::Coding, [199](#)
- textReceived
 - ArnRpc, [154](#)
- Threaded
 - Arn::LinkFlags, [203](#)
- Timeout
 - ArnZeroConf::Error, [201](#)
- toBool
 - ArnItem, [114](#)
 - ArnItemValve, [123](#)
- toByteArray
 - ArnItem, [114](#)
- toDouble
 - ArnItem, [114](#)
- toInt
 - ArnItem, [114](#)
- toString
 - ArnItem, [114](#)
- toVariant
 - ArnItem, [114](#)
- toVariantMap
 - Arn::XStringMap, [221](#)
- toXString
 - Arn::XStringMap, [221](#)
- toggleBool
 - ArnItem, [114](#)
- TriedAll
 - ArnClient::ConnectStat, [199](#)
- twinPath

- Arn, [49](#)
- txtRecord
 - ArnZeroConfRegister, [191](#)
 - ArnZeroConfResolve, [198](#)
- type
 - ArnDiscoverInfo, [89](#)
 - ArnItem, [115](#)
- typeString
 - ArnDiscoverInfo, [89](#)
- UDnsFail
 - ArnZeroConf::Error, [201](#)
- Undef
 - ArnError, [99](#)
- UseDefaultCall
 - ArnRpc::Mode, [204](#)
- UuidAutoDestroy
 - ArnRpc::Mode, [204](#)
- UuidPipe
 - ArnRpc::Mode, [204](#)
- value
 - Arn::XStringMap, [221](#), [222](#)
- valueByteArray
 - ArnM, [130](#)
- valueDouble
 - ArnM, [130](#)
- valueInt
 - ArnM, [131](#)
- valueRef
 - Arn::XStringMap, [222](#)
- valueString
 - Arn::XStringMap, [222](#)
 - ArnM, [131](#)
- valueVariant
 - ArnM, [131](#)
- values
 - Arn::XStringMap, [222](#)
- Variant
 - Arn::DataType, [200](#)
 - ArnItemB::ExportCode, [201](#)
- VariantBin
 - ArnItemB::ExportCode, [201](#)
- VariantTxt
 - ArnItemB::ExportCode, [201](#)
- Warning
 - ArnError, [98](#)
 - ArnError::StdCode, [212](#)
- warningMDNS
 - Arn, [51](#)
- watchDog
 - ArnScriptJob, [162](#)
- XStringMap
 - Arn::XStringMap, [217](#)
- yield
 - ArnScriptJob, [161](#)