

Hosting Environment (Daemon)

Generated by Doxygen 1.7.4

Mon Oct 10 2011 16:48:27

Contents

1 Namespace Index	1
1.1 Namespace List	1
2 Data Structure Index	3
2.1 Class Hierarchy	3
3 Data Structure Index	11
3.1 Data Structures	11
4 Namespace Documentation	19
4.1 Arc Namespace Reference	19
4.1.1 Detailed Description	30
4.1.2 Typedef Documentation	30
4.1.2.1 AttrConstIter	30
4.1.2.2 AttrIter	31
4.1.2.3 AttrMap	31
4.1.2.4 get_plugin_instance	31
4.1.3 Enumeration Type Documentation	31
4.1.3.1 escape_type	31
4.1.3.2 LogFormat	31
4.1.3.3 LogLevel	31
4.1.3.4 StatusKind	32
4.1.3.5 WSAFault	32
4.1.4 Function Documentation	32
4.1.4.1 addVOMSAC	32
4.1.4.2 ContentFromPayload	33

4.1.4.3	CreateThreadFunction	33
4.1.4.4	createVOMSAC	33
4.1.4.5	DirCreate	33
4.1.4.6	DirDelete	33
4.1.4.7	EnvLockUnwrap	34
4.1.4.8	EnvLockWrap	34
4.1.4.9	escape_chars	34
4.1.4.10	FileCopy	34
4.1.4.11	FileCreate	34
4.1.4.12	FileDelete	34
4.1.4.13	FileLink	35
4.1.4.14	FileRead	35
4.1.4.15	FileReadLink	35
4.1.4.16	FileStat	35
4.1.4.17	final_xmlsec	35
4.1.4.18	get_cert_str	35
4.1.4.19	get_key_from_certfile	35
4.1.4.20	get_key_from_certstr	35
4.1.4.21	get_key_from_keyfile	36
4.1.4.22	get_key_from_keystr	36
4.1.4.23	get_node	36
4.1.4.24	getCredentialProperty	36
4.1.4.25	GUID	36
4.1.4.26	init_xmlsec	36
4.1.4.27	inttostr	36
4.1.4.28	inttostr	37
4.1.4.29	inttostr	37
4.1.4.30	inttostr	37
4.1.4.31	inttostr	37
4.1.4.32	inttostr	37
4.1.4.33	istring_to_level	37
4.1.4.34	load_key_from_certfile	38
4.1.4.35	load_key_from_certstr	38
4.1.4.36	load_key_from_keyfile	38

4.1.4.37	load_trusted_cert_file	38
4.1.4.38	load_trusted_cert_str	38
4.1.4.39	load_trusted_certs	38
4.1.4.40	MatchXMLName	38
4.1.4.41	MatchXMLName	39
4.1.4.42	MatchXMLName	39
4.1.4.43	MatchXMLNamespace	39
4.1.4.44	MatchXMLNamespace	39
4.1.4.45	MatchXMLNamespace	39
4.1.4.46	OpenSSLInit	39
4.1.4.47	operator<<	39
4.1.4.48	operator<<	39
4.1.4.49	operator<<	39
4.1.4.50	parseVOMSAC	40
4.1.4.51	parseVOMSAC	41
4.1.4.52	passphrase_callback	41
4.1.4.53	string	41
4.1.4.54	strtoint	41
4.1.4.55	strtoint	41
4.1.4.56	strtoint	41
4.1.4.57	strtoint	41
4.1.4.58	strtoint	41
4.1.4.59	strtoint	42
4.1.4.60	TimeStamp	42
4.1.4.61	TimeStamp	42
4.1.4.62	TmpDirCreate	42
4.1.4.63	TmpFileCreate	42
4.1.4.64	uri_encode	42
4.1.4.65	VOMSDecode	42
4.1.4.66	WSAFaultAssign	43
4.1.4.67	WSAFaultExtract	43
4.1.5	Variable Documentation	43
4.1.5.1	CredentialLogger	43
4.1.5.2	plugins_table_name	43

4.1.5.3	thread_stacksize	43
4.2	ArcCredential Namespace Reference	43
4.2.1	Detailed Description	44
4.2.2	Enumeration Type Documentation	44
4.2.2.1	certType	44
4.3	DataStaging Namespace Reference	45
4.3.1	Detailed Description	46
4.3.2	Enumeration Type Documentation	46
4.3.2.1	CacheState	46
5	Data Structure Documentation	49
5.1	ArcCredential::ACACI Struct Reference	49
5.2	ArcCredential::ACATHOLDER Struct Reference	49
5.3	ArcCredential::ACATTR Struct Reference	49
5.4	ArcCredential::ACATTRIBUTE Struct Reference	49
5.5	ArcCredential::ACC Struct Reference	49
5.6	ArcCredential::ACCERTS Struct Reference	50
5.7	ArcCredential::ACDIGEST Struct Reference	50
5.8	ArcCredential::ACFORM Struct Reference	50
5.9	ArcCredential::ACFULLATTRIBUTES Struct Reference	50
5.10	ArcCredential::ACHOLDER Struct Reference	50
5.11	ArcCredential::ACIETFATTR Struct Reference	50
5.12	ArcCredential::ACINFO Struct Reference	51
5.13	ArcCredential::ACIS Struct Reference	51
5.14	ArcCredential::ACSEQ Struct Reference	51
5.15	ArcCredential::ACTARGET Struct Reference	51
5.16	ArcCredential::ACTARGETS Struct Reference	51
5.17	ArcCredential::ACVAL Struct Reference	51
5.18	Arc::Adler32Sum Class Reference	51
5.18.1	Detailed Description	52
5.18.2	Member Function Documentation	52
5.18.2.1	add	52
5.18.2.2	end	53
5.18.2.3	print	53

5.18.2.4	scan	53
5.18.2.5	start	53
5.19	ArcSec::AlgFactory Class Reference	54
5.19.1	Detailed Description	54
5.19.2	Member Function Documentation	54
5.19.2.1	createAlg	54
5.20	ArcSec::AnyURIAttribute Class Reference	55
5.20.1	Member Function Documentation	55
5.20.1.1	encode	55
5.20.1.2	equal	55
5.20.1.3	getId	55
5.20.1.4	getType	55
5.21	Arc::ApplicationEnvironment Class Reference	56
5.21.1	Detailed Description	56
5.22	Arc::ApplicationType Class Reference	56
5.23	Arc::ArcLocation Class Reference	56
5.23.1	Detailed Description	57
5.23.2	Member Function Documentation	57
5.23.2.1	GetPlugins	57
5.23.2.2	Init	57
5.24	ArcSec::ArcPeriod Struct Reference	57
5.25	Arc::ARCPolicyHandlerConfig Class Reference	57
5.26	Arc::ArcVersion Class Reference	58
5.26.1	Detailed Description	58
5.27	ArcSec::Attr Struct Reference	58
5.27.1	Detailed Description	58
5.28	ArcSec::AttributeFactory Class Reference	58
5.28.1	Detailed Description	59
5.29	Arc::Attributeliterator Class Reference	59
5.29.1	Detailed Description	60
5.29.2	Constructor & Destructor Documentation	60
5.29.2.1	Attributeliterator	60
5.29.2.2	Attributeliterator	60
5.29.3	Member Function Documentation	60

5.29.3.1	hasMore	60
5.29.3.2	key	61
5.29.3.3	operator*	61
5.29.3.4	operator++	61
5.29.3.5	operator++	61
5.29.3.6	operator->	61
5.29.4	Friends And Related Function Documentation	62
5.29.4.1	MessageAttributes	62
5.29.5	Field Documentation	62
5.29.5.1	current_	62
5.29.5.2	end_	62
5.30	ArcSec::AttributeProxy Class Reference	62
5.30.1	Detailed Description	62
5.30.2	Member Function Documentation	63
5.30.2.1	getAttribute	63
5.31	ArcSec::AttributeValue Class Reference	63
5.31.1	Detailed Description	64
5.31.2	Member Function Documentation	64
5.31.2.1	encode	64
5.31.2.2	equal	64
5.31.2.3	getId	64
5.31.2.4	getType	64
5.32	ArcSec::Attrs Class Reference	65
5.32.1	Detailed Description	65
5.33	ArcSec::AuthzRequest Struct Reference	65
5.34	ArcSec::AuthzRequestSection Struct Reference	65
5.34.1	Detailed Description	65
5.35	Arc::AutoPointer< T > Class Template Reference	66
5.35.1	Detailed Description	66
5.36	Arc::Base64 Class Reference	66
5.37	Arc::BaseConfig Class Reference	66
5.37.1	Detailed Description	67
5.37.2	Member Function Documentation	67
5.37.2.1	AddCADir	67

5.37.2.2	AddCAFile	67
5.37.2.3	AddCertificate	67
5.37.2.4	AddOverlay	67
5.37.2.5	AddPluginsPath	68
5.37.2.6	AddPrivateKey	68
5.37.2.7	AddProxy	68
5.37.2.8	GetOverlay	68
5.37.2.9	MakeConfig	68
5.38	ArcSec::BooleanAttribute Class Reference	68
5.38.1	Member Function Documentation	69
5.38.1.1	encode	69
5.38.1.2	equal	69
5.38.1.3	getId	69
5.38.1.4	getType	69
5.39	Arc::Broker Class Reference	69
5.39.1	Member Function Documentation	70
5.39.1.1	GetBestTarget	70
5.39.1.2	PreFilterTargets	70
5.39.1.3	SortTargets	71
5.39.2	Field Documentation	71
5.39.2.1	PossibleTargets	71
5.40	Arc::BrokerLoader Class Reference	71
5.40.1	Detailed Description	71
5.40.2	Constructor & Destructor Documentation	72
5.40.2.1	BrokerLoader	72
5.40.2.2	~BrokerLoader	72
5.40.3	Member Function Documentation	72
5.40.3.1	GetBrokers	72
5.40.3.2	load	72
5.41	Arc::BrokerPluginArgument Class Reference	72
5.42	Arc::ByteArray Class Reference	73
5.43	DataStaging::CacheParameters Class Reference	73
5.43.1	Detailed Description	73
5.44	ArcCredential::cert_verify_context Struct Reference	74

5.45 Arc::CertEnvLocker Class Reference	74
5.46 Arc::ChainContext Class Reference	74
5.46.1 Detailed Description	74
5.46.2 Member Function Documentation	74
5.46.2.1 operator PluginsFactory *	74
5.47 Arc::CheckSum Class Reference	75
5.47.1 Detailed Description	75
5.47.2 Member Function Documentation	75
5.47.2.1 add	75
5.47.2.2 end	76
5.47.2.3 print	76
5.47.2.4 scan	76
5.47.2.5 start	77
5.48 Arc::CheckSumAny Class Reference	77
5.48.1 Detailed Description	78
5.48.2 Member Function Documentation	78
5.48.2.1 add	78
5.48.2.2 end	78
5.48.2.3 FileChecksum	78
5.48.2.4 print	79
5.48.2.5 scan	79
5.48.2.6 start	80
5.49 Arc::CIStrValue Class Reference	80
5.49.1 Detailed Description	80
5.49.2 Constructor & Destructor Documentation	81
5.49.2.1 CIStrValue	81
5.49.2.2 CIStrValue	81
5.49.2.3 CIStrValue	81
5.49.3 Member Function Documentation	81
5.49.3.1 equal	81
5.49.3.2 operator bool	81
5.50 Arc::ClassLoader Class Reference	81
5.51 Arc::ClassLoaderPluginArgument Class Reference	82
5.52 Arc::ClientHTTP Class Reference	82

5.52.1	Detailed Description	82
5.53	Arc::ClientHTTPwithSAML2SSO Class Reference	83
5.53.1	Constructor & Destructor Documentation	83
5.53.1.1	ClientHTTPwithSAML2SSO	83
5.53.2	Member Function Documentation	83
5.53.2.1	process	83
5.54	Arc::ClientInterface Class Reference	83
5.54.1	Detailed Description	84
5.55	Arc::ClientSOAP Class Reference	84
5.55.1	Detailed Description	84
5.55.2	Constructor & Destructor Documentation	85
5.55.2.1	ClientSOAP	85
5.55.3	Member Function Documentation	85
5.55.3.1	AddSecHandler	85
5.55.3.2	GetEntry	85
5.55.3.3	Load	85
5.55.3.4	process	85
5.55.3.5	process	85
5.56	Arc::ClientSOAPwithSAML2SSO Class Reference	85
5.56.1	Constructor & Destructor Documentation	86
5.56.1.1	ClientSOAPwithSAML2SSO	86
5.56.2	Member Function Documentation	86
5.56.2.1	process	86
5.56.2.2	process	86
5.57	Arc::ClientTCP Class Reference	86
5.57.1	Detailed Description	87
5.58	Arc::ClientX509Delegation Class Reference	87
5.58.1	Constructor & Destructor Documentation	87
5.58.1.1	ClientX509Delegation	87
5.58.2	Member Function Documentation	87
5.58.2.1	acquireDelegation	88
5.58.2.2	createDelegation	88
5.59	ArcSec::CombiningAlg Class Reference	89
5.59.1	Detailed Description	89

5.59.2 Member Function Documentation	89
5.59.2.1 combine	89
5.59.2.2 getAlgId	89
5.60 Arc::Config Class Reference	90
5.60.1 Detailed Description	90
5.60.2 Constructor & Destructor Documentation	91
5.60.2.1 Config	91
5.60.2.2 Config	91
5.60.2.3 Config	91
5.60.2.4 Config	91
5.60.2.5 Config	91
5.60.2.6 Config	91
5.60.3 Member Function Documentation	91
5.60.3.1 getFileName	91
5.60.3.2 parse	91
5.60.3.3 print	91
5.60.3.4 save	92
5.60.3.5 setFileName	92
5.61 Arc::ConfusaCertHandler Class Reference	92
5.61.1 Detailed Description	92
5.61.2 Constructor & Destructor Documentation	92
5.61.2.1 ConfusaCertHandler	92
5.61.3 Member Function Documentation	92
5.61.3.1 createCertRequest	92
5.61.3.2 getCertRequestB64	93
5.62 Arc::ConfusaParserUtils Class Reference	93
5.62.1 Detailed Description	93
5.62.2 Member Function Documentation	93
5.62.2.1 destroy_doc	93
5.62.2.2 evaluate_path	93
5.62.2.3 extract_body_information	94
5.62.2.4 get_doc	94
5.62.2.5 handle_redirect_step	94
5.62.2.6 urlencode	94

5.62.2.7	urlencode_params	94
5.63	Arc::CountedPointer< T > Class Template Reference	94
5.63.1	Detailed Description	95
5.64	Arc::Counter Class Reference	95
5.64.1	Detailed Description	96
5.64.2	Member Typedef Documentation	97
5.64.2.1	IDType	97
5.64.3	Constructor & Destructor Documentation	98
5.64.3.1	Counter	98
5.64.3.2	~Counter	98
5.64.4	Member Function Documentation	98
5.64.4.1	cancel	98
5.64.4.2	changeExcess	98
5.64.4.3	changeLimit	99
5.64.4.4	extend	99
5.64.4.5	getCounterTicket	99
5.64.4.6	getCurrentTime	100
5.64.4.7	getExcess	100
5.64.4.8	getExpirationReminder	100
5.64.4.9	getExpiryTime	101
5.64.4.10	getLimit	101
5.64.4.11	getValue	101
5.64.4.12	reserve	102
5.64.4.13	setExcess	102
5.64.4.14	setLimit	102
5.65	Arc::CounterTicket Class Reference	103
5.65.1	Detailed Description	103
5.65.2	Constructor & Destructor Documentation	104
5.65.2.1	CounterTicket	104
5.65.3	Member Function Documentation	104
5.65.3.1	cancel	104
5.65.3.2	extend	104
5.65.3.3	isValid	104
5.66	Arc::CRC32Sum Class Reference	105

5.66.1	Detailed Description	105
5.66.2	Member Function Documentation	105
5.66.2.1	add	105
5.66.2.2	end	106
5.66.2.3	print	106
5.66.2.4	scan	106
5.66.2.5	start	106
5.67	Arc::Credential Class Reference	107
5.67.1	Detailed Description	108
5.67.2	Constructor & Destructor Documentation	109
5.67.2.1	Credential	109
5.67.2.2	Credential	109
5.67.2.3	Credential	109
5.67.2.4	Credential	109
5.67.2.5	Credential	110
5.67.2.6	Credential	110
5.67.3	Member Function Documentation	110
5.67.3.1	AddCertExtObj	110
5.67.3.2	AddExtension	110
5.67.3.3	AddExtension	111
5.67.3.4	GenerateEECRequest	111
5.67.3.5	GenerateEECRequest	111
5.67.3.6	GenerateEECRequest	111
5.67.3.7	GenerateRequest	111
5.67.3.8	GenerateRequest	112
5.67.3.9	GenerateRequest	112
5.67.3.10	GetCAName	112
5.67.3.11	GetCert	112
5.67.3.12	GetCertNumofChain	112
5.67.3.13	GetCertReq	112
5.67.3.14	GetDN	112
5.67.3.15	GetEndTime	112
5.67.3.16	GetExtension	112
5.67.3.17	getFormat	113

5.67.3.18 GetIdentityName	113
5.67.3.19 GetIssuerName	113
5.67.3.20 GetLifeTime	113
5.67.3.21 GetPrivKey	113
5.67.3.22 GetProxyPolicy	113
5.67.3.23 GetPubKey	113
5.67.3.24 GetStartTime	113
5.67.3.25 GetType	113
5.67.3.26 GetVerification	113
5.67.3.27 InitProxyCertInfo	114
5.67.3.28 InquireRequest	114
5.67.3.29 InquireRequest	114
5.67.3.30 InquireRequest	114
5.67.3.31 IsCredentialsValid	114
5.67.3.32 IsValid	114
5.67.3.33 LogError	114
5.67.3.34 OutputCertificate	114
5.67.3.35 OutputCertificateChain	115
5.67.3.36 OutputPrivatekey	115
5.67.3.37 OutputPublickey	115
5.67.3.38 SelfSignEECRequest	115
5.67.3.39 SetLifeTime	116
5.67.3.40 SetProxyPolicy	116
5.67.3.41 SetStartTime	116
5.67.3.42 SignEECRequest	116
5.67.3.43 SignEECRequest	116
5.67.3.44 SignEECRequest	116
5.67.3.45 SignRequest	116
5.67.3.46 SignRequest	116
5.67.3.47 SignRequest	117
5.67.3.48 STACK_OF	117
5.68 Arc::CredentialError Class Reference	117
5.68.1 Detailed Description	117
5.68.2 Constructor & Destructor Documentation	117

5.68.2.1	CredentialError	117
5.69	Arc::CredentialStore Class Reference	118
5.69.1	Detailed Description	118
5.70	Arc::Database Class Reference	118
5.70.1	Detailed Description	119
5.70.2	Constructor & Destructor Documentation	119
5.70.2.1	Database	119
5.70.2.2	Database	119
5.70.2.3	Database	119
5.70.2.4	~Database	119
5.70.3	Member Function Documentation	119
5.70.3.1	close	119
5.70.3.2	connect	119
5.70.3.3	enable_ssl	120
5.70.3.4	isconnected	120
5.70.3.5	shutdown	120
5.71	DataStaging::DataDelivery Class Reference	120
5.71.1	Detailed Description	121
5.71.2	Member Function Documentation	121
5.71.2.1	receiveDTR	121
5.72	DataStaging::DataDeliveryComm Class Reference	121
5.72.1	Detailed Description	123
5.72.2	Member Enumeration Documentation	123
5.72.2.1	CommStatusType	123
5.72.3	Constructor & Destructor Documentation	123
5.72.3.1	DataDeliveryComm	123
5.72.4	Member Function Documentation	123
5.72.4.1	PullStatus	123
5.73	DataStaging::DataDeliveryCommHandler Class Reference	124
5.73.1	Detailed Description	124
5.74	DataStaging::DataDeliveryLocalComm Class Reference	124
5.74.1	Detailed Description	125
5.75	DataStaging::DataDeliveryRemoteComm Class Reference	125
5.75.1	Detailed Description	125

5.76 ArcSec::DateAttribute Class Reference	126
5.76.1 Member Function Documentation	126
5.76.1.1 encode	126
5.76.1.2 equal	126
5.76.1.3 getId	126
5.76.1.4 getType	126
5.77 ArcSec::DateTimeAttribute Class Reference	127
5.77.1 Detailed Description	127
5.77.2 Member Function Documentation	127
5.77.2.1 encode	127
5.77.2.2 equal	127
5.77.2.3 getId	127
5.77.2.4 getType	128
5.78 Arc::DBranch Class Reference	128
5.79 Arc::DelegationConsumer Class Reference	128
5.79.1 Detailed Description	129
5.79.2 Constructor & Destructor Documentation	129
5.79.2.1 DelegationConsumer	129
5.79.2.2 DelegationConsumer	129
5.79.3 Member Function Documentation	129
5.79.3.1 Acquire	129
5.79.3.2 Acquire	129
5.79.3.3 Backup	129
5.79.3.4 Generate	129
5.79.3.5 ID	129
5.79.3.6 LogError	130
5.79.3.7 Request	130
5.79.3.8 Restore	130
5.80 Arc::DelegationConsumerSOAP Class Reference	130
5.80.1 Detailed Description	130
5.80.2 Constructor & Destructor Documentation	131
5.80.2.1 DelegationConsumerSOAP	131
5.80.2.2 DelegationConsumerSOAP	131
5.80.3 Member Function Documentation	131

5.80.3.1	DelegateCredentialsInit	131
5.80.3.2	DelegatedToken	131
5.80.3.3	UpdateCredentials	131
5.80.3.4	UpdateCredentials	131
5.81	Arc::DelegationContainerSOAP Class Reference	131
5.81.1	Detailed Description	132
5.81.2	Member Function Documentation	132
5.81.2.1	DelegateCredentialsInit	132
5.81.2.2	DelegatedToken	132
5.81.2.3	UpdateCredentials	132
5.81.3	Field Documentation	133
5.81.3.1	context_lock_	133
5.81.3.2	max_duration_	133
5.81.3.3	max_size_	133
5.81.3.4	max_usage_	133
5.82	Arc::DelegationProvider Class Reference	133
5.82.1	Detailed Description	134
5.82.2	Constructor & Destructor Documentation	134
5.82.2.1	DelegationProvider	134
5.82.2.2	DelegationProvider	134
5.82.3	Member Function Documentation	134
5.82.3.1	Delegate	134
5.83	Arc::DelegationProviderSOAP Class Reference	134
5.83.1	Detailed Description	135
5.83.2	Constructor & Destructor Documentation	135
5.83.2.1	DelegationProviderSOAP	135
5.83.2.2	DelegationProviderSOAP	135
5.83.3	Member Function Documentation	135
5.83.3.1	DelegateCredentialsInit	136
5.83.3.2	DelegateCredentialsInit	136
5.83.3.3	DelegatedToken	136
5.83.3.4	ID	136
5.83.3.5	UpdateCredentials	136
5.83.3.6	UpdateCredentials	136

5.84 ArcSec::DenyOverridesCombiningAlg Class Reference	137
5.84.1 Detailed Description	137
5.84.2 Member Function Documentation	137
5.84.2.1 combine	137
5.84.2.2 getalgId	138
5.85 Arc::DiskSpaceRequirementType Class Reference	138
5.85.1 Field Documentation	138
5.85.1.1 CacheDiskSpace	138
5.85.1.2 DiskSpace	138
5.85.1.3 SessionDiskSpace	138
5.86 Arc::DItem Class Reference	139
5.87 Arc::DItemString Class Reference	139
5.88 Arc::DNListHandlerConfig Class Reference	139
5.89 DataStaging::DTR Class Reference	140
5.89.1 Detailed Description	142
5.89.2 Constructor & Destructor Documentation	142
5.89.2.1 DTR	143
5.89.3 Member Function Documentation	143
5.89.3.1 registerCallback	143
5.89.3.2 reset	143
5.89.3.3 set_error_status	143
5.90 DataStaging::DTRCallback Class Reference	144
5.90.1 Detailed Description	144
5.90.2 Constructor & Destructor Documentation	144
5.90.2.1 ~DTRCallback	144
5.90.3 Member Function Documentation	144
5.90.3.1 receiveDTR	144
5.91 DataStaging::DTRErrorStatus Class Reference	145
5.91.1 Detailed Description	145
5.91.2 Member Enumeration Documentation	146
5.91.2.1 DTRErrorLocation	146
5.91.2.2 DTRErrorStatusType	146
5.92 DataStaging::DTRLList Class Reference	146
5.92.1 Detailed Description	147

5.92.2 Member Function Documentation	147
5.92.2.1 add_dtr	147
5.92.2.2 delete_dtr	147
5.92.2.3 dumpState	147
5.92.2.4 filter_dtrs_by_job	148
5.92.2.5 filter_dtrs_by_next_receiver	148
5.92.2.6 filter_dtrs_by_owner	148
5.92.2.7 filter_dtrs_by_status	148
5.92.2.8 filter_pending_dtrs	148
5.93 DataStaging::DTRStatus Class Reference	149
5.93.1 Detailed Description	150
5.93.2 Member Enumeration Documentation	150
5.93.2.1 DTRStatusType	150
5.94 ArcSec::DurationAttribute Class Reference	151
5.94.1 Detailed Description	151
5.94.2 Member Function Documentation	152
5.94.2.1 encode	152
5.94.2.2 equal	152
5.94.2.3 getId	152
5.94.2.4 getType	152
5.95 Arc::EnvLockWrapper Class Reference	152
5.96 ArcSec::EqualFunction Class Reference	152
5.96.1 Detailed Description	153
5.96.2 Member Function Documentation	153
5.96.2.1 evaluate	153
5.96.2.2 evaluate	153
5.96.2.3 getFunctionName	153
5.97 ArcSec::EvalResult Struct Reference	154
5.97.1 Detailed Description	154
5.98 ArcSec::EvaluationCtx Class Reference	154
5.98.1 Detailed Description	154
5.98.2 Constructor & Destructor Documentation	154
5.98.2.1 EvaluationCtx	154
5.99 ArcSec::Evaluator Class Reference	154

5.99.1	Detailed Description	155
5.99.2	Member Function Documentation	155
5.99.2.1	addPolicy	155
5.99.2.2	addPolicy	156
5.99.2.3	evaluate	156
5.99.2.4	evaluate	156
5.99.2.5	evaluate	156
5.99.2.6	evaluate	156
5.99.2.7	evaluate	156
5.99.2.8	evaluate	156
5.99.2.9	evaluate	157
5.99.2.10	getAlgFactory	157
5.99.2.11	getAttrFactory	157
5.99.2.12	getFnFactory	157
5.99.2.13	getName	157
5.99.2.14	setCombiningAlg	157
5.99.2.15	setCombiningAlg	157
5.100	ArcSec::EvaluatorContext Class Reference	158
5.100.1	Detailed Description	158
5.100.2	Member Function Documentation	158
5.100.2.1	operator AlgFactory *	158
5.100.2.2	operator AttributeFactory *	158
5.100.2.3	operator FnFactory *	158
5.101	ArcSec::EvaluatorLoader Class Reference	158
5.101.1	Detailed Description	159
5.101.2	Member Function Documentation	159
5.101.2.1	getEvaluator	159
5.101.2.2	getEvaluator	159
5.101.2.3	getEvaluator	159
5.101.2.4	getPolicy	159
5.101.2.5	getPolicy	159
5.101.2.6	getRequest	160
5.101.2.7	getRequest	160
5.102	Arc::ExecutableType Class Reference	160

5.103Arc::ExecutionTarget Class Reference	160
5.103.1 Detailed Description	161
5.103.2 Constructor & Destructor Documentation	161
5.103.2.1 ExecutionTarget	161
5.103.2.2 ExecutionTarget	161
5.103.2.3 ExecutionTarget	161
5.103.3 Member Function Documentation	161
5.103.3.1 GetSubmitter	161
5.103.3.2 operator=	162
5.103.3.3 Print	162
5.103.3.4 SaveToStream	162
5.103.3.5 Update	162
5.103.4 Field Documentation	163
5.103.4.1 ApplicationEnvironments	163
5.103.4.2 ComputingShareName	163
5.103.4.3 FreeSlotsWithDuration	163
5.103.4.4 MaxDiskSpace	163
5.103.4.5 MaxMainMemory	163
5.103.4.6 MaxVirtualMemory	163
5.103.4.7 OperatingSystem	164
5.104Arc::ExpirationReminder Class Reference	164
5.104.1 Detailed Description	164
5.104.2 Member Function Documentation	165
5.104.2.1 getExpiryTime	165
5.104.2.2 getReservationID	165
5.104.2.3 operator<	165
5.105Arc::FileAccess Class Reference	165
5.105.1 Detailed Description	166
5.105.2 Member Function Documentation	167
5.105.2.1 copy	167
5.105.2.2 geterrno	167
5.105.2.3 mkdirp	167
5.105.2.4 mkstemp	167
5.105.2.5 open	167

5.105.2.6 opendir	167
5.105.2.7 setuid	167
5.106 Arc::FileLock Class Reference	167
5.106.1 Detailed Description	168
5.106.2 Constructor & Destructor Documentation	168
5.106.2.1 FileLock	168
5.106.3 Member Function Documentation	169
5.106.3.1 acquire	169
5.106.3.2 acquire	169
5.106.3.3 check	169
5.106.3.4 release	169
5.107 Arc::FileType Class Reference	170
5.107.1 Field Documentation	170
5.107.1.1 Checksum	170
5.108 Arc::FinderLoader Class Reference	170
5.109 ArcSec::FnFactory Class Reference	170
5.109.1 Detailed Description	171
5.109.2 Member Function Documentation	171
5.109.2.1 createFn	171
5.110 ArcSec::Function Class Reference	171
5.110.1 Detailed Description	172
5.110.2 Member Function Documentation	172
5.110.2.1 evaluate	172
5.110.2.2 evaluate	172
5.111 GDS20::GDS20Service Class Reference	172
5.111.1 Member Function Documentation	172
5.111.1.1 process	173
5.112 DataStaging::Generator Class Reference	173
5.112.1 Detailed Description	173
5.112.2 Member Function Documentation	174
5.112.2.1 receiveDTR	174
5.113 ArcSec::GenericAttribute Class Reference	174
5.113.1 Member Function Documentation	174
5.113.1.1 encode	174

5.113.1.2 equal	175
5.113.1.3 getId	175
5.113.1.4 getType	175
5.114 Arc::GlobusResult Class Reference	175
5.115 Arc::GLUE2 Class Reference	175
5.115.1 Detailed Description	175
5.116 Arc::GSSCredential Class Reference	176
5.117 Arc::HakaClient Class Reference	176
5.117.1 Member Function Documentation	176
5.117.1.1 processConsent	176
5.117.1.2 processIdP2Confusa	176
5.117.1.3 processIdPLogin	177
5.118 Arc::FileAccess::header_t Struct Reference	177
5.119 Arc::HTTPClientInfo Struct Reference	177
5.120 Arc::InfoCache Class Reference	177
5.120.1 Detailed Description	177
5.120.2 Constructor & Destructor Documentation	177
5.120.2.1 InfoCache	178
5.121 Arc::InfoCacheInterface Class Reference	178
5.121.1 Member Function Documentation	178
5.121.1.1 Get	178
5.122 Arc::InfoFilter Class Reference	178
5.122.1 Detailed Description	179
5.122.2 Constructor & Destructor Documentation	179
5.122.2.1 InfoFilter	179
5.122.3 Member Function Documentation	179
5.122.3.1 Filter	179
5.122.3.2 Filter	179
5.123 Arc::InfoRegister Class Reference	180
5.123.1 Detailed Description	180
5.124 Arc::InfoRegisterContainer Class Reference	180
5.124.1 Detailed Description	180
5.124.2 Member Function Documentation	180
5.124.2.1 addRegistrar	180

5.124.2.2 addService	181
5.124.2.3 removeService	181
5.125 Arc::InfoRegisters Class Reference	181
5.125.1 Detailed Description	181
5.125.2 Constructor & Destructor Documentation	181
5.125.2.1 InfoRegisters	181
5.126 Arc::InfoRegistrar Class Reference	182
5.126.1 Detailed Description	182
5.126.2 Member Function Documentation	182
5.126.2.1 addService	182
5.126.2.2 registration	182
5.127 Arc::InformationContainer Class Reference	182
5.127.1 Detailed Description	183
5.127.2 Constructor & Destructor Documentation	183
5.127.2.1 InformationContainer	183
5.127.3 Member Function Documentation	183
5.127.3.1 Acquire	183
5.127.3.2 Assign	184
5.127.3.3 Get	184
5.127.4 Field Documentation	184
5.127.4.1 doc_	184
5.128 Arc::InformationInterface Class Reference	184
5.128.1 Detailed Description	185
5.128.2 Constructor & Destructor Documentation	185
5.128.2.1 InformationInterface	185
5.128.3 Member Function Documentation	185
5.128.3.1 Get	185
5.128.4 Field Documentation	185
5.128.4.1 lock_	185
5.129 Arc::InformationRequest Class Reference	185
5.129.1 Detailed Description	186
5.129.2 Constructor & Destructor Documentation	186
5.129.2.1 InformationRequest	186
5.129.2.2 InformationRequest	186

5.129.2.3 InformationRequest	186
5.129.2.4 InformationRequest	186
5.129.3 Member Function Documentation	186
5.129.3.1 SOAP	186
5.130 Arc::InformationResponse Class Reference	187
5.130.1 Detailed Description	187
5.130.2 Constructor & Destructor Documentation	187
5.130.2.1 InformationResponse	187
5.130.3 Member Function Documentation	187
5.130.3.1 Result	187
5.131 Arc::IniConfig Class Reference	187
5.132 Arc::InitializeCredentialsType Class Reference	188
5.133 ArcSec::InRangeFunction Class Reference	188
5.133.1 Member Function Documentation	188
5.133.1.1 evaluate	188
5.133.1.2 evaluate	188
5.134 Arc::IntraProcessCounter Class Reference	189
5.134.1 Detailed Description	189
5.134.2 Constructor & Destructor Documentation	190
5.134.2.1 IntraProcessCounter	190
5.134.2.2 ~IntraProcessCounter	190
5.134.3 Member Function Documentation	190
5.134.3.1 cancel	190
5.134.3.2 changeExcess	190
5.134.3.3 changeLimit	191
5.134.3.4 extend	191
5.134.3.5 getExcess	191
5.134.3.6 getLimit	192
5.134.3.7 getValue	192
5.134.3.8 reserve	192
5.134.3.9 setExcess	193
5.134.3.10 setLimit	193
5.135 Arc::ISIS_description Struct Reference	193
5.136 Arc::String Class Reference	194

5.137 Arc::JobDescriptionParserLoader::iterator Class Reference	194
5.138 Arc::Job Class Reference	194
5.138.1 Detailed Description	195
5.138.2 Constructor & Destructor Documentation	195
5.138.2.1 Job	195
5.138.3 Member Function Documentation	195
5.138.3.1 operator=	195
5.138.3.2 Print	195
5.138.3.3 ReadAllJobsFromFile	196
5.138.3.4 ReadJobIDsFromFile	196
5.138.3.5 RemoveJobsFromFile	197
5.138.3.6 SaveToStream	198
5.138.3.7 ToXML	198
5.138.3.8 WriteJobIDsToFile	198
5.138.3.9 WriteJobIDToFile	199
5.138.3.10 WriteJobsToFile	199
5.138.3.11 WriteJobsToFile	200
5.138.3.12 WriteJobsToTruncatedFile	200
5.139 Arc::JobController Class Reference	201
5.139.1 Detailed Description	201
5.139.2 Member Function Documentation	202
5.139.2.1 Cat	202
5.139.2.2 Cat	202
5.139.2.3 Migrate	203
5.139.2.4 PrintJobStatus	203
5.139.2.5 SaveJobStatusToStream	204
5.140 Arc::JobControllerLoader Class Reference	204
5.140.1 Detailed Description	205
5.140.2 Constructor & Destructor Documentation	205
5.140.2.1 JobControllerLoader	205
5.140.2.2 ~JobControllerLoader	205
5.140.3 Member Function Documentation	205
5.140.3.1 GetJobControllers	205
5.140.3.2 load	206

5.141 Arc::JobControllerPluginArgument Class Reference	206
5.142 Arc::JobDescription Class Reference	206
5.142.1 Detailed Description	207
5.142.2 Member Function Documentation	207
5.142.2.1 GetSourceLanguage	207
5.142.2.2 operator bool	207
5.142.2.3 Parse	208
5.142.2.4 Parse	208
5.142.2.5 Parse	208
5.142.2.6 Print	208
5.142.2.7 SaveToStream	209
5.142.2.8 UnParse	209
5.142.2.9 UnParse	209
5.142.3 Field Documentation	210
5.142.3.1 OtherAttributes	210
5.143 Arc::JobDescriptionParser Class Reference	210
5.143.1 Detailed Description	210
5.144 Arc::JobDescriptionParserLoader Class Reference	210
5.144.1 Detailed Description	211
5.144.2 Constructor & Destructor Documentation	211
5.144.2.1 JobDescriptionParserLoader	211
5.144.2.2 ~JobDescriptionParserLoader	211
5.144.3 Member Function Documentation	211
5.144.3.1 GetJobDescriptionParsers	211
5.144.3.2 load	212
5.145 Arc::JobIdentificationType Class Reference	212
5.146 Arc::JobState Class Reference	212
5.146.1 Detailed Description	212
5.146.2 Member Function Documentation	213
5.146.2.1 IsFinished	213
5.147 Arc::JobSupervisor Class Reference	213
5.147.1 Detailed Description	214
5.147.2 Constructor & Destructor Documentation	214
5.147.2.1 JobSupervisor	214

5.147.2.2 JobSupervisor	214
5.147.3 Member Function Documentation	215
5.147.3.1 Cancel	215
5.147.3.2 Clean	215
5.147.3.3 Get	216
5.147.3.4 GetJobControllers	217
5.147.3.5 Kill	217
5.147.3.6 Migrate	217
5.147.3.7 Renew	219
5.147.3.8 Resubmit	219
5.147.3.9 Resume	220
5.148 Arc::LoadableModuleDescription Class Reference	221
5.149 Arc::Loader Class Reference	221
5.149.1 Detailed Description	222
5.149.2 Constructor & Destructor Documentation	222
5.149.2.1 Loader	222
5.149.2.2 ~Loader	222
5.149.3 Field Documentation	222
5.149.3.1 factory_	222
5.150 Arc::LogDestination Class Reference	223
5.150.1 Detailed Description	223
5.150.2 Constructor & Destructor Documentation	223
5.150.2.1 LogDestination	223
5.150.2.2 LogDestination	223
5.151 Arc::LogFile Class Reference	224
5.151.1 Detailed Description	224
5.151.2 Constructor & Destructor Documentation	224
5.151.2.1 LogFile	224
5.151.2.2 LogFile	225
5.151.3 Member Function Documentation	225
5.151.3.1 log	225
5.151.3.2 setBackups	225
5.151.3.3 setMaxSize	225
5.151.3.4 setReopen	225

5.152Arc::Logger Class Reference	226
5.152.1 Detailed Description	227
5.152.2 Constructor & Destructor Documentation	227
5.152.2.1 Logger	227
5.152.2.2 Logger	227
5.152.2.3 ~Logger	227
5.152.3 Member Function Documentation	228
5.152.3.1 addDestination	228
5.152.3.2 addDestinations	228
5.152.3.3 getDestinations	228
5.152.3.4 getLogger	228
5.152.3.5 getThreshold	228
5.152.3.6 msg	228
5.152.3.7 msg	229
5.152.3.8 setThreadContext	229
5.152.3.9 setThreshold	229
5.152.3.10 setThresholdForDomain	229
5.152.3.11 setThresholdForDomain	230
5.153Arc::LoggerContext Class Reference	230
5.153.1 Detailed Description	230
5.154Arc::LoggerFormat Struct Reference	230
5.155Arc::LogMessage Class Reference	231
5.155.1 Detailed Description	231
5.155.2 Constructor & Destructor Documentation	231
5.155.2.1 LogMessage	231
5.155.2.2 LogMessage	232
5.155.3 Member Function Documentation	232
5.155.3.1 getLevel	232
5.155.3.2 setIdentifier	232
5.155.4 Friends And Related Function Documentation	232
5.155.4.1 Logger	232
5.155.4.2 operator<<	233
5.156Arc::LogStream Class Reference	233
5.156.1 Detailed Description	233

5.156.2 Constructor & Destructor Documentation	233
5.156.2.1 LogStream	233
5.156.2.2 LogStream	234
5.156.3 Member Function Documentation	234
5.156.3.1 log	234
5.157 ArcSec::MatchFunction Class Reference	234
5.157.1 Detailed Description	235
5.157.2 Member Function Documentation	235
5.157.2.1 evaluate	235
5.157.2.2 evaluate	235
5.157.2.3 getFunctionName	235
5.158 Arc::MCC Class Reference	236
5.158.1 Detailed Description	236
5.158.2 Constructor & Destructor Documentation	237
5.158.2.1 MCC	237
5.158.3 Member Function Documentation	237
5.158.3.1 AddSecHandler	237
5.158.3.2 Next	237
5.158.3.3 process	237
5.158.3.4 ProcessSecHandlers	237
5.158.3.5 Unlink	238
5.158.4 Field Documentation	238
5.158.4.1 logger	238
5.158.4.2 next_	238
5.158.4.3 sechandlers_	238
5.159 Arc::MCC_Status Class Reference	238
5.159.1 Detailed Description	239
5.159.2 Constructor & Destructor Documentation	239
5.159.2.1 MCC_Status	239
5.159.3 Member Function Documentation	239
5.159.3.1 getExplanation	239
5.159.3.2 getKind	239
5.159.3.3 getOrigin	239
5.159.3.4 isOk	240

5.159.3.5 operator bool	240
5.159.3.6 operator std::string	240
5.159.3.7 operator!	240
5.160 Arc::MCCConfig Class Reference	241
5.160.1 Member Function Documentation	241
5.160.1.1 MakeConfig	241
5.161 Arc::MCCTrface Class Reference	241
5.161.1 Detailed Description	242
5.161.2 Member Function Documentation	242
5.161.2.1 process	242
5.162 Arc::MCCLoader Class Reference	242
5.162.1 Detailed Description	243
5.162.2 Constructor & Destructor Documentation	243
5.162.2.1 MCCLoader	243
5.162.2.2 ~MCCLoader	243
5.162.3 Member Function Documentation	243
5.162.3.1 operator[]	244
5.163 Arc::MCCPluginArgument Class Reference	244
5.164 Arc::MD5Sum Class Reference	244
5.164.1 Detailed Description	245
5.164.2 Member Function Documentation	245
5.164.2.1 add	245
5.164.2.2 end	245
5.164.2.3 print	245
5.164.2.4 scan	246
5.164.2.5 start	246
5.165 Arc::MemoryAllocationException Class Reference	246
5.166 Arc::Message Class Reference	246
5.166.1 Detailed Description	247
5.166.2 Constructor & Destructor Documentation	247
5.166.2.1 Message	247
5.166.2.2 Message	248
5.166.2.3 Message	248
5.166.2.4 ~Message	248

5.166.3 Member Function Documentation	248
5.166.3.1 Attributes	248
5.166.3.2 Auth	248
5.166.3.3 AuthContext	248
5.166.3.4 AuthContext	248
5.166.3.5 Context	248
5.166.3.6 Context	249
5.166.3.7 operator=	249
5.166.3.8 Payload	249
5.166.3.9 Payload	249
5.167 Arc::MessageAttributes Class Reference	249
5.167.1 Detailed Description	250
5.167.2 Constructor & Destructor Documentation	250
5.167.2.1 MessageAttributes	250
5.167.3 Member Function Documentation	250
5.167.3.1 add	250
5.167.3.2 count	251
5.167.3.3 get	251
5.167.3.4 getAll	251
5.167.3.5 remove	251
5.167.3.6 removeAll	252
5.167.3.7 set	252
5.167.4 Field Documentation	252
5.167.4.1 attributes_	252
5.168 Arc::MessageAuth Class Reference	252
5.168.1 Detailed Description	253
5.168.2 Member Function Documentation	253
5.168.2.1 Export	253
5.168.2.2 Filter	253
5.169 Arc::MessageAuthContext Class Reference	254
5.169.1 Detailed Description	254
5.170 Arc::MessageContext Class Reference	254
5.170.1 Detailed Description	254
5.170.2 Member Function Documentation	254

5.170.2.1 Add	255
5.171 Arc::MessageContextElement Class Reference	255
5.171.1 Detailed Description	255
5.172 Arc::MessagePayload Class Reference	255
5.172.1 Detailed Description	256
5.173 Arc::ModuleDesc Class Reference	256
5.173.1 Detailed Description	256
5.174 Arc::ModuleManager Class Reference	256
5.174.1 Detailed Description	257
5.174.2 Constructor & Destructor Documentation	257
5.174.2.1 ModuleManager	257
5.174.3 Member Function Documentation	257
5.174.3.1 find	257
5.174.3.2 findLocation	257
5.174.3.3 load	257
5.174.3.4 makePersistent	257
5.174.3.5 makePersistent	258
5.174.3.6 reload	258
5.174.3.7 setCfg	258
5.174.3.8 unload	258
5.174.3.9 unload	258
5.175 Arc::MultiSecAttr Class Reference	258
5.175.1 Detailed Description	259
5.175.2 Member Function Documentation	259
5.175.2.1 Export	259
5.175.2.2 operator bool	259
5.176 Arc::MySQLDatabase Class Reference	259
5.176.1 Detailed Description	260
5.176.2 Member Function Documentation	260
5.176.2.1 close	260
5.176.2.2 connect	260
5.176.2.3 enable_ssl	260
5.176.2.4 isconnected	261
5.176.2.5 shutdown	261

5.177 Arc::MySQLQuery Class Reference	261
5.177.1 Member Function Documentation	261
5.177.1.1 execute	261
5.177.1.2 get_array	262
5.177.1.3 get_num_columns	262
5.177.1.4 get_num_rows	262
5.177.1.5 get_row	262
5.177.1.6 get_row	262
5.177.1.7 get_row_field	263
5.178 Arc::NotificationType Class Reference	263
5.179 Arc::NS Class Reference	263
5.180 Arc::OAuthConsumer Class Reference	263
5.180.1 Detailed Description	264
5.180.2 Constructor & Destructor Documentation	264
5.180.2.1 OAuthConsumer	264
5.180.3 Member Function Documentation	264
5.180.3.1 approveCSR	264
5.180.3.2 parseDN	265
5.180.3.3 processLogin	265
5.180.3.4 pushCSR	265
5.180.3.5 storeCert	265
5.181 Arc::OpenIdpClient Class Reference	265
5.181.1 Member Function Documentation	266
5.181.1.1 processConsent	266
5.181.1.2 processIdP2Confusa	266
5.181.1.3 processIdPLogin	266
5.182 Arc::OptionParser Class Reference	266
5.183 ArcSec::OrderedCombiningAlg Class Reference	266
5.184 passwd Struct Reference	267
5.185 Arc::PathIterator Class Reference	267
5.185.1 Detailed Description	267
5.185.2 Constructor & Destructor Documentation	267
5.185.2.1 PathIterator	267
5.185.3 Member Function Documentation	268

5.185.3.1 operator bool	268
5.185.3.2 operator*	268
5.185.3.3 operator++	268
5.185.3.4 operator--	268
5.185.3.5 Rest	268
5.186 Arc::PayloadRaw Class Reference	268
5.186.1 Detailed Description	269
5.186.2 Constructor & Destructor Documentation	269
5.186.2.1 PayloadRaw	269
5.186.2.2 ~PayloadRaw	269
5.186.3 Member Function Documentation	269
5.186.3.1 Buffer	269
5.186.3.2 BufferPos	269
5.186.3.3 BufferSize	270
5.186.3.4 Content	270
5.186.3.5 Insert	270
5.186.3.6 Insert	270
5.186.3.7 operator[]	270
5.186.3.8 Size	270
5.186.3.9 Truncate	270
5.187 Arc::PayloadRawBuf Struct Reference	271
5.187.1 Field Documentation	271
5.187.1.1 allocated	271
5.187.1.2 length	271
5.187.1.3 size	271
5.188 Arc::PayloadRawInterface Class Reference	271
5.188.1 Detailed Description	272
5.188.2 Member Function Documentation	272
5.188.2.1 Buffer	272
5.188.2.2 BufferPos	272
5.188.2.3 BufferSize	273
5.188.2.4 Content	273
5.188.2.5 Insert	273
5.188.2.6 Insert	273

5.188.2.7 operator[]	273
5.188.2.8 Size	273
5.188.2.9 Truncate	274
5.189 Arc::PayloadSOAP Class Reference	274
5.189.1 Detailed Description	274
5.189.2 Constructor & Destructor Documentation	274
5.189.2.1 PayloadSOAP	274
5.189.2.2 PayloadSOAP	275
5.189.2.3 PayloadSOAP	275
5.190 Arc::PayloadStream Class Reference	275
5.190.1 Detailed Description	276
5.190.2 Constructor & Destructor Documentation	276
5.190.2.1 PayloadStream	276
5.190.2.2 ~PayloadStream	276
5.190.3 Member Function Documentation	276
5.190.3.1 Get	276
5.190.3.2 Get	276
5.190.3.3 Get	276
5.190.3.4 Limit	277
5.190.3.5 operator bool	277
5.190.3.6 operator!	277
5.190.3.7 Pos	277
5.190.3.8 Put	277
5.190.3.9 Put	277
5.190.3.10 Put	277
5.190.3.11 Size	278
5.190.3.12 Timeout	278
5.190.3.13 Timeout	278
5.190.4 Field Documentation	278
5.190.4.1 handle_	278
5.190.4.2 seekable_	278
5.191 Arc::PayloadStreamInterface Class Reference	278
5.191.1 Detailed Description	279
5.191.2 Member Function Documentation	279

5.191.2.1 Get	279
5.191.2.2 Get	280
5.191.2.3 Get	280
5.191.2.4 Limit	280
5.191.2.5 operator bool	280
5.191.2.6 operator!=	280
5.191.2.7 Pos	280
5.191.2.8 Put	280
5.191.2.9 Put	281
5.191.2.10Put	281
5.191.2.11Size	281
5.191.2.12Timeout	281
5.191.2.13Timeout	281
5.192Arc::PayloadWSRF Class Reference	281
5.192.1 Detailed Description	282
5.192.2 Constructor & Destructor Documentation	282
5.192.2.1 PayloadWSRF	282
5.192.2.2 PayloadWSRF	282
5.192.2.3 PayloadWSRF	282
5.193ArcSec::PDP Class Reference	282
5.193.1 Detailed Description	283
5.194ArcSec::PDPConfigContext Class Reference	283
5.195ArcSec::PDPPPluginArgument Class Reference	283
5.196Arc::Period Class Reference	284
5.196.1 Constructor & Destructor Documentation	284
5.196.1.1 Period	284
5.196.1.2 Period	284
5.196.1.3 Period	284
5.196.1.4 Period	285
5.196.2 Member Function Documentation	285
5.196.2.1 GetPeriod	285
5.196.2.2 istr	285
5.196.2.3 operator std::string	285
5.196.2.4 operator!=	285

5.196.2.5 operator<	285
5.196.2.6 operator<=	285
5.196.2.7 operator=	285
5.196.2.8 operator!=	285
5.196.2.9 operator==	285
5.196.2.10 operator>	286
5.196.2.11 operator>=	286
5.196.2.12 SetPeriod	286
5.197 ArcSec::PeriodAttribute Class Reference	286
5.197.1 Detailed Description	286
5.197.2 Member Function Documentation	286
5.197.2.1 encode	286
5.197.2.2 equal	287
5.197.2.3 getId	287
5.197.2.4 getType	287
5.198 ArcSec::PermitOverridesCombiningAlg Class Reference	287
5.198.1 Detailed Description	288
5.198.2 Member Function Documentation	288
5.198.2.1 combine	288
5.198.2.2 getAlgId	288
5.199 Arc::Plexer Class Reference	288
5.199.1 Detailed Description	289
5.199.2 Constructor & Destructor Documentation	289
5.199.2.1 Plexer	289
5.199.2.2 ~Plexer	289
5.199.3 Member Function Documentation	290
5.199.3.1 Next	290
5.199.3.2 process	290
5.199.4 Field Documentation	290
5.199.4.1 logger	290
5.200 Arc::PlexerEntry Class Reference	290
5.200.1 Detailed Description	290
5.201 Arc::Plugin Class Reference	291
5.201.1 Detailed Description	291

5.202Arc::PluginArgument Class Reference	292
5.202.1 Detailed Description	292
5.202.2 Member Function Documentation	293
5.202.2.1 get_factory	293
5.202.2.2 get_module	293
5.203Arc::PluginDesc Class Reference	293
5.203.1 Detailed Description	293
5.204Arc::PluginDescriptor Struct Reference	293
5.204.1 Detailed Description	294
5.205Arc::PluginsFactory Class Reference	294
5.205.1 Detailed Description	294
5.205.2 Constructor & Destructor Documentation	295
5.205.2.1 PluginsFactory	295
5.205.3 Member Function Documentation	295
5.205.3.1 FilterByKind	295
5.205.3.2 load	295
5.205.3.3 report	295
5.205.3.4 scan	295
5.205.3.5 TryLoad	295
5.206ArcSec::Policy Class Reference	296
5.206.1 Detailed Description	296
5.206.2 Constructor & Destructor Documentation	296
5.206.2.1 Policy	296
5.206.2.2 Policy	297
5.206.3 Member Function Documentation	297
5.206.3.1 addPolicy	297
5.206.3.2 eval	297
5.206.3.3 getEffect	297
5.206.3.4 getEvalName	297
5.206.3.5 getEvalResult	297
5.206.3.6 getName	297
5.206.3.7 make_policy	298
5.206.3.8 setEvalResult	298
5.206.3.9 setEvaluatorContext	298

5.207 ArcSec::PolicyStore::PolicyElement Class Reference	298
5.208 ArcSec::PolicyParser Class Reference	298
5.208.1 Detailed Description	298
5.208.2 Member Function Documentation	299
5.208.2.1 parsePolicy	299
5.209 ArcSec::PolicyStore Class Reference	299
5.209.1 Detailed Description	299
5.209.2 Constructor & Destructor Documentation	299
5.209.2.1 PolicyStore	299
5.210 Arc::PrintF< T0, T1, T2, T3, T4, T5, T6, T7 > Class Template Reference	300
5.211 Arc::PrintFBase Class Reference	300
5.212 DataStaging::Processor Class Reference	300
5.212.1 Detailed Description	301
5.212.2 Member Function Documentation	301
5.212.2.1 receiveDTR	301
5.212.2.2 start	301
5.212.2.3 stop	302
5.213 Arc::Profile Class Reference	302
5.214 ArcCredential::PROXYCERTINFO_st Struct Reference	302
5.215 ArcCredential::PROXYPOLICY_st Struct Reference	302
5.216 Arc::Query Class Reference	302
5.216.1 Constructor & Destructor Documentation	303
5.216.1.1 Query	303
5.216.1.2 Query	303
5.216.1.3 ~Query	303
5.216.2 Member Function Documentation	303
5.216.2.1 execute	304
5.216.2.2 get_array	304
5.216.2.3 get_num_colums	304
5.216.2.4 get_num_rows	304
5.216.2.5 get_row	304
5.216.2.6 get_row	305
5.216.2.7 get_row_field	305
5.217 Arc::Range< T > Class Template Reference	305

5.218Arc::Register_Info_Type Struct Reference	305
5.219Arc::RegisteredService Class Reference	305
5.219.1 Detailed Description	306
5.219.2 Constructor & Destructor Documentation	306
5.219.2.1 RegisteredService	306
5.220Arc::RegularExpression Class Reference	306
5.220.1 Detailed Description	307
5.220.2 Member Function Documentation	307
5.220.2.1 match	307
5.221ArcSec::Request Class Reference	307
5.221.1 Detailed Description	308
5.221.2 Constructor & Destructor Documentation	308
5.221.2.1 Request	308
5.221.2.2 Request	308
5.221.3 Member Function Documentation	308
5.221.3.1 addRequestItem	308
5.221.3.2 getEvalName	308
5.221.3.3 getName	309
5.221.3.4 getRequestItems	309
5.221.3.5 make_request	309
5.221.3.6 setAttributeFactory	309
5.221.3.7 setRequestItems	309
5.222ArcSec::RequestAttribute Class Reference	309
5.222.1 Detailed Description	309
5.222.2 Constructor & Destructor Documentation	310
5.222.2.1 RequestAttribute	310
5.222.3 Member Function Documentation	310
5.222.3.1 duplicate	310
5.223ArcSec::RequestItem Class Reference	310
5.223.1 Detailed Description	310
5.223.2 Constructor & Destructor Documentation	310
5.223.2.1 RequestItem	310
5.224ArcSec::RequestTuple Class Reference	311
5.225Arc::ResourceSlotType Class Reference	311

5.226 Arc::ResourcesType Class Reference	311
5.227 ArcSec::Response Class Reference	311
5.227.1 Detailed Description	311
5.228 ArcSec::ResponseItem Class Reference	311
5.228.1 Detailed Description	312
5.229 ArcSec::ResponseList Class Reference	312
5.230 Arc::Run Class Reference	312
5.230.1 Detailed Description	313
5.230.2 Constructor & Destructor Documentation	313
5.230.2.1 Run	313
5.230.2.2 Run	313
5.230.2.3 ~Run	313
5.230.3 Member Function Documentation	313
5.230.3.1 Abandon	313
5.230.3.2 AfterFork	313
5.230.3.3 AssignStderr	313
5.230.3.4 AssignStdin	314
5.230.3.5 AssignStdout	314
5.230.3.6 AssignWorkingDirectory	314
5.230.3.7 CloseStderr	314
5.230.3.8 CloseStdin	314
5.230.3.9 CloseStdout	314
5.230.3.10 KeepStderr	314
5.230.3.11 KeepStdin	314
5.230.3.12 KeepStdout	314
5.230.3.13 Kill	314
5.230.3.14 operator bool	315
5.230.3.15 operator!	315
5.230.3.16 ReadStderr	315
5.230.3.17 ReadStdout	315
5.230.3.18 Result	315
5.230.3.19 Running	315
5.230.3.20 Start	315
5.230.3.21 Wait	315

5.230.3.22	Wait	315
5.230.3.23	WriteStdin	316
5.231	Arc::SAML2LoginClient Class Reference	316
5.231.1	Constructor & Destructor Documentation	316
5.231.1.1	SAML2LoginClient	316
5.231.2	Member Function Documentation	316
5.231.2.1	findSimpleSAMLInstallation	317
5.231.2.2	processLogin	317
5.232	Arc::SAML2SSOHTTPClient Class Reference	317
5.232.1	Member Function Documentation	318
5.232.1.1	approveCSR	318
5.232.1.2	parseDN	318
5.232.1.3	processConsent	318
5.232.1.4	processIdP2Confusa	318
5.232.1.5	processIdPLogin	318
5.232.1.6	processLogin	318
5.232.1.7	pushCSR	319
5.232.1.8	storeCert	319
5.233	Arc::SAMLToken Class Reference	319
5.233.1	Detailed Description	319
5.233.2	Member Enumeration Documentation	320
5.233.2.1	SAMLVersion	320
5.233.3	Constructor & Destructor Documentation	321
5.233.3.1	SAMLToken	321
5.233.3.2	SAMLToken	321
5.233.3.3	~SAMLToken	321
5.233.4	Member Function Documentation	321
5.233.4.1	Authenticate	321
5.233.4.2	Authenticate	322
5.233.4.3	operator bool	322
5.234	Arc::ScalableTime< T > Class Template Reference	322
5.235	Arc::ScalableTime< int > Class Template Reference	322
5.236	DataStaging::Scheduler Class Reference	322
5.236.1	Detailed Description	323

5.236.2 Member Function Documentation	323
5.236.2.1 receiveDTR	323
5.236.2.2 start	324
5.236.2.3 stop	324
5.237 Arc::SecAttr Class Reference	324
5.237.1 Detailed Description	325
5.237.2 Member Function Documentation	325
5.237.2.1 Export	325
5.237.2.2 Export	325
5.237.2.3 get	325
5.237.2.4 getAll	325
5.237.2.5 Import	326
5.237.2.6 operator bool	326
5.237.2.7 operator!=	326
5.237.2.8 operator==	326
5.238 Arc::SecAttrFormat Class Reference	326
5.238.1 Detailed Description	326
5.239 Arc::SecAttrValue Class Reference	327
5.239.1 Detailed Description	327
5.239.2 Member Function Documentation	327
5.239.2.1 operator bool	327
5.239.2.2 operator!=	328
5.239.2.3 operator==	328
5.240 ArcSec::SecHandler Class Reference	328
5.240.1 Detailed Description	328
5.241 Arc::SecHandlerConfig Class Reference	328
5.242 ArcSec::SecHandlerConfig Class Reference	329
5.242.1 Detailed Description	329
5.243 ArcSec::SecHandlerPluginArgument Class Reference	329
5.244 ArcSec::Security Class Reference	330
5.244.1 Detailed Description	330
5.245 Arc::Service Class Reference	330
5.245.1 Detailed Description	331
5.245.2 Constructor & Destructor Documentation	331

5.245.2.1 Service	331
5.245.3 Member Function Documentation	331
5.245.3.1 AddSecHandler	331
5.245.3.2 getID	332
5.245.3.3 ProcessSecHandlers	332
5.245.3.4 RegistrationCollector	332
5.245.4 Field Documentation	332
5.245.4.1 logger	332
5.245.4.2 sechandlers_	332
5.246 Arc::ServicePluginArgument Class Reference	332
5.247 Arc::SharedMutex Class Reference	333
5.248 Arc::SimpleCondition Class Reference	333
5.248.1 Detailed Description	333
5.248.2 Member Function Documentation	333
5.248.2.1 broadcast	333
5.248.2.2 lock	333
5.248.2.3 reset	334
5.248.2.4 signal	334
5.248.2.5 signal_nonblock	334
5.248.2.6 unlock	334
5.248.2.7 wait	334
5.248.2.8 wait	334
5.248.2.9 wait_nonblock	334
5.249 Arc::SimpleCounter Class Reference	334
5.249.1 Member Function Documentation	334
5.249.1.1 wait	334
5.250 Arc::SOAPMessage Class Reference	335
5.250.1 Detailed Description	335
5.250.2 Constructor & Destructor Documentation	335
5.250.2.1 SOAPMessage	335
5.250.2.2 SOAPMessage	335
5.250.2.3 SOAPMessage	335
5.250.2.4 ~SOAPMessage	336
5.250.3 Member Function Documentation	336

5.250.3.1 Attributes	336
5.250.3.2 Payload	336
5.250.3.3 Payload	336
5.251 Arc::Software Class Reference	336
5.251.1 Detailed Description	337
5.251.2 Member Typedef Documentation	338
5.251.2.1 ComparisonOperator	338
5.251.3 Member Enumeration Documentation	338
5.251.3.1 ComparisonOperatorEnum	338
5.251.4 Constructor & Destructor Documentation	338
5.251.4.1 Software	338
5.251.4.2 Software	339
5.251.4.3 Software	339
5.251.4.4 Software	339
5.251.5 Member Function Documentation	339
5.251.5.1 convert	339
5.251.5.2 empty	340
5.251.5.3 getFamily	340
5.251.5.4 getName	340
5.251.5.5 getVersion	340
5.251.5.6 operator std::string	341
5.251.5.7 operator!=	341
5.251.5.8 operator()	341
5.251.5.9 operator<	341
5.251.5.10 operator<=	342
5.251.5.11 operator==	342
5.251.5.12 operator>	343
5.251.5.13 operator>=	343
5.251.5.14 toString	344
5.251.6 Friends And Related Function Documentation	344
5.251.6.1 operator<<	344
5.251.7 Field Documentation	344
5.251.7.1 VERSIONTOKENS	344
5.252 Arc::SoftwareRequirement Class Reference	344

5.252.1 Detailed Description	345
5.252.2 Constructor & Destructor Documentation	345
5.252.2.1 SoftwareRequirement	345
5.252.2.2 SoftwareRequirement	346
5.252.2.3 SoftwareRequirement	346
5.252.2.4 SoftwareRequirement	346
5.252.3 Member Function Documentation	346
5.252.3.1 add	346
5.252.3.2 add	347
5.252.3.3 clear	347
5.252.3.4 empty	347
5.252.3.5 getComparisonOperatorList	347
5.252.3.6 getSoftwareList	348
5.252.3.7 isResolved	348
5.252.3.8 isSatisfied	348
5.252.3.9 isSatisfied	349
5.252.3.10 isSatisfied	349
5.252.3.11 operator=	350
5.252.3.12 selectSoftware	350
5.252.3.13 selectSoftware	351
5.252.3.14 selectSoftware	351
5.253 ArcSec::Source Class Reference	352
5.253.1 Detailed Description	352
5.253.2 Constructor & Destructor Documentation	352
5.253.2.1 Source	352
5.253.2.2 Source	353
5.254 ArcSec::SourceFile Class Reference	353
5.254.1 Detailed Description	353
5.255 ArcSec::SourceURL Class Reference	353
5.255.1 Detailed Description	354
5.256 DataStaging::DataDeliveryComm::Status Struct Reference	354
5.256.1 Detailed Description	355
5.257 ArcSec::StringAttribute Class Reference	355
5.257.1 Member Function Documentation	355

5.257.1.1 encode	355
5.257.1.2 equal	355
5.257.1.3 getId	355
5.257.1.4 getType	356
5.258 Arc::Submitter Class Reference	356
5.258.1 Detailed Description	356
5.258.2 Member Function Documentation	356
5.258.2.1 GetTestJob	357
5.258.2.2 Migrate	357
5.258.2.3 Submit	357
5.259 Arc::SubmitterLoader Class Reference	357
5.259.1 Detailed Description	358
5.259.2 Constructor & Destructor Documentation	358
5.259.2.1 SubmitterLoader	358
5.259.2.2 ~SubmitterLoader	358
5.259.3 Member Function Documentation	358
5.259.3.1 GetSubmitters	358
5.259.3.2 load	358
5.260 Arc::SubmitterPluginArgument Class Reference	359
5.261 Arc::TargetGenerator Class Reference	359
5.261.1 Detailed Description	360
5.261.2 Constructor & Destructor Documentation	360
5.261.2.1 TargetGenerator	360
5.261.3 Member Function Documentation	360
5.261.3.1 AddIndexServer	360
5.261.3.2 AddJob	361
5.261.3.3 AddJob	361
5.261.3.4 AddService	361
5.261.3.5 AddTarget	361
5.261.3.6 FoundJobs	362
5.261.3.7 FoundTargets	362
5.261.3.8 GetExecutionTargets	362
5.261.3.9 GetJobs	362
5.261.3.10 GetTargets	363

5.261.3.11ModifyFoundTargets	363
5.261.3.12PrintTargetInfo	363
5.261.3.13RetrieveExecutionTargets	363
5.261.3.14RetrieveJobs	364
5.261.3.15SaveTargetInfoToStream	364
5.261.3.16ServiceCounter	364
5.262Arc::TargetRetriever Class Reference	365
5.262.1 Detailed Description	365
5.262.2 Constructor & Destructor Documentation	365
5.262.2.1 TargetRetriever	365
5.262.3 Member Function Documentation	366
5.262.3.1 GetExecutionTargets	366
5.262.3.2 GetJobs	366
5.262.3.3 GetTargets	366
5.263Arc::TargetRetrieverLoader Class Reference	367
5.263.1 Detailed Description	367
5.263.2 Constructor & Destructor Documentation	367
5.263.2.1 TargetRetrieverLoader	367
5.263.2.2 ~TargetRetrieverLoader	367
5.263.3 Member Function Documentation	367
5.263.3.1 GetTargetRetrievers	367
5.263.3.2 load	368
5.264Arc::TargetRetrieverPluginArgument Class Reference	368
5.265Arc::ThreadDataItem Class Reference	368
5.265.1 Detailed Description	369
5.265.2 Constructor & Destructor Documentation	369
5.265.2.1 ThreadDataItem	369
5.265.2.2 ThreadDataItem	369
5.265.2.3 ThreadDataItem	369
5.265.3 Member Function Documentation	369
5.265.3.1 Attach	369
5.265.3.2 Attach	370
5.265.3.3 Dup	370
5.265.3.4 Get	370

5.266 Arc::ThreadInitializer Class Reference	370
5.267 Arc::ThreadRegistry Class Reference	370
5.267.1 Detailed Description	370
5.267.2 Member Function Documentation	371
5.267.2.1 WaitForExit	371
5.267.2.2 WaitOrCancel	371
5.268 Arc::Time Class Reference	371
5.268.1 Detailed Description	372
5.268.2 Constructor & Destructor Documentation	372
5.268.2.1 Time	372
5.268.2.2 Time	372
5.268.2.3 Time	372
5.268.2.4 Time	372
5.268.3 Member Function Documentation	372
5.268.3.1 GetFormat	372
5.268.3.2 GetTime	372
5.268.3.3 operator std::string	372
5.268.3.4 operator!=	373
5.268.3.5 operator+	373
5.268.3.6 operator-	373
5.268.3.7 operator-	373
5.268.3.8 operator<	373
5.268.3.9 operator<=	373
5.268.3.10 operator=	373
5.268.3.11 operator=	373
5.268.3.12 operator=	373
5.268.3.13 operator=	373
5.268.3.14 operator==	374
5.268.3.15 operator>	374
5.268.3.16 operator>=	374
5.268.3.17 SetFormat	374
5.268.3.18 SetTime	374
5.268.3.19 SetTime	374
5.268.3.20 str	374

CONTENTS

5.269ArcSec::TimeAttribute Class Reference	374
5.269.1 Detailed Description	375
5.269.2 Member Function Documentation	375
5.269.2.1 encode	375
5.269.2.2 equal	375
5.269.2.3 getId	375
5.269.2.4 getType	375
5.270Arc::TimedMutex Class Reference	375
5.271DataStaging::TransferParameters Class Reference	376
5.271.1 Detailed Description	376
5.271.2 Field Documentation	376
5.271.2.1 max_inactivity_time	376
5.271.2.2 min_average_bandwidth	376
5.271.2.3 min_current_bandwidth	376
5.272DataStaging::TransferShares Class Reference	377
5.272.1 Detailed Description	377
5.272.2 Member Enumeration Documentation	378
5.272.2.1 ShareType	378
5.272.3 Member Function Documentation	378
5.272.3.1 calculate_shares	378
5.272.3.2 decrease_number_of_slots	378
5.272.3.3 decrease_transfer_share	378
5.272.3.4 increase_transfer_share	378
5.273Arc::URL Class Reference	379
5.273.1 Detailed Description	381
5.273.2 Member Enumeration Documentation	382
5.273.2.1 Scope	382
5.273.3 Constructor & Destructor Documentation	382
5.273.3.1 URL	382
5.273.3.2 URL	382
5.273.3.3 ~URL	382
5.273.4 Member Function Documentation	382
5.273.4.1 AddHTTPOption	383
5.273.4.2 AddLDAPAttribute	383

5.273.4.3 AddLocation	383
5.273.4.4 AddMetaDataOption	383
5.273.4.5 AddOption	383
5.273.4.6 AddOption	383
5.273.4.7 BaseDN2Path	383
5.273.4.8 ChangeFullPath	383
5.273.4.9 ChangeHost	384
5.273.4.10 ChangeLDAPFilter	384
5.273.4.11 ChangeLDAPScope	384
5.273.4.12 ChangePath	384
5.273.4.13 ChangePort	384
5.273.4.14 ChangeProtocol	384
5.273.4.15 CommonLocOption	384
5.273.4.16 CommonLocOptions	384
5.273.4.17 ConnectionURL	384
5.273.4.18 FullPath	385
5.273.4.19 FullPathURIEncoded	385
5.273.4.20 fullstr	385
5.273.4.21 Host	385
5.273.4.22 HTTPOption	385
5.273.4.23 HTTPOptions	385
5.273.4.24 lsSecureProtocol	385
5.273.4.25 LDAPAttributes	385
5.273.4.26 LDAPFilter	385
5.273.4.27 LDAPScope	386
5.273.4.28 Locations	386
5.273.4.29 MetaDataOption	386
5.273.4.30 MetaDataOptions	386
5.273.4.31 operator bool	386
5.273.4.32 operator<	386
5.273.4.33 operator==	386
5.273.4.34 Option	386
5.273.4.35 Options	387
5.273.4.36 OptionString	387

5.273.4.37ParseOptions	387
5.273.4.38ParsePath	387
5.273.4.39Passwd	387
5.273.4.40Path	387
5.273.4.41Path2BaseDN	387
5.273.4.42plainstr	387
5.273.4.43Port	387
5.273.4.44Protocol	387
5.273.4.45RemoveHTTPOption	388
5.273.4.46RemoveMetaDataOption	388
5.273.4.47RemoveOption	388
5.273.4.48str	388
5.273.4.49Username	388
5.273.5 Friends And Related Function Documentation	388
5.273.5.1 operator<<	388
5.273.6 Field Documentation	388
5.273.6.1 commonlocoptions	389
5.273.6.2 host	389
5.273.6.3 httpoptions	389
5.273.6.4 ip6addr	389
5.273.6.5 ldapattributes	389
5.273.6.6 ldapfilter	389
5.273.6.7 ldapscope	389
5.273.6.8 locations	389
5.273.6.9 metadataoptions	389
5.273.6.10passwd	389
5.273.6.11path	390
5.273.6.12port	390
5.273.6.13protocol	390
5.273.6.14urloptions	390
5.273.6.15username	390
5.273.6.16valid	390
5.274 Arc::URLLocation Class Reference	390
5.274.1 Detailed Description	391

5.274.2 Constructor & Destructor Documentation	391
5.274.2.1 URLLocation	391
5.274.2.2 URLLocation	391
5.274.2.3 URLLocation	391
5.274.2.4 URLLocation	391
5.274.2.5 URLLocation	391
5.274.2.6 ~URLLocation	392
5.274.3 Member Function Documentation	392
5.274.3.1 fullstr	392
5.274.3.2 Name	392
5.274.3.3 str	392
5.274.4 Field Documentation	392
5.274.4.1 name	392
5.275Arc::User Class Reference	392
5.276Arc::UserConfig Class Reference	392
5.276.1 Detailed Description	394
5.276.2 Constructor & Destructor Documentation	396
5.276.2.1 UserConfig	396
5.276.2.2 UserConfig	396
5.276.2.3 UserConfig	397
5.276.2.4 UserConfig	397
5.276.3 Member Function Documentation	398
5.276.3.1 AddBartender	398
5.276.3.2 AddServices	398
5.276.3.3 AddServices	399
5.276.3.4 ApplyToConfig	399
5.276.3.5 Bartender	400
5.276.3.6 Bartender	400
5.276.3.7 Broker	401
5.276.3.8 Broker	401
5.276.3.9 Broker	401
5.276.3.10CACertificatePath	402
5.276.3.11CACertificatePath	402
5.276.3.12CACertificatesDirectory	403

5.276.3.13CACertificatesDirectory	403
5.276.3.14CertificateLifeTime	403
5.276.3.15CertificateLifeTime	404
5.276.3.16CertificatePath	404
5.276.3.17CertificatePath	405
5.276.3.18ClearRejectedServices	405
5.276.3.19ClearRejectedServices	405
5.276.3.20ClearSelectedServices	406
5.276.3.21ClearSelectedServices	406
5.276.3.22CredentialsFound	406
5.276.3.23GetRejectedServices	407
5.276.3.24GetSelectedServices	407
5.276.3.25 GetUser	407
5.276.3.26dPName	408
5.276.3.27dPName	408
5.276.3.28InitializeCredentials	408
5.276.3.29JobDownloadDirectory	410
5.276.3.30JobDownloadDirectory	410
5.276.3.31JobListFile	410
5.276.3.32JobListFile	411
5.276.3.33KeyPassword	411
5.276.3.34KeyPassword	412
5.276.3.35KeyPath	412
5.276.3.36KeyPath	413
5.276.3.37KeySize	413
5.276.3.38KeySize	413
5.276.3.39LoadConfigurationFile	414
5.276.3.40operator bool	415
5.276.3.41operator!	416
5.276.3.42OverlayFile	416
5.276.3.43OverlayFile	416
5.276.3.44Password	417
5.276.3.45Password	417
5.276.3.46ProxyPath	417

5.276.3.47ProxyPath	418
5.276.3.48SaveToFile	418
5.276.3.49SetUser	418
5.276.3.50SLCS	419
5.276.3.51SLCS	419
5.276.3.52StoreDirectory	419
5.276.3.53StoreDirectory	420
5.276.3.54Timeout	420
5.276.3.55Timeout	421
5.276.3.56UserName	421
5.276.3.57UserName	421
5.276.3.58UtilsDirPath	422
5.276.3.59UtilsDirPath	422
5.276.3.60Verbosity	422
5.276.3.61Verbosity	422
5.276.3.62VOMSESPath	423
5.276.3.63VOMSESPath	423
5.276.4 Field Documentation	423
5.276.4.1 ARCUSERDIRECTORY	424
5.276.4.2 DEFAULT_BROKER	424
5.276.4.3 DEFAULT_TIMEOUT	424
5.276.4.4 DEFAULTCONFIG	424
5.276.4.5 EXAMPLECONFIG	424
5.276.4.6 SYSCONFIG	425
5.276.4.7 SYSCONFIGARCLOC	425
5.277 Arc::UsernameToken Class Reference	425
5.277.1 Detailed Description	425
5.277.2 Member Enumeration Documentation	426
5.277.2.1 PasswordType	426
5.277.3 Constructor & Destructor Documentation	426
5.277.3.1 UsernameToken	426
5.277.3.2 UsernameToken	426
5.277.3.3 UsernameToken	426
5.277.4 Member Function Documentation	426

5.277.4.1 Authenticate	427
5.277.4.2 Authenticate	427
5.277.4.3 operator bool	427
5.277.4.4 Username	427
5.278 Arc::UserSwitch Class Reference	427
5.278.1 Detailed Description	427
5.279 Arc::VOMSACInfo Class Reference	428
5.280 Arc::VOMSTrustList Class Reference	428
5.280.1 Detailed Description	428
5.280.2 Constructor & Destructor Documentation	428
5.280.2.1 VOMSTrustList	428
5.280.2.2 VOMSTrustList	429
5.280.3 Member Function Documentation	429
5.280.3.1 AddChain	429
5.280.3.2 AddChain	429
5.280.3.3 AddRegex	429
5.281 Arc::WSAEndpointReference Class Reference	429
5.281.1 Detailed Description	430
5.281.2 Constructor & Destructor Documentation	430
5.281.2.1 WSAEndpointReference	430
5.281.2.2 WSAEndpointReference	430
5.281.2.3 WSAEndpointReference	430
5.281.2.4 WSAEndpointReference	430
5.281.2.5 ~WSAEndpointReference	431
5.281.3 Member Function Documentation	431
5.281.3.1 Address	431
5.281.3.2 Address	431
5.281.3.3 MetaData	431
5.281.3.4 operator XMLNode	431
5.281.3.5 operator=	431
5.281.3.6 ReferenceParameters	431
5.282 Arc::WSAHeader Class Reference	431
5.282.1 Detailed Description	432
5.282.2 Constructor & Destructor Documentation	432

5.282.2.1 WSAHeader	432
5.282.2.2 WSAHeader	433
5.282.3 Member Function Documentation	433
5.282.3.1 Action	433
5.282.3.2 Action	433
5.282.3.3 Check	433
5.282.3.4 FaultTo	433
5.282.3.5 From	433
5.282.3.6 MessageID	433
5.282.3.7 MessageID	433
5.282.3.8 NewReferenceParameter	433
5.282.3.9 operator XMLNode	434
5.282.3.10 ReferenceParameter	434
5.282.3.11 ReferenceParameter	434
5.282.3.12 RelatesTo	434
5.282.3.13 RelatesTo	434
5.282.3.14 RelationshipType	434
5.282.3.15 RelationshipType	434
5.282.3.16 ReplyTo	434
5.282.3.17 To	434
5.282.3.18 To	434
5.282.4 Field Documentation	435
5.282.4.1 header_allocated_	435
5.283 Arc::WSRF Class Reference	435
5.283.1 Detailed Description	436
5.283.2 Constructor & Destructor Documentation	436
5.283.2.1 WSRF	436
5.283.2.2 WSRF	436
5.283.3 Member Function Documentation	436
5.283.3.1 operator bool	436
5.283.3.2 set_namespaces	436
5.283.3.3 SOAP	436
5.283.4 Field Documentation	437
5.283.4.1 allocated_	437

5.283.4.2 valid_	437
5.284 Arc::WSRFBaseFault Class Reference	437
5.284.1 Detailed Description	437
5.284.2 Constructor & Destructor Documentation	438
5.284.2.1 WSRFBaseFault	438
5.284.2.2 WSRFBaseFault	438
5.284.3 Member Function Documentation	438
5.284.3.1 set_namespaces	438
5.285 Arc::WSRFResourceUnavailableFault Class Reference	438
5.286 Arc::WSRFResourceUnknownFault Class Reference	438
5.287 Arc::WSRP Class Reference	439
5.287.1 Detailed Description	440
5.287.2 Constructor & Destructor Documentation	441
5.287.2.1 WSRP	441
5.287.2.2 WSRP	441
5.287.3 Member Function Documentation	441
5.287.3.1 set_namespaces	441
5.288 Arc::WSRPDeleteResourceProperties Class Reference	441
5.289 Arc::WSRPDeleteResourcePropertiesRequest Class Reference	441
5.290 Arc::WSRPDeleteResourcePropertiesRequestFailedFault Class Reference	442
5.291 Arc::WSRPDeleteResourcePropertiesResponse Class Reference	442
5.292 Arc::WSRPFault Class Reference	443
5.292.1 Detailed Description	443
5.292.2 Constructor & Destructor Documentation	443
5.292.2.1 WSRPFault	443
5.292.2.2 WSRPFault	443
5.293 Arc::WSRPGetMultipleResourcePropertiesRequest Class Reference	444
5.294 Arc::WSRPGetMultipleResourcePropertiesResponse Class Reference	444
5.295 Arc::WSRPGetPropertyDocumentRequest Class Reference	444
5.296 Arc::WSRPGetPropertyDocumentResponse Class Reference	445
5.297 Arc::WSRPGetPropertyRequest Class Reference	445
5.298 Arc::WSRPGetPropertyResponse Class Reference	446
5.299 Arc::WSRPIinsertResourceProperties Class Reference	446

5.300 Arc::WSRPIinsertResourcePropertiesRequest Class Reference	446
5.301 Arc::WSRPIinsertResourcePropertiesRequestFailedFault Class Reference	447
5.302 Arc::WSRPIinsertResourcePropertiesResponse Class Reference	447
5.303 Arc::WSRPInvalidModificationFault Class Reference	448
5.304 Arc::WSRPInvalidResourcePropertyQNameFault Class Reference	448
5.305 Arc::WSRPMModifyResourceProperties Class Reference	449
5.306 Arc::WSRPPutResourcePropertyDocumentRequest Class Reference . .	449
5.307 Arc::WSRPPutResourcePropertyDocumentResponse Class Reference .	450
5.308 Arc::WSRPQueryResourcePropertiesRequest Class Reference	450
5.309 Arc::WSRPQueryResourcePropertiesResponse Class Reference	450
5.310 Arc::WSRPResourcePropertyChangeFailure Class Reference	451
5.310.1 Detailed Description	451
5.310.2 Constructor & Destructor Documentation	451
5.310.2.1 WSRPResourcePropertyChangeFailure	451
5.310.2.2 WSRPResourcePropertyChangeFailure	451
5.311 Arc::WSRPSetResourcePropertiesRequest Class Reference	452
5.312 Arc::WSRPSetResourcePropertiesResponse Class Reference	452
5.313 Arc::WSRPSetResourcePropertyRequestFailedFault Class Reference .	452
5.314 Arc::WSRPUnableToModifyResourcePropertyFault Class Reference .	453
5.315 Arc::WSRPUnableToPutResourcePropertyDocumentFault Class Refer- ence	453
5.316 Arc::WSRPUpdateResourceProperties Class Reference	454
5.317 Arc::WSRPUpdateResourcePropertiesRequest Class Reference . . .	454
5.318 Arc::WSRPUpdateResourcePropertiesRequestFailedFault Class Refer- ence	455
5.319 Arc::WSRPUpdateResourcePropertiesResponse Class Reference . .	455
5.320 ArcSec::X500NameAttribute Class Reference	456
5.320.1 Member Function Documentation	456
5.320.1.1 encode	456
5.320.1.2 equal	456
5.320.1.3 getId	456
5.320.1.4 getType	456
5.321 Arc::X509Token Class Reference	457
5.321.1 Detailed Description	457

5.321.2 Member Enumeration Documentation	457
5.321.2.1 X509TokenType	457
5.321.3 Constructor & Destructor Documentation	457
5.321.3.1 X509Token	458
5.321.3.2 X509Token	458
5.321.3.3 ~X509Token	458
5.321.4 Member Function Documentation	458
5.321.4.1 Authenticate	458
5.321.4.2 Authenticate	459
5.321.4.3 operator bool	459
5.322Arc::XmlContainer Class Reference	459
5.323Arc::XmlDatabase Class Reference	459
5.324Arc::XmlNode Class Reference	459
5.324.1 Detailed Description	462
5.324.2 Constructor & Destructor Documentation	462
5.324.2.1 XmlNode	462
5.324.2.2 XmlNode	462
5.324.2.3 XmlNode	462
5.324.2.4 XmlNode	462
5.324.2.5 XmlNode	462
5.324.2.6 XmlNode	462
5.324.2.7 XmlNode	463
5.324.2.8 ~XmlNode	463
5.324.3 Member Function Documentation	463
5.324.3.1 Attribute	463
5.324.3.2 Attribute	463
5.324.3.3 Attribute	463
5.324.3.4 AttributesSize	463
5.324.3.5 Child	463
5.324.3.6 Destroy	463
5.324.3.7 Exchange	464
5.324.3.8 FullName	464
5.324.3.9 Get	464
5.324.3.10GetDoc	464

5.324.3.11GetRoot	464
5.324.3.12GetXML	464
5.324.3.13GetXML	464
5.324.3.14Move	465
5.324.3.15Name	465
5.324.3.16Name	465
5.324.3.17Name	465
5.324.3.18Namespace	465
5.324.3.19NamespacePrefix	465
5.324.3.20Namespaces	465
5.324.3.21Namespaces	465
5.324.3.22New	466
5.324.3.23NewAttribute	466
5.324.3.24NewAttribute	466
5.324.3.25NewChild	466
5.324.3.26NewChild	466
5.324.3.27NewChild	466
5.324.3.28NewChild	467
5.324.3.29NewChild	467
5.324.3.30operator bool	467
5.324.3.31operator std::string	467
5.324.3.32operator!	467
5.324.3.33operator!=	467
5.324.3.34operator!=	467
5.324.3.35operator!=	467
5.324.3.36operator!=	467
5.324.3.37operator++	468
5.324.3.38operator--	468
5.324.3.39operator=	468
5.324.3.40operator=	468
5.324.3.41operator=	468
5.324.3.42operator==	468
5.324.3.43operator==	468
5.324.3.44operator==	468

5.324.3.45operator==	468
5.324.3.46operator[]	469
5.324.3.47operator[]	469
5.324.3.48operator[]	469
5.324.3.49Parent	469
5.324.3.50Path	469
5.324.3.51Prefix	469
5.324.3.52ReadFromFile	470
5.324.3.53ReadFromStream	470
5.324.3.54Replace	470
5.324.3.55Same	470
5.324.3.56SaveToFile	470
5.324.3.57SaveToStream	470
5.324.3.58Set	470
5.324.3.59Size	470
5.324.3.60Swap	470
5.324.3.61Validate	471
5.324.3.62XPathLookup	471
5.324.4 Friends And Related Function Documentation	471
5.324.4.1 MatchXMLName	471
5.324.4.2 MatchXMLName	471
5.324.4.3 MatchXMLName	471
5.324.4.4 MatchXMLNamespace	471
5.324.4.5 MatchXMLNamespace	471
5.324.4.6 MatchXMLNamespace	472
5.324.5 Field Documentation	472
5.324.5.1 is_owner_	472
5.324.5.2 is_temporary_	472
5.325Arc::XMLNodeContainer Class Reference	472
5.325.1 Detailed Description	472
5.325.2 Constructor & Destructor Documentation	473
5.325.2.1 XMLNodeContainer	473
5.325.2.2 XMLNodeContainer	473
5.325.3 Member Function Documentation	473

5.325.3.1 Add	473
5.325.3.2 Add	473
5.325.3.3 AddNew	473
5.325.3.4 AddNew	473
5.325.3.5 Nodes	473
5.325.3.6 operator=	473
5.325.3.7 operator[]	474
5.325.3.8 Size	474
5.326 Arc::XMLSecNode Class Reference	474
5.326.1 Detailed Description	474
5.326.2 Constructor & Destructor Documentation	475
5.326.2.1 XMLSecNode	475
5.326.3 Member Function Documentation	475
5.326.3.1 AddSignatureTemplate	475
5.326.3.2 DecryptNode	475
5.326.3.3 EncryptNode	475
5.326.3.4 SignNode	476
5.326.3.5 VerifyNode	476

Chapter 1

Namespace Index

1.1 Namespace List

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

Arc (Arc namespace contains all core ARC classes)	19
ArcCredential	43
DataStaging (DataStaging contains all components for data transfer scheduling and execution)	45

Chapter 2

Data Structure Index

2.1 Class Hierarchy

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

ArcCredential::ACACI	49
ArcCredential::ACATHOLDER	49
ArcCredential::ACATTR	49
ArcCredential::ACATTRIBUTE	49
ArcCredential::ACC	49
ArcCredential::ACCERTS	50
ArcCredential::ACDIGEST	50
ArcCredential::ACFORM	50
ArcCredential::ACFULLATTRIBUTES	50
ArcCredential::ACHOLDER	50
ArcCredential::ACIETFATTR	50
ArcCredential::ACINFO	51
ArcCredential::ACIS	51
ArcCredential::ACSEQ	51
ArcCredential::ACTARGET	51
ArcCredential::ACTARGETS	51
ArcCredential::ACVAL	51
Arc::ApplicationType	56
Arc::ArcLocation	56
ArcSec::ArcPeriod	57
Arc::ArcVersion	58
ArcSec::Attr	58
Arc::AttributeIterator	59
ArcSec::AttributeProxy	62
ArcSec::AttributeValue	63
ArcSec::AnyURIAttribute	55
ArcSec::BooleanAttribute	68
ArcSec::DateAttribute	126
ArcSec::DateTimeAttribute	127

ArcSec::DurationAttribute	151
ArcSec::GenericAttribute	174
ArcSec::PeriodAttribute	286
ArcSec::StringAttribute	355
ArcSec::TimeAttribute	374
ArcSec::X500NameAttribute	456
ArcSec::Attrs	65
ArcSec::AuthzRequest	65
ArcSec::AuthzRequestSection	65
Arc::AutoPointer< T >	66
Arc::Base64	66
Arc::BaseConfig	66
Arc::MCCConfig	241
Arc::ByteArray	73
DataStaging::CacheParameters	73
ArcCredential::cert_verify_context	74
Arc::CertEnvLocker	74
Arc::ChainContext	74
Arc::CheckSum	75
Arc::Adler32Sum	51
Arc::CheckSumAny	77
Arc::CRC32Sum	105
Arc::MD5Sum	244
Arc::ClientHTTPwithSAML2SSO	83
Arc::ClientInterface	83
Arc::ClientTCP	86
Arc::ClientHTTP	82
Arc::ClientSOAP	84
Arc::ClientSOAPwithSAML2SSO	85
Arc::ClientX509Delegation	87
ArcSec::CombiningAlg	89
ArcSec::DenyOverridesCombiningAlg	137
ArcSec::OrderedCombiningAlg	266
ArcSec::PermitOverridesCombiningAlg	287
Arc::ConfusaCertHandler	92
Arc::ConfusaParserUtils	93
Arc::CountedPointer< T >	94
Arc::Counter	95
Arc::IntraProcessCounter	189
Arc::CounterTicket	103
Arc::Credential	107
Arc::CredentialError	117
Arc::CredentialStore	118
Arc::Database	118
Arc::MySQLDatabase	259
DataStaging::DataDeliveryComm	121
DataStaging::DataDeliveryLocalComm	124

DataStaging::DataDeliveryRemoteComm	125
DataStaging::DataDeliveryCommHandler	124
Arc::DBranch	128
Arc::DelegationConsumer	128
Arc::DelegationConsumerSOAP	130
Arc::DelegationContainerSOAP	131
Arc::DelegationProvider	133
Arc::DelegationProviderSOAP	134
Arc::DiskSpaceRequirementType	138
Arc::DItem	139
Arc::DItemString	139
DataStaging::DTR	140
DataStaging::DTRCallback	144
DataStaging::DataDelivery	120
DataStaging::Generator	173
DataStaging::Processor	300
DataStaging::Scheduler	322
DataStaging::DTRErrorStatus	145
DataStaging::DTRLList	146
DataStaging::DTRStatus	149
Arc::EnvLockWrapper	152
ArcSec::EvalResult	154
ArcSec::EvaluationCtx	154
ArcSec::EvaluatorContext	158
ArcSec::EvaluatorLoader	158
Arc::ExecutableType	160
Arc::ExecutionTarget	160
Arc::ExpirationReminder	164
Arc::FileAccess	165
Arc::FileLock	167
Arc::FileType	170
Arc::FinderLoader	170
ArcSec::Function	171
ArcSec::EqualFunction	152
ArcSec::InRangeFunction	188
ArcSec::MatchFunction	234
Arc::GlobusResult	175
Arc::GLUE2	175
Arc::GSSCredential	176
Arc::FileAccess::header_t	177
Arc::HTTPClientInfo	177
Arc::InfoCache	177
Arc::InfoFilter	178
Arc::InfoRegister	180
Arc::InfoRegisterContainer	180
Arc::InfoRegisters	181
Arc::InfoRegistrar	182
Arc::InformationInterface	184

Arc::InfoCacheInterface	178
Arc::InformationContainer	182
Arc::InformationRequest	185
Arc::InformationResponse	187
Arc::InitializeCredentialsType	188
Arc::ISIS_description	193
Arc::IString	194
Arc::JobDescriptionParserLoader::iterator	194
Arc::Job	194
Arc::JobDescription	206
Arc::JobIdentificationType	212
Arc::JobState	212
Arc::JobSupervisor	213
Arc::LoadableModuleDescription	221
Arc::Loader	221
Arc::BrokerLoader	71
Arc::JobControllerLoader	204
Arc::JobDescriptionParserLoader	210
Arc::MCCLoader	242
Arc::SubmitterLoader	357
Arc::TargetRetrieverLoader	367
Arc::LogDestination	223
Arc::LogFile	224
Arc::LogStream	233
Arc::Logger	226
Arc::LoggerContext	230
Arc::LoggerFormat	230
Arc::LogMessage	231
Arc::MCC_Status	238
Arc::MemoryAllocationException	246
Arc::Message	246
Arc::MessageAttributes	249
Arc::MessageAuth	252
Arc::MessageAuthContext	254
Arc::MessageContext	254
Arc::MessageContextElement	255
ArcSec::PDPCConfigContext	283
Arc::MessagePayload	255
Arc::PayloadRawInterface	271
Arc::PayloadRaw	268
Arc::PayloadSOAP	274
Arc::PayloadStreamInterface	278
Arc::PayloadStream	275
Arc::PayloadWSRF	281
Arc::ModuleDesc	256
Arc::ModuleManager	256
Arc::PluginsFactory	294
Arc::ClassLoader	81

Arc::NotificationType	263
Arc::NS	263
Arc::OptionParser	266
passwd	267
Arc::PathIterator	267
Arc::PayloadRawBuf	271
Arc::Period	284
Arc::PlexerEntry	290
Arc::Plugin	291
Arc::Broker	69
Arc::JobController	201
Arc::JobDescriptionParser	210
Arc::MCCInterface	241
Arc::MCC	236
Arc::Plexer	288
Arc::Service	330
Arc::RegisteredService	305
GDS20::GDS20Service	172
Arc::Submitter	356
Arc::TargetRetriever	365
ArcSec::AlgFactory	54
ArcSec::AttributeFactory	58
ArcSec::Evaluator	154
ArcSec::FnFactory	170
ArcSec::PDP	282
ArcSec::Policy	296
ArcSec::Request	307
ArcSec::SecHandler	328
Arc::PluginArgument	292
Arc::BrokerPluginArgument	72
Arc::ClassLoaderPluginArgument	82
Arc::JobControllerPluginArgument	206
Arc::MCCPluginArgument	244
Arc::ServicePluginArgument	332
Arc::SubmitterPluginArgument	359
Arc::TargetRetrieverPluginArgument	368
ArcSec::PDPPluginArgument	283
ArcSec::SecHandlerPluginArgument	329
Arc::PluginDesc	293
Arc::PluginDescriptor	293
ArcSec::PolicyStore::PolicyElement	298
ArcSec::PolicyParser	298
ArcSec::PolicyStore	299
Arc::PrintFBase	300
Arc::PrintF< T0, T1, T2, T3, T4, T5, T6, T7 >	300
ArcCredential::PROXYCERTINFO_st	302
ArcCredential::PROXYPOLICY_st	302
Arc::Query	302

Arc::MySQLQuery	261
Arc::Range< T >	305
Arc::Register_Info_Type	305
Arc::RegularExpression	306
ArcSec::RequestAttribute	309
ArcSec::RequestItem	310
ArcSec::RequestTuple	311
Arc::ResourceSlotType	311
Arc::ResourcesType	311
ArcSec::Response	311
ArcSec::ResponseItem	311
ArcSec::ResponseList	312
Arc::Run	312
Arc::SAML2LoginClient	316
Arc::OAuthConsumer	263
Arc::SAML2SSOHTTPClient	317
Arc::HakaClient	176
Arc::OpenIdpClient	265
Arc::SAMLToken	319
Arc::ScalableTime< T >	322
Arc::ScalableTime< int >	322
Arc::SecAttr	324
Arc::MultiSecAttr	258
Arc::SecAttrFormat	326
Arc::SecAttrValue	327
Arc::CIStringValue	80
ArcSec::Security	330
Arc::SharedMutex	333
Arc::SimpleCondition	333
Arc::SimpleCounter	334
Arc::SOAPMessage	335
Arc::Software	336
Arc::ApplicationEnvironment	56
Arc::SoftwareRequirement	344
ArcSec::Source	352
ArcSec::SourceFile	353
ArcSec::SourceURL	353
DataStaging::DataDeliveryComm::Status	354
Arc::TargetGenerator	359
Arc::ThreadDataItem	368
Arc::ThreadInitializer	370
Arc::ThreadRegistry	370
Arc::Time	371
Arc::TimedMutex	375
DataStaging::TransferParameters	376
DataStaging::TransferShares	377
Arc::URL	379
Arc::URLLocation	390

Arc::User	392
Arc::UserConfig	392
Arc::UsernameToken	425
Arc::UserSwitch	427
Arc::VOMSACInfo	428
Arc::VOMSTrustList	428
Arc::WSAEndpointReference	429
Arc::WSAHeader	431
Arc::WSRF	435
Arc::WSRFBaseFault	437
Arc::WSRFResourceUnavailableFault	438
Arc::WSRFResourceUnknownFault	438
Arc::WSRPFault	443
Arc::WSRPInvalidResourceProperty QNameFault	448
Arc::WSRPRResourcePropertyChangeFailure	451
Arc::WSRPDeleteResourcePropertiesRequestFailedFault	442
Arc::WSRPInsertResourcePropertiesRequestFailedFault	447
Arc::WSRPInvalidModificationFault	448
Arc::WSRPSetResourcePropertyRequestFailedFault	452
Arc::WSRPUnableToModifyResourcePropertyFault	453
Arc::WSRPUnableToPutResourcePropertyDocumentFault	453
Arc::WSRPUpdateResourcePropertiesRequestFailedFault	455
Arc::WSRP	439
Arc::WSRPDeleteResourcePropertiesRequest	441
Arc::WSRPDeleteResourcePropertiesResponse	442
Arc::WSRPGetMultipleResourcePropertiesRequest	444
Arc::WSRPGetMultipleResourcePropertiesResponse	444
Arc::WSRPGetPropertyDocumentRequest	444
Arc::WSRPGetPropertyDocumentResponse	445
Arc::WSRPGetPropertyRequest	445
Arc::WSRPGetPropertyResponse	446
Arc::WSRPInsertResourcePropertiesRequest	446
Arc::WSRPInsertResourcePropertiesResponse	447
Arc::WSRPPutResourcePropertyDocumentRequest	449
Arc::WSRPPutResourcePropertyDocumentResponse	450
Arc::WSRPQueryResourcePropertiesRequest	450
Arc::WSRPQueryResourcePropertiesResponse	450
Arc::WSRPSetResourcePropertiesRequest	452
Arc::WSRPSetResourcePropertiesResponse	452
Arc::WSRPUpdateResourcePropertiesRequest	454
Arc::WSRPUpdateResourcePropertiesResponse	455
Arc::WSRPModifyResourceProperties	449
Arc::WSRPDeleteResourceProperties	441
Arc::WSRPInsertResourceProperties	446
Arc::WSRPUpdateResourceProperties	454
Arc::X509Token	457
Arc::XmlContainer	459
Arc::XmlDatabase	459
Arc::XmlNode	459

Arc::Config	90
Arc::IniConfig	187
Arc::Profile	302
Arc::SecHandlerConfig	328
Arc::ARCPolicyHandlerConfig	57
Arc::DNListHandlerConfig	139
Arc::XMLSecNode	474
ArcSec::SecHandlerConfig	329
Arc::XMLNodeContainer	472

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

ArcCredential::ACACI	49
ArcCredential::ACATTRHOLDER	49
ArcCredential::ACATTR	49
ArcCredential::ACATTRIBUTE	49
ArcCredential::ACC	49
ArcCredential::ACCERTS	50
ArcCredential::ACDIGEST	50
ArcCredential::ACFORM	50
ArcCredential::ACFULLATTRIBUTES	50
ArcCredential::ACHOLDER	50
ArcCredential::ACIETFATTR	50
ArcCredential::ACINFO	51
ArcCredential::ACIS	51
ArcCredential::ACSEQ	51
ArcCredential::ACTARGET	51
ArcCredential::ACTARGETS	51
ArcCredential::ACVAL	51
Arc::Adler32Sum (Implementation of Adler32 checksum)	51
ArcSec::AlgFactory (Interface for algorithm factory class)	54
ArcSec::AnyURIAttribute	55
Arc::ApplicationEnvironment (ApplicationEnvironment)	56
Arc::ApplicationType	56
Arc::ArcLocation (Determines ARC installation location)	56
ArcSec::ArcPeriod	57
Arc::ARCPolicyHandlerConfig	57
Arc::ArcVersion (Determines ARC HED libraries version)	58
ArcSec::Attr (Attr contains a tuple of attribute type and value)	58
ArcSec::AttributeFactory	58

Arc::AttributeIterator (A const iterator class for accessing multiple values of an attribute)	59
ArcSec::AttributeProxy (Interface for creating the AttributeValue object, it will be used by AttributeFactory)	62
ArcSec::AttributeValue (Interface for containing different type of <Attribute> node for both policy and request)	63
ArcSec::Attrs (Attrs is a container for one or more Attr)	65
ArcSec::AuthzRequest	65
ArcSec::AuthzRequestSection	65
Arc::AutoPointer< T > (Wrapper for pointer with automatic destruction)	66
Arc::Base64	66
Arc::BaseConfig	66
ArcSec::BooleanAttribute	68
Arc::Broker	69
Arc::BrokerLoader	71
Arc::BrokerPluginArgument	72
Arc::ByteArray	73
DataStaging::CacheParameters (The configured cache directories)	73
ArcCredential::cert_verify_context	74
Arc::CertEnvLocker	74
Arc::ChainContext (Interface to chain specific functionality)	74
Arc::CheckSum (Interface for checksum manipulations)	75
Arc::CheckSumAny (Wrapper for CheckSum class)	77
Arc::CIStringValue (This class implements case insensitive strings as security attributes)	80
Arc::ClassLoader	81
Arc::ClassLoaderPluginArgument	82
Arc::ClientHTTP (Class for setting up a MCC chain for HTTP communication) .	82
Arc::ClientHTTPwithSAML2SSO	83
Arc::ClientInterface (Utility base class for MCC)	83
Arc::ClientSOAP	84
Arc::ClientSOAPwithSAML2SSO	85
Arc::ClientTCP (Class for setting up a MCC chain for TCP communication) .	86
Arc::ClientX509Delegation	87
ArcSec::CombiningAlg (Interface for combining algorithm)	89
Arc::Config (Configuration element - represents (sub)tree of ARC configuration)	90
Arc::ConfusaCertHandler	92
Arc::ConfusaParserUtils	93
Arc::CountedPointer< T > (Wrapper for pointer with automatic destruction and mutiple references)	94
Arc::Counter (A class defining a common interface for counters)	95
Arc::CounterTicket (A class for "tickets" that correspond to counter reservations)	103
Arc::CRC32Sum (Implementation of CRC32 checksum)	105
Arc::Credential	107
Arc::CredentialError	117
Arc::CredentialStore	118
Arc::Database (Interface for calling database client library)	118
DataStaging::DataDelivery (DataDelivery transfers data between specified physical locations)	120

DataStaging::DataDeliveryComm (This class provides an abstract interface for the Delivery layer)	121
DataStaging::DataDeliveryCommHandler (Singleton class handling all active DataDeliveryComm objects)	124
DataStaging::DataDeliveryLocalComm (This class starts, monitors and controls a local Delivery process)	124
DataStaging::DataDeliveryRemoteComm (This class contacts a remote service to make a Delivery request)	125
ArcSec::DateAttribute	126
ArcSec::DateTimeAttribute	127
Arc::DBranch	128
Arc::DelegationConsumer	128
Arc::DelegationConsumerSOAP	130
Arc::DelegationContainerSOAP	131
Arc::DelegationProvider	133
Arc::DelegationProviderSOAP	134
ArcSec::DenyOverridesCombiningAlg (Implement the "Deny-Overrides" algorithm)	137
Arc::DiskSpaceRequirementType	138
Arc::DItem	139
Arc::DItemString	139
Arc::DNListHandlerConfig	139
DataStaging::DTR (Data Transfer Request)	140
DataStaging::DTRCallback (The base class from which all callback-enabled classes should be derived)	144
DataStaging::DTRErrorStatus (A class to represent error states reported by various components)	145
DataStaging::DTRLList (Global list of all active DTRs in the system)	146
DataStaging::DTRStatus (Class representing the status of a DTR)	149
ArcSec::DurationAttribute	151
Arc::EnvLockWrapper	152
ArcSec::EqualFunction (Evaluate whether the two values are equal)	152
ArcSec::EvalResult (Struct to record the xml node and effect, which will be used by Evaluator to get the information about which rule/policy(in xmlnode) is satisfied)	154
ArcSec::EvaluationCtx (EvaluationCtx, in charge of storing some context information for)	154
ArcSec::Evaluator (Interface for policy evaluation. Execute the policy evaluation, based on the request and policy)	154
ArcSec::EvaluatorContext (Context for evaluator. It includes the factories which will be used to create related objects)	158
ArcSec::EvaluatorLoader (EvaluatorLoader is implemented as a helper class for loading different Evaluator objects, like ArcEvaluator)	158
Arc::ExecutableType	160
Arc::ExecutionTarget (ExecutionTarget)	160
Arc::ExpirationReminder (A class intended for internal use within counters)	164
Arc::FileAccess (Defines interface for accessing filesystems)	165
Arc::FileLock (A general file locking class)	167
Arc::FileType	170
Arc::FinderLoader	170

ArcSec::FnFactory (Interface for function factory class)	170
ArcSec::Function (Interface for function, which is in charge of evaluating two AttributeValue)	171
GDS20::GDS20Service	172
DataStaging::Generator (Simple Generator implementation)	173
ArcSec::GenericAttribute	174
Arc::GlobusResult	175
Arc::GLUE2 (GLUE2 parser)	175
Arc::GSSCredential	176
Arc::HakaClient	176
Arc::FileAccess::header_t	177
Arc::HTTPClientInfo	177
Arc::InfoCache (Stores XML document in filesystem split into parts)	177
Arc::InfoCacheInterface	178
Arc::InfoFilter (Filters information document according to identity of requestor) .	178
Arc::InfoRegister (Registration to ISIS interface)	180
Arc::InfoRegisterContainer	180
Arc::InfoRegisters (Handling multiple registrations to ISISes)	181
Arc::InfoRegistrar (Registration process associated with particular ISIS) . .	182
Arc::InformationContainer (Information System document container and pro- cessor)	182
Arc::InformationInterface (Information System message processor)	184
Arc::InformationRequest (Request for information in InfoSystem)	185
Arc::InformationResponse (Informational response from InfoSystem)	187
Arc::IniConfig	187
Arc::initializeCredentialsType	188
ArcSec::InRangeFunction	188
Arc::IntraProcessCounter (A class for counters used by threads within a single process)	189
Arc::ISIS_description	193
Arc::IString	194
Arc::JobDescriptionParserLoader::iterator	194
Arc::Job (Job)	194
Arc::JobController (Base class for the JobControllers)	201
Arc::JobControllerLoader	204
Arc::JobControllerPluginArgument	206
Arc::JobDescription	206
Arc::JobDescriptionParser (Abstract class for the different parsers)	210
Arc::JobDescriptionParserLoader	210
Arc::JobIdentificationType	212
Arc::JobState	212
Arc::JobSupervisor (% JobSupervisor class)	213
Arc::LoadableModuleDescription	221
Arc::Loader (Plugins loader)	221
Arc::LogDestination (A base class for log destinations)	223
Arc::LogFile (A class for logging to files)	224
Arc::Logger (A logger class)	226
Arc::LoggerContext (Container for logger configuration)	230
Arc::LoggerFormat	230
Arc::LogMessage (A class for log messages)	231

Arc::LogStream (A class for logging to ostreams)	233
ArcSec::MatchFunction (Evaluate whether arg1 (value in regular expression) matched arg0 (lable in regular expression))	234
Arc::MCC (Message Chain Component - base class for every MCC plugin)	236
Arc::MCC_Status (A class for communication of MCC processing results)	238
Arc::MCCConfig	241
Arc::MCCIInterface (Interface for communication between MCC, Service and Plexer objects)	241
Arc::MCCLoader (Creator of Message Component Chains (MCC))	242
Arc::MCCPluginArgument	244
Arc::MD5Sum (Implementation of MD5 checksum)	244
Arc::MemoryAllocationException	246
Arc::Message (Object being passed through chain of MCCs)	246
Arc::MessageAttributes (A class for storage of attribute values)	249
Arc::MessageAuth (Contains authenticity information, authorization tokens and decisions)	252
Arc::MessageAuthContext (Handler for content of message auth* context)	254
Arc::MessageContext (Handler for content of message context)	254
Arc::MessageContextElement (Top class for elements contained in message context)	255
Arc::MessagePayload (Base class for content of message passed through chain)	255
Arc::ModuleDesc (Description of loadable module)	256
Arc::ModuleManager (Manager of shared libraries)	256
Arc::MultiSecAttr (Container of multiple SecAttr attributes)	258
Arc::MySQLDatabase	259
Arc::MySQLQuery	261
Arc::NotificationType	263
Arc::NS	263
Arc::OAuthConsumer	263
Arc::OpenLdpClient	265
Arc::OptionParser	266
ArcSec::OrderedCombiningAlg	266
passwd	267
Arc::PathIterator (Class to iterate through elements of path)	267
Arc::PayloadRaw (Raw byte multi-buffer)	268
Arc::PayloadRawBuf	271
Arc::PayloadRawInterface (Random Access Payload for Message objects)	271
Arc::PayloadSOAP (Payload of Message with SOAP content)	274
Arc::PayloadStream (POSIX handle as Payload)	275
Arc::PayloadStreamInterface (Stream-like Payload for Message object)	278
Arc::PayloadWSRF (This class combines MessagePayload with WSRF)	281
ArcSec::PDP (Base class for Policy Decision Point plugins)	282
ArcSec::PDPConfigContext	283
ArcSec::PDPPPluginArgument	283
Arc::Period	284
ArcSec::PeriodAttribute	286
ArcSec::PermitOverridesCombiningAlg (Implement the "Permit-Overrides" algorithm)	287
Arc::Plexer (The Plexer class, used for routing messages to services)	288

Arc::PlexerEntry (A pair of label (regex) and pointer to MCC)	290
Arc::Plugin (Base class for loadable ARC components)	291
Arc::PluginArgument (Base class for passing arguments to loadable ARC components)	292
Arc::PluginDesc (Description of plugin)	293
Arc::PluginDescriptor (Description of ARC loadable component)	293
Arc::PluginsFactory (Generic ARC plugins loader)	294
ArcSec::Policy (Interface for containing and processing different types of policy)	296
ArcSec::PolicyStore::PolicyElement	298
ArcSec::PolicyParser (A interface which will isolate the policy object from actual policy storage (files, urls, database))	298
ArcSec::PolicyStore (Storage place for policy objects)	299
Arc::PrintF< T0, T1, T2, T3, T4, T5, T6, T7 >	300
Arc::PrintFBase	300
DataStaging::Processor (The Processor performs pre- and post-transfer operations)	300
Arc::Profile	302
ArcCredential::PROXYCERTINFO_st	302
ArcCredential::PROXYPOLICY_st	302
Arc::Query	302
Arc::Range< T >	305
Arc::Register_Info_Type	305
Arc::RegisteredService (RegisteredService - extension of Service performing self-registration)	305
Arc::RegularExpression (A regular expression class)	306
ArcSec::Request (Base class/Interface for request, includes a container for RequestItems and some operations)	307
ArcSec::RequestAttribute (Wrapper which includes AttributeValue object which is generated according to date type of one specific node in Request.xml)	309
ArcSec::RequestItem (Interface for request item container, <subjects, actions, objects, ctxs> tuple)	310
ArcSec::RequestTuple	311
Arc::ResourceSlotType	311
Arc::ResourcesType	311
ArcSec::Response (Container for the evaluation results)	311
ArcSec::ResponseItem (Evaluation result concerning one RequestTuple)	311
ArcSec::ResponseList	312
Arc::Run	312
Arc::SAML2LoginClient	316
Arc::SAML2SSOHTTPClient	317
Arc::SAMLToken (Class for manipulating SAML Token Profile)	319
Arc::ScalableTime< T >	322
Arc::ScalableTime< int >	322
DataStaging::Scheduler (The Scheduler is the control centre of the data staging framework)	322
Arc::SecAttr (This is an abstract interface to a security attribute)	324
Arc::SecAttrFormat (Export/import format)	326
Arc::SecAttrValue (This is an abstract interface to a security attribute)	327
ArcSec::SecHandler (Base class for simple security handling plugins)	328

Arc::SecHandlerConfig	328
ArcSec::SecHandlerConfig	329
ArcSec::SecHandlerPluginArgument	329
ArcSec::Security (Common stuff used by security related classes)	330
Arc::Service (Service - last component in a Message Chain)	330
Arc::ServicePluginArgument	332
Arc::SharedMutex	333
Arc::SimpleCondition (Simple triggered condition)	333
Arc::SimpleCounter	334
Arc::SOAPMessage (Message restricted to SOAP payload)	335
Arc::Software (Used to represent software (names and version) and comparison)	336
Arc::SoftwareRequirement (Class used to express and resolve version requirements on software)	344
ArcSec::Source (Acquires and parses XML document from specified source) .	352
ArcSec::SourceFile (Convenience class for obtaining XML document from file)	353
ArcSec::SourceURL (Convenience class for obtaining XML document from remote URL)	353
DataStaging::DataDeliveryComm::Status (Plain C struct to pass information from executing process back to main thread)	354
ArcSec::StringAttribute	355
Arc::Submitter (Base class for the Submitters)	356
Arc::SubmitterLoader	357
Arc::SubmitterPluginArgument	359
Arc::TargetGenerator (Target generation class)	359
Arc::TargetRetriever (TargetRetriever base class)	365
Arc::TargetRetrieverLoader	367
Arc::TargetRetrieverPluginArgument	368
Arc::ThreadDataItem (Base class for per-thread object)	368
Arc::ThreadInitializer	370
Arc::ThreadRegistry	370
Arc::Time (A class for storing and manipulating times)	371
ArcSec::TimeAttribute	374
Arc::TimedMutex	375
DataStaging::TransferParameters	376
DataStaging::TransferShares (TransferShares is used to implement fair-sharing and priorities)	377
Arc::URL (Class to hold general URLs)	379
Arc::URLLocation (Class to hold a resolved URL location)	390
Arc::User	392
Arc::UserConfig (User configuration class)	392
Arc::UsernameToken (Interface for manipulation of WS-Security according to Username Token Profile)	425
Arc::UserSwitch	427
Arc::VOMSACInfo	428
Arc::VOMSTrustList	428
Arc::WSAEndpointReference (Interface for manipulation of WS-Addressing Endpoint Reference)	429
Arc::WSAHeader (Interface for manipulation WS-Addressing information in SOAP header)	431

Arc::WSRF (Base class for every WSRF message)	435
Arc::WSRFBaseFault (Base class for WSRF fault messages)	437
Arc::WSRFResourceUnavailableFault	438
Arc::WSRFResourceUnknownFault	438
Arc::WSRP (Base class for WS-ResourceProperties structures)	439
Arc::WSRPDeleteResourceProperties	441
Arc::WSRPDeleteResourcePropertiesRequest	441
Arc::WSRPDeleteResourcePropertiesRequestFailedFault	442
Arc::WSRPDeleteResourcePropertiesResponse	442
Arc::WSRPFault (Base class for WS-ResourceProperties faults)	443
Arc::WSRPGetMultipleResourcePropertiesRequest	444
Arc::WSRPGetMultipleResourcePropertiesResponse	444
Arc::WSRPGetResourcePropertyDocumentRequest	444
Arc::WSRPGetResourcePropertyDocumentResponse	445
Arc::WSRPGetPropertyRequest	445
Arc::WSRPGetPropertyResponse	446
Arc::WSRPIinsertResourceProperties	446
Arc::WSRPIinsertResourcePropertiesRequest	446
Arc::WSRPIinsertResourcePropertiesRequestFailedFault	447
Arc::WSRPIinsertResourcePropertiesResponse	447
Arc::WSRPIinvalidModificationFault	448
Arc::WSRPIinvalidResourcePropertyQNameFault	448
Arc::WSRPMModifyResourceProperties	449
Arc::WSRPPutResourcePropertyDocumentRequest	449
Arc::WSRPPutResourcePropertyDocumentResponse	450
Arc::WSRPQueryResourcePropertiesRequest	450
Arc::WSRPQueryResourcePropertiesResponse	450
Arc::WSRPResourcePropertyChangeFailure	451
Arc::WSRPSSetResourcePropertiesRequest	452
Arc::WSRPSSetResourcePropertiesResponse	452
Arc::WSRPSSetResourcePropertyRequestFailedFault	452
Arc::WSRPUnableToModifyResourcePropertyFault	453
Arc::WSRPUnableToPutResourcePropertyDocumentFault	453
Arc::WSRPUpdateResourceProperties	454
Arc::WSRPUpdateResourcePropertiesRequest	454
Arc::WSRPUpdateResourcePropertiesRequestFailedFault	455
Arc::WSRPUpdateResourcePropertiesResponse	455
ArcSec::X500NameAttribute	456
Arc::X509Token (Class for manipulating X.509 Token Profile)	457
Arc::XmlContainer	459
Arc::XmlDatabase	459
Arc::XMLNode (Wrapper for LibXML library Tree interface)	459
Arc::XMLNodeContainer	472
Arc::XMLSecNode (Extends XMLNode class to support XML security operation)	474

Chapter 4

Namespace Documentation

4.1 Arc Namespace Reference

[Arc](#) namespace contains all core ARC classes.

Data Structures

- class [Broker](#)
- class [BrokerLoader](#)
- class [BrokerPluginArgument](#)
- class [ClientInterface](#)

Utility base class for MCC.
- class [ClientTCP](#)

Class for setting up a MCC chain for TCP communication.
- struct [HTTPClientInfo](#)
- class [ClientHTTP](#)

Class for setting up a MCC chain for HTTP communication.

 - class [ClientSOAP](#)
 - class [SecHandlerConfig](#)
 - class [DNListHandlerConfig](#)
 - class [ARCPolicyHandlerConfig](#)
 - class [ClientHTTPwithSAML2SSO](#)
 - class [ClientSOAPwithSAML2SSO](#)
 - class [ClientX509Delegation](#)
 - class [ConfusaCertHandler](#)
 - class [ConfusaParserUtils](#)
 - class [HakaClient](#)
 - class [OpenIdpClient](#)
 - class [OAuthConsumer](#)
 - class [SAML2LoginClient](#)
 - class [SAML2SSOHTTPClient](#)

- class [ApplicationEnvironment](#)
ApplicationEnvironment.
- class [ExecutionTarget](#)
ExecutionTarget.
- class [GLUE2](#)
GLUE2 parser.
- class [Job](#)
Job.
- class [JobController](#)
Base class for the JobControllers.
- class [JobControllerLoader](#)
- class [JobControllerPluginArgument](#)
- class [Range](#)
- class [ScalableTime](#)
- class [ScalableTime< int >](#)
- class [JobIdentificationType](#)
- class [ExecutableType](#)
- class [NotificationType](#)
- class [ApplicationType](#)
- class [ResourceSlotType](#)
- class [DiskSpaceRequirementType](#)
- class [ResourcesType](#)
- class [FileType](#)
- class [JobDescription](#)
- class [JobDescriptionParser](#)
Abstract class for the different parsers.
- class [JobDescriptionParserLoader](#)
- class [JobState](#)
- class [JobSupervisor](#)
% JobSupervisor class
- class [Software](#)
Used to represent software (names and version) and comparison.
- class [SoftwareRequirement](#)
Class used to express and resolve version requirements on software.
- class [Submitter](#)
Base class for the Submitters.
- class [SubmitterLoader](#)
- class [SubmitterPluginArgument](#)
- class [TargetGenerator](#)
Target generation class
- class [TargetRetriever](#)
TargetRetriever base class
- class [TargetRetrieverLoader](#)
- class [TargetRetrieverPluginArgument](#)

- class [Config](#)
Configuration element - represents (sub)tree of ARC configuration.
- class [BaseConfig](#)
- class [ArcLocation](#)
Determines ARC installation location.
- class [RegularExpression](#)
A regular expression class.
- class [ArcVersion](#)
Determines ARC HED libraries version.
- class [Base64](#)
- class [MemoryAllocationException](#)
- class [ByteArray](#)
- class [CheckSum](#)
Interface for checksum manipulations.
- class [CRC32Sum](#)
Implementation of CRC32 checksum.
- class [MD5Sum](#)
Implementation of MD5 checksum.
- class [Adler32Sum](#)
Implementation of Adler32 checksum.
- class [CheckSumAny](#)
Wrapper for [CheckSum](#) class.
- class [Counter](#)
A class defining a common interface for counters.
- class [CounterTicket](#)
A class for "tickets" that correspond to counter reservations.
- class [ExpirationReminder](#)
A class intended for internal use within counters.
- class [Period](#)
- class [Time](#)
A class for storing and manipulating times.
- class [Database](#)
Interface for calling database client library.
- class [Query](#)
- class [DItem](#)
- class [DBranch](#)
- class [DItemString](#)
- class [FileAccess](#)
Defines interface for accessing filesystems.
- class [FileLock](#)
A general file locking class.
- class [IniConfig](#)
- class [IntraProcessCounter](#)
A class for counters used by threads within a single process.

- class [PrintFBase](#)
- class [PrintF](#)
- class [IString](#)
- struct [LoggerFormat](#)
- class [LogMessage](#)
 - A class for log messages.*
- class [LogDestination](#)
 - A base class for log destinations.*
- class [LogStream](#)
 - A class for logging to ostreams.*
- class [LogFile](#)
 - A class for logging to files.*
- class [LoggerContext](#)
 - Container for logger configuration.*
- class [Logger](#)
 - A logger class.*
- class [MySQLDatabase](#)
- class [MySQLQuery](#)
- class [OptionParser](#)
- class [Profile](#)
- class [Run](#)
- class [ThreadDataItem](#)
 - Base class for per-thread object.*
- class [SimpleCondition](#)
 - Simple triggered condition.*
- class [SimpleCounter](#)
- class [TimedMutex](#)
- class [SharedMutex](#)
- class [ThreadRegistry](#)
- class [ThreadInitializer](#)
- class [URL](#)
 - Class to hold general URLs.*
- class [URLLocation](#)
 - Class to hold a resolved URL location.*
- class [PathIterator](#)
 - Class to iterate through elements of path.*
- class [User](#)
- class [UserSwitch](#)
- class [initializeCredentialsType](#)
- class [UserConfig](#)
 - User configuration class*
- class [CertEnvLocker](#)
- class [EnvLockWrapper](#)
- class [AutoPointer](#)

Wrapper for pointer with automatic destruction.

- class [CountedPointer](#)

Wrapper for pointer with automatic destruction and multiple references.

- class [NS](#)
- class [XMLNode](#)

Wrapper for LibXML library Tree interface.

- class [XMLNodeContainer](#)
- class [CredentialError](#)
- class [Credential](#)
- class [VOMSACInfo](#)
- class [VOMSTrustList](#)
- class [CredentialStore](#)
- class [XmlContainer](#)
- class [XmlDatabase](#)
- class [DelegationConsumer](#)
- class [DelegationProvider](#)
- class [DelegationConsumerSOAP](#)
- class [DelegationProviderSOAP](#)
- class [DelegationContainerSOAP](#)
- class [GlobusResult](#)
- class [GSSCredential](#)
- class [InfoCache](#)

Stores XML document in filesystem split into parts.

- class [InfoCacheInterface](#)
- class [InfoFilter](#)

Filters information document according to identity of requestor.

- class [InfoRegister](#)

Registration to ISIS interface.

- class [InfoRegisters](#)

Handling multiple registrations to ISISes.

- struct [Register_Info_Type](#)
- struct [ISIS_description](#)
- class [InfoRegistrar](#)

Registration process associated with particular ISIS.

- class [InfoRegisterContainer](#)
- class [InformationInterface](#)

Information System message processor.

- class [InformationContainer](#)

Information System document container and processor.

- class [InformationRequest](#)

Request for information in InfoSystem.

- class [InformationResponse](#)

Informational response from InfoSystem.

- class [RegisteredService](#)

- RegisteredService* - extension of [Service](#) performing self-registration.
- class [FinderLoader](#)
 - class [Loader](#)

Plugins loader.
 - class [LoadableModuleDescription](#)
 - class [ModuleManager](#)

Manager of shared libraries.
 - class [Plugin](#)

Base class for loadable ARC components.
 - class [PluginArgument](#)

Base class for passing arguments to loadable ARC components.
 - struct [PluginDescriptor](#)

Description of ARC loadable component.
 - class [PluginDesc](#)

Description of plugin.
 - class [ModuleDesc](#)

Description of loadable module.
 - class [PluginsFactory](#)

Generic ARC plugins loader.
 - class [MCCInterface](#)

Interface for communication between [MCC](#), [Service](#) and [Plexer](#) objects.
 - class [MCC](#)

Message Chain Component - base class for every MCC plugin.
 - class [MCCConfig](#)
 - class [MCCPluginArgument](#)
 - class [MCC_Status](#)

A class for communication of MCC processing results.
 - class [MCCLoader](#)

Creator of Message Component Chains (MCC).
 - class [ChainContext](#)

Interface to chain specific functionality.
 - class [MessagePayload](#)

Base class for content of message passed through chain.
 - class [MessageContextElement](#)

Top class for elements contained in message context.
 - class [MessageContext](#)

Handler for content of message context.
 - class [MessageAuthContext](#)

Handler for content of message auth context.*
 - class [Message](#)

Object being passed through chain of MCCs.
 - class [Attributeliterator](#)

A const iterator class for accessing multiple values of an attribute.

- class [MessageAttributes](#)
A class for storage of attribute values.
- class [MessageAuth](#)
Contains authenticity information, authorization tokens and decisions.
- class [PayloadRawInterface](#)
Random Access Payload for [Message](#) objects.
- struct [PayloadRawBuf](#)
- class [PayloadRaw](#)
Raw byte multi-buffer.
- class [PayloadSOAP](#)
Payload of [Message](#) with SOAP content.
- class [PayloadStreamInterface](#)
Stream-like Payload for [Message](#) object.
- class [PayloadStream](#)
POSIX handle as Payload.
- class [PlexerEntry](#)
A pair of label (regex) and pointer to [MCC](#).
- class [Plexer](#)
The [Plexer](#) class, used for routing messages to services.
- class [CIStrongValue](#)
This class implements case insensitive strings as security attributes.
- class [SecAttrValue](#)
This is an abstract interface to a security attribute.
- class [SecAttrFormat](#)
Export/import format.
- class [SecAttr](#)
This is an abstract interface to a security attribute.
- class [MultiSecAttr](#)
Container of multiple [SecAttr](#) attributes.
- class [Service](#)
Service - last component in a [Message](#) Chain.
- class [ServicePluginArgument](#)
- class [SOAPMessage](#)
Message restricted to SOAP payload.
- class [ClassLoader](#)
- class [ClassLoaderPluginArgument](#)
- class [WSAEndpointReference](#)
Interface for manipulation of WS-Addressing Endpoint Reference.
- class [WSAHeader](#)
Interface for manipulation WS-Addressing information in SOAP header.
- class [SAMLToken](#)
Class for manipulating SAML Token [Profile](#).
- class [UsernameToken](#)

Interface for manipulation of WS-Security according to Username Token Profile.

- class [X509Token](#)
Class for manipulating X.509 Token Profile.
- class [PayloadWSRF](#)
This class combines MessagePayload with WSRF.
- class [WSRP](#)
Base class for WS-ResourceProperties structures.
- class [WSRPFault](#)
Base class for WS-ResourceProperties faults.
- class [WSRPIInvalidResourcePropertyQNameFault](#)
- class [WSRPResourcePropertyChangeFailure](#)
- class [WSRPUnableToPutResourcePropertyDocumentFault](#)
- class [WSRPIInvalidModificationFault](#)
- class [WSRPUnableToModifyResourcePropertyFault](#)
- class [WSRPSetResourcePropertyRequestFailedFault](#)
- class [WSRPInsertResourcePropertiesRequestFailedFault](#)
- class [WSRPUpdateResourcePropertiesRequestFailedFault](#)
- class [WSRPDeleteResourcePropertiesRequestFailedFault](#)
- class [WSRPGetResourcePropertyDocumentRequest](#)
- class [WSRPGetResourcePropertyDocumentResponse](#)
- class [WSRPGetResourcePropertyRequest](#)
- class [WSRPGetResourcePropertyResponse](#)
- class [WSRPGetMultipleResourcePropertiesRequest](#)
- class [WSRPGetMultipleResourcePropertiesResponse](#)
- class [WSRPPutResourcePropertyDocumentRequest](#)
- class [WSRPPutResourcePropertyDocumentResponse](#)
- class [WSRPMModifyResourceProperties](#)
- class [WSRPInsertResourceProperties](#)
- class [WSRPUpdateResourceProperties](#)
- class [WSRPDeleteResourceProperties](#)
- class [WSRPSetResourcePropertiesRequest](#)
- class [WSRPSetResourcePropertiesResponse](#)
- class [WSRPInsertResourcePropertiesRequest](#)
- class [WSRPInsertResourcePropertiesResponse](#)
- class [WSRPUpdateResourcePropertiesRequest](#)
- class [WSRPUpdateResourcePropertiesResponse](#)
- class [WSRPDeleteResourcePropertiesRequest](#)
- class [WSRPDeleteResourcePropertiesResponse](#)
- class [WSRPQueryResourcePropertiesRequest](#)
- class [WSRPQueryResourcePropertiesResponse](#)
- class [WSRF](#)
Base class for every WSRF message.
- class [WSRFBaseFault](#)
Base class for WSRF fault messages.
- class [WSRFResourceUnknownFault](#)
- class [WSRFResourceUnavailableFault](#)
- class [XMLSecNode](#)

Extends XMLNode class to support XML security operation.

Typedefs

- `typedef Plugin *(* get_plugin_instance)(PluginArgument *arg)`
- `typedef std::multimap< std::string, std::string > AttrMap`
- `typedef AttrMap::const_iterator AttrConstIter`
- `typedef AttrMap::iterator AttrIter`

Enumerations

- `enum TimeFormat`
- `enum LogLevel`
- `enum LogFormat`
- `enum escape_type { , escape_octal, escape_hex }`
- `enum StatusKind { , STATUS_OK = 1, GENERIC_ERROR = 2, PARSING_ERROR = 4, PROTOCOL_- RECOGNIZED_ERROR = 8, UNKNOWN_SERVICE_ERROR = 16, BUSY_ERROR = 32, SESSION_CLOSE = 64 }`
- `enum WSAFault { , WSAFaultUnknown, WSAFaultInvalidAddressingHeader }`

Functions

- `std::ostream & operator<< (std::ostream &, const Period &)`
- `std::ostream & operator<< (std::ostream &, const Time &)`
- `std::string TimeStamp (const TimeFormat &=Time::GetFormat())`
- `std::string TimeStamp (Time, const TimeFormat &=Time::GetFormat())`
- `bool FileCopy (const std::string &source_path, const std::string &destination_- path, uid_t uid, gid_t gid)`
- `bool FileCopy (const std::string &source_path, const std::string &destination_- path)`
- `bool FileCopy (const std::string &source_path, int destination_handle)`
- `bool FileCopy (int source_handle, const std::string &destination_path)`
- `bool FileCopy (int source_handle, int destination_handle)`
- `bool FileRead (const std::string &filename, std::list< std::string > &data, uid_t uid=0, gid_t gid=0)`
- `bool FileCreate (const std::string &filename, const std::string &data, uid_t uid=0, gid_t gid=0)`
- `bool FileStat (const std::string &path, struct stat *st, bool follow_symlinks)`
- `bool FileStat (const std::string &path, struct stat *st, uid_t uid, gid_t gid, bool follow_symlinks)`
- `bool FileLink (const std::string &oldpath, const std::string &newpath, bool sym- bolic)`
- `bool FileLink (const std::string &oldpath, const std::string &newpath, uid_t uid, gid_t gid, bool symbolic)`
- `std::string FileReadLink (const std::string &path)`
- `std::string FileReadLink (const std::string &path, uid_t uid, gid_t gid)`

- bool `FileDelete` (const std::string &path)
- bool `FileDelete` (const std::string &path, uid_t uid, gid_t gid)
- bool `DirCreate` (const std::string &path, mode_t mode, bool with_parents=false)
- bool `DirCreate` (const std::string &path, uid_t uid, gid_t gid, mode_t mode, bool with_parents=false)
- bool `DirDelete` (const std::string &path)
- bool `DirDelete` (const std::string &path, uid_t uid, gid_t gid)
- bool `TmpDirCreate` (std::string &path)
- bool `TmpFileCreate` (std::string &filename, const std::string &data, uid_t uid=0, gid_t gid=0)
- void `GUID` (std::string &guid)
- std::string `UUID` (void)
- std::ostream & `operator<<` (std::ostream &os, `LogLevel` level)
- `LogLevel string_to_level` (const std::string &str)
- bool `istring_to_level` (const std::string &llStr, `LogLevel` &ll)
- bool `string_to_level` (const std::string &str, `LogLevel` &ll)
- std::string `level_to_string` (const `LogLevel` &level)
- `LogLevel old_level_to_level` (unsigned int old_level)
- template<typename T >
T `stringto` (const std::string &s)
- template<typename T >
bool `stringto` (const std::string &s, T &t)
- bool `strtoint` (const std::string &s, signed int &t, int base=10)
- bool `strtoint` (const std::string &s, unsigned int &t, int base=10)
- bool `strtoint` (const std::string &s, signed long &t, int base=10)
- bool `strtoint` (const std::string &s, unsigned long &t, int base=10)
- bool `strtoint` (const std::string &s, signed long long &t, int base=10)
- bool `strtoint` (const std::string &s, unsigned long long &t, int base=10)
- template<typename T >
std::string `tostring` (T t, int width=0, int precision=0)
- std::string `inttostr` (signed long long t, int base=10, int width=0)
- std::string `inttostr` (unsigned long long t, int base=10, int width=0)
- std::string `inttostr` (signed int t, int base=10, int width=0)
- std::string `inttostr` (unsigned int t, int base=10, int width=0)
- std::string `inttostr` (signed long t, int base=10, int width=0)
- std::string `inttostr` (unsigned long t, int base=10, int width=0)
- std::string `lower` (const std::string &s)
- std::string `upper` (const std::string &s)
- void `tokenize` (const std::string &str, std::vector< std::string > &tokens, const std::string &delimiters=" ", const std::string &start_quotes="", const std::string &end_quotes "")
- void `tokenize` (const std::string &str, std::list< std::string > &tokens, const std::string &delimiters=" ", const std::string &start_quotes="", const std::string &end_quotes "")
- std::string::size_type `get_token` (std::string &token, const std::string &str, std::string::size_type pos, const std::string &delimiters=" ", const std::string &start_quotes="", const std::string &end_quotes "")

- std::string `trim` (const std::string &str, const char *sep=NULL)
- std::string `strip` (const std::string &str)
- std::string `uri_encode` (const std::string &str, bool encode_slash)
- std::string `uri_unencode` (const std::string &str)
- std::string `convert_to_rdn` (const std::string &dn)
- std::string `escape_chars` (const std::string &str, const std::string &chars, char esc, bool excl, `escape_type` type=escape_char)
- std::string `unescape_chars` (const std::string &str, char esc, `escape_type` type=escape_-char)
- bool `CreateThreadFunction` (void(*func)(void *), void *arg, `SimpleCounter` *count=NULL)
- std::list< `URL` > `ReadURLList` (const `URL` &urllist)
- std::string `GetEnv` (const std::string &var)
- std::string `GetEnv` (const std::string &var, bool &found)
- bool `SetEnv` (const std::string &var, const std::string &value, bool overwrite=true)

- void `UnsetEnv` (const std::string &var)
- void `EnvLockWrap` (bool all=false)
- void `EnvLockUnwrap` (bool all=false)
- void `EnvLockUnwrapComplete` (void)
- std::string `StrError` (int errnum=errno)
- bool `MatchXMLName` (const `XMLNode` &node1, const `XMLNode` &node2)
- bool `MatchXMLName` (const `XMLNode` &node, const char *name)
- bool `MatchXMLName` (const `XMLNode` &node, const std::string &name)
- bool `MatchXMLNamespace` (const `XMLNode` &node1, const `XMLNode` &node2)
- bool `MatchXMLNamespace` (const `XMLNode` &node, const char *uri)
- bool `MatchXMLNamespace` (const `XMLNode` &node, const std::string &uri)
- bool `createVOMSAC` (std::string &codedac, `Credential` &issuer_cred, `Credential` &holder_cred, std::vector< std::string > &fqan, std::vector< std::string > &targets, std::vector< std::string > &attributes, std::string &vuname, std::string &uri, int lifetime)
- bool `addVOMSAC` (`ArcCredential::AC` **&aclist, std::string &acorder, std::string &decodedac)
- bool `parseVOMSAC` (X509 *holder, const std::string &ca_cert_dir, const std::string &ca_cert_file, `VOMSTrustList` &vomscert_trust_dn, std::vector< `VOMSACInfo` > &output, bool verify=true, bool reportall=false)
- bool `parseVOMSAC` (const `Credential` &holder_cred, const std::string &ca_cert_dir, const std::string &ca_cert_file, `VOMSTrustList` &vomscert_trust_dn, std::vector< `VOMSACInfo` > &output, bool verify=true, bool reportall=false)
- char * `VOMSDecode` (const char *data, int size, int *)
- std::string `getCredentialProperty` (const `Arc::Credential` &u, const std::string &property, const std::string &ca_cert_dir=std::string(""), const std::string &ca_cert_file=std::string(""), const std::vector< std::string > &voms_trust_list=std::vector< std::string >())
- bool `OpenSSLInit` (void)
- void `HandleOpenSSLError` (void)
- void `HandleOpenSSLError` (int code)
- std::string `string` (`StatusKind` kind)
- const char * `ContentFromPayload` (const `MessagePayload` &payload)

- void [WSAFaultAssign](#) (SOAPEnvelope &mesage, [WSAFault](#) fid)
- [WSAFault](#) [WSAFaultExtract](#) (SOAPEnvelope &message)
- int [passphrase_callback](#) (char *buf, int size, int rwflag, void *)
- bool [init_xmlsec](#) (void)
- bool [final_xmlsec](#) (void)
- std::string [get_cert_str](#) (const char *certfile)
- xmlSecKey * [get_key_from_keystr](#) (const std::string &value)
- xmlSecKey * [get_key_from_keyfile](#) (const char *keyfile)
- std::string [get_key_from_certfile](#) (const char *certfile)
- xmlSecKey * [get_key_from_certstr](#) (const std::string &value)
- xmlSecKeysMngrPtr [load_key_from_keyfile](#) (xmlSecKeysMngrPtr *keys_manager, const char *keyfile)
- xmlSecKeysMngrPtr [load_key_from_certfile](#) (xmlSecKeysMngrPtr *keys_manager, const char *certfile)
- xmlSecKeysMngrPtr [load_key_from_certstr](#) (xmlSecKeysMngrPtr *keys_manager, const std::string &certstr)
- xmlSecKeysMngrPtr [load_trusted_cert_file](#) (xmlSecKeysMngrPtr *keys_manager, const char *cert_file)
- xmlSecKeysMngrPtr [load_trusted_cert_str](#) (xmlSecKeysMngrPtr *keys_manager, const std::string &cert_str)
- xmlSecKeysMngrPtr [load_trusted_certs](#) (xmlSecKeysMngrPtr *keys_manager, const char *cafile, const char *capath)
- [XMLNode](#) [get_node](#) ([XMLNode](#) &parent, const char *name)

Variables

- const Glib::TimeVal [ETERNAL](#)
- const Glib::TimeVal [HISTORIC](#)
- const size_t [thread_stacksize](#) = (16 * 1024 * 1024)
- [Logger](#) [CredentialLogger](#)
- const char * [plugins_table_name](#)

4.1.1 Detailed Description

[Arc](#) namespace contains all core ARC classes.

4.1.2 Typedef Documentation

4.1.2.1 [typedef AttrMap::const_iterator Arc::AttrConstIter](#)

A typedef of a const_iterator for AttrMap.

This typedef is used as a shorthand for a const_iterator for AttrMap. It is used extensively within the [MessageAttributes](#) class as well as the AttributesIterator class, but is not visible externally.

4.1.2.2 `typedef AttrMap::iterator Arc::AttrIter`

A `typedef` of an (non-const) iterator for `AttrMap`.

This `typedef` is used as a shorthand for a (non-const) iterator for `AttrMap`. It is used in one method within the [MessageAttributes](#) class, but is not visible externally.

4.1.2.3 `typedef std::multimap<std::string, std::string> Arc::AttrMap`

A `typedef` of a multimap for storage of message attributes.

This `typedef` is used as a shorthand for a multimap that uses strings for keys as well as values. It is used within the [MessageAttributes](#) class for internal storage of message attributes, but is not visible externally.

4.1.2.4 `typedef Plugin*(* Arc::get_plugin_instance)(PluginArgument *arg)`

Constructor function of ARC loadable component.

This function is called with plugin-specific argument and should produce and return valid instance of plugin. If plugin can't be produced by any reason (for example because passed argument is not applicable) then NULL is returned. No exceptions should be raised.

4.1.3 Enumeration Type Documentation

4.1.3.1 `enum Arc::escape_type`

Type of escaping or encoding to use.

Enumerator:

- `escape_octal`** place the escape character before the character being escaped
- `escape_hex`** octal encoding of the character hex encoding of the character

4.1.3.2 `enum Arc::LogFormat`

Output formats.

Defines prefix for every message. LongFormat - all information about message is printed ShortFormat - only message level is printed DebugFormat - message time (microsecond precision) and time difference from previous message are printed. This format is mostly meant for profiling. EmptyFormat - only message is printed

4.1.3.3 `enum Arc::LogLevel`

Logging levels.

Logging levels for tagging and filtering log messages. FATAL level designates very severe error events that will presumably lead the application to abort. ERROR level designates error events that might still allow the application to continue running. WARNING level designates potentially harmful situations. INFO level designates informational messages that highlight the progress of the application at coarse-grained level. VERBOSE level designates fine-grained informational events that will give additional information about the application. DEBUG level designates finer-grained informational events which should only be used for debugging purposes.

4.1.3.4 enum Arc::StatusKind

Status kinds (types)

This enum defines a set of possible status kinds.

Enumerator:

STATUS_OK Default status - undefined error.

GENERIC_ERROR No error.

PARSING_ERROR Error does not fit any class.

PROTOCOL_RECOGNIZED_ERROR Error detected while parsing request/response.

UNKNOWN_SERVICE_ERROR [Message](#) does not fit into expected protocol.

BUSY_ERROR There is no destination configured for this message.

SESSION_CLOSE Message can't be processed now.

4.1.3.5 enum Arc::WSAFault

WS-Addressing possible faults.

Enumerator:

WSAFaultUnknown This is not a fault

WSAFaultInvalidAddressingHeader This is not a WS-Addressing fault

4.1.4 Function Documentation

4.1.4.1 bool Arc::addVOMSAC (ArcCredential::AC **& *aclist*, std::string & *acorder*, std::string & *decodedac*)

Add decoded AC string into a list of AC objects

Parameters

<i>aclist</i>	The list of AC objects (output)
<i>acorder</i>	The order of AC objects (output)
<i>decodedac</i>	The AC string that is decoded from the string returned from voms server (input)

4.1.4.2 const char* Arc::ContentFromPayload (const MessagePayload & payload)

Returns pointer to main memory chunk of [Message](#) payload.

If no buffer is present or if payload is not of [PayloadRawInterface](#) type NULL is returned.

4.1.4.3 bool Arc::CreateThreadFunction (void(*)(void *) func, void * arg, SimpleCounter * count = NULL)

This macro behaves like function which makes thread of class' method.

It accepts class instance and full name of method - like class::method. 'method' should not be static member of the class. Result is true if creation of thread succeeded. Specified instance must be valid during whole lifetime of thread. So probably it is safer to destroy 'instance' in 'method' just before exiting. Helper function to create simple thread. It takes care of all peculiarities of Glib::Thread API. As result it runs function 'func' with argument 'arg' in a separate thread. If count parameter not NULL then corresponding object will be incremented before function returns and then decremented then thread finished. Returns true on success.

4.1.4.4 bool Arc::createVOMSAC (std::string & codedac, Credential & issuer_cred, Credential & holder_cred, std::vector< std::string > & fqan, std::vector< std::string > & targets, std::vector< std::string > & attributes, std::string & vname, std::string & uri, int lifetime)

Create AC(Attribute Certificate) with voms specific format.

Parameters

<i>codedac</i>	The coded AC as output of this method
<i>issuer_cred</i>	The issuer credential which is used to sign the AC
<i>holder_cred</i>	The holder credential, the holder certificate is the one which carries AC The rest arguments are the same as the above method

4.1.4.5 bool Arc::DirCreate (const std::string & path, uid_t uid, gid_t gid, mode_t mode, bool with_parents = false)

Create a new directory using the specified uid and gid Specified uid and gid are used for accessing filesystem.

4.1.4.6 bool Arc::DirDelete (const std::string & path, uid_t uid, gid_t gid)

Delete a directory using the specified uid and gid Specified uid and gid are used for accessing filesystem.

4.1.4.7 void Arc::EnvLockUnwrap (bool *all* = false)

End code which is using setenv/getenv. Value of all must be same as in corresponding EnvLockWrap.

4.1.4.8 void Arc::EnvLockWrap (bool *all* = false)

Start code which is using setenv/getenv. Use all=true for setenv and all=false for getenv. Must always have corresponding EnvLockUnwrap.

4.1.4.9 std::string Arc::escape_chars (const std::string & str, const std::string & chars, char esc, bool excl, escape_type type = escape_char)

Escape or encode the given chars in str using the escape character esc. If excl is true then escape all characters not in chars

4.1.4.10 bool Arc::FileCopy (const std::string & source_path, const std::string & destination_path, uid_t uid, gid_t gid)

Utility functions for handling files and directories. Those functions offer possibility to access files and directories under user and group ids different from those of current user. Id switching is done in way safe for multi-threaded application. If any of specified ids is 0 then such id is not switched and current id is used instead. Copy file source_path to file destination_path. Specified uid and gid are used for accessing filesystem.

4.1.4.11 bool Arc::FileCreate (const std::string & filename, const std::string & data, uid_t uid = 0, gid_t gid = 0)

Simple method to create a new file containing given data. Specified uid and gid are used for accessing filesystem. An existing file is overwritten with the new data. Permissions of the created file are determined using the current umask. For more complex file handling or large files, FileOpen() should be used. If protected access is required, [FileLock](#) should be used in addition to FileRead. If uid/gid are zero then no real switch of uid/gid is done.

4.1.4.12 bool Arc::FileDelete (const std::string & path, uid_t uid, gid_t gid)

Deletes file at path using the specified uid and gid. Specified uid and gid are used for accessing filesystem.

4.1.4.13 `bool Arc::FileLink (const std::string & oldpath, const std::string & newpath, uid_t uid, gid_t gid, bool symbolic)`

Make symbolic or hard link of file using the specified uid and gid Specified uid and gid are used for accessing filesystem.

4.1.4.14 `bool Arc::FileRead (const std::string & filename, std::list< std::string > & data, uid_t uid = 0, gid_t gid = 0)`

Simple method to read file content from filename. Specified uid and gid are used for accessing filesystem. The content is split into lines with the new line character removed, and the lines are returned in the data list. If protected access is required, [FileLock](#) should be used in addition to FileRead.

4.1.4.15 `std::string Arc::FileReadLink (const std::string & path, uid_t uid, gid_t gid)`

Returns path at which symbolic link is pointing using the specified uid and gid Specified uid and gid are used for accessing filesystem.

4.1.4.16 `bool Arc::FileStat (const std::string & path, struct stat * st, uid_t uid, gid_t gid, bool follow_symlinks)`

Stat a file using the specified uid and gid and put info into the st struct Specified uid and gid are used for accessing filesystem.

4.1.4.17 `bool Arc::final_xmlsec (void)`

Finalize the xml security library

4.1.4.18 `std::string Arc::get_cert_str (const char * certfile)`

Get certificate in string format from certificate file

4.1.4.19 `std::string Arc::get_key_from_certfile (const char * certfile)`

Get public key in string format from certificate file

4.1.4.20 `xmlSecKey* Arc::get_key_from_certstr (const std::string & value)`

Get public key in xmlSecKey structure from certificate string (the string under "---- BEGIN CERTIFICATE----" and "----END CERTIFICATE----")

4.1.4.21 `xmlSecKey* Arc::get_key_from_keyfile (const char * keyfile)`

Get key in xmlSecKey structure from key file

4.1.4.22 `xmlSecKey* Arc::get_key_from_keystr (const std::string & value)`

Get key in xmlSecKey structure from key in string format

4.1.4.23 `XMLNode Arc::get_node (XMLNode & parent, const char * name)`

Generate a new child [XMLNode](#) with specified name

4.1.4.24 `std::string Arc::getCredentialProperty (const Arc::Credential & u, const std::string & property, const std::string & ca_cert_dir = std::string(""), const std::string & ca_cert_file = std::string(""), const std::vector< std::string > & voms_trust_list = std::vector< std::string >())`

Extract the needed field from the certificate.

Parameters

<i>u</i>	The proxy certificate which includes the voms specific formated AC.
<i>property</i>	The property that caller would get, including: dn, voms:vo, voms:role, voms:group
<i>ca_cert_dir</i>	
<i>ca_cert_file</i>	
<i>voms_- trust_list</i>	the dn chain that is trusted when parsing voms AC

4.1.4.25 `void Arc::GUID (std::string & guid)`

Utilities for generating unique identifiers in the form 12345678-90ab-cdef-1234-567890abcdef.

Generates a unique identifier using information such as IP address, current time etc.

4.1.4.26 `bool Arc::init_xmlsec (void)`

Initialize the xml security library, it should be called before the xml security functionality is used.

4.1.4.27 `std::string Arc::inttostr (signed long long t, int base = 10, int width = 0)`

Convert long long integer to textual representation for specied base. Result is padded with zeroes on left till width.

Referenced by [inttostr\(\)](#).

4.1.4.28 std::string Arc::inttostr (signed int *t*, int *base* = 10, int *width* = 0) [inline]

Convert integer to textual representation for specied base. Result is padded with zeroes on left till width.

References [inttostr\(\)](#).

4.1.4.29 std::string Arc::inttostr (unsigned long long *t*, int *base* = 10, int *width* = 0)

Convert unsigned long long integer to textual representation for specied base. Result is padded with zeroes on left till width.

4.1.4.30 std::string Arc::inttostr (unsigned int *t*, int *base* = 10, int *width* = 0) [inline]

Convert unsigned integer to textual representation for specied base. Result is padded with zeroes on left till width.

References [inttostr\(\)](#).

4.1.4.31 std::string Arc::inttostr (unsigned long *t*, int *base* = 10, int *width* = 0) [inline]

Convert unsigned long integer to textual representation for specied base. Result is padded with zeroes on left till width.

References [inttostr\(\)](#).

4.1.4.32 std::string Arc::inttostr (signed long *t*, int *base* = 10, int *width* = 0) [inline]

Convert long integer to textual representation for specied base. Result is padded with zeroes on left till width.

References [inttostr\(\)](#).

4.1.4.33 bool Arc::istring_to_level (const std::string & *lStr*, LogLevel & *l*)

Case-insensitive parsing of a string to a LogLevel with error response.

The method will try to parse (case-insensitive) the argument string to a corresponding LogLevel. If the method succeeds, true will be returned and the argument *l* will be set to the parsed LogLevel. If the parsing fails false will be returned. The parsing succeeds if *lStr* match (case-insensitively) one of the names of the LogLevel members.

Parameters

<i>lStr</i>	a string which should be parsed to a Arc::LogLevel .
<i>l</i>	a Arc::LogLevel reference which will be set to the matching Arc::LogLevel upon successful parsing.

Returns

true in case of successful parsing, otherwise false.

See also

[LogLevel](#)

4.1.4.34 `xmlSecKeysMngrPtr Arc::load_key_from_certfile (xmlSecKeysMngrPtr * keys_manager,
const char * certfile)`

Load public key from a certificate file into key manager

4.1.4.35 `xmlSecKeysMngrPtr Arc::load_key_from_certstr (xmlSecKeysMngrPtr * keys_manager,
const std::string & certstr)`

Load public key from a certificate string into key manager

4.1.4.36 `xmlSecKeysMngrPtr Arc::load_key_from_keyfile (xmlSecKeysMngrPtr * keys_manager,
const char * keyfile)`

Load private or public key from a key file into key manager

4.1.4.37 `xmlSecKeysMngrPtr Arc::load_trusted_cert_file (xmlSecKeysMngrPtr * keys_manager,
const char * cert_file)`

Load trusted certificate from certificate file into key manager

4.1.4.38 `xmlSecKeysMngrPtr Arc::load_trusted_cert_str (xmlSecKeysMngrPtr * keys_manager,
const std::string & cert_str)`

Load trusted certificate from certificate string into key manager

4.1.4.39 `xmlSecKeysMngrPtr Arc::load_trusted_certs (xmlSecKeysMngrPtr * keys_manager,
const char * cafile, const char * capath)`

Load trusted certificates from a file or directory into key manager

4.1.4.40 `bool Arc::MatchXMLName (const XMLNode & node1, const XMLNode & node2)`

Returns true if underlying XML elements have same names

4.1.4.41 `bool Arc::MatchXMLName (const XMLNode & node, const char * name)`

Returns true if 'name' matches name of 'node'. If name contains prefix it's checked too

4.1.4.42 `bool Arc::MatchXMLName (const XMLNode & node, const std::string & name)`

Returns true if 'name' matches name of 'node'. If name contains prefix it's checked too

4.1.4.43 `bool Arc::MatchXMLNamespace (const XMLNode & node1, const XMLNode & node2)`

Returns true if underlying XML elements belong to same namespaces

4.1.4.44 `bool Arc::MatchXMLNamespace (const XMLNode & node, const char * uri)`

Returns true if 'namespace' matches 'node's namespace.

4.1.4.45 `bool Arc::MatchXMLNamespace (const XMLNode & node, const std::string & uri)`

Returns true if 'namespace' matches 'node's namespace.

4.1.4.46 `bool Arc::OpenSSLInit (void)`

This module contains various convenience utilities for using OpenSSL.

Application may be linked to this module instead of OpenSSL libraries directly. This function initializes OpenSSL library. It may be called multiple times and makes sure everything is done properly and OpenSSL may be used in multi-threaded environment. Because this function makes use of [ArcLocation](#) it is advisable to call it after [ArcLocation::Init\(\)](#).

4.1.4.47 `std::ostream& Arc::operator<< (std::ostream & , const Period &)`

Prints a Period-object to the given ostream -- typically cout.

4.1.4.48 `std::ostream& Arc::operator<< (std::ostream & , const Time &)`

Prints a Time-object to the given ostream -- typically cout.

4.1.4.49 `std::ostream& Arc::operator<< (std::ostream & os, LogLevel level)`

Printing of LogLevel values to streams.

Output operator so that LogLevel values can be printed in a nicer way.

```
4.1.4.50 bool Arc::parseVOMSAC ( X509 * holder, const std::string & ca_cert_dir, const
                                std::string & ca_cert_file, VOMSTrustList & vomscert_trust_dn, std::vector<
                                VOMSACInfo > & output, bool verify = true, bool reportall = false )
```

Parse the certificate, and output the attributes.

Parameters

<i>holder</i>	The proxy certificate which includes the voms specific formated AC.
<i>ca_cert_dir</i>	The trusted certificates which are used to verify the certificate which is used to sign the AC
<i>ca_cert_file</i>	The same as <i>ca_cert_dir</i> except it is a file instead of a directory. Only one of them need to be set
<i>vomsdir</i>	The directory which include *.lsc file for each vo. For instance, a vo called "knowarc.eu" should have file \$prefix/vomsdir/knowarc/voms.knowarc.eu.lsc which contains on the first line the DN of the VOMS server, and on the second line the corresponding CA DN: /O=Grid/O=NorduGrid/OU=KnowARC/CN=voms.knowarc.eu /O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority See more in : https://twiki.cern.ch/twiki/bin/view/LCG/VomsFAQforServiceManagement
<i>output</i>	The parsed attributes (Role and Generic Attribute) . Each attribute is stored in element of a vector as a string. It is up to the consumer to understand the meaning of the attribute. There are two types of attributes stored in VOMS AC: AC_IETFATTR, AC_FULL_ATTRIBUTES. The AC_IETFATTR will be like /Role=Employee/Group=Tester/Capability=NULL The AC_FULL_ATTRIBUTES will be like knowarc:Degree=PhD (qualifier::name=value) In order to make the output attribute values be identical, the voms server information is added as prefix of the original attributes in AC. for AC_FULL_ATTRIBUTES, the vename + hostname is added: /vname=knowarc.eu/hostname=arthur.hep.lu.se:15001/knowarc.eu/coredev:attribute1=1 for AC_IETFATTR, the 'VO' (vename) is added: /VO=knowarc.eu/Group=coredev/Role=NULL/Capability=NULL /VO=knowarc.eu/Group= testers/Role=NULL/Capability=NULL

some other redundant attributes is provided: vename=knowarc.eu/hostname=arthur.hep.lu.se:15001

Parameters

<i>verify</i>	true: Verify the voms certificate is trusted based on the <i>ca_cert_dir/ca_cert_file</i> which specifies the CA certificates, and the <i>vomscert_trust_dn</i> which specifies the trusted DN chain from voms server certificate to CA certificate. false: Not verify, which means the issuer of AC (voms server certificate is supposed to be trusted by default). In this case the parameters ' <i>ca_cert_dir</i> ', ' <i>ca_cert_file</i> ' and ' <i>vomscert_trust_dn</i> ' will not effect, and may be left empty. This case is specifically used by 'arcproxy --info' to list all of the attributes in AC, and not to need to verify if the AC's issuer is trusted.
<i>reportall</i>	If set to true fills output with all attributes including those which failed passing test procedures. Validity of attributes can be checked through status members of output items. Combination of <i>verify=true</i> and <i>reportall=true</i> provides most information.

```
4.1.4.51 bool Arc::parseVOMSAC ( const Credential & holder_cred, const std::string &
    ca_cert_dir, const std::string & ca_cert_file, VOMSTrustList & vomsCertTrust_dn,
    std::vector< VOMSACInfo > & output, bool verify = true, bool reportAll = false )
```

Parse the certificate. Similar to above one, but collects information from all certificates in a chain.

```
4.1.4.52 int Arc::passphrase_callback ( char * buf, int size, int rwflag, void * )
```

callback method for inputting passphrase of key file

```
4.1.4.53 std::string Arc::string ( StatusKind kind )
```

Conversion to string.

Conversion from StatusKind to string.

Parameters

<i>kind</i>	The StatusKind to convert.
-------------	----------------------------

```
4.1.4.54 bool Arc::strtoint ( const std::string & s, unsigned int & t, int base = 10 )
```

Convert string to unsigned integer with specified base. Returns false if any argument is wrong.

```
4.1.4.55 bool Arc::strtoint ( const std::string & s, unsigned long long & t, int base = 10 )
```

Convert string to unsigned long long integer with specified base. Returns false if any argument is wrong.

```
4.1.4.56 bool Arc::strtoint ( const std::string & s, signed int & t, int base = 10 )
```

Convert string to integer with specified base. Returns false if any argument is wrong.

```
4.1.4.57 bool Arc::strtoint ( const std::string & s, unsigned long & t, int base = 10 )
```

Convert string to unsigned long integer with specified base. Returns false if any argument is wrong.

```
4.1.4.58 bool Arc::strtoint ( const std::string & s, signed long long & t, int base = 10 )
```

Convert string to long long integer with specified base. Returns false if any argument is wrong.

4.1.4.59 `bool Arc::strtoint (const std::string & s, signed long & t, int base = 10)`

Convert string to long integer with specified base. Returns false if any argument is wrong.

4.1.4.60 `std::string Arc::TimeStamp (const TimeFormat & = Time::GetFormat ())`

Returns a time-stamp of the current time in some format.

4.1.4.61 `std::string Arc::TimeStamp (Time , const TimeFormat & = Time::GetFormat ())`

Returns a time-stamp of some specified time in some format.

4.1.4.62 `bool Arc::TmpDirCreate (std::string & path)`

Create a temporary directory under the system defined temp location, and return its path.

Uses mkdtemp if available, and a combination of random parameters if not. This latter method is not as safe as mkdtemp.

4.1.4.63 `bool Arc::TmpFileCreate (std::string & filename, const std::string & data, uid_t uid = 0, gid_t gid = 0)`

Simple method to create a temporary file containing given data. Specified uid and gid are used for accessing filesystem. Permissions of the created file are determined using the current umask. If uid/gid are zero then no real switch of uid/gid is done.

4.1.4.64 `std::string Arc::uri_encode (const std::string & str, bool encode_slash)`

This method %-encodes characters in URI str.

Characters which are not unreserved according to RFC 3986 are encoded. If encode_slash is true forward slashes will also be encoded. It is useful to set encode_slash to false when encoding full paths. be encoded

4.1.4.65 `char* Arc::VOMSDecode (const char * data, int size, int * j)`

Decode the data which is encoded by voms server. Since voms code uses some specific coding method (not base64 encoding), we simply copy the method from voms code to here

4.1.4.66 void Arc::WSAFaultAssign (SOAPEnvelope & *mesage*, WSAGlobalFault *fid*)

Makes WS-Addressing fault.

It fills SOAP Fault message with WS-Addressing fault related information.

4.1.4.67 WSAGlobalFault Arc::WSAFaultExtract (SOAPEnvelope & *message*)

Gets WS-addressing fault.

Analyzes SOAP Fault message and returns WS-Addressing fault it represents.

4.1.5 Variable Documentation

4.1.5.1 Logger Arc::CredentialLogger

[Logger](#) to be used by all modules of credentials library

4.1.5.2 const char* Arc::plugins_table_name

Name of symbol referring to table of plugins.

This C null terminated string specifies name of symbol which shared library should export to give an access to an array of [PluginDescriptor](#) elements. The array is terminated by element with all components set to NULL.

4.1.5.3 const size_t Arc::thread_stacksize = (16 * 1024 * 1024)

This module provides convenient helpers for Glibmm interface for thread management.

So far it takes care of automatic initialization of threading environment and creation of simple detached threads. Always use it instead of glibmm/thread.h and keep among first includes. It safe to use it multiple times and to include it both from source files and other include files. Defines size of stack assigned to every new thread.

4.2 ArcCredential Namespace Reference

Data Structures

- struct [cert_verify_context](#)
- struct [PROXYPOLICY_st](#)
- struct [PROXYCERTINFO_st](#)
- struct [ACDIGEST](#)
- struct [ACIS](#)
- struct [ACFORM](#)
- struct [ACACI](#)

- struct [ACHOLDER](#)
- struct [ACVAL](#)
- struct [ACIETFATTR](#)
- struct [ACTARGET](#)
- struct [ACTARGETS](#)
- struct [ACATTR](#)
- struct [ACINFO](#)
- struct [ACC](#)
- struct [ACSEQ](#)
- struct [ACCERTS](#)
- struct [ACATTRIBUTE](#)
- struct [ACATHOLDER](#)
- struct [ACFULLATTRIBUTES](#)

Enumerations

- enum [certType](#) {

CERT_TYPE_EEC, CERT_TYPE_CA, CERT_TYPE_GSI_3_IMPERSONATION_PROXY, CERT_TYPE_GSI_3_INDEPENDENT_PROXY,

CERT_TYPE_GSI_3_LIMITED_PROXY, CERT_TYPE_GSI_3_RESTRICTED_PROXY, CERT_TYPE_GSI_2_PROXY, CERT_TYPE_GSI_2_LIMITED_PROXY,

CERT_TYPE_RFC_IMPERSONATION_PROXY, CERT_TYPE_RFC_INDEPENDENT_PROXY, CERT_TYPE_RFC_LIMITED_PROXY, CERT_TYPE_RFC_RESTRICTED_PROXY,

CERT_TYPE_RFC_ANYLANGUAGE_PROXY }

4.2.1 Detailed Description

Functions and constants for maintaining proxy certificates. The code is derived from globus gsi, voms, and openssl-0.9.8e. The existing code for maintaining proxy certificates in OpenSSL only covers standard proxies and does not cover old Globus proxies, so here the Globus code is introduced.

Borrow the code about Attribute Certificate from VOMS. The [VOMSAttribute.h](#) and [VOMSAttribute.cpp](#) are integration about code written by VOMS project, so here the original license follows.

4.2.2 Enumeration Type Documentation

4.2.2.1 enum ArcCredential::certType

Enumerator:

- CERT_TYPE_EEC*** A end entity certificate
- CERT_TYPE_CA*** A CA certificate

CERT_TYPE_GSI_3_IMPERSONATION_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant impersonation proxy

CERT_TYPE_GSI_3_INDEPENDENT_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant independent proxy

CERT_TYPE_GSI_3_LIMITED_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant limited proxy

CERT_TYPE_GSI_3_RESTRICTED_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant restricted proxy

CERT_TYPE_GSI_2_PROXY A legacy Globus impersonation proxy

CERT_TYPE_GSI_2_LIMITED_PROXY A legacy Globus limited impersonation proxy

CERT_TYPE_RFC_IMPERSONATION_PROXY A X.509 Proxy Certificate Profile RFC compliant impersonation proxy; RFC inheritAll proxy

CERT_TYPE_RFC_INDEPENDENT_PROXY A X.509 Proxy Certificate Profile RFC compliant independent proxy; RFC independent proxy

CERT_TYPE_RFC_LIMITED_PROXY A X.509 Proxy Certificate Profile RFC compliant limited proxy

CERT_TYPE_RFC_RESTRICTED_PROXY A X.509 Proxy Certificate Profile RFC compliant restricted proxy

CERT_TYPE_RFC_ANYLANGUAGE_PROXY RFC anyLanguage proxy

4.3 DataStaging Namespace Reference

DataStaging contains all components for data transfer scheduling and execution.

Data Structures

- class [DataDelivery](#)
DataDelivery transfers data between specified physical locations.
- class [DataDeliveryComm](#)
This class provides an abstract interface for the Delivery layer.
- class [DataDeliveryCommHandler](#)
Singleton class handling all active DataDeliveryComm objects.
- class [DataDeliveryLocalComm](#)
This class starts, monitors and controls a local Delivery process.
- class [DataDeliveryRemoteComm](#)
This class contacts a remote service to make a Delivery request.
- class [TransferParameters](#)
- class [CacheParameters](#)
The configured cache directories.
- class [DTRCallback](#)
The base class from which all callback-enabled classes should be derived.

- class [DTR](#)
Data Transfer Request.
- class [DTRLList](#)
Global list of all active DTRs in the system.
- class [DTRStatus](#)
Class representing the status of a [DTR](#).
- class [DTRErrorStatus](#)
A class to represent error states reported by various components.
- class [Generator](#)
Simple [Generator](#) implementation.
- class [Processor](#)
The [Processor](#) performs pre- and post-transfer operations.
- class [Scheduler](#)
The [Scheduler](#) is the control centre of the data staging framework.
- class [TransferShares](#)
[TransferShares](#) is used to implement fair-sharing and priorities.

Enumerations

- enum [StagingProcesses](#)
- enum [ProcessState](#)
- enum [CacheState](#) {

CACHEABLE, **NON_CACHEABLE**, **CACHE_RENEW**, **CACHE_ALREADY_PRESENT**,

CACHE_DOWNLOADED, **CACHE_LOCKED**, **CACHE_SKIP**, **CACHE_NOT_USED**

 }

4.3.1 Detailed Description

[DataStaging](#) contains all components for data transfer scheduling and execution.

4.3.2 Enumeration Type Documentation

4.3.2.1 enum [DataStaging::CacheState](#)

Represents possible cache states of this [DTR](#).

Enumerator:

- CACHEABLE*** Source should be cached.
- NON_CACHEABLE*** Source should not be cached.
- CACHE_RENEW*** Cache file should be deleted then re-downloaded.
- CACHE_ALREADY_PRESENT*** Source is available in cache from before.
- CACHE_DOWNLOADED*** Source has just been downloaded and put in cache.

CACHE_LOCKED Cache file is locked.

CACHE_SKIP Source is cacheable but due to some problem should not be cached.

CACHE_NOT_USED Cache was started but was not used.

Chapter 5

Data Structure Documentation

5.1 ArcCredential::ACACI Struct Reference

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

- VOMSAttribute.h

5.2 ArcCredential::ACATHOLDER Struct Reference

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

- VOMSAttribute.h

5.3 ArcCredential::ACATTR Struct Reference

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

- VOMSAttribute.h

5.4 ArcCredential::ACATTRIBUTE Struct Reference

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

- VOMSAttribute.h

5.5 ArcCredential::ACC Struct Reference

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

- VOMSAttribute.h

5.6 ArcCredential::ACCERTS Struct Reference

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

- VOMSAttribute.h

5.7 ArcCredential::ACDIGEST Struct Reference

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

- VOMSAttribute.h

5.8 ArcCredential::ACFORM Struct Reference

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

- VOMSAttribute.h

5.9 ArcCredential::ACFULLATTRIBUTES Struct Reference

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

- VOMSAttribute.h

5.10 ArcCredential::ACHOLDER Struct Reference

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

- VOMSAttribute.h

5.11 ArcCredential::ACIETFATTR Struct Reference

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

- VOMSAttribute.h

5.12 ArcCredential::ACINFO Struct Reference

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

- VOMSAttribute.h

5.13 ArcCredential::ACIS Struct Reference

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

- VOMSAttribute.h

5.14 ArcCredential::ACSEQ Struct Reference

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

- VOMSAttribute.h

5.15 ArcCredential::ACTARGET Struct Reference

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

- VOMSAttribute.h

5.16 ArcCredential::ACTARGETS Struct Reference

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

- VOMSAttribute.h

5.17 ArcCredential::ACVAL Struct Reference

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

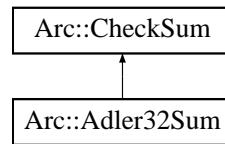
- VOMSAttribute.h

5.18 Arc::Adler32Sum Class Reference

Implementation of Adler32 checksum.

```
#include <CheckSum.h>
```

Inheritance diagram for Arc::Adler32Sum:



Public Member Functions

- virtual void [start](#) (void)
- virtual void [add](#) (void *buf, unsigned long long int len)
- virtual void [end](#) (void)
- virtual void [result](#) (unsigned char *&res, unsigned int &len) const
- virtual int [print](#) (char *buf, int len) const
- virtual void [scan](#) (const char *)
- virtual operator bool [\(void\)](#) const
- virtual bool [operator!](#) (void) const

5.18.1 Detailed Description

Implementation of Adler32 checksum.

This class is a specialized class of the [CheckSum](#) class. It provides an implementation of the Adler-32 checksum algorithm.

5.18.2 Member Function Documentation

5.18.2.1 virtual void Arc::Adler32Sum::add (void * buf, unsigned long long int len) [inline, virtual]

Add data to be checksummed.

This method calculates the checksum of the passed data chuck, taking into account the previous state of this object.

Parameters

<i>buf</i>	pointer to data chuck to be checksummed.
<i>len</i>	size of the data chuck.

Implements [Arc::CheckSum](#).

5.18.2.2 virtual void Arc::Adler32Sum::end(void) [inline, virtual]

Finalize the checksumming.

This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::CheckSum](#).

5.18.2.3 virtual int Arc::Adler32Sum::print(char * buf, int len) const [inline, virtual]

Retrieve result of checksum into a string.

The passed string buf is filled with result of checksum algorithm in base 16. At most len characters is filled into buffer buf. The hexadecimal value is prepended with "<algorithm>:", where <algorithm> is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters

<i>buf</i>	pointer to buffer which should be filled with checksum result.
<i>len</i>	max number of character filled into buffer.

Reimplemented from [Arc::CheckSum](#).

5.18.2.4 virtual void Arc::Adler32Sum::scan(const char * buf) [inline, virtual]

Set internal checksum state.

This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [CheckSum::print](#) method.

Parameters

<i>buf</i>	string containing textual representation of checksum
------------	--

See also

[CheckSum::print](#)

Implements [Arc::CheckSum](#).

5.18.2.5 virtual void Arc::Adler32Sum::start(void) [inline, virtual]

Initiate the checksum algorithm.

This method must be called before starting a new checksum calculation.

Implements [Arc::CheckSum](#).

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

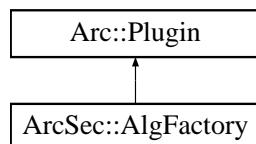
- CheckSum.h

5.19 ArcSec::AlgFactory Class Reference

Interface for algorithm factory class.

```
#include <AlgFactory.h>
```

Inheritance diagram for ArcSec::AlgFactory:



Public Member Functions

- virtual [CombiningAlg * createAlg \(const std::string &type\)=0](#)

5.19.1 Detailed Description

Interface for algorithm factory class.

[AlgFactory](#) is in charge of creating [CombiningAlg](#) according to the algorithm type given as argument of method `createAlg`. This class can be inherited for implementing a factory class which can create some specific combining algorithm objects.

5.19.2 Member Function Documentation

5.19.2.1 virtual [CombiningAlg* ArcSec::AlgFactory::createAlg \(const std::string & type \) \[pure virtual\]](#)

creat algorithm object based on the type algorithm type

Parameters

<code>type</code>	The type of combining algorithm
-------------------	---------------------------------

Returns

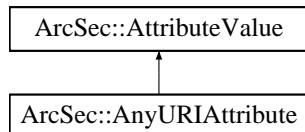
The object of [CombiningAlg](#)

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

- AlgFactory.h

5.20 ArcSec::AnyURIAttribute Class Reference

Inheritance diagram for ArcSec::AnyURIAttribute:



Public Member Functions

- virtual bool [equal \(AttributeValue *other, bool check_id=true\)](#)
- virtual std::string [encode \(\)](#)
- std::string [getId \(\)](#)
- virtual std::string [getType \(\)](#)

5.20.1 Member Function Documentation

5.20.1.1 virtual std::string [ArcSec::AnyURIAttribute::encode \(\) \[inline, virtual\]](#)

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.20.1.2 virtual bool [ArcSec::AnyURIAttribute::equal \(AttributeValue * value, bool check_id =true \) \[virtual\]](#)

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.20.1.3 std::string [ArcSec::AnyURIAttribute::getId \(\) \[inline, virtual\]](#)

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.20.1.4 virtual std::string [ArcSec::AnyURIAttribute::getType \(\) \[inline, virtual\]](#)

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

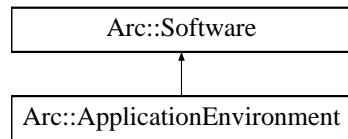
- AnyURIAttribute.h

5.21 Arc::ApplicationEnvironment Class Reference

[ApplicationEnvironment](#).

```
#include <ExecutionTarget.h>
```

Inheritance diagram for Arc::ApplicationEnvironment:



5.21.1 Detailed Description

[ApplicationEnvironment](#).

The ApplicationEnvironment is closely related to the definition given in [GLUE2](#). By extending the [Software](#) class the two [GLUE2](#) attributes AppName and AppVersion are mapped to two private members. However these can be obtained through the inherited member methods getName and getVersion.

[GLUE2](#) description: A description of installed application software or software environment characteristics available within one or more Execution Environments.

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

- ExecutionTarget.h

5.22 Arc::ApplicationType Class Reference

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

- JobDescription.h

5.23 Arc::ArcLocation Class Reference

Determines ARC installation location.

```
#include <ArcLocation.h>
```

Static Public Member Functions

- static void [Init](#) (std::string path)
- static const std::string & [Get](#) ()
- static std::list< std::string > [GetPlugins](#) ()

5.23.1 Detailed Description

Determines ARC installation location.

5.23.2 Member Function Documentation

5.23.2.1 static std::list<std::string> Arc::ArcLocation::GetPlugins() [static]

Returns ARC plugins directory location.

Main source is value of variable ARC_PLUGIN_PATH, otherwise path is derived from installation location.

5.23.2.2 static void Arc::ArcLocation::Init(std::string path) [static]

Initializes location information.

Main source is value of variable ARC_LOCATION, otherwise path to executable provided in is used. If nothing works then warning message is sent to logger and initial installation prefix is used.

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

- [ArcLocation.h](#)

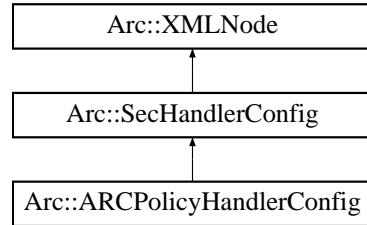
5.24 ArcSec::ArcPeriod Struct Reference

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

- [DateTimeAttribute.h](#)

5.25 Arc::ARCPolicyHandlerConfig Class Reference

Inheritance diagram for Arc::ARCPolicyHandlerConfig:



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

- ClientInterface.h

5.26 Arc::ArcVersion Class Reference

Determines ARC HED libraries version.

```
#include <ArcVersion.h>
```

5.26.1 Detailed Description

Determines ARC HED libraries version.

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

- ArcVersion.h

5.27 ArcSec::Attr Struct Reference

[Attr](#) contains a tuple of attribute type and value.

```
#include <Request.h>
```

5.27.1 Detailed Description

[Attr](#) contains a tuple of attribute type and value.

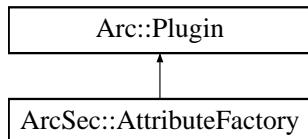
The documentation for this struct was generated from the following file:

- Request.h

5.28 ArcSec::AttributeFactory Class Reference

```
#include <AttributeFactory.h>
```

Inheritance diagram for ArcSec::AttributeFactory:



5.28.1 Detailed Description

Base attribute factory class

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

- [AttributeFactory.h](#)

5.29 Arc::AttributelIterator Class Reference

A const iterator class for accessing multiple values of an attribute.

```
#include <MessageAttributes.h>
```

Public Member Functions

- [AttributelIterator \(\)](#)
- [const std::string & operator* \(\) const](#)
- [const std::string * operator-> \(\) const](#)
- [const std::string & key \(void\) const](#)
- [const AttributelIterator & operator++ \(\)](#)
- [AttributelIterator operator++ \(int\)](#)
- [bool hasMore \(\) const](#)

Protected Member Functions

- [AttributelIterator \(AttrConstIter begin, AttrConstIter end\)](#)

Protected Attributes

- [AttrConstIter current_](#)
- [AttrConstIter end_](#)

Friends

- [class MessageAttributes](#)

5.29.1 Detailed Description

A const iterator class for accessing multiple values of an attribute.

This is an iterator class that is used when accessing multiple values of an attribute. The `getAll()` method of the [MessageAttributes](#) class returns an [AttributeIterator](#) object that can be used to access the values of the attribute.

Typical usage is:

```
MessageAttributes attributes;
...
for (AttributeIterator iterator=attributes.getAll("Foo:Bar");
     iterator.hasMore(); ++iterator)
    std::cout << *iterator << std::endl;
```

5.29.2 Constructor & Destructor Documentation

5.29.2.1 Arc::AttributeIterator::AttributeIterator ()

Default constructor.

The default constructor. Does nothing since all attributes are instances of well-behaving STL classes.

5.29.2.2 Arc::AttributeIterator::AttributeIterator (AttrConstIter begin, AttrConstIter end) [protected]

Protected constructor used by the [MessageAttributes](#) class.

This constructor is used to create an iterator for iteration over all values of an attribute. It is not supposed to be visible externally, but is only used from within the `getAll()` method of [MessageAttributes](#) class.

Parameters

<code>begin</code>	A const_iterator pointing to the first matching key-value pair in the internal multimap of the MessageAttributes class.
<code>end</code>	A const_iterator pointing to the first key-value pair in the internal multimap of the MessageAttributes class where the key is larger than the key searched for.

5.29.3 Member Function Documentation

5.29.3.1 bool Arc::AttributeIterator::hasMore () const

Predicate method for iteration termination.

This method determines whether there are more values for the iterator to refer to.

Returns

Returns true if there are more values, otherwise false.

5.29.3.2 const std::string& Arc::Attributeliterator::key (void) const

The key of attribute.

This method returns reference to key of attribute to which iterator refers.

5.29.3.3 const std::string& Arc::Attributeliterator::operator* () const

The dereference operator.

This operator is used to access the current value referred to by the iterator.

Returns

A (constant reference to a) string representation of the current value.

5.29.3.4 const Attributeliterator& Arc::Attributeliterator::operator++ ()

The prefix advance operator.

Advances the iterator to the next value. Works intuitively.

Returns

A const reference to this iterator.

5.29.3.5 Attributeliterator Arc::Attributeliterator::operator++ (int)

The postfix advance operator.

Advances the iterator to the next value. Works intuitively.

Returns

An iterator referring to the value referred to by this iterator before the advance.

5.29.3.6 const std::string* Arc::Attributeliterator::operator-> () const

The arrow operator.

Used to call methods for value objects (strings) conveniently.

5.29.4 Friends And Related Function Documentation

5.29.4.1 friend class [MessageAttributes](#) [friend]

The [MessageAttributes](#) class is a friend.

The constructor that creates an [Attributeliterator](#) that is connected to the internal multimap of the [MessageAttributes](#) class should not be exposed to the outside, but it still needs to be accessible from the getAll() method of the [MessageAttributes](#) class. Therefore, that class is a friend.

5.29.5 Field Documentation

5.29.5.1 AttrConstIter Arc::Attributeliterator::current_ [protected]

A const_iterator pointing to the current key-value pair.

This iterator is the internal representation of the current value. It points to the corresponding key-value pair in the internal multimap of the [MessageAttributes](#) class.

5.29.5.2 AttrConstIter Arc::Attributeliterator::end_ [protected]

A const_iterator pointing beyond the last key-value pair.

A const_iterator pointing to the first key-value pair in the internal multimap of the [MessageAttributes](#) class where the key is larger than the key searched for.

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

- [MessageAttributes.h](#)

5.30 ArcSec::AttributeProxy Class Reference

Interface for creating the [AttributeValue](#) object, it will be used by [AttributeFactory](#).

```
#include <AttributeProxy.h>
```

Public Member Functions

- virtual [AttributeValue](#) * [getAttribute](#) (const [Arc::XMLNode](#) &node)=0

5.30.1 Detailed Description

Interface for creating the [AttributeValue](#) object, it will be used by [AttributeFactory](#).

The [AttributeProxy](#) object will be insert into AttributeFactoty; and the [getAttribute\(node\)](#) method will be called inside AttributeFacroty.createvalue(node), in order to create a specific [AttributeValue](#)

5.30.2 Member Function Documentation

5.30.2.1 `virtual AttributeValue* ArcSec::AttributeProxy::getAttribute (const Arc::XMLNode & node) [pure virtual]`

Create a `AttributeValue` object according to the information inside the XMLNode as parameter.

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

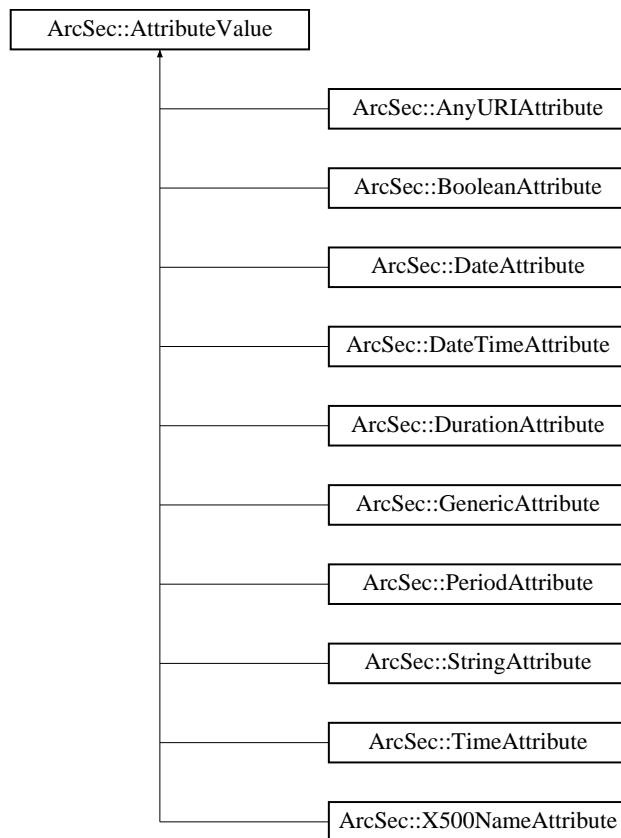
- `AttributeProxy.h`

5.31 ArcSec::AttributeValue Class Reference

Interface for containing different type of <Attribute> node for both policy and request.

```
#include <AttributeValue.h>
```

Inheritance diagram for ArcSec::AttributeValue:



Public Member Functions

- virtual bool `equal` (`AttributeValue` *`value`, `bool check_id=true`)=0
- virtual std::string `encode` ()=0
- virtual std::string `getType` ()=0
- virtual std::string `getId` ()=0

5.31.1 Detailed Description

Interface for containing different type of <Attribute> node for both policy and request.

<Attribute> contains different "Type" definition; Each type of <Attribute> needs different approach to compare the value. Any specific class which is for processing specific "Type" shoud inherit this class. The "Type" supported so far is: [StringAttribute](#), [DateAttribute](#), [TimeAttribute](#), [DurationAttribute](#), [PeriodAttribute](#), [AnyURIAttribute](#), [X500NameAttribute](#)

5.31.2 Member Function Documentation

5.31.2.1 virtual std::string ArcSec::AttributeValue::encode () [pure virtual]

encode the value in a string format

Implemented in [ArcSec::AnyURIAttribute](#), [ArcSec::BooleanAttribute](#), [ArcSec::DateTimeAttribute](#), [ArcSec::TimeAttribute](#), [ArcSec::DateAttribute](#), [ArcSec::DurationAttribute](#), [ArcSec::PeriodAttribute](#), [ArcSec::GenericAttribute](#), [ArcSec::StringAttribute](#), and [ArcSec::X500NameAttribute](#).

5.31.2.2 virtual bool ArcSec::AttributeValue::equal (`AttributeValue` * `value`, `bool check_id = true`) [pure virtual]

Evluate whether "this" equale to the parameter value

Implemented in [ArcSec::AnyURIAttribute](#), [ArcSec::BooleanAttribute](#), [ArcSec::DateTimeAttribute](#), [ArcSec::TimeAttribute](#), [ArcSec::DateAttribute](#), [ArcSec::DurationAttribute](#), [ArcSec::PeriodAttribute](#), [ArcSec::GenericAttribute](#), [ArcSec::StringAttribute](#), and [ArcSec::X500NameAttribute](#).

5.31.2.3 virtual std::string ArcSec::AttributeValue::getId () [pure virtual]

Get the Attributeld of the <Attribute>

Implemented in [ArcSec::AnyURIAttribute](#), [ArcSec::BooleanAttribute](#), [ArcSec::DateTimeAttribute](#), [ArcSec::TimeAttribute](#), [ArcSec::DateAttribute](#), [ArcSec::DurationAttribute](#), [ArcSec::PeriodAttribute](#), [ArcSec::GenericAttribute](#), [ArcSec::StringAttribute](#), and [ArcSec::X500NameAttribute](#).

5.31.2.4 virtual std::string ArcSec::AttributeValue::getType () [pure virtual]

Get the DataType of the <Attribute>

Implemented in [ArcSec::AnyURIAttribute](#), [ArcSec::BooleanAttribute](#), [ArcSec::DateTimeAttribute](#), [ArcSec::TimeAttribute](#), [ArcSec::DateAttribute](#), [ArcSec::DurationAttribute](#), [ArcSec::PeriodAttribute](#), [ArcSec::GenericAttribute](#), [ArcSec::StringAttribute](#), and [ArcSec::X500NameAttribute](#).

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

- `AttributeValue.h`

5.32 ArcSec::Attrs Class Reference

[Attrs](#) is a container for one or more [Attr](#).

```
#include <Request.h>
```

5.32.1 Detailed Description

[Attrs](#) is a container for one or more [Attr](#).

[Attrs](#) includes includes methods for inserting, getting items, and counting size as well

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

- `Request.h`

5.33 ArcSec::AuthzRequest Struct Reference

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

- `PDP.h`

5.34 ArcSec::AuthzRequestSection Struct Reference

```
#include <PDP.h>
```

5.34.1 Detailed Description

These structure are based on the request schema for [PDP](#), so far it can apply to the ArcPDP's request schema, see `src/hed/pdc/Request.xsd` and `src/hed/pdc/Request.xml`. It could also apply to the XACMLPDP's request schema, since the difference is minor.

Another approach is, the service composes/marshalls the xml structure directly, then the service should use difference code to compose for ArcPDP's request schema and XACMLPDP's schema, which is not so good.

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

- `PDP.h`

5.35 Arc::AutoPointer< T > Class Template Reference

Wrapper for pointer with automatic destruction.

```
#include <Utils.h>
```

Public Member Functions

- [AutoPointer \(void\)](#)
- [AutoPointer \(T *o\)](#)
- [~AutoPointer \(void\)](#)
- [T & operator* \(void\) const](#)
- [T * operator-> \(void\) const](#)
- [operator bool \(void\) const](#)
- [bool operator! \(void\) const](#)
- [operator T * \(void\) const](#)
- [T * Release \(void\)](#)

5.35.1 Detailed Description

```
template<typename T> class Arc::AutoPointer< T >
```

Wrapper for pointer with automatic destruction.

If ordinary pointer is wrapped in instance of this class it will be automatically destroyed when instance is destroyed. This is useful for maintaining pointers in scope of one function. Only pointers returned by new() are supported.

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

- [Utils.h](#)

5.36 Arc::Base64 Class Reference

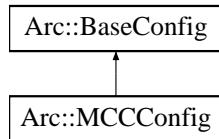
The documentation for this class was generated from the following file:

- [Base64.h](#)

5.37 Arc::BaseConfig Class Reference

```
#include <ArcConfig.h>
```

Inheritance diagram for Arc::BaseConfig:



Public Member Functions

- void [AddPluginsPath](#) (const std::string &path)
- void [AddPrivateKey](#) (const std::string &path)
- void [AddCertificate](#) (const std::string &path)
- void [AddProxy](#) (const std::string &path)
- void [AddCAFFile](#) (const std::string &path)
- void [AddCADir](#) (const std::string &path)
- void [AddOverlay](#) (XMLNode cfg)
- void [GetOverlay](#) (std::string fname)
- virtual XMLNode [MakeConfig](#) (XMLNode cfg) const

5.37.1 Detailed Description

Configuration for client interface. It contains information which can't be expressed in class constructor arguments. Most probably common things like software installation location, identity of user, etc.

5.37.2 Member Function Documentation

5.37.2.1 void Arc::BaseConfig::AddCADir (const std::string & path)

Add CA directory

5.37.2.2 void Arc::BaseConfig::AddCAFFile (const std::string & path)

Add CA file

5.37.2.3 void Arc::BaseConfig::AddCertificate (const std::string & path)

Add certificate

5.37.2.4 void Arc::BaseConfig::AddOverlay (XMLNode cfg)

Add configuration overlay

5.37.2.5 void Arc::BaseConfig::AddPluginsPath (const std::string & path)

Adds non-standard location of plugins

5.37.2.6 void Arc::BaseConfig::AddPrivateKey (const std::string & path)

Add private key

5.37.2.7 void Arc::BaseConfig::AddProxy (const std::string & path)

Add credentials proxy

5.37.2.8 void Arc::BaseConfig::GetOverlay (std::string fname)

Read overlay from file

5.37.2.9 virtual XMLNode Arc::BaseConfig::MakeConfig (XMLNode cfg) const [virtual]

Adds configuration part corresponding to stored information into common configuration tree supplied in 'cfg' argument. Returns reference to XML node representing configuration of [ModuleManager](#)

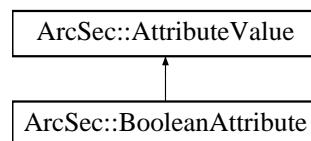
Reimplemented in [Arc::MCCConfig](#).

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

- ArcConfig.h

5.38 ArcSec::BooleanAttribute Class Reference

Inheritance diagram for ArcSec::BooleanAttribute:



Public Member Functions

- virtual bool [equal](#) ([AttributeValue](#) *o, bool check_id=true)
- virtual std::string [encode](#) ()
- std::string [getId](#) ()
- std::string [getType](#) ()

5.38.1 Member Function Documentation

5.38.1.1 `virtual std::string ArcSec::BooleanAttribute::encode() [inline, virtual]`

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.38.1.2 `virtual bool ArcSec::BooleanAttribute::equal(AttributeValue * value, bool check_id = true) [virtual]`

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.38.1.3 `std::string ArcSec::BooleanAttribute::getId() [inline, virtual]`

Get the AttributId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.38.1.4 `std::string ArcSec::BooleanAttribute::getType() [inline, virtual]`

Get the DataType of the <Attribute>

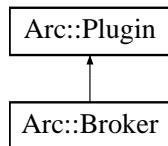
Implements [ArcSec::AttributeValue](#).

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

- BooleanAttribute.h

5.39 Arc::Broker Class Reference

Inheritance diagram for Arc::Broker:



Public Member Functions

- `const ExecutionTarget * GetBestTarget()`
- `void PreFilterTargets(std::list< ExecutionTarget > &targets, const JobDescription &jobdesc, const std::list< URL > &rejectTargets=std::list< URL >())`
- `void RegisterJobsubmission()`

Protected Member Functions

- virtual void [SortTargets \(\)=0](#)

Protected Attributes

- std::list< [ExecutionTarget *](#) > [PossibleTargets](#)
- bool [TargetSortingDone](#)

5.39.1 Member Function Documentation

5.39.1.1 const [ExecutionTarget*](#) [Arc::Broker::GetBestTarget \(\)](#)

Returns next target from the list of [ExecutionTarget](#) objects.

When first called this method will sort its list of [ExecutionTarget](#) objects, which have been filled by the [PreFilterTargets](#) method, and then the first target in the list will be returned.

If this is not the first call then the next target in the list is simply returned.

If there are no targets in the list or the end of the target list have been reached the NULL pointer is returned.

Returns

The pointer to the next [ExecutionTarget](#) in the list is returned.

5.39.1.2 void [Arc::Broker::PreFilterTargets \(std::list< ExecutionTarget > & targets, const JobDescription & jobdesc, const std::list< URL > & rejectTargets = std::list< URL > \(\) \)](#)

Filter [ExecutionTarget](#) objects according to attributes in [JobDescription](#) object.

Each of the [ExecutionTarget](#) objects in the passed list will be matched against attributes in the passed [JobDescription](#) object. For a list of which attributes are considered for matchmaking see appendix B of the libarcclient technical manual (NORDUGRID-TECH-20). If a [ExecutionTarget](#) object matches the job description, it is added to the internal list of [ExecutionTarget](#) objects. NOTE: The list of [ExecutionTarget](#) objects must be available through out the scope of this [Broker](#) object.

Parameters

<i>targets</i>	A list of ExecutionTarget objects to be considered for addition to the Broker .
<i>jobdesc</i>	JobDescription object holding requirements.
<i>rejectTargets</i>	

5.39.1.3 virtual void Arc::Broker::SortTargets() [protected, pure virtual]

Custom Brokers should implement this method.

The task is to sort the PossibleTargets list by "custom" way, for example: FastestQueue-Broker, [ExecutionTarget](#) which has the shortest queue lenght will be at the begining of the PossibleTargets list

5.39.2 Field Documentation

5.39.2.1 std::list<[ExecutionTarget](#)*> Arc::Broker::PossibleTargets [protected]

This content the Prefilteres Execution Targets.

If an Execution Target has enought memory, CPU, diskspace, etc. for the actual job requirement than it will be added to the PossibleTargets list

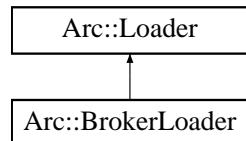
The documentation for this class was generated from the following file:

- Broker.h

5.40 Arc::BrokerLoader Class Reference

```
#include <Broker.h>
```

Inheritance diagram for Arc::BrokerLoader:



Public Member Functions

- [BrokerLoader\(\)](#)
- [~BrokerLoader\(\)](#)
- [Broker * load](#) (const std::string &name, const [UserConfig](#) &usercfg)
- [const std::list< Broker * > & GetBrokers\(\)](#) const

5.40.1 Detailed Description

Class responsible for loading [Broker](#) plugins. The [Broker](#) objects returned by a [Broker-Loader](#) must not be used after the [BrokerLoader](#) goes out of scope.

5.40.2 Constructor & Destructor Documentation

5.40.2.1 Arc::BrokerLoader::BrokerLoader ()

Constructor Creates a new [BrokerLoader](#).

5.40.2.2 Arc::BrokerLoader::~BrokerLoader ()

Destructor Calling the destructor destroys all Brokers loaded by the [BrokerLoader](#) instance.

5.40.3 Member Function Documentation

5.40.3.1 const std::list<[Broker](#)*>& Arc::BrokerLoader::GetBrokers () const [inline]

Retrieve the list of loaded Brokers.

Returns

A reference to the list of Brokers.

5.40.3.2 [Broker](#)* Arc::BrokerLoader::load (const std::string & name, const [UserConfig](#) & usercfg)

Load a new [Broker](#)

Parameters

<i>name</i>	The name of the Broker to load.
<i>usercfg</i>	The UserConfig object for the new Broker .

Returns

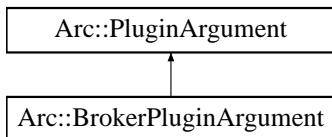
A pointer to the new [Broker](#) (NULL on error).

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

- Broker.h

5.41 Arc::BrokerPluginArgument Class Reference

Inheritance diagram for Arc::BrokerPluginArgument:



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

- Broker.h

5.42 Arc::ByteArray Class Reference

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

- ByteArray.h

5.43 DataStaging::CacheParameters Class Reference

The configured cache directories.

```
#include <DTR.h>
```

Public Member Functions

- [CacheParameters \(void\)](#)
- [CacheParameters \(std::vector< std::string > caches, std::vector< std::string > remote_caches, std::vector< std::string > drain_caches\)](#)

Data Fields

- std::vector< std::string > [cache_dirs](#)
- std::vector< std::string > [remote_cache_dirs](#)
- std::vector< std::string > [drain_cache_dirs](#)

5.43.1 Detailed Description

The configured cache directories.

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

- DTR.h

5.44 ArcCredential::cert_verify_context Struct Reference

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

- CertUtil.h

5.45 Arc::CertEnvLocker Class Reference

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

- UserConfig.h

5.46 Arc::ChainContext Class Reference

Interface to chain specific functionality.

```
#include <MCCLoader.h>
```

Public Member Functions

- `operator PluginsFactory *()`

5.46.1 Detailed Description

Interface to chain specific functionality.

Object of this class is associated with every [MCCLoader](#) object. It is accessible for [MCC](#) and [Service](#) components and provides an interface to manipulate chains stored in [Loader](#). This makes it possible to modify chains dynamically - like deploying new services on demand.

5.46.2 Member Function Documentation

5.46.2.1 Arc::ChainContext::operator PluginsFactory *() [inline]

Returns associated [PluginsFactory](#) object

References [Arc::Loader::factory_](#).

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

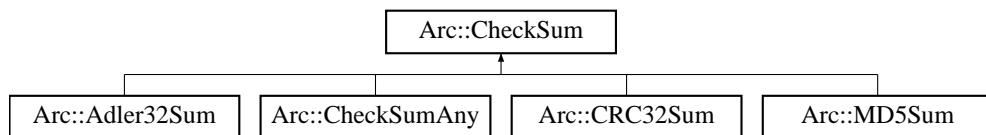
- MCCLoader.h

5.47 Arc::CheckSum Class Reference

Interface for checksum manipulations.

```
#include <CheckSum.h>
```

Inheritance diagram for Arc::CheckSum:



Public Member Functions

- [CheckSum \(void\)](#)
- virtual void [start \(void\)=0](#)
- virtual void [add \(void *buf, unsigned long long int len\)=0](#)
- virtual void [end \(void\)=0](#)
- virtual void [result \(unsigned char *&res, unsigned int &len\) const =0](#)
- virtual int [print \(char *buf, int len\) const](#)
- virtual void [scan \(const char *buf\)=0](#)
- virtual [operator bool \(void\) const](#)
- virtual bool [operator! \(void\) const](#)

5.47.1 Detailed Description

Interface for checksum manipulations.

This class is an interface and is extended in the specialized classes [CRC32Sum](#), [MD5Sum](#) and [Adler32Sum](#). The interface is among others used during data transfers through DataBuffer class

See also

[CRC32Sum](#)
[MD5Sum](#)
[Adler32Sum](#)

5.47.2 Member Function Documentation

5.47.2.1 virtual void Arc::CheckSum::add (void * buf, unsigned long long int len) [pure virtual]

Add data to be checksummed.

This method calculates the checksum of the passed data chuck, taking into account the previous state of this object.

Parameters

<i>buf</i>	pointer to data chuck to be checksummed.
<i>len</i>	size of the data chuck.

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::CheckSumAny](#).

Referenced by [Arc::CheckSumAny::add\(\)](#).

5.47.2.2 virtual void Arc::CheckSum::end(void) [pure virtual]

Finalize the checksumming.

This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::CheckSumAny](#).

Referenced by [Arc::CheckSumAny::end\(\)](#).

5.47.2.3 virtual int Arc::CheckSum::print(char * buf, int len) const [inline, virtual]

Retrieve result of checksum into a string.

The passed string buf is filled with result of checksum algorithm in base 16. At most len characters is filled into buffer buf. The hexadecimal value is prepended with "<algorithm>:", where <algorithm> is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters

<i>buf</i>	pointer to buffer which should be filled with checksum result.
<i>len</i>	max number of character filled into buffer.

Reimplemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::CheckSumAny](#).

Referenced by [Arc::CheckSumAny::print\(\)](#).

5.47.2.4 virtual void Arc::CheckSum::scan(const char * buf) [pure virtual]

Set internal checksum state.

This method sets the internal state to that of the passed textural representation. The format passed to this method must be the same as retrieved from the [CheckSum::print](#) method.

Parameters

<i>buf</i>	string containing textural representation of checksum
------------	---

See also[CheckSum::print](#)

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::CheckSumAny](#).

Referenced by [Arc::CheckSumAny::scan\(\)](#).

5.47.2.5 virtual void Arc::CheckSum::start (void) [pure virtual]

Initiate the checksum algorithm.

This method must be called before starting a new checksum calculation.

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::CheckSumAny](#).

Referenced by [Arc::CheckSumAny::start\(\)](#).

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

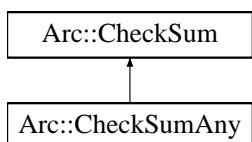
- [CheckSum.h](#)

5.48 Arc::CheckSumAny Class Reference

Wrapper for [CheckSum](#) class.

```
#include <CheckSum.h>
```

Inheritance diagram for Arc::CheckSumAny:



Public Member Functions

- [virtual void start \(void\)](#)
- [virtual void add \(void *buf, unsigned long long int len\)](#)
- [virtual void end \(void\)](#)
- [virtual void result \(unsigned char *&res, unsigned int &len\) const](#)
- [virtual int print \(char *buf, int len\) const](#)
- [virtual void scan \(const char *buf\)](#)
- [virtual operator bool \(void\) const](#)
- [virtual bool operator! \(void\) const](#)

Static Public Member Functions

- static std::string [FileChecksum](#) (const std::string &filepath, type tp=md5, bool decimalbase=false)

5.48.1 Detailed Description

Wrapper for [CheckSum](#) class.

To be used for manipulation of any supported checksum type in a transparent way.

5.48.2 Member Function Documentation

5.48.2.1 virtual void Arc::CheckSumAny::add (void * buf, unsigned long long int len) [inline, virtual]

Add data to be checksummed.

This method calculates the checksum of the passed data chuck, taking into account the previous state of this object.

Parameters

<i>buf</i>	pointer to data chuck to be checksummed.
<i>len</i>	size of the data chuck.

Implements [Arc::CheckSum](#).

References Arc::CheckSum::add().

5.48.2.2 virtual void Arc::CheckSumAny::end (void) [inline, virtual]

Finalize the checksumming.

This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::CheckSum](#).

References Arc::CheckSum::end().

5.48.2.3 static std::string Arc::CheckSumAny::FileChecksum (const std::string & filepath, type tp = md5, bool decimalbase = false) [static]

Get checksum of a file.

This method provide an easy way to get the checksum of a file, by only specifying the path to the file. Optionally the checksum type can be specified, if not the MD5 algorithm will be used.

Parameters

<i>filepath</i>	path to file of which checksum should be calculated
<i>tp</i>	type of checksum algorithm to use, default is md5.
<i>decimalbase</i>	specifies whether output should be in base 10 or 16

Returns

a string containing the calculated checksum is returned.

5.48.2.4 virtual int Arc::CheckSumAny::print (char * *buf*, int *len*) const [inline, virtual]

Retrieve result of checksum into a string.

The passed string buf is filled with result of checksum algorithm in base 16. At most len characters is filled into buffer buf. The hexadecimal value is prepended with "<algorithm>:", where <algorithm> is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters

<i>buf</i>	pointer to buffer which should be filled with checksum result.
<i>len</i>	max number of character filled into buffer.

Reimplemented from [Arc::CheckSum](#).

References [Arc::CheckSum::print\(\)](#).

5.48.2.5 virtual void Arc::CheckSumAny::scan (const char * *buf*) [inline, virtual]

Set internal checksum state.

This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [CheckSum::print](#) method.

Parameters

<i>buf</i>	string containing textual representation of checksum
------------	--

See also

[CheckSum::print](#)

Implements [Arc::CheckSum](#).

References [Arc::CheckSum::scan\(\)](#).

5.48.2.6 virtual void Arc::CheckSumAny::start(void) [inline, virtual]

Initiate the checksum algorithm.

This method must be called before starting a new checksum calculation.

Implements [Arc::CheckSum](#).

References [Arc::CheckSum::start\(\)](#).

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

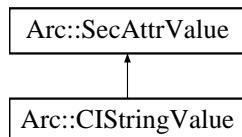
- [CheckSum.h](#)

5.49 Arc::CIStringValue Class Reference

This class implements case insensitive strings as security attributes.

```
#include <CIStringValue.h>
```

Inheritance diagram for Arc::CIStringValue:



Public Member Functions

- [CIStringValue \(\)](#)
- [CIStringValue \(const char *ss\)](#)
- [CIStringValue \(const std::string &ss\)](#)
- virtual [operator bool \(\)](#)

Protected Member Functions

- virtual [bool equal \(SecAttrValue &b\)](#)

5.49.1 Detailed Description

This class implements case insensitive strings as security attributes.

This is an example of how to inherit [SecAttrValue](#). The class is meant to implement security attributes that are case insensitive strings.

5.49.2 Constructor & Destructor Documentation

5.49.2.1 Arc::CIStringValue::CIStringValue()

Default constructor

5.49.2.2 Arc::CIStringValue::CIStringValue(const char * ss)

This is a constructor that takes a string litteral.

5.49.2.3 Arc::CIStringValue::CIStringValue(const std::string & ss)

This is a constructor that takes a string object.

5.49.3 Member Function Documentation

5.49.3.1 virtual bool Arc::CIStringValue::equal(SecAttrValue & b) [protected, virtual]

This function returns true if two strings are the same apart from letter case

Reimplemented from [Arc::SecAttrValue](#).

5.49.3.2 virtual Arc::CIStringValue::operator bool() [virtual]

This function returns false if the string is empty or uninitialized

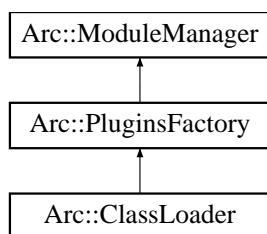
Reimplemented from [Arc::SecAttrValue](#).

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

- CIStringValue.h

5.50 Arc::ClassLoader Class Reference

Inheritance diagram for Arc::ClassLoader:

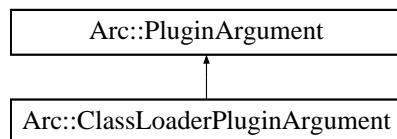


The documentation for this class was generated from the following file:

- ClassLoader.h

5.51 Arc::ClassLoaderPluginArgument Class Reference

Inheritance diagram for Arc::ClassLoaderPluginArgument:



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

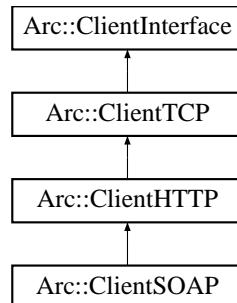
- ClassLoader.h

5.52 Arc::ClientHTTP Class Reference

Class for setting up a [MCC](#) chain for HTTP communication.

```
#include <ClientInterface.h>
```

Inheritance diagram for Arc::ClientHTTP:



5.52.1 Detailed Description

Class for setting up a [MCC](#) chain for HTTP communication.

The [ClientHTTP](#) class inherits from the [ClientTCP](#) class and adds an HTTP [MCC](#) to the chain.

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

- ClientInterface.h

5.53 Arc::ClientHTTPwithSAML2SSO Class Reference

Public Member Functions

- `ClientHTTPwithSAML2SSO ()`
- `MCC_Status process (const std::string &method, PayloadRawInterface *request, HTTPClientInfo *info, PayloadRawInterface **response, const std::string &idp_name, const std::string &username, const std::string &password, const bool reuse_authn=false)`

5.53.1 Constructor & Destructor Documentation

5.53.1.1 `Arc::ClientHTTPwithSAML2SSO::ClientHTTPwithSAML2SSO () [inline]`

Constructor creates [MCC](#) chain and connects to server.

5.53.2 Member Function Documentation

5.53.2.1 `MCC_Status Arc::ClientHTTPwithSAML2SSO::process (const std::string & method, PayloadRawInterface * request, HTTPClientInfo * info, PayloadRawInterface ** response, const std::string & idp_name, const std::string & username, const std::string & password, const bool reuse_authn = false)`

Send HTTP request and receive response.

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

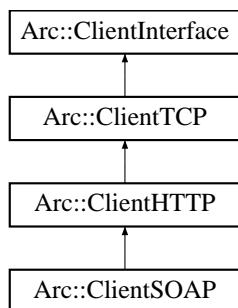
- `ClientSAML2SSO.h`

5.54 Arc::ClientInterface Class Reference

Utility base class for [MCC](#).

```
#include <ClientInterface.h>
```

Inheritance diagram for Arc::ClientInterface:



5.54.1 Detailed Description

Utility base class for [MCC](#).

The [ClientInterface](#) class is a utility base class used for configuring a client side [Message Chain Component \(MCC\)](#) chain and loading it into memory. It has several specializations of increasing complexity of the [MCC](#) chains.

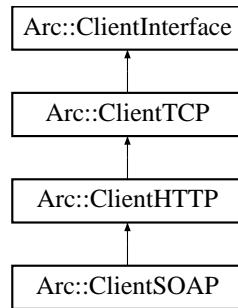
The documentation for this class was generated from the following file:

- ClientInterface.h

5.55 Arc::ClientSOAP Class Reference

```
#include <ClientInterface.h>
```

Inheritance diagram for Arc::ClientSOAP:



Public Member Functions

- [ClientSOAP \(\)](#)
- [MCC_Status process \(PayloadSOAP *request, PayloadSOAP **response\)](#)
- [MCC_Status process \(const std::string &action, PayloadSOAP *request, PayloadSOAP **response\)](#)
- [MCC * GetEntry \(\)](#)
- void [AddSecHandler \(XMLNode handlcfg, const std::string &libname="", const std::string &libpath=""\)](#)
- virtual bool [Load \(\)](#)

5.55.1 Detailed Description

Class with easy interface for sending/receiving SOAP messages over HTTP(S/G). It takes care of configuring [MCC](#) chain and making an entry point.

5.55.2 Constructor & Destructor Documentation

5.55.2.1 `Arc::ClientSOAP::ClientSOAP() [inline]`

Constructor creates [MCC](#) chain and connects to server.

5.55.3 Member Function Documentation

5.55.3.1 `void Arc::ClientSOAP::AddSecHandler(XMLNode handlercfg, const std::string & libname = "", const std::string & libpath = "")`

Adds security handler to configuration of SOAP [MCC](#)

Reimplemented from [Arc::ClientHTTP](#).

5.55.3.2 `MCC* Arc::ClientSOAP::GetEntry() [inline]`

Returns entry point to SOAP [MCC](#) in configured chain. To initialize entry point [Load\(\)](#) method must be called.

Reimplemented from [Arc::ClientHTTP](#).

5.55.3.3 `virtual bool Arc::ClientSOAP::Load() [virtual]`

Instantiates pluggable elements according to generated configuration

Reimplemented from [Arc::ClientHTTP](#).

5.55.3.4 `MCC_Status Arc::ClientSOAP::process(PayloadSOAP * request, PayloadSOAP ** response)`

Send SOAP request and receive response.

5.55.3.5 `MCC_Status Arc::ClientSOAP::process(const std::string & action, PayloadSOAP * request, PayloadSOAP ** response)`

Send SOAP request with specified SOAP action and receive response.

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

- ClientInterface.h

5.56 Arc::ClientSOAPwithSAML2SSO Class Reference

Public Member Functions

- [ClientSOAPwithSAML2SSO \(\)](#)
- [MCC_Status process \(PayloadSOAP *request, PayloadSOAP **response, const std::string &idp_name, const std::string &username, const std::string &password, const bool reuse_authn=false\)](#)
- [MCC_Status process \(const std::string &action, PayloadSOAP *request, PayloadSOAP **response, const std::string &idp_name, const std::string &username, const std::string &password, const bool reuse_authn=false\)](#)

5.56.1 Constructor & Destructor Documentation

5.56.1.1 Arc::ClientSOAPwithSAML2SSO::ClientSOAPwithSAML2SSO () [inline]

Constructor creates [MCC](#) chain and connects to server.

5.56.2 Member Function Documentation

5.56.2.1 MCC_Status Arc::ClientSOAPwithSAML2SSO::process (PayloadSOAP * request, PayloadSOAP ** response, const std::string & idp_name, const std::string & username, const std::string & password, const bool reuse_authn = false)

Send SOAP request and receive response.

5.56.2.2 MCC_Status Arc::ClientSOAPwithSAML2SSO::process (const std::string & action, PayloadSOAP * request, PayloadSOAP ** response, const std::string & idp_name, const std::string & username, const std::string & password, const bool reuse_authn = false)

Send SOAP request with specified SOAP action and receive response.

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

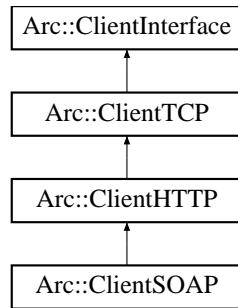
- ClientSAML2SSO.h

5.57 Arc::ClientTCP Class Reference

Class for setting up a [MCC](#) chain for TCP communication.

```
#include <ClientInterface.h>
```

Inheritance diagram for Arc::ClientTCP:



5.57.1 Detailed Description

Class for setting up a [MCC](#) chain for TCP communication.

The [ClientTCP](#) class is a specialization of the [ClientInterface](#) which sets up a client [MCC](#) chain for TCP communication, and optionally with a security layer on top which can be either TLS, GSI or SSL3.

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

- ClientInterface.h

5.58 Arc::ClientX509Delegation Class Reference

Public Member Functions

- [ClientX509Delegation \(\)](#)
- [bool createDelegation \(DelegationType deleg, std::string &delegation_id\)](#)
- [bool acquireDelegation \(DelegationType deleg, std::string &delegation_cred, std::string &delegation_id, const std::string cred_identity="", const std::string cred_delegator_ip="", const std::string username="", const std::string password ""\)](#)

5.58.1 Constructor & Destructor Documentation

5.58.1.1 Arc::ClientX509Delegation::ClientX509Delegation () [inline]

Constructor creates [MCC](#) chain and connects to server.

5.58.2 Member Function Documentation

5.58.2.1 bool Arc::ClientX509Delegation::acquireDelegation (DelegationType *deleg*, std::string & *delegation_cred*, std::string & *delegation_id*, const std::string *cred_identity* = " ", const std::string *cred_delegator_ip* = " ", const std::string *username* = " ", const std::string *password* = " ")

Acquire delegation credential from delegation service. This method should be called by intermediate service ('n+1' service as explained on above) in order to use this delegation credential on behalf of the EEC's holder.

Parameters

<i>deleg</i>	Delegation type
<i>delegation_id</i>	delegation ID which is used to look up the credential by delegation service
<i>cred_identity</i>	the identity (in case of x509 credential, it is the DN of EEC credential).
<i>cred_delegator_ip</i>	the IP address of the credential delegator. Regard of delegation, an intermediate service should accomplish three tasks: 1. Acquire 'n' level delegation credential (which is delegated by 'n-1' level delegator) from delegation service; 1. Create 'n+1' level delegation credential to delegation service; 2. Use 'n' level delegation credential to act on behalf of the EEC's holder. In case of absense of delegation_id, the 'n-1' level delegator's IP address and credential's identity are supposed to be used for look up the delegation credential from delegation service.

5.58.2.2 bool Arc::ClientX509Delegation::createDelegation (DelegationType *deleg*, std::string & *delegation_id*)

Create the delegation credential according to the different remote delegation service. This method should be called by holder of EEC(end entity credential) which would delegate its EEC credential, or by holder of delegated credential(normally, the holder is intermediate service) which would further delegate the credential (on behalf of the original EEC's holder) (for instance, the 'n' intermediate service creates a delegation credential, then the 'n+1' intermediate service aquires this delegation credential from the delegation service and also acts on behalf of the EEC's holder by using this delegation credential).

Parameters

<i>deleg</i>	Delegation type
<i>delegation_id</i>	For gridsite delegation service, the delegation_id is supposed to be created by client side, and sent to service side; for ARC delegation service, the delegation_id is supposed to be created by service side, and returned back. So for gridsite delegation service, this parameter is treated as input, while for ARC delegation service, it is treated as output.

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

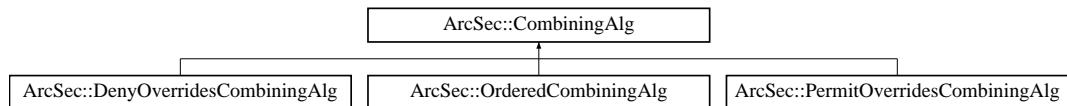
- ClientX509Delegation.h

5.59 ArcSec::CombiningAlg Class Reference

Interface for combining algorithm.

```
#include <CombiningAlg.h>
```

Inheritance diagram for ArcSec::CombiningAlg:



Public Member Functions

- virtual Result [combine \(EvaluationCtx *ctx, std::list< Policy * > policies\)=0](#)
- virtual const std::string & [getAlgId \(void\) const =0](#)

5.59.1 Detailed Description

Interface for combining algorithm.

This class is used to implement a specific combining algorithm for combining policies.

5.59.2 Member Function Documentation

5.59.2.1 virtual Result ArcSec::CombiningAlg::combine (EvaluationCtx * ctx, std::list< Policy * > policies) [pure virtual]

Evaluate request against policy, and if there are more than one policies, combine the evaluation results according to the combining algorithm implemented inside in the method `combine(ctx, policies)` itself.

Parameters

<code>ctx</code>	The information about request is included
<code>policies</code>	The "match" and "eval" method inside each policy will be called, and then those results from each policy will be combined according to the combining algorithm inside CombingAlg class.

Implemented in [ArcSec::DenyOverridesCombiningAlg](#), and [ArcSec::PermitOverridesCombiningAlg](#).

5.59.2.2 virtual const std::string& ArcSec::CombiningAlg::getAlgId (void) const [pure virtual]

Get the identifier of the combining algorithm class

Returns

The identity of the algorithm

Implemented in [ArcSec::DenyOverridesCombiningAlg](#), and [ArcSec::PermitOverridesCombiningAlg](#).

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

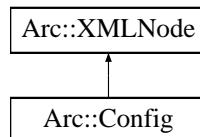
- [CombiningAlg.h](#)

5.60 Arc::Config Class Reference

Configuration element - represents (sub)tree of ARC configuration.

```
#include <ArcConfig.h>
```

Inheritance diagram for Arc::Config:



Public Member Functions

- [Config \(\)](#)
- [Config \(const char *filename\)](#)
- [Config \(const std::string &xml_str\)](#)
- [Config \(XMLNode xml\)](#)
- [Config \(long cfg_ptr_addr\)](#)
- [Config \(const Config &cfg\)](#)
- void [print \(void\)](#)
- bool [parse \(const char *filename\)](#)
- const std::string & [getFileName \(void\) const](#)
- void [setFileName \(const std::string &filename\)](#)
- void [save \(const char *filename\)](#)

5.60.1 Detailed Description

Configuration element - represents (sub)tree of ARC configuration.

This class is intended to be used to pass configuration details to various parts of HED and external modules. Currently it's just a wrapper over XML tree. But than may change in a future, although interface should be preserved. Currently it is capable of loading XML configuration document from file. In future it will be capable of loading more user-readable format and process it into tree-like structure convenient for machine processing (XML-like). So far there are no schema and/or namespaces assigned.

5.60.2 Constructor & Destructor Documentation

5.60.2.1 `Arc::Config::Config() [inline]`

Creates empty XML tree

5.60.2.2 `Arc::Config::Config(const char * filename)`

Loads configuration document from file '*filename*'

5.60.2.3 `Arc::Config::Config(const std::string & xml_str) [inline]`

Parse configuration document from memory

5.60.2.4 `Arc::Config::Config(XMLNode xml) [inline]`

Acquire existing XML (sub)tree. Content is not copied. Make sure XML tree is not destroyed while in use by this object.

5.60.2.5 `Arc::Config::Config(long cfg_ptr_addr)`

Copy constructor used by language bindings

5.60.2.6 `Arc::Config::Config(const Config & cfg)`

Copy constructor used by language bindings

5.60.3 Member Function Documentation

5.60.3.1 `const std::string& Arc::Config::getFileName(void) const [inline]`

Gives back file name of config file or empty string if it was generated from the `XMLNode` subtree

5.60.3.2 `bool Arc::Config::parse(const char * filename)`

Parse configuration document from file '*filename*'

5.60.3.3 `void Arc::Config::print(void)`

Print structure of document. For debugging purposes. Printed content is not an XML document.

5.60.3.4 void Arc::Config::save (const char * *filename*)

Save to file

5.60.3.5 void Arc::Config::setFileName (const std::string & *filename*) [inline]

Set the file name of config file

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

- ArcConfig.h

5.61 Arc::ConfusaCertHandler Class Reference

```
#include <ConfusaCertHandler.h>
```

Public Member Functions

- [ConfusaCertHandler](#) (int keyszie, const std::string dn)
- std::string [getCertRequestB64](#) ()
- bool [createCertRequest](#) (std::string password="", std::string storedir=".")

5.61.1 Detailed Description

Wrapper around [Credential](#) handling the Confusa specifics.

5.61.2 Constructor & Destructor Documentation

5.61.2.1 Arc::ConfusaCertHandler::ConfusaCertHandler (int *keysize*, const std::string *dn*)

Create a new [ConfusaCertHandler](#) for DN dn and given keyszie Basically Confusa cert handler wraps around [Credential](#)

5.61.3 Member Function Documentation

5.61.3.1 bool Arc::ConfusaCertHandler::createCertRequest (std::string *password* = " ", std::string *storedir* = " . / ")

Create a new end entity certificate, with a private key encrypted with password password. Private key and certificate will be stored in directory storedir.

5.61.3.2 std::string Arc::ConfusaCertHandler::getCertRequestB64()

Get the certificate request managed by this confusa cert handler in base 64 encoding

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

- ConfusaCertHandler.h

5.62 Arc::ConfusaParserUtils Class Reference

```
#include <ConfusaParserUtils.h>
```

Static Public Member Functions

- static std::string [urlencode](#) (const std::string url)
- static std::string [urlencode_params](#) (const std::string url)
- static xmlDocPtr [get_doc](#) (const std::string xml_file)
- static void [destroy_doc](#) (xmlDocPtr doc)
- static std::string [extract_body_information](#) (const std::string html_string)
- static std::string [handle_redirect_step](#) (Arc::MCCConfig cfg, const std::string remote_url, std::string *cookies=NULL, std::multimap<std::string, std::string> *httpAttributes=NULL)
- static std::string [evaluate_path](#) (xmlDocPtr doc, const std::string xpathExpr, std::list<std::string> *contentList=NULL)

5.62.1 Detailed Description

Methods often needed in evaluation web pages from the Confusa WebSSO workflow

5.62.2 Member Function Documentation

5.62.2.1 static void Arc::ConfusaParserUtils::destroy_doc (xmlDocPtr doc) [static]

Destroy a libxml2 doc representation

5.62.2.2 static std::string Arc::ConfusaParserUtils::evaluate_path (xmlDocPtr doc, const std::string xpathExpr, std::list<std::string> * contentList = NULL) [static]

Evaluate the given xPathExpr on the document ptr. Return a string with the FIRST result if contentList is NULL. Return a string with the first result and all results, including the first one, in contentList if contentList is not null.

5.62.2.3 static std::string Arc::ConfusaParserUtils::extract_body_information (const std::string *html_string*) [static]

Get the part only within <body> and </body> in a HTML string For parsing, usually only this part is interesting.

5.62.2.4 static xmlDocPtr Arc::ConfusaParserUtils::get_doc (const std::string *xml_file*) [static]

Construct a lixml2 doc representation from the xml file

5.62.2.5 static std::string Arc::ConfusaParserUtils::handle_redirect_step (Arc::MCCConfig *cfg*, const std::string *remote_url*, std::string * *cookies* = NULL, std::multimap< std::string, std::string > * *httpAttributes* = NULL) [static]

Handle a single redirect step from the SAML2 WebSSO profile. Store the received cookie in *cookie and pass the given httpAttributes to the site during redirect.

5.62.2.6 static std::string Arc::ConfusaParserUtils::urlencode (const std::string *url*) [static]

urlencode the passed string

5.62.2.7 static std::string Arc::ConfusaParserUtils::urlencode_params (const std::string *url*) [static]

Urlencode the passed string with respect to the parameters. The difference to urlencode is that the parameters will keep their separators, i.e. the ? and & separating parameters will be preserved.

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

- ConfusaParserUtils.h

5.63 Arc::CountedPointer< T > Class Template Reference

Wrapper for pointer with automatic destruction and multiple references.

```
#include <Utils.h>
```

Data Structures

- class **Base**

Public Member Functions

- T & `operator*` (void) const
- T * `operator->` (void) const
- `operator bool` (void) const
- bool `operator!` (void) const
- `operator T *` (void) const
- T * `Release` (void)

5.63.1 Detailed Description

```
template<typename T> class Arc::CountedPointer<T>
```

Wrapper for pointer with automatic destruction and multiple references.

If ordinary pointer is wrapped in instance of this class it will be automatically destroyed when all instances referring to it are destroyed. This is useful for maintaining pointers referred from multiple structures with automatic destruction of original object when last reference is destroyed. It is similar to Java approach with a difference that destruction time is strictly defined. Only pointers returned by new() are supported. This class is not thread-safe

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

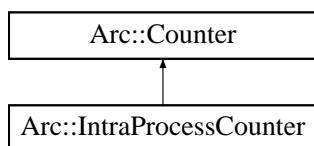
- `Utils.h`

5.64 Arc::Counter Class Reference

A class defining a common interface for counters.

```
#include <Counter.h>
```

Inheritance diagram for Arc::Counter:



Public Member Functions

- virtual `~Counter()`
- virtual int `getLimit()`=0
- virtual int `setLimit(int newLimit)=0`
- virtual int `changeLimit(int amount)=0`
- virtual int `getExcess()`=0

- virtual int [setExcess](#) (int newExcess)=0
- virtual int [changeExcess](#) (int amount)=0
- virtual int [getValue](#) ()=0
- virtual [CounterTicket reserve](#) (int amount=1, Glib::TimeVal duration=[ETERNAL](#), bool prioritized=false, Glib::TimeVal timeOut=[ETERNAL](#))=0

Protected Types

- [typedef unsigned long long int IDType](#)

Protected Member Functions

- [Counter](#) ()
- virtual void [cancel](#) ([IDType](#) reservationID)=0
- virtual void [extend](#) ([IDType](#) &reservationID, Glib::TimeVal &expiryTime, Glib::TimeVal duration=[ETERNAL](#))=0
- Glib::TimeVal [getCurrentTime](#) ()
- Glib::TimeVal [getExpiryTime](#) (Glib::TimeVal duration)
- [CounterTicket getCounterTicket](#) ([Counter::IDType](#) reservationID, Glib::TimeVal expiryTime, [Counter](#) *counter)
- [ExpirationReminder getExpirationReminder](#) (Glib::TimeVal expTime, [Counter::IDType](#) resID)

Friends

- class [CounterTicket](#)
- class [ExpirationReminder](#)

5.64.1 Detailed Description

A class defining a common interface for counters.

This class defines a common interface for counters as well as some common functionality.

The purpose of a counter is to provide housekeeping some resource such as e.g. disk space, memory or network bandwidth. The counter itself will not be aware of what kind of resource it limits the use of. Neither will it be aware of what unit is being used to measure that resource. Counters are thus very similar to semaphores. Furthermore, counters are designed to handle concurrent operations from multiple threads/processes in a consistent manner.

Every counter has a limit, an excess limit and a value. The limit is a number that specify how many units are available for reservation. The value is the number of units that are currently available for reservation, i.e. has not allready been reserved. The excess limit specify how many extra units can be reserved for high priority needs even if there are

no normal units available for reservation. The excess limit is similar to the credit limit of e.g. a VISA card.

The users of the resource must thus first call the counter in order to make a reservation of an appropriate amount of the resource, then allocate and use the resource and finally call the counter again to cancel the reservation.

Typical usage is:

```
// Declare a counter. Replace XYZ by some appropriate kind of
// counter and provide required parameters. Unit is MB.
XYZCounter memory(...);
...
// Make a reservation of memory for 2000000 doubles.
CounterTicket tick = memory.reserve(2*sizeof(double));
// Use the memory.
double* A=new double[2000000];
doSomething(A);
delete[] A;
// Cancel the reservation.
tick.cancel();
```

There are also alternative ways to make reservations, including self-expiring reservations, prioritized reservations and reservations that fail if they cannot be made fast enough.

For self expiring reservations, a duration is provided in the reserve call:

```
tick = memory.reserve(2*sizeof(double), Glib::TimeVal(1,0));
```

A self-expiring reservation can be cancelled explicitly before it expires, but if it is not cancelled it will expire automatically when the duration has passed. The default value for the duration is ETERNAL, which means that the reservation will not be cancelled automatically.

Prioritized reservations may use the excess limit and succeed immediately even if there are no normal units available for reservation. The value of the counter will in this case become negative. A prioritized reservation looks like this:

```
tick = memory.reserve(2*sizeof(double), Glib::TimeVal(1,0), true);
```

Finally, a time out option can be provided for a reservation. If some task should be performed within two seconds or not at all, the reservation can look like this:

```
tick = memory.reserve(2*sizeof(double), Glib::TimeVal(1,0),
                      true, Glib::TimeVal(2,0));
if (tick.isValid())
    doSomething(...);
```

5.64.2 Member Typedef Documentation

5.64.2.1 `typedef unsigned long long int Arc::Counter::IDType [protected]`

A typedef of identification numbers for reservation.

This is a type that is used as identification numbers (keys) for referencing of reservations. It is used internally in counters for book keeping of reservations as well as in the `CounterTicket` class in order to be able to cancel and extend reservations.

5.64.3 Constructor & Destructor Documentation

5.64.3.1 `Arc::Counter::Counter()` [protected]

Default constructor.

This is the default constructor. Since `Counter` is an abstract class, it should only be used by subclasses. Therefore it is protected. Furthermore, since the `Counter` class has no attributes, nothing needs to be initialized and thus this constructor is empty.

5.64.3.2 `virtual Arc::Counter::~Counter()` [virtual]

The destructor.

This is the destructor of the `Counter` class. Since the `Counter` class has no attributes, nothing needs to be cleaned up and thus the destructor is empty.

5.64.4 Member Function Documentation

5.64.4.1 `virtual void Arc::Counter::cancel(IDType reservationID)` [protected, pure virtual]

Cancellation of a reservation.

This method cancels a reservation. It is called by the `CounterTicket` that corresponds to the reservation.

Parameters

<code>reservationID</code>	The identity number (key) of the reservation to cancel.
----------------------------	---

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.2 `virtual int Arc::Counter::changeExcess(int amount)` [pure virtual]

Changes the excess limit of the counter.

Changes the excess limit of the counter by adding a certain amount to the current excess limit.

Parameters

<code>amount</code>	The amount by which to change the excess limit.
---------------------	---

Returns

The new excess limit.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.3 virtual int Arc::Counter::changeLimit(int *amount*) [pure virtual]

Changes the limit of the counter.

Changes the limit of the counter by adding a certain amount to the current limit.

Parameters

<i>amount</i>	The amount by which to change the limit.
---------------	--

Returns

The new limit.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.4 virtual void Arc::Counter::extend(IDType & *reservationID*, Glib::TimeVal & *expiryTime*, Glib::TimeVal *duration* = ETERNAL) [protected, pure virtual]

Extension of a reservation.

This method extends a reservation. It is called by the [CounterTicket](#) that corresponds to the reservation.

Parameters

<i>reservationID</i>	Used for input as well as output. Contains the identification number of the original reservation on entry and the new identification number of the extended reservation on exit.
<i>expiryTime</i>	Used for input as well as output. Contains the expiry time of the original reservation on entry and the new expiry time of the extended reservation on exit.
<i>duration</i>	The time by which to extend the reservation. The new expiration time is computed based on the current time, NOT the previous expiration time.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.5 CounterTicket Arc::Counter::getCounterTicket(Counter::IDType *reservationID*, Glib::TimeVal *expiryTime*, Counter * *counter*) [protected]

A "relay method" for a constructor of the [CounterTicket](#) class.

This method acts as a relay for one of the constructors of the [CounterTicket](#) class. That constructor is private, but needs to be accessible from the subclasses of [Counter](#) (but not from anywhere else). In order not to have to declare every possible subclass of [Counter](#) as a friend of [CounterTicket](#), only the base class [Counter](#) is a friend and its subclasses access the constructor through this method. (If C++ had supported "package access", as Java does, this trick would not have been necessary.)

Parameters

<i>reservationID</i>	The identity number of the reservation corresponding to the CounterTicket .
<i>expiryTime</i>	the expiry time of the reservation corresponding to the CounterTicket .
<i>counter</i>	The Counter from which the reservation has been made.

Returns

The counter ticket that has been created.

5.64.4.6 Glib::TimeVal Arc::Counter::getCurrentTime() [protected]

Get the current time.

Returns the current time. An "adapter method" for the `assign_current_time()` method in the [Glib::TimeVal](#) class. return The current time.

5.64.4.7 virtual int Arc::Counter::getExcess() [pure virtual]

Returns the excess limit of the counter.

Returns the excess limit of the counter, i.e. by how much the usual limit may be exceeded by prioritized reservations.

Returns

The excess limit.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.8 ExpirationReminder Arc::Counter::getExpirationReminder(Glib::TimeVal *expTime*, Counter::IDType *resID*) [protected]

A "relay method" for the constructor of [ExpirationReminder](#).

This method acts as a relay for one of the constructors of the [ExpirationReminder](#) class. That constructor is private, but needs to be accessible from the subclasses of [Counter](#) (but not from anywhere else). In order not to have to declare every possible subclass of [Counter](#) as a friend of [ExpirationReminder](#), only the base class [Counter](#) is a friend and its subclasses access the constructor through this method. (If C++ had supported "package access", as Java does, this trick would not have been necessary.)

Parameters

<i>expTime</i>	the expiry time of the reservation corresponding to the ExpirationReminder .
<i>resID</i>	The identity number of the reservation corresponding to the ExpirationReminder .

Returns

The [ExpirationReminder](#) that has been created.

5.64.4.9 Glib::TimeVal Arc::Counter::getExpiryTime (Glib::TimeVal *duration*)
[protected]

Computes an expiry time.

This method computes an expiry time by adding a duration to the current time.

Parameters

<i>duration</i>	The duration.
-----------------	---------------

Returns

The expiry time.

5.64.4.10 virtual int Arc::Counter::getLimit () [pure virtual]

Returns the current limit of the counter.

This method returns the current limit of the counter, i.e. how many units can be reserved simultaneously by different threads without claiming high priority.

Returns

The current limit of the counter.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.11 virtual int Arc::Counter::getValue () [pure virtual]

Returns the current value of the counter.

Returns the current value of the counter, i.e. the number of unreserved units. Initially, the value is equal to the limit of the counter. When a reservation is made, the the value is decreased. Normally, the value should never be negative, but this may happen if there are prioritized reservations. It can also happen if the limit is decreased after some reservations have been made, since reservations are never revoked.

Returns

The current value of the counter.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.12 virtual CounterTicket Arc::Counter::reserve (int *amount* = 1, Glib::TimeVal *duration* = ETERNAL, bool *prioritized* = false, Glib::TimeVal *timeOut* = ETERNAL) [pure virtual]

Makes a reservation from the counter.

This method makes a reservation from the counter. If the current value of the counter is too low to allow for the reservation, the method blocks until the reservation is possible or times out.

Parameters

<i>amount</i>	The amount to reserve, default value is 1.
<i>duration</i>	The duration of a self expiring reservation, default is that it lasts forever.
<i>prioritized</i>	Whether this reservation is prioritized and thus allowed to use the excess limit.
<i>timeOut</i>	The maximum time to block if the value of the counter is too low, default is to allow "eternal" blocking.

Returns

A [CounterTicket](#) that can be queried about the status of the reservation as well as for cancellations and extensions.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.13 virtual int Arc::Counter::setExcess (int *newExcess*) [pure virtual]

Sets the excess limit of the counter.

This method sets a new excess limit for the counter.

Parameters

<i>newExcess</i>	The new excess limit, an absolute number.
------------------	---

Returns

The new excess limit.

Implemented in [Arc::IntraProcessCounter](#).

5.64.4.14 virtual int Arc::Counter::setLimit (int *newLimit*) [pure virtual]

Sets the limit of the counter.

This method sets a new limit for the counter.

Parameters

<i>newLimit</i>	The new limit, an absolute number.
-----------------	------------------------------------

Returns

The new limit.

Implemented in [Arc::IntraProcessCounter](#).

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

- Counter.h

5.65 Arc::CounterTicket Class Reference

A class for "tickets" that correspond to counter reservations.

```
#include <Counter.h>
```

Public Member Functions

- [CounterTicket \(\)](#)
- [bool isValid \(\)](#)
- [void extend \(Glib::TimeVal duration\)](#)
- [void cancel \(\)](#)

Friends

- [class Counter](#)

5.65.1 Detailed Description

A class for "tickets" that correspond to counter reservations.

This is a class for reservation tickets. When a reservation is made from a [Counter](#), a [ReservationTicket](#) is returned. This ticket can then be queried about the validity of a reservation. It can also be used for cancelation and extension of reservations.

Typical usage is:

```
// Declare a counter. Replace XYZ by some appropriate kind of
// counter and provide required parameters. Unit is MB.
XYZCounter memory(...);
...
// Make a reservation of memory for 2000000 doubles.
CounterTicket tick = memory.reserve(2*sizeof(double));
// Use the memory.
double* A=new double[2000000];
doSomething(A);
delete[] A;
// Cancel the reservation.
tick.cancel();
```

5.65.2 Constructor & Destructor Documentation

5.65.2.1 `Arc::CounterTicket::CounterTicket()`

The default constructor.

This is the default constructor. It creates a `CounterTicket` that is not valid. The ticket object that is created can later be assigned a ticket that is returned by the `reserve()` method of a `Counter`.

5.65.3 Member Function Documentation

5.65.3.1 `void Arc::CounterTicket::cancel()`

Cancels a reservation.

This method is called to cancel a reservation. It may be called also for self-expiring reservations, which will then be cancelled before they were originally planned to expire.

5.65.3.2 `void Arc::CounterTicket::extend (Glib::TimeVal duration)`

Extends a reservation.

Extends a self-expiring reservation. In order to succeed the extension should be made before the previous reservation expires.

Parameters

<code>duration</code>	The time by which to extend the reservation. The new expiration time is computed based on the current time, NOT the previous expiration time.
-----------------------	---

5.65.3.3 `bool Arc::CounterTicket::isValid()`

Returns the validity of a `CounterTicket`.

This method checks whether a `CounterTicket` is valid. The ticket was probably returned earlier by the `reserve()` method of a `Counter` but the corresponding reservation may have expired.

Returns

The validity of the ticket.

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

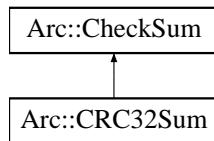
- Counter.h

5.66 Arc::CRC32Sum Class Reference

Implementation of CRC32 checksum.

```
#include <CheckSum.h>
```

Inheritance diagram for Arc::CRC32Sum:



Public Member Functions

- virtual void [start](#) (void)
- virtual void [add](#) (void *buf, unsigned long long int len)
- virtual void [end](#) (void)
- virtual void [result](#) (unsigned char *&res, unsigned int &len) const
- virtual int [print](#) (char *buf, int len) const
- virtual void [scan](#) (const char *buf)
- virtual [operator bool](#) (void) const
- virtual bool [operator!](#) (void) const

5.66.1 Detailed Description

Implementation of CRC32 checksum.

This class is a specialized class of the [CheckSum](#) class. It provides an implementation for the CRC-32 IEEE 802.3 standard.

5.66.2 Member Function Documentation

5.66.2.1 virtual void Arc::CRC32Sum::add (void * *buf*, unsigned long long int *len*) [virtual]

Add data to be checksummed.

This method calculates the checksum of the passed data chuck, taking into account the previous state of this object.

Parameters

<i>buf</i>	pointer to data chuck to be checksummed.
<i>len</i>	size of the data chuck.

Implements [Arc::CheckSum](#).

5.66.2.2 virtual void Arc::CRC32Sum::end(void) [virtual]

Finalize the checksumming.

This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::CheckSum](#).

5.66.2.3 virtual int Arc::CRC32Sum::print(char * buf, int len) const [virtual]

Retrieve result of checksum into a string.

The passed string buf is filled with result of checksum algorithm in base 16. At most len characters is filled into buffer buf. The hexadecimal value is prepended with "<algorithm>:", where <algorithm> is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters

<i>buf</i>	pointer to buffer which should be filled with checksum result.
<i>len</i>	max number of character filled into buffer.

Reimplemented from [Arc::CheckSum](#).

5.66.2.4 virtual void Arc::CRC32Sum::scan(const char * buf) [virtual]

Set internal checksum state.

This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [CheckSum::print](#) method.

Parameters

<i>buf</i>	string containing textual representation of checksum
------------	--

See also

[CheckSum::print](#)

Implements [Arc::CheckSum](#).

5.66.2.5 virtual void Arc::CRC32Sum::start(void) [virtual]

Initiate the checksum algorithm.

This method must be called before starting a new checksum calculation.

Implements [Arc::CheckSum](#).

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

- [CheckSum.h](#)

5.67 Arc::Credential Class Reference

```
#include <Credential.h>
```

Public Member Functions

- [Credential \(\)](#)
- [Credential \(int keybits\)](#)
- [Credential \(const std::string &CAfile, const std::string &CAkey, const std::string &CAserial, const std::string &extfile, const std::string &extsect, const std::string &passphrase4key\)](#)
- [Credential \(Time start, Period lifetime=Period\("PT12H"\), int keybits=1024, std::string proxyversion="rfc", std::string policylang="inheritAll", std::string policy="", int pathlength=-1\)](#)
- [Credential \(const std::string &cert, const std::string &key, const std::string &cadir, const std::string &cafile, const std::string &passphrase4key="", const bool is_file=true\)](#)
- [Credential \(const UserConfig &usercfg, const std::string &passphrase4key ""\)](#)
- void [AddCertExtObj \(std::string &sn, std::string &oid\)](#)
- void [LogError \(void\) const](#)
- bool [GetVerification \(void\) const](#)
- EVP_PKEY * [GetPrivKey \(void\) const](#)
- EVP_PKEY * [GetPubKey \(void\) const](#)
- X509 * [GetCert \(void\) const](#)
- X509_REQ * [GetCertReq \(void\) const](#)
- STACK_OF (X509)*[GetCertChain\(void\) const](#)
- int [GetCertNumofChain \(void\) const](#)
- Credformat [getFormat \(BIO *in, const bool is_file=true\) const](#)
- std::string [GetDN \(void\) const](#)
- std::string [GetIdentityName \(void\) const](#)
- ArcCredential::certType [GetType \(void\) const](#)
- std::string [GetIssuerName \(void\) const](#)
- std::string [GetCAName \(void\) const](#)
- std::string [GetProxyPolicy \(void\) const](#)
- void [SetProxyPolicy \(const std::string &proxyversion, const std::string &policylang, const std::string &policy, int pathlength\)](#)
- bool [OutputPrivatekey \(std::string &content, bool encryption=false, const std::string &passphrase ""\)](#)
- bool [OutputPublickey \(std::string &content\)](#)
- bool [OutputCertificate \(std::string &content, bool is_der=false\)](#)

- `bool OutputCertificateChain (std::string &content, bool is_der=false)`
- `Period GetLifeTime (void) const`
- `Time GetStartTime () const`
- `Time GetEndTime () const`
- `void SetLifeTime (const Period &period)`
- `void SetStartTime (const Time &start_time)`
- `bool IsValid (void)`
- `bool AddExtension (const std::string &name, const std::string &data, bool crit=false)`
- `bool AddExtension (const std::string &name, char **binary)`
- `std::string GetExtension (const std::string &name)`
- `bool GenerateEECRequest (BIO *reqbio, BIO *keybio, const std::string &dn "")`
- `bool GenerateEECRequest (std::string &reqcontent, std::string &keycontent, const std::string &dn "")`
- `bool GenerateEECRequest (const char *request_filename, const char *key_filename, const std::string &dn "")`
- `bool GenerateRequest (BIO *bio, bool if_der=false)`
- `bool GenerateRequest (std::string &content, bool if_der=false)`
- `bool GenerateRequest (const char *filename, bool if_der=false)`
- `bool InquireRequest (BIO *reqbio, bool if_eec=false, bool if_der=false)`
- `bool InquireRequest (std::string &content, bool if_eec=false, bool if_der=false)`
- `bool InquireRequest (const char *filename, bool if_eec=false, bool if_der=false)`
- `bool SignRequest (Credential *proxy, BIO *outputbio, bool if_der=false)`
- `bool SignRequest (Credential *proxy, std::string &content, bool if_der=false)`
- `bool SignRequest (Credential *proxy, const char *filename, bool foamat=false)`
- `bool SelfSignEECRequest (const std::string &dn, const char *extfile, const std::string &extsect, const char *certfile)`
- `bool SignEECRequest (Credential *eec, const std::string &dn, BIO *outputbio)`
- `bool SignEECRequest (Credential *eec, const std::string &dn, std::string &content)`
- `bool SignEECRequest (Credential *eec, const std::string &dn, const char *filename)`

Static Public Member Functions

- `static void InitProxyCertInfo (void)`
- `static bool IsCredentialsValid (const UserConfig &usercfg)`

5.67.1 Detailed Description

`Credential` class covers the functionality about general processing about certificate/key files, including: 1. certificate/key parsing, information extracting (such as subject name, issuer name, lifetime, etc.), chain verifying, extension processing about proxy certinfo, extension processing about other general certificate extension (such as voms attributes, it should be the extension-specific code itself to create, parse and verify the extension, not the `Credential` class. For voms, it is some code about writing and parsing voms-implementing Attribute Certificate/ RFC3281, the voms-attribute is then be looked as a

binary part and embeded into extension of X509 certificate/proxy certificate); 2. certificate request, extension emeding and certificate signing, for both proxy certificate and EEC (end entity certificate) certificate The [Credential](#) class support PEM, DER PKCS12 credential.

5.67.2 Constructor & Destructor Documentation

5.67.2.1 Arc::Credential::Credential ()

Default constructor, only acts as a container for inquiring certificate request, is meaningless for any other use.

5.67.2.2 Arc::Credential::Credential (int *keybits*)

Constructor with user-defined keylength. Needed for creation of EE certs, since some applications will only support keys with a certain minimum length > 1024

5.67.2.3 Arc::Credential::Credential (const std::string & *CAfile*, const std::string & *CAkey*, const std::string & *CAserial*, const std::string & *extfile*, const std::string & *extsect*, const std::string & *passphrase4key*)

Constructor, specific constructor for CA certificate is meaningless for any other use.

5.67.2.4 Arc::Credential::Credential (Time *start*, Period *lifetime* = Period ("PT12H") , int *keybits* = 1024, std::string *proxyversion* = "rfc", std::string *policylang* = "inheritAll", std::string *policy* = " ", int *pathlength* = -1)

Constructor, specific constructor for proxy certificate, only acts as a container for constraining certificate signing and/or generating certificate request(only keybits is useful for creating certificate request), is meaningless for any other use. The proxyversion and policylang is for specifying the proxy certificate type and the policy language inside proxy. The definition of proxyversion and policy language is based on http://dev.globus.org/wiki/Security/Proxy3820_Proxy_Certificates The code is supposed to support proxy version: GSI2(legacy proxy), GSI3(Proxy draft) and RFC(RFC3820 proxy), and correspoding policy language. GSI2(GSI2, GSI2_LIMITED) GSI3 and RFC (IMPERSONATION_PROXY--1.3.6.1.5.5.7.21.1, INDEPENDENT_PROXY--1.3.6.1.5.5.7.21.2, LIMITED_PROXY--1.3.6.1.4.1.3536.1.1.1.9, RESTRICTED_PROXY--policy language undefined) In openssl>=098, there are three types of policy languages: id-ppl-inheritAll--1.3.6.1.5.5.7.21.1, id-ppl-independent--1.3.6.1.5.5.7.21.2, and id-ppl-anyLanguage-1.3.6.1.5.5.7.21.0

Parameters

<i>start,start</i>	time of proxy certificate
<i>life-time,lifetime</i>	of proxy certificate

<i>key-bits, modulus</i>	size for RSA key generation, it should be greater than 1024 if 'this' class is used for generating X509 request; it should be '0' if 'this' class is used for constraining certificate signing.
--------------------------	---

5.67.2.5 `Arc::Credential::Credential (const std::string & cert, const std::string & key, const std::string & cadir, const std::string & cafile, const std::string & passphrase4key = "", const bool is_file = true)`

Constructor, specific constructor for usual certificate, constructing from credential files. only acts as a container for parsing the certificate and key files, is meaningless for any other use. this constructor will parse the credential information, and put them into "this" object

Parameters

<i>passphrase4key</i>	the password for decrypting private key (if needed). If value is empty then password will be asked interactively. To avoid asking for password use value provided by NoPassword() method.
<i>is_file, specifies</i>	if the cert/key are from file, otherwise they are supposed to be from string. default is from file

5.67.2.6 `Arc::Credential::Credential (const UserConfig & usercfg, const std::string & passphrase4key = "")`

Constructor, specific constructor for usual certificate, constructing from information in `UserConfig` object. Only acts as a container for parsing the certificate and key files, is meaningless for any other use. this constructor will parse the credential * information, and put them into "this" object

Parameters

<i>is_file, specify</i>	if the cert/key are from file, otherwise they are supposed to be from string. default is from file
-------------------------	--

5.67.3 Member Function Documentation

5.67.3.1 `void Arc::Credential::AddCertExtObj (std::string & sn, std::string & oid)`

General method for adding a new nid into openssl's global const

5.67.3.2 `bool Arc::Credential::AddExtension (const std::string & name, char ** binary)`

Add an extension to the extension part of the certificate

Parameters

<i>binary,the</i>	data which will be inserted into certificate extension part as a specific extension there should be specific methods defined inside specific X509V3_EXT_METHOD structure to parse the specific extension format. For example, VOMS attribute certificate is a specific extension to proxy certificate. There is specific X509V3_EXT_METHOD defined in VOMSAttribute.h and VOMSAttribute.c for parsing attribute certificate. In openssl, the specific X509V3_EXT_METHOD can be got according to the extension name/id, see X509V3_EXT_get_nid(ext_nid)
-------------------	--

5.67.3.3 `bool Arc::Credential::AddExtension (const std::string & name, const std::string & data, bool crit = false)`

Add an extension to the extension part of the certificate

Parameters

<i>name,the</i>	name of the extension, there OID related with the name should be registered into openssl firstly
<i>data,the</i>	data which will be inserted into certificate extension

5.67.3.4 `bool Arc::Credential::GenerateEECRequest (BIO * reqbio, BIO * keybio, const std::string & dn = " ")`

Generate an EEC request, based on the keybits and signing algorithm information inside this object output the certificate request to output BIO

The user will be asked for a private key password

5.67.3.5 `bool Arc::Credential::GenerateEECRequest (std::string & reqcontent, std::string & keycontent, const std::string & dn = " ")`

Generate an EEC request, output the certificate request to a string

5.67.3.6 `bool Arc::Credential::GenerateEECRequest (const char * request_filename, const char * key_filename, const std::string & dn = " ")`

Generate an EEC request, output the certificate request and the key to a file

5.67.3.7 `bool Arc::Credential::GenerateRequest (BIO * bio, bool if_der = false)`

Generate a proxy request, base on the keybits and signing algorithm information inside this object output the certificate request to output BIO

5.67.3.8 bool Arc::Credential::GenerateRequest (std::string & *content*, bool *if_der = false*)

Generate a proxy request, output the certificate request to a string

5.67.3.9 bool Arc::Credential::GenerateRequest (const char * *filename*, bool *if_der = false*)

Generate a proxy request, output the certificate request to a file

5.67.3.10 std::string Arc::Credential::GetCName (void) const

Get CA of the certificate attached to this object, if the certificate is an EEC, GetCName get the same value as GetIssuerName

5.67.3.11 X509* Arc::Credential::GetCert (void) const

Get the certificate attached to this object

5.67.3.12 int Arc::Credential::GetCertNumofChain (void) const

Get the number of certificates in the certificate chain attached to this object

5.67.3.13 X509_REQ* Arc::Credential::GetCertReq (void) const

Get the certificate request, if there is any

5.67.3.14 std::string Arc::Credential::GetDN (void) const

Get the DN of the certificate attached to this object

5.67.3.15 Time Arc::Credential::GetEndTime () const

Returns validity end time of certificate or proxy

5.67.3.16 std::string Arc::Credential::GetExtension (const std::string & *name*)

Get the specific extension (named by the parameter) in a certificate this function is only supposed to be called after certificate and key are loaded by the constructor for usual certificate

Parameters

<i>name</i> , the	name of the extension to get
-------------------	------------------------------

5.67.3.17 Credformat `Arc::Credential::getFormat(BIO * in, const bool is_file = true) const`

Get the certificate format, PEM PKCS12 or DER BIO could be memory or file, they should be processed differently.

5.67.3.18 std::string `Arc::Credential::GetIdentityName(void) const`

Get the Identity name of the certificate attached to this object, the result will not include proxy CN

5.67.3.19 std::string `Arc::Credential::GetIssuerName(void) const`

Get issuer of the certificate attached to this object

5.67.3.20 Period `Arc::Credential::GetLifeTime(void) const`

Returns lifetime of certificate or proxy

5.67.3.21 EVP_PKEY* `Arc::Credential::GetPrivKey(void) const`

Get the private key attached to this object

5.67.3.22 std::string `Arc::Credential::GetProxyPolicy(void) const`

Get the proxy policy attached to the "proxy certificate information" extension of the proxy certificate

5.67.3.23 EVP_PKEY* `Arc::Credential::GetPubKey(void) const`

Get the public key attached to this object

5.67.3.24 Time `Arc::Credential::GetStartTime() const`

Returns validity start time of certificate or proxy

5.67.3.25 ArcCredential::certType `Arc::Credential::GetType(void) const`

Get type of the certificate attached to this object

5.67.3.26 bool `Arc::Credential::GetVerification(void) const [inline]`

Get the verification result about certificate chain checking

5.67.3.27 `static void Arc::Credential::InitProxyCertInfo (void) [static]`

Initiate nid for proxy certificate extension

5.67.3.28 `bool Arc::Credential::InquireRequest (BIO * reqbio, bool if_eec = false, bool if_der = false)`

Inquire the certificate request from BIO, and put the request information to X509_REQ inside this object, and parse the certificate type from the PROXYCERTINFO of request' extension

Parameters

<code><i>if_der</i></code>	false for PEM; true for DER
----------------------------	-----------------------------

5.67.3.29 `bool Arc::Credential::InquireRequest (std::string & content, bool if_eec = false, bool if_der = false)`

Inquire the certificate request from a string

5.67.3.30 `bool Arc::Credential::InquireRequest (const char * filename, bool if_eec = false, bool if_der = false)`

Inquire the certificate request from a file

5.67.3.31 `static bool Arc::Credential::IsCredentialsValid (const UserConfig & usercfg) [static]`

Returns true if credentials are valid. Credentials are read from locations specified in `UserConfig` object. This method is deprecated. [User](#) per-instance method `IsValid()` instead.

5.67.3.32 `bool Arc::Credential::IsValid (void)`

Returns true if credentials are valid

5.67.3.33 `void Arc::Credential::LogError (void) const`

Log error information related with openssl

5.67.3.34 `bool Arc::Credential::OutputCertificate (std::string & content, bool is_der = false)`

Output the certificate into string

Parameters

<i>is_der</i>	false for PEM, true for DER
---------------	-----------------------------

5.67.3.35 **bool Arc::Credential::OutputCertificateChain (std::string & *content*, bool *is_der* = false)**

Output the certificate chain into string

Parameters

<i>is_der</i>	false for PEM, true for DER
---------------	-----------------------------

5.67.3.36 **bool Arc::Credential::OutputPrivatekey (std::string & *content*, bool *encryption* = false, const std::string & *passphrase* = " ")**

Output the private key into string

Parameters

<i>encryption,whether</i>	encrypt the output private key or not
<i>passphrase,th</i>	passphrase to encrypt the output private key

5.67.3.37 **bool Arc::Credential::OutputPublickey (std::string & *content*)**

Output the public key into string

5.67.3.38 **bool Arc::Credential::SelfSignEECRequest (const std::string & *dn*, const char * *extfile*, const std::string & *extsect*, const char * *certfile*)**

Self sign a certificate. This functionality is specific for creating a CA credential by using this [Credential](#) class.

Parameters

<i>dn</i>	the DN for the subject
<i>extfile</i>	the configuration file which includes the extension information, typically the openssl.cnf file
<i>extsect</i>	the section/group name for the extension, e.g. in openssl.cnf, usr_cert and v3_ca
<i>certfile</i>	the certificate file, which contains the signed certificate

5.67.3.39 void Arc::Credential::SetLifeTime (const Period & period)

Set lifetime of certificate or proxy

5.67.3.40 void Arc::Credential::SetProxyPolicy (const std::string & proxyversion, const std::string & policylang, const std::string & policy, int pathlength)

Set the proxy policy attached to the "proxy certificate information" extension of the proxy certificate

5.67.3.41 void Arc::Credential::SetStartTime (const Time & start_time)

Set start time of certificate or proxy

5.67.3.42 bool Arc::Credential::SignEECRequest (Credential * eec, const std::string & dn, const char * filename)

Sign request and output the signed certificate to a file

5.67.3.43 bool Arc::Credential::SignEECRequest (Credential * eec, const std::string & dn, std::string & content)

Sign request and output the signed certificate to a string

5.67.3.44 bool Arc::Credential::SignEECRequest (Credential * eec, const std::string & dn, BIO * outputbio)

Sign eec request, and output the signed certificate to output BIO

5.67.3.45 bool Arc::Credential::SignRequest (Credential * proxy, const char * filename, bool foamat = false)

Sign request and output the signed certificate to a file

Parameters

<i>if_der</i>	false for PEM, true for DER
---------------	-----------------------------

5.67.3.46 bool Arc::Credential::SignRequest (Credential * proxy, BIO * outputbio, bool if_der = false)

Sign request based on the information inside proxy, and output the signed certificate to output BIO

Parameters

<i>if_der</i>	false for PEM, true for DER
---------------	-----------------------------

5.67.3.47 `bool Arc::Credential::SignRequest (Credential * proxy, std::string & content, bool if_der = false)`

Sign request and output the signed certificate to a string

Parameters

<i>if_der</i>	false for PEM, true for DER
---------------	-----------------------------

5.67.3.48 `Arc::Credential::STACK_OF (X509) const`

Get the certificate chain attached to this object

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

- `Credential.h`

5.68 Arc::CredentialError Class Reference

```
#include <Credential.h>
```

Public Member Functions

- `CredentialError (const std::string &what="")`

5.68.1 Detailed Description

This is an exception class that is used to handle runtime errors discovered in the [Credential](#) class.

5.68.2 Constructor & Destructor Documentation

5.68.2.1 `Arc::CredentialError::CredentialError (const std::string & what = " ")`

This is the constructor of the [CredentialError](#) class.

Parameters

<i>what</i>	An explanation of the error.
-------------	------------------------------

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

- Credential.h

5.69 Arc::CredentialStore Class Reference

```
#include <CredentialStore.h>
```

5.69.1 Detailed Description

This class provides functionality for storing delegated credentials and retrieving them from some store services. This is very preliminary implementation and currently support only one type of credentials - X.509 proxies, and only one type of store service - MyProxy. Later it will be extended to support at least following services: ARC delegation service, VOMS service, local file system.

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

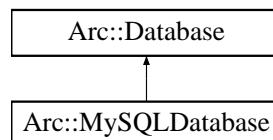
- CredentialStore.h

5.70 Arc::Database Class Reference

Interface for calling database client library.

```
#include <DBInterface.h>
```

Inheritance diagram for Arc::Database:



Public Member Functions

- [Database \(\)](#)
- [Database \(std::string &server, int port\)](#)
- [Database \(const Database &other\)](#)
- [virtual ~Database \(\)](#)
- [virtual bool connect \(std::string &dbname, std::string &user, std::string &password\)=0](#)
- [virtual bool isconnected \(\) const =0](#)
- [virtual void close \(\)=0](#)
- [virtual bool enable_ssl \(const std::string &keyfile="", const std::string &certfile="", const std::string &cafile="", const std::string &capath ""\)=0](#)
- [virtual bool shutdown \(\)=0](#)

5.70.1 Detailed Description

Interface for calling database client library.

For different types of database client library, different classes should be implemented by implementing this interface.

5.70.2 Constructor & Destructor Documentation

5.70.2.1 `Arc::Database::Database() [inline]`

Default constructor

5.70.2.2 `Arc::Database::Database(std::string & server, int port) [inline]`

Constructor which uses the server's name(or IP address) and port as parameters

5.70.2.3 `Arc::Database::Database(const Database & other) [inline]`

Copy constructor

5.70.2.4 `virtual Arc::Database::~Database() [inline, virtual]`

Deconstructor

5.70.3 Member Function Documentation

5.70.3.1 `virtual void Arc::Database::close() [pure virtual]`

Close the connection with database server

Implemented in [Arc::MySQLDatabase](#).

5.70.3.2 `virtual bool Arc::Database::connect(std::string & dbname, std::string & user, std::string & password) [pure virtual]`

Do connection with database server

Parameters

<code>dbname</code>	The database name which will be used.
<code>user</code>	The username which will be used to access database.
<code>password</code>	The password which will be used to access database.

Implemented in [Arc::MySQLDatabase](#).

5.70.3.3 virtual bool Arc::Database::enable_ssl (const std::string & *keyfile* = " ", const std::string & *certfile* = " ", const std::string & *cafile* = " ", const std::string & *capath* = " ") [pure virtual]

Enable ssl communication for the connection

Parameters

<i>keyfile</i>	The location of key file.
<i>certfile</i>	The location of certificate file.
<i>cafile</i>	The location of ca file.
<i>capath</i>	The location of ca directory

Implemented in [Arc::MySQLDatabase](#).

5.70.3.4 virtual bool Arc::Database::isconnected () const [pure virtual]

Get the connection status

Implemented in [Arc::MySQLDatabase](#).

5.70.3.5 virtual bool Arc::Database::shutdown () [pure virtual]

Ask database server to shutdown

Implemented in [Arc::MySQLDatabase](#).

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

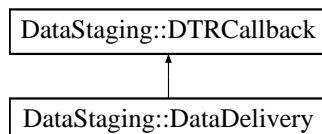
- DBInterface.h

5.71 DataStaging::DataDelivery Class Reference

[DataDelivery](#) transfers data between specified physical locations.

```
#include <DataDelivery.h>
```

Inheritance diagram for DataStaging::DataDelivery:



Public Member Functions

- [DataDelivery \(\)](#)

- `~DataDelivery ()`
- `virtual void receiveDTR (DTR &)`
- `bool cancelDTR (DTR *)`
- `bool start ()`
- `bool stop ()`
- `void SetTransferParameters (const TransferParameters ¶ms)`

5.71.1 Detailed Description

`DataDelivery` transfers data between specified physical locations.

All meta-operations for a `DTR` such as resolving replicas must be done before sending to `DataDelivery`. Calling `receiveDTR()` starts a new process which performs data transfer as specified in `DTR`.

5.71.2 Member Function Documentation

5.71.2.1 `virtual void DataStaging::DataDelivery::receiveDTR (DTR &) [virtual]`

Pass a `DTR` to Delivery.

This method is called by the scheduler to pass a `DTR` to the delivery. The `DataDelivery` starts a process to do the processing, and then returns. `DataDelivery`'s own thread then monitors the started process.

Implements `DataStaging::DTRCallback`.

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

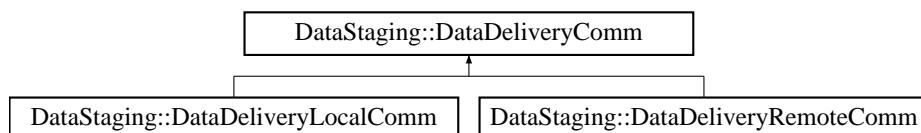
- `DataDelivery.h`

5.72 DataStaging::DataDeliveryComm Class Reference

This class provides an abstract interface for the Delivery layer.

```
#include <DataDeliveryComm.h>
```

Inheritance diagram for DataStaging::DataDeliveryComm:



Data Structures

- struct `Status`

Plain C struct to pass information from executing process back to main thread.

Public Types

- enum `CommStatusType` {
 `CommInit`, `CommNoError`, `CommTimeout`, `CommClosed`,
 `CommExited`, `CommFailed` }

Public Member Functions

- virtual `~DataDeliveryComm` ()
- `Status GetStatus` () const
- `std::string GetError` () const
- virtual `operator bool` () const =0
- virtual `bool operator!` () const =0

Static Public Member Functions

- static `DataDeliveryComm * CreateInstance` (const `DTR` &`dtr`, const `TransferParameters` &`params`)

Protected Member Functions

- virtual void `PullStatus` ()=0
- `DataDeliveryComm` (const `DTR` &`dtr`, const `TransferParameters` &`params`)

Protected Attributes

- `Status status_`
- `Status status_buf_`
- `unsigned int status_pos_`
- `Glib::Mutex lock_`
- `DataDeliveryCommHandler * handler_`
- `std::string dtr_id`
- `TransferParameters transfer_params`
- `Arc::Time start_`
- `Arc::Logger * logger_`

5.72.1 Detailed Description

This class provides an abstract interface for the Delivery layer.

Different implementations provide different ways of providing Delivery functionality. [DataDeliveryLocalComm](#) launches a local process to perform the transfer and [DataDeliveryRemoteComm](#) contacts a remote service which performs the transfer. The implementation is chosen depending on what is set in the [DTR](#), which the [Scheduler](#) should set based on various factors.

[CreateInstance\(\)](#) should be used to get a pointer to the instantiated object. This also starts the transfer. Deleting this object stops the transfer and cleans up any used resources. A singleton instance of [DataDeliveryCommHandler](#) regularly polls all active transfers using [PullStatus\(\)](#) and fills the [Status](#) object with current information, which can be obtained through [GetStatus\(\)](#).

5.72.2 Member Enumeration Documentation

5.72.2.1 enum DataStaging::DataDeliveryComm::CommStatusType

Communication status with transfer.

Enumerator:

- CommInit** Initializing/starting transfer, rest of information not valid.
- CommNoError** Communication going on smoothly.
- CommTimeout** Communication experienced timeout.
- CommClosed** Communication channel was closed.
- CommExited** Transfer exited. Mostly same as CommClosed but exit detected before pipe closed.
- CommFailed** Transfer failed. If we have CommFailed and no error code reported that normally means segfault or external kill.

5.72.3 Constructor & Destructor Documentation

5.72.3.1 DataStaging::DataDeliveryComm::DataDeliveryComm (const DTR & dtr, const TransferParameters & params) [protected]

Start transfer with parameters taken from [DTR](#) and supplied transfer limits.

Constructor should not be used directly, [CreateInstance\(\)](#) should be used instead.

5.72.4 Member Function Documentation

5.72.4.1 virtual void DataStaging::DataDeliveryComm::PullStatus () [protected, pure virtual]

Check for new state and fill state accordingly.

This method is periodically called by the comm handler to obtain status info. It detects communication and delivery failures and delivery termination.

Implemented in [DataStaging::DataDeliveryLocalComm](#), and [DataStaging::DataDeliveryRemoteComm](#).

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

- [DataDeliveryComm.h](#)

5.73 DataStaging::DataDeliveryCommHandler Class Reference

Singleton class handling all active [DataDeliveryComm](#) objects.

```
#include <DataDeliveryComm.h>
```

Public Member Functions

- void [Add \(DataDeliveryComm *item\)](#)
- void [Remove \(DataDeliveryComm *item\)](#)

Static Public Member Functions

- static [DataDeliveryCommHandler * getInstance \(\)](#)

5.73.1 Detailed Description

Singleton class handling all active [DataDeliveryComm](#) objects.

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

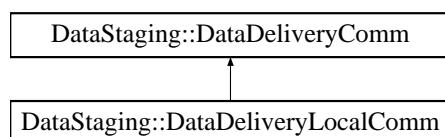
- [DataDeliveryComm.h](#)

5.74 DataStaging::DataDeliveryLocalComm Class Reference

This class starts, monitors and controls a local Delivery process.

```
#include <DataDeliveryLocalComm.h>
```

Inheritance diagram for DataStaging::DataDeliveryLocalComm:



Public Member Functions

- `DataDeliveryLocalComm (const DTR &dtr, const TransferParameters ¶ms)`
- `~DataDeliveryLocalComm ()`
- `virtual void PullStatus ()`
- `virtual operator bool () const`
- `virtual bool operator! () const`

5.74.1 Detailed Description

This class starts, monitors and controls a local Delivery process.

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

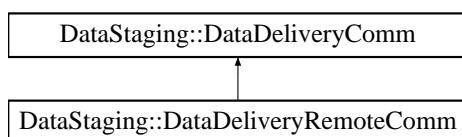
- `DataDeliveryLocalComm.h`

5.75 DataStaging::DataDeliveryRemoteComm Class Reference

This class contacts a remote service to make a Delivery request.

```
#include <DataDeliveryRemoteComm.h>
```

Inheritance diagram for DataStaging::DataDeliveryRemoteComm:



Public Member Functions

- `virtual void PullStatus ()`
- `virtual operator bool () const`
- `virtual bool operator! () const`

5.75.1 Detailed Description

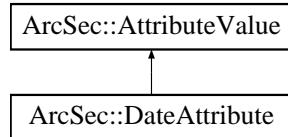
This class contacts a remote service to make a Delivery request.

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

- `DataDeliveryRemoteComm.h`

5.76 ArcSec::DateAttribute Class Reference

Inheritance diagram for ArcSec::DateAttribute:



Public Member Functions

- virtual bool [equal](#) ([AttributeValue](#) *other, bool check_id=true)
- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

5.76.1 Member Function Documentation

5.76.1.1 virtual std::string [ArcSec::DateAttribute::encode](#) () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.76.1.2 virtual bool [ArcSec::DateAttribute::equal](#) ([AttributeValue](#) * value, bool check_id = true) [virtual]

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.76.1.3 virtual std::string [ArcSec::DateAttribute::getId](#) () [inline, virtual]

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.76.1.4 virtual std::string [ArcSec::DateAttribute::getType](#) () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

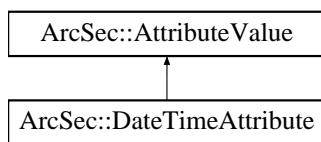
The documentation for this class was generated from the following file:

- [DateTimeAttribute.h](#)

5.77 ArcSec::DateTimeAttribute Class Reference

```
#include <DateTimeAttribute.h>
```

Inheritance diagram for ArcSec::DateTimeAttribute:



Public Member Functions

- virtual bool [equal \(AttributeValue *other, bool check_id=true\)](#)
- virtual std::string [encode \(\)](#)
- virtual std::string [getType \(\)](#)
- virtual std::string [getId \(\)](#)

5.77.1 Detailed Description

Format: YYYYMMDDHHMMSSZ Day Month DD HH:MM:SS YYYY YYYY-MM-DD HH:MM:SS
YYYY-MM-DDTHH:MM:SS+HH:MM YYYY-MM-DDTHH:MM:SSZ

5.77.2 Member Function Documentation

5.77.2.1 virtual std::string ArcSec::DateTimeAttribute::encode () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.77.2.2 virtual bool ArcSec::DateTimeAttribute::equal (AttributeValue * value, bool check_id = true) [virtual]

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.77.2.3 virtual std::string ArcSec::DateTimeAttribute::getId () [inline, virtual]

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.77.2.4 virtual std::string ArcSec::DateTimeAttribute::getType() [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

- [DateTimeAttribute.h](#)

5.78 Arc::DBranch Class Reference

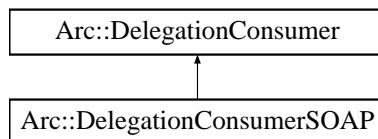
The documentation for this class was generated from the following file:

- [DBranch.h](#)

5.79 Arc::DelegationConsumer Class Reference

#include <DelegationInterface.h>

Inheritance diagram for Arc::DelegationConsumer:



Public Member Functions

- [DelegationConsumer \(void\)](#)
- [DelegationConsumer \(const std::string &content\)](#)
- [const std::string & ID \(void\)](#)
- [bool Backup \(std::string &content\)](#)
- [bool Restore \(const std::string &content\)](#)
- [bool Request \(std::string &content\)](#)
- [bool Acquire \(std::string &content\)](#)
- [bool Acquire \(std::string &content, std::string &identity\)](#)

Protected Member Functions

- [bool Generate \(void\)](#)
- [void LogError \(void\)](#)

5.79.1 Detailed Description

A consumer of delegated X509 credentials. During delegation procedure this class acquires delegated credentials aka proxy - certificate, private key and chain of previous certificates. Delegation procedure consists of calling [Request\(\)](#) method for generating certificate request followed by call to [Acquire\(\)](#) method for making complete credentials from certificate chain.

5.79.2 Constructor & Destructor Documentation

5.79.2.1 Arc::DelegationConsumer::DelegationConsumer (void)

Creates object with new private key

5.79.2.2 Arc::DelegationConsumer::DelegationConsumer (const std::string & *content*)

Creates object with provided private key

5.79.3 Member Function Documentation

5.79.3.1 bool Arc::DelegationConsumer::Acquire (std::string & *content*)

Ads private key into certificates chain in 'content' On exit content contains complete delegated credentials.

5.79.3.2 bool Arc::DelegationConsumer::Acquire (std::string & *content*, std::string & *identity*)

Includes the functionality of Acquire(content) plus extracting the credential identity.

5.79.3.3 bool Arc::DelegationConsumer::Backup (std::string & *content*)

Stores content of this object into a string

5.79.3.4 bool Arc::DelegationConsumer::Generate (void) [protected]

Private key

5.79.3.5 const std::string& Arc::DelegationConsumer::ID (void)

Return identifier of this object - not implemented

5.79.3.6 void Arc::DelegationConsumer::LogError (void) [protected]

Creates private key

5.79.3.7 bool Arc::DelegationConsumer::Request (std::string & content)

Make X509 certificate request from internal private key

5.79.3.8 bool Arc::DelegationConsumer::Restore (const std::string & content)

Restores content of object from string

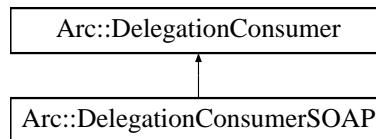
The documentation for this class was generated from the following file:

- DelegationInterface.h

5.80 Arc::DelegationConsumerSOAP Class Reference

```
#include <DelegationInterface.h>
```

Inheritance diagram for Arc::DelegationConsumerSOAP:



Public Member Functions

- [DelegationConsumerSOAP](#) (void)
- [DelegationConsumerSOAP](#) (const std::string &content)
- bool [DelegateCredentialsInit](#) (const std::string &id, const SOAPEnvelope &in, SOAPEnvelope &out)
- bool [UpdateCredentials](#) (std::string &credentials, const SOAPEnvelope &in, SOAPEnvelope &out)
- bool [UpdateCredentials](#) (std::string &credentials, std::string &identity, const SOAPEnvelope &in, SOAPEnvelope &out)
- bool [DelegatedToken](#) (std::string &credentials, [XMLNode](#) token)

5.80.1 Detailed Description

This class extends [DelegationConsumer](#) to support SOAP message exchange. Implements WS interface <http://www.nordugrid.org/schemas/delegation> described in delegation.wsdl.

5.80.2 Constructor & Destructor Documentation

5.80.2.1 `Arc::DelegationConsumerSOAP::DelegationConsumerSOAP (void)`

Creates object with new private key

5.80.2.2 `Arc::DelegationConsumerSOAP::DelegationConsumerSOAP (const std::string & content)`

Creates object with specified private key

5.80.3 Member Function Documentation

5.80.3.1 `bool Arc::DelegationConsumerSOAP::DelegateCredentialsInit (const std::string & id, const SOAPEnvelope & in, SOAPEnvelope & out)`

Process SOAP message which starts delegation. Generated message in 'out' is meant to be sent back to DelagationProviderSOAP. Argument 'id' contains identifier of procedure and is used only to produce SOAP message.

5.80.3.2 `bool Arc::DelegationConsumerSOAP::DelegatedToken (std::string & credentials, XMLNode token)`

Similar to UpdateCredentials but takes only DelegatedToken XML element

5.80.3.3 `bool Arc::DelegationConsumerSOAP::UpdateCredentials (std::string & credentials, std::string & identity, const SOAPEnvelope & in, SOAPEnvelope & out)`

Includes the functionality in above UpdateCredentials method; plus extracting the credential identity

5.80.3.4 `bool Arc::DelegationConsumerSOAP::UpdateCredentials (std::string & credentials, const SOAPEnvelope & in, SOAPEnvelope & out)`

Accepts delegated credentials. Process 'in' SOAP message and stores full proxy credentials in 'credentials'. 'out' message is generated for sending to DelagationProviderSOAP.

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

- `DelegationInterface.h`

5.81 Arc::DelegationContainerSOAP Class Reference

```
#include <DelegationInterface.h>
```

Public Member Functions

- bool [DelegateCredentialsInit](#) (const SOAPEvelope &in, SOAPEvelope &out, const std::string &client="")
- bool [UpdateCredentials](#) (std::string &credentials, const SOAPEvelope &in, SOAPEvelope &out, const std::string &client="")
- bool [DelegatedToken](#) (std::string &credentials, [XMLNode](#) token, const std::string &client="")

Protected Attributes

- int [max_size_](#)
- int [max_duration_](#)
- int [max_usage_](#)
- bool [context_lock_](#)

5.81.1 Detailed Description

Manages multiple delegated credentials. Delegation consumers are created automatically with [DelegateCredentialsInit](#) method up to [max_size_](#) and assigned unique identifier. Its methods are similar to those of [DelegationConsumerSOAP](#) with identifier included in SOAP message used to route execution to one of managed [DelegationConsumerSOAP](#) instances.

5.81.2 Member Function Documentation

5.81.2.1 bool Arc::DelegationContainerSOAP::DelegateCredentialsInit (const SOAPEvelope & in, SOAPEvelope & out, const std::string & client = " ")

See [DelegationConsumerSOAP::DelegateCredentialsInit](#) If 'client' is not empty then all subsequent calls involving access to generated credentials must contain same value in their 'client' arguments.

5.81.2.2 bool Arc::DelegationContainerSOAP::DelegatedToken (std::string & credentials, XMLNode token, const std::string & client = " ")

See [DelegationConsumerSOAP::DelegatedToken](#)

5.81.2.3 bool Arc::DelegationContainerSOAP::UpdateCredentials (std::string & credentials, const SOAPEvelope & in, SOAPEvelope & out, const std::string & client = " ")

See [DelegationConsumerSOAP::UpdateCredentials](#)

5.81.3 Field Documentation

5.81.3.1 `bool Arc::DelegationContainerSOAP::context_lock_ [protected]`

If true delegation consumer is deleted when connection context is destroyed

5.81.3.2 `int Arc::DelegationContainerSOAP::max_duration_ [protected]`

Lifetime of unused delegation consumer

5.81.3.3 `int Arc::DelegationContainerSOAP::max_size_ [protected]`

Max. number of delegation consumers

5.81.3.4 `int Arc::DelegationContainerSOAP::max_usage_ [protected]`

Max. times same delegation consumer may accept credentials

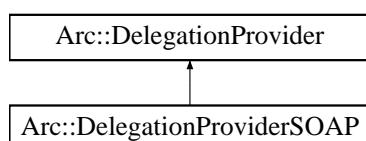
The documentation for this class was generated from the following file:

- `DelegationInterface.h`

5.82 Arc::DelegationProvider Class Reference

```
#include <DelegationInterface.h>
```

Inheritance diagram for Arc::DelegationProvider:



Public Member Functions

- `DelegationProvider (const std::string &credentials)`
- `DelegationProvider (const std::string &cert_file, const std::string &key_file, std::istream *inpwd=NULL)`
- `std::string Delegate (const std::string &request, const DelegationRestrictions &restrictions=DelegationRestrictions())`

5.82.1 Detailed Description

A provider of delegated credentials. During delegation procedure this class generates new credential to be used in proxy/delegated credential.

5.82.2 Constructor & Destructor Documentation

5.82.2.1 Arc::DelegationProvider::DelegationProvider (`const std::string & credentials`)

Creates instance from provided credentials. Credentials are used to sign delegated credentials. Arguments should contain PEM-encoded certificate, private key and optionally certificates chain.

5.82.2.2 Arc::DelegationProvider::DelegationProvider (`const std::string & cert_file, const std::string & key_file, std::istream * inpwd = NULL`)

Creates instance from provided credentials. Credentials are used to sign delegated credentials. Arguments should contain filesystem path to PEM-encoded certificate and private key. Optionally cert_file may contain certificates chain.

5.82.3 Member Function Documentation

5.82.3.1 `std::string Arc::DelegationProvider::Delegate (const std::string & request, const DelegationRestrictions & restrictions = DelegationRestrictions ())`

Perform delegation. Takes X509 certificate request and creates proxy credentials excluding private key. Result is then to be fed into [DelegationConsumer::Acquire](#)

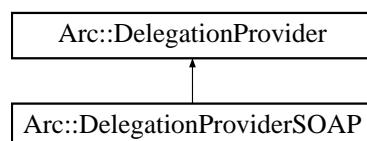
The documentation for this class was generated from the following file:

- `DelegationInterface.h`

5.83 Arc::DelegationProviderSOAP Class Reference

```
#include <DelegationInterface.h>
```

Inheritance diagram for Arc::DelegationProviderSOAP:



Public Member Functions

- `DelegationProviderSOAP (const std::string &credentials)`
- `DelegationProviderSOAP (const std::string &cert_file, const std::string &key_file, std::istream *inpwd=NULL)`
- `bool DelegateCredentialsInit (MCCIInterface &mcc_interface, MessageContext *context, ServiceType stype=ARCDelegation)`
- `bool DelegateCredentialsInit (MCCIInterface &mcc_interface, MessageAttributes *attributes_in, MessageAttributes *attributes_out, MessageContext *context, ServiceType stype=ARCDelegation)`
- `bool UpdateCredentials (MCCIInterface &mcc_interface, MessageContext *context, const DelegationRestrictions &restrictions=DelegationRestrictions(), ServiceType stype=ARCDelegation)`
- `bool UpdateCredentials (MCCIInterface &mcc_interface, MessageAttributes *attributes_in, MessageAttributes *attributes_out, MessageContext *context, const DelegationRestrictions &restrictions=DelegationRestrictions(), ServiceType stype=ARCDelegation)`
- `bool DelegatedToken (XMLNode parent)`
- `const std::string & ID (void)`

5.83.1 Detailed Description

Extension of `DelegationProvider` with SOAP exchange interface. This class is also a temporary container for intermediate information used during delegation procedure.

5.83.2 Constructor & Destructor Documentation

5.83.2.1 Arc::DelegationProviderSOAP::DelegationProviderSOAP (`const std::string & credentials`)

Creates instance from provided credentials. Credentials are used to sign delegated credentials.

5.83.2.2 Arc::DelegationProviderSOAP::DelegationProviderSOAP (`const std::string & cert_file, const std::string & key_file, std::istream * inpwd = NULL`)

Creates instance from provided credentials. Credentials are used to sign delegated credentials. Arguments should contain filesystem path to PEM-encoded certificate and private key. Optionally cert_file may contain certificates chain.

5.83.3 Member Function Documentation

5.83.3.1 bool Arc::DelegationProviderSOAP::DelegateCredentialsInit (MCCInterface & *mcc_interface*, MessageContext * *context*, ServiceType *stype* = ARCDelegation)

Performs DelegateCredentialsInit SOAP operation. As result request for delegated credentials is received by this instance and stored internally. Call to UpdateCredentials should follow.

5.83.3.2 bool Arc::DelegationProviderSOAP::DelegateCredentialsInit (MCCInterface & *mcc_interface*, MessageAttributes * *attributes_in*, MessageAttributes * *attributes_out*, MessageContext * *context*, ServiceType *stype* = ARCDelegation)

Extended version of DelegateCredentialsInit(MCCInterface&,MessageContext*). Additionally takes attributes for request and response message to make fine control on message processing possible.

5.83.3.3 bool Arc::DelegationProviderSOAP::DelegatedToken (XMLNode *parent*)

Generates DelegatedToken element. Element is created as child of provided XML element and contains structure described in delegation.wsdl.

5.83.3.4 const std::string& Arc::DelegationProviderSOAP::ID (void) [inline]

Returns the identifier provided by service accepting delegated credentials. This identifier may then be used to refer to credentials stored at service.

5.83.3.5 bool Arc::DelegationProviderSOAP::UpdateCredentials (MCCInterface & *mcc_interface*, MessageAttributes * *attributes_in*, MessageAttributes * *attributes_out*, MessageContext * *context*, const DelegationRestrictions & *restrictions* = DelegationRestrictions(), ServiceType *stype* = ARCDelegation)

Extended version of UpdateCredentials(MCCInterface&,MessageContext*). Additionally takes attributes for request and response message to make fine control on message processing possible.

5.83.3.6 bool Arc::DelegationProviderSOAP::UpdateCredentials (MCCInterface & *mcc_interface*, MessageContext * *context*, const DelegationRestrictions & *restrictions* = DelegationRestrictions(), ServiceType *stype* = ARCDelegation)

Performs UpdateCredentials SOAP operation. This concludes delegation procedure and passes delegatated credentials to [DelegationConsumerSOAP](#) instance.

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

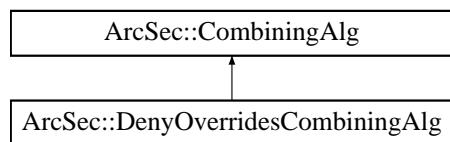
- DelegationInterface.h

5.84 ArcSec::DenyOverridesCombiningAlg Class Reference

Implement the "Deny-Overrides" algorithm.

```
#include <DenyOverridesAlg.h>
```

Inheritance diagram for ArcSec::DenyOverridesCombiningAlg:



Public Member Functions

- virtual Result [combine \(EvaluationCtx *ctx, std::list< Policy * > policies\)](#)
- virtual const std::string & [getAlgId \(void\) const](#)

5.84.1 Detailed Description

Implement the "Deny-Overrides" algorithm.

Deny-Overrides, scans the policy set which is given as the parameters of "combine" method, if gets "deny" result from any policy, then stops scanning and gives "deny" as result, otherwise gives "permit".

5.84.2 Member Function Documentation

5.84.2.1 virtual Result ArcSec::DenyOverridesCombiningAlg::combine (EvaluationCtx * ctx, std::list< Policy * > policies) [virtual]

If there is one policy which return negative evaluation result, then omit the other policies and return DECISION_DENY

Parameters

<i>ctx</i>	This object contains request information which will be used to evaluated against policy.
<i>policies</i>	This is a container which contains policy objects.

Returns

The combined result according to the algorithm.

Implements [ArcSec::CombiningAlg](#).

5.84.2.2 virtual const std::string& ArcSec::DenyOverridesCombiningAlg::getalgId (void) const [inline, virtual]

Get the identifier

Implements [ArcSec::CombiningAlg](#).

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

- DenyOverridesAlg.h

5.85 Arc::DiskSpaceRequirementType Class Reference

Data Fields

- [Range< int > DiskSpace](#)
- int [CacheDiskSpace](#)
- int [SessionDiskSpace](#)

5.85.1 Field Documentation

5.85.1.1 int Arc::DiskSpaceRequirementType::CacheDiskSpace

Specifies the required size of cache which must be available to the job in mega-bytes (MB). A negative value undefines this attribute

5.85.1.2 Range<int> Arc::DiskSpaceRequirementType::DiskSpace

Specifies the required size of disk space which must be available to the job in mega-bytes (MB). A negative value undefines this attribute

5.85.1.3 int Arc::DiskSpaceRequirementType::SessionDiskSpace

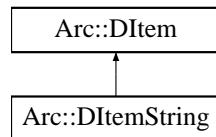
Specifies the required size of job session disk space which must be available to the job in mega-byte (MB). A negative value undefines this attribute.

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

- JobDescription.h

5.86 Arc::DItem Class Reference

Inheritance diagram for Arc::DItem:

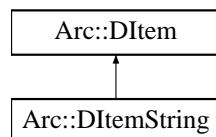


The documentation for this class was generated from the following file:

- DBranch.h

5.87 Arc::DItemString Class Reference

Inheritance diagram for Arc::DItemString:

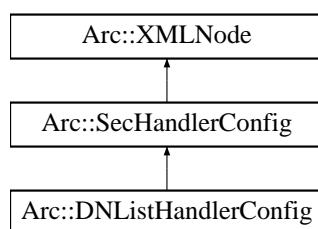


The documentation for this class was generated from the following file:

- DBranch.h

5.88 Arc::DNListHandlerConfig Class Reference

Inheritance diagram for Arc::DNListHandlerConfig:



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

- ClientInterface.h

5.89 DataStaging::DTR Class Reference

Data Transfer Request.

```
#include <DTR.h>
```

Public Member Functions

- `DTR ()`
- `DTR (const DTR &dtr)`
- `DTR (const std::string &source, const std::string &destination, const Arc::UserConfig &usercfg, const std::string &jobid, const uid_t &uid, Arc::Logger *log)`
- `~DTR ()`
- `operator bool () const`
- `bool operator! () const`
- `void registerCallback (DTRCallback *cb, StagingProcesses owner)`
- `std::list< DTRCallback * > get_callbacks (const std::map< StagingProcesses, std::list< DTRCallback * > > &proc_callback, StagingProcesses owner)`
- `void reset ()`
- `void set_id (const std::string &id)`
- `std::string get_id () const`
- `std::string get_short_id () const`
- `Arc::DataHandle & get_source ()`
- `const Arc::DataHandle & get_source () const`
- `Arc::DataHandle & get_destination ()`
- `const Arc::DataHandle & get_destination () const`
- `const Arc::UserConfig & get_usercfg () const`
- `void set_timeout (time_t value)`
- `Arc::Time get_timeout () const`
- `void set_process_time (const Arc::Period &process_time)`
- `Arc::Time get_process_time () const`
- `Arc::Time get_creation_time () const`
- `Arc::Time get_modification_time () const`
- `std::string get_parent_job_id () const`
- `void set_priority (int pri)`
- `int get_priority () const`
- `void set_transfer_share (std::string share_name)`
- `std::string get_transfer_share () const`
- `void set_sub_share (const std::string &share)`
- `std::string get_sub_share () const`
- `void set_tries_left (unsigned int tries)`
- `unsigned int get_tries_left () const`
- `void decrease_tries_left ()`
- `void set_status (DTRStatus stat)`
- `DTRStatus get_status ()`
- `void set_error_status (DTRErrorStatus::DTRErrorStatusType error_stat, DTRErrorStatus::DTRErrorLocation error_loc, const std::string &desc="")`

- void `reset_error_status ()`
- `DTRErrorStatus get_error_status ()`
- void `set_bytes_transferred` (unsigned long long int bytes)
- unsigned long long int `get_bytes_transferred () const`
- void `set_cancel_request ()`
- bool `cancel_requested () const`
- void `set_delivery_endpoint` (const `Arc::URL &endpoint`)
- const `Arc::URL & get_delivery_endpoint () const`
- void `host_cert_for_remote_delivery` (bool host)
- bool `host_cert_for_remote_delivery () const`
- void `set_cache_file` (const std::string &filename)
- std::string `get_cache_file () const`
- void `set_cache_parameters` (const `CacheParameters ¶m`)
- const `CacheParameters & get_cache_parameters () const`
- void `set_cache_state` (`CacheState state`)
- `CacheState get_cache_state () const`
- void `set_mapped_source` (const std::string &file="")
- std::string `get_mapped_source () const`
- `StagingProcesses get_owner () const`
- `Arc::User get_local_user () const`
- void `set_replication` (bool rep)
- bool `is_replication () const`
- void `set_force_registration` (bool force)
- bool `is_force_registration () const`
- `Arc::Logger * get_logger () const`
- void `connect_logger ()`
- void `disconnect_logger ()`
- void `push` (`StagingProcesses new_owner`)
- bool `suspend ()`
- bool `error () const`
- bool `is_designed_for_pre_processor () const`
- bool `is_designed_for_post_processor () const`
- bool `is_designed_for_delivery () const`
- bool `came_from_pre_processor () const`
- bool `came_from_post_processor () const`
- bool `came_from_delivery () const`
- bool `came_from_generator () const`
- bool `is_in_final_state () const`

Static Public Attributes

- static const `Arc::URL LOCAL_DELIVERY`

5.89.1 Detailed Description

Data Transfer Request.

[DTR](#) stands for Data Transfer Request and a [DTR](#) describes a data transfer between two endpoints, a source and a destination. There are several parameters and options relating to the transfer contained in a [DTR](#). The normal workflow is for a [Generator](#) to create a [DTR](#) and send it to the [Scheduler](#) for processing using `dtr.push(SCEDULER)`. If the [Generator](#) is a subclass of [DTRCallback](#), when the [Scheduler](#) has finished with the [DTR](#) the `receiveDTR()` callback method is called.

`registerCallback(this,DataStaging::GENERATOR)` can be used to activate the callback. The following simple [Generator](#) code sample illustrates how to use DTRs:

```
class MyGenerator : public DTRCallback {
public:
    void receiveDTR(DTR& dtr);
    void run();
private:
    Arc::SimpleCondition cond;
};

void MyGenerator::receiveDTR(DTR& dtr) {
    // DTR received back, so notify waiting condition
    std::cout << "Received DTR " << dtr.get_id() << std::endl;
    cond.signal();
}

void MyGenerator::run() {
    // start Scheduler thread
    Scheduler scheduler;
    scheduler.start();

    // create a DTR
    DTR dtr(source, destination,...);

    // register this callback
    dtr.registerCallback(this,DataStaging::GENERATOR);
    // this line must be here in order to pass the DTR to the Scheduler
    dtr.registerCallback(&scheduler,DataStaging::SCEDULER);

    // push the DTR to the Scheduler
    dtr.push(DataStaging::SCEDULER);

    // wait until callback is called
    cond.wait();
    // DTR is finished, so stop Scheduler
    scheduler.stop();
}
```

A lock protects member variables that are likely to be accessed and modified by multiple threads.

5.89.2 Constructor & Destructor Documentation

5.89.2.1 DataStaging::DTR::DTR (const std::string & *source*, const std::string & *destination*, const Arc::UserConfig & *usercfg*, const std::string & *jobid*, const uid_t & *uid*, Arc::Logger * *log*)

Normal constructor.

Construct a new [DTR](#).

Parameters

<i>source</i>	Endpoint from which to read data
<i>destination</i>	Endpoint to which to write data
<i>usercfg</i>	Provides some user configuration information
<i>jobid</i>	ID of the job associated with this data transfer
<i>uid</i>	UID to use when accessing local file system if source or destination is a local file. If this is different to the current uid then the current uid must have sufficient privileges to change uid.
<i>log</i>	Pointer to log object. If NULL the root logger is used.

5.89.3 Member Function Documentation

5.89.3.1 void DataStaging::DTR::registerCallback (DTRCallback * *cb*, StagingProcesses *owner*)

Register callback objects to be used during [DTR](#) processing.

Objects deriving from [DTRCallback](#) can be registered with this method. The callback method of these objects will then be called when the [DTR](#) is passed to the specified owner. Protected by lock.

5.89.3.2 void DataStaging::DTR::reset ()

Reset information held on this [DTR](#), such as resolved replicas, error state etc.

Useful when a failed [DTR](#) is to be retried.

5.89.3.3 void DataStaging::DTR::set_error_status (DTRErrorStatus::DTRErrorStatusType *error_stat*, DTRErrorStatus::DTRErrorLocation *error_loc*, const std::string & *desc* = " ")

Set the error status.

The [DTRErrorStatus](#) last error state field is set to the current status of the [DTR](#). Protected by lock.

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

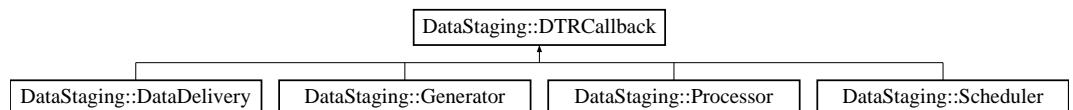
- [DTR.h](#)

5.90 DataStaging::DTRCallback Class Reference

The base class from which all callback-enabled classes should be derived.

```
#include <DTR.h>
```

Inheritance diagram for DataStaging::DTRCallback:



Public Member Functions

- virtual [~DTRCallback \(\)](#)
- virtual void [receiveDTR \(DTR &dtr\)=0](#)

5.90.1 Detailed Description

The base class from which all callback-enabled classes should be derived.

This class is a container for a callback method which is called when a [DTR](#) is to be passed to a component. Several components in data staging (eg [Scheduler](#), [Generator](#)) are subclasses of [DTRCallback](#), which allows them to receive DTRs through the callback system.

5.90.2 Constructor & Destructor Documentation

5.90.2.1 virtual DataStaging::DTRCallback::~DTRCallback() [inline, virtual]

Empty virtual destructor

5.90.3 Member Function Documentation

5.90.3.1 virtual void DataStaging::DTRCallback::receiveDTR (DTR & dtr) [pure virtual]

Defines the callback method called when a [DTR](#) is pushed to this object. Note that the [DTR](#) object is passed by reference and so there is no guarantee that it will exist after this callback method is called.

Implemented in [DataStaging::DataDelivery](#), [DataStaging::Generator](#), [DataStaging::Processor](#), and [DataStaging::Scheduler](#).

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

- DTR.h

5.91 DataStaging::DTRErrorStatus Class Reference

A class to represent error states reported by various components.

```
#include <DTRStatus.h>
```

Public Types

- enum DTRErrorStatusType {
 NONE_ERROR, INTERNAL_LOGIC_ERROR, INTERNAL_PROCESS_ERROR,
 SELF_REPLICATION_ERROR,
 CACHE_ERROR, TEMPORARY_REMOTE_ERROR, PERMANENT_REMOTE_-
 ERROR, LOCAL_FILE_ERROR,
 TRANSFER_SPEED_ERROR, STAGING_TIMEOUT_ERROR }
- enum DTRErrorLocation {
 NO_ERROR_LOCATION, ERROR_SOURCE, ERROR_DESTINATION, ERROR_-
 TRANSFER,
 ERROR_UNKNOWN }

Public Member Functions

- DTRErrorStatus (DTRErrorStatusType status, DTRStatus::DTRStatusType error_-
state, DTRErrorLocation location, const std::string &desc="")
- DTRErrorStatus ()
- DTRErrorStatusType GetErrorStatus () const
- DTRStatus::DTRStatusType GetLastErrorState () const
- DTRErrorLocation GetErrorLocation () const
- std::string GetDesc () const
- bool operator== (const DTRErrorStatusType &s) const
- bool operator== (const DTRErrorStatus &s) const
- bool operator!= (const DTRErrorStatusType &s) const
- bool operator!= (const DTRErrorStatus &s) const
- DTRErrorStatus & operator= (const DTRErrorStatusType &s)

5.91.1 Detailed Description

A class to represent error states reported by various components.

5.91.2 Member Enumeration Documentation

5.91.2.1 enum DataStaging::DTRErrorStatus::DTRErrorLocation

Describes where the error occurred.

Enumerator:

NO_ERROR_LOCATION No error.

ERROR_SOURCE Error with source.

ERROR_DESTINATION Error with destination.

ERROR_TRANSFER Error during transfer not directly related to source or destination.

ERROR_UNKNOWN Error occurred in an unknown location.

5.91.2.2 enum DataStaging::DTRErrorStatus::DTRErrorStatusType

A list of error types.

Enumerator:

NONE_ERROR No error.

INTERNAL_LOGIC_ERROR Internal error in Data Staging logic.

INTERNAL_PROCESS_ERROR Internal processing error, like losing contact with external process.

SELF_REPLICATION_ERROR Attempt to replicate a file to itself.

CACHE_ERROR Permanent error with cache.

TEMPORARY_REMOTE_ERROR Temporary error with remote service.

PERMANENT_REMOTE_ERROR Permanent error with remote service.

LOCAL_FILE_ERROR Error with local file.

TRANSFER_SPEED_ERROR Transfer rate was too slow.

STAGING_TIMEOUT_ERROR Waited for too long to become staging.

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

- DTRStatus.h

5.92 DataStaging::DTRLList Class Reference

Global list of all active DTRs in the system.

```
#include <DTRLList.h>
```

Public Member Functions

- bool `add_dtr` (const `DTR` &`DTRToAdd`)
- bool `delete_dtr` (`DTR` *`DTRToDelete`)
- bool `filter_dtrs_by_owner` (`StagingProcesses` `OwnerToFilter`, `std::list< DTR * >` &`FilteredList`)
- int `number_of_dtrs_by_owner` (`StagingProcesses` `OwnerToFilter`)
- bool `filter_dtrs_by_status` (`DTRStatus` `StatusToFilter`, `std::list< DTR * >` &`FilteredList`)
- bool `filter_dtrs_by_next_receiver` (`StagingProcesses` `NextReceiver`, `std::list< DTR * >` &`FilteredList`)
- bool `filter_pending_dtrs` (`std::list< DTR * >` &`FilteredList`)
- bool `filter_dtrs_by_job` (const `std::string` &`jobid`, `std::list< DTR * >` &`FilteredList`)
- `std::list< DTR * >` `all_dtrs` ()
- `std::list< std::string >` `all_jobs` ()
- void `dumpState` (const `std::string` &`path`)

5.92.1 Detailed Description

Global list of all active DTRs in the system.

This class contains several methods for filtering the list by owner, state etc

5.92.2 Member Function Documentation

5.92.2.1 bool DataStaging::DTRLList::add_dtr (const DTR & DTRToAdd)

Put a new `DTR` into the list.

A (pointer to a) copy of the `DTR` is added to the list, and so `DTRToAdd` can be deleted after this method is called.

5.92.2.2 bool DataStaging::DTRLList::delete_dtr (DTR * DTRToDelete)

Remove a `DTR` from the list.

The `DTRToDelete` object is destroyed, and hence should not be used after calling this method.

5.92.2.3 void DataStaging::DTRLList::dumpState (const std::string & path)

Dump state of all current DTRs to a destination, eg file, database, url...

Currently only file is supported.

Parameters

<code>path</code>	Path to the file in which to dump state.
-------------------	--

5.92.2.4 bool DataStaging::DTRLList::filter_dtrs_by_job (const std::string & *jobid*, std::list< DTR * > & *FilteredList*)

Get the list of DTRs corresponding to the given job ID.

Parameters

<i>FilteredList</i>	This list is filled with filtered DTRs
---------------------	--

5.92.2.5 bool DataStaging::DTRLList::filter_dtrs_by_next_receiver (StagingProcesses *NextReceiver*, std::list< DTR * > & *FilteredList*)

Select DTRs that are about to go to the specified process.

This selection is actually a virtual queue for pre-, post-processor and delivery.

Parameters

<i>FilteredList</i>	This list is filled with filtered DTRs
---------------------	--

5.92.2.6 bool DataStaging::DTRLList::filter_dtrs_by_owner (StagingProcesses *OwnerToFilter*, std::list< DTR * > & *FilteredList*)

Filter the queue to select DTRs owned by a specified process.

Parameters

<i>FilteredList</i>	This list is filled with filtered DTRs
---------------------	--

5.92.2.7 bool DataStaging::DTRLList::filter_dtrs_by_status (DTRStatus *StatusToFilter*, std::list< DTR * > & *FilteredList*)

Filter the queue to select DTRs with particular status.

If we have only one common queue for all DTRs, this method is necessary to make virtual queues for the DTRs about to go into the pre-, post-processor or delivery stages.

Parameters

<i>FilteredList</i>	This list is filled with filtered DTRs
---------------------	--

5.92.2.8 bool DataStaging::DTRLList::filter_pending_dtrs (std::list< DTR * > & *FilteredList*)

Select DTRs that have just arrived from pre-, post-processor, delivery or generator.

These DTRs need some reaction from the scheduler. This selection is actually a virtual queue of DTRs that need to be processed.

Parameters

<i>FilteredList</i>	This list is filled with filtered DTRs
---------------------	--

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

- `DTRList.h`

5.93 DataStaging::DTRStatus Class Reference

Class representing the status of a [DTR](#).

```
#include <DTRStatus.h>
```

Public Types

- enum `DTRStatusType` {
 `NEW, CHECK_CACHE, RESOLVE, QUERY_REPLICA,`
 `PRE_CLEAN, STAGE_PREPARE, TRANSFER_WAIT, TRANSFER,`
 `RELEASE_REQUEST, REGISTER_REPLICA, PROCESS_CACHE, DONE,`
 `CANCELLED, CANCELLED_FINISHED, ERROR, CHECKING_CACHE,`
 `CACHE_WAIT, CACHE_CHECKED, RESOLVING, RESOLVED,`
 `QUERYING_REPLICA, REPLICA_QUIERIED, PRE_CLEANING, PRE_CLEANED,`
 `STAGING_PREPARING, STAGING_PREPARING_WAIT, STAGED_PREPARED,`
 `TRANSFERRING,`
 `TRANSFERRING_CANCEL, TRANSFERRED, RELEASING_REQUEST, REQUEST_-`
 `RELEASED,`
 `REGISTERING_REPLICA, REPLICA_REGISTERED, PROCESSING_CACHE, CACHE_-`
 `PROCESSED,`
 `NULL_STATE` }

Public Member Functions

- `DTRStatus (const DTRStatusType &status, std::string desc=""")`
- `DTRStatus ()`
- `bool operator== (const DTRStatusType &s) const`
- `bool operator== (const DTRStatus &s) const`
- `bool operator!= (const DTRStatusType &s) const`
- `bool operator!= (const DTRStatus &s) const`
- `DTRStatus & operator= (const DTRStatusType &s)`
- `std::string str () const`
- `void SetDesc (const std::string &d)`
- `std::string GetDesc () const`
- `DTRStatusType GetStatus () const`

5.93.1 Detailed Description

Class representing the status of a [DTR](#).

5.93.2 Member Enumeration Documentation

5.93.2.1 enum DataStaging::DTRStatus::DTRStatusType

Possible state values.

Enumerator:

NEW Just created.

CHECK_CACHE Check the cache for the file may be already there.

RESOLVE Resolve a meta-protocol.

QUERY_REPLICA Query a replica.

PRE_CLEAN The destination should be deleted.

STAGE_PREPARE Prepare or stage the source and/or destination.

TRANSFER_WAIT Hold the ready transfer.

TRANSFER Transfer ready and can be started.

RELEASE_REQUEST Transfer finished, release requests on the storage.

REGISTER_REPLICA Register a new replica of the destination.

PROCESS_CACHE Destination is cacheable, process cache.

DONE Everything completed successfully.

CANCELLED Cancellation request fulfilled successfully.

CANCELLED_FINISHED Cancellation request fulfilled but [DTR](#) also completed transfer successfully.

ERROR Error occurred.

CHECKING_CACHE Checking the cache.

CACHE_WAIT Cache file is locked, waiting for its release.

CACHE_CHECKED Cache check completed.

RESOLVING Resolving replicas.

RESOLVED Replica resolution completed.

QUERYING_REPLICA Replica is being queried.

REPLICA_QUERIED Replica was queried.

PRE_CLEANING Deleting the destination.

PRE_CLEANED The destination file has been deleted.

STAGING_PREPARING Making a staging or preparing request.

STAGING_PREPARING_WAIT Wait for the status of the staging/preparing request.

STAGED_PREPARED Staging/preparing request completed.

TRANSFERRING Transfer is going.

TRANSFERRING_CANCEL Transfer is on-going but scheduled for cancellation.

TRANSFERRED Transfer completed.

RELEASING_REQUEST Releasing staging/preparing request.

REQUEST_RELEASED Release of staging/preparing request completed.

REGISTERING_REPLICA Registering a replica in an index service.

REPLICA_REGISTERED Replica registration completed.

PROCESSING_CACHE Releasing locks and copying/linking cache files to the session dir.

CACHE_PROCESSED Cache processing completed.

NULL_STATE "Stateless" [DTR](#)

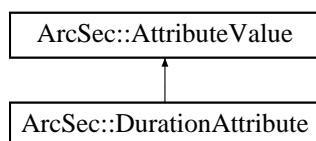
The documentation for this class was generated from the following file:

- DTRStatus.h

5.94 ArcSec::DurationAttribute Class Reference

```
#include <DateTimeAttribute.h>
```

Inheritance diagram for ArcSec::DurationAttribute:



Public Member Functions

- virtual bool [equal](#) ([AttributeValue](#) *other, bool check_id=true)
- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

5.94.1 Detailed Description

Format: P??Y??M??DT??H??M??S

5.94.2 Member Function Documentation

5.94.2.1 `virtual std::string ArcSec::DurationAttribute::encode() [virtual]`

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.94.2.2 `virtual bool ArcSec::DurationAttribute::equal(AttributeValue * value, bool check_id = true) [virtual]`

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.94.2.3 `virtual std::string ArcSec::DurationAttribute::getId() [inline, virtual]`

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.94.2.4 `virtual std::string ArcSec::DurationAttribute::getType() [inline, virtual]`

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

- `DateTimeAttribute.h`

5.95 Arc::EnvLockWrapper Class Reference

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

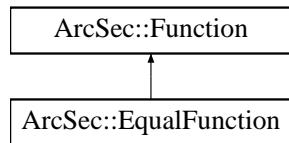
- `Utils.h`

5.96 ArcSec::EqualFunction Class Reference

Evaluate whether the two values are equal.

```
#include <EqualFunction.h>
```

Inheritance diagram for ArcSec::EqualFunction:



Public Member Functions

- virtual `AttributeValue * evaluate (AttributeValue *arg0, AttributeValue *arg1, bool check_id=true)`
- virtual `std::list<AttributeValue * > evaluate (std::list<AttributeValue * > args, bool check_id=true)`

Static Public Member Functions

- static `std::string getFunctionName (std::string datatype)`

5.96.1 Detailed Description

Evaluate whether the two values are equal.

5.96.2 Member Function Documentation

5.96.2.1 virtual `AttributeValue* ArcSec::EqualFunction::evaluate (AttributeValue * arg0, AttributeValue * arg1, bool check_id = true) [virtual]`

Evaluate two `AttributeValue` objects, and return one `AttributeValue` object

Implements `ArcSec::Function`.

5.96.2.2 virtual `std::list<AttributeValue*> ArcSec::EqualFunction::evaluate (std::list<AttributeValue * > args, bool check_id = true) [virtual]`

Evaluate a list of `AttributeValue` objects, and return a list of Attribute objects

Implements `ArcSec::Function`.

5.96.2.3 static `std::string ArcSec::EqualFunction::getFunctionName (std::string datatype) [static]`

help function to get the FunctionName

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

- `EqualFunction.h`

5.97 ArcSec::EvalResult Struct Reference

Struct to record the xml node and effect, which will be used by [Evaluator](#) to get the information about which rule/policy(in xmlnode) is satisfied.

```
#include <Result.h>
```

5.97.1 Detailed Description

Struct to record the xml node and effect, which will be used by [Evaluator](#) to get the information about which rule/policy(in xmlnode) is satisfied.

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

- Result.h

5.98 ArcSec::EvaluationCtx Class Reference

[EvaluationCtx](#), in charge of storing some context information for.

```
#include <EvaluationCtx.h>
```

Public Member Functions

- [EvaluationCtx \(Request *request\)](#)

5.98.1 Detailed Description

[EvaluationCtx](#), in charge of storing some context information for.

5.98.2 Constructor & Destructor Documentation

5.98.2.1 ArcSec::EvaluationCtx::EvaluationCtx (Request * request) [inline]

Construct a new [EvaluationCtx](#) based on the given request

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

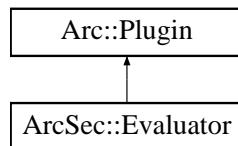
- EvaluationCtx.h

5.99 ArcSec::Evaluator Class Reference

Interface for policy evaluation. Execute the policy evaluation, based on the request and policy.

```
#include <Evaluator.h>
```

Inheritance diagram for ArcSec::Evaluator:



Public Member Functions

- virtual `Response * evaluate (Request *request)=0`
- virtual `Response * evaluate (const Source &request)=0`
- virtual `Response * evaluate (Request *request, const Source &policy)=0`
- virtual `Response * evaluate (const Source &request, const Source &policy)=0`
- virtual `Response * evaluate (Request *request, Policy *policyobj)=0`
- virtual `Response * evaluate (const Source &request, Policy *policyobj)=0`
- virtual `AttributeFactory * getAttrFactory ()=0`
- virtual `FnFactory * getFnFactory ()=0`
- virtual `AlgFactory * getAlgFactory ()=0`
- virtual `void addPolicy (const Source &policy, const std::string &id="")=0`
- virtual `void addPolicy (Policy *policy, const std::string &id="")=0`
- virtual `void setCombiningAlg (EvaluatorCombiningAlg alg)=0`
- virtual `void setCombiningAlg (CombiningAlg *alg=NULL)=0`
- virtual const char * `getName (void) const =0`

Protected Member Functions

- virtual `Response * evaluate (EvaluationCtx *ctx)=0`

5.99.1 Detailed Description

Interface for policy evaluation. Execute the policy evaluation, based on the request and policy.

5.99.2 Member Function Documentation

5.99.2.1 virtual void ArcSec::Evaluator::addPolicy (const Source & policy, const std::string & id = " ") [pure virtual]

Add policy from specified source to the evaluator. `Policy` will be marked with id.

```
5.99.2.2 virtual void ArcSec::Evaluator::addPolicy ( Policy * policy, const std::string & id = "" ) [pure virtual]
```

Add policy to the evaluator. [Policy](#) will be marked with id. The policy object is taken over by this instance and will be destroyed in destructor.

```
5.99.2.3 virtual Response* ArcSec::Evaluator::evaluate ( const Source & request, const Source & policy ) [pure virtual]
```

Evaluate the request from specified source against the policy from specified source. In some implementations all of the existing policies inside the evaluator may be destroyed by this method.

```
5.99.2.4 virtual Response* ArcSec::Evaluator::evaluate ( Request * request ) [pure virtual]
```

Evaluates the request by using a [Request](#) object. Evaluation is done till at least one of policies is satisfied.

```
5.99.2.5 virtual Response* ArcSec::Evaluator::evaluate ( Request * request, Policy * policyobj ) [pure virtual]
```

Evaluate the specified request against the specified policy. In some implementations all of the existing policy inside the evaluator may be destroyed by this method.

```
5.99.2.6 virtual Response* ArcSec::Evaluator::evaluate ( const Source & request ) [pure virtual]
```

Evaluates the request by using a specified source

```
5.99.2.7 virtual Response* ArcSec::Evaluator::evaluate ( Request * request, const Source & policy ) [pure virtual]
```

Evaluate the specified request against the policy from specified source. In some implementations all of the existing policies inside the evaluator may be destroyed by this method.

```
5.99.2.8 virtual Response* ArcSec::Evaluator::evaluate ( const Source & request, Policy * policyobj ) [pure virtual]
```

Evaluate the request from specified source against the specified policy. In some implementations all of the existing policies inside the evaluator may be destroyed by this method.

5.99.2.9 **virtual Response* ArcSec::Evaluator::evaluate (EvaluationCtx * ctx)**
[protected, pure virtual]

Evaluate the request by using the [EvaluationCtx](#) object (which includes the information about request). The ctx is destroyed inside this method (why?!?!?).

5.99.2.10 **virtual AlgFactory* ArcSec::Evaluator::getAlgFactory () [pure virtual]**

Get the [AlgFactory](#) object

Referenced by ArcSec::EvaluatorContext::operator AlgFactory *().

5.99.2.11 **virtual AttributeFactory* ArcSec::Evaluator::getAttrFactory () [pure virtual]**

Get the [AttributeFactory](#) object

Referenced by ArcSec::EvaluatorContext::operator AttributeFactory *().

5.99.2.12 **virtual FnFactory* ArcSec::Evaluator::getFnFactory () [pure virtual]**

Get the [FnFactory](#) object

Referenced by ArcSec::EvaluatorContext::operator FnFactory *().

5.99.2.13 **virtual const char* ArcSec::Evaluator::getName (void) const [pure virtual]**

Get the name of this evaluator

5.99.2.14 **virtual void ArcSec::Evaluator::setCombiningAlg (EvaluatorCombiningAlg alg) [pure virtual]**

Specifies one of simple combining algorithms. In case of multiple policies their results will be combined using this algorithm.

5.99.2.15 **virtual void ArcSec::Evaluator::setCombiningAlg (CombiningAlg * alg =NULL) [pure virtual]**

Specifies loadable combining algorithms. In case of multiple policies their results will be combined using this algorithm. To switch to simple algorithm specify NULL argument.

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

- Evaluator.h

5.100 ArcSec::EvaluatorContext Class Reference

Context for evaluator. It includes the factories which will be used to create related objects.

```
#include <Evaluator.h>
```

Public Member Functions

- [operator AttributeFactory *\(\)](#)
- [operator FnFactory *\(\)](#)
- [operator AlgFactory *\(\)](#)

5.100.1 Detailed Description

Context for evaluator. It includes the factories which will be used to create related objects.

5.100.2 Member Function Documentation

5.100.2.1 ArcSec::EvaluatorContext::operator AlgFactory *() [inline]

Returns associated [AlgFactory](#) object

References ArcSec::Evaluator::getAlgFactory().

5.100.2.2 ArcSec::EvaluatorContext::operator AttributeFactory *() [inline]

Returns associated [AttributeFactory](#) object

References ArcSec::Evaluator::getAttrFactory().

5.100.2.3 ArcSec::EvaluatorContext::operator FnFactory *() [inline]

Returns associated [FnFactory](#) object

References ArcSec::Evaluator::getFnFactory().

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

- Evaluator.h

5.101 ArcSec::EvaluatorLoader Class Reference

[EvaluatorLoader](#) is implemented as a helper class for loading different [Evaluator](#) objects, like ArcEvaluator.

```
#include <EvaluatorLoader.h>
```

Public Member Functions

- `Evaluator * getEvaluator (const std::string &classname)`
- `Evaluator * getEvaluator (const Policy *policy)`
- `Evaluator * getEvaluator (const Request *request)`
- `Request * getRequest (const std::string &classname, const Source &request-source)`
- `Request * getRequest (const Source &requestsource)`
- `Policy * getPolicy (const std::string &classname, const Source &policysource)`
- `Policy * getPolicy (const Source &policysource)`

5.101.1 Detailed Description

`EvaluatorLoader` is implemented as a helper class for loading different `Evaluator` objects, like `ArcEvaluator`.

The object loading is based on the configuration information about evaluator, including information for factory class, request, policy and evaluator itself

5.101.2 Member Function Documentation

5.101.2.1 `Evaluator* ArcSec::EvaluatorLoader::getEvaluator (const std::string & classname)`

Get evaluator object according to the class name

5.101.2.2 `Evaluator* ArcSec::EvaluatorLoader::getEvaluator (const Policy * policy)`

Get evaluator object suitable for presented policy

5.101.2.3 `Evaluator* ArcSec::EvaluatorLoader::getEvaluator (const Request * request)`

Get evaluator object suitable for presented request

5.101.2.4 `Policy* ArcSec::EvaluatorLoader::getPolicy (const Source & policysource)`

Get proper policy object according to the policy source

5.101.2.5 `Policy* ArcSec::EvaluatorLoader::getPolicy (const std::string & classname, const Source & policysource)`

Get policy object according to the class name, based on the policy source

5.101.2.6 Request* ArcSec::EvaluatorLoader::getRequest (const std::string & *classname*, const Source & *requestsource*)

Get request object according to the class name, based on the request source

5.101.2.7 Request* ArcSec::EvaluatorLoader::getRequest (const Source & *requestsource*)

Get request object according to the request source

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

- EvaluatorLoader.h

5.102 Arc::ExecutableType Class Reference

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

- JobDescription.h

5.103 Arc::ExecutionTarget Class Reference

[ExecutionTarget](#).

```
#include <ExecutionTarget.h>
```

Public Member Functions

- [ExecutionTarget \(\)](#)
- [ExecutionTarget \(const ExecutionTarget &target\)](#)
- [ExecutionTarget \(const long int addrptr\)](#)
- [ExecutionTarget & operator= \(const ExecutionTarget &target\)](#)
- [Submitter * GetSubmitter \(const UserConfig &ucfg\) const](#)
- [void Update \(const JobDescription &jobdesc\)](#)
- [void Print \(bool longlist\) const](#)
- [void SaveToStream \(std::ostream &out, bool longlist\) const](#)

Data Fields

- std::string [ComputingShareName](#)
- int [MaxMainMemory](#)
- int [MaxVirtualMemory](#)
- int [MaxDiskSpace](#)
- std::map< [Period](#), int > [FreeSlotsWithDuration](#)
- [Software OperatingSystem](#)
- std::list< [ApplicationEnvironment](#) > [ApplicationEnvironments](#)

5.103.1 Detailed Description

[ExecutionTarget](#).

This class describe a target which accept computing jobs. All of the members contained in this class, with a few exceptions, are directly linked to attributes defined in the GLUE Specification v. 2.0 (GFD-R-P.147).

5.103.2 Constructor & Destructor Documentation

5.103.2.1 Arc::ExecutionTarget::ExecutionTarget ()

Create an [ExecutionTarget](#).

Default constructor to create an [ExecutionTarget](#). Takes no arguments.

5.103.2.2 Arc::ExecutionTarget::ExecutionTarget (const ExecutionTarget & target)

Create an [ExecutionTarget](#).

Copy constructor.

Parameters

<code>target</code>	ExecutionTarget to copy.
---------------------	--

5.103.2.3 Arc::ExecutionTarget::ExecutionTarget (const long int addrptr)

Create an [ExecutionTarget](#).

Copy constructor? Needed from Python?

Parameters

<code>addrptr</code>

5.103.3 Member Function Documentation

5.103.3.1 Submitter* Arc::ExecutionTarget::GetSubmitter (const UserConfig & ucfg) const

Get [Submitter](#) to the computing resource represented by the [ExecutionTarget](#).

Method which returns a specialized [Submitter](#) which can be used for submitting jobs to the computing resource represented by the [ExecutionTarget](#). In order to return the correct specialized [Submitter](#) the GridFlavour variable must be correctly set.

Parameters

<i>ucfg</i>	UserConfig object with paths to user credentials etc.
-------------	---

5.103.3.2 **ExecutionTarget&** Arc::ExecutionTarget::operator= (const ExecutionTarget & *target*)

Create an [ExecutionTarget](#).

Assignment operator

Parameters

<i>target</i>	is ExecutionTarget to copy.
---------------	---

5.103.3.3 void Arc::ExecutionTarget::Print (bool *longlist*) const

DEPRECATED: Print the [ExecutionTarget](#) information to std::cout.

This method is deprecated, use the SaveToStream method instead. Method to print the [ExecutionTarget](#) attributes to std::cout

Parameters

<i>longlist</i>	is true for long list printing.
-----------------	---------------------------------

See also

[SaveToStream](#)

5.103.3.4 void Arc::ExecutionTarget::SaveToStream (std::ostream & *out*, bool *longlist*) const

Print the [ExecutionTarget](#) information to a std::ostream object.

Method to print the [ExecutionTarget](#) attributes to a std::ostream object.

Parameters

<i>out</i>	is the std::ostream to print the attributes to.
<i>longlist</i>	should be set to true for printing a long list.

5.103.3.5 void Arc::ExecutionTarget::Update (const JobDescription & *jobdesc*)

Update [ExecutionTarget](#) after successful job submission.

Method to update the [ExecutionTarget](#) after a job successfully has been submitted to the computing resource it represents. E.g. if a job is sent to the computing resource and is expected to enter the queue, then the WaitingJobs attribute is incremented with 1.

Parameters

<i>jobdesc</i>	contains all information about the job submitted.
----------------	---

5.103.4 Field Documentation**5.103.4.1 std::list<ApplicationEnvironment>
Arc::ExecutionTarget::ApplicationEnvironments**

ApplicationEnvironments.

The ApplicationEnvironments member is a list of ApplicationEnvironment's, defined in section 6.7 [GLUE2](#).

5.103.4.2 std::string Arc::ExecutionTarget::ComputingShareName

ComputingShareName String 0..1.

Human-readable name. This variable represents the ComputingShare.Name attribute of [GLUE2](#).

5.103.4.3 std::map<Period, int> Arc::ExecutionTarget::FreeSlotsWithDuration

FreeSlotsWithDuration std::map<Period, int>

This attribute express the number of free slots with their time limits. The keys in the std::map are the time limit ([Period](#)) for the number of free slots stored as the value (int). If no time limit has been specified for a set of free slots then the key will equal Period(LONG_MAX).

5.103.4.4 int Arc::ExecutionTarget::MaxDiskSpace

MaxDiskSpace UInt64 0..1 GB.

The maximum disk space that a job is allowed use in the working; if the limit is hit, then the LRMS MAY kill the job. A negative value specifies that this member is undefined.

5.103.4.5 int Arc::ExecutionTarget::MaxMainMemory

MaxMainMemory UInt64 0..1 MB.

The maximum physical RAM that a job is allowed to use; if the limit is hit, then the LRMS MAY kill the job. A negative value specifies that this member is undefined.

5.103.4.6 int Arc::ExecutionTarget::MaxVirtualMemory

MaxVirtualMemory UInt64 0..1 MB.

The maximum total memory size (RAM plus swap) that a job is allowed to use; if the limit is hit, then the LRMS MAY kill the job. A negative value specifies that this member is undefined.

5.103.4.7 Software Arc::ExecutionTarget::OperatingSystem

OperatingSystem.

The OperatingSystem member is not present in [GLUE2](#) but contains the three [GLUE2](#) attributes OSFamily, OSName and OSVersion.

- OSFamily OSFamily_t 1 .. * The general family to which the Execution Environment operating * system belongs.
- OSName OSName_t 0..1 .. * The specific name of the operating system
- OSVersion String 0..1 .. * The version of the operating system, as defined by the vendor.

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

- ExecutionTarget.h

5.104 Arc::ExpirationReminder Class Reference

A class intended for internal use within counters.

```
#include <Counter.h>
```

Public Member Functions

- bool [operator<](#) (const [ExpirationReminder](#) &other) const
- Glib::TimeVal [getExpiryTime](#) () const
- [Counter::IDType getReservationID](#) () const

Friends

- class [Counter](#)

5.104.1 Detailed Description

A class intended for internal use within counters.

This class is used for "reminder objects" that are used for automatic deallocation of self-expiring reservations.

5.104.2 Member Function Documentation

5.104.2.1 Glib::TimeVal Arc::ExpirationReminder::getExpiryTime () const

Returns the expiry time.

This method returns the expiry time of the reservation that this [ExpirationReminder](#) is associated with.

Returns

The expiry time.

5.104.2.2 Counter::IDType Arc::ExpirationReminder::getReservationID () const

Returns the identification number of the reservation.

This method returns the identification number of the self-expiring reservation that this [ExpirationReminder](#) is associated with.

Returns

The identification number.

5.104.2.3 bool Arc::ExpirationReminder::operator< (const ExpirationReminder & other) const

Less than operator, compares "soonness".

This is the less than operator for the [ExpirationReminder](#) class. It compares the priority of such objects with respect to which reservation expires first. It is used when reminder objects are inserted in a priority queue in order to always place the next reservation to expire at the top.

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

- Counter.h

5.105 Arc::FileAccess Class Reference

Defines interface for accessing filesystems.

```
#include <FileAccess.h>
```

Data Structures

- struct [header_t](#)

Public Member Functions

- bool `ping` (void)
- bool `setuid` (int uid, int gid)
- bool `mkdir` (const std::string &path, mode_t mode)
- bool `mkdirm` (const std::string &path, mode_t mode)
- bool `link` (const std::string &oldpath, const std::string &newpath)
- bool `softlink` (const std::string &oldpath, const std::string &newpath)
- bool `copy` (const std::string &oldpath, const std::string &newpath, mode_t mode)
- bool `chmod` (const std::string &path, mode_t mode)
- bool `stat` (const std::string &path, struct stat &st)
- bool `lstat` (const std::string &path, struct stat &st)
- bool `fstat` (struct stat &st)
- bool `truncate` (off_t length)
- off_t `fallocate` (off_t length)
- bool `readlink` (const std::string &path, std::string &linkpath)
- bool `remove` (const std::string &path)
- bool `unlink` (const std::string &path)
- bool `rmdir` (const std::string &path)
- bool `rmdirr` (const std::string &path)
- bool `opendir` (const std::string &path)
- bool `closedir` (void)
- bool `readdir` (std::string &name)
- bool `open` (const std::string &path, int flags, mode_t mode)
- bool `close` (void)
- bool `mkstemp` (std::string &path, mode_t mode)
- off_t `lseek` (off_t offset, int whence)
- ssize_t `read` (void *buf, size_t size)
- ssize_t `write` (const void *buf, size_t size)
- ssize_t `pread` (void *buf, size_t size, off_t offset)
- ssize_t `pwrite` (const void *buf, size_t size, off_t offset)
- int `geterrno` ()
- `operator bool` (void)
- bool `operator!` (void)

Static Public Member Functions

- static void `testtune` (void)

5.105.1 Detailed Description

Defines interface for accessing filesystems.

This class accesses local filesystem through proxy executable which allows to switch user id in multithreaded systems without introducing conflict with other threads. Its methods are mostly replicas of corresponding POSIX functions with some convenience tweaking.

5.105.2 Member Function Documentation

5.105.2.1 `bool Arc::FileAccess::copy (const std::string & oldpath, const std::string & newpath, mode_t mode)`

Copy file to new location. If new file is created it is assigned specified mode.

5.105.2.2 `int Arc::FileAccess::geterrno () [inline]`

Get errno of last operation. Every operation resets errno.

5.105.2.3 `bool Arc::FileAccess::mkdirp (const std::string & path, mode_t mode)`

Make a directory and assign it specified mode. If missing all intermediate directories are created too.

5.105.2.4 `bool Arc::FileAccess::mkstemp (std::string & path, mode_t mode)`

Open new temporary file for writing. On input path contains template of file name ending with XXXXXX. On output path is path to created file.

5.105.2.5 `bool Arc::FileAccess::open (const std::string & path, int flags, mode_t mode)`

Open file. Only one file may be open at a time.

5.105.2.6 `bool Arc::FileAccess::opendir (const std::string & path)`

Open directory. Only one directory may be open at a time.

5.105.2.7 `bool Arc::FileAccess::setuid (int uid, int gid)`

Modify user uid and gid. If any is set to 0 then executable is switched to original uid/gid.

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

- FileAccess.h

5.106 Arc::FileLock Class Reference

A general file locking class.

```
#include <FileLock.h>
```

Public Member Functions

- `FileLock (const std::string &filename, unsigned int timeout=DEFAULT_LOCK_TIMEOUT, bool use_pid=true)`
- `bool acquire (bool &lock_removed)`
- `bool acquire ()`
- `bool release (bool force=false)`
- `bool check ()`

Static Public Member Functions

- `static std::string getLockSuffix ()`

Static Public Attributes

- `static const int DEFAULT_LOCK_TIMEOUT`
- `static const std::string LOCK_SUFFIX`

5.106.1 Detailed Description

A general file locking class.

This class can be used when protected access is required to files which are used by multiple processes or threads. Call `acquire()` to obtain a lock and `release()` to release it when finished. `check()` can be used to verify if a lock is valid for the current process. Locks are independent of `FileLock` objects - locks are only created and destroyed through `acquire()` and `release()`, not on creation or destruction of `FileLock` objects.

Unless `use_pid` is set false, the process ID and hostname of the calling process are stored in a file `filename.lock` in the form `pid`. This information is used to determine whether a lock is still valid. It is also possible to specify a timeout on the lock.

To ensure an atomic locking operation, `acquire()` first creates a temporary lock file `filename.lock.XXXXXX`, then attempts to rename this file to `filename.lock`. After a successful rename the lock file is checked to make sure the correct process ID and hostname are inside. This eliminates race conditions where multiple processes compete to obtain the lock.

5.106.2 Constructor & Destructor Documentation

5.106.2.1 `Arc::FileLock::FileLock (const std::string & filename, unsigned int timeout = DEFAULT_LOCK_TIMEOUT, bool use_pid = true)`

Create a new `FileLock` object.

Parameters

<code>filename</code>	The name of the file to be locked
-----------------------	-----------------------------------

<i>timeout</i>	The timeout of the lock
<i>use_pid</i>	If true, use process id in the lock and to determine lock validity

5.106.3 Member Function Documentation

5.106.3.1 bool Arc::FileLock::acquire (bool & *lock_removed*)

Acquire the lock.

Returns true if the lock was acquired successfully. Locks are acquired if no lock file currently exists, or if the current lock file is invalid. A lock is invalid if the process ID inside the lock no longer exists on the host inside the lock, or the age of the lock file is greater than the lock timeout.

Parameters

<i>lock_removed</i>	Set to true if an existing lock was removed due to being invalid. In this case the caller may decide to check or delete the file as it is potentially corrupted.
---------------------	--

Returns

True if lock is successfully acquired

5.106.3.2 bool Arc::FileLock::acquire ()

Acquire the lock.

Callers can use this version of [acquire\(\)](#) if they do not care whether an invalid lock was removed in the process of obtaining the lock.

5.106.3.3 bool Arc::FileLock::check ()

Check the lock is valid.

Returns true if the lock is valid for the current process

5.106.3.4 bool Arc::FileLock::release (bool *force* = false)

Release the lock.

Parameters

<i>force</i>	Remove the lock without checking ownership or timeout
--------------	---

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

- FileLock.h

5.107 Arc::FileType Class Reference

Data Fields

- std::string [Checksum](#)

5.107.1 Field Documentation

5.107.1.1 std::string Arc::FileType::Checksum

MD5 checksum of file.

The Checksum attribute specifies the textual representation of MD5 checksum of file in base 16.

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

- JobDescription.h

5.108 Arc::FinderLoader Class Reference

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

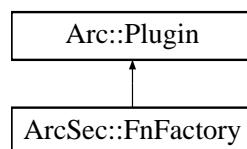
- FinderLoader.h

5.109 ArcSec::FnFactory Class Reference

Interface for function factory class.

```
#include <FnFactory.h>
```

Inheritance diagram for ArcSec::FnFactory:



Public Member Functions

- virtual [Function * createFn](#) (const std::string &type)=0

5.109.1 Detailed Description

Interface for function factory class.

[FnFactory](#) is in charge of creating [Function](#) object according to the algorithm type given as argument of method `createFn`. This class can be inherited for implementing a factory class which can create some specific [Function](#) objects.

5.109.2 Member Function Documentation

**5.109.2.1 virtual [Function](#)* ArcSec::FnFactory::createFn (const std::string & *type*)
[pure virtual]**

creat algorithm object based on the type algorithm type

Parameters

<i>type</i>	The type of Function
-------------	--------------------------------------

Returns

The object of [Function](#)

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

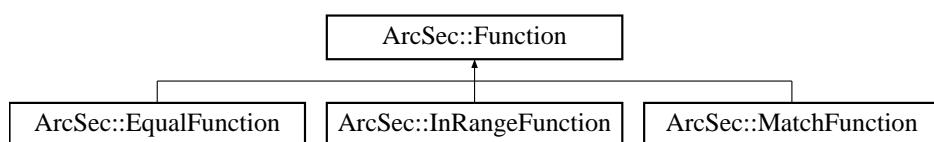
- FnFactory.h

5.110 ArcSec::Function Class Reference

Interface for function, which is in charge of evaluating two [AttributeValue](#).

```
#include <Function.h>
```

Inheritance diagram for ArcSec::Function:



Public Member Functions

- virtual [AttributeValue](#) * `evaluate` ([AttributeValue](#) **arg0*, [AttributeValue](#) **arg1*, bool *check_id*=true)=0
- virtual std::list< [AttributeValue](#) * > `evaluate` (std::list< [AttributeValue](#) * > *args*, bool *check_id*=true)=0

5.110.1 Detailed Description

Interface for function, which is in charge of evaluating two [AttributeValue](#).

5.110.2 Member Function Documentation

5.110.2.1 `virtual AttributeValue* ArcSec::Function::evaluate (AttributeValue * arg0, AttributeValue * arg1, bool check_id = true) [pure virtual]`

Evaluate two [AttributeValue](#) objects, and return one [AttributeValue](#) object

Implemented in [ArcSec::EqualFunction](#), [ArcSec::InRangeFunction](#), and [ArcSec::MatchFunction](#).

5.110.2.2 `virtual std::list<AttributeValue*> ArcSec::Function::evaluate (std::list<AttributeValue * > args, bool check_id = true) [pure virtual]`

Evaluate a list of [AttributeValue](#) objects, and return a list of Attribute objects

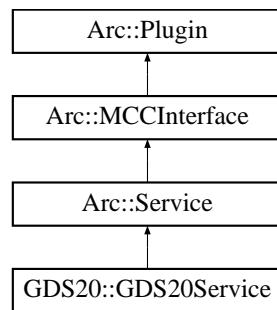
Implemented in [ArcSec::EqualFunction](#), [ArcSec::InRangeFunction](#), and [ArcSec::MatchFunction](#).

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

- [Function.h](#)

5.111 GDS20::GDS20Service Class Reference

Inheritance diagram for GDS20::GDS20Service:



Public Member Functions

- `virtual Arc::MCC_Status process (Arc::Message &inmsg, Arc::Message &outmsg)`

5.111.1 Member Function Documentation

5.111.1.1 virtual Arc::MCC_Status GDS20::GDS20Service::process (Arc::Message & request, Arc::Message & response) [virtual]

Method for processing of requests and responses. This method is called by preceding MCC in chain when a request needs to be processed. This method must call similar method of next MCC in chain unless any failure happens. Result returned by call to next MCC should be processed and passed back to previous MCC. In case of failure this method is expected to generate valid error response and return it back to previous MCC without calling the next one.

Parameters

<i>request</i>	The request that needs to be processed.
<i>response</i>	A Message object that will contain the response of the request when the method returns.

Returns

An object representing the status of the call.

Implements [Arc::MCCLInterface](#).

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

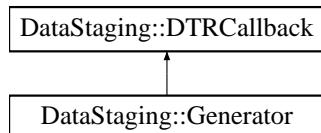
- gds20.h

5.112 DataStaging::Generator Class Reference

Simple [Generator](#) implementation.

```
#include <Generator.h>
```

Inheritance diagram for DataStaging::Generator:



Public Member Functions

- virtual void [receiveDTR \(DTR &dtr\)](#)
- void [run \(const std::string &source, const std::string &destination\)](#)

5.112.1 Detailed Description

Simple [Generator](#) implementation.

This [Generator](#) implementation is included in the data staging library for basic direct testing of the library and to show how a [Generator](#) can be written. It has one method, [run\(\)](#), which creates a single [DTR](#) and submits it to the [Scheduler](#).

5.112.2 Member Function Documentation

5.112.2.1 virtual void DataStaging::Generator::receiveDTR (DTR & dtr) [virtual]

Implementation of callback from [DTRCallback](#).

Callback method used when [DTR](#) processing is complete to pass back to the generator. The [DTR](#) is passed by value so that the scheduler can delete its copy of the object after calling this method.

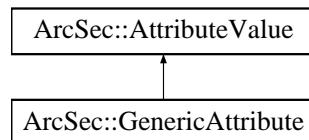
Implements [DataStaging::DTRCallback](#).

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

- Generator.h

5.113 ArcSec::GenericAttribute Class Reference

Inheritance diagram for ArcSec::GenericAttribute:



Public Member Functions

- virtual bool [equal](#) ([AttributeValue](#) *other, bool check_id=true)
- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

5.113.1 Member Function Documentation

5.113.1.1 virtual std::string ArcSec::GenericAttribute::encode () [inline, virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

```
5.113.1.2 virtual bool ArcSec::GenericAttribute::equal ( AttributeValue * value, bool check_id  
=true ) [virtual]
```

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

```
5.113.1.3 virtual std::string ArcSec::GenericAttribute::getId ( ) [inline, virtual]
```

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

```
5.113.1.4 virtual std::string ArcSec::GenericAttribute::getType ( ) [inline,  
virtual]
```

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

- GenericAttribute.h

5.114 Arc::GlobusResult Class Reference

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

- GlobusErrorUtils.h

5.115 Arc::GLUE2 Class Reference

[GLUE2](#) parser.

```
#include <GLUE2.h>
```

5.115.1 Detailed Description

[GLUE2](#) parser.

This class pparses [GLUE2](#) infromation rendeed in XML and transfers information into various classes representing different types of objects which [GLUE2](#) information model can describe. This parser uses GLUE Specification v. 2.0 (GFD-R-P.147).

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

- GLUE2.h

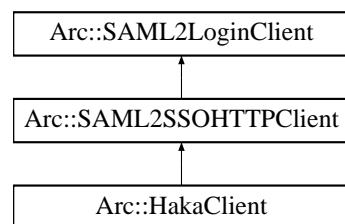
5.116 Arc::GSSCredential Class Reference

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

- GSSCredential.h

5.117 Arc::HakaClient Class Reference

Inheritance diagram for Arc::HakaClient:



Protected Member Functions

- [MCC_Status processIdPLogin \(const std::string username, const std::string password\)](#)
- [MCC_Status processConsent \(\)](#)
- [MCC_Status processIdP2Confusa \(\)](#)

5.117.1 Member Function Documentation

5.117.1.1 [MCC_Status Arc::HakaClient::processConsent \(\) \[protected, virtual\]](#)

If the IdP has a consent module and the user has not saved her consent, this method will ask the user for consent to transmission of her data to Confusa

Implements [Arc::SAML2SSOHTTPClient](#).

5.117.1.2 [MCC_Status Arc::HakaClient::processIdP2Confusa \(\) \[protected, virtual\]](#)

Redirects the user back from identity provider to the Confusa SP

Implements [Arc::SAML2SSOHTTPClient](#).

5.117.1.3 **MCC_Status** `Arc::HakaClient::processIdPLogin (const std::string username, const std::string password) [protected, virtual]`

Actual identity provider parsers for next three methods implemented in subdirectory idp/
Parse identity provider login page and submit username and password in the pre-
sioned way

Implements [Arc::SAML2SSOHTTPClient](#).

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

- `HakaClient.h`

5.118 Arc::FileAccess::header_t Struct Reference

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

- `FileAccess.h`

5.119 Arc::HTTPClientInfo Struct Reference

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

- `ClientInterface.h`

5.120 Arc::InfoCache Class Reference

Stores XML document in filesystem split into parts.

```
#include <InfoCache.h>
```

Public Member Functions

- `InfoCache (const Config &cfg, const std::string &service_id)`

5.120.1 Detailed Description

Stores XML document in filesystem split into parts.

5.120.2 Constructor & Destructor Documentation

5.120.2.1 Arc::InfoCache::InfoCache (const Config & cfg, const std::string & service_id)

Creates object according to configuration (see InfoCacheConfig.xsd)

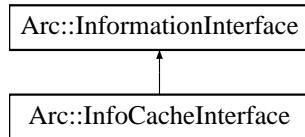
XML configuration is passed in cfg. Argument service_id is used to distinguish between various documents stored under same path - corresponding files will be stored in sub-directory with service_id name.

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

- InfoCache.h

5.121 Arc::InfoCacheInterface Class Reference

Inheritance diagram for Arc::InfoCacheInterface:



Protected Member Functions

- virtual void [Get](#) (const std::list< std::string > &path, [XMLNodeContainer](#) &result)

5.121.1 Member Function Documentation

5.121.1.1 virtual void Arc::InfoCacheInterface::Get (const std::list< std::string > & path, XMLNodeContainer & result) [protected, virtual]

This method is called by this object's Process method. Real implementation of this class should return (sub)tree of XML document. This method may be called multiple times per single Process call. Here is a set on XML element names specifying how to reach requested node(s).

Reimplemented from [Arc::InformationInterface](#).

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

- InfoCache.h

5.122 Arc::InfoFilter Class Reference

Filters information document according to identity of requestor.

```
#include <InfoFilter.h>
```

Public Member Functions

- `InfoFilter (MessageAuth &id)`
- `bool Filter (XMLNode doc) const`
- `bool Filter (XMLNode doc, const InfoFilterPolicies &policies, const NS &ns) const`

5.122.1 Detailed Description

Filters information document according to identity of requestor.

Identity is compared to policies stored inside information document and external ones. Parts of document which do not pass policy evaluation are removed.

5.122.2 Constructor & Destructor Documentation

5.122.2.1 `Arc::InfoFilter::InfoFilter (MessageAuth & id)`

Creates object and associates identity.

Associated identity is not copied, hence passed argument must not be destroyed while this method is used.

5.122.3 Member Function Documentation

5.122.3.1 `bool Arc::InfoFilter::Filter (XMLNode doc) const`

Filter information document according to internal policies.

In provided document all policies and nodes which have their policies evaluated to negative result are removed.

5.122.3.2 `bool Arc::InfoFilter::Filter (XMLNode doc, const InfoFilterPolicies & policies, const NS & ns) const`

Filter information document according to internal and external policies.

In provided document all policies and nodes which have their policies evaluated to negative result are removed. External policies are provided in policies argument. First element of every pair is XPath defining to which XML node policy must be applied. Second element is policy itself. Argument ns defines XML namespaces for XPath evaluation.

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

- `InfoFilter.h`

5.123 Arc::InfoRegister Class Reference

Registration to ISIS interface.

```
#include <InfoRegister.h>
```

5.123.1 Detailed Description

Registration to ISIS interface.

This class represents service registering to Information Indexing [Service](#). It does not perform registration itself. It only collects configuration information. Configuration is as described in InfoRegisterConfig.xsd for element InfoRegistration.

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

- InfoRegister.h

5.124 Arc::InfoRegisterContainer Class Reference

```
#include <InfoRegister.h>
```

Public Member Functions

- [InfoRegistrar * addRegistrar \(XMLNode doc\)](#)
- void [addService \(InfoRegister *reg, const std::list< std::string > &ids, XMLNode cfg=XMLNode\(\)\)](#)
- void [removeService \(InfoRegister *reg\)](#)

5.124.1 Detailed Description

Singleton class for scanning configuration and storing references to registration elements.

5.124.2 Member Function Documentation

5.124.2.1 [InfoRegistrar* Arc::InfoRegisterContainer::addRegistrar \(XMLNode doc \)](#)

Adds ISISes to list of handled services.

Supplied configuration document is scanned for [InfoRegistrar](#) elements and those are turned into [InfoRegistrar](#) classes for handling connection to ISIS service each.

5.124.2.2 void Arc::InfoRegisterContainer::addService (InfoRegister * reg, const std::list< std::string > & ids, XMLNode cfg = XMLNode ())

Adds service to list of handled.

This method must be called first time after last addRegistrar was called - services will be only associated with ISISes which are already added. Argument ids contains list of ISIS identifiers to which service is associated. If ids is empty then service is associated to all ISISes currently added. If argument cfg is available and no ISISes are configured then addRegistrars is called with cfg used as configuration document.

5.124.2.3 void Arc::InfoRegisterContainer::removeService (InfoRegister * reg)

This method must be called if service being destroyed.

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

- InfoRegister.h

5.125 Arc::InfoRegisters Class Reference

Handling multiple registrations to ISISes.

```
#include <InfoRegister.h>
```

Public Member Functions

- [InfoRegisters \(XMLNode &cfg, Service *service_\)](#)

5.125.1 Detailed Description

Handling multiple registrations to ISISes.

5.125.2 Constructor & Destructor Documentation

5.125.2.1 Arc::InfoRegisters::InfoRegisters (XMLNode & cfg, Service * service_)

Constructor creates [InfoRegister](#) objects according to configuration.

Inside cfg elements InfoRegistration are found and for each corresponding [InfoRegister](#) object is created. Those objects are destroyed in destructor of this class.

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

- InfoRegister.h

5.126 Arc::InfoRegistrar Class Reference

Registration process associated with particular ISIS.

```
#include <InfoRegister.h>
```

Public Member Functions

- void `registration` (void)
- bool `addService` (`InfoRegister *`, `XMLNode &`)
- bool `removeService` (`InfoRegister *`)

5.126.1 Detailed Description

Registration process associated with particular ISIS.

Instance of this class starts thread which takes care passing information about associated services to ISIS service defined in configuration. Configuration is as described in InfoRegister.xsd for element `InfoRegistrar`.

5.126.2 Member Function Documentation

5.126.2.1 bool Arc::InfoRegistrar::addService (`InfoRegister *`, `XMLNode &`)

Adds new service to list of handled services.

`Service` is described by it's `InfoRegister` object which must be valid as long as this object is functional.

5.126.2.2 void Arc::InfoRegistrar::registration (`void`)

Performs registration in a loop.

Never exits unless there is a critical error or requested by destructor.

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

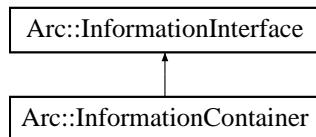
- `InfoRegister.h`

5.127 Arc::InformationContainer Class Reference

Information System document container and processor.

```
#include <InformationInterface.h>
```

Inheritance diagram for Arc::InformationContainer:



Public Member Functions

- [InformationContainer \(XMLNode doc, bool copy=false\)](#)
- [XMLNode Acquire \(void\)](#)
- void [Assign \(XMLNode doc, bool copy=false\)](#)

Protected Member Functions

- virtual void [Get \(const std::list< std::string > &path, XMLNodeContainer &result\)](#)

Protected Attributes

- [XMLNode doc_](#)

5.127.1 Detailed Description

Information System document container and processor.

This class inherits form [InformationInterface](#) and offers container for storing informational XML document.

5.127.2 Constructor & Destructor Documentation

5.127.2.1 Arc::InformationContainer::InformationContainer (XMLNode doc, bool copy = false)

Creates an instance with XML document . If is true this method makes a copy of for internal use.

5.127.3 Member Function Documentation

5.127.3.1 XMLNode Arc::InformationContainer::Acquire (void)

Get a lock on contained XML document. To be used in multi-threaded environment. Do not forget to release it with Release()

5.127.3.2 void Arc::InformationContainer::Assign (XMLNode doc, bool copy = false)

Replaces internal XML document with . If is true this method makes a copy of for internal use.

5.127.3.3 virtual void Arc::InformationContainer::Get (const std::list< std::string > & path, XMLNodeContainer & result) [protected, virtual]

This method is called by this object's Process method. Real implementation of this class should return (sub)tree of XML document. This method may be called multiple times per single Process call. Here is a set on XML element names specifying how to reach requested node(s).

Reimplemented from [Arc::InformationInterface](#).

5.127.4 Field Documentation

5.127.4.1 XMLNode Arc::InformationContainer::doc_ [protected]

Either link or container of XML document

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

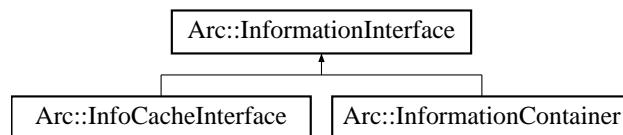
- [InformationInterface.h](#)

5.128 Arc::InformationInterface Class Reference

Information System message processor.

```
#include <InformationInterface.h>
```

Inheritance diagram for Arc::InformationInterface:



Public Member Functions

- [InformationInterface \(bool safe=true\)](#)

Protected Member Functions

- virtual void [Get](#) (const std::list< std::string > &path, [XMLNodeContainer](#) &result)

Protected Attributes

- Glib::Mutex [lock_](#)

5.128.1 Detailed Description

Information System message processor.

This class provides callback for 2 operations of WS-ResourceProperties and convenient parsing/generation of corresponding SOAP messages. In a future it may extend range of supported specifications.

5.128.2 Constructor & Destructor Documentation

5.128.2.1 Arc::InformationInterface::InformationInterface (bool *safe* = `true`)

Constructor. If 'safe' is true all calls to Get will be locked.

5.128.3 Member Function Documentation

5.128.3.1 virtual void Arc::InformationInterface::Get (const std::list< std::string > & *path*, XMLNodeContainer & *result*) [protected, virtual]

This method is called by this object's Process method. Real implementation of this class should return (sub)tree of XML document. This method may be called multiple times per single Process call. Here is a set on XML element names specifying how to reach requested node(s).

Reimplemented in [Arc::InfoCacheInterface](#), and [Arc::InformationContainer](#).

5.128.4 Field Documentation

5.128.4.1 Glib::Mutex Arc::InformationInterface::lock_ [protected]

Mutex used to protect access to Get methods in multi-threaded env.

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

- InformationInterface.h

5.129 Arc::InformationRequest Class Reference

Request for information in InfoSystem.

```
#include <InformationInterface.h>
```

Public Member Functions

- [InformationRequest \(void\)](#)
- [InformationRequest \(const std::list< std::string > &path\)](#)
- [InformationRequest \(const std::list< std::list< std::string > > &paths\)](#)
- [InformationRequest \(XMLNode query\)](#)
- [SOAPEnvelope * SOAP \(void\)](#)

5.129.1 Detailed Description

Request for information in InfoSystem.

This is a convenience wrapper creating proper WS-ResourceProperties request targeted InfoSystem interface of service.

5.129.2 Constructor & Destructor Documentation

5.129.2.1 Arc::InformationRequest::InformationRequest (void)

Dummy constructor

5.129.2.2 Arc::InformationRequest::InformationRequest (const std::list< std::string > & path)

Request for attribute specified by elements of path. Currently only first element is used.

5.129.2.3 Arc::InformationRequest::InformationRequest (const std::list< std::list< std::string > > & paths)

Request for attribute specified by elements of paths. Currently only first element of every path is used.

5.129.2.4 Arc::InformationRequest::InformationRequest (XMLNode query)

Request for attributes specified by XPath query.

5.129.3 Member Function Documentation

5.129.3.1 SOAPEnvelope* Arc::InformationRequest::SOAP (void)

Returns generated SOAP message

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

- [InformationInterface.h](#)

5.130 Arc::InformationResponse Class Reference

Informational response from InfoSystem.

```
#include <InformationInterface.h>
```

Public Member Functions

- [InformationResponse \(SOAPEnvelope &soap\)](#)
- std::list< [XMLNode](#) > [Result](#) (void)

5.130.1 Detailed Description

Informational response from InfoSystem.

This is a convenience wrapper analyzing WS-ResourceProperties response from InfoSystem interface of service.

5.130.2 Constructor & Destructor Documentation

5.130.2.1 Arc::InformationResponse::InformationResponse (SOAPEnvelope & soap)

Constructor parses WS-ResourceProperties response. Provided SOAPEnvelope object must be valid as long as this object is in use.

5.130.3 Member Function Documentation

5.130.3.1 std::list<XMLNode> Arc::InformationResponse::Result (void)

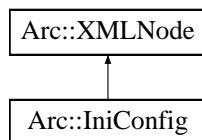
Returns set of attributes which were in SOAP message passed to constructor.

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

- [InformationInterface.h](#)

5.131 Arc::IniConfig Class Reference

Inheritance diagram for Arc::IniConfig:



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

- IniConfig.h

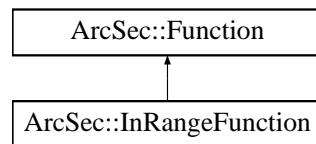
5.132 Arc::initializeCredentialsType Class Reference

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

- UserConfig.h

5.133 ArcSec::InRangeFunction Class Reference

Inheritance diagram for ArcSec::InRangeFunction:



Public Member Functions

- virtual [AttributeValue](#) * [evaluate](#) ([AttributeValue](#) *[arg0](#), [AttributeValue](#) *[arg1](#), bool [check_id=true](#))
- virtual std::list<[AttributeValue](#) * > [evaluate](#) (std::list<[AttributeValue](#) * > [args](#), bool [check_id=true](#))

5.133.1 Member Function Documentation

5.133.1.1 virtual [AttributeValue](#)* ArcSec::InRangeFunction::evaluate ([AttributeValue](#) * [arg0](#), [AttributeValue](#) * [arg1](#), bool [check_id = true](#)) [virtual]

Evaluate two [AttributeValue](#) objects, and return one [AttributeValue](#) object

Implements [ArcSec::Function](#).

5.133.1.2 virtual std::list<[AttributeValue](#)*> ArcSec::InRangeFunction::evaluate (std::list<[AttributeValue](#) * > [args](#), bool [check_id = true](#)) [virtual]

Evaluate a list of [AttributeValue](#) objects, and return a list of Attribute objects

Implements [ArcSec::Function](#).

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

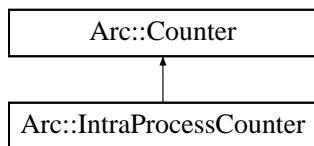
- InRangeFunction.h

5.134 Arc::IntraProcessCounter Class Reference

A class for counters used by threads within a single process.

```
#include <IntraProcessCounter.h>
```

Inheritance diagram for Arc::IntraProcessCounter:



Public Member Functions

- [IntraProcessCounter \(int limit, int excess\)](#)
- virtual [~IntraProcessCounter \(\)](#)
- virtual int [getLimit \(\)](#)
- virtual int [setLimit \(int newLimit\)](#)
- virtual int [changeLimit \(int amount\)](#)
- virtual int [getExcess \(\)](#)
- virtual int [setExcess \(int newExcess\)](#)
- virtual int [changeExcess \(int amount\)](#)
- virtual int [getValue \(\)](#)
- virtual [CounterTicket reserve \(int amount=1, Glib::TimeVal duration=ETERNAL, bool prioritized=false, Glib::TimeVal timeOut=ETERNAL\)](#)

Protected Member Functions

- virtual void [cancel \(IDType reservationID\)](#)
- virtual void [extend \(IDType &reservationID, Glib::TimeVal &expiryTime, Glib::TimeVal duration=ETERNAL\)](#)

5.134.1 Detailed Description

A class for counters used by threads within a single process.

This is a class for shared among different threads within a single process. See the [Counter](#) class for further information about counters and examples of usage.

5.134.2 Constructor & Destructor Documentation

5.134.2.1 Arc::IntraProcessCounter::IntraProcessCounter (int *limit*, int *excess*)

Creates an [IntraProcessCounter](#) with specified limit and excess.

This constructor creates a counter with the specified limit (amount of resources available for reservation) and excess limit (an extra amount of resources that may be used for prioritized reservations).

Parameters

<i>limit</i>	The limit of the counter.
<i>excess</i>	The excess limit of the counter.

5.134.2.2 virtual Arc::IntraProcessCounter::~IntraProcessCounter () [virtual]

Destructor.

This is the destructor of the [IntraProcessCounter](#) class. Does not need to do anything.

5.134.3 Member Function Documentation

5.134.3.1 virtual void Arc::IntraProcessCounter::cancel (IDType *reservationID*) [protected, virtual]

Cancellation of a reservation.

This method cancels a reservation. It is called by the [CounterTicket](#) that corresponds to the reservation.

Parameters

<i>reservationID</i>	The identity number (key) of the reservation to cancel.
----------------------	---

Implements [Arc::Counter](#).

5.134.3.2 virtual int Arc::IntraProcessCounter::changeExcess (int *amount*) [virtual]

Changes the excess limit of the counter.

Changes the excess limit of the counter by adding a certain amount to the current excess limit.

Parameters

<i>amount</i>	The amount by which to change the excess limit.
---------------	---

Returns

The new excess limit.

Implements [Arc::Counter](#).

5.134.3.3 virtual int Arc::IntraProcessCounter::changeLimit (int *amount*) [virtual]

Changes the limit of the counter.

Changes the limit of the counter by adding a certain amount to the current limit.

Parameters

<i>amount</i>	The amount by which to change the limit.
---------------	--

Returns

The new limit.

Implements [Arc::Counter](#).

5.134.3.4 virtual void Arc::IntraProcessCounter::extend (IDType & *reservationID*,
Glib::TimeVal & *expiryTime*, Glib::TimeVal *duration* = ETERNAL)
[protected, virtual]

Extension of a reservation.

This method extends a reservation. It is called by the [CounterTicket](#) that corresponds to the reservation.

Parameters

<i>reservationID</i>	Used for input as well as output. Contains the identification number of the original reservation on entry and the new identification number of the extended reservation on exit.
<i>expiryTime</i>	Used for input as well as output. Contains the expiry time of the original reservation on entry and the new expiry time of the extended reservation on exit.
<i>duration</i>	The time by which to extend the reservation. The new expiration time is computed based on the current time, NOT the previous expiration time.

Implements [Arc::Counter](#).

5.134.3.5 virtual int Arc::IntraProcessCounter::getExcess () [virtual]

Returns the excess limit of the counter.

Returns the excess limit of the counter, i.e. by how much the usual limit may be exceeded by prioritized reservations.

Returns

The excess limit.

Implements [Arc::Counter](#).

5.134.3.6 virtual int Arc::IntraProcessCounter::getLimit() [virtual]

Returns the current limit of the counter.

This method returns the current limit of the counter, i.e. how many units can be reserved simultaneously by different threads without claiming high priority.

Returns

The current limit of the counter.

Implements [Arc::Counter](#).

5.134.3.7 virtual int Arc::IntraProcessCounter::getValue() [virtual]

Returns the current value of the counter.

Returns the current value of the counter, i.e. the number of unreserved units. Initially, the value is equal to the limit of the counter. When a reservation is made, the the value is decreased. Normally, the value should never be negative, but this may happen if there are prioritized reservations. It can also happen if the limit is decreased after some reservations have been made, since reservations are never revoked.

Returns

The current value of the counter.

Implements [Arc::Counter](#).

5.134.3.8 virtual CounterTicket Arc::IntraProcessCounter::reserve(int amount = 1, Glib::TimeVal duration = ETERNAL, bool prioritized = false, Glib::TimeVal timeOut = ETERNAL) [virtual]

Makes a reservation from the counter.

This method makes a reservation from the counter. If the current value of the counter is too low to allow for the reservation, the method blocks until the reservation is possible or times out.

Parameters

<i>amount</i>	The amount to reserve, default value is 1.
<i>duration</i>	The duration of a self expiring reservation, default is that it lasts forever.
<i>prioritized</i>	Whether this reservation is prioritized and thus allowed to use the excess limit.
<i>timeOut</i>	The maximum time to block if the value of the counter is too low, default is to allow "eternal" blocking.

Returns

A [CounterTicket](#) that can be queried about the status of the reservation as well as for cancellations and extensions.

Implements [Arc::Counter](#).

5.134.3.9 virtual int Arc::IntraProcessCounter::setExcess (int *newExcess*) [virtual]

Sets the excess limit of the counter.

This method sets a new excess limit for the counter.

Parameters

<i>newExcess</i>	The new excess limit, an absolute number.
------------------	---

Returns

The new excess limit.

Implements [Arc::Counter](#).

5.134.3.10 virtual int Arc::IntraProcessCounter::setLimit (int *newLimit*) [virtual]

Sets the limit of the counter.

This method sets a new limit for the counter.

Parameters

<i>newLimit</i>	The new limit, an absolute number.
-----------------	------------------------------------

Returns

The new limit.

Implements [Arc::Counter](#).

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

- IntraProcessCounter.h

5.135 Arc::ISIS_description Struct Reference

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

- InfoRegister.h

5.136 Arc::IString Class Reference

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

- IString.h

5.137 Arc::JobDescriptionParserLoader::iterator Class Reference

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

- JobDescriptionParser.h

5.138 Arc::Job Class Reference

[Job](#).

```
#include <Job.h>
```

Public Member Functions

- [Job \(\)](#)
- void [Print](#) (bool longlist) const
- void [SaveToStream](#) (std::ostream &out, bool longlist) const
- [Job & operator=](#) (XMLNode job)
- void [ToXML](#) (XMLNode job) const

Static Public Member Functions

- static bool [ReadAllJobsFromFile](#) (const std::string &filename, std::list< [Job](#) > &jobs, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [WriteJobsToTruncatedFile](#) (const std::string &filename, const std::list< [Job](#) > &jobs, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [WriteJobsToFile](#) (const std::string &filename, const std::list< [Job](#) > &jobs, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [WriteJobsToFile](#) (const std::string &filename, const std::list< [Job](#) > &jobs, std::list< const [Job](#) * > &newJobs, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [RemoveJobsFromFile](#) (const std::string &filename, const std::list< [URL](#) > &jobids, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [ReadJobIDsFromFile](#) (const std::string &filename, std::list< std::string > &jobids, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [WriteJobIDToFile](#) (const [URL](#) &jobid, const std::string &filename, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [WriteJobIDsToFile](#) (const std::list< [URL](#) > &jobids, const std::string &filename, unsigned nTries=10, unsigned tryInterval=500000)

5.138.1 Detailed Description

[Job](#).

This class describe a Grid job. Most of the members contained in this class are directly linked to the ComputingActivity defined in the GLUE Specification v. 2.0 (GFD-R-P.147).

5.138.2 Constructor & Destructor Documentation

5.138.2.1 Arc::Job::Job()

Create a [Job](#) object.

Default constructor. Takes no arguments.

5.138.3 Member Function Documentation

5.138.3.1 Job& Arc::Job::operator= (XMLNode *job*)

Set [Job](#) attributes from a [XMLNode](#).

The attributes of the [Job](#) object is set to the values specified in the [XMLNode](#). The [XMLNode](#) should be a ComputingActivity type using the [GLUE2](#) XML hierarchical rendering, see <http://forge.gridforum.org/sf/wiki/do/viewPage/projects.glue-wg/wiki/GLUE2X> for more information. Note that associations are not parsed.

Parameters

<i>job</i>	is a XMLNode of GLUE2 ComputingActivity type.
------------	---

See also

[ToXML](#)

5.138.3.2 void Arc::Job::Print(bool *longlist*) const

DEPRECATED: Print the [Job](#) information to std::cout.

This method is DEPRECATED, use the SaveToStream method instead. Method to print the [Job](#) attributes to std::cout

Parameters

<i>longlist</i>	is boolean for long listing (more details).
-----------------	---

See also

[SaveToStream](#)

**5.138.3.3 static bool Arc::Job::ReadAllJobsFromFile (const std::string & *filename*, std::list<
Job > & *jobs*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000)
[static]**

Read all jobs from file.

This static method will read jobs (in XML format) from the specified file, and they will be stored in the referenced list of jobs. The XML element in the file representing a job should be named "Job", and have the same format as accepted by the [operator=\(XMLNode\)](#) method.

File locking: To avoid simultaneous use (writing and reading) of the file, reading will not be initiated before a lock on the file has been acquired. For this purpose the [FileLock](#) class is used. *nTries* specifies the maximal number of times the method will try to acquire a lock on the file, with an interval of *tryInterval* micro seconds between each attempt. If a lock is not acquired* this method returns false.

The method will also return false if the content of file is not in XML format. Otherwise it returns true.

Parameters

<i>filename</i>	is the filename of the job list to read jobs from.
<i>jobs</i>	is a reference to a list of Job objects, which will be filled with the jobs read from file (cleared before use).
<i>nTries</i>	specifies the maximal number of times the method will try to acquire a lock on file to read.
<i>tryInterval</i>	specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns

true in case of success, otherwise false.

See also

[operator=\(XMLNode\)](#)
[WriteJobsToTruncatedFile](#)
[WriteJobsToFile](#)
[RemoveJobsFromFile](#)
[FileLock](#)
[XMLNode::ReadFromFile](#)

**5.138.3.4 static bool Arc::Job::ReadJobIDsFromFile (const std::string & *filename*, std::list<
std::string > & *jobids*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000)
[static]**

Read a list of [Job](#) IDs from a file, and append them to a list.

This static method will read job IDs from the given file, and append the strings to the string list given as parameter. File locking will be done as described for the [ReadAllJobsFromFile](#) method. It returns false if the file was not readable, true otherwise, even if

there were no IDs in the file. The lines of the file will be trimmed, and lines starting with # will be ignored.

Parameters

<i>filename</i>	is the filename of the jobidfile
<i>jobids</i>	is a list of strings, to which the IDs read from the file will be appended
<i>nTries</i>	specifies the maximal number of times the method will try to acquire a lock on file to read.
<i>tryInterval</i>	specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns

true in case of success, otherwise false.

```
5.138.3.5 static bool Arc::Job::RemoveJobsFromFile ( const std::string & filename, const
std::list< URL > & jobids, unsigned nTries = 10, unsigned tryInterval = 500000 )
[static]
```

Truncate file and write jobs to it.

This static method will remove the jobs having IDFromEndpoint identical to any of those in the passed list jobids. File locking will be done as described for the ReadAllJobsFromFile method. The method will return false if reading from or writing jobs to the file fails. Otherwise it returns true.

Parameters

<i>filename</i>	is the filename of the job list to write jobs to.
<i>jobids</i>	is a list of URL objects which specifies which jobs from the file to remove.
<i>nTries</i>	specifies the maximal number of times the method will try to acquire a lock on file to read.
<i>tryInterval</i>	specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns

true in case of success, otherwise false.

See also

[ReadAllJobsFromFile](#)
[WriteJobsToTruncatedFile](#)
[WriteJobsToFile](#)
[FileLock](#)
[XMLNode::ReadFromFile](#)
[XMLNode::SaveToFile](#)

5.138.3.6 void Arc::Job::SaveToStream (std::ostream & *out*, bool *longlist*) const

Write job information to a std::ostream object.

This method will write job information to the passed std::ostream object. The longlist boolean specifies whether more (true) or less (false) information should be printed.

Parameters

<i>out</i>	is the std::ostream object to print the attributes to.
<i>longlist</i>	is a boolean for switching on long listing (more details).

5.138.3.7 void Arc::Job::ToXML (XMLNode *job*) const

Add job information to a [XMLNode](#).

Child nodes of GLUE ComputingActivity type containing job information of this object will be added to the passed [XMLNode](#).

Parameters

<i>job</i>	is the XMLNode to add job information to in form of GLUE2 ComputingActivity type child nodes.
------------	---

See also

operator=

5.138.3.8 static bool Arc::Job::WriteJobIDsToFile (const std::list< URL > & *jobids*, const std::string & *filename*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000) [static]

Append list of URLs to a file.

This static method will put the ID given as a string, and append it to the given file. File locking will be done as described for the ReadAllJobsFromFile method. It returns false if the file was not writable, true otherwise.

Parameters

<i>jobid</i>	is a list of URL objects to be written to file
<i>filename</i>	is the filename of file, where the URL objects will be appended to.
<i>nTries</i>	specifies the maximal number of times the method will try to acquire a lock on file to read.
<i>tryInterval</i>	specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns

true in case of success, otherwise false.

5.138.3.9 static bool Arc::Job::WriteJobIDToFile (const URL & *jobid*, const std::string & *filename*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000) [static]

Append a jobID to a file.

This static method will put the ID represented by a [URL](#) object, and append it to the given file. File locking will be done as described for the ReadAllJobsFromFile method. It returns false if the file is not writable, true otherwise.

Parameters

<i>jobid</i>	is a jobID as a URL object
<i>filename</i>	is the filename of the jobidfile, where the jobID will be appended
<i>nTries</i>	specifies the maximal number of times the method will try to acquire a lock on file to read.
<i>tryInterval</i>	specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns

true in case of success, otherwise false.

5.138.3.10 static bool Arc::Job::WriteJobsToFile (const std::string & *filename*, const std::list<
Job > & *jobs*, std::list< const Job * > & *newJobs*, unsigned *nTries* = 10,
unsigned *tryInterval* = 500000) [static]

Write jobs to file.

This static method will write (appending) the passed list of jobs to the specified file. Jobs will be written in XML format as returned by the ToXML method, and each job will be contained in a element named "Job". The passed list of jobs must not contain two identical jobs (i.e. IDFromEndpoint identical), if that is the case false will be returned. If on the other hand a job in the list is identical to one in file, the one in file will be overwritten. A pointer (no new) to those jobs from the list which are not in the file will be added to newJobs list, thus these pointers goes out of scope when jobs list goes out of scope. File locking will be done as described for the ReadAllJobsFromFile method. The method will return false if writing jobs to the file fails. Otherwise it returns true.

Parameters

<i>filename</i>	is the filename of the job list to write jobs to.
<i>jobs</i>	is the list of Job objects which should be written to file.
<i>newJobs</i>	is a reference to a list of pointers to Job objects which are not duplicates (cleared before use).
<i>nTries</i>	specifies the maximal number of times the method will try to acquire a lock on file to read.
<i>tryInterval</i>	specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns

true in case of success, otherwise false.

See also

[ToXML](#)
[ReadAllJobsFromFile](#)
[WriteJobsToTruncatedFile](#)
[RemoveJobsFromFile](#)
[FileLock](#)
[XMLNode::SaveToFile](#)

**5.138.3.11 static bool Arc::Job::WriteJobsToFile (const std::string & *filename*, const std::list<
Job > & *jobs*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000)
[static]**

Write jobs to file.

This method is in all respects identical to the [WriteJobsToFile\(const std::string&, const std::list<Job>&, std::list<const Job*>&, unsigned, unsigned\)](#) method, except for the information about new jobs which is disregarded.

See also

[WriteJobsToFile\(const std::string&, const std::list<Job>&, std::list<const Job*>&, unsigned, unsigned\)](#)

**5.138.3.12 static bool Arc::Job::WriteJobsToTruncatedFile (const std::string & *filename*, const std::list< *Job* > & *jobs*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000)
[static]**

Truncate file and write jobs to it.

This static method will write the passed list of jobs to the specified file, but before writing the file will be truncated. Jobs will be written in XML format as returned by the [ToXML](#) method, and each job will be contained in a element named "Job". The passed list of jobs must not contain two identical jobs (i.e. [IDFromEndpoint](#) identical), if that is the case false will be returned. File locking will be done as described for the [ReadAllJobsFromFile](#) method. The method will return false if writing jobs to the file fails. Otherwise it returns true.

Parameters

<i>filename</i>	is the filename of the job list to write jobs to.
<i>jobs</i>	is the list of Job objects which should be written to file.
<i>nTries</i>	specifies the maximal number of times the method will try to acquire a lock on file to read.
<i>tryInterval</i>	specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns

true in case of success, otherwise false.

See also

[ToXML](#)
[ReadAllJobsFromFile](#)
[WriteJobsToFile](#)
[RemoveJobsFromFile](#)
[FileLock](#)
[XMLNode::SaveToFile](#)

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

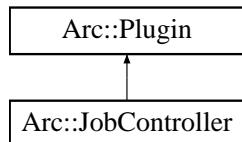
- Job.h

5.139 Arc::JobController Class Reference

Base class for the JobControllers.

```
#include <JobController.h>
```

Inheritance diagram for Arc::JobController:



Public Member Functions

- void [FillJobStore](#) (const [Job](#) &job)
- bool [Cat](#) (const std::list< std::string > &status, const std::string &whichfile)
- bool [Cat](#) (std::ostream &out, const std::list< std::string > &status, const std::string &whichfile)
- bool [PrintJobStatus](#) (const std::list< std::string > &status, bool longlist)
- bool [SaveJobStatusToStream](#) (std::ostream &out, const std::list< std::string > &status, bool longlist)
- bool [Migrate](#) ([TargetGenerator](#) &targetGen, [Broker](#) *broker, const [UserConfig](#) &usercfg, bool forcemigration, std::list< [URL](#) > &migratedJobIDs)

5.139.1 Detailed Description

Base class for the JobControllers.

The [JobController](#) is the base class for middleware specialized derived classes. The [JobController](#) base class is also the implementer of all public functionality that should be offered by the middleware specific specializations. In other words all virtual functions of the [JobController](#) are private. The initialization of a (specialized) [JobController](#) object takes two steps. First the [JobController](#) specialization for the required grid flavour must be loaded by the [JobControllerLoader](#), which sees to that the [JobController](#) receives information about its Grid flavour and the local joblist file containing information about all active jobs (flavour independent). The next step is the filling of the [JobController](#) job pool (JobStore) which is the pool of jobs that the [JobController](#) can manage. Must be specialised for each supported middleware flavour.

5.139.2 Member Function Documentation

5.139.2.1 `bool Arc::JobController::Cat (const std::list< std::string > & status, const std::string & whichfile)`

DEPRECATED: Catenate a log-file to standard out.

This method is DEPRECATED, use the [Cat\(std::ostream&, const std::list<std::string>&, const std::string&\)](#) instead.

This method is not supposed to be overloaded by extending classes.

Parameters

<i>status</i>	a list of strings representing states to be considered.
<i>longlist</i>	a boolean indicating whether verbose job information should be printed.

Returns

This method always returns true.

See also

[Cat\(std::ostream&, const std::list<std::string>&, const std::string&\)](#)
[GetJobInformation](#)
[JobState](#)

5.139.2.2 `bool Arc::JobController::Cat (std::ostream & out, const std::list< std::string > & status, const std::string & whichfile)`

Catenate a output log-file to a std::ostream object.

The method catenates one of the log-files standard out or error, or the job log file from the CE for each of the jobs contained in this object. A file can only be catenated if the location relative to the session directory are set in Job::StdOut, Job::StdErr and Job::LogDir respectively, and if supported so in the specialised ACC module. If the status parameter is non-empty only jobs having a job status specified in this list will be considered. The whichfile parameter specifies what log-file to catenate. Possible values are "stdout", "stderr" and "joblog" respectively specifying standard out, error and job log file.

This method is not supposed to be overloaded by extending classes.

Parameters

<i>status</i>	a list of strings representing states to be considered.
<i>longlist</i>	a boolean indicating whether verbose job information should be printed.

Returns

This method always returns true.

See also

[SaveJobStatusToStream](#)
[GetJobInformation](#)
[JobState](#)

5.139.2.3 bool Arc::JobController::Migrate (TargetGenerator & targetGen, Broker * broker, const UserConfig & usercfg, bool forcemigration, std::list< URL > & migratedJobIDs)

Migrate job from cluster A to Cluster B.

Method to migrate the jobs contained in the jobstore.

Parameters

<i>targetGen</i>	TargetGenerator with targets to migrate the job to.
<i>broker</i>	Broker to be used when selecting target.
<i>forcemigration</i>	boolean which specifies whether a migrated job should persist if the new cluster does not succeed sending a kill/terminate request for the job.

5.139.2.4 bool Arc::JobController::PrintJobStatus (const std::list< std::string > & status, bool longlist)

DEPRECATED: Print job status to std::cout.

This method is DEPRECATED, use the [SaveJobStatusToStream](#) instead.

This method is not supposed to be overloaded by extending classes.

Parameters

<i>status</i>	a list of strings representing states to be considered.
<i>longlist</i>	a boolean indicating whether verbose job information should be printed.

Returns

This method always returns true.

See also

[SaveJobStatusToStream](#)
[GetJobInformation](#)
[JobState](#)

5.139.2.5 `bool Arc::JobController::SaveJobStatusToStream (std::ostream & out, const std::list< std::string > & status, bool longlist)`

Print job status to a std::ostream object.

The job status is printed to a std::ostream object when calling this method. More specifically the [Job::SaveToStream](#) method is called on each of the [Job](#) objects stored in this object, and the boolean argument *longlist* is passed directly to the method indicating whether verbose job status should be printed. The *status* argument is a list of strings each representing a job state ([JobState](#)) which is used to indicate that only jobs with a job state in the list should be considered. If the list *status* is empty all jobs will be considered.

This method is not supposed to be overloaded by extending classes.

Parameters

<i>out</i>	a std::ostream object to direct job status information to.
<i>status</i>	a list of strings representing states to be considered.
<i>longlist</i>	a boolean indicating whether verbose job information should be printed.

Returns

This method always returns true.

See also

[GetJobInformation](#)
[Job::SaveToStream](#)
[JobState](#)

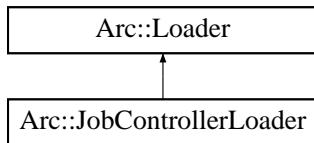
The documentation for this class was generated from the following file:

- JobController.h

5.140 Arc::JobControllerLoader Class Reference

```
#include <JobController.h>
```

Inheritance diagram for Arc::JobControllerLoader:



Public Member Functions

- `JobControllerLoader ()`
- `~JobControllerLoader ()`
- `JobController * load (const std::string &name, const UserConfig &usercfg)`
- `const std::list< JobController * > & GetJobControllers () const`

5.140.1 Detailed Description

Class responsible for loading [JobController](#) plugins. The [JobController](#) objects returned by a [JobControllerLoader](#) must not be used after the [JobControllerLoader](#) goes out of scope.

5.140.2 Constructor & Destructor Documentation

5.140.2.1 Arc::JobControllerLoader::JobControllerLoader ()

Constructor Creates a new [JobControllerLoader](#).

5.140.2.2 Arc::JobControllerLoader::~JobControllerLoader ()

Destructor Calling the destructor destroys all JobControllers loaded by the [JobControllerLoader](#) instance.

5.140.3 Member Function Documentation

5.140.3.1 const std::list<JobController*>& Arc::JobControllerLoader::GetJobControllers () const [inline]

Retrieve the list of loaded JobControllers.

Returns

A reference to the list of JobControllers.

Referenced by [Arc::JobSupervisor::GetJobControllers\(\)](#).

5.140.3.2 JobController* Arc::JobControllerLoader::load (const std::string & name, const UserConfig & usercfg)

Load a new [JobController](#)

Parameters

<i>name</i>	The name of the JobController to load.
<i>usercfg</i>	The UserConfig object for the new JobController .

Returns

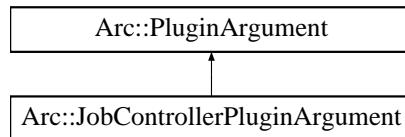
A pointer to the new [JobController](#) (NULL on error).

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

- [JobController.h](#)

5.141 Arc::JobControllerPluginArgument Class Reference

Inheritance diagram for [Arc::JobControllerPluginArgument](#):



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

- [JobController.h](#)

5.142 Arc::JobDescription Class Reference

```
#include <JobDescription.h>
```

Public Member Functions

- [operator bool \(\) const](#)
- [bool Parse \(const std::string &source, const std::string &language="", const std::string &dialect=""\)](#)
- [bool Parse \(const XMLNode &xmlSource\)](#)
- [std::string UnParse \(const std::string &language="nordugrid:jsdl"\) const](#)
- [bool UnParse \(std::string &product, std::string language, const std::string &dialect=""\) const](#)

- const std::string & [GetSourceLanguage \(\) const](#)
- void [Print \(bool longlist=false\) const](#)
- bool [SaveToStream \(std::ostream &out, const std::string &format\) const](#)

Static Public Member Functions

- static bool [Parse \(const std::string &source, std::list< JobDescription > &jobdes-
scs, const std::string &language="", const std::string &dialect ""\)](#)

Data Fields

- std::map< std::string, std::string > [OtherAttributes](#)

5.142.1 Detailed Description

The [JobDescription](#) class is the internal representation of a job description in the ARC-lib. It is structured into a number of other classes/objects which should strictly follow the description given in the job description document <http://svn.nordugrid.org/trac/nordugrid/browser/doc/client/job_description.odt>.

The class consist of a parsing method [JobDescription::Parse](#) which tries to parse the passed source using a number of different parsers. The parser method is complemented by the [JobDescription::UnParse](#) method, a method to generate a job description document in one of the supported formats. Additionally the internal representation is contained in public members which makes it directly accessible and modifiable from outside the scope of the class.

5.142.2 Member Function Documentation

5.142.2.1 const std::string& Arc::JobDescription::GetSourceLanguage () const [inline]

Get input source language.

If this object was created by a [JobDescriptionParser](#), then this method returns a string which indicates the job description language of the parsed source. If not created by a JobDescripionParser the string returned is empty.

Returns

const std::string& source langauge of parsed input source.

5.142.2.2 Arc::JobDescription::operator bool () const

DEPRECATED: Check whether [JobDescription](#) is valid.

The [JobDescription](#) class itself is not able to tell whether its objects are valid or not. Instead when parsing/outputting, [JobDescriptionParser](#) classes checks the validity. Thus the Parse and UnParse methods should be used for this purpose.

5.142.2.3 static bool Arc::JobDescription::Parse (const std::string & *source*, std::list<
JobDescription > & *jobdescs*, const std::string & *language* = " ", const
std::string & *dialect* = " ") [static]

Parse string into [JobDescription](#) objects.

The passed string will be tried parsed into the list of [JobDescription](#) objects. The available specialized [JobDescriptionParser](#) classes will be tried one by one, parsing the string, and if one succeeds the list of [JobDescription](#) objects is filled with the parsed contents and true is returned, otherwise false is returned. If no language specified, each [JobDescriptionParser](#) will try all its supported languages. On the other hand if a language is specified, only the [JobDescriptionParser](#) supporting that language will be tried. A dialect can also be specified, which only has an effect on the parsing if the [JobDescriptionParser](#) supports that dialect.

Parameters

<i>source</i>	
<i>jobdescs</i>	
<i>language</i>	
<i>dialect</i>	

Returns

true if the passed string can be parsed successfully by any of the available parsers.

5.142.2.4 bool Arc::JobDescription::Parse (const std::string & *source*, const std::string &
language = " ", const std::string & *dialect* = " ")

DEPRECATED: Parse source string.

This method is deprecated, use the [Parse\(const std::string&, std::list<JobDescription>&, const std::string&, const std::string&\)](#) method instead.

5.142.2.5 bool Arc::JobDescription::Parse (const XMLNode & *xmlSource*)

DEPRECATED: Parse source string.

This method is deprecated, use the [Parse\(const std::string&, std::list<JobDescription>&, const std::string&, const std::string&\)](#) method instead.

5.142.2.6 void Arc::JobDescription::Print (bool *longlist* = false) const

DEPRECATED: Print all values to standard output.

This method is DEPRECATED, use the [SaveToStream](#) method instead.

Parameters

<i>longlist</i>

See also[SaveToStream](#)

5.142.2.7 `bool Arc::JobDescription::SaveToStream (std::ostream & out, const std::string & format) const`

Print job description to a std::ostream object.

The job description will be written to the passed std::ostream object *out* in the format indicated by the *format* parameter. The *format* parameter should specify the format of one of the job description languages supported by the library. Or by specifying the special "user" or "userlong" format the job description will be written as a attribute/value pair list with respectively less or more attributes.

The mote

Returns

true if writing the job description to the *out* object succeeds, otherwise false.

Parameters

<i>out</i>	a std::ostream reference specifying the ostream to write the job description to.
<i>format</i>	specifies the format the job description should written in.

5.142.2.8 `std::string Arc::JobDescription::UnParse (const std::string & language = "nordugrid:jsdl") const`

DEPRECATED: Output contents in the specified language.

This method is deprecated, use the `UnParse(std::string&, std::string, const std::string&)` method instead.

5.142.2.9 `bool Arc::JobDescription::UnParse (std::string & product, std::string language, const std::string & dialect = " ") const`

Output contents in the specified language.

Parameters

<i>product</i>	
<i>language</i>	
<i>dialect</i>	

Returns

5.142.3 Field Documentation

5.142.3.1 std::map<std::string, std::string> Arc::JobDescription::OtherAttributes

Holds attributes not fitting into this class.

This member is used by [JobDescriptionParser](#) classes to store attribute/value pairs not fitting into attributes stored in this class. The form of the attribute (the key in the map) should be as follows: <language>;<attribute-name> E.g.: "nordugrid:xrsl;hostname".

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

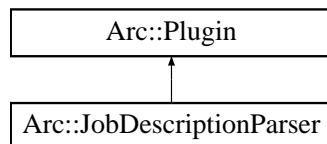
- JobDescription.h

5.143 Arc::JobDescriptionParser Class Reference

Abstract class for the different parsers.

```
#include <JobDescriptionParser.h>
```

Inheritance diagram for Arc::JobDescriptionParser:



5.143.1 Detailed Description

Abstract class for the different parsers.

The [JobDescriptionParser](#) class is abstract which provide a interface for job description parsers. A job description parser should inherit this class and overwrite the JobDescriptionParser::Parse and JobDescriptionParser::UnParse methods.

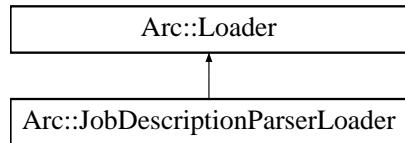
The documentation for this class was generated from the following file:

- JobDescriptionParser.h

5.144 Arc::JobDescriptionParserLoader Class Reference

```
#include <JobDescriptionParser.h>
```

Inheritance diagram for Arc::JobDescriptionParserLoader:



Data Structures

- class [iterator](#)

Public Member Functions

- [JobDescriptionParserLoader \(\)](#)
- [~JobDescriptionParserLoader \(\)](#)
- [JobDescriptionParser * load \(const std::string &name\)](#)
- [const std::list< JobDescriptionParser * > & GetJobDescriptionParsers \(\) const](#)

5.144.1 Detailed Description

Class responsible for loading [JobDescriptionParser](#) plugins. The [JobDescriptionParser](#) objects returned by a [JobDescriptionParserLoader](#) must not be used after the [JobDescriptionParserLoader](#) goes out of scope.

5.144.2 Constructor & Destructor Documentation

5.144.2.1 Arc::JobDescriptionParserLoader::JobDescriptionParserLoader ()

Constructor Creates a new [JobDescriptionParserLoader](#).

5.144.2.2 Arc::JobDescriptionParserLoader::~JobDescriptionParserLoader ()

Destructor Calling the destructor destroys all [JobDescriptionParser](#) object loaded by the [JobDescriptionParserLoader](#) instance.

5.144.3 Member Function Documentation

5.144.3.1 const std::list<[JobDescriptionParser](#)*>& Arc::JobDescriptionParserLoader::GetJobDescriptionParsers () const [inline]

Retrieve the list of loaded [JobDescriptionParser](#) objects.

Returns

A reference to the list of [JobDescriptionParser](#) objects.

5.144.3.2 JobDescriptionParser* Arc::JobDescriptionParserLoader::load (const std::string & name)

Load a new [JobDescriptionParser](#)

Parameters

<code>name</code>	The name of the JobDescriptionParser to load.
-------------------	---

Returns

A pointer to the new [JobDescriptionParser](#) (NULL on error).

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

- [JobDescriptionParser.h](#)

5.145 Arc::JobIdentificationType Class Reference

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

- [JobDescription.h](#)

5.146 Arc::JobState Class Reference

```
#include <JobState.h>
```

Public Member Functions

- bool [IsFinished \(\) const](#)

5.146.1 Detailed Description

ARC general state model. The class comprise the general state model of the ARC-lib, and are herein used to compare job states from the different middlewares supported by the plugin structure of the ARC-lib. Which is why every ACC plugin should contain a class derived from this class. The derived class should consist of a constructor and a mapping function (a [JobStateMap](#)) which maps a `std::string` to a [JobState::StateType](#). An example of a constructor in a plugin could be: `JobStatePlugin::JobStatePluing(const std::string& state) : JobState(state, &pluginStateMap) {}` where `&pluginStateMap` is a reference to the [JobStateMap](#) defined by the derived class.

5.146.2 Member Function Documentation

5.146.2.1 bool Arc::JobState::IsFinished () const [inline]

Check if state is finished.

Returns

true is returned if the StateType is equal to FINISHED, KILLED, FAILED or DELETED, otherwise false is returned.

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

- JobState.h

5.147 Arc::JobSupervisor Class Reference

```
% JobSupervisor class
#include <JobSupervisor.h>
```

Public Member Functions

- `JobSupervisor (const UserConfig &usercfg, const std::list< std::string > &jobs)`
- `JobSupervisor (const UserConfig &usercfg, const std::list< Job > &jobs=std::list< Job >())`
- `bool Get (const std::list< std::string > &statusfilter, const std::string &download-dir, bool usejobname, bool force, std::list< URL > &retrievedJobs)`
- `bool Kill (const std::list< std::string > &statusfilter, std::list< URL > &killedJobs)`
- `bool Renew (const std::list< std::string > &statusfilter, std::list< URL > &renewedJobs)`
- `bool Resume (const std::list< std::string > &statusfilter, std::list< URL > &resumedJobs)`
- `bool Resubmit (const std::list< std::string > &statusfilter, int destination, std::list< Job > &resubmittedJobs, std::list< URL > ¬resubmitted)`
- `bool Migrate (bool forcemigration, std::list< Job > &migratedJobs, std::list< URL > ¬migrated)`
- `std::list< URL > Cancel (const std::list< URL > &jobids, std::list< URL > ¬-cancelled)`
- `std::list< URL > Clean (const std::list< URL > &jobids, std::list< URL > ¬-cleaned)`
- `const std::list< JobController * > & GetJobControllers ()`

5.147.1 Detailed Description

% [JobSupervisor](#) class

The [JobSupervisor](#) class is tool for loading [JobController](#) plugins for managing Grid jobs.

5.147.2 Constructor & Destructor Documentation

5.147.2.1 [Arc::JobSupervisor::JobSupervisor \(const UserConfig & usercfg, const std::list< std::string > & jobs \)](#)

Create a [JobSupervisor](#) object.

Default constructor to create a [JobSupervisor](#). Automatically loads [JobController](#) plug-ins based upon the input jobids.

Parameters

<i>usercfg</i>	Reference to UserConfig object with information about user credentials and joblistfile.
<i>jobs</i>	List of jobs(jobid or job name) to be managed.

5.147.2.2 [Arc::JobSupervisor::JobSupervisor \(const UserConfig & usercfg, const std::list< Job > & jobs = std::list< Job >\(\) \)](#)

Create a [JobSupervisor](#).

The list of [Job](#) objects passed to the constructor will be managed by this [JobSupervisor](#), through the [JobController](#) class. It is important that the Flavour member of each [Job](#) object is set and correspond to the [JobController](#) plugin which are capable of managing that specific job. The [JobController](#) plugin will be loaded using the [JobControllerLoader](#) class, loading a plugin of type "HED:JobController" and name specified by the Flavour member, and the a reference to the [UserConfig](#) object *usercfg* will be passed to the plugin. Additionally a reference to the [UserConfig](#) object *usercfg* will be stored, thus *usercfg* must exist throughout the scope of the created object. If the Flavour member of a [Job](#) object is unset, a VERBOSE log message will be reported and that [Job](#) object will be ignored. If the [JobController](#) plugin for a given Flavour cannot be loaded, a WARNING log message will be reported and any [Job](#) object with that Flavour will be ignored. If loading of a specific plugin failed, that plugin will not be tried loaded for subsequent [Job](#) objects requiring that plugin. [Job](#) objects, for which the corresponding [JobController](#) plugin loaded successfully, will be added to that plugin using the [JobController::FillJobStore\(const Job&\)](#) method.

Parameters

<i>usercfg</i>	UserConfig object to pass to JobController plugins and to use in member methods.
<i>jobs</i>	List of Job objects which will be managed by the created object.

5.147.3 Member Function Documentation

5.147.3.1 `std::list<URL> Arc::JobSupervisor::Cancel (const std::list< URL > & jobids,
std::list< URL > & notcancelled)`

Cancel jobs.

This method will request cancellation of jobs, identified by their IDFromEndpoint member, for which that `URL` is equal to any in the `jobids` list. Only jobs corresponding to a `Job` object managed by this `JobSupervisor` will be considered for cancellation. `Job` objects not in a valid state (see `JobState`) will not be considered, and the IDFromEndpoint URLs of those objects will be appended to the `notcancelled` `URL` list. For jobs not in a finished state (see `JobState::IsFinished`), the `JobController::Cancel` method will be called, passing the corresponding `Job` object, in order to cancel the job. If the `JobController::Cancel` call succeeds or if the job is in a finished state the IDFromEndpoint `URL` will be appended to the list to be returned. If the `JobController::Cancel` call fails the IDFromEndpoint `URL` is appended to the `notkilled` `URL` list.

Note: If there is any `URL` in the `jobids` list for which there is no corresponding `Job` object, then the size of the returned list plus the size of the `notcancelled` list will not equal that of the `jobids` list.

Parameters

<code>jobids</code>	List of <code>Job::IDFromEndpoint</code> <code>URL</code> objects for which a corresponding job, managed by this <code>JobSupervisor</code> should be cancelled.
<code>notcancelled</code>	List of <code>Job::IDFromEndpoint</code> <code>URL</code> objects for which the corresponding job were not cancelled.

Returns

The list of `Job::IDFromEndpoint` `URL` objects of successfully cancelled or finished jobs is returned.

5.147.3.2 `std::list<URL> Arc::JobSupervisor::Clean (const std::list< URL > & jobids,
std::list< URL > & notcleaned)`

Clean jobs.

This method will request cleaning of jobs, identified by their IDFromEndpoint member, for which that `URL` is equal to any in the `jobids` list. Only jobs corresponding to a `Job` object managed by this `JobSupervisor` will be considered for cleaning. `Job` objects not in a valid state (see `JobState`) will not be considered, and the IDFromEndpoint URLs of those objects will be appended to the `notcleaned` `URL` list, otherwise the `JobController::Clean` method will be called, passing the corresponding `Job` object, in order to clean the job. If that method fails the IDFromEndpoint `URL` of the `Job` object will be appended to the `notcleaned` `URL` list, and if it succeeds the IDFromEndpoint `URL` will be appended to the list of `URL` objects to be returned.

Note: If there is any `URL` in the `jobids` list for which there is no corresponding `Job` object, then the size of the returned list plus the size of the `notcleaned` list will not equal that of the `jobids` list.

Parameters

<i>jobids</i>	List of Job::IDFromEndpoint URL objects for which a corresponding job, managed by this JobSupervisor should be cleaned.
<i>notcleaned</i>	List of Job::IDFromEndpoint URL objects for which the corresponding job were not cleaned.

Returns

The list of Job::IDFromEndpoint [URL](#) objects of successfully cleaned jobs is returned.

5.147.3.3 bool Arc::JobSupervisor::Get (const std::list< std::string > & *statusfilter*, const std::string & *downloadaddir*, bool *usejobname*, bool *force*, std::list< [URL](#) > & *retrievedJobs*)

Retrieve job output files.

This method retrieves output files of jobs managed by this [JobSupervisor](#).

Before identifying jobs for which to retrieve output files, the JobController::GetJobInformation method is called for each loaded [JobController](#) in order to retrieve the most up to date job information. If an empty status-filter is specified, all jobs managed by this [JobSupervisor](#) will be considered for retrieval, except jobs in the undefined state (see [JobState](#)). If the status-filter is not empty, then only jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter will be considered, except jobs in the undefined state. Jobs in the state [JobState::DELETED](#) and unfinished jobs (see [JobState::IsFinished](#)) will also not be considered.

For each of the jobs considered for retrieval, the files will be downloaded to a directory named either as the last part of the job ID or the job name, which is determined by the 'usejobname' argument. The download directories will be located in the directory specified by the 'downloadaddir' argument, as either a relative or absolute path. If the 'force' argument is set to 'true', and a download directory for a given job already exist it will be overwritten, otherwise files for that job will not be downloaded. This method calls the JobController::GetJob method in order to download jobs, and if a job is successfully retrieved the job ID will be appended to the 'retrievedJobs' list. If all jobs are successfully retrieved this method returns true, otherwise false.

Parameters

<i>statusfilter</i>	list of job status used for filtering jobs.
<i>downloadaddir</i>	specifies the path to in which job download directories will be located.
<i>usejobname</i>	specifies whether to use the job name or job ID as directory name to store job output files in.
<i>force</i>	indicates whether existing job directories should be overwritten or not.
<i>retrieved-Jobs</i>	job IDs of successfully retrieved jobs will be appended to this list.

See also

[JobController::GetJob](#).

Returns

true if all jobs are successfully retrieved, otherwise false.

5.147.3.4 const std::list<JobController*>& Arc::JobSupervisor::GetJobControllers() [inline]

Get list of JobControllers.

Method to get the list of JobControllers loaded by constructor.

References Arc::JobControllerLoader::GetJobControllers().

5.147.3.5 bool Arc::JobSupervisor::Kill(const std::list< std::string > & statusfilter, std::list< URL > & killedJobs)

Kill jobs.

This method kills jobs managed by this [JobSupervisor](#).

Before identifying jobs to kill, the JobController::GetJobInformation method is called for each loaded [JobController](#) in order to retrieve the most up to date job information.

Since jobs in the JobState::DELETED, JobState::FINISHED, JobState::KILLED or JobState::FAILED states is already in a terminal state, a terminate action will not be send for those. Also no terminate action will be send for jobs in the JobState::UNDEFINED state, since job information is not available. If the status-filter is non-empty, a terminate action will only be send to jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter, excluding the states mentioned above.

For each job to be killed, the specialized JobController::CancelJob method is called and is responsible for terminating the given job. If the method fails to terminate a job, this method will return false, otherwise true is returned. The job ID of successfully terminated jobs will be appended to the passed killedJobs list.

Parameters

<i>statusfilter</i>	list of job status used for filtering jobs.
<i>killedJobs</i>	list of URLs which to append job IDs of successfully terminated jobs to.

See also

[JobController::CancelJob](#).

Returns

true if all jobs terminated successfully, otherwise false.

5.147.3.6 bool Arc::JobSupervisor::Migrate(bool forcemigration, std::list< Job > & migratedJobs, std::list< URL > & notmigrated)

Migrate jobs.

Jobs managed by this [JobSupervisor](#) will be migrated when invoking this method, that is the job description of a job will be tried obtained, and if successful a job migration request will be sent, based on that job description.

Before identifying jobs to be migrated, the `JobController::GetJobInformation` method is called for each loaded [JobController](#) in order to retrieve the most up to date job information. Only jobs for which the `State` member of the [Job](#) object has the value `JobState::QUEUEING`, will be considered for migration. Furthermore the job description must be obtained (either locally or remote) and successfully parsed in order for a job to be migrated. If the job description cannot be obtained or parsed an ERROR log message is reported, and the `IDFromEndpoint URL` of the [Job](#) object is appended to the `notmigrated` list. If no jobs have been identified for migration, false will be returned in case ERRORS were reported, otherwise true is returned.

The execution services which can be targeted for migration are those specified in the [UserConfig](#) object of this class, as selected services. Before initiating any job migration request, resource discovery and broker* loading is carried out using the [TargetGenerator](#) and [Broker](#) classes, initialised by the [UserConfig](#) object of this class. If [Broker](#) loading fails, or no [ExecutionTargets](#) are found, an ERROR log message is reported and all `IDFromEndpoint URLs` for job considered for migration will be appended to the `notmigrated` list and then false will be returned.

When the above checks have been carried out successfully, the following is done for each job considered for migration. The `ActivityOldId` member of the [Identification](#) member in the job description will be set to that of the [Job](#) object, and the `IDFromEndpoint URL` will be appended to `ActivityOldId` member of the job description. After that the [Broker](#) object will be used to find a suitable [ExecutionTarget](#) object, and if found a migrate request will tried sent using the `ExecutionTarget::Migrate` method, passing the [UserConfig](#) object of this class. The passed `forcemigration` boolean indicates whether the migration request at the service side should ignore failures in cancelling the existing queuing job. If the request succeeds, the corresponding new [Job](#) object is appended to the `migratedJobs` list. If no suitable [ExecutionTarget](#) objects are found an ERROR log message is reported and the `IDFromEndpoint URL` of the [Job](#) object is appended to the `notmigrated` list. When all jobs have been processed, false is returned if any ERRORS were reported, otherwise true.

Parameters

<code>forcemigration</code>	indicates whether migration should succeed if service fails to cancel the existing queuing job.
<code>migratedJobs</code>	list of Job objects which migrated jobs will be appended to.
<code>notmigrated</code>	list of URL objects which the <code>IDFromEndpoint URL</code> will be appended to.

Returns

false if any error is encountered, otherwise true.

**5.147.3.7 bool Arc::JobSupervisor::Renew (const std::list< std::string > & *statusfilter*,
std::list< URL > & *renewedJobs*)**

Renew job credentials.

This method renew credentials of the jobs managed by this [JobSupervisor](#).

Before identifying jobs for which to renew credentials, the [JobController::GetJobInformation](#) method is called for each loaded [JobController](#) in order to retrieve the most up to date job information.

Since jobs in the [JobState::DELETED](#), [JobState::FINISHED](#) or [JobState::KILLED](#) states is in a terminal state credentials for those jobs will not be renewed. Also jobs in the [JobState::UNDEFINED](#) state will not get their credentials renewed, since job information is not available. The [JobState::FAILED](#) state is also a terminal state, but since jobs in this state can be restarted, credentials for such jobs can be renewed. If the status-filter is non-empty, a renewal of credentials will be done for jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter, excluding the already filtered states as mentioned above.

For each job for which to renew credentials, the specialized [JobController::RenewJob](#) method is called and is responsible for renewing the credentials for the given job. If the method fails to renew job credentials, this method will return false, otherwise true is returned. The job ID of successfully renewed jobs will be appended to the passed *renewedJobs* list.

Parameters

<i>statusfilter</i>	list of job status used for filtering jobs.
<i>renewed-Jobs</i>	list of URLs which to append job IDs to, of jobs for which credentials was successfully renewed.

See also

[JobController::RenewJob](#).

Returns

true if credentials for all jobs were successfully renewed, otherwise false.

**5.147.3.8 bool Arc::JobSupervisor::Resubmit (const std::list< std::string > & *statusfilter*,
int *destination*, std::list< Job > & *resubmittedJobs*, std::list< URL > &
notresubmitted)**

Resubmit jobs.

Jobs managed by this [JobSupervisor](#) will be resubmitted when invoking this method, that is the job description of a job will be tried obtained, and if successful a new job will be submitted.

Before identifying jobs to be resubmitted, the [JobController::GetJobInformation](#) method is called for each loaded [JobController](#) in order to retrieve the most up to date job information. If an empty status-filter is specified, all jobs managed by this [JobSupervisor](#) will

be considered for resubmission, except jobs in the undefined state (see [JobState](#)). If the status-filter is not empty, then only jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter will be considered, except jobs in the undefined state. Jobs for which a job description cannot be obtained and successfully parsed will not be considered and an ERROR log message is reported, and the IDFromEndpoint [URL](#) is appended to the notresubmitted list. [Job](#) descriptions will be tried obtained either from [Job](#) object itself, or fetching them remotely. Furthermore if a [Job](#) object has the LocalInputFiles object set, then the checksum of each of the local input files specified in that object (key) will be calculated and verified to match the checksum LocalInputFiles object (value). If checksums are not matching the job will be filtered, and an ERROR log message is reported and the IDFromEndpoint [URL](#) is appended to the notresubmitted list. If no job have been identified for resubmission, false will be returned if ERRORs were reported, otherwise true is returned.

The destination for jobs is partly determined by the destination parameter. If a value of 1 is specified a job will only be targeted to the execution service (ES) on which it reside. A value of 2 indicates that a job should not be targeted to the ES it currently reside. Specifying any other value will target any ES. The ESs which can be targeted are those specified in the [UserConfig](#) object of this class, as selected services. Before initiating any job submission, resource discovery and broker loading is carried out using the [TargetGenerator](#) and [Broker](#) classes, initialised by the [UserConfig](#) object of this class. If [Broker](#) loading fails, or no ExecutionTargets are found, an ERROR log message is reported and all IDFromEndpoint URLs for job considered for resubmission will be appended to the notresubmitted list and then false will be returned.

When the above checks have been carried out successfully, then the [Broker::Submit](#) method will be invoked for each considered for resubmission. If it fails the IDFromEndpoint [URL](#) for the job is appended to the notresubmitted list, and an ERROR is reported. If submission succeeds the new job represented by a [Job](#) object will be appended to the resubmittedJobs list - it will not be added to this [JobSupervisor](#). The method returns false if ERRORs were reported otherwise true is returned.

Parameters

<i>statusfilter</i>	list of job status used for filtering jobs.
<i>destination</i>	specifies how target destination should be determined (1 = same target, 2 = not same, any other = any target).
<i>resubmitted-Jobs</i>	list of Job objects which resubmitted jobs will be appended to.
<i>notresubmitted-ted</i>	list of URL objects which the IDFromEndpoint URL will be appended to.

Returns

false if any error is encountered, otherwise true.

5.147.3.9 bool Arc::JobSupervisor::Resume (const std::list< std::string > & *statusfilter*, std::list< [URL](#) > & *resumedJobs*)

Resume job.

This method resumes jobs managed by this [JobSupervisor](#).

Before identifying jobs to resume, the [JobController::GetJobInformation](#) method is called for each loaded [JobController](#) in order to retrieve the most up to date job information.

Since jobs in the [JobState::DELETED](#), [JobState::FINISHED](#) or [JobState::KILLED](#) states is in a terminal state credentials for those jobs will not be renewed. Also jobs in the [JobState::UNDEFINED](#) state will not be resumed, since job information is not available. The [JobState::FAILED](#) state is also a terminal state, but jobs in this state are allowed to be restarted. If the status-filter is non-empty, only jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter will be resumed, excluding the already filtered states as mentioned above.

For each job to resume, the specialized [JobController::ResumeJob](#) method is called and is responsible for resuming the particular job. If the method fails to resume a job, this method will return false, otherwise true is returned. The job ID of successfully resumed jobs will be appended to the passed `resumedJobs` list.

Parameters

<code>statusfilter</code>	list of job status used for filtering jobs.
<code>resumeJobs</code>	list of URLs which to append job IDs to, of jobs which was successfully resumed.

See also

[JobController::ResumeJob](#).

Returns

true if all jobs were successfully resumed, otherwise false.

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

- [JobSupervisor.h](#)

5.148 Arc::LoadableModuleDescription Class Reference

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

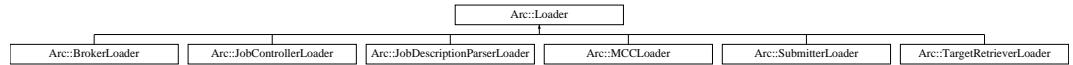
- [ModuleManager.h](#)

5.149 Arc::Loader Class Reference

Plugins loader.

```
#include <Loader.h>
```

Inheritance diagram for Arc::Loader:



Public Member Functions

- [Loader \(XMLNode cfg\)](#)
- [~Loader \(\)](#)

Protected Attributes

- [PluginsFactory * factory_](#)

5.149.1 Detailed Description

Plugins loader.

This class processes XML configuration and loads specified plugins. Accepted configuration is defined by XML schema mcc.xsd. "Plugins" elements are parsed by this class and corresponding libraries are loaded.

5.149.2 Constructor & Destructor Documentation

5.149.2.1 Arc::Loader::Loader (XMLNode cfg)

Constructor that takes whole XML configuration and performs common configuration part

5.149.2.2 Arc::Loader::~Loader ()

Destructor destroys all components created by constructor

5.149.3 Field Documentation

5.149.3.1 PluginsFactory* Arc::Loader::factory_ [protected]

Link to Factory responsible for loading and creation of [Plugin](#) and derived objects

Referenced by Arc::ChainContext::operator PluginsFactory *().

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

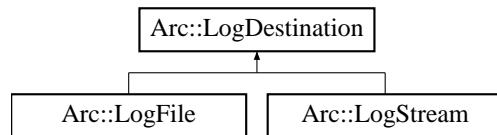
- Loader.h

5.150 Arc::LogDestination Class Reference

A base class for log destinations.

```
#include <Logger.h>
```

Inheritance diagram for Arc::LogDestination:



Public Member Functions

- `virtual void log (const LogMessage &message)=0`

Protected Member Functions

- `LogDestination ()`
- `LogDestination (const std::string &locale)`

5.150.1 Detailed Description

A base class for log destinations.

This class defines an interface for LogDestinations. `LogDestination` objects will typically contain synchronization mechanisms and should therefore never be copied.

5.150.2 Constructor & Destructor Documentation

5.150.2.1 Arc::LogDestination::LogDestination () [protected]

Default constructor.

This destination will use the default locale.

5.150.2.2 Arc::LogDestination::LogDestination (const std::string & locale) [protected]

Constructor with specific locale.

This destination will use the specified locale.

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

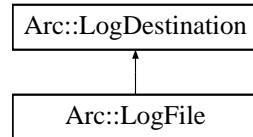
- `Logger.h`

5.151 Arc::LogFile Class Reference

A class for logging to files.

```
#include <Logger.h>
```

Inheritance diagram for Arc::LogFile:



Public Member Functions

- [LogFile](#) (const std::string &path)
- [LogFile](#) (const std::string &path, const std::string &locale)
- void [setMaxSize](#) (int newsize)
- void [setBackups](#) (int newbackup)
- void [setReopen](#) (bool newreopen)
- [operator bool](#) (void)
- bool [operator!](#) (void)
- virtual void [log](#) (const [LogMessage](#) &message)

5.151.1 Detailed Description

A class for logging to files.

This class is used for logging to files. It provides synchronization in order to prevent different LogMessages to appear mixed with each other in the stream. It is possible to limit size of created file. Whenever specified size is exceeded file is deleted and new one is created. Old files may be moved into backup files instead of being deleted. Those files have names same as initial file with additional number suffix - similar to those found in /var/log of many Unix-like systems.

5.151.2 Constructor & Destructor Documentation

5.151.2.1 Arc::LogFile::LogFile (const std::string & path)

Creates a [LogFile](#) connected to a file.

Creates a [LogFile](#) connected to the file located at specified path. In order not to break synchronization, it is important not to connect more than one [LogFile](#) object to a certain file. If file does not exist it will be created.

Parameters

<i>path</i>	The path to file to which to write LogMessages.
-------------	---

5.151.2.2 Arc::LogFile::LogFile (const std::string & path, const std::string & locale)

Creates a [LogFile](#) connected to a file.

Creates a [LogFile](#) connected to the file located at specified path. The output will be localised to the specified locale.

5.151.3 Member Function Documentation**5.151.3.1 virtual void Arc::LogFile::log (const LogMessage & message) [virtual]**

Writes a [LogMessage](#) to the file.

This method writes a [LogMessage](#) to the file that is connected to this [LogFile](#) object. If after writing size of file exceeds one set by [setMaxSize\(\)](#) file is moved to backup and new one is created.

Parameters

<i>message</i>	The LogMessage to write.
----------------	--

Implements [Arc::LogDestination](#).

5.151.3.2 void Arc::LogFile::setBackups (int newbackup)

Set number of backups to store.

Set number of backups to store. When file size exceeds one specified with [setMaxSize\(\)](#) file is closed and moved to one named path.1. If path.1 exists it is moved to path.2 and so on. Number of path.# files is one set in newbackup.

Parameters

<i>newbackup</i>	Number of backup files.
------------------	-------------------------

5.151.3.3 void Arc::LogFile::setMaxSize (int newsze)

Set maximal allowed size of file.

Set maximal allowed size of file. This value is not obeyed exactly. Specified size may be exceeded by amount of one [LogMessage](#). To disable limit specify -1.

Parameters

<i>newsze</i>	Max size of log file.
---------------	-----------------------

5.151.3.4 void Arc::LogFile::setReopen (bool newreopen)

Set file reopen on every write.

Set file reopen on every write. If set to true file is opened before writing every log record and closed afterward.

Parameters

<i>newreopen</i>	If file to be reopened for every log record.
------------------	--

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

- Logger.h

5.152 Arc::Logger Class Reference

A logger class.

```
#include <Logger.h>
```

Public Member Functions

- `Logger (Logger &parent, const std::string &subdomain)`
- `Logger (Logger &parent, const std::string &subdomain, LogLevel threshold)`
- `~Logger ()`
- `void addDestination (LogDestination &destination)`
- `void addDestinations (const std::list< LogDestination * > &destinations)`
- `const std::list< LogDestination * > & getDestinations (void) const`
- `void removeDestinations (void)`
- `void deleteDestinations (void)`
- `void setThreshold (LogLevel threshold)`
- `LogLevel getThreshold () const`
- `void setThreadContext (void)`
- `void msg (LogMessage message)`
- `void msg (LogLevel level, const std::string &str)`

Static Public Member Functions

- `static Logger & getRootLogger ()`
- `static void setThresholdForDomain (LogLevel threshold, const std::list< std::string > &subdomains)`
- `static void setThresholdForDomain (LogLevel threshold, const std::string &domain)`

5.152.1 Detailed Description

A logger class.

This class defines a [Logger](#) to which LogMessages can be sent.

Every [Logger](#) (except for the rootLogger) has a parent [Logger](#). The domain of a [Logger](#) (a string that indicates the origin of LogMessages) is composed by adding a subdomain to the domain of its parent [Logger](#).

A [Logger](#) also has a threshold. Every [LogMessage](#) that have a level that is greater than or equal to the threshold is forwarded to any [LogDestination](#) connected to this [Logger](#) as well as to the parent [Logger](#).

Typical usage of the [Logger](#) class is to declare a global [Logger](#) object for each library/-module/component to be used by all classes and methods there.

5.152.2 Constructor & Destructor Documentation

5.152.2.1 Arc::Logger::Logger ([Logger](#) & parent, const std::string & subdomain)

Creates a logger.

Creates a logger. The threshold is inherited from its parent [Logger](#).

Parameters

<i>parent</i>	The parent Logger of the new Logger .
<i>subdomain</i>	The subdomain of the new logger.

5.152.2.2 Arc::Logger::Logger ([Logger](#) & parent, const std::string & subdomain, LogLevel threshold)

Creates a logger.

Creates a logger.

Parameters

<i>parent</i>	The parent Logger of the new Logger .
<i>subdomain</i>	The subdomain of the new logger.
<i>threshold</i>	The threshold of the new logger.

5.152.2.3 Arc::Logger::~Logger ()

Destroys a logger.

Destructor

5.152.3 Member Function Documentation

5.152.3.1 void Arc::Logger::addDestination (**LogDestination & destination**)

Adds a [LogDestination](#).

Adds a [LogDestination](#) to which to forward LogMessages sent to this logger (if they pass the threshold). Since LogDestinatoins should not be copied, the new [LogDestination](#) is passed by reference and a pointer to it is kept for later use. It is therefore important that the [LogDestination](#) passed to this [Logger](#) exists at least as long as the [Logger](#) itslef.

5.152.3.2 void Arc::Logger::addDestinations (**const std::list< LogDestination * > & destinations**)

Adds LogDestinations.

See [addDestination\(LogDestination& destination\)](#).

Referenced by [DataStaging::DTR::connect_logger\(\)](#).

5.152.3.3 const std::list<LogDestination*>& Arc::Logger::getDestinations (**void**) const

Obtains current LogDestinations.

Returns list of pointers to [LogDestination](#) objects. Returned result refers directly to internal member of [Logger](#) intance. Hence it should not be used after this [Logger](#) is destroyed.

5.152.3.4 static Logger& Arc::Logger::getRootLogger () [static]

The root [Logger](#).

This is the root [Logger](#). It is an ancestor of any other [Logger](#) and allways exists.

5.152.3.5 LogLevel Arc::Logger::getThreshold () const

Returns the threshold.

Returns the threshold.

Returns

The threshold of this [Logger](#).

5.152.3.6 void Arc::Logger::msg (**LogLevel level, const std::string & str**) [inline]

Logs a message text.

Logs a message text string at the specified LogLevel. This is a convenience method to save some typing. It simply creates a [LogMessage](#) and sends it to the other [msg\(\)](#) method.

Parameters

<i>level</i>	The level of the message.
<i>str</i>	The message text.

References [msg\(\)](#).

5.152.3.7 void Arc::Logger::msg (LogMessage *message*)

Sends a [LogMessage](#).

Sends a [LogMessage](#).

Parameters

<i>The</i>	LogMessage to send.
------------	-------------------------------------

Referenced by [msg\(\)](#), and [Arc::stringto\(\)](#).

5.152.3.8 void Arc::Logger::setThreadContext (void)

Creates per-thread context.

Creates new context for this logger which becomes effective for operations initiated by this thread. All new threads started by this one will inherit new context. Context stores current threshold and pointers to destinations. Hence new context is identical to current one. One can modify new context using [setThreshold\(\)](#), [removeDestinations\(\)](#) and [addDestination\(\)](#). All such operations will not affect old context.

5.152.3.9 void Arc::Logger::setThreshold (LogLevel *threshold*)

Sets the threshold.

This method sets the threshold of the [Logger](#). Any message sent to this [Logger](#) that has a level below this threshold will be discarded.

Parameters

<i>The</i>	threshold
------------	-----------

5.152.3.10 static void Arc::Logger::setThresholdForDomain (LogLevel *threshold*, const std::list< std::string > & *subdomains*) [static]

Sets the threshold for domain.

This method sets the default threshold of the domain. All new loggers created with

specified domain will have specified threshold set by default. The subdomains of all loggers in chain are matched against list of provided subdomains.

Parameters

<i>threshold</i>	The threshold
<i>subdomains</i>	The subdomains of all loggers in chain

5.152.3.11 static void Arc::Logger::setThresholdForDomain (LogLevel *threshold*, const std::string & *domain*) [static]

Sets the threshold for domain.

This method sets the default threshold of the domain. All new loggers created with specified domain will have specified threshold set by default. The domain is composed of all subdomains of all loggers in chain by merging them with '.' as separator.

Parameters

<i>threshold</i>	The threshold
<i>domain</i>	The domain of logger

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

- Logger.h

5.153 Arc::LoggerContext Class Reference

Container for logger configuration.

```
#include <Logger.h>
```

5.153.1 Detailed Description

Container for logger configuration.

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

- Logger.h

5.154 Arc::LoggerFormat Struct Reference

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

- Logger.h

5.155 Arc::LogMessage Class Reference

A class for log messages.

```
#include <Logger.h>
```

Public Member Functions

- [LogMessage \(LogLevel level, const IString &message\)](#)
- [LogMessage \(LogLevel level, const IString &message, const std::string &identifier\)](#)
- [LogLevel getLevel \(\) const](#)

Protected Member Functions

- void [setIdentifier \(std::string identifier\)](#)

Friends

- class [Logger](#)
- std::ostream & [operator<< \(std::ostream &os, const LogMessage &message\)](#)

5.155.1 Detailed Description

A class for log messages.

This class is used to represent log messages internally. It contains the time the message was created, its level, from which domain it was sent, an identifier and the message text itself.

5.155.2 Constructor & Destructor Documentation

5.155.2.1 Arc::LogMessage::LogMessage (LogLevel level, const IString & message)

Creates a [LogMessage](#) with the specified level and message text.

This constructor creates a [LogMessage](#) with the specified level and message text. The time is set automatically, the domain is set by the [Logger](#) to which the [LogMessage](#) is sent and the identifier is composed from the process ID and the address of the Thread object corresponding to the calling thread.

Parameters

<i>level</i>	The level of the LogMessage .
<i>message</i>	The message text.

5.155.2.2 Arc::LogMessage::LogMessage (`LogLevel level`, `const IString & message`, `const std::string & identifier`)

Creates a [LogMessage](#) with the specified attributes.

This constructor creates a [LogMessage](#) with the specified level, message text and identifier. The time is set automatically and the domain is set by the [Logger](#) to which the [LogMessage](#) is sent.

Parameters

<i>level</i>	The level of the LogMessage .
<i>message</i>	The message text.
<i>ident</i>	The identifier of the LogMessage .

5.155.3 Member Function Documentation

5.155.3.1 LogLevel Arc::LogMessage::getLevel () const

Returns the level of the [LogMessage](#).

Returns the level of the [LogMessage](#).

Returns

The level of the [LogMessage](#).

5.155.3.2 void Arc::LogMessage::setIdentifier (`std::string identifier`) [protected]

Sets the identifier of the [LogMessage](#).

The purpose of this method is to allow subclasses (in case there are any) to set the identifier of a [LogMessage](#).

Parameters

<i>The</i>	identifier.
------------	-------------

5.155.4 Friends And Related Function Documentation

5.155.4.1 friend class Logger [friend]

The [Logger](#) class is a friend.

The [Logger](#) class must have some privileges (e.g. ability to call the `setDomain()` method), therefore it is a friend.

5.155.4.2 std::ostream& operator<< (std::ostream & os, const LogMessage & message)
[friend]

Printing of LogMessages to ostreams.

Output operator so that LogMessages can be printed conveniently by LogDestinations.

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

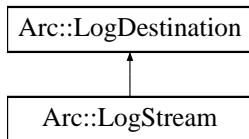
- [Logger.h](#)

5.156 Arc::LogStream Class Reference

A class for logging to ostreams.

```
#include <Logger.h>
```

Inheritance diagram for Arc::LogStream:



Public Member Functions

- [LogStream \(std::ostream &destination\)](#)
- [LogStream \(std::ostream &destination, const std::string &locale\)](#)
- [virtual void log \(const LogMessage &message\)](#)

5.156.1 Detailed Description

A class for logging to ostreams.

This class is used for logging to ostreams (cout, cerr, files). It provides synchronization in order to prevent different LogMessages to appear mixed with each other in the stream. In order not to break the synchronization, LogStreams should never be copied. Therefore the copy constructor and assignment operator are private. Furthermore, it is important to keep a [LogStream](#) object as long as the [Logger](#) to which it has been registered.

5.156.2 Constructor & Destructor Documentation

5.156.2.1 Arc::LogStream::LogStream (std::ostream & destination)

Creates a [LogStream](#) connected to an ostream.

Creates a [LogStream](#) connected to the specified ostream. In order not to break synchronization, it is important not to connect more than one [LogStream](#) object to a certain stream.

Parameters

<i>destination</i>	The ostream to which to erite LogMessages.
--------------------	--

5.156.2.2 Arc::LogStream::LogStream (std::ostream & *destination*, const std::string & *locale*)

Creates a [LogStream](#) connected to an ostream.

Creates a [LogStream](#) connected to the specified ostream. The output will be localised to the specified locale.

5.156.3 Member Function Documentation**5.156.3.1 virtual void Arc::LogStream::log (const LogMessage & *message*) [virtual]**

Writes a [LogMessage](#) to the stream.

This method writes a [LogMessage](#) to the ostream that is connected to this [LogStream](#) object. It is synchronized so that not more than one [LogMessage](#) can be written at a time.

Parameters

<i>message</i>	The LogMessage to write.
----------------	--

Implements [Arc::LogDestination](#).

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

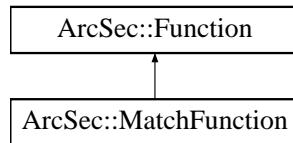
- Logger.h

5.157 ArcSec::MatchFunction Class Reference

Evaluate whether arg1 (value in regular expression) matched arg0 (lable in regular expression)

```
#include <MatchFunction.h>
```

Inheritance diagram for ArcSec::MatchFunction:



Public Member Functions

- virtual `AttributeValue * evaluate (AttributeValue *arg0, AttributeValue *arg1, bool check_id=true)`
- virtual `std::list<AttributeValue * > evaluate (std::list<AttributeValue * > args, bool check_id=true)`

Static Public Member Functions

- static `std::string getFunctionName (std::string datatype)`

5.157.1 Detailed Description

Evaluate whether arg1 (value in regular expression) matched arg0 (label in regular expression)

5.157.2 Member Function Documentation

5.157.2.1 virtual `AttributeValue* ArcSec::MatchFunction::evaluate (AttributeValue * arg0, AttributeValue * arg1, bool check_id = true) [virtual]`

Evaluate two `AttributeValue` objects, and return one `AttributeValue` object

Implements `ArcSec::Function`.

5.157.2.2 virtual `std::list<AttributeValue*> ArcSec::MatchFunction::evaluate (std::list<AttributeValue * > args, bool check_id = true) [virtual]`

Evaluate a list of `AttributeValue` objects, and return a list of Attribute objects

Implements `ArcSec::Function`.

5.157.2.3 static `std::string ArcSec::MatchFunction::getFunctionName (std::string datatype) [static]`

help function to get the FunctionName

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

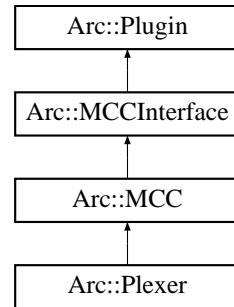
- MatchFunction.h

5.158 Arc::MCC Class Reference

[Message](#) Chain Component - base class for every [MCC](#) plugin.

```
#include <MCC.h>
```

Inheritance diagram for Arc::MCC:



Public Member Functions

- [MCC \(Config *\)](#)
- virtual void [Next \(MCCInterface *next, const std::string &label=""\)](#)
- virtual void [AddSecHandler \(Config *cfg, ArcSec::SecHandler *sechandler, const std::string &label=""\)](#)
- virtual void [Unlink \(\)](#)
- virtual [MCC_Status process \(Message &, Message &\)](#)

Protected Member Functions

- bool [ProcessSecHandlers \(Message &message, const std::string &label=""\) const](#)

Protected Attributes

- std::map< std::string, [MCCInterface *](#) > [next_](#)
- std::map< std::string, std::list< [ArcSec::SecHandler *](#) > > [sechandlers_](#)

Static Protected Attributes

- static [Logger logger](#)

5.158.1 Detailed Description

[Message](#) Chain Component - base class for every [MCC](#) plugin.

This is partially virtual class which defines interface and common functionality for every [MCC](#) plugin needed for managing of component in a chain.

5.158.2 Constructor & Destructor Documentation

5.158.2.1 `Arc::MCC::MCC(Config *) [inline]`

Example constructor - `MCC` takes at least its configuration subtree

5.158.3 Member Function Documentation

5.158.3.1 `virtual void Arc::MCC::AddSecHandler(Config * cfg, ArcSec::SecHandler * sechandler, const std::string & label = " ") [virtual]`

Add security components/handlers to this `MCC`. Security handlers are stacked into a few queues with each queue identified by its label. The queue labelled 'incoming' is executed for every 'request' message after the message is processed by the `MCC` on the service side and before processing on the client side. The queue labelled 'outgoing' is run for response message before it is processed by `MCC` algorithms on the service side and after processing on the client side. Those labels are just a matter of agreement and some MCCs may implement different queues executed at various message processing steps.

5.158.3.2 `virtual void Arc::MCC::Next(MCCInterface * next, const std::string & label = " ") [virtual]`

Add reference to next `MCC` in chain. This method is called by `Loader` for every potentially labeled link to next component which implements `MCCInterface`. If next is NULL corresponding link is removed.

Reimplemented in `Arc::Plexer`.

5.158.3.3 `virtual MCC_Status Arc::MCC::process(Message &, Message &) [inline, virtual]`

Dummy `Message` processing method. Just a placeholder.

Implements `Arc::MCCInterface`.

Reimplemented in `Arc::Plexer`.

5.158.3.4 `bool Arc::MCC::ProcessSecHandlers(Message & message, const std::string & label = " ") const [protected]`

Executes security handlers of specified queue. Returns true if the message is authorized for further processing or if there are no security handlers which implement authorization functionality. This is a convenience method and has to be called by the implementation of the `MCC`.

5.158.3.5 `virtual void Arc::MCC::Unlink() [virtual]`

Removing all links. Useful for destroying chains.

5.158.4 Field Documentation

5.158.4.1 `Logger Arc::MCC::logger [static, protected]`

A logger for MCCs.

A logger intended to be the parent of loggers in the different MCCs.

Reimplemented in [Arc::Plexer](#).

5.158.4.2 `std::map<std::string, MCCLInterface *> Arc::MCC::next_ [protected]`

Set of labeled "next" components. Each implemented [MCC](#) must call [process\(\)](#) method of corresponding [MCCLInterface](#) from this set in own [process\(\)](#) method.

5.158.4.3 `std::map<std::string, std::list<ArcSec::SecHandler *> > Arc::MCC::sechandlers_ [protected]`

Set of labeled authentication and authorization handlers. [MCC](#) calls sequence of handlers at specific point depending on associated identifier. In most cases those are "in" and "out" for incoming and outgoing messages correspondingly.

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

- `MCC.h`

5.159 `Arc::MCC_Status` Class Reference

A class for communication of [MCC](#) processing results.

```
#include <MCC_Status.h>
```

Public Member Functions

- `MCC_Status (StatusKind kind=STATUS_UNDEFINED, const std::string &origin="???", const std::string &explanation="No explanation.")`
- `boolisOk () const`
- `StatusKind getKind () const`
- `const std::string &getOrigin () const`
- `const std::string &getExplanation () const`
- `operator std::string () const`
- `operator bool (void) const`
- `bool operator! (void) const`

5.159.1 Detailed Description

A class for communication of [MCC](#) processing results.

This class is used to communicate result status between MCCs. It contains a status kind, a string specifying the origin ([MCC](#)) of the status object and an explanation.

5.159.2 Constructor & Destructor Documentation

5.159.2.1 `Arc::MCC_Status::MCC_Status (StatusKind kind = STATUS_UNDEFINED,
const std::string & origin = "???", const std::string & explanation =
"No explanation.")`

The constructor.

Creates a [MCC_Status](#) object.

Parameters

<i>kind</i>	The StatusKind (default: STATUS_UNDEFINED)
<i>origin</i>	The origin MCC (default: "??")
<i>explanation</i>	An explanation (default: "No explanation.")

5.159.3 Member Function Documentation

5.159.3.1 `const std::string& Arc::MCC_Status::getExplanation () const`

Returns an explanation.

This method returns an explanation of this object.

Returns

An explanation of this object.

5.159.3.2 `StatusKind Arc::MCC_Status::getKind () const`

Returns the status kind.

Returns the status kind of this object.

Returns

The status kind of this object.

5.159.3.3 `const std::string& Arc::MCC_Status::getOrigin () const`

Returns the origin.

This method returns a string specifying the origin [MCC](#) of this object.

Returns

A string specifying the origin [MCC](#) of this object.

5.159.3.4 bool Arc::MCC_Status::isOk () const

Is the status kind ok?

This method returns true if the status kind of this object is STATUS_OK

Returns

true if kind==STATUS_OK

Referenced by operator bool(), and operator!().

5.159.3.5 Arc::MCC_Status::operator bool (void) const [inline]

Is the status kind ok?

This method returns true if the status kind of this object is STATUS_OK

Returns

true if kind==STATUS_OK

References isOk().

5.159.3.6 Arc::MCC_Status::operator std::string () const

Conversion to string.

This operator converts a [MCC_Status](#) object to a string.

5.159.3.7 bool Arc::MCC_Status::operator! (void) const [inline]

not operator

Returns true if the status kind is not OK

Returns

true if kind!=STATUS_OK

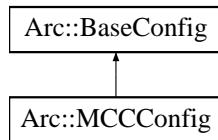
References isOk().

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

- MCC_Status.h

5.160 Arc::MCCConfig Class Reference

Inheritance diagram for Arc::MCCConfig:



Public Member Functions

- virtual `XMLNode MakeConfig (XMLNode cfg) const`

5.160.1 Member Function Documentation

5.160.1.1 virtual XMLNode Arc::MCCConfig::MakeConfig (XMLNode *cfg*) const [virtual]

Adds configuration part corresponding to stored information into common configuration tree supplied in 'cfg' argument. Returns reference to XML node representing configuration of [ModuleManager](#)

Reimplemented from [Arc::BaseConfig](#).

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

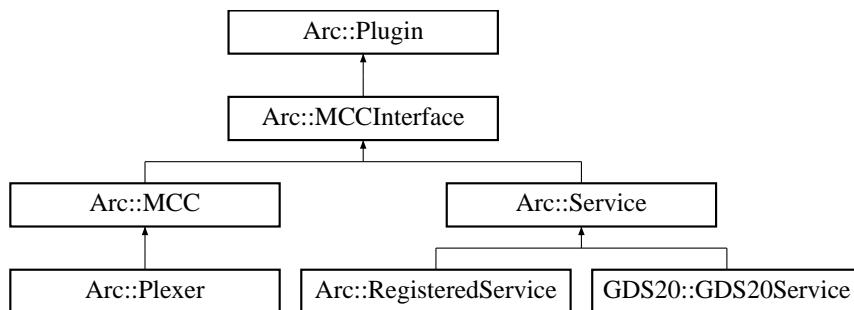
- `MCC.h`

5.161 Arc::MCCTInterface Class Reference

Interface for communication between [MCC](#), [Service](#) and [Plexer](#) objects.

```
#include <MCC.h>
```

Inheritance diagram for Arc::MCCTInterface:



Public Member Functions

- virtual [MCC_Status process](#) ([Message &request](#), [Message &response](#))=0

5.161.1 Detailed Description

Interface for communication between [MCC](#), [Service](#) and [Plexer](#) objects.

The Interface consists of the method [process\(\)](#) which is called by the previous [MCC](#) in the chain. For memory management policies please read the description of the [Message](#) class.

5.161.2 Member Function Documentation

5.161.2.1 virtual [MCC_Status Arc::MCCLoader::process](#) ([Message & request](#), [Message & response](#)) [pure virtual]

Method for processing of requests and responses. This method is called by preceeding [MCC](#) in chain when a request needs to be processed. This method must call similar method of next [MCC](#) in chain unless any failure happens. Result returned by call to next [MCC](#) should be processed and passed back to previous [MCC](#). In case of failure this method is expected to generate valid error response and return it back to previous [MCC](#) without calling the next one.

Parameters

<i>request</i>	The request that needs to be processed.
<i>response</i>	A Message object that will contain the response of the request when the method returns.

Returns

An object representing the status of the call.

Implemented in [GDS20::GDS20Service](#), [Arc::MCC](#), and [Arc::Plexer](#).

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

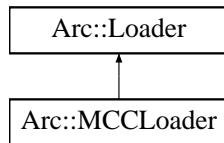
- MCC.h

5.162 Arc::MCCLoader Class Reference

Creator of [Message](#) Component Chains ([MCC](#)).

```
#include <MCCLoader.h>
```

Inheritance diagram for Arc::MCCLoader:



Public Member Functions

- [MCCLoader \(Config &cfg\)](#)
- [~MCCLoader \(\)](#)
- [MCC * operator\[\] \(const std::string &id\)](#)

5.162.1 Detailed Description

Creator of [Message](#) Component Chains ([MCC](#)).

This class processes XML configuration and creates message chains. Accepted configuration is defined by XML schema mcc.xsd. Supported components are of types [MCC](#), [Service](#) and [Plexer](#). [MCC](#) and [Service](#) are loaded from dynamic libraries. For [Plexer](#) only internal implementation is supported. This object is also a container for loaded components. All components and chains are destroyed if this object is destroyed. Chains are created in 2 steps. First all components are loaded and corresponding objects are created. Constructors are supplied with corresponding configuration subtrees. During next step components are linked together by calling their Next() methods. Each call creates labeled link to next component in a chain. 2 step method has an advantage over single step because it allows loops in chains and makes loading procedure more simple. But that also means during short period of time components are only partly configured. Components in such state must produce proper error response if [Message](#) arrives. Note: Current implementation requires all components and links to be labeled. All labels must be unique. Future implementation will be able to assign labels automatically.

5.162.2 Constructor & Destructor Documentation

5.162.2.1 Arc::MCCLoader::MCCLoader (Config & cfg)

Constructor that takes whole XML configuration and creates component chains

5.162.2.2 Arc::MCCLoader::~MCCLoader ()

Destructor destroys all components created by constructor

5.162.3 Member Function Documentation

5.162.3.1 **MCC* Arc::MCCLoader::operator[](const std::string & id)**

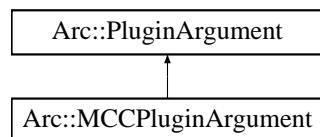
Access entry MCCs in chains. Those are components exposed for external access using 'entry' attribute

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

- MCCLoader.h

5.163 Arc::MCCPluginArgument Class Reference

Inheritance diagram for Arc::MCCPluginArgument:



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

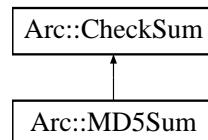
- MCC.h

5.164 Arc::MD5Sum Class Reference

Implementation of MD5 checksum.

```
#include <CheckSum.h>
```

Inheritance diagram for Arc::MD5Sum:



Public Member Functions

- virtual void **start** (void)
- virtual void **add** (void *buf, unsigned long long int len)
- virtual void **end** (void)
- virtual void **result** (unsigned char *&res, unsigned int &len) const
- virtual int **print** (char *buf, int len) const

- virtual void `scan` (const char *buf)
- virtual operator bool (void) const
- virtual bool operator! (void) const

5.164.1 Detailed Description

Implementation of MD5 checksum.

This class is a specialized class of the [CheckSum](#) class. It provides an implementation of the MD5 message-digest algorithm specified in RFC 1321.

5.164.2 Member Function Documentation

5.164.2.1 virtual void Arc::MD5Sum::add (void * buf, unsigned long long int len) [virtual]

Add data to be checksummed.

This method calculates the checksum of the passed data chuck, taking into account the previous state of this object.

Parameters

<code>buf</code>	pointer to data chuck to be checksummed.
<code>len</code>	size of the data chuck.

Implements [Arc::CheckSum](#).

5.164.2.2 virtual void Arc::MD5Sum::end (void) [virtual]

Finalize the checksumming.

This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::CheckSum](#).

5.164.2.3 virtual int Arc::MD5Sum::print (char * buf, int len) const [virtual]

Retrieve result of checksum into a string.

The passed string buf is filled with result of checksum algorithm in base 16. At most len characters is filled into buffer buf. The hexadecimal value is prepended with "<algorithm>:", where <algorithm> is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters

<code>buf</code>	pointer to buffer which should be filled with checksum result.
<code>len</code>	max number of character filled into buffer.

Reimplemented from [Arc::CheckSum](#).

5.164.2.4 virtual void Arc::MD5Sum::scan (const char * buf) [virtual]

Set internal checksum state.

This method sets the internal state to that of the passed textural representation. The format passed to this method must be the same as retrieved from the [CheckSum::print](#) method.

Parameters

<i>buf</i>	string containing textural representation of checksum
------------	---

See also

[CheckSum::print](#)

Implements [Arc::CheckSum](#).

5.164.2.5 virtual void Arc::MD5Sum::start (void) [virtual]

Initiate the checksum algorithm.

This method must be called before starting a new checksum calculation.

Implements [Arc::CheckSum](#).

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

- [CheckSum.h](#)

5.165 Arc::MemoryAllocationException Class Reference

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

- [ByteArray.h](#)

5.166 Arc::Message Class Reference

Object being passed through chain of MCCs.

```
#include <Message.h>
```

Public Member Functions

- [Message \(void\)](#)

- [Message \(Message &msg\)](#)
- [Message \(long msg_ptr_addr\)](#)
- [~Message \(void\)](#)
- [Message & operator= \(Message &msg\)](#)
- [MessagePayload * Payload \(void\)](#)
- [MessagePayload * Payload \(MessagePayload *payload\)](#)
- [MessageAttributes * Attributes \(void\)](#)
- [MessageAuth * Auth \(void\)](#)
- [MessageContext * Context \(void\)](#)
- [MessageAuthContext * AuthContext \(void\)](#)
- [void Context \(MessageContext *ctx\)](#)
- [void AuthContext \(MessageAuthContext *auth_ctx\)](#)

5.166.1 Detailed Description

Object being passed through chain of MCCs.

An instance of this class refers to objects with main content ([MessagePayload](#)), authentication/authorization information ([MessageAuth](#)) and common purpose attributes ([MessageAttributes](#)). [Message](#) class does not manage pointers to objects and their content. It only serves for grouping those objects. [Message](#) objects are supposed to be processed by MCCs and Services implementing [MCCIInterface](#) method `process()`. All objects constituting content of [Message](#) object are subject to following policies:

1. All objects created inside call to `process()` method using new command must be explicitly destroyed within same call using delete command with following exceptions.
a) Objects which are assigned to 'response' [Message](#). b) Objects whose management is completely acquired by objects assigned to 'response' [Message](#).
2. All objects not created inside call to `process()` method are not explicitly destroyed within that call with following exception. a) Objects which are part of 'response' Method returned from call to next's `process()` method. Unless those objects are passed further to calling `process()`, of course.
3. It is not allowed to make 'response' point to same objects as 'request' does on entry to `process()` method. That is needed to avoid double destruction of same object. (Note: if in a future such need arises it may be solved by storing additional flags in [Message](#) object).
4. It is allowed to change content of pointers of 'request' [Message](#). Calling `process()` method must not rely on that object to stay intact.
5. Called `process()` method should either fill 'response' [Message](#) with pointers to valid objects or to keep them intact. This makes it possible for calling `process()` to preload 'response' with valid error message.

5.166.2 Constructor & Destructor Documentation

5.166.2.1 Arc::Message::Message (void) [inline]

true if auth_ctx_ was created internally Dummy constructor

5.166.2.2 Arc::Message::Message (*Message & msg*) [inline]

Copy constructor. Ensures shallow copy.

5.166.2.3 Arc::Message::Message (*long msg_ptr.addr*)

Copy constructor. Used by language bindigs

5.166.2.4 Arc::Message::~Message (*void*) [inline]

Destructor does not affect referred objects except those created internally

5.166.3 Member Function Documentation

5.166.3.1 MessageAttributes* Arc::Message::Attributes (*void*) [inline]

Returns a pointer to the current attributes object or creates it if no attributes object has been assigned.

Referenced by operator=().

5.166.3.2 MessageAuth* Arc::Message::Auth (*void*) [inline]

Returns a pointer to the current authentication/authorization object or creates it if no object has been assigned.

Referenced by operator=().

5.166.3.3 MessageAuthContext* Arc::Message::AuthContext (*void*) [inline]

Returns a pointer to the current auth* context object or creates it if no object has been assigned.

Referenced by operator=().

5.166.3.4 void Arc::Message::AuthContext (*MessageAuthContext * auth_ctx*) [inline]

Assigns auth* context object

5.166.3.5 void Arc::Message::Context (*MessageContext * ctx*) [inline]

Assigns message context object

5.166.3.6 MessageContext* Arc::Message::Context (void) [inline]

Returns a pointer to the current context object or creates it if no object has been assigned. Last case should happen only if first [MCC](#) in a chain is connectionless like one implementing UDP protocol.

Referenced by [operator=\(\)](#).

5.166.3.7 Message& Arc::Message::operator= (Message & msg) [inline]

Assignment. Ensures shallow copy.

References [Attributes\(\)](#), [Auth\(\)](#), [AuthContext\(\)](#), and [Context\(\)](#).

5.166.3.8 MessagePayload* Arc::Message::Payload (void) [inline]

Returns pointer to current payload or NULL if no payload assigned.

5.166.3.9 MessagePayload* Arc::Message::Payload (MessagePayload * payload) [inline]

Replaces payload with new one. Returns the old one.

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

- [Message.h](#)

5.167 Arc::MessageAttributes Class Reference

A class for storage of attribute values.

```
#include <MessageAttributes.h>
```

Public Member Functions

- [MessageAttributes \(\)](#)
- [void set \(const std::string &key, const std::string &value\)](#)
- [void add \(const std::string &key, const std::string &value\)](#)
- [void removeAll \(const std::string &key\)](#)
- [void remove \(const std::string &key, const std::string &value\)](#)
- [int count \(const std::string &key\) const](#)
- [const std::string & get \(const std::string &key\) const](#)
- [AttributIterator getAll \(const std::string &key\) const](#)
- [AttributIterator getAll \(void\) const](#)

Protected Attributes

- AttrMap attributes_

5.167.1 Detailed Description

A class for storage of attribute values.

This class is used to store attributes of messages. All attribute keys and their corresponding values are stored as strings. Any key or value that is not a string must thus be represented as a string during storage. Furthermore, an attribute is usually a key-value pair with a unique key, but there may also be multiple such pairs with equal keys.

The key of an attribute is composed by the name of the [Message Chain Component \(MCC\)](#) which produce it and the name of the attribute itself with a colon (:) in between, i.e. MCC_Name:Attribute_Name. For example, the key of the "Content-Length" attribute of the HTTP [MCC](#) is thus "HTTP:Content-Length".

There are also "global attributes", which may be produced by different MCCs depending on the configuration. The keys of such attributes are NOT prefixed by the name of the producing [MCC](#). Before any new global attribute is introduced, it must be agreed upon by the core development team and added below. The global attributes decided so far are:

- Request-URI Identifies the service to which the message shall be sent. This attribute is produced by e.g. the HTTP [MCC](#) and used by the plexer for routing the message to the appropriate service.

5.167.2 Constructor & Destructor Documentation

5.167.2.1 Arc::MessageAttributes::MessageAttributes()

The default constructor.

This is the default constructor of the [MessageAttributes](#) class. It constructs an empty object that initially contains no attributes.

5.167.3 Member Function Documentation

5.167.3.1 void Arc::MessageAttributes::add(const std::string & key, const std::string & value)

Adds a value to an attribute.

This method adds a new value to an attribute. Any previous value will be preserved, i.e. the attribute may become multiple valued.

Parameters

<i>key</i>	The key of the attribute.
<i>value</i>	The (new) value of the attribute.

5.167.3.2 int Arc::MessageAttributes::count (const std::string & key) const

Returns the number of values of an attribute.

Returns the number of values of an attribute that matches a certain key.

Parameters

<i>key</i>	The key of the attribute for which to count values.
------------	---

Returns

The number of values that corresponds to the key.

5.167.3.3 const std::string& Arc::MessageAttributes::get (const std::string & key) const

Returns the value of a single-valued attribute.

This method returns the value of a single-valued attribute. If the attribute is not single valued (i.e. there is no such attribute or it is a multiple-valued attribute) an empty string is returned.

Parameters

<i>key</i>	The key of the attribute for which to return the value.
------------	---

Returns

The value of the attribute.

5.167.3.4 AttributeIterator Arc::MessageAttributes::getAll (const std::string & key) const

Access the value(s) of an attribute.

This method returns an [AttributeIterator](#) that can be used to access the values of an attribute.

Parameters

<i>key</i>	The key of the attribute for which to return the values.
------------	--

Returns

An [AttributeIterator](#) for access of the values of the attribute.

5.167.3.5 void Arc::MessageAttributes::remove (const std::string & key, const std::string & value)

Removes one value of an attribute.

This method removes a certain value from the attribute that matches a certain key.

Parameters

<i>key</i>	The key of the attribute from which the value shall be removed.
<i>value</i>	The value to remove.

5.167.3.6 void Arc::MessageAttributes::removeAll (const std::string & *key*)

Removes all attributes with a certain key.

This method removes all attributes that match a certain key.

Parameters

<i>key</i>	The key of the attributes to remove.
------------	--------------------------------------

5.167.3.7 void Arc::MessageAttributes::set (const std::string & *key*, const std::string & *value*)

Sets a unique value of an attribute.

This method removes any previous value of an attribute and sets the new value as the only value.

Parameters

<i>key</i>	The key of the attribute.
<i>value</i>	The (new) value of the attribute.

5.167.4 Field Documentation**5.167.4.1 AttrMap Arc::MessageAttributes::attributes_ [protected]**

Internal storage of attributes.

An AttrMap (multimap) in which all attributes (key-value pairs) are stored.

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

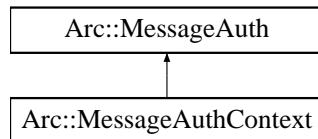
- MessageAttributes.h

5.168 Arc::MessageAuth Class Reference

Contains authencity information, authorization tokens and decisions.

```
#include <MessageAuth.h>
```

Inheritance diagram for Arc::MessageAuth:



Public Member Functions

- void [set](#) (const std::string &key, SecAttr *value)
- void [remove](#) (const std::string &key)
- SecAttr * [get](#) (const std::string &key)
- SecAttr * [operator\[\]](#) (const std::string &key)
- bool [Export](#) (SecAttrFormat format, XMLNode &val) const
- [MessageAuth](#) * [Filter](#) (const std::list< std::string > &selected_keys, const std::list< std::string > &rejected_keys)

5.168.1 Detailed Description

Contains authencity information, authorization tokens and decisions.

This class only supports string keys and [SecAttr](#) values.

5.168.2 Member Function Documentation

5.168.2.1 bool Arc::MessageAuth::Export (SecAttrFormat format, XMLNode & val) const

Returns properly catenated attributes in specified format.

Content of XML node at is replaced with generated information if XML tree is empty. If tree at is not empty then [Export\(\)](#) tries to merge generated information to already existing like everything would be generated inside same [Export\(\)](#) method. If does not represent valid node then new XML tree is created.

5.168.2.2 MessageAuth* Arc::MessageAuth::Filter (const std::list< std::string > & selected_keys, const std::list< std::string > & rejected_keys)

Creates new instance of [MessageAuth](#) with attributes filtered.

In new instance all attributes with keys listed in are removed. If is not empty only corresponding attributes are transferred to new instance. Created instance does not own referred attributes. Hence parent instance must not be deleted as long as this one is in use.

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

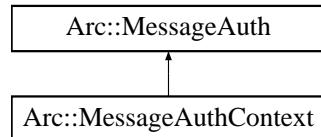
- [MessageAuth.h](#)

5.169 Arc::MessageAuthContext Class Reference

Handler for content of message auth* context.

```
#include <Message.h>
```

Inheritance diagram for Arc::MessageAuthContext:



5.169.1 Detailed Description

Handler for content of message auth* context.

This class is a container for authorization and authentication information. It gets associated with [Message](#) object usually by first [MCC](#) in a chain and is kept as long as connection persists.

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

- [Message.h](#)

5.170 Arc::MessageContext Class Reference

Handler for content of message context.

```
#include <Message.h>
```

Public Member Functions

- void [Add](#) (const std::string &name, [MessageContextElement](#) *element)

5.170.1 Detailed Description

Handler for content of message context.

This class is a container for objects derived from [MessageContextElement](#). It gets associated with [Message](#) object usually by first [MCC](#) in a chain and is kept as long as connection persists.

5.170.2 Member Function Documentation

```
5.170.2.1 void Arc::MessageContext::Add ( const std::string & name,
                                         MessageContextElement * element )
```

Provided element is taken over by this class. It is remembered by it and destroyed when this class is destroyed.

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

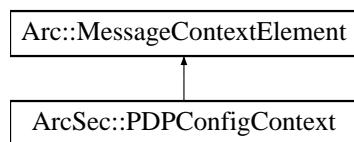
- Message.h

5.171 Arc::MessageContextElement Class Reference

Top class for elements contained in message context.

```
#include <Message.h>
```

Inheritance diagram for Arc::MessageContextElement:



5.171.1 Detailed Description

Top class for elements contained in message context.

Objects of classes inherited with this one may be stored in [MessageContext](#) container.

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

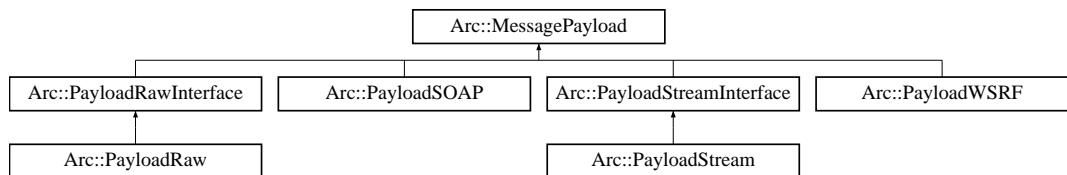
- Message.h

5.172 Arc::MessagePayload Class Reference

Base class for content of message passed through chain.

```
#include <Message.h>
```

Inheritance diagram for Arc::MessagePayload:



5.172.1 Detailed Description

Base class for content of message passed through chain.

It's not intended to be used directly. Instead functional classes must be derived from it.

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

- Message.h

5.173 Arc::ModuleDesc Class Reference

Description of loadable module.

```
#include <Plugin.h>
```

5.173.1 Detailed Description

Description of loadable module.

This class is used for reports

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

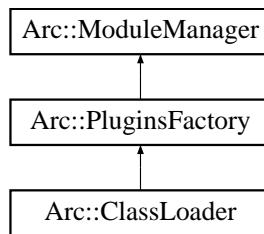
- Plugin.h

5.174 Arc::ModuleManager Class Reference

Manager of shared libraries.

```
#include <ModuleManager.h>
```

Inheritance diagram for Arc::ModuleManager:



Public Member Functions

- [ModuleManager \(XMLNode cfg\)](#)
- Glib::Module * [load](#) (const std::string &name, bool probe=false)
- std::string [find](#) (const std::string &name)

- `Glib::Module * reload (Glib::Module *module)`
- `void unload (Glib::Module *module)`
- `void unload (const std::string &name)`
- `std::string findLocation (const std::string &name)`
- `bool makePersistent (Glib::Module *module)`
- `bool makePersistent (const std::string &name)`
- `void setCfg (XMLNode cfg)`

5.174.1 Detailed Description

Manager of shared libraries.

This class loads shared libraries/modules. There supposed to be created one instance of it per executable. In such circumstances it would cache handles to loaded modules and not load them multiple times.

5.174.2 Constructor & Destructor Documentation

5.174.2.1 `Arc::ModuleManager::ModuleManager (XMLNode cfg)`

Cache of handles of loaded modules Constructor. It is supposed to process correponding configuration subtree and tune module loading parameters accordingly.

5.174.3 Member Function Documentation

5.174.3.1 `std::string Arc::ModuleManager::find (const std::string & name)`

Finds loadable module by 'name' looking in same places as `load()` does, but does not load it.

5.174.3.2 `std::string Arc::ModuleManager::findLocation (const std::string & name)`

Finds shared library corresponding to module 'name' and returns path to it

5.174.3.3 `Glib::Module* Arc::ModuleManager::load (const std::string & name, bool probe = false)`

Finds module 'name' in cache or loads corresponding loadable module

5.174.3.4 `bool Arc::ModuleManager::makePersistent (const std::string & name)`

Make sure this module is never unloaded. Even if `unload()` is called.

5.174.3.5 bool Arc::ModuleManager::makePersistent (Glib::Module * *module*)

Make sure this module is never unloaded. Even if [unload\(\)](#) is called.

5.174.3.6 Glib::Module* Arc::ModuleManager::reload (Glib::Module * *module*)

Reload module previously loaded in probe mode. New module is loaded with all symbols resolved and old module handler is unloaded. In case of error old module is not unloaded.

5.174.3.7 void Arc::ModuleManager::setCfg (XMLNode *cfg*)

Input the configuration subtree, and trigger the module loading (do almost the same as the Constructor); It is function designed for [ClassLoader](#) to adopt the singleton pattern

5.174.3.8 void Arc::ModuleManager::unload (const std::string & *name*)

Unload module by its name

5.174.3.9 void Arc::ModuleManager::unload (Glib::Module * *module*)

Unload module by its identifier

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

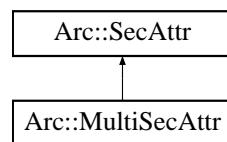
- [ModuleManager.h](#)

5.175 Arc::MultiSecAttr Class Reference

Container of multiple [SecAttr](#) attributes.

```
#include <SecAttr.h>
```

Inheritance diagram for Arc::MultiSecAttr:



Public Member Functions

- virtual [operator bool \(\) const](#)
- virtual bool [Export \(SecAttrFormat format, XMLNode &val\) const](#)

5.175.1 Detailed Description

Container of multiple [SecAttr](#) attributes.

This class combines multiple attributes. It's export/import methods catenate results of underlying objects. Primary meaning of this class is to serve as base for classes implementing multi level hierarchical tree-like descriptions of user identity. It may also be used for collecting information of same source or kind. Like all information extracted from X509 certificate.

5.175.2 Member Function Documentation

**5.175.2.1 virtual bool Arc::MultiSecAttr::Export (*SecAttrFormat format*, *XMLNode & val*)
const [virtual]**

Convert internal structure into specified format. Returns false if format is not supported/suitable for this attribute. XML node referenced by is turned into top level element of specified format.

Reimplemented from [Arc::SecAttr](#).

5.175.2.2 virtual Arc::MultiSecAttr::operator bool () const [virtual]

This function should return false if the value is to be considered null, e.g. if it hasn't been set or initialized. In other cases it should return true.

Reimplemented from [Arc::SecAttr](#).

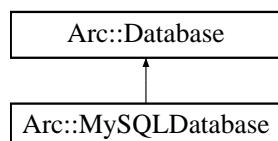
The documentation for this class was generated from the following file:

- [SecAttr.h](#)

5.176 Arc::MySQLDatabase Class Reference

```
#include <MysqlWrapper.h>
```

Inheritance diagram for Arc::MySQLDatabase:



Public Member Functions

- `virtual bool connect (std::string &dbname, std::string &user, std::string &password)`

- virtual bool `isconnected` () const
- virtual void `close` ()
- virtual bool `enable_ssl` (const std::string &`keyfile`= "", const std::string &`certfile`= "", const std::string &`cafile`= "", const std::string &`capath`= "")
- virtual bool `shutdown` ()

5.176.1 Detailed Description

Implement the database accessing interface in `DBInterface.h` by using mysql client library for accessing mysql database

5.176.2 Member Function Documentation

5.176.2.1 virtual void Arc::MySQLDatabase::close () [virtual]

Close the connection with database server

Implements [Arc::Database](#).

5.176.2.2 virtual bool Arc::MySQLDatabase::connect (std::string & *dbname*, std::string & *user*, std::string & *password*) [virtual]

Do connection with database server

Parameters

<code>dbname</code>	The database name which will be used.
<code>user</code>	The username which will be used to access database.
<code>password</code>	The password which will be used to access database.

Implements [Arc::Database](#).

5.176.2.3 virtual bool Arc::MySQLDatabase::enable_ssl (const std::string & *keyfile* = " ", const std::string & *certfile* = " ", const std::string & *cafile* = " ", const std::string & *capath* = " ") [virtual]

Enable ssl communication for the connection

Parameters

<code>keyfile</code>	The location of key file.
<code>certfile</code>	The location of certificate file.
<code>cafile</code>	The location of ca file.
<code>capath</code>	The location of ca directory

Implements [Arc::Database](#).

5.176.2.4 virtual bool Arc::MySQLDatabase::isconnected() const [inline, virtual]

Get the connection status

Implements [Arc::Database](#).

5.176.2.5 virtual bool Arc::MySQLDatabase::shutdown() [virtual]

Ask database server to shutdown

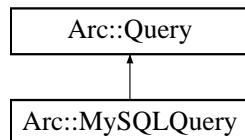
Implements [Arc::Database](#).

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

- MysqlWrapper.h

5.177 Arc::MySQLQuery Class Reference

Inheritance diagram for Arc::MySQLQuery:



Public Member Functions

- virtual int [get_num_columns\(\)](#)
- virtual int [get_num_rows\(\)](#)
- virtual bool [execute\(const std::string &sqlstr\)](#)
- virtual QueryRowResult [get_row\(int row_number\) const](#)
- virtual QueryRowResult [get_row\(\) const](#)
- virtual std::string [get_row_field\(int row_number, std::string &field_name\)](#)
- virtual bool [get_array\(std::string &sqlstr, QueryArrayResult &result, std::vector<std::string > &arguments\)](#)

5.177.1 Member Function Documentation

5.177.1.1 virtual bool Arc::MySQLQuery::execute(const std::string & sqlstr) [virtual]

Execute the query

Parameters

<i>sqlstr</i>	The sql sentence used to query
---------------	--------------------------------

Implements [Arc::Query](#).

5.177.1.2 virtual bool Arc::MySQLQuery::get_array (std::string & *sqlstr*, QueryArrayResult & *result*, std::vector< std::string > & *arguments*) [virtual]

[Query](#) the database by using some parameters into sql sentence e.g. "select table.value from table where table.name = ?"

Parameters

<i>sqlstr</i>	The sql sentence with some parameters marked with "?".
<i>result</i>	The result in an array which includes all of the value in query result.
<i>arguments</i>	The argument list which should exactly correspond with the parameters in sql sentence.

Implements [Arc::Query](#).

5.177.1.3 virtual int Arc::MySQLQuery::get_num_columns () [virtual]

Get the column number in the query result

Implements [Arc::Query](#).

5.177.1.4 virtual int Arc::MySQLQuery::get_num_rows () [virtual]

Get the row number in the query result

Implements [Arc::Query](#).

5.177.1.5 virtual QueryRowResult Arc::MySQLQuery::get_row (int *row_number*) const [virtual]

Get the value of one row in the query result

Parameters

<i>row_number</i>	The number of the row
-------------------	-----------------------

Returns

A vector includes all the values in the row

Implements [Arc::Query](#).

5.177.1.6 virtual QueryRowResult Arc::MySQLQuery::get_row () const [virtual]

Get the value of one row in the query result, the row number will be automatically increased each time the method is called

Implements [Arc::Query](#).

5.177.1.7 virtual std::string Arc::MySQLQuery::get_row_field (int *row_number*, std::string & *field_name*) [virtual]

Get the value of one specific field in one specific row

Parameters

<i>row_number</i>	The row number inside the query result
<i>field_name</i>	The field name for the value which will be return

Returns

The value of the specified filed in the specified row

Implements [Arc::Query](#).

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

- MysqlWrapper.h

5.178 Arc::NotificationType Class Reference

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

- JobDescription.h

5.179 Arc::NS Class Reference

Public Member Functions

- [NS](#) (void)
- [NS](#) (const char *prefix, const char *uri)
- [NS](#) (const char *nslist[][2])

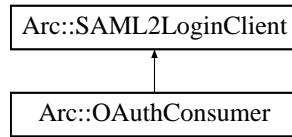
The documentation for this class was generated from the following file:

- XMLNode.h

5.180 Arc::OAuthConsumer Class Reference

```
#include <OAuthConsumer.h>
```

Inheritance diagram for Arc::OAuthConsumer:



Public Member Functions

- [OAuthConsumer \(const MCCConfig cfg, const URL url, std::list< std::string > idp_stack\)](#)
- [MCC_Status parseDN \(std::string *dn\)](#)
- [MCC_Status approveCSR \(const std::string approve_page\)](#)
- [MCC_Status pushCSR \(const std::string b64_pub_key, const std::string pub_key_hash, std::string *approve_page\)](#)
- [MCC_Status storeCert \(const std::string cert_path, const std::string auth_token, const std::string b64_dn\)](#)

Protected Member Functions

- [MCC_Status processLogin \(const std::string username="", const std::string password ""\)](#)

5.180.1 Detailed Description

The OAuth functionality depends on the availability of the liboauth C-bindings library

5.180.2 Constructor & Destructor Documentation

5.180.2.1 [Arc::OAuthConsumer::OAuthConsumer \(const MCCConfig cfg, const URL url, std::list< std::string > idp_stack \)](#)

Construct an OAuth consumer with url as service provider. idp_name is currently ignored, since the idp to which the SAML2 redirect will take place is presently a hardcoded value on the SAML2 SP side. This is expected to change in the future.

5.180.3 Member Function Documentation

5.180.3.1 [MCC_Status Arc::OAuthConsumer::approveCSR \(const std::string approve_page \) \[virtual\]](#)

Unsupported placeholder function until Confusa supports OAuth.

Implements [Arc::SAML2LoginClient](#).

5.180.3.2 MCC_Status Arc::OAuthConsumer::parseDN (std::string * dn) [virtual]

Unsupported placeholder function until Confusa supports OAuth.

Implements [Arc::SAML2LoginClient](#).

5.180.3.3 MCC_Status Arc::OAuthConsumer::processLogin (const std::string username = "", const std::string password = "") [protected, virtual]

Main function performing all the OAuth login steps. Username and password will be ignored.

Implements [Arc::SAML2LoginClient](#).

5.180.3.4 MCC_Status Arc::OAuthConsumer::pushCSR (const std::string b64_pub_key, const std::string pub_key_hash, std::string * approve_page) [virtual]

Unsupported placeholder function until Confusa supports OAuth.

Implements [Arc::SAML2LoginClient](#).

5.180.3.5 MCC_Status Arc::OAuthConsumer::storeCert (const std::string cert_path, const std::string auth_token, const std::string b64_dn) [virtual]

Unsupported placeholder function until Confusa supports OAuth.

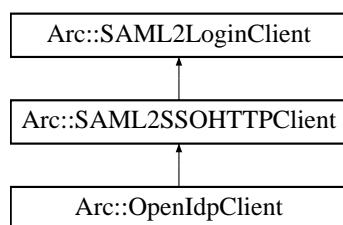
Implements [Arc::SAML2LoginClient](#).

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

- OAuthConsumer.h

5.181 Arc::OpenIdpClient Class Reference

Inheritance diagram for Arc::OpenIdpClient:



Protected Member Functions

- [MCC_Status processIdPLogin \(const std::string username, const std::string password\)](#)
- [MCC_Status processConsent \(\)](#)
- [MCC_Status processIdP2Confusa \(\)](#)

5.181.1 Member Function Documentation

5.181.1.1 **MCC_Status Arc::OpenIdpClient::processConsent ()** [protected, virtual]

If the IdP has a consent module and the user has not saved her consent, this method will ask the user for consent to transmission of her data to Confusa

Implements [Arc::SAML2SSOHTTPClient](#).

5.181.1.2 **MCC_Status Arc::OpenIdpClient::processIdP2Confusa ()** [protected, virtual]

Redirects the user back from identity provider to the Confusa SP

Implements [Arc::SAML2SSOHTTPClient](#).

5.181.1.3 **MCC_Status Arc::OpenIdpClient::processIdPLogin (const std::string *username*, const std::string *password*)** [protected, virtual]

Actual identity provider parsers for next three methods implemented in subdirectory idp/

Parse identity provider login page and submit username and password in the pre-
sioned way

Implements [Arc::SAML2SSOHTTPClient](#).

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

- [OpenIdpClient.h](#)

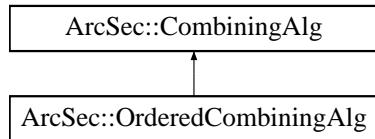
5.182 Arc::OptionParser Class Reference

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

- [OptionParser.h](#)

5.183 ArcSec::OrderedCombiningAlg Class Reference

Inheritance diagram for ArcSec::OrderedCombiningAlg:



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

- OrderedAlg.h

5.184 passwd Struct Reference

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

- win32.h

5.185 Arc::PathIterator Class Reference

Class to iterate through elements of path.

```
#include <URL.h>
```

Public Member Functions

- [PathIterator](#) (const std::string &path, bool end=false)
- [PathIterator & operator++ \(\)](#)
- [PathIterator & operator-- \(\)](#)
- [operator bool \(\) const](#)
- [std::string operator* \(\) const](#)
- [std::string Rest \(\) const](#)

5.185.1 Detailed Description

Class to iterate through elements of path.

5.185.2 Constructor & Destructor Documentation

5.185.2.1 [Arc::PathIterator::PathIterator \(const std::string & path, bool end = false \)](#)

Constructor accepts path and stores it internally. If end is set to false iterator is pointing at first element in path. Otherwise selected element is one before last.

5.185.3 Member Function Documentation

5.185.3.1 Arc::PathIterator::operator bool () const

Return false when iterator moved outside path elements

5.185.3.2 std::string Arc::PathIterator::operator* () const

Returns part of initial path from first till and including current

5.185.3.3 PathIterator& Arc::PathIterator::operator++ ()

Advances iterator to point at next path element

5.185.3.4 PathIterator& Arc::PathIterator::operator-- ()

Moves iterator to element before current

5.185.3.5 std::string Arc::PathIterator::Rest () const

Returns part of initial path from one after current till end

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

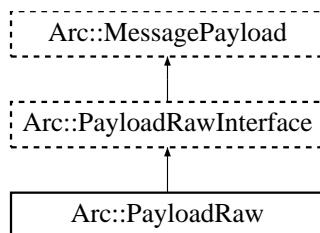
- URL.h

5.186 Arc::PayloadRaw Class Reference

Raw byte multi-buffer.

```
#include <PayloadRaw.h>
```

Inheritance diagram for Arc::PayloadRaw:



Public Member Functions

- [PayloadRaw \(void\)](#)

- virtual `~PayloadRaw` (void)
- virtual char `operator[]` (Size_t pos) const
- virtual char * `Content` (Size_t pos=-1)
- virtual Size_t `Size` (void) const
- virtual char * `Insert` (Size_t pos=0, Size_t size=0)
- virtual char * `Insert` (const char *s, Size_t pos=0, Size_t size=-1)
- virtual char * `Buffer` (unsigned int num=0)
- virtual Size_t `BufferSize` (unsigned int num=0) const
- virtual Size_t `BufferPos` (unsigned int num=0) const
- virtual bool `Truncate` (Size_t size)

5.186.1 Detailed Description

Raw byte multi-buffer.

This is implementation of [PayloadRawInterface](#). Buffers are memory blocks logically placed one after another.

5.186.2 Constructor & Destructor Documentation

5.186.2.1 Arc::PayloadRaw::PayloadRaw (void) [inline]

List of handled buffers. Constructor. Created object contains no buffers.

5.186.2.2 virtual Arc::PayloadRaw::~PayloadRaw (void) [virtual]

Destructor. Frees allocated buffers.

5.186.3 Member Function Documentation

5.186.3.1 virtual char* Arc::PayloadRaw::Buffer (unsigned int num = 0) [virtual]

Returns pointer to num'th buffer

Implements [Arc::PayloadRawInterface](#).

5.186.3.2 virtual Size_t Arc::PayloadRaw::BufferPos (unsigned int num = 0) const [virtual]

Returns position of num'th buffer

Implements [Arc::PayloadRawInterface](#).

5.186.3.3 `virtual Size_t Arc::PayloadRaw::BufferSize (unsigned int num = 0) const [virtual]`

Returns length of num'th buffer

Implements [Arc::PayloadRawInterface](#).

5.186.3.4 `virtual char* Arc::PayloadRaw::Content (Size_t pos = -1) [virtual]`

Get pointer to buffer content at global position 'pos'. By default to beginning of main buffer whatever that means.

Implements [Arc::PayloadRawInterface](#).

5.186.3.5 `virtual char* Arc::PayloadRaw::Insert (Size_t pos = 0, Size_t size = 0) [virtual]`

Create new buffer at global position 'pos' of size 'size'.

Implements [Arc::PayloadRawInterface](#).

5.186.3.6 `virtual char* Arc::PayloadRaw::Insert (const char * s, Size_t pos = 0, Size_t size = -1) [virtual]`

Create new buffer at global position 'pos' of size 'size'. Created buffer is filled with content of memory at 's'. If 'size' is negative content at 's' is expected to be null-terminated.

Implements [Arc::PayloadRawInterface](#).

5.186.3.7 `virtual char Arc::PayloadRaw::operator[] (Size_t pos) const [virtual]`

Returns content of byte at specified position. Specified position 'pos' is treated as global one and goes through all buffers placed one after another.

Implements [Arc::PayloadRawInterface](#).

5.186.3.8 `virtual Size_t Arc::PayloadRaw::Size (void) const [virtual]`

Returns logical size of whole structure.

Implements [Arc::PayloadRawInterface](#).

5.186.3.9 `virtual bool Arc::PayloadRaw::Truncate (Size_t size) [virtual]`

Change size of stored information. If size exceeds end of allocated buffer, buffers are not re-allocated, only logical size is extended. Buffers with location behind new size are deallocated.

Implements [Arc::PayloadRawInterface](#).

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

- [PayloadRaw.h](#)

5.187 Arc::PayloadRawBuf Struct Reference

Data Fields

- int [size](#)
- int [length](#)
- bool [allocated](#)

5.187.1 Field Documentation

5.187.1.1 bool Arc::PayloadRawBuf::allocated

size of used memory - size of buffer

5.187.1.2 int Arc::PayloadRawBuf::length

size of allocated memory

5.187.1.3 int Arc::PayloadRawBuf::size

pointer to buffer in memory

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

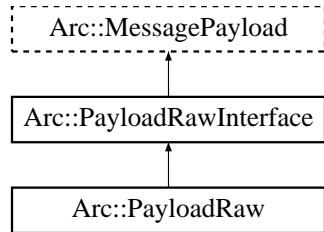
- [PayloadRaw.h](#)

5.188 Arc::PayloadRawInterface Class Reference

Random Access Payload for [Message](#) objects.

```
#include <PayloadRaw.h>
```

Inheritance diagram for Arc::PayloadRawInterface:



Public Member Functions

- virtual char [operator\[\]](#) (Size_t pos) const =0
- virtual char * [Content](#) (Size_t pos=-1)=0
- virtual Size_t [Size](#) (void) const =0
- virtual char * [Insert](#) (Size_t pos=0, Size_t size=0)=0
- virtual char * [Insert](#) (const char *s, Size_t pos=0, Size_t size=-1)=0
- virtual char * [Buffer](#) (unsigned int num)=0
- virtual Size_t [BufferSize](#) (unsigned int num) const =0
- virtual Size_t [BufferPos](#) (unsigned int num) const =0
- virtual bool [Truncate](#) (Size_t size)=0

5.188.1 Detailed Description

Random Access Payload for [Message](#) objects.

This class is a virtual interface for managing [Message](#) payload with arbitrarily accessible content. Inheriting classes are supposed to implement memory-resident or memory-mapped content made of optionally multiple chunks/buffers. Every buffer has own size and offset. This class is purely virtual.

5.188.2 Member Function Documentation

5.188.2.1 virtual char* Arc::PayloadRawInterface::Buffer (unsigned int num) [pure virtual]

Returns pointer to num'th buffer

Implemented in [Arc::PayloadRaw](#).

5.188.2.2 virtual Size_t Arc::PayloadRawInterface::BufferPos (unsigned int num) const [pure virtual]

Returns position of num'th buffer

Implemented in [Arc::PayloadRaw](#).

```
5.188.2.3 virtual Size_t Arc::PayloadRawInterface::BufferSize ( unsigned int num ) const  
[pure virtual]
```

Returns length of num'th buffer

Implemented in [Arc::PayloadRaw](#).

```
5.188.2.4 virtual char* Arc::PayloadRawInterface::Content ( Size_t pos = -1 ) [pure  
virtual]
```

Get pointer to buffer content at global position 'pos'. By default to beginning of main buffer whatever that means.

Implemented in [Arc::PayloadRaw](#).

```
5.188.2.5 virtual char* Arc::PayloadRawInterface::Insert ( Size_t pos = 0, Size_t size = 0 )  
[pure virtual]
```

Create new buffer at global position 'pos' of size 'size'.

Implemented in [Arc::PayloadRaw](#).

```
5.188.2.6 virtual char* Arc::PayloadRawInterface::Insert ( const char * s, Size_t pos = 0, Size_t  
size = -1 ) [pure virtual]
```

Create new buffer at global position 'pos' of size 'size'. Created buffer is filled with content of memory at 's'. If 'size' is negative content at 's' is expected to be null-terminated.

Implemented in [Arc::PayloadRaw](#).

```
5.188.2.7 virtual char Arc::PayloadRawInterface::operator[] ( Size_t pos ) const [pure  
virtual]
```

Returns content of byte at specified position. Specified position 'pos' is treated as global one and goes through all buffers placed one after another.

Implemented in [Arc::PayloadRaw](#).

```
5.188.2.8 virtual Size_t Arc::PayloadRawInterface::Size ( void ) const [pure virtual]
```

Returns logical size of whole structure.

Implemented in [Arc::PayloadRaw](#).

5.188.2.9 virtual bool Arc::PayloadRawInterface::Truncate (Size_t size) [pure virtual]

Change size of stored information. If size exceeds end of allocated buffer, buffers are not re-allocated, only logical size is extended. Buffers with location behind new size are deallocated.

Implemented in [Arc::PayloadRaw](#).

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

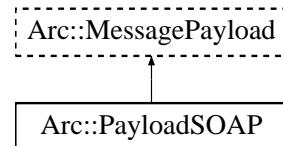
- [PayloadRaw.h](#)

5.189 Arc::PayloadSOAP Class Reference

Payload of [Message](#) with SOAP content.

```
#include <PayloadSOAP.h>
```

Inheritance diagram for Arc::PayloadSOAP:



Public Member Functions

- [PayloadSOAP \(const NS &ns, bool fault=false\)](#)
- [PayloadSOAP \(const SOAPEnvelope &soap\)](#)
- [PayloadSOAP \(const MessagePayload &source\)](#)

5.189.1 Detailed Description

Payload of [Message](#) with SOAP content.

This class combines [MessagePayload](#) with SOAPEnvelope to make it possible to pass SOAP messages through [MCC](#) chain.

5.189.2 Constructor & Destructor Documentation

5.189.2.1 Arc::PayloadSOAP::PayloadSOAP (const NS & ns, bool fault = false)

Constructor - creates new [Message](#) payload

5.189.2.2 Arc::PayloadSOAP::PayloadSOAP (const SOAPEnvelope & soap)

Constructor - creates [Message](#) payload from SOAP document. Provided SOAP document is copied to new object.

5.189.2.3 Arc::PayloadSOAP::PayloadSOAP (const MessagePayload & source)

Constructor - creates SOAP message from payload. [PayloadRawInterface](#) and derived classes are supported.

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

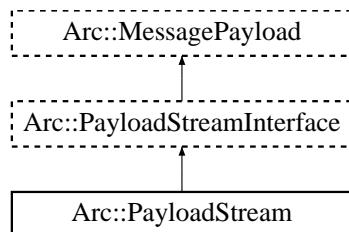
- PayloadSOAP.h

5.190 Arc::PayloadStream Class Reference

POSIX handle as Payload.

```
#include <PayloadStream.h>
```

Inheritance diagram for Arc::PayloadStream:



Public Member Functions

- [PayloadStream](#) (int h=-1)
- virtual [~PayloadStream](#) (void)
- virtual bool [Get](#) (char *buf, int &size)
- virtual bool [Get](#) (std::string &buf)
- virtual std::string [Get](#) (void)
- virtual bool [Put](#) (const char *buf, Size_t size)
- virtual bool [Put](#) (const std::string &buf)
- virtual bool [Put](#) (const char *buf)
- virtual [operator bool](#) (void)
- virtual bool [operator!](#) (void)
- virtual int [Timeout](#) (void) const
- virtual void [Timeout](#) (int to)
- virtual Size_t [Pos](#) (void) const
- virtual Size_t [Size](#) (void) const
- virtual Size_t [Limit](#) (void) const

Protected Attributes

- int `handle_`
- bool `seekable_`

5.190.1 Detailed Description

POSIX handle as Payload.

This is an implemetation of [PayloadStreamInterface](#) for generic POSIX handle.

5.190.2 Constructor & Destructor Documentation

5.190.2.1 `Arc::PayloadStream::PayloadStream(int h = -1)`

true if lseek operation is applicable to open handle Constructor. Attaches to already open handle. Handle is not managed by this class and must be closed by external code.

5.190.2.2 `virtual Arc::PayloadStream::~PayloadStream(void) [inline, virtual]`

Destructor.

5.190.3 Member Function Documentation

5.190.3.1 `virtual bool Arc::PayloadStream::Get(char * buf, int & size) [virtual]`

Extracts information from stream up to 'size' bytes. 'size' contains number of read bytes on exit. Returns true in case of success.

Implements [Arc::PayloadStreamInterface](#).

5.190.3.2 `virtual bool Arc::PayloadStream::Get(std::string & buf) [virtual]`

Read as many as possible (sane amount) of bytes into buf.

Implements [Arc::PayloadStreamInterface](#).

5.190.3.3 `virtual std::string Arc::PayloadStream::Get(void) [inline, virtual]`

Read as many as possible (sane amount) of bytes.

Implements [Arc::PayloadStreamInterface](#).

References Get().

Referenced by Get().

5.190.3.4 virtual `Size_t` `Arc::PayloadStream::Limit(void) const` [inline, virtual]

Returns position at which stream reading will stop if supported. That may be not same as [Size\(\)](#) if instance is meant to provide access to only part of underlying obejct.

Implements [Arc::PayloadStreamInterface](#).

5.190.3.5 virtual `Arc::PayloadStream::operator bool(void)` [inline, virtual]

Returns true if stream is valid.

Implements [Arc::PayloadStreamInterface](#).

References handle_.

5.190.3.6 virtual `bool Arc::PayloadStream::operator!(void)` [inline, virtual]

Returns true if stream is invalid.

Implements [Arc::PayloadStreamInterface](#).

References handle_.

5.190.3.7 virtual `Size_t Arc::PayloadStream::Pos(void) const` [inline, virtual]

Returns current position in stream if supported.

Implements [Arc::PayloadStreamInterface](#).

5.190.3.8 virtual `bool Arc::PayloadStream::Put(const char * buf, Size_t size)` [virtual]

Push 'size' bytes from 'buf' into stream. Returns true on success.

Implements [Arc::PayloadStreamInterface](#).

5.190.3.9 virtual `bool Arc::PayloadStream::Put(const char * buf)` [inline, virtual]

Push null terminated information from 'buf' into stream. Returns true on success.

Implements [Arc::PayloadStreamInterface](#).

References Put().

Referenced by Put().

5.190.3.10 virtual `bool Arc::PayloadStream::Put(const std::string & buf)` [inline, virtual]

Push information from 'buf' into stream. Returns true on success.

Implements [Arc::PayloadStreamInterface](#).

References [Put\(\)](#).

Referenced by [Put\(\)](#).

5.190.3.11 virtual Size_t Arc::PayloadStream::Size(void) const [inline, virtual]

Returns size of underlying object if supported.

Implements [Arc::PayloadStreamInterface](#).

5.190.3.12 virtual int Arc::PayloadStream::Timeout(void) const [inline, virtual]

Query current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implements [Arc::PayloadStreamInterface](#).

5.190.3.13 virtual void Arc::PayloadStream::Timeout(int to) [inline, virtual]

Set current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implements [Arc::PayloadStreamInterface](#).

5.190.4 Field Documentation

5.190.4.1 int Arc::PayloadStream::handle_ [protected]

Timeout for read/write operations

Referenced by [operator bool\(\)](#), and [operator!\(\)](#).

5.190.4.2 bool Arc::PayloadStream::seekable_ [protected]

Handle for operations

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

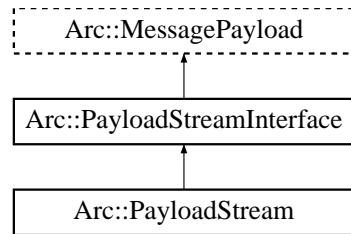
- [PayloadStream.h](#)

5.191 Arc::PayloadStreamInterface Class Reference

Stream-like Payload for [Message](#) object.

```
#include <PayloadStream.h>
```

Inheritance diagram for [Arc::PayloadStreamInterface](#):



Public Member Functions

- virtual bool [Get](#) (char *buf, int &size)=0
- virtual bool [Get](#) (std::string &buf)=0
- virtual std::string [Get](#) (void)=0
- virtual bool [Put](#) (const char *buf, Size_t size)=0
- virtual bool [Put](#) (const std::string &buf)=0
- virtual bool [Put](#) (const char *buf)=0
- virtual [operator bool](#) (void)=0
- virtual bool [operator!](#) (void)=0
- virtual int [Timeout](#) (void) const =0
- virtual void [Timeout](#) (int to)=0
- virtual Size_t [Pos](#) (void) const =0
- virtual Size_t [Size](#) (void) const =0
- virtual Size_t [Limit](#) (void) const =0

5.191.1 Detailed Description

Stream-like Payload for [Message](#) object.

This class is a virtual interface for managing stream-like source and destination. It's supposed to be passed through [MCC](#) chain as payload of [Message](#). It must be treated by MCCs and Services as dynamic payload. This class is purely virtual.

5.191.2 Member Function Documentation

5.191.2.1 virtual bool Arc::PayloadStreamInterface::Get (char * buf, int & size) [pure virtual]

Extracts information from stream up to 'size' bytes. 'size' contains number of read bytes on exit. Returns true in case of success.

Implemented in [Arc::PayloadStream](#).

5.191.2.2 **virtual bool Arc::PayloadStreamInterface::Get (std::string & buf) [pure virtual]**

Read as many as possible (sane amount) of bytes into buf.

Implemented in [Arc::PayloadStream](#).

5.191.2.3 **virtual std::string Arc::PayloadStreamInterface::Get (void) [pure virtual]**

Read as many as possible (sane amount) of bytes.

Implemented in [Arc::PayloadStream](#).

5.191.2.4 **virtual Size_t Arc::PayloadStreamInterface::Limit (void) const [pure virtual]**

Returns position at which stream reading will stop if supported. That may be not same as [Size\(\)](#) if instance is meant to provide access to only part of underlying obejct.

Implemented in [Arc::PayloadStream](#).

5.191.2.5 **virtual Arc::PayloadStreamInterface::operator bool (void) [pure virtual]**

Returns true if stream is valid.

Implemented in [Arc::PayloadStream](#).

5.191.2.6 **virtual bool Arc::PayloadStreamInterface::operator! (void) [pure virtual]**

Returns true if stream is invalid.

Implemented in [Arc::PayloadStream](#).

5.191.2.7 **virtual Size_t Arc::PayloadStreamInterface::Pos (void) const [pure virtual]**

Returns current position in stream if supported.

Implemented in [Arc::PayloadStream](#).

5.191.2.8 **virtual bool Arc::PayloadStreamInterface::Put (const char * buf, Size_t size) [pure virtual]**

Push 'size' bytes from 'buf' into stream. Returns true on success.

Implemented in [Arc::PayloadStream](#).

5.191.2.9 `virtual bool Arc::PayloadStreamInterface::Put(const char * buf) [pure virtual]`

Push null terminated information from 'buf' into stream. Returns true on success.

Implemented in [Arc::PayloadStream](#).

5.191.2.10 `virtual bool Arc::PayloadStreamInterface::Put(const std::string & buf) [pure virtual]`

Push information from 'buf' into stream. Returns true on success.

Implemented in [Arc::PayloadStream](#).

5.191.2.11 `virtual Size_t Arc::PayloadStreamInterface::Size(void) const [pure virtual]`

Returns size of underlying object if supported.

Implemented in [Arc::PayloadStream](#).

5.191.2.12 `virtual int Arc::PayloadStreamInterface::Timeout(void) const [pure virtual]`

Query current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implemented in [Arc::PayloadStream](#).

5.191.2.13 `virtual void Arc::PayloadStreamInterface::Timeout(int to) [pure virtual]`

Set current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implemented in [Arc::PayloadStream](#).

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

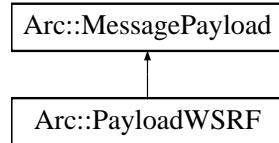
- [PayloadStream.h](#)

5.192 Arc::PayloadWSRF Class Reference

This class combines [MessagePayload](#) with [WSRF](#).

```
#include <PayloadWSRF.h>
```

Inheritance diagram for Arc::PayloadWSRF:



Public Member Functions

- [PayloadWSRF \(const SOAPEnvelope &soap\)](#)
- [PayloadWSRF \(WSRF &wsrp\)](#)
- [PayloadWSRF \(const MessagePayload &source\)](#)

5.192.1 Detailed Description

This class combines [MessagePayload](#) with [WSRF](#).

It's intention is to make it possible to pass [WSRF](#) messages through [MCC](#) chain as one more Payload type.

5.192.2 Constructor & Destructor Documentation

5.192.2.1 Arc::PayloadWSRF::PayloadWSRF (const SOAPEnvelope & soap)

Constructor - creates [Message](#) payload from SOAP message. Returns invalid [WSRF](#) if SOAP does not represent WS-ResourceProperties

5.192.2.2 Arc::PayloadWSRF::PayloadWSRF (WSRF & wsrp)

Constructor - creates [Message](#) payload with acquired [WSRF](#) message. [WSRF](#) message will be destroyed by destructor of this object.

5.192.2.3 Arc::PayloadWSRF::PayloadWSRF (const MessagePayload & source)

Constructor - creates [WSRF](#) message from payload. All classes derived from SOAPEnvelope are supported.

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

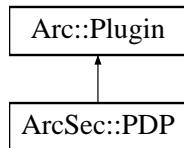
- [PayloadWSRF.h](#)

5.193 ArcSec::PDP Class Reference

Base class for [Policy](#) Decision Point plugins.

```
#include <PDP.h>
```

Inheritance diagram for ArcSec::PDP:



5.193.1 Detailed Description

Base class for [Policy](#) Decision Point plugins.

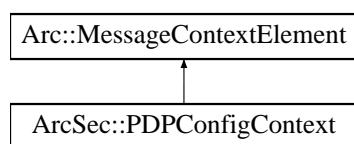
This virtual class defines method `isPermitted()` which processes security related information/attributes in Message and makes security decision - permit (true) or deny (false). Configuration of [PDP](#) is consumed during creation of instance through XML subtree fed to constructor.

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

- PDP.h

5.194 ArcSec::PDPConfigContext Class Reference

Inheritance diagram for ArcSec::PDPConfigContext:

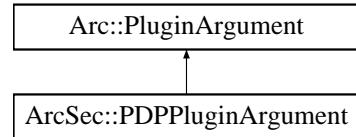


The documentation for this class was generated from the following file:

- PDP.h

5.195 ArcSec::PDPPPluginArgument Class Reference

Inheritance diagram for ArcSec::PDPPPluginArgument:



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

- PDP.h

5.196 Arc::Period Class Reference

Public Member Functions

- [Period \(\)](#)
- [Period \(time_t\)](#)
- [Period \(time_t seconds, uint32_t nanoseconds\)](#)
- [Period \(const std::string &, PeriodBase base=PeriodSeconds\)](#)
- [Period & operator= \(time_t\)](#)
- [Period & operator= \(const Period &\)](#)
- [void SetPeriod \(time_t\)](#)
- [time_t GetPeriod \(\) const](#)
- [const sigc::slot< const char * > * istr \(\) const](#)
- [operator std::string \(\) const](#)
- [bool operator< \(const Period &\) const](#)
- [bool operator> \(const Period &\) const](#)
- [bool operator<= \(const Period &\) const](#)
- [bool operator>= \(const Period &\) const](#)
- [bool operator== \(const Period &\) const](#)
- [bool operator!= \(const Period &\) const](#)

5.196.1 Constructor & Destructor Documentation

5.196.1.1 Arc::Period::Period ()

Default constructor. The period is set to 0 length.

5.196.1.2 Arc::Period::Period (time_t)

Constructor that takes a time_t variable and stores it.

5.196.1.3 Arc::Period::Period (time_t seconds, uint32_t nanoseconds)

Constructor that takes seconds and nanoseconds and stores them.

5.196.1.4 `Arc::Period::Period (const std::string &, PeriodBase base = PeriodSeconds)`

Constructor that tries to convert a string.

5.196.2 Member Function Documentation

5.196.2.1 `time_t Arc::Period::GetPeriod () const`

gets the period

5.196.2.2 `const sigc::slot<const char*>* Arc::Period::istr () const`

For use with [IString](#)

5.196.2.3 `Arc::Period::operator std::string () const`

Returns a string representation of the period.

5.196.2.4 `bool Arc::Period::operator!= (const Period &) const`

Comparing two [Period](#) objects.

5.196.2.5 `bool Arc::Period::operator< (const Period &) const`

Comparing two [Period](#) objects.

5.196.2.6 `bool Arc::Period::operator<= (const Period &) const`

Comparing two [Period](#) objects.

5.196.2.7 `Period& Arc::Period::operator= (time_t)`

Assignment operator from a `time_t`.

5.196.2.8 `Period& Arc::Period::operator= (const Period &)`

Assignment operator from a [Period](#).

5.196.2.9 `bool Arc::Period::operator== (const Period &) const`

Comparing two [Period](#) objects.

5.196.2.10 `bool Arc::Period::operator> (const Period &) const`

Comparing two `Period` objects.

5.196.2.11 `bool Arc::Period::operator>= (const Period &) const`

Comparing two `Period` objects.

5.196.2.12 `void Arc::Period::SetPeriod (time_t)`

sets the period

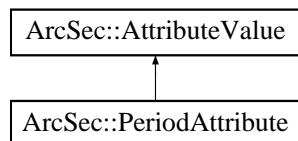
The documentation for this class was generated from the following file:

- `DateTime.h`

5.197 ArcSec::PeriodAttribute Class Reference

```
#include <DateTimeAttribute.h>
```

Inheritance diagram for ArcSec::PeriodAttribute:



Public Member Functions

- `virtual bool equal (AttributeValue *other, bool check_id=true)`
- `virtual std::string encode ()`
- `virtual std::string getType ()`
- `virtual std::string getId ()`

5.197.1 Detailed Description

Format: `datetime"/"duration datetime"/"datetime duration"/"datetime`

5.197.2 Member Function Documentation

5.197.2.1 `virtual std::string ArcSec::PeriodAttribute::encode () [virtual]`

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.197.2.2 virtual bool ArcSec::PeriodAttribute::equal (*AttributeValue* * *value*, *bool check_id* =true) [virtual]

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.197.2.3 virtual std::string ArcSec::PeriodAttribute::getId () [inline, virtual]

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.197.2.4 virtual std::string ArcSec::PeriodAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

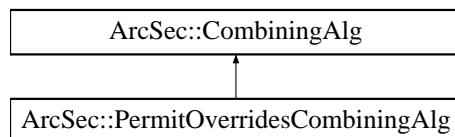
- [DateTimeAttribute.h](#)

5.198 ArcSec::PermitOverridesCombiningAlg Class Reference

Implement the "Permit-Overrides" algorithm.

```
#include <PermitOverridesAlg.h>
```

Inheritance diagram for ArcSec::PermitOverridesCombiningAlg:



Public Member Functions

- **virtual Result combine (*EvaluationCtx* *ctx, std::list<*Policy* *> policies)**
- **virtual const std::string & *getAlgId* (void) const**

5.198.1 Detailed Description

Implement the "Permit-Overrides" algorithm.

Permit-Overrides, scans the policy set which is given as the parameters of "combine" method, if gets "permit" result from any policy, then stops scanning and gives "permit" as result, otherwise gives "deny".

5.198.2 Member Function Documentation

5.198.2.1 virtual Result ArcSec::PermitOverridesCombiningAlg::combine (EvaluationCtx * ctx, std::list< Policy * > policies) [virtual]

If there is one policy which return positive evaluation result, then omit the other policies and return DECISION_PERMIT

Parameters

<i>ctx</i>	This object contains request information which will be used to evaluated against policy.
<i>policies</i>	This is a container which contains policy objects.

Returns

The combined result according to the algorithm.

Implements [ArcSec::CombiningAlg](#).

5.198.2.2 virtual const std::string& ArcSec::PermitOverridesCombiningAlg::getalgId (void) const [inline, virtual]

Get the identifier

Implements [ArcSec::CombiningAlg](#).

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

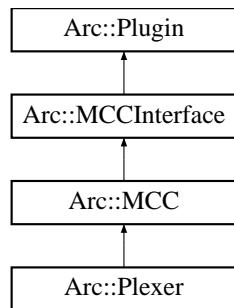
- PermitOverridesAlg.h

5.199 Arc::Plexer Class Reference

The [Plexer](#) class, used for routing messages to services.

```
#include <Plexer.h>
```

Inheritance diagram for Arc::Plexer:



Public Member Functions

- [Plexer \(Config *cfg\)](#)
- [virtual ~Plexer \(\)](#)
- [virtual void Next \(MCCInterface *next, const std::string &label\)](#)
- [virtual MCC_Status process \(Message &request, Message &response\)](#)

Static Public Attributes

- static [Logger logger](#)

5.199.1 Detailed Description

The [Plexer](#) class, used for routing messages to services.

This is the [Plexer](#) class. Its purpose is to route incoming messages to appropriate Services and [MCC](#) chains.

5.199.2 Constructor & Destructor Documentation

5.199.2.1 Arc::Plexer::Plexer (Config * *cfg*)

The constructor.

This is the constructor. Since all member variables are instances of "well-behaving" STL classes, nothing needs to be done.

5.199.2.2 virtual Arc::Plexer::~Plexer () [virtual]

The destructor.

This is the destructor. Since all member variables are instances of "well-behaving" STL classes, nothing needs to be done.

5.199.3 Member Function Documentation

5.199.3.1 `virtual void Arc::Plexer::Next (MCCInterface * next, const std::string & label) [virtual]`

Add reference to next [MCC](#) in chain.

This method is called by [Loader](#) for every potentially labeled link to next component which implements [MCCInterface](#). If next is set NULL corresponding link is removed.

Reimplemented from [Arc::MCC](#).

5.199.3.2 `virtual MCC_Status Arc::Plexer::process (Message & request, Message & response) [virtual]`

Route request messages to appropriate services.

Routes the request message to the appropriate service. Routing is based on the path part of value of the ENDPOINT attribute. Routed message is assigned following attributes: PLEXER:PATTERN - matched pattern, PLEXER:EXTENSION - last unmatched part of ENDPOINT path.

Reimplemented from [Arc::MCC](#).

5.199.4 Field Documentation

5.199.4.1 `Logger Arc::Plexer::logger [static]`

A logger for MCCs.

A logger intended to be the parent of loggers in the different MCCs.

Reimplemented from [Arc::MCC](#).

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

- `Plexer.h`

5.200 Arc::PlexerEntry Class Reference

A pair of label (regex) and pointer to [MCC](#).

```
#include <Plexer.h>
```

5.200.1 Detailed Description

A pair of label (regex) and pointer to [MCC](#).

A helper class that stores a label (regex) and a pointer to a service.

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

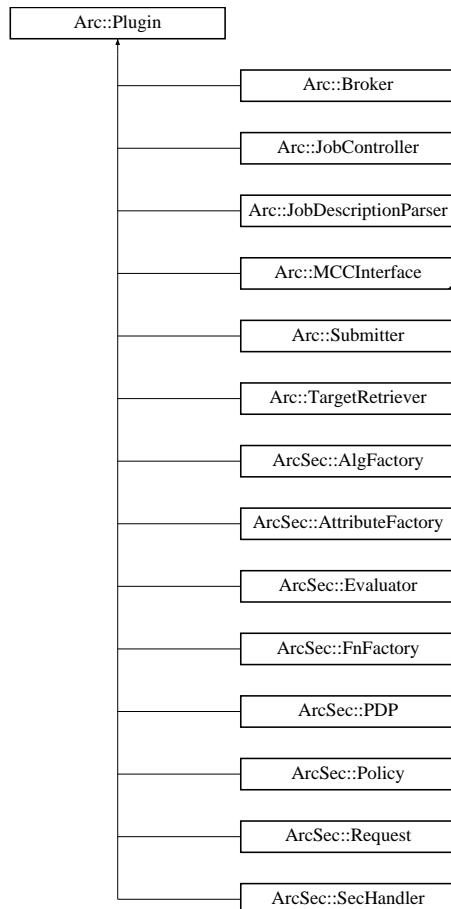
- `Plexer.h`

5.201 Arc::Plugin Class Reference

Base class for loadable ARC components.

```
#include <Plugin.h>
```

Inheritance diagram for Arc::Plugin:



5.201.1 Detailed Description

Base class for loadable ARC components.

All classes representing loadable ARC components must be either descendants of this class or be wrapped by its offspring.

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

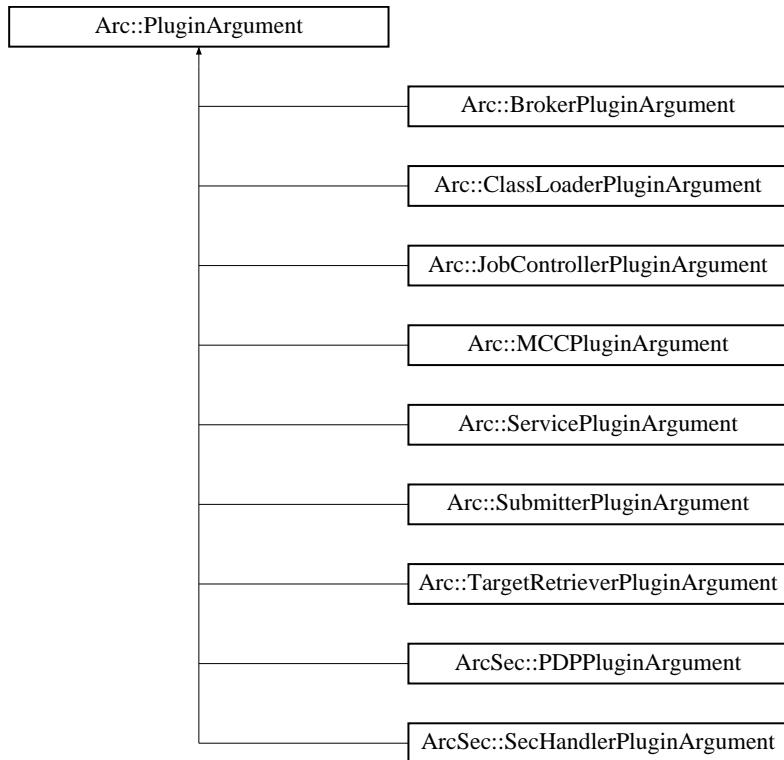
- `Plugin.h`

5.202 Arc::PluginArgument Class Reference

Base class for passing arguments to loadable ARC components.

```
#include <Plugin.h>
```

Inheritance diagram for Arc::PluginArgument:



Public Member Functions

- `PluginsFactory * get_factory (void)`
- `Glib::Module * get_module (void)`

5.202.1 Detailed Description

Base class for passing arguments to loadable ARC components.

During its creation constructor function of ARC loadable component expects instance of class inherited from this one or wrapped in it. Then dynamic type casting is used for obtaining class of expected kind.

5.202.2 Member Function Documentation

5.202.2.1 PluginsFactory* Arc::PluginArgument::get_factory (void)

Returns pointer to factory which instantiated plugin.

Because factory usually destroys/unloads plugins in its destructor it should be safe to keep this pointer inside plugin for later use. But one must always check.

5.202.2.2 Glib::Module* Arc::PluginArgument::get_module (void)

Returns pointer to loadable module/library which contains plugin.

Corresponding factory keeps list of modules till itself is destroyed. So it should be safe to keep that pointer. But care must be taken if module contains persistent plugins. Such modules stay in memory after factory is destroyed. So it is advisable to use obtained pointer only in constructor function of plugin.

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

- Plugin.h

5.203 Arc::PluginDesc Class Reference

Description of plugin.

```
#include <Plugin.h>
```

5.203.1 Detailed Description

Description of plugin.

This class is used for reports

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

- Plugin.h

5.204 Arc::PluginDescriptor Struct Reference

Description of ARC loadable component.

```
#include <Plugin.h>
```

5.204.1 Detailed Description

Description of ARC loadable component.

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

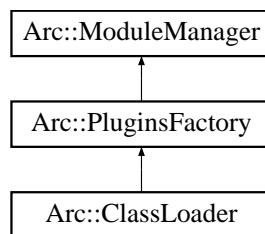
- `Plugin.h`

5.205 Arc::PluginsFactory Class Reference

Generic ARC plugins loader.

```
#include <Plugin.h>
```

Inheritance diagram for `Arc::PluginsFactory`:



Public Member Functions

- `PluginsFactory (XMLNode cfg)`
- `void TryLoad (bool v=true)`
- `bool load (const std::string &name)`
- `bool scan (const std::string &name, ModuleDesc &desc)`
- `void report (std::list<ModuleDesc> &descs)`

Static Public Member Functions

- `static void FilterByKind (const std::string &kind, std::list<ModuleDesc> &descs)`

5.205.1 Detailed Description

Generic ARC plugins loader.

The instance of this class provides functionality of loading pluggable ARC components stored in shared libraries. For more information please check HED documentation. This class is thread-safe - its methods are protected from simultaneous use from multiple threads. Current thread protection implementation is suboptimal and will be revised in future.

5.205.2 Constructor & Destructor Documentation

5.205.2.1 `Arc::PluginsFactory::PluginsFactory (XMLNode cfg)`

Constructor - accepts configuration (not yet used) meant to tune loading of modules.

5.205.3 Member Function Documentation

5.205.3.1 `static void Arc::PluginsFactory::FilterByKind (const std::string & kind, std::list< ModuleDesc > & descs) [static]`

Filter list of modules by kind.

5.205.3.2 `bool Arc::PluginsFactory::load (const std::string & name)`

These methods load module named lib'name' and check if it contains ARC plugin(s) of specified 'kind' and 'name'. If there are no specified plugins or module does not contain any ARC plugins it is unloaded. All loaded plugins are also registered in internal list of this instance of [PluginsFactory](#) class. Returns true if any plugin was loaded.

5.205.3.3 `void Arc::PluginsFactory::report (std::list< ModuleDesc > & descs)`

Provides information about currently loaded modules and plugins.

5.205.3.4 `bool Arc::PluginsFactory::scan (const std::string & name, ModuleDesc & desc)`

Collect information about plugins stored in module(s) with specified names. Returns true if any of specified modules has plugins.

5.205.3.5 `void Arc::PluginsFactory::TryLoad (bool v=true) [inline]`

These methods load module named lib'name', locate plugin constructor functions of specified 'kind' and 'name' (if specified) and call it. Supplied argument affects way plugin instance is created in plugin-specific way. If name of plugin is not specified then all plugins of specified kind are tried with supplied argument till valid instance is created. All loaded plugins are also registered in internal list of this instance of [PluginsFactory](#) class. If search is set to false then no attempt is made to find plugins in loadable modules. Only plugins already loaded with previous calls to `get_instance()` and `load()` are checked. Returns created instance or NULL if failed. Specifies if loadable module may be loaded while looking for analyzing its content. If set to false only *.apd files are checked. Modules without corresponding *.apd will be ignored. Default is true;

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

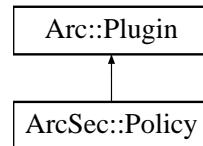
- `Plugin.h`

5.206 ArcSec::Policy Class Reference

Interface for containing and processing different types of policy.

```
#include <Policy.h>
```

Inheritance diagram for ArcSec::Policy:



Public Member Functions

- [Policy \(\)](#)
- [Policy \(const Arc::XMLNode\)](#)
- [Policy \(const Arc::XMLNode, EvaluatorContext *\)](#)
- virtual [operator bool \(void\) const =0](#)
- virtual [MatchResult match \(EvaluationCtx *\)=0](#)
- virtual [Result eval \(EvaluationCtx *\)=0](#)
- virtual void [addPolicy \(Policy *pl\)](#)
- virtual void [setEvaluatorContext \(EvaluatorContext *\)](#)
- virtual void [make_policy \(\)](#)
- virtual std::string [getEffect \(\) const =0](#)
- virtual [EvalResult & getEvalResult \(\)=0](#)
- virtual void [setEvalResult \(EvalResult &res\)=0](#)
- virtual const char * [getEvalName \(\) const =0](#)
- virtual const char * [getName \(\) const =0](#)

5.206.1 Detailed Description

Interface for containing and processing different types of policy.

Basically, each policy object is a container which includes a few elements e.g., ArcPolicySet objects includes a few ArcPolicy objects; ArcPolicy object includes a few ArcRule objects. There is logical relationship between ArcRules or ArcPolicies, which is called combining algorithm. According to algorithm, evaluation results from the elements are combined, and then the combined evaluation result is returned to the up-level.

5.206.2 Constructor & Destructor Documentation

5.206.2.1 ArcSec::Policy::Policy (const Arc::XMLNode) [inline]

Template constructor - creates policy based on XML document.

If XML document is empty then empty policy is created. If it is not empty then it must be valid policy document - otherwise created object should be invalid.

5.206.2.2 ArcSec::Policy::Policy (const Arc::XMLNode , EvaluatorContext *)
[inline]

Template constructor - creates policy based on XML document.

If XML document is empty then empty policy is created. If it is not empty then it must be valid policy document - otherwise created object should be invalid. This constructor is based on the policy node and i the [EvaluatorContext](#) which includes the factory objects for combining algorithm and function

5.206.3 Member Function Documentation

5.206.3.1 virtual void ArcSec::Policy::addPolicy (Policy * pl) [inline, virtual]

Add a policy element to into "this" object

5.206.3.2 virtual Result ArcSec::Policy::eval (EvaluationCtx *) [pure virtual]

Evaluate policy For the <Rule> of [Arc](#), only get the "Effect" from rules; For the <Policy> of [Arc](#), combine the evaluation result from <Rule>; For the <Rule> of XACML, evaluate the <Condition> node by using information from request, and use the "Effect" attribute of <Rule>; For the <Policy> of XACML, combine the evaluation result from <Rule>

5.206.3.3 virtual std::string ArcSec::Policy::getEffect () const [pure virtual]

Get the "Effect" attribute

5.206.3.4 virtual const char* ArcSec::Policy::getEvalName () const [pure virtual]

Get the name of [Evaluator](#) which can evaluate this policy

5.206.3.5 virtual EvalResult& ArcSec::Policy::getEvalResult () [pure virtual]

Get evalution result

5.206.3.6 virtual const char* ArcSec::Policy::getName () const [pure virtual]

Get the name of this policy

5.206.3.7 `virtual void ArcSec::Policy::make_policy() [inline, virtual]`

Parse XMLNode, and construct the low-level Rule object

5.206.3.8 `virtual void ArcSec::Policy::setEvalResult(EvalResult & res) [pure virtual]`

Set evaluation result

5.206.3.9 `virtual void ArcSec::Policy::setEvaluatorContext(EvaluatorContext *) [inline, virtual]`

Set [Evaluator](#) Context for the usage in creating low-level policy object

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

- Policy.h

5.207 ArcSec::PolicyStore::PolicyElement Class Reference

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

- PolicyStore.h

5.208 ArcSec::PolicyParser Class Reference

A interface which will isolate the policy object from actual policy storage (files, urls, database)

```
#include <PolicyParser.h>
```

Public Member Functions

- `virtual Policy * parsePolicy(const Source &source, std::string policyclassname, EvaluatorContext *ctx)`

5.208.1 Detailed Description

A interface which will isolate the policy object from actual policy storage (files, urls, database)

Parse the policy from policy source (e.g. files, urls, database, etc.).

5.208.2 Member Function Documentation

5.208.2.1 `virtual Policy* ArcSec::PolicyParser::parsePolicy (const Source & source, std::string policyclassname, EvaluatorContext * ctx) [virtual]`

Parse policy

Parameters

<code>source</code>	location of the policy
<code>policyclassname</code>	name of the policy for ClassLoader
<code>ctx</code>	EvaluatorContext which includes the <code>**Factory</code>

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

- [PolicyParser.h](#)

5.209 ArcSec::PolicyStore Class Reference

Storage place for policy objects.

```
#include <PolicyStore.h>
```

Data Structures

- class [PolicyElement](#)

Public Member Functions

- `PolicyStore (const std::string &alg, const std::string &policyclassname, EvaluatorContext *ctx)`

5.209.1 Detailed Description

Storage place for policy objects.

5.209.2 Constructor & Destructor Documentation

5.209.2.1 `ArcSec::PolicyStore::PolicyStore (const std::string & alg, const std::string & policyclassname, EvaluatorContext * ctx)`

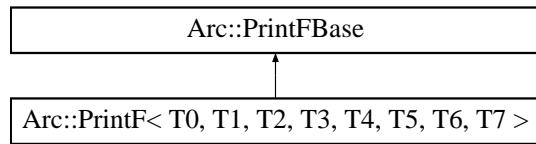
Creates policy store with specified combining algorithm (alg - not used yet), policy name (policyclassname) and context (ctx)

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

- PolicyStore.h

5.210 Arc::PrintF< T0, T1, T2, T3, T4, T5, T6, T7 > Class Template Reference

Inheritance diagram for Arc::PrintF< T0, T1, T2, T3, T4, T5, T6, T7 >:



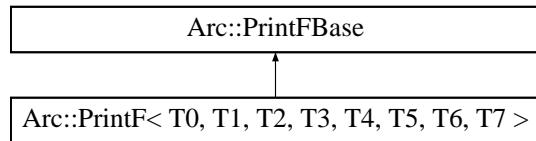
`template<class T0 = int, class T1 = int, class T2 = int, class T3 = int, class T4 = int, class T5 = int, class T6 = int, class T7 = int> class Arc::PrintF< T0, T1, T2, T3, T4, T5, T6, T7 >`

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

- IString.h

5.211 Arc::PrintFBase Class Reference

Inheritance diagram for Arc::PrintFBase:



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

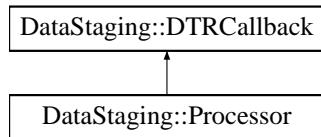
- IString.h

5.212 DataStaging::Processor Class Reference

The [Processor](#) performs pre- and post-transfer operations.

```
#include <Processor.h>
```

Inheritance diagram for DataStaging::Processor:



Data Structures

- class **ThreadArgument**
Class used to pass information to spawned thread.

Public Member Functions

- [Processor \(\)](#)
- [~Processor \(\)](#)
- void [start \(void\)](#)
- void [stop \(void\)](#)
- virtual void [receiveDTR \(DTR &dtr\)](#)

5.212.1 Detailed Description

The [Processor](#) performs pre- and post-transfer operations.

The [Processor](#) takes care of everything that should happen before and after a transfer takes place. Calling [receiveDTR\(\)](#) spawns a thread to perform the required operation depending on the [DTR](#) state.

5.212.2 Member Function Documentation

5.212.2.1 virtual void DataStaging::Processor::receiveDTR (DTR & dtr) [virtual]

Send a [DTR](#) to the [Processor](#).

The [DTR](#) is sent to the [Processor](#) through this method when some long-latency processing is to be performed, eg contacting a remote service. The [Processor](#) spawns a thread to do the processing, and then returns. The thread notifies the scheduler when it is finished.

Implements [DataStaging::DTRCallback](#).

5.212.2.2 void DataStaging::Processor::start (void)

Start [Processor](#).

This method actually does nothing. It is here only to make all classes of data staging to look alike. But it is better to call it before starting to use object because it may do something in the future.

5.212.2.3 void DataStaging::Processor::stop(void)

Stop Processor.

This method sends waits for all started threads to end and exits. Since threads a short-lived it is better to wait rather than interrupt them.

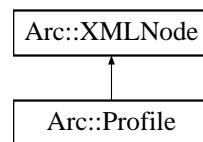
Referenced by ~Processor().

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

- Processor.h

5.213 Arc::Profile Class Reference

Inheritance diagram for Arc::Profile:



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

- Profile.h

5.214 ArcCredential::PROXYCERTINFO_st Struct Reference

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

- Proxycertinfo.h

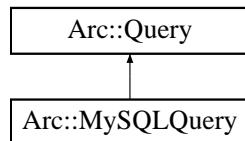
5.215 ArcCredential::PROXYPOLICY_st Struct Reference

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

- Proxycertinfo.h

5.216 Arc::Query Class Reference

Inheritance diagram for Arc::Query:



Public Member Functions

- [Query \(\)](#)
- [Query \(Database *db\)](#)
- virtual [~Query \(\)](#)
- virtual int [get_num_columns \(\)=0](#)
- virtual int [get_num_rows \(\)=0](#)
- virtual bool [execute \(const std::string &sqlstr\)=0](#)
- virtual QueryRowResult [get_row \(int row_number\) const =0](#)
- virtual QueryRowResult [get_row \(\) const =0](#)
- virtual std::string [get_row_field \(int row_number, std::string &field_name\)=0](#)
- virtual bool [get_array \(std::string &sqlstr, QueryArrayResult &result, std::vector< std::string > &arguments\)=0](#)

5.216.1 Constructor & Destructor Documentation

5.216.1.1 Arc::Query::Query () [inline]

Default constructor

5.216.1.2 Arc::Query::Query (Database * db) [inline]

Constructor

Parameters

<i>db</i>	The database object which will be used by Query class to get the database connection
-----------	--

5.216.1.3 virtual Arc::Query::~Query () [inline, virtual]

Deconstructor

5.216.2 Member Function Documentation

5.216.2.1 virtual bool Arc::Query::execute (const std::string & *sqlstr*) [pure virtual]

Execute the query

Parameters

<i>sqlstr</i>	The sql sentence used to query
---------------	--------------------------------

Implemented in [Arc::MySQLQuery](#).

5.216.2.2 virtual bool Arc::Query::get_array (std::string & *sqlstr*, QueryArrayResult & *result*, std::vector< std::string > & *arguments*) [pure virtual]

Query the database by using some parameters into sql sentence e.g. "select table.value from table where table.name = ?"

Parameters

<i>sqlstr</i>	The sql sentence with some parameters marked with "?".
<i>result</i>	The result in an array which includes all of the value in query result.
<i>arguments</i>	The argument list which should exactly correspond with the parameters in sql sentence.

Implemented in [Arc::MySQLQuery](#).

5.216.2.3 virtual int Arc::Query::get_num_columns () [pure virtual]

Get the column number in the query result

Implemented in [Arc::MySQLQuery](#).

5.216.2.4 virtual int Arc::Query::get_num_rows () [pure virtual]

Get the row number in the query result

Implemented in [Arc::MySQLQuery](#).

5.216.2.5 virtual QueryRowResult Arc::Query::get_row (int *row_number*) const [pure virtual]

Get the value of one row in the query result

Parameters

<i>row_number</i>	The number of the row
-------------------	-----------------------

Returns

A vector includes all the values in the row

Implemented in [Arc::MySQLQuery](#).

5.216.2.6 virtual QueryRowResult Arc::Query::get_row() const [pure virtual]

Get the value of one row in the query result, the row number will be automatically increased each time the method is called

Implemented in [Arc::MySQLQuery](#).

5.216.2.7 virtual std::string Arc::Query::get_row_field(int row_number, std::string & field_name) [pure virtual]

Get the value of one specific field in one specific row

Parameters

<code>row_number</code>	The row number inside the query result
<code>field_name</code>	The field name for the value which will be return

Returns

The value of the specified filed in the specified row

Implemented in [Arc::MySQLQuery](#).

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

- DBInterface.h

5.217 Arc::Range< T > Class Template Reference

`template<class T> class Arc::Range< T >`

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

- JobDescription.h

5.218 Arc::Register_Info_Type Struct Reference

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

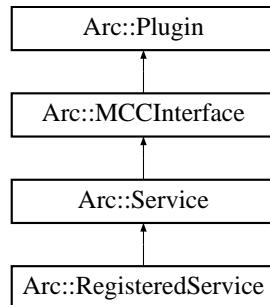
- InfoRegister.h

5.219 Arc::RegisteredService Class Reference

[RegisteredService](#) - extension of [Service](#) performing self-registration.

```
#include <RegisteredService.h>
```

Inheritance diagram for Arc::RegisteredService:



Public Member Functions

- [RegisteredService \(Config *\)](#)

5.219.1 Detailed Description

[RegisteredService](#) - extension of [Service](#) performing self-registration.

5.219.2 Constructor & Destructor Documentation

5.219.2.1 Arc::RegisteredService::RegisteredService (Config *)

Example constructor - Server takes at least its configuration subtree

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

- [RegisteredService.h](#)

5.220 Arc::RegularExpression Class Reference

A regular expression class.

```
#include <ArcRegex.h>
```

Public Member Functions

- [RegularExpression \(\)](#)
- [RegularExpression \(std::string pattern\)](#)
- [RegularExpression \(const RegularExpression ®ex\)](#)
- [~RegularExpression \(\)](#)

- `RegularExpression & operator= (const RegularExpression ®ex)`
- `bool isOk ()`
- `bool hasPattern (std::string str)`
- `bool match (const std::string &str) const`
- `bool match (const std::string &str, std::list< std::string > &unmatched, std::list< std::string > &matched) const`
- `std::string getPattern () const`

5.220.1 Detailed Description

A regular expression class.

This class is a wrapper around the functions provided in regex.h.

5.220.2 Member Function Documentation

5.220.2.1 `bool Arc::RegularExpression::match (const std::string & str, std::list< std::string > & unmatched, std::list< std::string > & matched) const`

Returns true if this regex matches the string provided.

Unmatched parts of the string are stored in 'unmatched'. Matched parts of the string are stored in 'matched'. The first entry in matched is the string that matched the regex, and the following entries are parenthesised elements of the regex

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

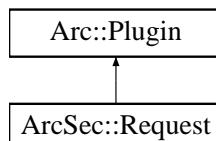
- `ArcRegex.h`

5.221 ArcSec::Request Class Reference

Base class/Interface for request, includes a container for RequestItems and some operations.

```
#include <Request.h>
```

Inheritance diagram for ArcSec::Request:



Public Member Functions

- `virtual ReqItemList getRequestItems () const`

- virtual void `setRequestItems` (`ReqItemList`)
- virtual void `addRequestItem` (`Attrs &, Attrs &, Attrs &, Attrs &`)
- virtual void `setAttributeFactory` (`AttributeFactory *attributefactory`)=0
- virtual void `make_request` ()=0
- virtual const char * `getEvalName` () const =0
- virtual const char * `getName` () const =0
- `Request` ()
- `Request` (const `Source` &)

5.221.1 Detailed Description

Base class/Interface for request, includes a container for RequestItems and some operations.

A `Request` object can has a few <subjects, actions, objects> tuples, i.e. `RequestItem`. The `Request` class and any customized class which inherit from it, should be loadable, which means these classes can be dynamically loaded according to the configuration information, see the example configuration below: <Service name="pdp.service" id="pdp_service"> <pdp:PDPCConfig> <.....> <pdp:Request name="arc.request" /> <.....> </pdp:PDPCConfig> </Service>

There can be different types of subclass which inherit `Request`, such like XACMLRequest, ArcRequest, GACLRequest

5.221.2 Constructor & Destructor Documentation

5.221.2.1 `ArcSec::Request::Request()` [inline]

Default constructor

5.221.2.2 `ArcSec::Request::Request(const Source &)` [inline]

Constructor: Parse request information from a xml stucture in memory

5.221.3 Member Function Documentation

5.221.3.1 `virtual void ArcSec::Request::addRequestItem(Attrs &, Attrs &, Attrs &, Attrs &)` [inline, virtual]

Add request tuple from non-XMLNode

5.221.3.2 `virtual const char* ArcSec::Request::getEvalName() const` [pure virtual]

Get the name of corresponding evaluator

5.221.3.3 `virtual const char* ArcSec::Request::getName() const [pure virtual]`

Get the name of this request

5.221.3.4 `virtual ReqItemList ArcSec::Request::getRequestItems() const [inline, virtual]`

Get all the [RequestItem](#) inside [RequestItem](#) container

5.221.3.5 `virtual void ArcSec::Request::make_request() [pure virtual]`

Create the objects included in [Request](#) according to the node attached to the [Request](#) object

5.221.3.6 `virtual void ArcSec::Request::setAttributeFactory(AttributeFactory * attributefactory) [pure virtual]`

Set the attribute factory for the usage of [Request](#)

5.221.3.7 `virtual void ArcSec::Request::setRequestItems(ReqItemList) [inline, virtual]`

Set the content of the container

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

- Request.h

5.222 ArcSec::RequestAttribute Class Reference

Wrapper which includes [AttributeValue](#) object which is generated according to date type of one specific node in Request.xml.

```
#include <RequestAttribute.h>
```

Public Member Functions

- [RequestAttribute \(Arc::XMLNode &node, AttributeFactory *attrfactory\)](#)
- [RequestAttribute & duplicate \(RequestAttribute &\)](#)

5.222.1 Detailed Description

Wrapper which includes [AttributeValue](#) object which is generated according to date type of one specific node in Request.xml.

5.222.2 Constructor & Destructor Documentation

5.222.2.1 **ArcSec::RequestAttribute::RequestAttribute (Arc::XMLNode & *node*, AttributeFactory * *attrfactory*)**

Constructor - create attribute value object according to the "Type" in the node <Attribute attributeid="urn:arc:subject:voms-attribute" type="string">urn:mace:shibboleth:examples</Attribute>

5.222.3 Member Function Documentation

5.222.3.1 **RequestAttribute& ArcSec::RequestAttribute::duplicate (RequestAttribute &)**

Duplicate the parameter into "this"

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

- RequestAttribute.h

5.223 ArcSec::RequestItem Class Reference

Interface for request item container, <subjects, actions, objects, ctxs> tuple.

```
#include <RequestItem.h>
```

Public Member Functions

- [RequestItem \(Arc::XMLNode &, AttributeFactory *\)](#)

5.223.1 Detailed Description

Interface for request item container, <subjects, actions, objects, ctxs> tuple.

5.223.2 Constructor & Destructor Documentation

5.223.2.1 **ArcSec::RequestItem::RequestItem (Arc::XMLNode & , AttributeFactory *) [inline]**

Constructor

Parameters

<i>node</i>	The XMLNode structure of the request item
<i>attributefactory</i>	The AttributeFactory which will be used to generate RequestAttribute

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

- RequestItem.h

5.224 ArcSec::RequestTuple Class Reference

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

- EvaluationCtx.h

5.225 Arc::ResourceSlotType Class Reference

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

- JobDescription.h

5.226 Arc::ResourcesType Class Reference

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

- JobDescription.h

5.227 ArcSec::Response Class Reference

Container for the evaluation results.

```
#include <Response.h>
```

5.227.1 Detailed Description

Container for the evaluation results.

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

- Response.h

5.228 ArcSec::ResponseItem Class Reference

Evaluation result concerning one [RequestTuple](#).

```
#include <Response.h>
```

5.228.1 Detailed Description

Evaluation result concerning one [RequestTuple](#).

Include the [RequestTuple](#), related XMLNode, the set of policy objects which give positive evaluation result, and the related XMLNode

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

- Response.h

5.229 ArcSec::ResponseList Class Reference

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

- Response.h

5.230 Arc::Run Class Reference

```
#include <Run.h>
```

Public Member Functions

- [Run](#) (const std::string &cmdline)
- [Run](#) (const std::list< std::string > &argv)
- [~Run](#) (void)
- [operator bool](#) (void)
- bool [operator!](#) (void)
- bool [Start](#) (void)
- bool [Wait](#) (int timeout)
- bool [Wait](#) (void)
- int [Result](#) (void)
- bool [Running](#) (void)
- int [ReadStdout](#) (int timeout, char *buf, int size)
- int [ReadStderr](#) (int timeout, char *buf, int size)
- int [WriteStdin](#) (int timeout, const char *buf, int size)
- void [AssignStdout](#) (std::string &str)
- void [AssignStderr](#) (std::string &str)
- void [AssignStdin](#) (std::string &str)
- void [KeepStdout](#) (bool keep=true)
- void [KeepStderr](#) (bool keep=true)
- void [KeepStdin](#) (bool keep=true)
- void [CloseStdout](#) (void)
- void [CloseStderr](#) (void)
- void [CloseStdin](#) (void)

- void [AssignWorkingDirectory](#) (std::string &wd)
- void [Kill](#) (int timeout)
- void [Abandon](#) (void)

Static Public Member Functions

- static void [AfterFork](#) (void)

5.230.1 Detailed Description

This class runs external executable. It is possible to read/write it's standard handles or to redirect them to std::string elements.

5.230.2 Constructor & Destructor Documentation

5.230.2.1 Arc::Run::Run (const std::string & cmdline)

Constructor prepares object to run cmdline

5.230.2.2 Arc::Run::Run (const std::list< std::string > & argv)

Constructor prepares object to run executable and arguments specified in argv

5.230.2.3 Arc::Run::~Run (void)

Destructor kills running executable and releases associated resources

5.230.3 Member Function Documentation

5.230.3.1 void Arc::Run::Abandon (void)

Detach this object from running process. After calling this method instance is not associated with external process anymore. As result destructor will not kill process.

5.230.3.2 static void Arc::Run::AfterFork (void) [static]

Call this method after fork() in child process. It will reinitialize internal structures for new environment. Do not call it in any other case than defined.

5.230.3.3 void Arc::Run::AssignStderr (std::string & str)

Associate stderr handle of executable with string. This method must be called before [Start\(\)](#). str object must be valid as long as this object exists.

5.230.3.4 void Arc::Run::AssignStdin (std::string & str)

Associate stdin handle of executable with string. This method must be called before [Start\(\)](#). str object must be valid as long as this object exists.

5.230.3.5 void Arc::Run::AssignStdout (std::string & str)

Associate stdout handle of executable with string. This method must be called before [Start\(\)](#). str object must be valid as long as this object exists.

5.230.3.6 void Arc::Run::AssignWorkingDirectory (std::string & wd) [inline]

Assign working directory of the running process

5.230.3.7 void Arc::Run::CloseStderr (void)

Closes pipe associated with stderr handle

5.230.3.8 void Arc::Run::CloseStdin (void)

Closes pipe associated with stdin handle

5.230.3.9 void Arc::Run::CloseStdout (void)

Closes pipe associated with stdout handle

5.230.3.10 void Arc::Run::KeepStderr (bool keep = true)

Keep stderr same as parent's if keep = true

5.230.3.11 void Arc::Run::KeepStdin (bool keep = true)

Keep stdin same as parent's if keep = true

5.230.3.12 void Arc::Run::KeepStdout (bool keep = true)

Keep stdout same as parent's if keep = true

5.230.3.13 void Arc::Run::Kill (int timeout)

Kill running executable. First soft kill signal (SIGTERM) is sent to executable. If after timeout seconds executable is still running it's killed completely. Currently this method does not work for Windows OS

5.230.3.14 `Arc::Run::operator bool (void) [inline]`

Returns true if object is valid

5.230.3.15 `bool Arc::Run::operator! (void) [inline]`

Returns true if object is invalid

5.230.3.16 `int Arc::Run::ReadStderr (int timeout, char *buf, int size)`

Read from stderr handle of running executable. Parameter timeout specifies upper limit for which method will block in milliseconds. Negative means infinite. This method may be used while stderr is directed to string. But result is unpredictable. Returns number of read bytes.

5.230.3.17 `int Arc::Run::ReadStdout (int timeout, char *buf, int size)`

Read from stdout handle of running executable. Parameter timeout specifies upper limit for which method will block in milliseconds. Negative means infinite. This method may be used while stdout is directed to string. But result is unpredictable. Returns number of read bytes.

5.230.3.18 `int Arc::Run::Result (void) [inline]`

Returns exit code of execution.

5.230.3.19 `bool Arc::Run::Running (void)`

Return true if execution is going on.

5.230.3.20 `bool Arc::Run::Start (void)`

Starts running executable. This method may be called only once.

5.230.3.21 `bool Arc::Run::Wait (int timeout)`

Wait till execution finished or till timeout seconds expires. Returns true if execution is complete.

5.230.3.22 `bool Arc::Run::Wait (void)`

Wait till execution finished

5.230.3.23 int Arc::Run::WriteStdin (int *timeout*, const char * *buf*, int *size*)

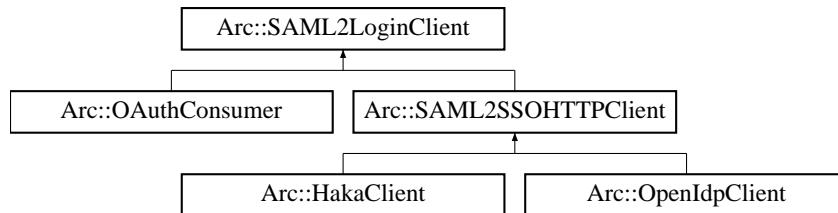
Write to stdin handle of running executable. Parameter timeout specifies upper limit for which method will block in milliseconds. Negative means infinite. This method may be used while stdin is directed to string. But result is unpredictable. Returns number of written bytes.

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

- Run.h

5.231 Arc::SAML2LoginClient Class Reference

Inheritance diagram for Arc::SAML2LoginClient:



Public Member Functions

- [SAML2LoginClient](#) (const [MCCConfig](#) cfg, const [URL](#) url, std::list< std::string > idp_stack)
- virtual [MCC_Status processLogin](#) (const std::string username="", const std::string password "")=0
- [MCC_Status findSimpleSAMLInstallation](#) ()

5.231.1 Constructor & Destructor Documentation

5.231.1.1 Arc::SAML2LoginClient::SAML2LoginClient (const MCCConfig cfg, const URL url, std::list< std::string > idp_stack)

list with the idp for nested wayf For example, Confusa can use betawayf.wayf.dk as an identity provider, which is itself only a wayf and shares the metadata with concrete service providers or even further nested ways. Since due to mutual authentication with metadata, we are obliged to follow the SSO redirects from WAYF to WAYF, the WAYFs are stored in a list.

5.231.2 Member Function Documentation

5.231.2.1 MCC_Status Arc::SAML2LoginClient::findSimpleSAMLInstallation()

find the location of the simplesamlphp installation on the SP side Will be stored in (*sso_pages)[SimpleSAML]

5.231.2.2 virtual MCC_Status Arc::SAML2LoginClient::processLogin (const std::string username = "", const std::string password = "") [pure virtual]

Base interface for all login procedures

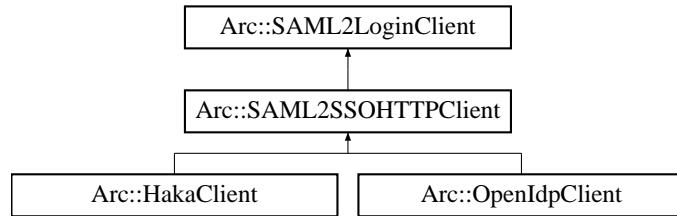
Implemented in [Arc::OAuthConsumer](#), and [Arc::SAML2SSOHTTPClient](#).

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

- SAML2LoginClient.h

5.232 Arc::SAML2SSOHTTPClient Class Reference

Inheritance diagram for Arc::SAML2SSOHTTPClient:



Public Member Functions

- [MCC_Status processLogin](#) (const std::string username, const std::string password)
- [MCC_Status parseDN](#) (std::string *dn)
- [MCC_Status approveCSR](#) (const std::string approve_page)
- [MCC_Status pushCSR](#) (const std::string b64_pub_key, const std::string pub_key_hash, std::string *approve_page)
- [MCC_Status storeCert](#) (const std::string cert_path, const std::string auth_token, const std::string b64_dn)

Protected Member Functions

- virtual [MCC_Status processIdPLogin](#) (const std::string username, const std::string password)=0
- virtual [MCC_Status processConsent](#) ()=0
- virtual [MCC_Status processIdP2Confusa](#) ()=0

5.232.1 Member Function Documentation

5.232.1.1 **MCC_Status Arc::SAML2SSOHTTPClient::approveCSR (const std::string approve_page) [virtual]**

Simulate click on the approve cert signing request link

Implements [Arc::SAML2LoginClient](#).

5.232.1.2 **MCC_Status Arc::SAML2SSOHTTPClient::parseDN (std::string * dn) [virtual]**

Parse the used DN from the Confusa about_you page

Implements [Arc::SAML2LoginClient](#).

5.232.1.3 **virtual MCC_Status Arc::SAML2SSOHTTPClient::processConsent () [protected, pure virtual]**

If the IdP has a consent module and the user has not saved her consent, this method will ask the user for consent to transmission of her data to Confusa

Implemented in [Arc::HakaClient](#), and [Arc::OpenIdpClient](#).

5.232.1.4 **virtual MCC_Status Arc::SAML2SSOHTTPClient::processIdP2Confusa () [protected, pure virtual]**

Redirects the user back from identity provider to the Confusa SP

Implemented in [Arc::HakaClient](#), and [Arc::OpenIdpClient](#).

5.232.1.5 **virtual MCC_Status Arc::SAML2SSOHTTPClient::processIdPLogin (const std::string username, const std::string password) [protected, pure virtual]**

Actual identity provider parsers for next three methods implemented in subdirectory idp/

Parse identity provider login page and submit username and password in the pre-specified way

Implemented in [Arc::HakaClient](#), and [Arc::OpenIdpClient](#).

5.232.1.6 **MCC_Status Arc::SAML2SSOHTTPClient::processLogin (const std::string username, const std::string password) [virtual]**

Models complete SAML2 WebSSO authN flow with start -> WAYF -> Login -> (consent) -> start

Implements [Arc::SAML2LoginClient](#).

5.232.1.7 **MCC_Status** `Arc::SAMLSSOHTTPClient::pushCSR (const std::string b64_pub_key, const std::string pub_key_hash, std::string * approve_page) [virtual]`

Send the cert signing request to Confusa for signing

Implements [Arc::SAML2LoginClient](#).

5.232.1.8 **MCC_Status** `Arc::SAMLSSOHTTPClient::storeCert (const std::string cert_path, const std::string auth_token, const std::string b64_dn) [virtual]`

Download the signed certificate from Confusa and store it locally

Implements [Arc::SAML2LoginClient](#).

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

- `SAML2LoginClient.h`

5.233 Arc::SAMLToken Class Reference

Class for manipulating SAML Token [Profile](#).

```
#include <SAMLToken.h>
```

Public Types

- enum [SAMLVersion](#)

Public Member Functions

- `SAMLToken (SOAPEnvelope &soap)`
- `SAMLToken (SOAPEnvelope &soap, const std::string &certfile, const std::string &keyfile, SAMLVersion saml_version=SAML2, XMLNode saml_assertion=XMLNode())`
- `~SAMLToken (void)`
- `operator bool (void)`
- `bool Authenticate (const std::string &cafile, const std::string &capath)`
- `bool Authenticate (void)`

5.233.1 Detailed Description

Class for manipulating SAML Token [Profile](#).

This class is for generating/consuming SAML Token profile. See WS-Security SAML Token [Profile](#) v1.1 (www.oasis-open.org/committees/wss) Currently this class is used by samltoken handler (will appear in `src/hed/pdc/samltokensh/`) It is not a must to directly call this class. If we need to use SAML Token functionality, we only need to

configure the samltoken handler into service and client. Currently, only a minor part of the specification has been implemented.

About how to identify and reference security token for signing message, currently, only the "SAML Assertion Referenced from KeyInfo" (part 3.4.2 of WS-Security SAML Token [Profile](#) v1.1 specification) is supported, which means the implementation can only process SAML assertion "referenced from KeyInfo", and also can only generate SAML Token with SAML assertion "referenced from KeyInfo". More complete support need to implement.

About subject confirmation method, the implementation can process "hold-of-key" (part 3.5.1 of WS-Security SAML Token [Profile](#) v1.1 specification) subject subject confirmation method.

About SAML verson, the implementation can process SAML assertion with SAML version 1.1 and 2.0; can only generate SAML assertion with SAML verson 2.0.

In the SAML Token profile, for the hold-of-key subject confirmation method, there are three interaction parts: the attesting entity, the relying party and the issuing authority. In the hold-of-key subject confirmation method, it is the attesting entity's subject identity which will be inserted into the SAML assertion.

Firstly the attesting entity authenticates to issuing authority by using some authentication scheme such as WSS x509 Token profile (Alterbatively the username/password authentication scheme or other different authentication scheme can also be used, unless the issuing authority can retrive the key from a trusted certificate server after firmly establishing the subject's identity under the username/password scheme). So then issuing authority is able to make a definitive statement (sign a SAML assertion) about an act of authentication that has already taken place.

The attesting entity gets the SAML assertion and then signs the soap message together with the assertion by using its private key (the relevant certificate has been authenticated by issuing authority, and its relevant public key has been put into SubjectConfirmation element under saml assertion by issuing authority. Only the actual owner of the saml assertion can do this, as only the subject possesses the private key paired with the public key in the assertion. This establishes an irrefutable connection between the author of the SOAP message and the assertion describing an authentication event.)

The relying party is supposed to trust the issuing authority. When it receives a message from the asserting entity, it will check the saml assertion based on its predetermined trust relationship with the SAML issuing authority, and check the signature of the soap message based on the public key in the saml assertion without directly trust relationship with attesting entity (subject owner).

5.233.2 Member Enumeration Documentation

5.233.2.1 enum Arc::SAMLToken::SAMLVersion

Since the specification SAMLVersion is for distinguishing two types of saml version. It is used as the parameter of constructor.

5.233.3 Constructor & Destructor Documentation

5.233.3.1 Arc::SAMLToken::SAMLToken (*SOAPEnvelope & soap*)

Constructor. Parse SAML Token information from SOAP header. SAML Token related information is extracted from SOAP header and stored in class variables. And then it the [SAMLToken](#) object will be used for authentication.

Parameters

<i>soap</i>	The SOAP message which contains the SAMLToken in the soap header
-------------	--

5.233.3.2 Arc::SAMLToken::SAMLToken (*SOAPEnvelope & soap, const std::string & certfile, const std::string & keyfile, SAMLVersion saml_version = SAML2, XMLNode saml_assertion = XMLNode ()*)

Constructor. Add SAML Token information into the SOAP header. Generated token contains elements SAML token and signature, and is meant to be used for authentication on the consuming side. This constructor is for a specific SAML Token profile usage, in which the attesting entity signs the SAML assertion for itself (self-sign). This usage implicitly requires that the relying party trust the attesting entity. More general (requires issuing authority) usage will be provided by other constructor. And the under-developing SAML service will be used as the issuing authority.

Parameters

<i>soap</i>	The SOAP message to which the SAML Token will be inserted.
<i>certfile</i>	The certificate file.
<i>keyfile</i>	The key file which will be used to create signature.
<i>samlversion</i>	The SAML version, only SAML2 is supported currently.
<i>samlassertion</i>	The SAML assertion got from 3rd party, and used for protecting the SOAP message; If not present, then self-signed assertion will be generated.

5.233.3.3 Arc::SAMLToken::~SAMLToken (*void*)

Deconstructor. Nothing to be done except finalizing the xmlsec library.

5.233.4 Member Function Documentation

5.233.4.1 bool Arc::SAMLToken::Authenticate (*const std::string & cafile, const std::string & capath*)

Check signature by using the trusted certificates It is used by relying parting after calling [SAMLToken\(SOAPEnvelope& soap\)](#) This method will check the SAML assertion based on the trusted certificates specified as parameter cafile or capath; and also check the signature to soap message (the signature is generated by attesting entity by signing soap body together with SAML assertion) by using the public key inside SAML assertion.

Parameters

<i>cafile</i>	ca file
<i>capath</i>	ca directory

5.233.4.2 `bool Arc::SAMLToken::Authenticate (void)`

Check signature by using the cert information in soap message

5.233.4.3 `Arc::SAMLToken::operator bool (void)`

Returns true if constructor succeeded

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

- SAMLToken.h

5.234 Arc::ScalableTime< T > Class Template Reference

`template<class T> class Arc::ScalableTime< T >`

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

- JobDescription.h

5.235 Arc::ScalableTime< int > Class Template Reference

`template<> class Arc::ScalableTime< int >`

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

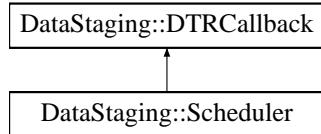
- JobDescription.h

5.236 DataStaging::Scheduler Class Reference

The **Scheduler** is the control centre of the data staging framework.

```
#include <Scheduler.h>
```

Inheritance diagram for DataStaging::Scheduler:



Public Member Functions

- `Scheduler ()`
- `~Scheduler ()`
- `void SetSlots (int pre_processor=0, int post_processor=0, int delivery=0, int delivery_emergency=0)`
- `void AddURLMapping (const Arc::URL &template_url, const Arc::URL &replacement_url, const Arc::URL &access_url=Arc::URL())`
- `void SetURLMapping (const Arc::URLMap &mapping=Arc::URLMap())`
- `void SetPreferredPattern (const std::string &pattern)`
- `void SetTransferShares (const TransferShares &shares)`
- `void AddSharePriority (const std::string &name, int priority)`
- `void SetSharePriorities (const std::map< std::string, int > &shares)`
- `void SetShareType (TransferShares::ShareType share_type)`
- `void SetTransferParameters (const TransferParameters ¶ms)`
- `void SetDeliveryServices (const std::vector< Arc::URL > &endpoints)`
- `void SetDumpLocation (const std::string &location)`
- `bool start (void)`
- `virtual void receiveDTR (DTR &dtr)`
- `bool cancelDTRs (const std::string &jobid)`
- `bool stop ()`

5.236.1 Detailed Description

The `Scheduler` is the control centre of the data staging framework.

The `Scheduler` manages a global list of DTRs and schedules when they should go into the next state or be sent to other processes. The `DTR` priority is used to decide each DTR's position in a queue.

5.236.2 Member Function Documentation

5.236.2.1 virtual void DataStaging::Scheduler::receiveDTR (DTR & dtr) [virtual]

Callback method implemented from `DTRCallback`.

This method is called by the generator when it wants to pass a `DTR` to the scheduler.

Implements `DataStaging::DTRCallback`.

5.236.2.2 bool DataStaging::Scheduler::start(void)

Start scheduling activity.

This method must be called after all configuration parameters are set properly. [Scheduler](#) can be stopped either by calling [stop\(\)](#) method or by destroying its instance.

5.236.2.3 bool DataStaging::Scheduler::stop()

Tell the [Scheduler](#) to shut down all threads and exit.

All active DTRs are cancelled and this method waits until they finish (all DTRs go to CANCELLED state)

Referenced by [~Scheduler\(\)](#).

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

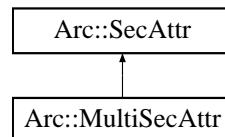
- Scheduler.h

5.237 Arc::SecAttr Class Reference

This is an abstract interface to a security attribute.

```
#include <SecAttr.h>
```

Inheritance diagram for Arc::SecAttr:



Public Member Functions

- [SecAttr \(\)](#)
- bool [operator== \(const SecAttr &b\) const](#)
- bool [operator!= \(const SecAttr &b\) const](#)
- virtual [operator bool \(\) const](#)
- virtual bool [Export \(SecAttrFormat format, std::string &val\) const](#)
- virtual bool [Export \(SecAttrFormat format, XMLNode &val\) const](#)
- virtual bool [Import \(SecAttrFormat format, const std::string &val\)](#)
- virtual std::string [get \(const std::string &id\) const](#)
- virtual std::list< std::string > [getAll \(const std::string &id\) const](#)

Static Public Attributes

- static [SecAttrFormat ARCAuth](#)
- static [SecAttrFormat XACML](#)
- static [SecAttrFormat SAML](#)
- static [SecAttrFormat GACL](#)

5.237.1 Detailed Description

This is an abstract interface to a security attribute.

This class is meant to be inherited to implement security attributes. Depending on what data it needs to store inheriting classes may need to implement constructor and destructor. They must however override the equality and the boolean operators. The equality is meant to compare security attributes. The prototype implies that all attributes are comparable to all others. This behaviour should be modified as needed by using dynamic_cast operations. The boolean cast operation is meant to embody "nullness" if that is applicable to the particular type.

5.237.2 Member Function Documentation

5.237.2.1 virtual bool Arc::SecAttr::Export ([SecAttrFormat format](#), std::string & val) const [virtual]

Convert internal structure into specified format. Returns false if format is not supported/suitable for this attribute.

5.237.2.2 virtual bool Arc::SecAttr::Export ([SecAttrFormat format](#), XMLNode & val) const [virtual]

Convert internal structure into specified format. Returns false if format is not supported/suitable for this attribute. XML node referenced by is turned into top level element of specified format.

Reimplemented in [Arc::MultiSecAttr](#).

5.237.2.3 virtual std::string Arc::SecAttr::get (const std::string & id) const [virtual]

Access to specific item of the security attribute. If there are few items of same id the first one is presented. It is meant to be used for tightly coupled SecHandlers and provides more effective interface than Export.

5.237.2.4 virtual std::list<std::string> Arc::SecAttr::getAll (const std::string & id) const [virtual]

Access to specific items of the security attribute. This method returns all items which have id assigned. It is meant to be used for tightly coupled SecHandlers and provides

more effective interface than Export.

5.237.2.5 virtual bool Arc::SecAttr::Import (SecAttrFormat format, const std::string & val) [virtual]

Fills internal structure from external object of specified format. Returns false if failed to do. The usage pattern for this method is not defined and it is provided only to make class symmetric. Hence its implementation is not required yet.

5.237.2.6 virtual Arc::SecAttr::operator bool () const [virtual]

This function should return false if the value is to be considered null, e.g. if it hasn't been set or initialized. In other cases it should return true.

Reimplemented in [Arc::MultiSecAttr](#).

5.237.2.7 bool Arc::SecAttr::operator!= (const SecAttr & b) const [inline]

This is a convenience function to allow the usage of "not equal" conditions and need not be overridden.

5.237.2.8 bool Arc::SecAttr::operator== (const SecAttr & b) const [inline]

This function should (in inheriting classes) return true if this and b are considered to represent same content. Identifying and restricting the type of b should be done using dynamic_cast operations. Currently it is not defined how comparison methods to be used. Hence their implementation is not required.

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

- SecAttr.h

5.238 Arc::SecAttrFormat Class Reference

Export/import format.

```
#include <SecAttr.h>
```

5.238.1 Detailed Description

Export/import format.

Format is identified by textual identity string. Class description includes basic formats only. That list may be extended.

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

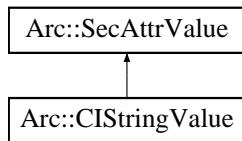
- SecAttr.h

5.239 Arc::SecAttrValue Class Reference

This is an abstract interface to a security attribute.

```
#include <SecAttrValue.h>
```

Inheritance diagram for Arc::SecAttrValue:



Public Member Functions

- bool [operator== \(SecAttrValue &b\)](#)
- bool [operator!= \(SecAttrValue &b\)](#)
- virtual [operator bool \(\)](#)

5.239.1 Detailed Description

This is an abstract interface to a security attribute.

This class is meant to be inherited to implement security attributes. Depending on what data it needs to store inheriting classes may need to implement constructor and destructor. They must however override the equality and the boolean operators. The equality is meant to compare security attributes. The prototype implies that all attributes are comparable to all others. This behaviour should be modified as needed by using dynamic_cast operations. The boolean cast operation is meant to embody "nullness" if that is applicable to the particular type.

5.239.2 Member Function Documentation

5.239.2.1 virtual Arc::SecAttrValue::operator bool () [virtual]

This function should return false if the value is to be considered null, e.g. if it hasn't been set or initialized. In other cases it should return true.

Reimplemented in [Arc::CIStringValue](#).

5.239.2.2 bool Arc::SecAttrValue::operator!= (SecAttrValue & b)

This is a convenience function to allow the usage of "not equal" conditions and need not be overridden.

5.239.2.3 bool Arc::SecAttrValue::operator== (SecAttrValue & b)

This function should (in inheriting classes) return true if this and b are considered to be the same. Identifying and restricting the type of b should be done using dynamic_cast operations.

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

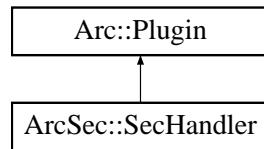
- SecAttrValue.h

5.240 ArcSec::SecHandler Class Reference

Base class for simple security handling plugins.

```
#include <SecHandler.h>
```

Inheritance diagram for ArcSec::SecHandler:



5.240.1 Detailed Description

Base class for simple security handling plugins.

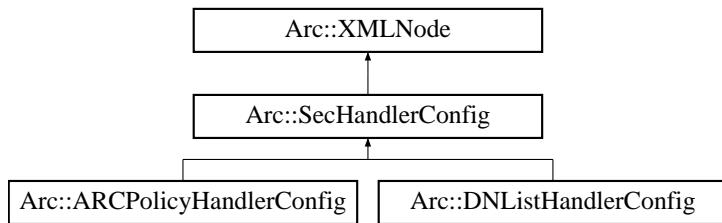
This virtual class defines method Handle() which processes security related information/attributes in Message and optionally makes security decision. Instances of such classes are normally arranged in chains and are called on incoming and outgoing messages in various MCC and Service plugins. Return value of Handle() defines either processing should continue (true) or stop with error (false). Configuration of [SecHandler](#) is consumed during creation of instance through XML subtree fed to constructor.

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

- SecHandler.h

5.241 Arc::SecHandlerConfig Class Reference

Inheritance diagram for Arc::SecHandlerConfig:



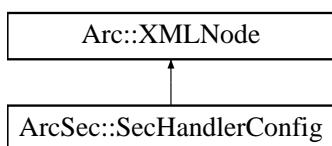
The documentation for this class was generated from the following file:

- ClientInterface.h

5.242 ArcSec::SecHandlerConfig Class Reference

#include <SecHandler.h>

Inheritance diagram for ArcSec::SecHandlerConfig:



5.242.1 Detailed Description

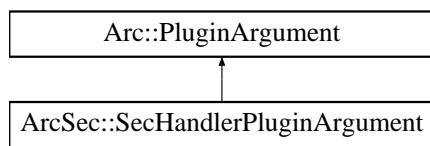
Helper class to create [Security](#) Handler configuration

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

- SecHandler.h

5.243 ArcSec::SecHandlerPluginArgument Class Reference

Inheritance diagram for ArcSec::SecHandlerPluginArgument:



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

- SecHandler.h

5.244 ArcSec::Security Class Reference

Common stuff used by security related classes.

```
#include <Security.h>
```

5.244.1 Detailed Description

Common stuff used by security related classes.

This class is just a place where to put common stuff that is used by security related classes. So far it only contains a logger.

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

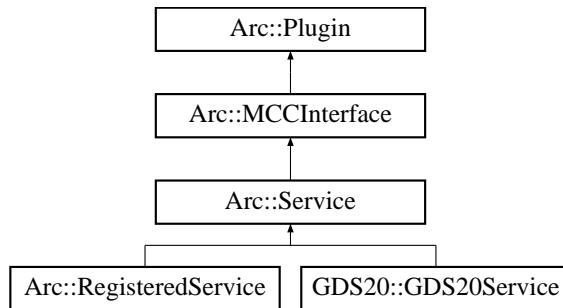
- Security.h

5.245 Arc::Service Class Reference

[Service](#) - last component in a [Message Chain](#).

```
#include <Service.h>
```

Inheritance diagram for Arc::Service:



Public Member Functions

- [Service \(Config *\)](#)
- virtual void [AddSecHandler \(Config *cfg, ArcSec::SecHandler *sechandler, const std::string &label=""\)](#)
- virtual bool [RegistrationCollector \(XMLNode &doc\)](#)
- virtual std::string [getID \(\)](#)

Protected Member Functions

- bool [ProcessSecHandlers](#) ([Message](#) &message, const std::string &label="") const

Protected Attributes

- std::map< std::string, std::list< [ArcSec::SecHandler](#) * > > [sechandlers_](#)

Static Protected Attributes

- static [Logger](#) logger

5.245.1 Detailed Description

[Service](#) - last component in a [Message](#) Chain.

This class which defines interface and common functionality for every [Service](#) plugin. Interface is made of method [process\(\)](#) which is called by [Plexer](#) or [MCC](#) class. There is one [Service](#) object created for every service description processed by [Loader](#) class objects. Classes derived from [Service](#) class must implement [process\(\)](#) method of [MCCInterface](#). It is up to developer how internal state of service is stored and communicated to other services and external utilities. [Service](#) is free to expect any type of payload passed to it and generate any payload as well. Useful types depend on MCCs in chain which leads to that service. For example if service is expected to be linked to SOAP [MCC](#) it must accept and generate messages with [PayloadSOAP](#) payload. Method [process\(\)](#) of class derived from [Service](#) class may be called concurrently in multiple threads. Developers must take that into account and write thread-safe implementation. Simple example of service is provided in /src/tests/echo/echo.cpp of source tree. The way to write client counterpart of corresponding service is undefined yet. For example see /src/tests/echo/test.cpp .

5.245.2 Constructor & Destructor Documentation

5.245.2.1 [Arc::Service::Service \(Config * \)](#)

Example constructor - Server takes at least its configuration subtree

5.245.3 Member Function Documentation

5.245.3.1 [virtual void Arc::Service::AddSecHandler \(Config * cfg, ArcSec::SecHandler * sechandler, const std::string & label = " " \) \[virtual\]](#)

Add security components/handlers to this [MCC](#). For more information please see description of [MCC::AddSecHandler](#)

5.245.3.2 virtual std::string Arc::Service::getID() [inline, virtual]

[Service](#) may implement own service identifier gathering method. This method return identifier of service which is used for registering it Information Services.

5.245.3.3 bool Arc::Service::ProcessSecHandlers(Message & message, const std::string & label = " ") const [protected]

Executes security handlers of specified queue. For more information please see description of [MCC::ProcessSecHandlers](#)

5.245.3.4 virtual bool Arc::Service::RegistrationCollector(XMLNode & doc) [virtual]

[Service](#) specific registration collector, used for generate service registrations. In implemented service this method should generate [GLUE2](#) document with part of service description which service wishes to advertise to Information Services.

5.245.4 Field Documentation

5.245.4.1 Logger Arc::Service::logger [static, protected]

[Logger](#) object used to print messages generated by this class.

5.245.4.2 std::map<std::string, std::list<ArcSec::SecHandler*> > Arc::Service::sechandlers_ [protected]

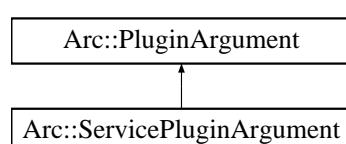
Set of labeled authentication and authorization handlers. [MCC](#) calls sequence of handlers at specific point depending on associated identifier. in most cases those are "in" and "out" for incoming and outgoing messages correspondingly.

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

- Service.h

5.246 Arc::ServicePluginArgument Class Reference

Inheritance diagram for Arc::ServicePluginArgument:



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

- Service.h

5.247 Arc::SharedMutex Class Reference

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

- Thread.h

5.248 Arc::SimpleCondition Class Reference

Simple triggered condition.

```
#include <Thread.h>
```

Public Member Functions

- void [lock](#) (void)
- void [unlock](#) (void)
- void [signal](#) (void)
- void [signal_nonblock](#) (void)
- void [broadcast](#) (void)
- void [wait](#) (void)
- void [wait_nonblock](#) (void)
- bool [wait](#) (int t)
- void [reset](#) (void)

5.248.1 Detailed Description

Simple triggered condition.

Provides condition and semaphor objects in one element.

5.248.2 Member Function Documentation

5.248.2.1 void Arc::SimpleCondition::broadcast (void) [inline]

Signal about condition to all waiting threads

5.248.2.2 void Arc::SimpleCondition::lock (void) [inline]

Acquire semaphor

5.248.2.3 void Arc::SimpleCondition::reset (void) [inline]

Reset object to initial state

5.248.2.4 void Arc::SimpleCondition::signal (void) [inline]

Signal about condition

5.248.2.5 void Arc::SimpleCondition::signal_nonblock (void) [inline]

Signal about condition without using semaphor

5.248.2.6 void Arc::SimpleCondition::unlock (void) [inline]

Release semaphor

5.248.2.7 bool Arc::SimpleCondition::wait (int t) [inline]

Wait for condition no longer than t milliseconds

5.248.2.8 void Arc::SimpleCondition::wait (void) [inline]

Wait for condition

5.248.2.9 void Arc::SimpleCondition::wait_nonblock (void) [inline]

Wait for condition without using semaphor

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

- Thread.h

5.249 Arc::SimpleCounter Class Reference

Public Member Functions

- bool [wait](#) (int t)

5.249.1 Member Function Documentation

5.249.1.1 bool Arc::SimpleCounter::wait (int t) [inline]

Wait for condition no longer than t milliseconds

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

- Thread.h

5.250 Arc::SOAPMessage Class Reference

[Message](#) restricted to SOAP payload.

```
#include <SOAPMessage.h>
```

Public Member Functions

- [SOAPMessage \(void\)](#)
- [SOAPMessage \(long msg_ptr_addr\)](#)
- [SOAPMessage \(Message &msg\)](#)
- [~SOAPMessage \(void\)](#)
- [SOAPEnvelope * Payload \(void\)](#)
- [void Payload \(SOAPEnvelope *new_payload\)](#)
- [MessageAttributes * Attributes \(void\)](#)

5.250.1 Detailed Description

[Message](#) restricted to SOAP payload.

This is a special [Message](#) intended to be used in language bindings for programming languages which are not flexible enough to support all kinds of Payloads. It is passed through chain of MCCs and works like the [Message](#) but can carry only SOAP content.

5.250.2 Constructor & Destructor Documentation

5.250.2.1 Arc::SOAPMessage::SOAPMessage (void) [inline]

Dummy constructor

5.250.2.2 Arc::SOAPMessage::SOAPMessage (long msg_ptr_addr)

Copy constructor. Used by language bindigs

5.250.2.3 Arc::SOAPMessage::SOAPMessage (Message & msg)

Copy constructor. Ensures shallow copy.

5.250.2.4 Arc::SOAPMessage::~SOAPMessage (void)

Destructor does not affect referred objects

5.250.3 Member Function Documentation

5.250.3.1 MessageAttributes* Arc::SOAPMessage::Attributes (void) [inline]

Returns a pointer to the current attributes object or NULL if no attributes object has been assigned.

5.250.3.2 SOAPEnvelope* Arc::SOAPMessage::Payload (void)

Returns pointer to current payload or NULL if no payload assigned.

5.250.3.3 void Arc::SOAPMessage::Payload (SOAPEnvelope * new_payload)

Replace payload with a COPY of new one

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

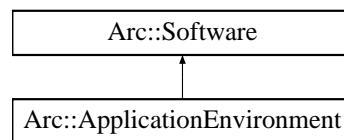
- SOAPMessage.h

5.251 Arc::Software Class Reference

Used to represent software (names and version) and comparison.

```
#include <Software.h>
```

Inheritance diagram for Arc::Software:



Public Types

- enum ComparisonOperatorEnum {

NOTEQUAL = 0, EQUAL = 1, GREATERTHAN = 2, LESSTHAN = 3,

GREATERTHANOREQUAL = 4, LESSTHANOREQUAL = 5
 }
- typedef bool(Software::* ComparisonOperator)(const Software &) const

Public Member Functions

- `Software ()`
- `Software (const std::string &name_version)`
- `Software (const std::string &name, const std::string &version)`
- `Software (const std::string &family, const std::string &name, const std::string &version)`
- `bool empty () const`
- `bool operator== (const Software &sw) const`
- `bool operator!= (const Software &sw) const`
- `bool operator> (const Software &sw) const`
- `bool operator< (const Software &sw) const`
- `bool operator>= (const Software &sw) const`
- `bool operator<= (const Software &sw) const`
- `std::string operator() () const`
- `operator std::string (void) const`
- `const std::string & getFamily () const`
- `const std::string & getName () const`
- `const std::string & getVersion () const`

Static Public Member Functions

- `static ComparisonOperator convert (const ComparisonOperatorEnum &co)`
- `static std::string toString (ComparisonOperator co)`

Static Public Attributes

- `static const std::string VERSIONTOKENS`

Friends

- `std::ostream & operator<< (std::ostream &out, const Software &sw)`

5.251.1 Detailed Description

Used to represent software (names and version) and comparison.

The `Software` class is used to represent the name of a piece of software internally. Generally software are identified by a name and possibly a version number. Some software can also be categorized by type or family (compilers, operating system, etc.). A software object can be compared to other software objects using the comparison operators contained in this class. The basic usage of this class is to test if some specified software requirement (`SoftwareRequirement`) are fulfilled, by using the comparability of the class.

Internally the `Software` object is represented by a family and name identifier, and the software version is tokenized at the characters defined in `VERSIONTOKENS`, and stored as a list of tokens.

5.251.2 Member Typedef Documentation

5.251.2.1 `typedef bool(Software::* Arc::Software::ComparisonOperator)(const Software &) const`

Definition of a comparison operator method pointer.

This `typedef` defines a comparison operator method pointer.

See also

`operator==,`
`operator!=,`
`operator>,`
`operator<,`
`operator>=,`
`operator<=,`
`ComparisonOperatorEnum.`

5.251.3 Member Enumeration Documentation

5.251.3.1 `enum Arc::Software::ComparisonOperatorEnum`

Comparison operator enum.

The `ComparisonOperatorEnum` enumeration is a 1-1 correspondance between the defined comparison method operators (`Software::ComparisonOperator`), and can be used in circumstances where method pointers are not supported.

Enumerator:

`NOTEQUAL` see `operator!=`
`EQUAL` see `operator==`
`GREATERTHAN` see `operator>`
`LESSTHAN` see `operator<`
`GREATERTHANOREQUAL` see `operator>=`
`LESSTHANOREQUAL` see `operator<=`

5.251.4 Constructor & Destructor Documentation

5.251.4.1 `Arc::Software::Software() [inline]`

Dummy constructor.

This constructor creates a empty object.

5.251.4.2 Arc::Software::Software (const std::string & name_version)

Create a [Software](#) object.

Create a [Software](#) object from a single string composed of a name and a version part. The created object will contain a empty family part. The name and version part of the string will be split at the first occurrence of a dash (-) which is followed by a digit (0-9). If the string does not contain such a pattern, the passed string will be taken to be the name and version will be empty.

Parameters

<i>name_version</i>	should be a string composed of the name and version of the software to represent.
---------------------	---

5.251.4.3 Arc::Software::Software (const std::string & name, const std::string & version)

Create a [Software](#) object.

Create a [Software](#) object with the specified name and version. The family part will be left empty.

Parameters

<i>name</i>	the software name to represent.
<i>version</i>	the software version to represent.

5.251.4.4 Arc::Software::Software (const std::string & family, const std::string & name, const std::string & version)

Create a [Software](#) object.

Create a [Software](#) object with the specified family, name and version.

Parameters

<i>family</i>	the software family to represent.
<i>name</i>	the software name to represent.
<i>version</i>	the software version to represent.

5.251.5 Member Function Documentation

5.251.5.1 static ComparisonOperator Arc::Software::convert (const ComparisonOperatorEnum & co) [static]

Convert a [ComparisonOperatorEnum](#) value to a comparison method pointer.

The passed [ComparisonOperatorEnum](#) will be converted to a comparison method pointer

defined by the [Software::ComparisonOperator](#) typedef.

This static method is not defined in language bindings created with Swig, since method pointers are not supported by Swig.

Parameters

<i>co</i>	a ComparisonOperatorEnum value.
-----------	---

Returns

A method pointer to a comparison method is returned.

5.251.5.2 bool Arc::Software::empty () const [inline]

Indicates whether the object is empty.

Returns

`true` if the name of this object is empty, otherwise `false`.

5.251.5.3 const std::string& Arc::Software::getFamily () const [inline]

Get family.

Returns

The family the represented software belongs to is returned.

5.251.5.4 const std::string& Arc::Software::getName () const [inline]

Get name.

Returns

The name of the represented software is returned.

5.251.5.5 const std::string& Arc::Software::getVersion () const [inline]

Get version.

Returns

The version of the represented software is returned.

5.251.5.6 Arc::Software::operator std::string(void) const [inline]

Cast to string.

This casting operator behaves exactly as [operator\(\)\(\)](#) does. The cast is used like `(std::string) <software-object>`.

See also

[operator\(\)\(\)](#).

References [operator\(\)\(\)](#).

5.251.5.7 bool Arc::Software::operator!= (const Software & sw) const [inline]

Inequality operator.

The behaviour of the inequality operator is just opposite that of the equality operator [\(operator==\(\)\)](#).

Parameters

<code>sw</code>	is the RHS Software object.
-----------------	---

Returns

`true` when the two objects are unequal, otherwise `false`.

References [operator==\(\)](#).

5.251.5.8 std::string Arc::Software::operator()() const

Get string representation.

Returns the string representation of this object, which is 'family'-'name'-'version'.

Returns

The string representation of this object is returned.

See also

[operator std::string\(\)](#).

Referenced by [operator std::string\(\)](#).

5.251.5.9 bool Arc::Software::operator< (const Software & sw) const [inline]

Less-than operator.

The behaviour of this less-than operator is equivalent to the greater-than operator [\(operator>\(\)\)](#) with the LHS and RHS swapped.

Parameters

<code>sw</code>	is the RHS object.
-----------------	--------------------

Returns

`true` if the LHS is less than the RHS, otherwise `false`.

See also

[operator>\(\)](#).

5.251.5.10 `bool Arc::Software::operator<=(const Software & sw) const` [inline]

Less-than or equal operator.

The LHS object is greater than or equal to the RHS object if the LHS equal the RHS ([operator==\(\)](#)) or if the LHS is greater than the RHS ([operator>\(\)](#)).

Parameters

<code>sw</code>	is the RHS object.
-----------------	--------------------

Returns

`true` if the LHS is less than or equal the RHS, otherwise `false`.

See also

[operator==\(\)](#),
[operator<\(\)](#).

5.251.5.11 `bool Arc::Software::operator==(const Software & sw) const` [inline]

Equality operator.

Two [Software](#) objects are equal only if they are of the same family, have the same name and is of same version. This operator can also be represented by the [Software::EQUAL ComparisonOperatorEnum](#) value.

Parameters

<code>sw</code>	is the RHS Software object.
-----------------	---

Returns

`true` when the two objects equals, otherwise `false`.

Referenced by [operator!=\(\)](#).

5.251.5.12 bool Arc::Software::operator> (const Software & sw) const

Greater-than operator.

For the LHS object to be greater than the RHS object they must first share the same family and name. If the version of the LHS is empty or the LHS and RHS versions equal then LHS is not greater than RHS. If the LHS version is not empty while the RHS is then LHS is greater than RHS. If both versions are non empty and not equal then, the first version token of each object is compared and if they are identical, the two next version tokens will be compared. If not identical, the two tokens will be parsed as integers, and if parsing fails the LHS is not greater than the RHS. If parsing succeeds and the integers equals, the two next tokens will be compared, otherwise the comparison is resolved by the integer comparison.

If the LHS contains more version tokens than the RHS, and the comparison have not been resolved at the point of equal number of tokens, then if the additional tokens contains a token which cannot be parsed to a integer the LHS is not greater than the RHS. If the parsed integer is not 0 then the LHS is greater than the RHS. If the rest of the additional tokens are 0, the LHS is not greater than the RHS.

If the RHS contains more version tokens than the LHS and comparison have not been resolved at the point of equal number of tokens, or simply if comparison have not been resolved at the point of equal number of tokens, then the LHS is not greater than the RHS.

Parameters

<code>sw</code>	is the RHS object.
-----------------	--------------------

Returns

`true` if the LHS is greater than the RHS, otherwise `false`.

5.251.5.13 bool Arc::Software::operator>= (const Software & sw) const [inline]

Greater-than or equal operator.

The LHS object is greater than or equal to the RHS object if the LHS equal the RHS ([operator==\(\)](#)) or if the LHS is greater than the RHS ([operator>\(\)](#)).

Parameters

<code>sw</code>	is the RHS object.
-----------------	--------------------

Returns

`true` if the LHS is created than or equal the RHS, otherwise `false`.

See also

[operator==\(\)](#),
[operator>\(\)](#).

5.251.5.14 static std::string Arc::Software::toString (ComparisonOperator *co*) [static]

Convert [Software::ComparisonOperator](#) to a string.

This method is not available in language bindings created by Swig, since method pointers are not supported by Swig.

Parameters

<i>co</i>	is a Software::ComparisonOperator .
-----------	---

Returns

The string representation of the passed [Software::ComparisonOperator](#) is returned.

5.251.6 Friends And Related Function Documentation

5.251.6.1 std::ostream& operator<< (std::ostream & *out*, const Software & *sw*) [friend]

Write [Software](#) string representation to a std::ostream.

Write the string representation of a [Software](#) object to a std::ostream.

Parameters

<i>out</i>	is a std::ostream to write the string representation of the Software object to.
<i>sw</i>	is the Software object to write to the std::ostream.

Returns

The passed std::ostream *out* is returned.

5.251.7 Field Documentation

5.251.7.1 const std::string Arc::Software::VERSIONTOKENS [static]

Tokens used to split version string.

This string constant specifies which tokens will be used to split the version string.

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

- Software.h

5.252 Arc::SoftwareRequirement Class Reference

Class used to express and resolve version requirements on software.

```
#include <Software.h>
```

Public Member Functions

- `SoftwareRequirement ()`
- `SoftwareRequirement (const Software &sw, Software::ComparisonOperator swComOp=&Software::operator==)`
- `SoftwareRequirement (const Software &sw, Software::ComparisonOperatorEnum co)`
- `SoftwareRequirement & operator= (const SoftwareRequirement &sr)`
- `SoftwareRequirement (const SoftwareRequirement &sr)`
- `void add (const Software &sw, Software::ComparisonOperator swComOp=&Software::operator==)`
- `void add (const Software &sw, Software::ComparisonOperatorEnum co)`
- `bool isSatisfied (const Software &sw) const`
- `bool isSatisfied (const std::list< Software > &swList) const`
- `bool isSatisfied (const std::list< ApplicationEnvironment > &swList) const`
- `bool selectSoftware (const Software &sw)`
- `bool selectSoftware (const std::list< Software > &swList)`
- `bool selectSoftware (const std::list< ApplicationEnvironment > &swList)`
- `bool isResolved () const`
- `bool empty () const`
- `void clear ()`
- `const std::list< Software > & getSoftwareList () const`
- `const std::list< Software::ComparisonOperator > & getComparisonOperatorList () const`

5.252.1 Detailed Description

Class used to express and resolve version requirements on software.

A requirement in this class is defined as a pair composed of a `Software` object and either a `Software::ComparisonOperator` method pointer or equally a `Software::ComparisonOperatorEnum` enum value. A `SoftwareRequirement` object can contain multiple of such requirements, and then it can be specified if all these requirements should be satisfied, or if it is enough to satisfy only one of them. The requirements can be satisfied by a single `Software` object or a list of either `Software` or `ApplicationEnvironment` objects, by using the method `isSatisfied()`. This class also contains a number of methods (`selectSoftware()`) to select `Software` objects which are satisfying the requirements, and in this way resolving requirements.

5.252.2 Constructor & Destructor Documentation

5.252.2.1 Arc::SoftwareRequirement::SoftwareRequirement() [inline]

Create an empty `SoftwareRequirement` object.

The created `SoftwareRequirement` object will contain no requirements.

5.252.2.2 Arc::SoftwareRequirement::SoftwareRequirement (const Software & *sw*, Software::ComparisonOperator *swComOp* = &Software::operator==)

Create a [SoftwareRequirement](#) object.

The created [SoftwareRequirement](#) object will contain one requirement specified by the [Software](#) object *sw*, and the [Software::ComparisonOperator](#) *swComOp*.

This constructor is not available in language bindings created by Swig, since method pointers are not supported by Swig, see [SoftwareRequirement\(const Software&, Software::ComparisonOperatorEnum\)](#) instead.

Parameters

<i>sw</i>	is the Software object of the requirement to add.
<i>swComOp</i>	is the Software::ComparisonOperator of the requirement to add.

5.252.2.3 Arc::SoftwareRequirement::SoftwareRequirement (const Software & *sw*, Software::ComparisonOperatorEnum *co*)

Create a [SoftwareRequirement](#) object.

The created [SoftwareRequirement](#) object will contain one requirement specified by the [Software](#) object *sw*, and the [Software::ComparisonOperatorEnum](#) *co*.

Parameters

<i>sw</i>	is the Software object of the requirement to add.
<i>co</i>	is the Software::ComparisonOperatorEnum of the requirement to add.

5.252.2.4 Arc::SoftwareRequirement::SoftwareRequirement (const SoftwareRequirement & *sr*) [inline]

Copy constructor.

Create a [SoftwareRequirement](#) object from another [SoftwareRequirement](#) object.

Parameters

<i>sr</i>	is the SoftwareRequirement object to make a copy of.
-----------	--

5.252.3 Member Function Documentation

5.252.3.1 void Arc::SoftwareRequirement::add (const Software & *sw*, Software::ComparisonOperator *swComOp* = &Software::operator==)

Add a [Software](#) object a corresponding comparison operator to this object.

Adds software name and version to list of requirements and associates the comparison operator with it (equality by default).

This method is not available in language bindings created by Swig, since method pointers are not supported by Swig, see [add\(const Software&, Software::ComparisonOperatorEnum\)](#) instead.

Parameters

<i>sw</i>	is the Software object to add as part of a requirement.
<i>swComOp</i>	is the Software::ComparisonOperator method pointer to add as part of a requirement, the default operator will be Software::operator==() .

5.252.3.2 void Arc::SoftwareRequirement::add (const Software & *sw*, Software::ComparisonOperatorEnum *co*)

Add a [Software](#) object a corresponding comparion operator to this object.

Adds software name and version to list of requirements and associates the comparison operator with it (equality by default).

Parameters

<i>sw</i>	is the Software object to add as part of a requirement.
<i>co</i>	is the Software::ComparisonOperator value to add as part of a requirement, the default enum will be Software::EQUAL .

5.252.3.3 void Arc::SoftwareRequirement::clear () [inline]

Clear the object.

The requirements in this object will be cleared when invoking this method.

5.252.3.4 bool Arc::SoftwareRequirement::empty () const [inline]

Test if the object is empty.

Returns

`true` if this object do no contain any requirements, otherwise `false`.

5.252.3.5 const std::list<Software::ComparisonOperator>& Arc::SoftwareRequirement::getComparisonOperatorList () const [inline]

Get list of comparison operators.

Returns

The list of internally stored comparison operators is returned.

See also

[Software::ComparisonOperator](#),
[getSoftwareList](#).

5.252.3.6 `const std::list<Software>& Arc::SoftwareRequirement::getSoftwareList () const [inline]`

Get list of [Software](#) objects.

Returns

The list of internally stored [Software](#) objects is returned.

See also

[Software](#),
[getComparisonOperatorList](#).

5.252.3.7 `bool Arc::SoftwareRequirement::isResolved () const`

Indicates whether requirements have been resolved or not.

If specified that only one requirement has to be satisfied, then for this object to be resolved it can only contain one requirement and it has use the equal operator ([Software::operator==](#)).

If specified that all requirements has to be satisfied, then for this object to be resolved each requirement must have a [Software](#) object with a unique family/name composition, i.e. no other requirements have a [Software](#) object with the same family/name composition, and each requirement must use the equal operator ([Software::operator==](#)).

If this object has been resolved then `true` is returned when invoking this method, otherwise `false` is returned.

Returns

`true` if this object have been resolved, otherwise `false`.

5.252.3.8 `bool Arc::SoftwareRequirement::isSatisfied (const std::list<ApplicationEnvironment > & swList) const`

Test if requirements are satisfied.

This method behaves in exactly the same way as the [isSatisfied\(const Software&\)](#) const method does.

Parameters

<code>swList</code>	is the list of ApplicationEnvironment objects which should be used to try satisfy the requirements.
---------------------	---

Returns

`true` if requirements are satisfied, otherwise `false`.

See also

`isSatisfied(const Software&)` const,
`isSatisfied(const std::list<Software>&)` const,
`selectSoftware(const std::list<ApplicationEnvironment>&)`,
`isResolved()` const.

5.252.3.9 `bool Arc::SoftwareRequirement::isSatisfied (const std::list< Software > & swList) const [inline]`

Test if requirements are satisfied.

Returns `true` if stored requirements are satisfied by software specified in `swList`, otherwise `false` is returned.

Note that if all requirements must be satisfied and multiple requirements exist having identical name and family all these requirements should be satisfied by a single `Software` object.

Parameters

<code>swList</code>	is the list of <code>Software</code> objects which should be used to try satisfy the requirements.
---------------------	--

Returns

`true` if requirements are satisfied, otherwise `false`.

See also

`isSatisfied(const Software&)` const,
`isSatisfied(const std::list<ApplicationEnvironment>&)` const,
`selectSoftware(const std::list<Software>&)`,
`isResolved()` const.

5.252.3.10 `bool Arc::SoftwareRequirement::isSatisfied (const Software & sw) const [inline]`

Test if requirements are satisfied.

Returns `true` if the requirements are satisfied by the specified `Software` `sw`, otherwise `false` is returned.

Parameters

<code>sw</code>	is the <code>Software</code> which should satisfy the requirements.
-----------------	---

Returns

true if requirements are satisfied, otherwise false.

See also

[isSatisfied\(const std::list<Software>&\) const](#),
[isSatisfied\(const std::list<ApplicationEnvironment>&\) const](#),
[selectSoftware\(const Software&\)](#),
[isResolved\(\) const](#).

References [isSatisfied\(\)](#).

Referenced by [isSatisfied\(\)](#).

5.252.3.11 SoftwareRequirement& Arc::SoftwareRequirement::operator= (const SoftwareRequirement & sr)

Assignment operator.

Set this object equal to that of the passed [SoftwareRequirement](#) object *sr*.

Parameters

<i>sr</i>	is the SoftwareRequirement object to set object equal to.
-----------	---

5.252.3.12 bool Arc::SoftwareRequirement::selectSoftware (const std::list< Software > & swList)

Select software.

If the passed list of [Software](#) objects *swList* do not satisfy the requirements *false* is returned and this object is not modified. If however the list of [Software](#) objects *swList* do satisfy the requirements *true* is returned and the [Software](#) objects satisfying the requirements will replace these with the equality operator ([Software::operator==](#)) used as the comparator for the new requirements.

Note that if all requirements must be satisfied and multiple requirements exist having identical name and family all these requirements should be satisfied by a single [Software](#) object and it will replace all these requirements.

Parameters

<i>swList</i>	is a list of Software objects used to satisfy requirements.
---------------	---

Returns

true if requirements are satisfied, otherwise false.

See also

[selectSoftware\(const Software&\)](#),

```
selectSoftware(const std::list<ApplicationEnvironment>&),
isSatisfied(const std::list<Software>&) const,
isResolved() const.
```

5.252.3.13 bool Arc::SoftwareRequirement::selectSoftware (const Software & sw) [inline]

Select software.

If the passed [Software](#) *sw* do not satisfy the requirements `false` is returned and this object is not modified. If however the [Software](#) object *sw* do satisfy the requirements `true` is returned and the requirements are set to equal the *sw* [Software](#) object.

Parameters

<code>sw</code>	is the Software object used to satisfy requirements.
-----------------	--

Returns

`true` if requirements are satisfied, otherwise `false`.

See also

```
selectSoftware(const std::list<Software>&),
selectSoftware(const std::list<ApplicationEnvironment>&),
isSatisfied(const Software&) const,
isResolved() const.
```

References [selectSoftware\(\)](#).

Referenced by [selectSoftware\(\)](#).

5.252.3.14 bool Arc::SoftwareRequirement::selectSoftware (const std::list<ApplicationEnvironment > & swList)

Select software.

This method behaves exactly as the [selectSoftware\(const std::list<Software>&\)](#) method does.

Parameters

<code>swList</code>	is a list of ApplicationEnvironment objects used to satisfy requirements.
---------------------	---

Returns

`true` if requirements are satisfied, otherwise `false`.

See also

```
selectSoftware(const Software&),
selectSoftware(const std::list<Software>&),
```

```
isSatisfied(const std::list<ApplicationEnvironment>&) const,
isResolved() const.
```

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

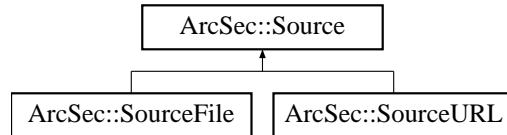
- Software.h

5.253 ArcSec::Source Class Reference

Acquires and parses XML document from specified source.

```
#include <Source.h>
```

Inheritance diagram for ArcSec::Source:



Public Member Functions

- `Source (const Source &s)`
- `Source (Arc::XMLNode &xml)`
- `Source (std::istream &stream)`
- `Source (Arc::URL &url)`
- `Source (const std::string &str)`
- `Arc::XMLNode Get (void) const`
- `operator bool (void)`

5.253.1 Detailed Description

Acquires and parses XML document from specified source.

This class is to be used to provide easy way to specify different sources for XML Authorization Policies and Requests.

5.253.2 Constructor & Destructor Documentation

5.253.2.1 ArcSec::Source::Source (const Source & s) [inline]

Copy constructor.

Use this constructor only for temporary objects. Parsed XML document is still owned by copied source and hence lifetime of create object should not exceed that of copied one.

5.253.2.2 ArcSec::Source::Source (Arc::URL & url)

Fetch XML document from specified url and parse it.

This constructor is not implemented yet.

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

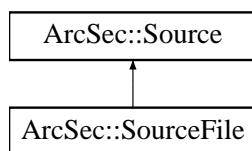
- Source.h

5.254 ArcSec::SourceFile Class Reference

Convenience class for obtaining XML document from file.

```
#include <Source.h>
```

Inheritance diagram for ArcSec::SourceFile:



Public Member Functions

- `SourceFile (const SourceFile &s)`
- `SourceFile (const char *name)`
- `SourceFile (const std::string &name)`

5.254.1 Detailed Description

Convenience class for obtaining XML document from file.

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

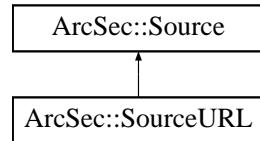
- Source.h

5.255 ArcSec::SourceURL Class Reference

Convenience class for obtaining XML document from remote URL.

```
#include <Source.h>
```

Inheritance diagram for ArcSec::SourceURL:



Public Member Functions

- [SourceURL \(const SourceURL &s\)](#)
- [SourceURL \(const char *url\)](#)
- [SourceURL \(const std::string &url\)](#)

5.255.1 Detailed Description

Convenience class for obtaining XML document from remote URL.

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

- [Source.h](#)

5.256 DataStaging::DataDeliveryComm::Status Struct Reference

Plain C struct to pass information from executing process back to main thread.

```
#include <DataDeliveryComm.h>
```

Data Fields

- [CommStatusType commstatus](#)
- [time_t timestamp](#)
- [DTRStatus::DTRStatusType status](#)
- [DTRErrorStatus::DTRErrorStatusType error](#)
- [DTRErrorStatus::DTRErrorLocation error_location](#)
- [char error_desc \[256\]](#)
- [unsigned int streams](#)
- [unsigned long long int transferred](#)
- [unsigned long long int offset](#)
- [unsigned long long int size](#)
- [unsigned int speed](#)
- [char checksum \[128\]](#)

5.256.1 Detailed Description

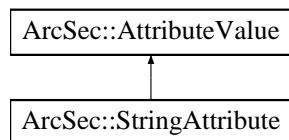
Plain C struct to pass information from executing process back to main thread.

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

- DataDeliveryComm.h

5.257 ArcSec::StringAttribute Class Reference

Inheritance diagram for ArcSec::StringAttribute:



Public Member Functions

- virtual bool [equal](#) ([AttributeValue](#) *other, bool check_id=true)
- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

5.257.1 Member Function Documentation

5.257.1.1 virtual std::string ArcSec::StringAttribute::encode () [inline, virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.257.1.2 virtual bool ArcSec::StringAttribute::equal ([AttributeValue](#) * value, bool check_id = true) [virtual]

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.257.1.3 virtual std::string ArcSec::StringAttribute::getId () [inline, virtual]

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.257.1.4 virtual std::string ArcSec::StringAttribute::getType() [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

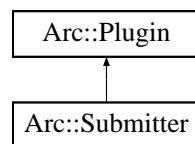
- [StringAttribute.h](#)

5.258 Arc::Submitter Class Reference

Base class for the Submitters.

```
#include <Submitter.h>
```

Inheritance diagram for Arc::Submitter:



Public Member Functions

- bool [GetTestJob](#) (const int &testid, [JobDescription](#) &jobdescription)
- [URL Submit](#) (const [JobDescription](#) &jobdesc, const [ExecutionTarget](#) &et)
- [URL Migrate](#) (const [URL](#) &jobid, const [JobDescription](#) &jobdesc, const [ExecutionTarget](#) &et, bool forcemigration)

Protected Attributes

- const [ExecutionTarget](#) * target

5.258.1 Detailed Description

Base class for the Submitters.

[Submitter](#) is the base class for Grid middleware specialized [Submitter](#) objects. The class submits job(s) to the computing resource it represents and uploads (needed by the job) local input files.

5.258.2 Member Function Documentation

5.258.2.1 bool Arc::Submitter::GetTestJob (const int & *testid*, JobDescription & *jobdescription*)

This virtual method can be overridden by plugins which should be capable of getting test job descriptions for the specified flavour. This method should return with the [JobDescription](#) or NULL if there is no test description defined with the requested id.

5.258.2.2 URL Arc::Submitter::Migrate (const URL & *jobid*, const JobDescription & *jobdesc*, const ExecutionTarget & *et*, bool *forcemigration*)

This virtual method should be overridden by plugins which should be capable of migrating jobs. The active job which should be migrated is pointed to by the [URL](#) jobid, and is represented by the [JobDescription](#) jobdesc. The forcemigration boolean specifies if the migration should succeed if the active job cannot be terminated. The protected method AddJob can be used to save job information. This method should return the [URL](#) of the migrated job. In case migration fails an empty [URL](#) should be returned.

5.258.2.3 URL Arc::Submitter::Submit (const JobDescription & *jobdesc*, const ExecutionTarget & *et*)

This virtual method should be overridden by plugins which should be capable of submitting jobs, defined in the [JobDescription](#) jobdesc, to the [ExecutionTarget](#) et. The protected convenience method AddJob can be used to save job information. This method should return the [URL](#) of the submitted job. In case submission fails an empty [URL](#) should be returned.

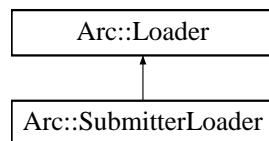
The documentation for this class was generated from the following file:

- Submitter.h

5.259 Arc::SubmitterLoader Class Reference

```
#include <Submitter.h>
```

Inheritance diagram for Arc::SubmitterLoader:



Public Member Functions

- [SubmitterLoader \(\)](#)
- [~SubmitterLoader \(\)](#)

- `Submitter * load (const std::string &name, const UserConfig &usercfg)`
- `const std::list< Submitter * > & GetSubmitters () const`

5.259.1 Detailed Description

Class responsible for loading `Submitter` plugins. The `Submitter` objects returned by a `SubmitterLoader` must not be used after the `SubmitterLoader` goes out of scope.

5.259.2 Constructor & Destructor Documentation

5.259.2.1 Arc::SubmitterLoader::SubmitterLoader ()

Constructor Creates a new `SubmitterLoader`.

5.259.2.2 Arc::SubmitterLoader::~SubmitterLoader ()

Destructor Calling the destructor destroys all Submitters loaded by the `SubmitterLoader` instance.

5.259.3 Member Function Documentation

5.259.3.1 const std::list<Submitter*>& Arc::SubmitterLoader::GetSubmitters () const [inline]

Retrieve the list of loaded Submitters.

Returns

A reference to the list of Submitters.

5.259.3.2 Submitter* Arc::SubmitterLoader::load (const std::string & name, const UserConfig & usercfg)

Load a new `Submitter`

Parameters

<code>name</code>	The name of the <code>Submitter</code> to load.
<code>usercfg</code>	The <code>UserConfig</code> object for the new <code>Submitter</code> .

Returns

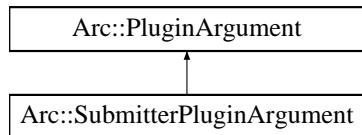
A pointer to the new `Submitter` (NULL on error).

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

- `Submitter.h`

5.260 Arc::SubmitterPluginArgument Class Reference

Inheritance diagram for Arc::SubmitterPluginArgument:



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

- Submitter.h

5.261 Arc::TargetGenerator Class Reference

Target generation class

```
#include <TargetGenerator.h>
```

Public Member Functions

- [TargetGenerator](#) (const [UserConfig](#) &usercfg, unsigned int startRetrieval=0)
- void [GetTargets](#) (int targetType, int detailLevel)
- void [RetrieveExecutionTargets](#) ()
- void [RetrieveJobs](#) ()
- const std::list< [ExecutionTarget](#) > & [GetExecutionTargets](#) () const
- std::list< [ExecutionTarget](#) > & [ModifyFoundTargets](#) ()
- const std::list< [ExecutionTarget](#) > & [FoundTargets](#) () const
- const std::list< [XMLNode](#) * > & [FoundJobs](#) () const
- const std::list< [Job](#) > & [GetJobs](#) () const
- bool [AddService](#) (const std::string &Flavour, const [URL](#) &url)
- bool [AddIndexServer](#) (const std::string &Flavour, const [URL](#) &url)
- void [AddTarget](#) (const [ExecutionTarget](#) &target)
- void [AddJob](#) (const [XMLNode](#) &job)
- void [AddJob](#) (const [Job](#) &job)
- void [PrintTargetInfo](#) (bool longlist) const
- void [SaveTargetInfoToStream](#) (std::ostream &out, bool longlist) const
- [SimpleCounter](#) & [ServiceCounter](#) (void)

5.261.1 Detailed Description

Target generation class

The [TargetGenerator](#) class is the umbrella class for resource discovery and information retrieval (index servers and execution services). It can also be used to discover user Grid jobs and detailed information. The [TargetGenerator](#) loads [TargetRetriever](#) plugins (which implements the actual information retrieval) from [URL](#) objects found in the [UserConfig](#) object passed to its constructor using the custom [TargetRetrieverLoader](#).

5.261.2 Constructor & Destructor Documentation

5.261.2.1 Arc::TargetGenerator::TargetGenerator (const UserConfig & *usercfg*, unsigned int *startRetrieval* = 0)

Create a [TargetGenerator](#) object.

Default constructor to create a TargetGenerator. The constructor reads the computing and index service [URL](#) objects from the passed [UserConfig](#) object using the [UserConfig::GetSelectedServices](#) method. From each [URL](#) a matching specialized [TargetRetriever](#) plugin is loaded using the [TargetRetrieverLoader](#). If the second parameter, *startRetrieval*, is specified, and matches bitwise either a value of 1, 2 or both, retrieval of execution services, jobs or both will be initiated.

Parameters

<i>usercfg</i>	is a reference to a UserConfig object from which endpoints to execution and/or index services will be used. The object also hold information about user credentials.
<i>startRetrieval</i>	specifies whether retrieval should be started directly. It will be parsed bitwise. A value of 1 will start execution service retrieval (RetrieveExecutionTargets), 2 jobs (RetrieveJobs), and 3 both, while 0 will not start retrieval at all. If not specified, default is 0.

5.261.3 Member Function Documentation

5.261.3.1 bool Arc::TargetGenerator::AddIndexServer (const std::string & *Flavour*, const URL & *url*)

Add a new index server to the foundIndexServers list.

Method to add a new index server to the list of foundIndexServers in a thread secure way. Compares the argument [URL](#) against the servers returned by [UserConfig::GetRejectedServices](#) and only allows to add the service if not specifically rejected.

Parameters

<i>flavour</i>	The flavour if the the index server.
<i>url</i>	URL pointing to the index server.

5.261.3.2 void Arc::TargetGenerator::AddJob (const XMLNode & job)

DEPRECATED: Add a new [Job](#) to this object.

This method is DEPRECATED, use the [AddJob\(const Job&\)](#) method instead. Method to add a new [Job](#) (usually discovered by a [TargetRetriever](#)) to the internal list of jobs in a thread secure way.

Parameters

<i>job</i>	XMLNode describing the job.
------------	---

5.261.3.3 void Arc::TargetGenerator::AddJob (const Job & job)

Add a new [Job](#) to this object.

Method to add a new [Job](#) (usually discovered by a [TargetRetriever](#)) to the internal list of jobs in a thread secure way.

Parameters

<i>job</i>	Job describing the job.
------------	---

See also

[AddJob\(const Job&\)](#)

5.261.3.4 bool Arc::TargetGenerator::AddService (const std::string & Flavour, const URL & url)

Add a new computing service to the foundServices list.

Method to add a new service to the list of foundServices in a thread secure way. Compares the argument [URL](#) against the services returned by [UserConfig::GetRejectedServices](#) and only allows to add the service if not specifically rejected.

Parameters

<i>flavour</i>	The flavour if the the computing service.
<i>url</i>	URL pointing to the information system of the computing service.

5.261.3.5 void Arc::TargetGenerator::AddTarget (const ExecutionTarget & target)

Add a new [ExecutionTarget](#) to the foundTargets list.

Method to add a new [ExecutionTarget](#) (usually discovered by a [TargetRetriever](#)) to the list of foundTargets in a thread secure way.

Parameters

<i>target</i>	ExecutionTarget to be added.
---------------	--

5.261.3.6 `const std::list<XMLNode*>& Arc::TargetGenerator::FoundJobs() const`

DEPRECATED: Return jobs found by GetTargets.

This method is DEPRECATED, use the GetFoundJobs method instead. Method to return the list of jobs found by a call to the GetJobs method.

Returns

A list of jobs in XML format is returned.

5.261.3.7 `const std::list<ExecutionTarget>& Arc::TargetGenerator::FoundTargets() const [inline]`

DEPRECATED: Return targets found by GetTargets.

This method is DEPRECATED, use the [FoundTargets\(\)](#) instead. Method to return the list of [ExecutionTarget](#) objects (currently only supported Target type) found by the GetTarget method.

5.261.3.8 `const std::list<ExecutionTarget>& Arc::TargetGenerator::GetExecutionTargets() const [inline]`

Return targets fetched by RetrieveExecutionTargets method.

Method to return a const list of [ExecutionTarget](#) objects retrieved by the RetrieveExecutionTargets method.

See also

[RetrieveExecutionTargets](#)
[GetExecutionTargets](#)

5.261.3.9 `const std::list<Job>& Arc::TargetGenerator::GetJobs() const [inline]`

Return jobs retrieved by RetrieveJobs method.

Method to return the list of jobs found by a call to the GetJobs method.

Returns

A list of the discovered jobs as [Job](#) objects is returned

See also

[RetrieveJobs](#)

5.261.3.10 void Arc::TargetGenerator::GetTargets (int *targetType*, int *detailLevel*)

DEPRECATED: Find available targets.

This method is DEPRECATED, use the [RetrieveExecutionTargets\(\)](#) or [RetrieveJobs\(\)](#) method instead. Method to prepare a list of chosen Targets with a specified detail level. Current implementation supports finding computing elements ([ExecutionTarget](#)) with full detail level and jobs with limited detail level.

Parameters

<i>targetType</i>	0 = ExecutionTarget , 1 = Grid jobs
<i>detailLevel</i>	

See also

[RetrieveExecutionTargets\(\)](#)
[RetrieveJobs\(\)](#)

5.261.3.11 std::list<[ExecutionTarget](#)>& Arc::TargetGenerator::ModifyFoundTargets ()

DEPRECATED: Return targets found by GetTargets.

This method is DEPRECATED, use the [FoundTargets\(\)](#) instead. Method to return the list of [ExecutionTarget](#) objects (currently only supported Target type) found by the GetTarget method.

5.261.3.12 void Arc::TargetGenerator::PrintTargetInfo (bool *longlist*) const

DEPRECATED: Prints target information.

This method is DEPRECATED, use the [SaveTargetInfoToStream](#) method instead. Method to print information of the found targets to std::cout.

Parameters

<i>longlist</i>	false for minimal information, true for detailed information
-----------------	--

See also

[SaveTargetInfoToStream](#)

5.261.3.13 void Arc::TargetGenerator::RetrieveExecutionTargets ()

Retrieve available execution services.

The endpoints specified in the [UserConfig](#) object passed to this object will be used to retrieve information about execution services ([ExecutionTarget](#) objects). The discovery and information retrieval of targets is carried out in parallel threads to speed up the process. If a endpoint is a index service each execution service registered will be queried.

See also

[RetrieveJobs](#)
[GetExecutionTargets](#)

5.261.3.14 void Arc::TargetGenerator::RetrieveJobs()

Retrieve job information from execution services.

The endpoints specified in the [UserConfig](#) object passed to this object will be used to retrieve job information from these endpoints. Only jobs owned by the user which is identified by the credentials specified in the passed [UserConfig](#) object will be considered (exception being services which has no user authentication). If a endpoint is a index service, each execution service registered will be queried, and searched for job information.

See also

[RetrieveExecutionTargets](#)

5.261.3.15 void Arc::TargetGenerator::SaveTargetInfoToStream(std::ostream & out, bool longlist) const

Prints target information.

Method to print information of the found targets to std::cout.

Parameters

<i>out</i>	is a std::ostream object which to direct target information to.
<i>longlist</i>	false for minimal information, true for detailed information

5.261.3.16 SimpleCounter& Arc::TargetGenerator::ServiceCounter(void)

Returns reference to worker counter.

This method returns reference to counter which keeps amount of started worker threads communicating with services asynchronously. The counter must be incremented for every thread started and decremented when thread exits. Main thread will then wait till counters drops to zero.

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

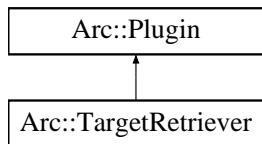
- TargetGenerator.h

5.262 Arc::TargetRetriever Class Reference

TargetRetriever base class

```
#include <TargetRetriever.h>
```

Inheritance diagram for Arc::TargetRetriever:



Public Member Functions

- virtual void [GetTargets](#) ([TargetGenerator](#) &mom, int targetType, int detailLevel)=0
- virtual void [GetExecutionTargets](#) ([TargetGenerator](#) &mom)=0
- virtual void [GetJobs](#) ([TargetGenerator](#) &mom)=0

Protected Member Functions

- [TargetRetriever](#) (const [UserConfig](#) &usercfg, const [URL](#) &url, ServiceType st, const std::string &flavour)

5.262.1 Detailed Description

TargetRetriever base class

The [TargetRetriever](#) class is a pure virtual base class to be used for grid flavour specializations. It is designed to work in conjunction with the [TargetGenerator](#).

5.262.2 Constructor & Destructor Documentation

5.262.2.1 Arc::TargetRetriever::TargetRetriever (const UserConfig & usercfg, const URL & url, ServiceType st, const std::string & flavour) [protected]

[TargetRetriever](#) constructor.

Default constructor to create a TargeGenerator. The constructor reads the computing and index service [URL](#) objects from the

Parameters

<i>usercfg</i>
<i>url</i>
<i>st</i>
<i>flavour</i>

5.262.3 Member Function Documentation

5.262.3.1 `virtual void Arc::TargetRetriever::GetExecutionTargets (TargetGenerator & mom) [pure virtual]`

Method for collecting targets.

Pure virtual method for collecting targets. Implementation depends on the Grid middleware in question and is thus left to the specialized class.

Parameters

<code>mom</code>	is the reference to the TargetGenerator which has loaded the TargetRetriever
<code>detailLevel</code>	is the required level of details (1 = All details, 2 = Limited details)

5.262.3.2 `virtual void Arc::TargetRetriever::GetJobs (TargetGenerator & mom) [pure virtual]`

Method for collecting targets.

Pure virtual method for collecting targets. Implementation depends on the Grid middleware in question and is thus left to the specialized class.

Parameters

<code>mom</code>	is the reference to the TargetGenerator which has loaded the TargetRetriever
<code>detailLevel</code>	is the required level of details (1 = All details, 2 = Limited details)

5.262.3.3 `virtual void Arc::TargetRetriever::GetTargets (TargetGenerator & mom, int targetType, int detailLevel) [pure virtual]`

DEPRECATED: Method for collecting targets.

This method is DEPRECATED, the GetExecutionTargets and GetJobs methods replaces it.

Pure virtual method for collecting targets. Implementation depends on the Grid middleware in question and is thus left to the specialized class.

Parameters

<code>mom</code>	is the reference to the TargetGenerator which has loaded the TargetRetriever
<code>targetType</code>	is the identification of targets to find (0 = ExecutionTargets, 1 = Grid Jobs)
<code>detailLevel</code>	is the required level of details (1 = All details, 2 = Limited details)

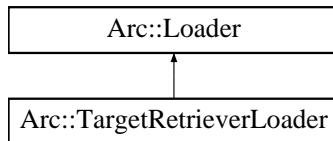
The documentation for this class was generated from the following file:

- TargetRetriever.h

5.263 Arc::TargetRetrieverLoader Class Reference

```
#include <TargetRetriever.h>
```

Inheritance diagram for Arc::TargetRetrieverLoader:



Public Member Functions

- [TargetRetrieverLoader \(\)](#)
- [~TargetRetrieverLoader \(\)](#)
- [TargetRetriever * load \(const std::string &name, const UserConfig &usercfg, const std::string &service, const ServiceType &st\)](#)
- [const std::list< TargetRetriever * > & GetTargetRetrievers \(\) const](#)

5.263.1 Detailed Description

Class responsible for loading [TargetRetriever](#) plugins. The [TargetRetriever](#) objects returned by a [TargetRetrieverLoader](#) must not be used after the [TargetRetrieverLoader](#) goes out of scope.

5.263.2 Constructor & Destructor Documentation

5.263.2.1 Arc::TargetRetrieverLoader::TargetRetrieverLoader ()

Constructor Creates a new [TargetRetrieverLoader](#).

5.263.2.2 Arc::TargetRetrieverLoader::~TargetRetrieverLoader ()

Destructor Calling the destructor destroys all TargetRetrievers loaded by the [TargetRetrieverLoader](#) instance.

5.263.3 Member Function Documentation

5.263.3.1 const std::list<TargetRetriever*>& Arc::TargetRetrieverLoader::GetTargetRetrievers () const [inline]

Retrieve the list of loaded TargetRetrievers.

Returns

A reference to the list of TargetRetrievers.

5.263.3.2 TargetRetriever* Arc::TargetRetrieverLoader::load (const std::string & name, const UserConfig & usercfg, const std::string & service, const ServiceType & st)

Load a new [TargetRetriever](#)

Parameters

<i>name</i>	The name of the TargetRetriever to load.
<i>usercfg</i>	The UserConfig object for the new TargetRetriever .
<i>service</i>	The URL used to contact the target.
<i>st</i>	specifies service type of the target.

Returns

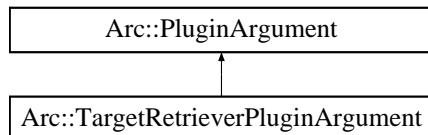
A pointer to the new [TargetRetriever](#) (NULL on error).

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

- TargetRetriever.h

5.264 Arc::TargetRetrieverPluginArgument Class Reference

Inheritance diagram for Arc::TargetRetrieverPluginArgument:



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

- TargetRetriever.h

5.265 Arc::ThreadDataItem Class Reference

Base class for per-thread object.

```
#include <Thread.h>
```

Public Member Functions

- [ThreadDataItem \(void\)](#)
- [ThreadDataItem \(std::string &key\)](#)
- [ThreadDataItem \(const std::string &key\)](#)
- void [Attach \(std::string &key\)](#)
- void [Attach \(const std::string &key\)](#)
- virtual void [Dup \(void\)](#)

Static Public Member Functions

- static [ThreadDataItem * Get \(const std::string &key\)](#)

5.265.1 Detailed Description

Base class for per-thread object.

Classes inherited from this one are attached to current thread under specified key and destroyed only when thread ends or object is replaced by another one with same key.

5.265.2 Constructor & Destructor Documentation

5.265.2.1 Arc::ThreadDataItem::ThreadDataItem (void)

Dummy constructor which does nothing. To make object usable one of Attach(...) methods must be used.

5.265.2.2 Arc::ThreadDataItem::ThreadDataItem (std::string & key)

Creates instance and attaches it to current thread under key. If supplied key is empty random one is generated and stored in key variable.

5.265.2.3 Arc::ThreadDataItem::ThreadDataItem (const std::string & key)

Creates instance and attaches it to current thread under key.

5.265.3 Member Function Documentation

5.265.3.1 void Arc::ThreadDataItem::Attach (std::string & key)

Attaches object to current thread under key. If supplied key is empty random one is generated and stored in key variable. This method must be used only if object was created using dummy constructor.

5.265.3.2 void Arc::ThreadDataItem::Attach (const std::string & key)

Attaches object to current thread under key. This method must be used only if object was created using dummy constructor.

5.265.3.3 virtual void Arc::ThreadDataItem::Dup (void) [virtual]

Creates copy of object. This method is called when new thread is created from current thread. It is called in new thread, so new object - if created - gets attached to new thread. If object is not meant to be inherited by new threads then this method should do nothing.

5.265.3.4 static ThreadDataItem* Arc::ThreadDataItem::Get (const std::string & key) [static]

Retrieves object attached to thread under key. Returns NULL if no such obejct.

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

- Thread.h

5.266 Arc::ThreadInitializer Class Reference

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

- Thread.h

5.267 Arc::ThreadRegistry Class Reference

```
#include <Thread.h>
```

Public Member Functions

- void [RegisterThread](#) (void)
- void [UnregisterThread](#) (void)
- bool [WaitOrCancel](#) (int timeout)
- bool [WaitForExit](#) (int timeout=-1)

5.267.1 Detailed Description

This class is a set of conditions, mutexes, etc. conveniently exposed to monitor running child threads and to wait till they exit. There are no protections against race conditions. So use it carefully.

5.267.2 Member Function Documentation

5.267.2.1 bool Arc::ThreadRegistry::WaitForExit (int *timeout* = -1)

Wait for registered threads to exit. Leave after timeout milliseconds if failed. Returns true if all registered threads reported their exit.

5.267.2.2 bool Arc::ThreadRegistry::WaitOrCancel (int *timeout*)

Wait for timeout milliseconds or cancel request. Returns true if cancel request received.

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

- Thread.h

5.268 Arc::Time Class Reference

A class for storing and manipulating times.

```
#include <DateTime.h>
```

Public Member Functions

- [Time \(\)](#)
- [Time \(time_t\)](#)
- [Time \(time_t time, uint32_t nanosec\)](#)
- [Time \(const std::string &\)](#)
- [Time & operator= \(time_t\)](#)
- [Time & operator= \(const Time &\)](#)
- [Time & operator= \(const char *\)](#)
- [Time & operator= \(const std::string &\)](#)
- [void SetTime \(time_t\)](#)
- [void SetTime \(time_t time, uint32_t nanosec\)](#)
- [time_t GetTime \(\) const](#)
- [operator std::string \(\) const](#)
- [std::string str \(const TimeFormat &=time_format\) const](#)
- [bool operator< \(const Time &\) const](#)
- [bool operator> \(const Time &\) const](#)
- [bool operator<= \(const Time &\) const](#)
- [bool operator>= \(const Time &\) const](#)
- [bool operator== \(const Time &\) const](#)
- [bool operator!= \(const Time &\) const](#)
- [Time operator+ \(const Period &\) const](#)
- [Time operator- \(const Period &\) const](#)
- [Period operator- \(const Time &\) const](#)

Static Public Member Functions

- static void `SetFormat` (const `TimeFormat` &)
- static `TimeFormat GetFormat()`

5.268.1 Detailed Description

A class for storing and manipulating times.

5.268.2 Constructor & Destructor Documentation

5.268.2.1 `Arc::Time::Time()`

Default constructor. The time is put equal the current time.

5.268.2.2 `Arc::Time::Time(time_t)`

Constructor that takes a `time_t` variable and stores it.

5.268.2.3 `Arc::Time::Time(time_t time, uint32_t nanosec)`

Constructor that takes a fine grained time variables and stores them.

5.268.2.4 `Arc::Time::Time(const std::string &)`

Constructor that tries to convert a string into a `time_t`.

5.268.3 Member Function Documentation

5.268.3.1 static `TimeFormat Arc::Time::GetFormat() [static]`

Gets the default format for time strings.

5.268.3.2 `time_t Arc::Time::GetTime() const`

gets the time

5.268.3.3 `Arc::Time::operator std::string() const`

Returns a string representation of the time, using the default format.

5.268.3.4 `bool Arc::Time::operator!= (const Time &) const`

Comparing two [Time](#) objects.

5.268.3.5 `Time Arc::Time::operator+ (const Period &) const`

Adding [Time](#) object with [Period](#) object.

5.268.3.6 `Time Arc::Time::operator- (const Period &) const`

Subtracting [Period](#) object from [Time](#) object.

5.268.3.7 `Period Arc::Time::operator- (const Time &) const`

Subtracting [Time](#) object from the other [Time](#) object.

5.268.3.8 `bool Arc::Time::operator< (const Time &) const`

Comparing two [Time](#) objects.

5.268.3.9 `bool Arc::Time::operator<= (const Time &) const`

Comparing two [Time](#) objects.

5.268.3.10 `Time& Arc::Time::operator=(const char *)`

Assignment operator from a char pointer.

5.268.3.11 `Time& Arc::Time::operator=(const std::string &)`

Assignment operator from a string.

5.268.3.12 `Time& Arc::Time::operator=(const Time &)`

Assignment operator from a [Time](#).

5.268.3.13 `Time& Arc::Time::operator=(time_t)`

Assignment operator from a `time_t`.

5.268.3.14 `bool Arc::Time::operator== (const Time &) const`

Comparing two [Time](#) objects.

5.268.3.15 `bool Arc::Time::operator> (const Time &) const`

Comparing two [Time](#) objects.

5.268.3.16 `bool Arc::Time::operator>= (const Time &) const`

Comparing two [Time](#) objects.

5.268.3.17 `static void Arc::Time::SetFormat (const TimeFormat &) [static]`

Sets the default format for time strings.

5.268.3.18 `void Arc::Time::SetTime (time_t)`

sets the time

Referenced by [DataStaging::DTR::set_timeout\(\)](#).

5.268.3.19 `void Arc::Time::SetTime (time_t time, uint32_t nanosec)`

sets the fine grained time

5.268.3.20 `std::string Arc::Time::str (const TimeFormat & =time_format) const`

Returns a string representation of the time, using the specified format.

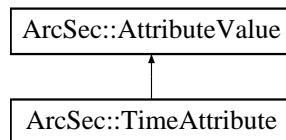
The documentation for this class was generated from the following file:

- [DateTime.h](#)

5.269 ArcSec::TimeAttribute Class Reference

```
#include <DateTimeAttribute.h>
```

Inheritance diagram for ArcSec::TimeAttribute:



Public Member Functions

- virtual bool [equal \(AttributeValue *other, bool check_id=true\)](#)
- virtual std::string [encode \(\)](#)
- virtual std::string [getType \(\)](#)
- virtual std::string [getId \(\)](#)

5.269.1 Detailed Description

Format: HHMMSSZ HH:MM:SS HH:MM:SS+HH:MM HH:MM:SSZ

5.269.2 Member Function Documentation

5.269.2.1 virtual std::string ArcSec::TimeAttribute::encode () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.269.2.2 virtual bool ArcSec::TimeAttribute::equal (AttributeValue * value, bool check_id = true) [virtual]

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.269.2.3 virtual std::string ArcSec::TimeAttribute::getId () [inline, virtual]

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.269.2.4 virtual std::string ArcSec::TimeAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

- [DateTimeAttribute.h](#)

5.270 Arc::TimedMutex Class Reference

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

- [Thread.h](#)

5.271 DataStaging::TransferParameters Class Reference

```
#include <DTR.h>
```

Public Member Functions

- [TransferParameters \(\)](#)

Data Fields

- unsigned long long int [min_average_bandwidth](#)
- unsigned int [max_inactivity_time](#)
- unsigned long long int [min_current_bandwidth](#)
- unsigned int [averaging_time](#)

5.271.1 Detailed Description

Represents limits and properties of a [DTR](#) transfer. These generally apply to all DTRs.

5.271.2 Field Documentation

5.271.2.1 unsigned int DataStaging::TransferParameters::max_inactivity_time

Maximum inactivity time in sec - if transfer stops for longer than this time it should be killed

5.271.2.2 unsigned long long int DataStaging::TransferParameters::min_average_bandwidth

Minimum average bandwidth in bytes/sec - if the average bandwidth used drops below this level the transfer should be killed

5.271.2.3 unsigned long long int DataStaging::TransferParameters::min_current_bandwidth

Minimum current bandwidth - if bandwidth averaged over averaging_time is less than minimum the transfer should be killed (allows transfers which slow down to be killed quicker)

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

- [DTR.h](#)

5.272 DataStaging::TransferShares Class Reference

[TransferShares](#) is used to implement fair-sharing and priorities.

```
#include <TransferShares.h>
```

Public Types

- enum [ShareType](#) {
 USER, VO, GROUP, ROLE,
 NONE }

Public Member Functions

- [TransferShares \(\)](#)
- [~TransferShares \(\)](#)
- [TransferShares \(const TransferShares &shares\)](#)
- [TransferShares & operator= \(const TransferShares &shares\)](#)
- [std::string extract_share_info \(const DTR &DTRToExtract\)](#)
- [void calculate_shares \(int TotalNumberOfSlots\)](#)
- [void increase_transfer_share \(const std::string &ShareToIncrease\)](#)
- [void decrease_transfer_share \(const std::string &ShareToDecrease\)](#)
- [void decrease_number_of_slots \(const std::string &ShareToDecrease\)](#)
- [bool can_start \(const std::string &ShareToStart\)](#)
- [bool is_configured \(const std::string &ShareToCheck\)](#)
- [int get_basic_priority \(const std::string &ShareToCheck\)](#)
- [void set_reference_share \(const std::string &RefShare, int Priority\)](#)
- [void set_reference_shares \(const std::map< std::string, int > &shares\)](#)
- [void set_share_type \(ShareType Type\)](#)
- [void set_share_type \(const std::string &type\)](#)
- [std::string conf \(\) const](#)

5.272.1 Detailed Description

[TransferShares](#) is used to implement fair-sharing and priorities.

[TransferShares](#) defines the algorithm used to prioritise and share transfers among different users or groups. It contains configuration configuration information on the share type and reference shares. The [Scheduler](#) uses [TransferShares](#) to determine which DTRs in the queue for Delivery go first to Delivery. The calculation is based on the configuration and the currently active shares (the DTRs already in Delivery). [can_start\(\)](#) is the method called by the [Scheduler](#) to determine whether a particular share has an available slot in Delivery.

5.272.2 Member Enumeration Documentation

5.272.2.1 enum DataStaging::TransferShares::ShareType

The criterion for assigning a share to a [DTR](#).

Enumerator:

USER Shares are defined per DN of the user's proxy.

VO Shares are defined per VOMS VO of the user's proxy.

GROUP Shares are defined per VOMS group of the user's proxy.

ROLE Shares are defined per VOMS role of the user's proxy.

NONE No share criterion - all DTRs will be assigned to a single share.

5.272.3 Member Function Documentation

5.272.3.1 void DataStaging::TransferShares::calculate_shares (int *TotalNumberOfSlots*)

Calculate how many slots to assign to each active share.

This method is called each time the [Scheduler](#) loops to calculate the number of slots to assign to each share, based on the current number of active shares and the shares' relative priorities.

5.272.3.2 void DataStaging::TransferShares::decrease_number_of_slots (const std::string & *ShareToDecrease*)

Decrease by one the number of slots available to the given share.

Called when there is a Delivery slot already used by this share to reduce the number available.

5.272.3.3 void DataStaging::TransferShares::decrease_transfer_share (const std::string & *ShareToDecrease*)

Decrease by one the active count for the given share.

Called when a completed [DTR](#) leaves the system.

5.272.3.4 void DataStaging::TransferShares::increase_transfer_share (const std::string & *ShareToIncrease*)

Increase by one the active count for the given share.

Called when a new [DTR](#) enters the system.

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

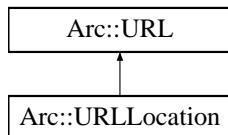
- TransferShares.h

5.273 Arc::URL Class Reference

Class to hold general URLs.

```
#include <URL.h>
```

Inheritance diagram for Arc::URL:



Public Types

- enum [Scope](#)

Public Member Functions

- [URL \(\)](#)
- [URL \(const std::string &url\)](#)
- virtual [~URL \(\)](#)
- const std::string & [Protocol \(\) const](#)
- void [ChangeProtocol \(const std::string &newprot\)](#)
- bool [IsSecureProtocol \(\) const](#)
- const std::string & [Username \(\) const](#)
- const std::string & [Passwd \(\) const](#)
- const std::string & [Host \(\) const](#)
- void [ChangeHost \(const std::string &newhost\)](#)
- int [Port \(\) const](#)
- void [ChangePort \(int newport\)](#)
- const std::string & [Path \(\) const](#)
- std::string [FullPath \(\) const](#)
- std::string [FullPathURIEncoded \(\) const](#)
- void [ChangePath \(const std::string &newpath\)](#)
- void [ChangeFullPath \(const std::string &newpath\)](#)
- const std::map< std::string, std::string > & [HTTPOptions \(\) const](#)
- const std::string & [HTTPOption \(const std::string &option, const std::string &undefined="""\) const](#)
- bool [AddHTTPOption \(const std::string &option, const std::string &value, bool overwrite=true\)](#)
- void [RemoveHTTPOption \(const std::string &option\)](#)
- const std::list< std::string > & [LDAPAttributes \(\) const](#)
- void [AddLDAPAttribute \(const std::string &attribute\)](#)
- Scope [LDAPScope \(\) const](#)

- void `ChangeLDAPScope` (const `Scope` newscope)
- const std::string & `LDAPFilter` () const
- void `ChangeLDAPFilter` (const std::string &newfilter)
- const std::map< std::string, std::string > & `Options` () const
- const std::string & `Option` (const std::string &option, const std::string &undefined="") const
- const std::map< std::string, std::string > & `MetaDataTable` () const
- const std::string & `MetaDataTable` (const std::string &option, const std::string &undefined="") const
- bool `AddOption` (const std::string &option, const std::string &value, bool overwrite=true)
- bool `AddOption` (const std::string &option, bool overwrite=true)
- void `AddMetaDataTable` (const std::string &option, const std::string &value, bool overwrite=true)
- void `AddLocation` (const `URLLocation` &location)
- const std::list< `URLLocation` > & `Locations` () const
- const std::map< std::string, std::string > & `CommonLocOptions` () const
- const std::string & `CommonLocOption` (const std::string &option, const std::string &undefined="") const
- void `RemoveOption` (const std::string &option)
- void `RemoveMetaDataTable` (const std::string &option)
- virtual std::string `str` () const
- virtual std::string `plainstr` () const
- virtual std::string `fullstr` () const
- virtual std::string `ConnectionURL` () const
- bool `operator<` (const `URL` &url) const
- bool `operator==` (const `URL` &url) const
- operator bool () const
- bool `StringMatches` (const std::string &str) const
- std::map< std::string, std::string > `ParseOptions` (const std::string &optstring, char separator)

Static Public Member Functions

- static std::string `OptionString` (const std::map< std::string, std::string > &options, char separator)

Protected Member Functions

- void `ParsePath` (void)

Static Protected Member Functions

- static std::string `BaseDN2Path` (const std::string &)
- static std::string `Path2BaseDN` (const std::string &)

Protected Attributes

- std::string [protocol](#)
- std::string [username](#)
- std::string [passwd](#)
- std::string [host](#)
- bool [ip6addr](#)
- int [port](#)
- std::string [path](#)
- std::map< std::string, std::string > [httpoptions](#)
- std::map< std::string, std::string > [metadataoptions](#)
- std::list< std::string > [ldapattributes](#)
- Scope [ldapscope](#)
- std::string [ldapfilter](#)
- std::map< std::string, std::string > [urloptions](#)
- std::list< [URLLocation](#) > [locations](#)
- std::map< std::string, std::string > [commonlocoptions](#)
- bool [valid](#)

Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [URL](#) &u)

5.273.1 Detailed Description

Class to hold general URLs.

The [URL](#) is split into protocol, hostname, port and path. This class tries to follow RFC 3986 for splitting URLs, at least for protocol + host part. It also accepts local file paths which are converted to [file:path](#). The usual system dependent file paths are supported. Relative paths are converted to absolute paths by prepending them with current working directory path. A file path can't start from # symbol. If the string representation of [URL](#) starts from '@' then it is treated as path to a file containing a list of URLs.

A [URL](#) is parsed in the following way:

[protocol:]//[username:passwd@[host][:port]][;urloptions[...]][/path[?httpoption[&...]][:metadataoption[...]]]

The 'protocol' and 'host' parts are treated as case-insensitive and to avoid confusion are converted to lowercase in constructor. Note that 'path' is always converted to absolute path in constructor. The meaning of 'absolute' may depend upon [URL](#) type. For generic [URL](#) and local POSIX file paths that means path starts from / like

/path/to/file

For Windows paths absolute path may look like

C:\path\to\file

It is important to note that path still can be empty. For referencing local file using absolute path on POSIX filesystem one may use either

`file:///path/to/file` or `file:/path/to/file`

Relative path will look like

`file:to/file`

For local Windows files possible URLs are

`file:C:\path\to\file` or `file:to\file`

URLs representing LDAP resources have different structure of options following 'path' part:

`ldap://host[:port][;urloptions[...]][/path[?attributes[?scope[?filter]]]]`

For LDAP URLs paths are converted from `/key1=value1/.../keyN=valueN` notation to `keyN=valueN,...,key1=value1` and hence path does not contain leading `/`. If LDAP URL initially had path in second notation leading `/` is treated as separator only and is stripped.

URLs of indexing services optionally may have locations specified before 'host' part

`protocol://[location[;location[...]]@][host][:port]...`

The structure of 'location' element is protocol specific.

5.273.2 Member Enumeration Documentation

5.273.2.1 enum Arc::URL::Scope

Scope for LDAP URLs

5.273.3 Constructor & Destructor Documentation

5.273.3.1 Arc::URL::URL()

Empty constructor. Necessary when the class is part of another class and the like.

5.273.3.2 Arc::URL::URL(const std::string & url)

Constructs a new [URL](#) from a string representation.

5.273.3.3 virtual Arc::URL::~URL() [virtual]

[URL](#) Destructor

5.273.4 Member Function Documentation

5.273.4.1 `bool Arc::URL::AddHTTPOption (const std::string & option, const std::string & value, bool overwrite = true)`

Adds a HTP option with the given value. Returns false if overwrite is false and option already exists, true otherwise.

5.273.4.2 `void Arc::URL::AddLDAPAttribute (const std::string & attribute)`

Adds an LDAP attribute.

5.273.4.3 `void Arc::URL::AddLocation (const URLLocation & location)`

Adds a Location

5.273.4.4 `void Arc::URL::AddMetaDataOption (const std::string & option, const std::string & value, bool overwrite = true)`

Adds a metadata option

5.273.4.5 `bool Arc::URL::AddOption (const std::string & option, const std::string & value, bool overwrite = true)`

Adds a [URL](#) option with the given value. Returns false if overwrite is false and option already exists, true otherwise. Note that some compilers may interpret AddOption("name", "value") as a call to AddOption(string, bool) so it is recommended to use explicit string types when calling this method.

5.273.4.6 `bool Arc::URL::AddOption (const std::string & option, bool overwrite = true)`

Adds a [URL](#) option where option has the format "name=value". Returns false if overwrite is true and option already exists or if option does not have the correct format. Returns true otherwise.

5.273.4.7 `static std::string Arc::URL::BaseDN2Path (const std::string &) [static, protected]`

a private method that converts an ldap basedn to a path.

5.273.4.8 `void Arc::URL::ChangeFullPath (const std::string & newpath)`

Changes the path of the [URL](#) and all options attached.

5.273.4.9 void Arc::URL::ChangeHost (const std::string & *newhost*)

Changes the hostname of the [URL](#).

5.273.4.10 void Arc::URL::ChangeLDAPFilter (const std::string & *newfilter*)

Changes the LDAP filter.

5.273.4.11 void Arc::URL::ChangeLDAPScope (const Scope *newscope*)

Changes the LDAP scope.

5.273.4.12 void Arc::URL::ChangePath (const std::string & *newpath*)

Changes the path of the [URL](#).

5.273.4.13 void Arc::URL::ChangePort (int *newport*)

Changes the port of the [URL](#).

5.273.4.14 void Arc::URL::ChangeProtocol (const std::string & *newprot*)

Changes the protocol of the [URL](#).

5.273.4.15 const std::string& Arc::URL::CommonLocOption (const std::string & *option*, const std::string & *undefined* = " ") const

Returns the value of a common location option.

Parameters

<i>option</i>	The option whose value is returned.
<i>undefined</i>	This value is returned if the common location option is not defined.

5.273.4.16 const std::map<std::string, std::string>& Arc::URL::CommonLocOptions () const

Returns the common location options if any.

5.273.4.17 virtual std::string Arc::URL::ConnectionURL () const [virtual]

Returns a string representation with protocol, host and port only

5.273.4.18 `std::string Arc::URL::FullPath() const`

Returns the path of the [URL](#) with all options attached.

5.273.4.19 `std::string Arc::URL::FullPathURIEncoded() const`

Returns the path and all options, URI-encoded according to RFC 3986. Forward slashes ('/') in the path are not encoded but are encoded in the options.

5.273.4.20 `virtual std::string Arc::URL::fullstr() const [virtual]`

Returns a string representation including options and locations

Reimplemented in [Arc::URLLocation](#).

5.273.4.21 `const std::string& Arc::URL::Host() const`

Returns the hostname of the [URL](#).

5.273.4.22 `const std::string& Arc::URL::HTTPOption(const std::string & option, const std::string & undefined = " ") const`

Returns the value of an HTTP option.

Parameters

<code>option</code>	The option whose value is returned.
<code>undefined</code>	This value is returned if the HTTP option is not defined.

5.273.4.23 `const std::map<std::string, std::string>& Arc::URL::HTTPOptions() const`

Returns HTTP options if any.

5.273.4.24 `bool Arc::URL::IsSecureProtocol() const`

Indicates whether the protocol is secure or not.

5.273.4.25 `const std::list<std::string>& Arc::URL::LDAPAttributes() const`

Returns the LDAP attributes if any.

5.273.4.26 `const std::string& Arc::URL::LDAPFilter() const`

Returns the LDAP filter.

5.273.4.27 `Scope Arc::URL::LDAPScope() const`

Returns the LDAP scope.

5.273.4.28 `const std::list<URLLocation>& Arc::URL::Locations() const`

Returns the locations if any.

5.273.4.29 `const std::string& Arc::URL::MetaDataTable(const std::string & option, const std::string & undefined = "") const`

Returns the value of a metadata option.

Parameters

<code>option</code>	The option whose value is returned.
<code>undefined</code>	This value is returned if the metadata option is not defined.

5.273.4.30 `const std::map<std::string, std::string>& Arc::URL::MetaDataTable() const`

Returns metadata options if any.

5.273.4.31 `Arc::URL::operator bool() const`

Check if instance holds valid [URL](#)

5.273.4.32 `bool Arc::URL::operator<(const URL & url) const`

Compares one [URL](#) to another

5.273.4.33 `bool Arc::URL::operator==(const URL & url) const`

Is one [URL](#) equal to another?

5.273.4.34 `const std::string& Arc::URL::Option(const std::string & option, const std::string & undefined = "") const`

Returns the value of a [URL](#) option.

Parameters

<code>option</code>	The option whose value is returned.
<code>undefined</code>	This value is returned if the URL option is not defined.

5.273.4.35 `const std::map<std::string, std::string>& Arc::URL::Options() const`

Returns [URL](#) options if any.

5.273.4.36 `static std::string Arc::URL::OptionString(const std::map< std::string, std::string > & options, char separator) [static]`

Returns a string representation of the options given in the options map

5.273.4.37 `std::map<std::string, std::string> Arc::URL::ParseOptions(const std::string & optstring, char separator)`

Parse a string of options separated by separator into an attribute->value map

5.273.4.38 `void Arc::URL::ParsePath(void) [protected]`

Convenience method for splitting schema specific part into path and options

5.273.4.39 `const std::string& Arc::URL::Passwd() const`

Returns the password of the [URL](#).

5.273.4.40 `const std::string& Arc::URL::Path() const`

Returns the path of the [URL](#).

5.273.4.41 `static std::string Arc::URL::Path2BaseDN(const std::string &) [static, protected]`

a private method that converts an ldap path to a basedn.

5.273.4.42 `virtual std::string Arc::URL::plainstr() const [virtual]`

Returns a string representation of the [URL](#) without any options

5.273.4.43 `int Arc::URL::Port() const`

Returns the port of the [URL](#).

5.273.4.44 `const std::string& Arc::URL::Protocol() const`

Returns the protocol of the [URL](#).

5.273.4.45 void Arc::URL::RemoveHTTPOption (const std::string & *option*)

Removes a HTTP option if exists.

Parameters

<i>option</i>	The option to remove.
---------------	-----------------------

5.273.4.46 void Arc::URL::RemoveMetaDataTable (const std::string & *option*)

Remove a metadata option if exists.

Parameters

<i>option</i>	The option to remove.
---------------	-----------------------

5.273.4.47 void Arc::URL::RemoveOption (const std::string & *option*)

Removes a [URL](#) option if exists.

Parameters

<i>option</i>	The option to remove.
---------------	-----------------------

5.273.4.48 virtual std::string Arc::URL::str () const [virtual]

Returns a string representation of the [URL](#) including meta-options.

Reimplemented in [Arc::URLLocation](#).

5.273.4.49 const std::string& Arc::URL::Username () const

Returns the username of the [URL](#).

5.273.5 Friends And Related Function Documentation

5.273.5.1 std::ostream& operator<< (std::ostream & *out*, const URL & *u*) [friend]

Overloaded operator << to print a [URL](#).

5.273.6 Field Documentation

5.273.6.1 `std::map<std::string, std::string> Arc::URL::commonloptions` [protected]

common location options for index server URLs.

5.273.6.2 `std::string Arc::URL::host` [protected]

hostname of the url.

5.273.6.3 `std::map<std::string, std::string> Arc::URL::httpoptions` [protected]

HTTP options of the url.

5.273.6.4 `bool Arc::URL::ip6addr` [protected]

if host is IPv6 numerical address notation.

5.273.6.5 `std::list<std::string> Arc::URL::ldapattributes` [protected]

LDAP attributes of the url.

5.273.6.6 `std::string Arc::URL::ldapfilter` [protected]

LDAP filter of the url.

5.273.6.7 `Scope Arc::URL::ldapscope` [protected]

LDAP scope of the url.

5.273.6.8 `std::list<URLlocation> Arc::URL::locations` [protected]

locations for index server URLs.

5.273.6.9 `std::map<std::string, std::string> Arc::URL::metadataoptions` [protected]

Meta data options

5.273.6.10 `std::string Arc::URL::passwd` [protected]

password of the url.

5.273.6.11 `std::string Arc::URL::path` [protected]

the url path.

5.273.6.12 `int Arc::URL::port` [protected]

portnumber of the url.

5.273.6.13 `std::string Arc::URL::protocol` [protected]

the url protocol.

5.273.6.14 `std::map<std::string, std::string> Arc::URL::urloptions` [protected]

options of the url.

5.273.6.15 `std::string Arc::URL::username` [protected]

username of the url.

5.273.6.16 `bool Arc::URL::valid` [protected]

flag to describe validity of [URL](#)

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

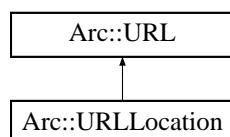
- `URL.h`

5.274 `Arc::URLLocation` Class Reference

Class to hold a resolved [URL](#) location.

```
#include <URL.h>
```

Inheritance diagram for `Arc::URLLocation`:



Public Member Functions

- [URLLocation \(const std::string &url=""\)](#)
- [URLLocation \(const std::string &url, const std::string &name\)](#)
- [URLLocation \(const URL &url\)](#)
- [URLLocation \(const URL &url, const std::string &name\)](#)
- [URLLocation \(const std::map< std::string, std::string > &options, const std::string &name\)](#)
- [virtual ~URLLocation \(\)](#)
- [const std::string & Name \(\) const](#)
- [virtual std::string str \(\) const](#)
- [virtual std::string fullstr \(\) const](#)

Protected Attributes

- [std::string name](#)

5.274.1 Detailed Description

Class to hold a resolved [URL](#) location.

It is specific to file indexing service registrations.

5.274.2 Constructor & Destructor Documentation

5.274.2.1 [Arc::URLLocation::URLLocation \(const std::string & url = " " \)](#)

Creates a [URLLocation](#) from a string representaion.

5.274.2.2 [Arc::URLLocation::URLLocation \(const std::string & url, const std::string & name \)](#)

Creates a [URLLocation](#) from a string representaion and a name.

5.274.2.3 [Arc::URLLocation::URLLocation \(const URL & url \)](#)

Creates a [URLLocation](#) from a [URL](#).

5.274.2.4 [Arc::URLLocation::URLLocation \(const URL & url, const std::string & name \)](#)

Creates a [URLLocation](#) from a [URL](#) and a name.

5.274.2.5 [Arc::URLLocation::URLLocation \(const std::map< std::string, std::string > & options, const std::string & name \)](#)

Creates a [URLLocation](#) from options and a name.

5.274.2.6 virtual Arc::URLLocation::~URLLocation() [virtual]

[URLLocation](#) destructor.

5.274.3 Member Function Documentation

5.274.3.1 virtual std::string Arc::URLLocation::fullstr() const [virtual]

Returns a string representation including options and locations

Reimplemented from [Arc::URL](#).

5.274.3.2 const std::string& Arc::URLLocation::Name() const

Returns the [URLLocation](#) name.

5.274.3.3 virtual std::string Arc::URLLocation::str() const [virtual]

Returns a string representation of the [URLLocation](#).

Reimplemented from [Arc::URL](#).

5.274.4 Field Documentation

5.274.4.1 std::string Arc::URLLocation::name [protected]

the [URLLocation](#) name as registered in the indexing service.

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

- URL.h

5.275 Arc::User Class Reference

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

- User.h

5.276 Arc::UserConfig Class Reference

User configuration class

```
#include <UserConfig.h>
```

Public Member Functions

- `UserConfig (initializeCredentialsType initializeCredentials=initializeCredentialsType())`
- `UserConfig (const std::string &conffile, initializeCredentialsType initializeCredentials=initializeCredentialsType(), bool loadSysConfig=true)`
- `UserConfig (const std::string &conffile, const std::string &jfile, initializeCredentialsType initializeCredentials=initializeCredentialsType(), bool loadSysConfig=true)`
- `UserConfig (const long int &ptraddr)`
- `void InitializeCredentials ()`
- `bool CredentialsFound () const`
- `bool LoadConfigurationFile (const std::string &conffile, bool ignoreJobListFile=true)`
- `bool SaveToFile (const std::string &filename) const`
- `void ApplyToConfig (BaseConfig &ccfg) const`
- `operator bool () const`
- `bool operator! () const`
- `bool JobListFile (const std::string &path)`
- `const std::string & JobListFile () const`
- `bool AddServices (const std::list< std::string > &services, ServiceType st)`
- `bool AddServices (const std::list< std::string > &selected, const std::list< std::string > &rejected, ServiceType st)`
- `const std::list< std::string > & GetSelectedServices (ServiceType st) const`
- `const std::list< std::string > & GetRejectedServices (ServiceType st) const`
- `void ClearSelectedServices ()`
- `void ClearSelectedServices (ServiceType st)`
- `void ClearRejectedServices ()`
- `void ClearRejectedServices (ServiceType st)`
- `bool Timeout (int newTimeout)`
- `int Timeout () const`
- `bool Verbosity (const std::string &newVerbosity)`
- `const std::string & Verbosity () const`
- `bool Broker (const std::string &name)`
- `bool Broker (const std::string &name, const std::string &argument)`
- `const std::pair< std::string, std::string > & Broker () const`
- `bool Bartender (const std::vector< URL > &urls)`
- `void AddBartender (const URL &url)`
- `const std::vector< URL > & Bartender () const`
- `bool VOMSESPath (const std::string &path)`
- `const std::string & VOMSESPath ()`
- `bool UserName (const std::string &name)`
- `const std::string & UserName () const`
- `bool Password (const std::string &newPassword)`
- `const std::string & Password () const`
- `bool ProxyPath (const std::string &newProxyPath)`
- `const std::string & ProxyPath () const`
- `bool CertificatePath (const std::string &newCertificatePath)`
- `const std::string & CertificatePath () const`
- `bool KeyPath (const std::string &newKeyPath)`

- const std::string & **KeyPath** () const
- bool **KeyPassword** (const std::string &newKeyPassword)
- const std::string & **KeyPassword** () const
- bool **KeySize** (int newKeySize)
- int **KeySize** () const
- bool **CACertificatePath** (const std::string &newCACertificatePath)
- const std::string & **CACertificatePath** () const
- bool **CACertificatesDirectory** (const std::string &newCACertificatesDirectory)
- const std::string & **CACertificatesDirectory** () const
- bool **CertificateLifeTime** (const **Period** &newCertificateLifeTime)
- const **Period** & **CertificateLifeTime** () const
- bool **SLCS** (const **URL** &newSLCS)
- const **URL** & **SLCS** () const
- bool **StoreDirectory** (const std::string &newStoreDirectory)
- const std::string & **StoreDirectory** () const
- bool **JobDownloadDirectory** (const std::string &newDownloadDirectory)
- const std::string & **JobDownloadDirectory** () const
- bool **IdPName** (const std::string &name)
- const std::string & **IdPName** () const
- bool **OverlayFile** (const std::string &path)
- const std::string & **OverlayFile** () const
- bool **UtilsDirPath** (const std::string &dir)
- const std::string & **UtilsDirPath** () const
- void **SetUser** (const **User** &u)
- const **User** & **GetUser** () const

Static Public Attributes

- static const std::string **ARCUSERDIRECTORY**
- static const std::string **SYSCONFIG**
- static const std::string **SYSCONFIGARCLOC**
- static const std::string **DEFAULTCONFIG**
- static const std::string **EXAMPLECONFIG**
- static const int **DEFAULT_TIMEOUT** = 20
- static const std::string **DEFAULT_BROKER**

5.276.1 Detailed Description

User configuration class

This class provides a container for a selection of various attributes/parameters which can be configured to needs of the user, and can be read by implementing instances or programs. The class can be used in two ways. One can create a object from a configuration file, or simply set the desired attributes by using the setter method, associated with every setable attribute. The list of attributes which can be configured in this class are:

- certificatepath / [CertificatePath\(const std::string&\)](#)
- keypath / [KeyPath\(const std::string&\)](#)
- proxypath / [ProxyPath\(const std::string&\)](#)
- cacertificatesdirectory / [CACertificatesDirectory\(const std::string&\)](#)
- cacertificatepath / [CACertificatePath\(const std::string&\)](#)
- timeout / [Timeout\(int\)](#)
- joblist / [JobListFile\(const std::string&\)](#)
- defaultservices / [AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
- rejectservices / [AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
- verbosity / [Verbosity\(const std::string&\)](#)
- brokername / [Broker\(const std::string&\) or Broker\(const std::string&, const std::string&\)](#)
- brokerarguments / [Broker\(const std::string&\) or Broker\(const std::string&, const std::string&\)](#)
- bartender / [Bartender\(const std::list<URL>&\)](#)
- vomsserverpath / [VOMSESPath\(const std::string&\)](#)
- username / [UserName\(const std::string&\)](#)
- password / [Password\(const std::string&\)](#)
- keypassword / [KeyPassword\(const std::string&\)](#)
- keysize / [KeySize\(int\)](#)
- certificatelifetime / [CertificateLifeTime\(const Period&\)](#)
- slcs / [SLCS\(const URL&\)](#)
- storedirectory / [StoreDirectory\(const std::string&\)](#)
- jobdownloaddirectory / [JobDownloadDirectory\(const std::string&\)](#)
- idpname / [IdPName\(const std::string&\)](#)

where the first term is the name of the attribute used in the configuration file, and the second term is the associated setter method (for more information about a given attribute see the description of the setter method).

The configuration file should have a INI-style format and the [IniConfig](#) class will thus be used to parse the file. The above mentioned attributes should be placed in the common section. Another section is also valid in the configuration file, which is the alias section. Here it is possible to define aliases representing one or multiple services. These aliases can be used in the [AddServices\(const std::list<std::string>&, ServiceType\)](#) and

`AddServices(const std::list<std::string>&, const std::list<std::string>&, ServiceType)` methods.

The `UserConfig` class also provides a method `InitializeCredentials()` for locating user credentials by searching in different standard locations. The `CredentialsFound()` method can be used to test if locating the credentials succeeded.

5.276.2 Constructor & Destructor Documentation

5.276.2.1 `Arc::UserConfig::UserConfig (initializeCredentialsType initializeCredentials = initializeCredentialsType ())`

Create a `UserConfig` object.

The `UserConfig` object created by this constructor initializes only default values, and if specified by the `initializeCredentials` boolean credentials will be tried initialized using the `InitializeCredentials()` method. The object is only non-valid if initialization of credentials fails which can be checked with the `operator bool()` method.

Parameters

<code>initializeCredentials</code>	is a optional boolean indicating if the <code>InitializeCredentials()</code> method should be invoked, the default is <code>true</code> .
------------------------------------	---

See also

`InitializeCredentials()`
`operator bool()`

5.276.2.2 `Arc::UserConfig::UserConfig (const std::string & conffile, initializeCredentialsType initializeCredentials = initializeCredentialsType (), bool loadSysConfig = true)`

Create a `UserConfig` object.

The `UserConfig` object created by this constructor will, if specified by the `loadSysConfig` boolean, first try to load the system configuration file by invoking the `LoadConfigurationFile()` method, and if this fails a `::WARNING` is reported. Then the configuration file passed will be tried loaded using the before mentioned method, and if this fails an `::ERROR` is reported, and the created object will be non-valid. Note that if the passed file path is empty the example configuration will be tried copied to the default configuration file path specified by `DEFAULTCONFIG`. If the example file cannot be copied one or more `::WARNING` messages will be reported and no configuration will be loaded. If loading the configurations file succeeded and if `initializeCredentials` is `true` then credentials will be initialized using the `InitializeCredentials()` method, and if no valid credentials are found the created object will be non-valid.

Parameters

<code>conffile</code>	is the path to a INI-configuration file.
-----------------------	--

<i>initializeCredentials</i>	is a boolean indicating if credentials should be initialized, the default is true.
<i>loadSysConfig</i>	is a boolean indicating if the system configuration file should be loaded aswell, the default is true.

See also

[LoadConfigurationFile\(const std::string&, bool\)](#)
[InitializeCredentials\(\)](#)
[operator bool\(\)](#)
[SYSCONFIG](#)
[EXAMPLECONFIG](#)

5.276.2.3 Arc::UserConfig::UserConfig (const std::string & *confi*, const std::string & *jfile*, initializeCredentialsType *initializeCredentials* = initializeCredentialsType(), bool *loadSysConfig* = true)

Create a [UserConfig](#) object.

The [UserConfig](#) object created by this constructor does only differ from the [UserConfig\(const std::string&, bool, bool\)](#) constructor in that it is possible to pass the path of the job list file directly to this constructor. If the job list file *joblistfile* is empty, the behaviour of this constructor is exactly the same as the before mentioned, otherwise the job list file will be initilized by invoking the setter method [JobListFile\(const std::string&\)](#). If it fails the created object will be non-valid, otherwise the specified configuration file *confi* will be loaded with the *ignoreJobListFile* argument set to true.

Parameters

<i>confi</i>	is the path to a INI-configuration file
<i>jfile</i>	is the path to a (non-)existing job list file.
<i>initializeCredentials</i>	is a boolean indicating if credentials should be initialized, the default is true.
<i>loadSysConfig</i>	is a boolean indicating if the system configuration file should be loaded aswell, the default is true.

See also

[JobListFile\(const std::string&\)](#)
[LoadConfigurationFile\(const std::string&, bool\)](#)
[InitializeCredentials\(\)](#)
[operator bool\(\)](#)

5.276.2.4 Arc::UserConfig::UserConfig (const long int & *ptraddr*)

Language binding constructor.

The passed long int should be a pointer address to a [UserConfig](#) object, and this address is then casted into this [UserConfig](#) object.

Parameters

<i>ptraddr</i>	is an memory address to a UserConfig object.
----------------	--

5.276.3 Member Function Documentation**5.276.3.1 void Arc::UserConfig::AddBartender (const URL & *url*) [inline]**

Set bartenders, used to contact Chelonia.

Takes as input a Bartender [URL](#) and adds this to the list of bartenders.

Parameters

<i>url</i>	is a URL to be added to the list of bartenders.
------------	---

See also

[Bartender\(const std::list<URL>&\)](#)
[Bartender\(\) const](#)

5.276.3.2 bool Arc::UserConfig::AddServices (const std::list< std::string > & *services*, ServiceType *st*)

Add selected and rejected services.

This method adds selected services and adds services to reject from the specified list *services*, which contains string objects. The syntax of a single element in the list must be expressed in the following two formats:

[−] <flavour>:<service_url> |[−] <alias>

where the optional ‘-’ indicate that the service should be added to the private list of services to reject. In the first format the <flavour> part indicates the type of ACC plugin to use when contacting the service, which is specified by the [URL](#) <service_url>, and in the second format the <alias> part specifies a alias defined in a parsed configuration file, note that the alias must not contain any of the characters ‘:’, ‘,’ or ‘\t’. If a alias cannot be resolved an ::ERROR will be reported to the logger and the method will return false. If a element in the list *services* cannot be parsed an ::ERROR will be reported, and the element is skipped.

Two attributes are indirectly associated with this setter method ‘defaultservices’ and ‘rejectservices’. The values specified with the ‘defaultservices’ attribute will be added to the list of selected services, and like-wise with the ‘rejectservices’ attribute.

Parameters

<i>services</i>	is a list of services to either select or reject.
<i>st</i>	indicates the type of the specfied services.

Returns

This method returns `false` in case an alias cannot be resolved. In any other case `true` is returned.

See also

[AddServices\(const std::string&, const std::string&, ServiceType\)](#)
[GetSelectedServices\(\)](#)
[GetRejectedServices\(\)](#)
[ClearSelectedServices\(\)](#)
[ClearRejectedServices\(\)](#)
[LoadConfigurationFile\(\)](#)

5.276.3.3 bool Arc::UserConfig::AddServices (const std::list< std::string > & selected, const std::list< std::string > & rejected, ServiceType st)

Add selected and rejected services.

The only difference in behaviour of this method compared to the [AddServices\(const std::list<std::string>&, ServiceType\)](#) method is the input parameters and the format these parameters should follow. Instead of having an optional '-' in front of the string selected and rejected services should be specified in the two different arguments.

Two attributes are indirectly associated with this setter method 'defaultservices' and 'rejectservices'. The values specified with the 'defaultservices' attribute will be added to the list of selected services, and like-wise with the 'rejectservices' attribute.

Parameters

<code>selected</code>	is a list of services which will be added to the selected services of this object.
<code>rejected</code>	is a list of services which will be added to the rejected services of this object.
<code>st</code>	specifies the ServiceType of the services to add.

Returns

This method return `false` in case an alias cannot be resolved. In any other case `true` is returned.

See also

[AddServices\(const std::list<std::string>&, ServiceType\)](#)
[GetSelectedServices\(\)](#)
[GetRejectedServices\(\)](#)
[ClearSelectedServices\(\)](#)
[ClearRejectedServices\(\)](#)
[LoadConfigurationFile\(\)](#)

5.276.3.4 void Arc::UserConfig::ApplyToConfig (BaseConfig & ccfg) const

Apply credentials to [BaseConfig](#).

This methods sets the [BaseConfig](#) credentials to the credentials contained in this object. It also passes user defined configuration overlay if any.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\)](#)
[BaseConfig](#)

Parameters

<code>ccfg</code>	a BaseConfig object which will configured with the credentials of this object.
-------------------	--

5.276.3.5 bool Arc::UserConfig::Bartender (const std::vector< URL > & urls) [inline]

Set bartenders, used to contact Chelonia.

Takes as input a vector of Bartender URLs.

The attribute associated with this setter method is 'bartender'.

Parameters

<code>urls</code>	is a list of URL object to be set as bartenders.
-------------------	--

Returns

This method always returns `true`.

See also

[AddBartender\(const URL&\)](#)
[Bartender\(\) const](#)

5.276.3.6 const std::vector<URL>& Arc::UserConfig::Bartender () const [inline]

Get bartenders.

Returns a list of Bartender URLs

Returns

The list of bartender [URL](#) objects is returned.

See also

[Bartender\(const std::list<URL>&\)](#)
[AddBartender\(const URL&\)](#)

5.276.3.7 bool Arc::UserConfig::Broker (const std::string & name)

Set broker to use in target matching.

The string passed to this method should be in the format:

<name> [:<*argument*>]

where the *<name>* is the name of the broker and cannot contain any ':', and the optional *<argument>* should contain arguments which should be passed to the broker.

Two attributes are associated with this setter method 'brokername' and 'brokerarguments'.

Parameters

<i>name</i>	the broker name and argument specified in the format given above.
-------------	---

Returns

This method always returns true.

See also

[Broker](#)

[Broker\(const std::string&, const std::string&\)](#)

[Broker\(\) const](#)

[DEFAULT_BROKER](#)

5.276.3.8 const std::pair<std::string, std::string>& Arc::UserConfig::Broker () const [inline]

Get the broker and corresponding arguments.

The returned pair contains the broker name as the first component and the argument as the second.

See also

[Broker\(const std::string&\)](#)

[Broker\(const std::string&, const std::string&\)](#)

[DEFAULT_BROKER](#)

5.276.3.9 bool Arc::UserConfig::Broker (const std::string & name, const std::string & argument) [inline]

Set broker to use in target matching.

As opposed to the [Broker\(const std::string&\)](#) method this method sets broker name and arguments directly from the passed two arguments.

Two attributes are associated with this setter method 'brokername' and 'brokerarguments'.

Parameters

<i>name</i>	is the name of the broker.
<i>argument</i>	is the arguments of the broker.

Returns

This method always returns `true`.

See also

[Broker](#)
[Broker\(const std::string&\)](#)
[Broker\(\) const](#)
[DEFAULT_BROKER](#)

5.276.3.10 `bool Arc::UserConfig::CACertificatePath(const std::string & newCACertificatePath) [inline]`

Set CA-certificate path.

The path to the file containing CA-certificate will be set when calling this method. This configuration parameter is deprecated - use `CACertificatesDirectory` instead. Only arc-slcs uses it.

The attribute associated with this setter method is 'cacertificatepath'.

Parameters

<i>newCACertificatePath</i>	is the path to the CA-certificate.
-----------------------------	------------------------------------

Returns

This method always returns `true`.

See also

[CACertificatePath\(\) const](#)

5.276.3.11 `const std::string& Arc::UserConfig::CACertificatePath() const [inline]`

Get path to CA-certificate.

Retrieve the path to the file containing CA-certificate. This configuration parameter is deprecated.

Returns

The path to the CA-certificate is returned.

See also

[CACertificatePath\(const std::string&\)](#)

5.276.3.12 `bool Arc::UserConfig::CACertificatesDirectory (const std::string & newCACertificatesDirectory) [inline]`

Set path to CA-certificate directory.

The path to the directory containing CA-certificates will be set when calling this method. Note that the [InitializeCredentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'cacertificatesdirectory'.

Parameters

<code>newCACertificatesDirectory</code>	is the path to the CA-certificate directory.
---	--

Returns

This method always returns `true`.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CACertificatesDirectory\(\) const](#)

5.276.3.13 `const std::string& Arc::UserConfig::CACertificatesDirectory () const [inline]`

Get path to CA-certificate directory.

Retrieve the path to the CA-certificate directory.

Returns

The path to the CA-certificate directory is returned.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CACertificatesDirectory\(const std::string&\)](#)

5.276.3.14 `bool Arc::UserConfig::CertificateLifeTime (const Period & newCertificateLifeTime) [inline]`

Set certificate life time.

Sets lifetime of user certificate which will be obtained from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'certificatelifetime'.

Parameters

<i>newCertificateLifeTime</i>	is the life time of a certificate, as a Period object.
-------------------------------	--

Returns

This method always returns `true`.

See also

[CertificateLifeTime\(\) const](#)

5.276.3.15 const Period& Arc::UserConfig::CertificateLifeTime() const [inline]

Get certificate life time.

Gets lifetime of user certificate which will be obtained from Short Lived Credentials Service.

Returns

The certificate life time is returned as a [Period](#) object.

See also

[CertificateLifeTime\(const Period&\)](#)

5.276.3.16 bool Arc::UserConfig::CertificatePath(const std::string & newCertificatePath) [inline]

Set path to certificate.

The path to user certificate will be set by this method. The path to the corresponding key can be set with the [KeyPath\(const std::string&\)](#) method. Note that the [InitializeCredentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'certificatepath'.

Parameters

<i>newCertificatePath</i>	is the path to the new certificate.
---------------------------	-------------------------------------

Returns

This method always returns `true`.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CertificatePath\(\) const](#)

[KeyPath\(const std::string&\)](#)

5.276.3.17 `const std::string& Arc::UserConfig::CertificatePath() const [inline]`

Get path to certificate.

The path to the certificate is returned when invoking this method.

Returns

The certificate path is returned.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CertificatePath\(const std::string&\)](#)
[KeyPath\(\) const](#)

5.276.3.18 `void Arc::UserConfig::ClearRejectedServices(ServiceType st)`

Clear rejected services with specified ServiceType.

Calling this method will cause the internally stored rejected services with the ServiceType *st* to be cleared.

See also

[ClearRejectedServices\(\)](#)
[ClearSelectedServices\(ServiceType\)](#)
[AddServices\(const std::list<std::string>&, ServiceType\)](#)
[AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
[GetRejectedServices\(\)](#)

5.276.3.19 `void Arc::UserConfig::ClearSelectedServices()`

Clear selected services.

Calling this method will cause the internally stored rejected services to be cleared.

See also

[ClearRejectedServices\(ServiceType\)](#)
[ClearSelectedServices\(\)](#)
[AddServices\(const std::list<std::string>&, ServiceType\)](#)
[AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
[GetRejectedServices\(\)](#)

5.276.3.20 void Arc::UserConfig::ClearSelectedServices (ServiceType st)

Clear selected services with specified ServiceType.

Calling this method will cause the internally stored selected services with the ServiceType *st* to be cleared.

See also

[ClearSelectedServices\(\)](#)
[ClearRejectedServices\(ServiceType\)](#)
[AddServices\(const std::list<std::string>&, ServiceType\)](#)
[AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
[GetSelectedServices\(\)](#)

5.276.3.21 void Arc::UserConfig::ClearSelectedServices ()

Clear selected services.

Calling this method will cause the internally stored selected services to be cleared.

See also

[ClearSelectedServices\(ServiceType\)](#)
[ClearRejectedServices\(\)](#)
[AddServices\(const std::list<std::string>&, ServiceType\)](#)
[AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
[GetSelectedServices\(\)](#)

5.276.3.22 bool Arc::UserConfig::CredentialsFound () const [inline]

Validate credential location.

Valid credentials consists of a combination of a path to existing CA-certificate directory and either a path to existing proxy or a path to existing user key/certificate pair. If valid credentials are found this method returns `true`, otherwise `false` is returned.

Returns

`true` if valid credentials are found, otherwise `false`.

See also

[InitializeCredentials\(\)](#)

5.276.3.23 const std::list<std::string>& Arc::UserConfig::GetRejectedServices (ServiceType
st) const

Get rejected services.

Get the rejected services with the ServiceType specified by *st*.

Parameters

<i>st</i>	specifies which ServiceType should be returned by the method.
-----------	---

Returns

The rejected services is returned.

See also

[AddServices\(const std::list<std::string>&, ServiceType\)](#)
[AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
[GetSelectedServices\(ServiceType\)](#)
[ClearRejectedServices\(\)](#)

5.276.3.24 const std::list<std::string>& Arc::UserConfig::GetSelectedServices (ServiceType
st) const

Get selected services.

Get the selected services with the ServiceType specified by *st*.

Parameters

<i>st</i>	specifies which ServiceType should be returned by the method.
-----------	---

Returns

The selected services is returned.

See also

[AddServices\(const std::list<std::string>&, ServiceType\)](#)
[AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#)
[GetRejectedServices\(ServiceType\) const](#)
[ClearSelectedServices\(\)](#)

5.276.3.25 const User& Arc::UserConfig:: GetUser () const [inline]

Get [User](#) for filesystem access.

Returns

The user identity to use for file system access

See also

[SetUser\(const User&\)](#)

5.276.3.26 `bool Arc::UserConfig::IdPName (const std::string & name) [inline]`

Set IdP name.

Sets Identity Provider name (Shibboleth) to which user belongs. It is used for contacting Short Lived Certificate Service.

The attribute associated with this setter method is 'idpname'.

Parameters

<code>name</code>	is the new IdP name.
-------------------	----------------------

Returns

This method always returns `true`.

See also

5.276.3.27 `const std::string& Arc::UserConfig::IdPName () const [inline]`

Get IdP name.

Gets Identity Provider name (Shibboleth) to which user belongs.

Returns

The IdP name

See also

[IdPName\(const std::string&\)](#)

5.276.3.28 `void Arc::UserConfig::InitializeCredentials ()`

Initialize user credentials.

The location of the user credentials will be tried located when calling this method and stored internally when found. The method searches in different locations. First the user proxy or the user key/certificate pair is tried located in the following order:

- Proxy path specified by the environment variable X509_USER_PROXY
- Key/certificate path specified by the environment X509_USER_KEY and X509_USER_CERT
- Proxy path specified in either configuration file passed to the constructor or explicitly set using the setter method [ProxyPath\(const std::string&\)](#)
- Key/certificate path specified in either configuration file passed to the constructor or explicitly set using the setter methods [KeyPath\(const std::string&\)](#) and [CertificatePath\(const std::string&\)](#)
- ProxyPath with file name x509up_u concatenated with the user ID located in the OS temporary directory.

If the proxy or key/certificate pair have been explicitly specified only the specified path(s) will be tried, and if not found a ::ERROR is reported. If the proxy or key/certificate have not been specified and it is not located in the temporary directory a ::WARNING will be reported and the host key/certificate pair is tried and then the Globus key/certificate pair and a ::ERROR will be reported if not found in any of these locations.

Together with the proxy and key/certificate pair, the path to the directory containing CA certificates is also tried located when invoking this method. The directory will be tried located in the following order:

- Path specified by the X509_CERT_DIR environment variable.
- Path explicitly specified either in a parsed configuration file using the cacertificatedirectory or by using the setter method [CACertificatesDirectory\(\)](#).
- Path created by concatenating the output of User::Home() with '.globus' and 'certificates' separated by the directory delimiter.
- Path created by concatenating the output of Glib::get_home_dir() with '.globus' and 'certificates' separated by the directory delimiter.
- Path created by concatenating the output of [ArcLocation::Get\(\)](#), with 'etc' and 'certificates' separated by the directory delimiter.
- Path created by concatenating the output of [ArcLocation::Get\(\)](#), with 'etc', 'grid-security' and 'certificates' separated by the directory delimiter.
- Path created by concatenating the output of [ArcLocation::Get\(\)](#), with 'share' and 'certificates' separated by the directory delimiter.
- Path created by concatenating 'etc', 'grid-security' and 'certificates' separated by the directory delimiter.

If the CA certificate directory have explicitly been specified and the directory does not exist a ::ERROR is reported. If none of the directories above does not exist a ::ERROR is reported.

See also

[CredentialsFound\(\)](#)
[ProxyPath\(const std::string&\)](#)
[KeyPath\(const std::string&\)](#)
[CertificatePath\(const std::string&\)](#)
[CACertificatesDirectory\(const std::string&\)](#)

5.276.3.29 `bool Arc::UserConfig::JobDownloadDirectory (const std::string & newDownloadDirectory) [inline]`

Set download directory.

Sets directory which will be used to download the job directory using arcget command.

The attribute associated with this setter method is 'jobdownloaddirectory'.

Parameters

<code>newDownloadDirectory</code>	is the path to the download directory.
-----------------------------------	--

Returns

This method always returns `true`.

See also

5.276.3.30 `const std::string& Arc::UserConfig::JobDownloadDirectory () const [inline]`

Get download directory.

returns directory which will be used to download the job directory using arcget command.

The attribute associated with the method is 'jobdownloaddirectory'.

Returns

This method returns the job download directory.

See also

5.276.3.31 `bool Arc::UserConfig::JobListFile (const std::string & path)`

Set path to job list file.

The method takes a path to a file which will be used as the job list file for storing and reading job information. If the specified path *path* does not exist a empty job list file will be tried created. If creating the job list file in any way fails *false* will be returned and a ::ERROR message will be reported. Otherwise *true* is returned. If the directory containing the file does not exist, it will be tried created. The method will also return *false* if the file is not a regular file.

The attribute associated with this setter method is 'joblist'.

Parameters

<i>path</i>	the path to the job list file.
-------------	--------------------------------

Returns

If the job list file is a regular file or if it can be created *true* is returned, otherwise *false* is returned.

See also

[JobListFile\(\) const](#)

5.276.3.32 const std::string& Arc::UserConfig::JobListFile() const [inline]

Get a reference to the path of the job list file.

The job list file is used to store and fetch information about submitted computing jobs to computing services. This method will return the path to the specified job list file.

Returns

The path to the job list file is returned.

See also

[JobListFile\(const std::string&\)](#)

5.276.3.33 bool Arc::UserConfig::KeyPassword(const std::string & newKeyPassword) [inline]

Set password for generated key.

Set password to be used to encode private key of credentials obtained from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'keypassword'.

Parameters

<i>newKey- Password</i>	is the new password to the key.
-----------------------------	---------------------------------

Returns

This method always returns `true`.

See also

[KeyPassword\(\) const](#)
[KeyPath\(const std::string&\)](#)
[KeySize\(int\)](#)

5.276.3.34 `const std::string& Arc::UserConfig::KeyPassword() const` [inline]

Get password for generated key.

Get password to be used to encode private key of credentials obtained from Short Lived Credentials [Service](#).

Returns

The key password is returned.

See also

[KeyPassword\(const std::string&\)](#)
[KeyPath\(\) const](#)
[KeySize\(\) const](#)

5.276.3.35 `bool Arc::UserConfig::KeyPath(const std::string & newKeyPath)` [inline]

Set path to key.

The path to user key will be set by this method. The path to the corresponding certificate can be set with the [CertificatePath\(const std::string&\)](#) method. Note that the [Initialize-Credentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'keypath'.

Parameters

`newKeyPath` is the path to the new key.

Returns

This method always returns `true`.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[KeyPath\(\) const](#)
[CertificatePath\(const std::string&\)](#)
[KeyPassword\(const std::string&\)](#)

[KeySize\(int\)](#)

5.276.3.36 const std::string& Arc::UserConfig::KeyPath() const [inline]

Get path to key.

The path to the key is returned when invoking this method.

Returns

The path to the user key is returned.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[KeyPath\(const std::string&\)](#)
[CertificatePath\(\) const](#)
[KeyPassword\(\) const](#)
[KeySize\(\) const](#)

5.276.3.37 int Arc::UserConfig::KeySize() const [inline]

Get key size.

Get size/strengt of private key of credentials obtained from Short Lived Credentials [Service](#).

Returns

The key size, as an integer, is returned.

See also

[KeySize\(int\)](#)
[KeyPath\(\) const](#)
[KeyPassword\(\) const](#)

5.276.3.38 bool Arc::UserConfig::KeySize(int *newKeySize*) [inline]

Set key size.

Set size/strengt of private key of credentials obtained from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'keysize'.

Parameters

newKeySize is the size, an integer, of the key.

Returns

This method always returns `true`.

See also

[KeySize\(\) const](#)
[KeyPath\(const std::string&\)](#)
[KeyPassword\(const std::string&\)](#)

5.276.3.39 `bool Arc::UserConfig::LoadConfigurationFile (const std::string & configfile, bool ignoreJobListFile = true)`

Load specified configuration file.

The configuration file passed is parsed by this method by using the [IniConfig](#) class. If the parsing is unsuccessful a ::WARNING is reported.

The format of the configuration file should follow that of INI, and every attribute present in the file is only allowed once, if otherwise a ::WARNING will be reported. The file can contain at most two sections, one named common and the other name alias. If other sections exist a ::WARNING will be reported. Only the following attributes is allowed in the common section of the configuration [file](#):

- certificatepath ([CertificatePath\(const std::string&\)](#))
- keypath ([KeyPath\(const std::string&\)](#))
- proxypath ([ProxyPath\(const std::string&\)](#))
- cacertificatesdirectory ([CACertificatesDirectory\(const std::string&\)](#))
- cacertificatepath ([CACertificatePath\(const std::string&\)](#))
- timeout ([Timeout\(int\)](#))
- joblist ([JobListFile\(const std::string&\)](#))
- defaultservices ([AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#))
- rejectservices ([AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#))
- verbosity ([Verbosity\(const std::string&\)](#))
- brokername ([Broker\(const std::string&\)](#) or [Broker\(const std::string&, const std::string&\)](#))
- brokerarguments ([Broker\(const std::string&\)](#) or [Broker\(const std::string&, const std::string&\)](#))
- bartender ([Bartender\(const std::list<URL>&\)](#))
- vomsserverpath ([VOMSESPPath\(const std::string&\)](#))

- username ([UserName\(const std::string&\)](#))
- password ([Password\(const std::string&\)](#))
- keypassword ([KeyPassword\(const std::string&\)](#))
- keyszie ([KeySize\(int\)](#))
- certificatelifetime ([CertificateLifeTime\(const Period&\)](#))
- slcs ([SLCS\(const URL&\)](#))
- storedirectory ([StoreDirectory\(const std::string&\)](#))
- jobdownloaddirory ([JobDownloadDirectory\(const std::string&\)](#))
- idpname ([IdPName\(const std::string&\)](#))

where the method in parentheses is the associated setter method. If other attributes exist in the common section a ::WARNING will be reported for each of these attributes. In the alias section aliases can be defined, and should represent a selection of services. The alias can then refered to by input to the [AddServices\(const std::list<std::string>&, ServiceType\)](#) and [AddServices\(const std::list<std::string>&, const std::list<std::string>&, ServiceType\)](#) methods. An alias can not contain any of the characters '!', '.', '' or '\t' and should be defined as follows:

```
<alias_name>=<service_type>:<flavour>:<service_url> | <alias_ref> [...]
```

where <alias_name> is the name of the defined alias, <service_type> is the service type in lower case, <flavour> is the type of middleware plugin to use, <service_url> is the [URL](#) which should be used to contact the service and <alias_ref> is another defined alias. The parsed aliases will be stored internally and resolved when needed. If a alias already exist, and another alias with the same name is parsed then this other alias will overwrite the existing alias.

Parameters

<code>confiile</code>	is the path to the configuration file.
<code>ignoreJob- ListFile</code>	is a optional boolean which indicates whether the joblistfile attribute in the configuration file should be ignored. Default is to ignored it (<code>true</code>).

Returns

If loading the configuration file succeeds `true` is returned, otherwise `false` is returned.

See also

[SaveToFile\(\)](#)

5.276.3.40 Arc::UserConfig::operator bool (void) const [inline]

Check for validity.

The validity of an object created from this class can be checked using this casting operator. An object is valid if the constructor did not encounter any errors.

See also

[operator!\(\)](#)

5.276.3.41 `bool Arc::UserConfig::operator! (void) const [inline]`

Check for non-validity.

See [operator bool\(\)](#) for a description.

See also

[operator bool\(\)](#)

5.276.3.42 `bool Arc::UserConfig::OverlayFile (const std::string & path) [inline]`

Set path to configuration overlay file.

Content of specified file is a backdoor to configuration XML generated from information stored in this class. The content of file is passed to [BaseConfig](#) class in `ApplyToConfig(BaseConfig&)` then merged with internal configuration XML representation. This feature is meant for quick prototyping/testing/tuning of functionality without rewriting code. It is meant for developers and most users won't need it.

The attribute associated with this setter method is 'overlayfile'.

Parameters

`path` is the new overlay file path.

Returns

This method always returns `true`.

See also

5.276.3.43 `const std::string& Arc::UserConfig::OverlayFile () const [inline]`

Get path to configuration overlay file.

Returns

The overlay file path

See also

[OverlayFile\(const std::string&\)](#)

5.276.3.44 `bool Arc::UserConfig::Password(const std::string & newPassword) [inline]`

Set password.

Set password which is used for requesting credentials from Short Lived Credentials Service.

The attribute associated with this setter method is 'password'.

Parameters

<code>newPass-</code> <code>word</code>	is the new password to set.
--	-----------------------------

Returns

This method always returns true.

See also

[Password\(\) const](#)

5.276.3.45 `const std::string& Arc::UserConfig::Password() const [inline]`

Get password.

Get password which is used for requesting credentials from Short Lived Credentials Service.

Returns

The password is returned.

See also

[Password\(const std::string&\)](#)

5.276.3.46 `bool Arc::UserConfig::ProxyPath(const std::string & newProxyPath) [inline]`

Set path to user proxy.

This method will set the path of the user proxy. Note that the [InitializeCredentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'proxypath'

Parameters

<code>newProxy-</code> <code>Path</code>	is the path to a user proxy.
---	------------------------------

Returns

This method always returns `true`.

See also

[InitializeCredentials\(\)](#)
[CredentialsFound\(\)](#)
[ProxyPath\(\) const](#)

5.276.3.47 `const std::string& Arc::UserConfig::ProxyPath() const` [inline]

Get path to user proxy.

Retrieve path to user proxy.

Returns

Returns the path to the user proxy.

See also

[ProxyPath\(const std::string&\)](#)

5.276.3.48 `bool Arc::UserConfig::SaveToFile(const std::string & filename) const`

Save to INI file.

This method will save the object data as a INI file. The saved file can be loaded with the `LoadConfigurationFile` method.

Parameters

<code>filename</code>	the name of the file which the data will be saved to.
-----------------------	---

Returns

`false` if unable to get handle on file, otherwise `true` is returned.

See also

[LoadConfigurationFile\(\)](#)

5.276.3.49 `void Arc::UserConfig::SetUser(const User & u)` [inline]

Set `User` for filesystem access.

Sometimes it is desirable to use the identity of another user when accessing the filesystem. This user can be specified through this method. By default this user is the same as the user running the process.

Parameters

<i>u</i>	User identity to use
----------	----------------------

5.276.3.50 const URL& Arc::UserConfig::SLCS() const [inline]

Get the [URL](#) to the Short Lived Certificate [Service](#) (SLCS).

Returns

The SLCS is returned.

See also

[SLCS\(const URL&](#)

5.276.3.51 bool Arc::UserConfig::SLCS(const URL & *newSLCS*) [inline]

Set the [URL](#) to the Short Lived Certificate [Service](#) (SLCS).

The attribute associated with this setter method is 'slcs'.

Parameters

<i>newSLCS</i>	is the URL to the SLCS
----------------	--

Returns

This method always returns `true`.

See also

[SLCS\(\) const](#)

5.276.3.52 bool Arc::UserConfig::StoreDirectory(const std::string & *newStoreDirectory*) [inline]

Set store directory.

Sets directory which will be used to store credentials obtained from Short Lived [Credential Service](#).

The attribute associated with this setter method is 'storedirectory'.

Parameters

<i>newStoreDirectory</i>	is the path to the store directory.
--------------------------	-------------------------------------

Returns

This method always returns `true`.

See also

[5.276.3.53 const std::string& Arc::UserConfig::StoreDirectory \(\) const \[inline\]](#)

Get store directory.

Sets directory which is used to store credentials obtained from Short Lived [Credential Service](#).

Returns

The path to the store directory is returned.

See also

[StoreDirectory\(const std::string&\)](#)

[5.276.3.54 bool Arc::UserConfig::Timeout \(int newTimeout \)](#)

Set timeout.

When communicating with a service the timeout specifies how long, in seconds, the communicating instance should wait for a response. If the response have not been received before this period in time, the connection is typically dropped, and an error will be reported.

This method will set the timeout to the specified integer. If the passed integer is less than or equal to 0 then `false` is returned and the timeout will not be set, otherwise `true` is returned and the timeout will be set to the new value.

The attribute associated with this setter method is 'timeout'.

Parameters

`newTimeout` | the new timeout value in seconds.

Returns

`false` in case `newTimeout <= 0`, otherwise `true`.

See also

[Timeout\(\) const](#)
[DEFAULT_TIMEOUT](#)

5.276.3.55 int Arc::UserConfig::Timeout() const [inline]

Get timeout.

Returns the timeout in seconds.

Returns

timeout in seconds.

See also

[Timeout\(int\)](#)

[DEFAULT_TIMEOUT](#)

5.276.3.56 bool Arc::UserConfig::UserName(const std::string & name) [inline]

Set user-name for SLCS.

Set username which is used for requesting credentials from Short Lived Credentials Service.

The attribute associated with this setter method is 'username'.

Parameters

name is the name of the user.

Returns

This method always return true.

See also

[UserName\(\) const](#)

5.276.3.57 const std::string& Arc::UserConfig::UserName() const [inline]

Get user-name.

Get username which is used for requesting credentials from Short Lived Credentials Service.

Returns

The username is returned.

See also

[UserName\(const std::string&\)](#)

5.276.3.58 bool Arc::UserConfig::UtilsDirPath (const std::string & *dir*)

Set path to directory storing utility files for DataPoints.

Some DataPoints can store information on remote services in local files. This method sets the path to the directory containing these files. For example arc* tools set it to ARCUSERDIRECTORY and A-REX sets it to the control directory. The directory is created if it does not exist.

Parameters

<i>path</i>	is the new utils dir path.
-------------	----------------------------

Returns

This method always returns true.

5.276.3.59 const std::string& Arc::UserConfig::UtilsDirPath () const [inline]

Get path to directory storing utility files for DataPoints.

Returns

The utils dir path

See also

[UtilsDirPath\(const std::string&\)](#)

5.276.3.60 const std::string& Arc::UserConfig::Verbosity () const [inline]

Get the user selected level of verbosity.

The string representation of the verbosity level specified by the user is returned when calling this method. If the user have not specified the verbosity level the empty string will be referenced.

Returns

the verbosity level, or empty if it has not been set.

See also

[Verbosity\(const std::string&\)](#)

5.276.3.61 bool Arc::UserConfig::Verbosity (const std::string & *newVerbosity*)

Set verbosity.

The verbosity will be set when invoking this method. If the string passed cannot be parsed into a corresponding LogLevel, using the function a ::WARNING is reported and false is returned, otherwise true is returned.

The attribute associated with this setter method is 'verbosity'.

Returns

true in case the verbosity could be set to a allowed LogLevel, otherwise false.

See also

[Verbosity\(\) const](#)

5.276.3.62 const std::string& Arc::UserConfig::VOMSESPath()

Get path to file containing VOMS configuration.

Get path to file which contains list of VOMS services and associated configuration parameters.

Returns

The path to VOMS configuration file is returned.

See also

[VOMSESPath\(const std::string&\)](#)

5.276.3.63 bool Arc::UserConfig::VOMSESPath(const std::string & path) [inline]

Set path to file containing VOMS configuration.

Set path to file which contains list of VOMS services and associated configuration parameters needed to contact those services. It is used by arcproxy.

The attribute associated with this setter method is 'vomsserverpath'.

Parameters

<i>path</i>	the path to VOMS configuration file
-------------	-------------------------------------

Returns

This method always return true.

See also

[VOMSESPath\(\) const](#)

5.276.4 Field Documentation

5.276.4.1 const std::string Arc::UserConfig::ARCUSERDIRECTORY [static]

Path to ARC user home directory.

The *ARCUSERDIRECTORY* variable is the path to the ARC home directory of the current user. This path is created using the User::Home() method.

See also

[User::Home\(\)](#)

5.276.4.2 const std::string Arc::UserConfig::DEFAULT_BROKER [static]

Default broker.

The *DEFAULT_BROKER* specifies the name of the broker which should be used in case no broker is explicitly chosen.

See also

[Broker](#)
[Broker\(const std::string&\)](#)
[Broker\(const std::string&, const std::string&\)](#)
[Broker\(\) const](#)

5.276.4.3 const int Arc::UserConfig::DEFAULT_TIMEOUT = 20 [static]

Default timeout in seconds.

The *DEFAULT_TIMEOUT* specifies interval which will be used in case no timeout interval have been explicitly specified. For a description about timeout see [Timeout\(int\)](#).

See also

[Timeout\(int\)](#)
[Timeout\(\) const](#)

5.276.4.4 const std::string Arc::UserConfig::DEFAULTCONFIG [static]

Path to default configuration file.

The *DEFAULTCONFIG* variable is the path to the default configuration file used in case no configuration file have been specified. The path is created from the ARCUSERDIRECTORY object.

5.276.4.5 const std::string Arc::UserConfig::EXAMPLECONFIG [static]

Path to example configuration.

The *EXAMPLECONFIG* variable is the path to the example configuration file.

5.276.4.6 const std::string Arc::UserConfig::SYSCONFIG [static]

Path to system configuration.

The *SYSCONFIG* variable is the path to the system configuration file. This variable is only equal to *SYSCONFIGARCLOC* if ARC is installed in the root (highly unlikely).

5.276.4.7 const std::string Arc::UserConfig::SYSCONFIGARCLOC [static]

Path to system configuration at ARC location.

The *SYSCONFIGARCLOC* variable is the path to the system configuration file which reside at the ARC installation location.

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

- UserConfig.h

5.277 Arc::UsernameToken Class Reference

Interface for manipulation of WS-Security according to Username Token [Profile](#).

```
#include <UsernameToken.h>
```

Public Types

- enum [PasswordType](#)

Public Member Functions

- [UsernameToken](#) (SOAPEnvelope &soap)
- [UsernameToken](#) (SOAPEnvelope &soap, const std::string &username, const std::string &password, const std::string &uid, [PasswordType](#) pwdtype)
- [UsernameToken](#) (SOAPEnvelope &soap, const std::string &username, const std::string &id, bool mac, int iteration)
- [operator bool](#) (void)
- std::string [Username](#) (void)
- bool [Authenticate](#) (const std::string &password, std::string &derived_key)
- bool [Authenticate](#) (std::istream &password, std::string &derived_key)

5.277.1 Detailed Description

Interface for manipulation of WS-Security according to Username Token [Profile](#).

5.277.2 Member Enumeration Documentation

5.277.2.1 enum Arc::UsernameToken::PasswordType

SOAP header element

5.277.3 Constructor & Destructor Documentation

5.277.3.1 Arc::UsernameToken::UsernameToken (SOAPEnvelope & soap)

Link to existing SOAP header and parse Username Token information. Username Token related information is extracted from SOAP header and stored in class variables.

5.277.3.2 Arc::UsernameToken::UsernameToken (SOAPEnvelope & soap, const std::string & username, const std::string & password, const std::string & uid, PasswordType pwdtype)

Add Username Token information into the SOAP header. Generated token contains elements Username and Password and is meant to be used for authentication.

Parameters

<i>soap</i>	the SOAP message
<i>username</i>	<wsse:Username>...</wsse:Username> - if empty it is entered interactively from stdin
<i>password</i>	<wsse:Password Type="...">...</wsse:Password> - if empty it is entered interactively from stdin
<i>uid</i>	<wsse:UsernameToken wsu:ID="...">
<i>pwdtype</i>	<wsse:Password Type="...">...</wsse:Password>

5.277.3.3 Arc::UsernameToken::UsernameToken (SOAPEnvelope & soap, const std::string & username, const std::string & id, bool mac, int iteration)

Add Username Token information into the SOAP header. Generated token contains elements Username and Salt and is meant to be used for deriving Key Derivation.

Parameters

<i>soap</i>	the SOAP message
<i>username</i>	<wsse:Username>...</wsse:Username>
<i>mac</i>	if derived key is meant to be used for Message Authentication Code
<i>iteration</i>	<wsse11:Iteration>...</wsse11:Iteration>

5.277.4 Member Function Documentation

5.277.4.1 `bool Arc::UsernameToken::Authenticate (const std::string & password, std::string & derived_key)`

Checks parsed/generated token against specified password. If token is meant to be used for deriving a key then key is returned in derived_key. In that case authentication is performed outside of [UsernameToken](#) class using obtained derived_key.

5.277.4.2 `bool Arc::UsernameToken::Authenticate (std::istream & password, std::string & derived_key)`

Checks parsed token against password stored in specified stream. If token is meant to be used for deriving a key then key is returned in derived_key

5.277.4.3 `Arc::UsernameToken::operator bool (void)`

Returns true if constructor succeeded

5.277.4.4 `std::string Arc::UsernameToken::Username (void)`

Returns username associated with this instance

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

- [UsernameToken.h](#)

5.278 Arc::UserSwitch Class Reference

```
#include <User.h>
```

5.278.1 Detailed Description

If this class is created user identity is switched to provided uid and gid. Due to internal lock there will be only one valid instance of this class. Any attempt to create another instance will block till first one is destroyed. If uid and gid are set to 0 then user identity is not switched. But lock is applied anyway. The lock has dual purpose. First and most important is to protect communication with underlying operating system which may depend on user identity. For that it is advisable for code which talks to operating system to acquire valid instance of this class. Care must be taken for not to hold that instance too long cause that may block other code in multithreaded environment. Other purpose of this lock is to provide workaround for glibc bug in __nptl_setuid. That bug causes lockup of seteuid() function if racing with fork. To avoid this problem the lock mentioned above is used by [Run](#) class while spawning new process.

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

- [User.h](#)

5.279 Arc::VOMSACInfo Class Reference

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

- VOMSUtil.h

5.280 Arc::VOMSTrustList Class Reference

```
#include <VOMSUtil.h>
```

Public Member Functions

- [VOMSTrustList](#) (const std::vector< std::string > &encoded_list)
- [VOMSTrustList](#) (const std::vector< VOMSTrustChain > &chains, const std::vector< VOMSTrustRegex > ®exs)
- VOMSTrustChain & [AddChain](#) (const VOMSTrustChain &chain)
- VOMSTrustChain & [AddChain](#) (void)
- [RegularExpression](#) & [AddRegex](#) (const VOMSTrustRegex ®)

5.280.1 Detailed Description

Stores definitions for making decision if VOMS server is trusted

5.280.2 Constructor & Destructor Documentation

```
5.280.2.1 Arc::VOMSTrustList::VOMSTrustList( const std::vector< std::string > & encoded_list
) }
```

Creates chain lists and regexps from plain list. List is made of chunks delimited by elements containing pattern "NEXT CHAIN". Each chunk with more than one element is converted into one instance of VOMSTrustChain. Chunks with single element are converted to VOMSTrustChain if element does not have special symbols. Otherwise it is treated as regular expression. Those symbols are '^','\$' and '*'. Trusted chains can be configured in two ways: one way is: <tls:VOMSCertTrustDNChain><tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=host/arthur.hep.lu.se</tls:VOMSCertTrustDN><tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority</tls:VOMSCertTrustDN><tls:VOMSCertTrustDN>---NEXT CHAIN---</tls:VOMSCertTrustDN> <tls:VOMSCertTrustDN>/DC=ch/DC=cern/DC=cert0</tls:VOMSCertTrustDN><tls:VOMSCertTrustDN>/DC=ch/DC=cern/CN=CERN Trusted Certification Authority</tls:VOMSCertTrustDN></tls:VOMSCertTrustDN> the other way is: <tls:VOMSCertTrustDNChain> <tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority</tls:VOMSCertTrustDN><tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority</tls:VOMSCertTrustDN><tls:VOMSCertTrustDNChain> <tls:VOMSCertTrustDNChain> <tls:VOMSCertTrustDN>/DC=ch/DC=cern/OU=Grid/DC=NorduGrid/CN=NorduGrid Certification Authority</tls:VOMSCertTrustDN><tls:VOMSCertTrustDN>/DC=ch/DC=cern/CN=CERN Trusted Certification Authority</tls:VOMSCertTrustDN></tls:VOMSCertTrustDN> each chunk is supposed to contain a suit of DN of trusted certificate chain, in which the first DN is the DN of the certificate (cert0) which

is used to sign the Attribute Certificate (AC), the second DN is the DN of the issuer certificate(cert1) which is used to sign cert0. So if there are one or more intermediate issuers, then there should be 3 or more than 3 DNs in this chunk (considering cert0 and the root certificate, plus the intermediate certificate) .

5.280.2.2 `Arc::VOMSTrustList::VOMSTrustList (const std::vector< VOMSTrustChain > & chains, const std::vector< VOMSTrustRegex > & regexs)`

Creates chain lists and regexps from those specified in arguments. See [AddChain\(\)](#) and [AddRegex\(\)](#) for more information.

5.280.3 Member Function Documentation

5.280.3.1 `VOMSTrustChain& Arc::VOMSTrustList::AddChain (const VOMSTrustChain & chain)`

Adds chain of trusted DNs to list. During verification each signature of AC is checked against all stored chains. DNs of chain of certificate used for signing AC are compared against DNs stored in these chains one by one. If needed DN of issuer of last certificate is checked too. Comparison succeeds if DNs in at least one stored chain are same as those in certificate chain. Comparison stops when all DNs in stored chain are compared. If there are more DNs in stored chain than in certificate chain then comparison fails. Empty stored list matches any certificate chain. Taking into account that certificate chains are verified down to trusted CA anyway, having more than one DN in stored chain seems to be useless. But such feature may be found useful by some very strict sysadmins. ??? IMO, DN list here is not only for authentication, it is also kind of ACL, which means the AC consumer only trusts those DNs which issues AC.

5.280.3.2 `VOMSTrustChain& Arc::VOMSTrustList::AddChain (void)`

Adds empty chain of trusted DNs to list.

5.280.3.3 `RegularExpression& Arc::VOMSTrustList::AddRegex (const VOMSTrustRegex & reg)`

Adds regular expression to list. During verification each signature of AC is checked against all stored regular expressions. DN of signing certificate must match at least one of stored regular expressions.

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

- `VOMSUtil.h`

5.281 Arc::WSAEndpointReference Class Reference

Interface for manipulation of WS-Addressing Endpoint Reference.

```
#include <WSA.h>
```

Public Member Functions

- [WSAEndpointReference \(XMLNode epr\)](#)
- [WSAEndpointReference \(const WSAEndpointReference &wsa\)](#)
- [WSAEndpointReference \(const std::string &address\)](#)
- [WSAEndpointReference \(void\)](#)
- [~WSAEndpointReference \(void\)](#)
- [std::string Address \(void\) const](#)
- [void Address \(const std::string &uri\)](#)
- [WSAEndpointReference & operator= \(const std::string &address\)](#)
- [XMLNode ReferenceParameters \(void\)](#)
- [XMLNode MetaData \(void\)](#)
- [operator XMLNode \(void\)](#)

5.281.1 Detailed Description

Interface for manipulation of WS-Addressing Endpoint Reference.

It works on Endpoint Reference stored in XML tree. No information is stored in this object except reference to corresponding XML subtree.

5.281.2 Constructor & Destructor Documentation

5.281.2.1 Arc::WSAEndpointReference::WSAEndpointReference (XMLNode epr)

Link to top level EPR XML node Linking to existing EPR in XML tree

5.281.2.2 Arc::WSAEndpointReference::WSAEndpointReference (const WSAEndpointReference & wsa)

Copy constructor

5.281.2.3 Arc::WSAEndpointReference::WSAEndpointReference (const std::string & address)

Creating independent EPR - not implemented

5.281.2.4 Arc::WSAEndpointReference::WSAEndpointReference (void)

Dummy constructor - creates invalid instance

5.281.2.5 Arc::WSAEndpointReference::~WSAEndpointReference (void)

Destructor. All empty elements of EPR XML are destroyed here too

5.281.3 Member Function Documentation**5.281.3.1 std::string Arc::WSAEndpointReference::Address (void) const**

Returns Address ([URL](#)) encoded in EPR

5.281.3.2 void Arc::WSAEndpointReference::Address (const std::string & uri)

Assigns new Address value. If EPR had no Address element it is created.

5.281.3.3 XMLNode Arc::WSAEndpointReference::MetaData (void)

Access to MetaData element of EPR. Obtained XML element should be manipulated directly in application-dependent way. If EPR had no MetaData element it is created.

5.281.3.4 Arc::WSAEndpointReference::operator XMLNode (void)

Returns reference to EPR top XML node

5.281.3.5 WSAEndpointReference& Arc::WSAEndpointReference::operator= (const std::string & address)

Same as Address(uri)

5.281.3.6 XMLNode Arc::WSAEndpointReference::ReferenceParameters (void)

Access to ReferenceParameters element of EPR. Obtained XML element should be manipulated directly in application-dependent way. If EPR had no ReferenceParameters element it is created.

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

- WSA.h

5.282 Arc::WSAHeader Class Reference

Interface for manipulation WS-Addressing information in SOAP header.

```
#include <WSA.h>
```

Public Member Functions

- `WSAHeader (SOAPEnvelope &soap)`
- `WSAHeader (const std::string &action)`
- `std::string To (void) const`
- `void To (const std::string &uri)`
- `WSAEndpointReference From (void)`
- `WSAEndpointReference ReplyTo (void)`
- `WSAEndpointReference FaultTo (void)`
- `std::string Action (void) const`
- `void Action (const std::string &uri)`
- `std::string MessageID (void) const`
- `void MessageID (const std::string &uri)`
- `std::string RelatesTo (void) const`
- `void RelatesTo (const std::string &uri)`
- `std::string RelationshipType (void) const`
- `void RelationshipType (const std::string &uri)`
- `XMLNode ReferenceParameter (int n)`
- `XMLNode ReferenceParameter (const std::string &name)`
- `XMLNode NewReferenceParameter (const std::string &name)`
- `operator XMLNode (void)`

Static Public Member Functions

- static bool `Check (SOAPEnvelope &soap)`

Protected Attributes

- bool `header_allocated_`

5.282.1 Detailed Description

Interface for manipulation WS-Addressing information in SOAP header.

It works on Endpoint Reference stored in XML tree. No information is stored in this object except reference to corresponding XML subtree.

5.282.2 Constructor & Destructor Documentation

5.282.2.1 `Arc::WSAHeader::WSAHeader (SOAPEnvelope & soap)`

Linking to a header of existing SOAP message

5.282.2.2 Arc::WSAHeader::WSAHeader (const std::string & *action*)

Creating independent SOAP header - not implemented

5.282.3 Member Function Documentation**5.282.3.1 std::string Arc::WSAHeader::Action (void) const**

Returns content of Action element of SOAP Header.

5.282.3.2 void Arc::WSAHeader::Action (const std::string & *uri*)

Set content of Action element of SOAP Header. If such element does not exist it's created.

5.282.3.3 static bool Arc::WSAHeader::Check (SOAPEnvelope & *soap*) [static]

Tells if specified SOAP message has WSA header

5.282.3.4 WSAEndpointReference Arc::WSAHeader::FaultTo (void)

Returns FaultTo element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulted.

5.282.3.5 WSAEndpointReference Arc::WSAHeader::From (void)

Returns From element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulted.

5.282.3.6 std::string Arc::WSAHeader::MessageID (void) const

Returns content of MessageID element of SOAP Header.

5.282.3.7 void Arc::WSAHeader::MessageID (const std::string & *uri*)

Set content of MessageID element of SOAP Header. If such element does not exist it's created.

5.282.3.8 XMLNode Arc::WSAHeader::NewReferenceParameter (const std::string & *name*)

Creates new ReferenceParameter element with specified name. Returns reference to created element.

5.282.3.9 Arc::WSAHeader::operator XMLNode (void)

Returns reference to SOAP Header - not implemented

5.282.3.10 XMLNode Arc::WSAHeader::ReferenceParameter (const std::string & name)

Returns first ReferenceParameter element with specified name

5.282.3.11 XMLNode Arc::WSAHeader::ReferenceParameter (int n)

Return n-th ReferenceParameter element

5.282.3.12 void Arc::WSAHeader::RelatesTo (const std::string & uri)

Set content of RelatesTo element of SOAP Header. If such element does not exist it's created.

5.282.3.13 std::string Arc::WSAHeader::RelatesTo (void) const

Returns content of RelatesTo element of SOAP Header.

5.282.3.14 void Arc::WSAHeader::RelationshipType (const std::string & uri)

Set content of RelationshipType element of SOAP Header. If such element does not exist it's created.

5.282.3.15 std::string Arc::WSAHeader::RelationshipType (void) const

Returns content of RelationshipType element of SOAP Header.

5.282.3.16 WSAEndpointReference Arc::WSAHeader::ReplyTo (void)

Returns ReplyTo element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulted.

5.282.3.17 std::string Arc::WSAHeader::To (void) const

Returns content of To element of SOAP Header.

5.282.3.18 void Arc::WSAHeader::To (const std::string & uri)

Set content of To element of SOAP Header. If such element does not exist it's created.

5.282.4 Field Documentation

5.282.4.1 `bool Arc::WSAHeader::header_allocated_ [protected]`

SOAP header element

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

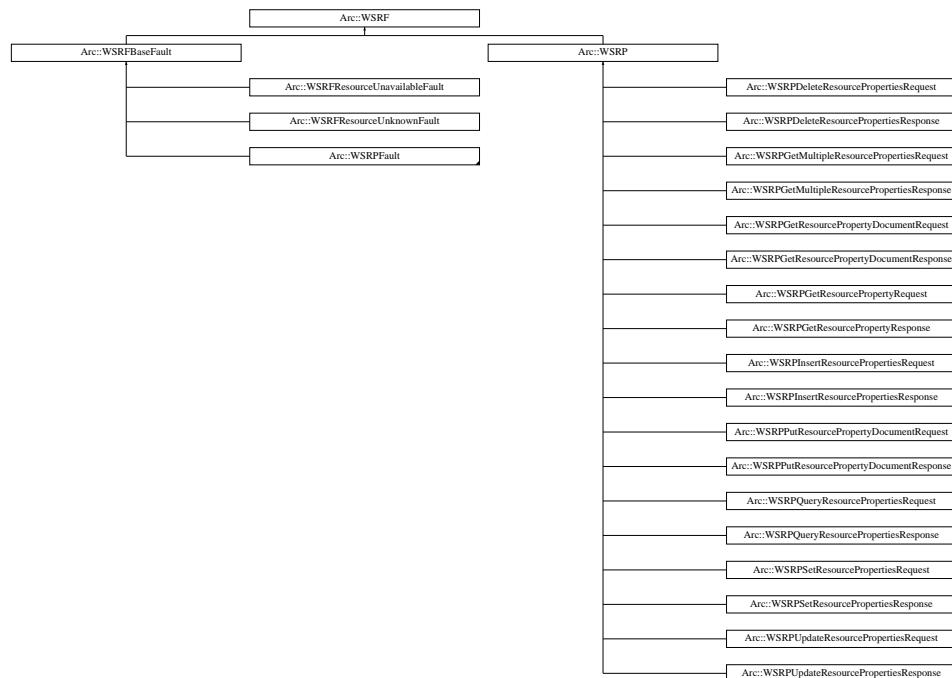
- `WSA.h`

5.283 Arc::WSRF Class Reference

Base class for every [WSRF](#) message.

```
#include <WSRF.h>
```

Inheritance diagram for Arc::WSRF:



Public Member Functions

- [`WSRF`](#) (`SOAPEnvelope &soap, const std::string &action=""`)
- [`WSRF`](#) (`bool fault=false, const std::string &action=""`)
- `virtual SOAPEnvelope & SOAP (void)`
- `virtual operator bool (void)`

Protected Member Functions

- void [set_namespaces](#) (void)

Protected Attributes

- bool [allocated_](#)
- bool [valid_](#)

5.283.1 Detailed Description

Base class for every [WSRF](#) message.

This class is not intended to be used directly. Use it like reference while passing through unknown [WSRF](#) message or use classes derived from it.

5.283.2 Constructor & Destructor Documentation

5.283.2.1 `Arc::WSRF::WSRF (SOAPEnvelope & soap, const std::string & action = " ")`

Constructor - creates object out of supplied SOAP tree.

5.283.2.2 `Arc::WSRF::WSRF (bool fault = false, const std::string & action = " ")`

Constructor - creates new [WSRF](#) object

5.283.3 Member Function Documentation

5.283.3.1 `virtual Arc::WSRF::operator bool (void) [inline, virtual]`

Returns true if instance is valid

References [valid_](#).

5.283.3.2 `void Arc::WSRF::set_namespaces (void) [protected]`

true if object represents valid [WSRF](#) message set WS Resource namespaces and default prefixes in SOAP message

Reimplemented in [Arc::WSRP](#), and [Arc::WSRFBaseFault](#).

5.283.3.3 `virtual SOAPEnvelope& Arc::WSRF::SOAP (void) [inline, virtual]`

Direct access to underlying SOAP element

5.283.4 Field Documentation

5.283.4.1 `bool Arc::WSRF::allocated_ [protected]`

Associated SOAP message - it's SOAP message after all

5.283.4.2 `bool Arc::WSRF::valid_ [protected]`

true if soap_ needs to be deleted in destructor

Referenced by operator `bool()`.

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

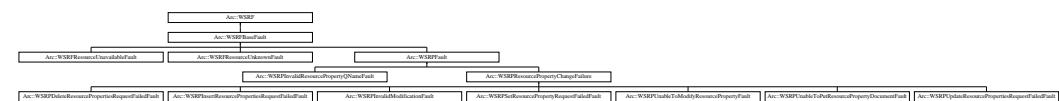
- `WSRF.h`

5.284 Arc::WSRFBaseFault Class Reference

Base class for [WSRF](#) fault messages.

```
#include <WSRFBaseFault.h>
```

Inheritance diagram for `Arc::WSRFBaseFault`:



Public Member Functions

- [WSRFBaseFault](#) (`SOAPEnvelope &soap`)
- [WSRFBaseFault](#) (`const std::string &type`)

Protected Member Functions

- `void set_namespaces (void)`

5.284.1 Detailed Description

Base class for [WSRF](#) fault messages.

Use classes inherited from it for specific faults.

5.284.2 Constructor & Destructor Documentation

5.284.2.1 Arc::WSRFBaseFault::WSRFBaseFault (SOAPEnvelope & soap)

Constructor - creates object out of supplied SOAP tree.

5.284.2.2 Arc::WSRFBaseFault::WSRFBaseFault (const std::string & type)

Constructor - creates new [WSRF](#) fault

5.284.3 Member Function Documentation

5.284.3.1 void Arc::WSRFBaseFault::set_namespaces (void) [protected]

set WS-ResourceProperties namespaces and default prefixes in SOAP message

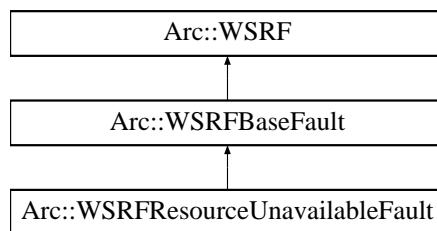
Reimplemented from [Arc::WSRF](#).

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

- WSRFBaseFault.h

5.285 Arc::WSRFResourceUnavailableFault Class Reference

Inheritance diagram for Arc::WSRFResourceUnavailableFault:

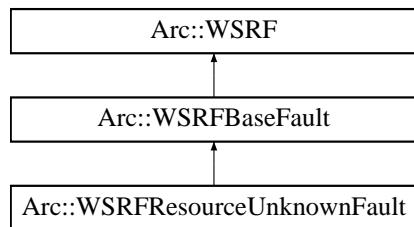


The documentation for this class was generated from the following file:

- WSRFBaseFault.h

5.286 Arc::WSRFResourceUnknownFault Class Reference

Inheritance diagram for Arc::WSRFResourceUnknownFault:



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

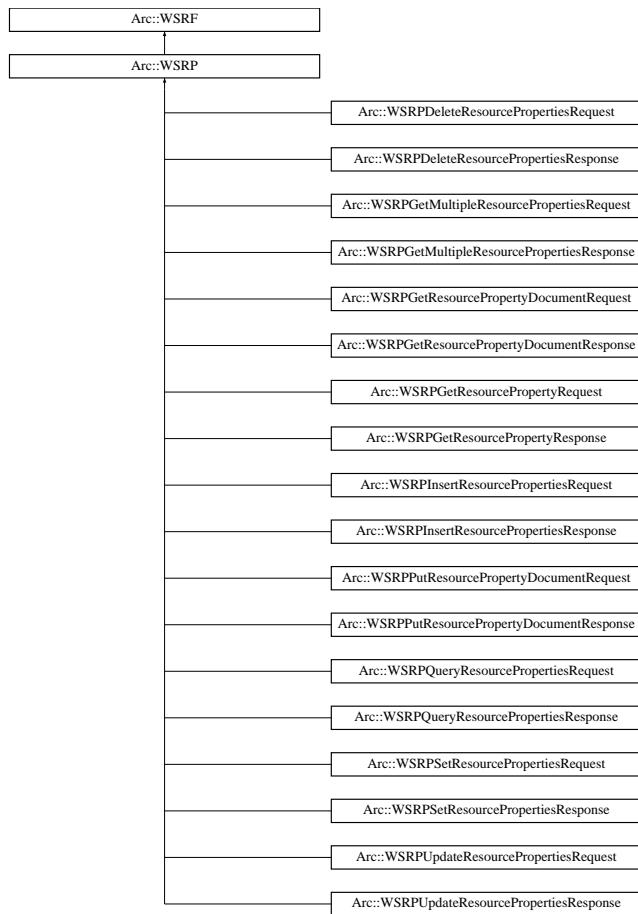
- WSRFBaseFault.h

5.287 Arc::WSRP Class Reference

Base class for WS-ResourceProperties structures.

```
#include <WSResourceProperties.h>
```

Inheritance diagram for Arc::WSRP:



Public Member Functions

- [WSRP](#) (bool fault=false, const std::string &action="")
- [WSRP](#) (SOAPEnvelope &soap, const std::string &action="")

Protected Member Functions

- void [set_namespaces](#) (void)

5.287.1 Detailed Description

Base class for WS-ResourceProperties structures.

Inheriting classes implement specific WS-ResourceProperties messages and their properties/elements. Refer to WS-ResourceProperties specifications for things specific to every message.

5.287.2 Constructor & Destructor Documentation

5.287.2.1 `Arc::WSRP::WSRP (bool fault = false, const std::string & action = " ")`

Constructor - prepares object for creation of new [WSRP](#) request/response/fault

5.287.2.2 `Arc::WSRP::WSRP (SOAPEnvelope & soap, const std::string & action = " ")`

Constructor - creates object out of supplied SOAP tree. It does not check if 'soap' represents valid WS-ResourceProperties structure. Actual check for validity of structure has to be done by derived class.

5.287.3 Member Function Documentation

5.287.3.1 `void Arc::WSRP::set_namespaces (void) [protected]`

set WS-ResourceProperties namespaces and default prefixes in SOAP message

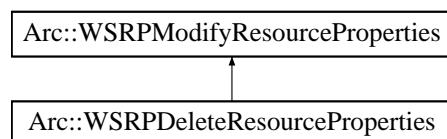
Reimplemented from [Arc::WSRF](#).

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

- `WSResourceProperties.h`

5.288 Arc::WSRPDeleteResourceProperties Class Reference

Inheritance diagram for Arc::WSRPDeleteResourceProperties:

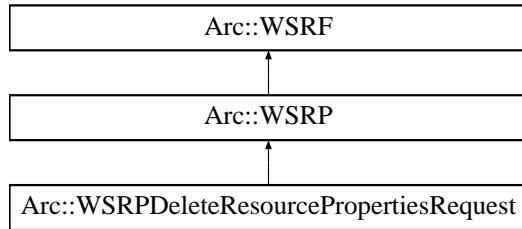


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.289 Arc::WSRPDeleteResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPDeleteResourcePropertiesRequest:

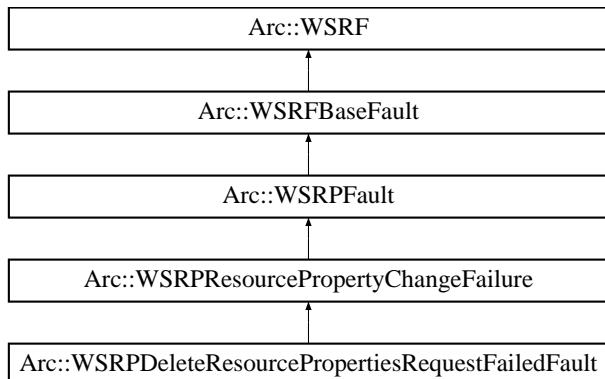


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.290 Arc::WSRPDeleteResourcePropertiesRequestFailedFault Class Reference

Inheritance diagram for Arc::WSRPDeleteResourcePropertiesRequestFailedFault:

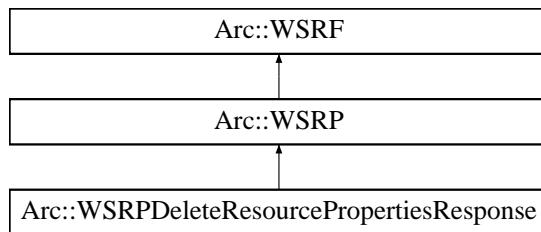


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.291 Arc::WSRPDeleteResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPDeleteResourcePropertiesResponse:



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

- `WSResourceProperties.h`

5.292 Arc::WSRPFault Class Reference

Base class for WS-ResourceProperties faults.

```
#include <WSResourceProperties.h>
```

Inheritance diagram for Arc::WSRPFault:



Public Member Functions

- [WSRPFault](#) (SOAPEnvelope &soap)
- [WSRPFault](#) (const std::string &type)

5.292.1 Detailed Description

Base class for WS-ResourceProperties faults.

5.292.2 Constructor & Destructor Documentation

5.292.2.1 Arc::WSRPFault::WSRPFault (SOAPEnvelope & soap)

Constructor - creates object out of supplied SOAP tree.

5.292.2.2 Arc::WSRPFault::WSRPFault (const std::string & type)

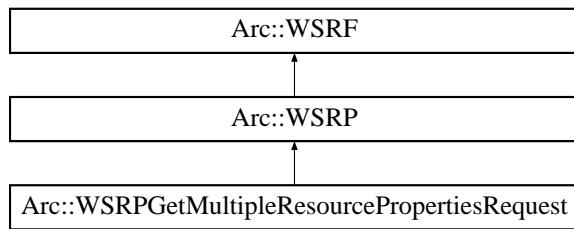
Constructor - creates new [WSRP](#) fault

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

- WSResourceProperties.h

5.293 Arc::WSRPGetMultipleResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPGetMultipleResourcePropertiesRequest:

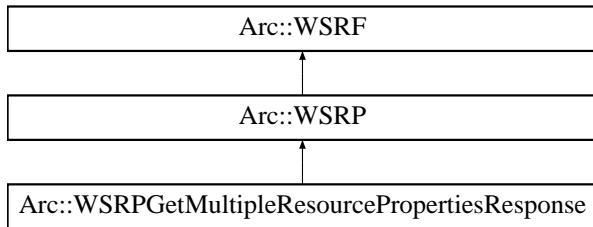


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.294 Arc::WSRPGetMultipleResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPGetMultipleResourcePropertiesResponse:



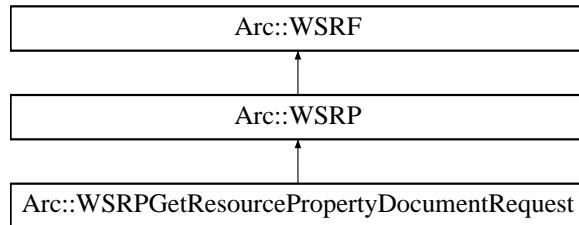
The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.295 Arc::WSRPGetPropertyDocumentRequest Class Reference

Inheritance diagram for Arc::WSRPGetPropertyDocumentRequest:

5.296 Arc::WSRPGetResourcePropertyDocumentResponse Class Reference 445

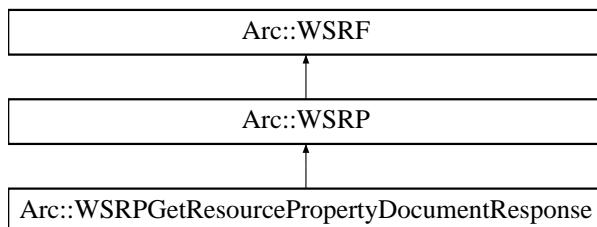


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.296 Arc::WSRPGetResourcePropertyDocumentResponse Class Reference

Inheritance diagram for `Arc::WSRPGetResourcePropertyDocumentResponse`:

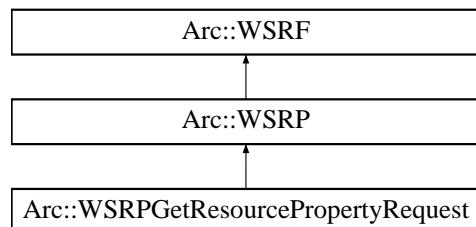


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.297 Arc::WSRPGetResourcePropertyRequest Class Reference

Inheritance diagram for `Arc::WSRPGetResourcePropertyRequest`:

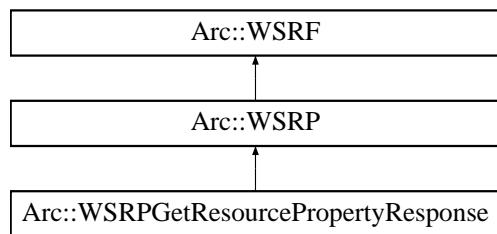


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.298 Arc::WSRPGetResourcePropertyResponse Class Reference

Inheritance diagram for Arc::WSRPGetResourcePropertyResponse:

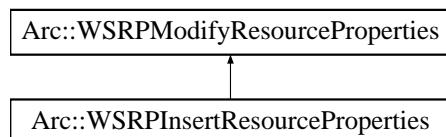


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.299 Arc::WSRPIinsertResourceProperties Class Reference

Inheritance diagram for Arc::WSRPIinsertResourceProperties:



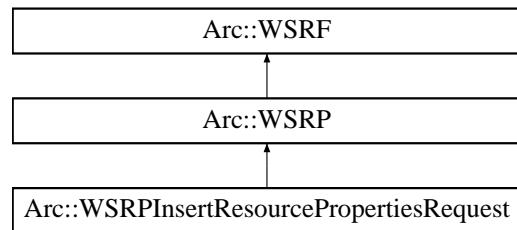
The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.300 Arc::WSRPIinsertResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPIinsertResourcePropertiesRequest:

5.301 Arc::WSRPIinsertResourcePropertiesRequestFailedFault Class Reference

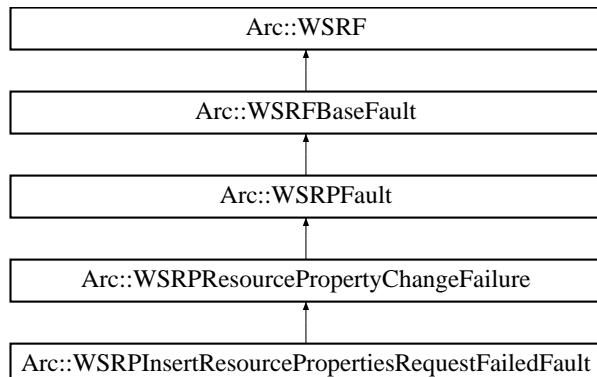


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.301 Arc::WSRPIinsertResourcePropertiesRequestFailedFault Class Reference

Inheritance diagram for `Arc::WSRPIinsertResourcePropertiesRequestFailedFault`:

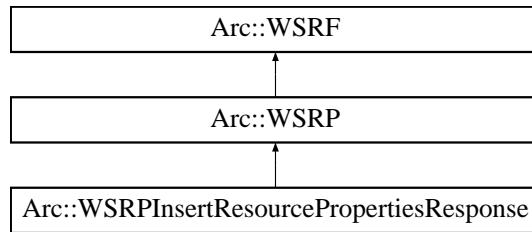


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.302 Arc::WSRPIinsertResourcePropertiesResponse Class Reference

Inheritance diagram for `Arc::WSRPIinsertResourcePropertiesResponse`:

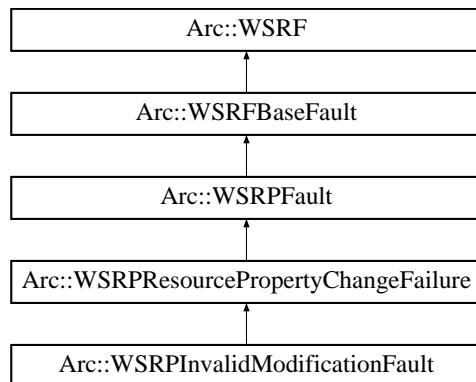


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.303 Arc::WSRPIinvalidModificationFault Class Reference

Inheritance diagram for Arc::WSRPIinvalidModificationFault:

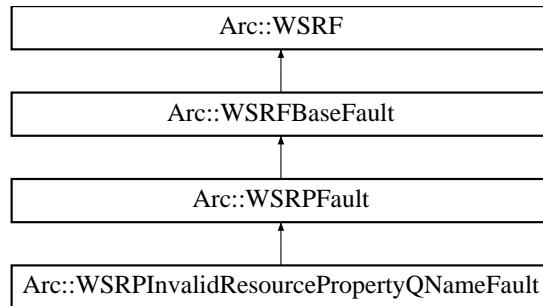


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.304 Arc::WSRPIinvalidResourcePropertyQNameFault Class Reference

Inheritance diagram for Arc::WSRPIinvalidResourcePropertyQNameFault:

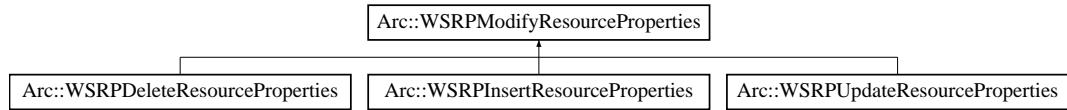


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.305 Arc::WSRPMModifyResourceProperties Class Reference

Inheritance diagram for `Arc::WSRPMModifyResourceProperties`:

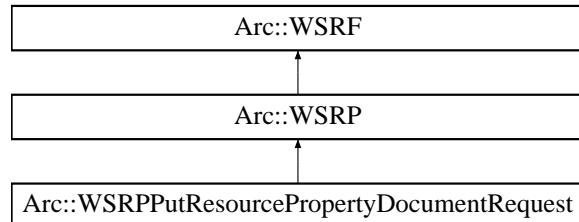


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.306 Arc::WSRPPutResourcePropertyDocumentRequest Class Reference

Inheritance diagram for `Arc::WSRPPutResourcePropertyDocumentRequest`:

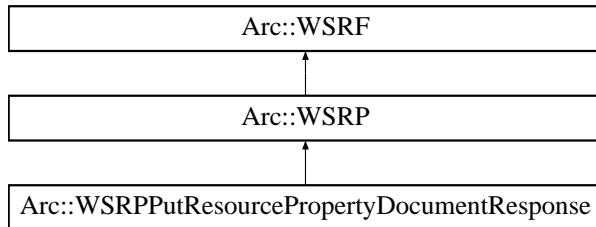


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.307 Arc::WSRPPutResourcePropertyDocumentResponse Class Reference

Inheritance diagram for Arc::WSRPPutResourcePropertyDocumentResponse:

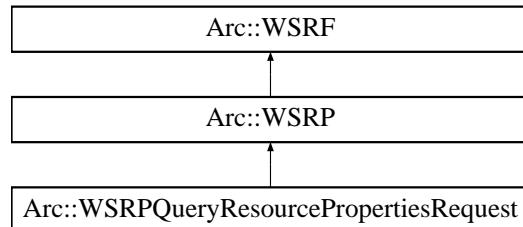


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.308 Arc::WSRPQueryResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPQueryResourcePropertiesRequest:

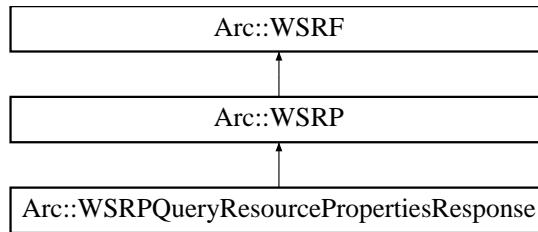


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.309 Arc::WSRPQueryResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPQueryResourcePropertiesResponse:



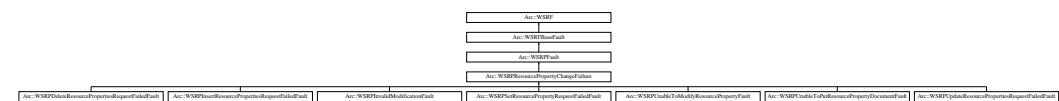
The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.310 Arc::WSRPResourcePropertyChangeFailure Class Reference

```
#include <WSResourceProperties.h>
```

Inheritance diagram for Arc::WSRPResourcePropertyChangeFailure:



Public Member Functions

- `WSRPResourcePropertyChangeFailure (SOAPEnvelope &soap)`
- `WSRPResourcePropertyChangeFailure (const std::string &type)`

5.310.1 Detailed Description

Base class for WS-ResourceProperties faults which contain ResourcePropertyChangeFailure

5.310.2 Constructor & Destructor Documentation

5.310.2.1 Arc::WSRPResourcePropertyChangeFailure::WSRPResourcePropertyChangeFailure (SOAPEnvelope & soap) [inline]

Constructor - creates object out of supplied SOAP tree.

5.310.2.2 Arc::WSRPResourcePropertyChangeFailure::WSRPResourcePropertyChangeFailure (const std::string & type) [inline]

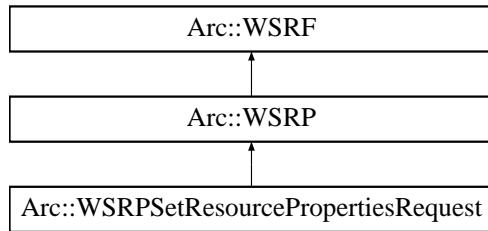
Constructor - creates new **WSRP** fault

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

- WSResourceProperties.h

5.311 Arc::WSRPSetResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPSetResourcePropertiesRequest:

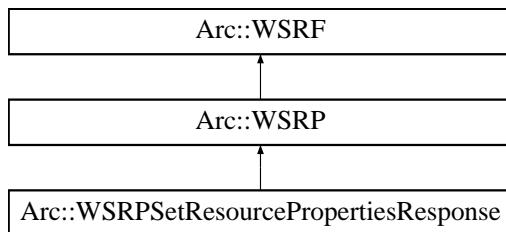


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.312 Arc::WSRPSetResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPSetResourcePropertiesResponse:

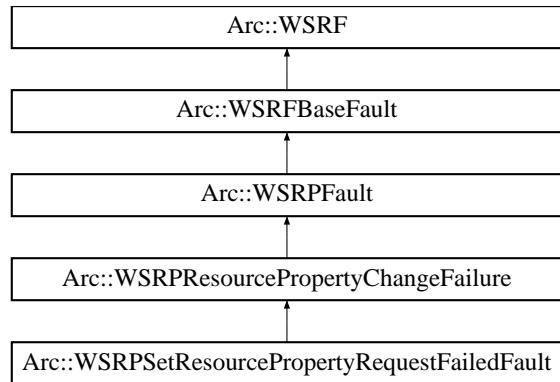


The documentation for this class was generated from the following file:

- WSResourceProperties.h

5.313 Arc::WSRPSetResourcePropertyRequestFailedFault Class Reference

Inheritance diagram for Arc::WSRPSetResourcePropertyRequestFailedFault:

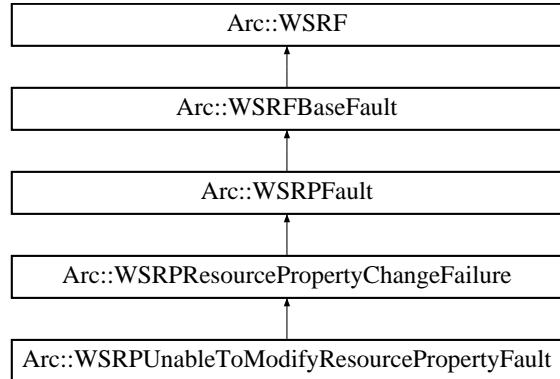


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.314 Arc::WSRPUnableToModifyResourcePropertyFault Class Reference

Inheritance diagram for `Arc::WSRPUnableToModifyResourcePropertyFault`:

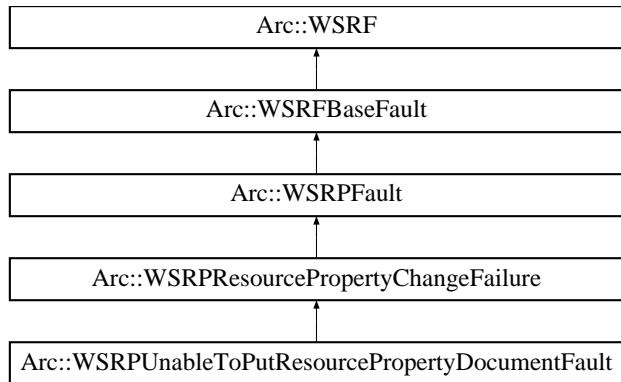


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.315 Arc::WSRPUnableToPutResourcePropertyDocumentFault Class Reference

Inheritance diagram for `Arc::WSRPUnableToPutResourcePropertyDocumentFault`:

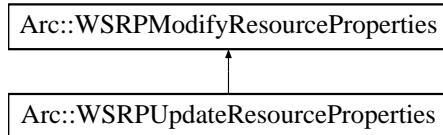


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.316 Arc::WSRPUpdateResourceProperties Class Reference

Inheritance diagram for `Arc::WSRPUpdateResourceProperties`:

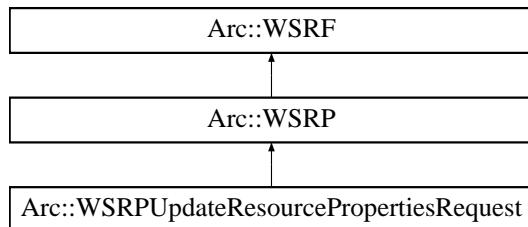


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.317 Arc::WSRPUpdateResourcePropertiesRequest Class Reference

Inheritance diagram for `Arc::WSRPUpdateResourcePropertiesRequest`:



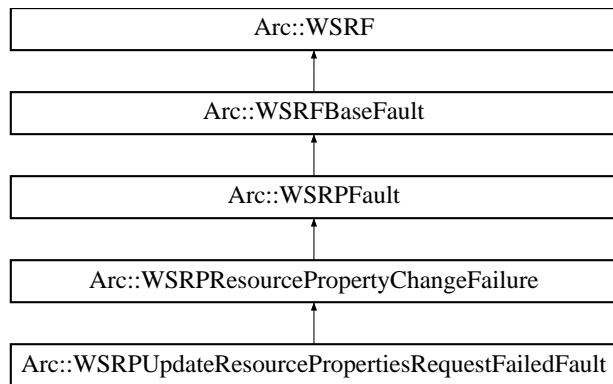
5.318 Arc::WSRPUpdateResourcePropertiesRequestFailedFault Class Reference

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

- `WSResourceProperties.h`

5.318 Arc::WSRPUpdateResourcePropertiesRequestFailedFault Class Reference

Inheritance diagram for `Arc::WSRPUpdateResourcePropertiesRequestFailedFault`:

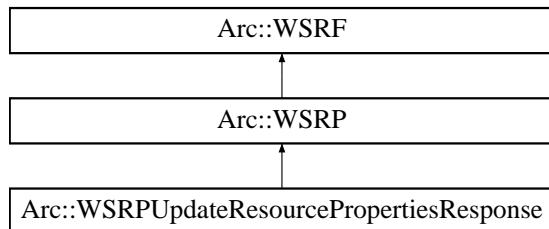


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.319 Arc::WSRPUpdateResourcePropertiesResponse Class Reference

Inheritance diagram for `Arc::WSRPUpdateResourcePropertiesResponse`:

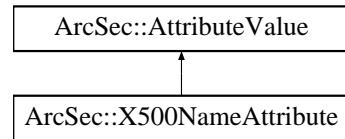


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

5.320 ArcSec::X500NameAttribute Class Reference

Inheritance diagram for ArcSec::X500NameAttribute:



Public Member Functions

- virtual bool [equal \(AttributeValue *other, bool check_id=true\)](#)
- virtual std::string [encode \(\)](#)
- virtual std::string [getType \(\)](#)
- virtual std::string [getId \(\)](#)

5.320.1 Member Function Documentation

5.320.1.1 virtual std::string [ArcSec::X500NameAttribute::encode \(\) \[inline, virtual\]](#)

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

5.320.1.2 virtual bool [ArcSec::X500NameAttribute::equal \(AttributeValue * value, bool check_id = true \) \[virtual\]](#)

Evluate whether "this" equale to the parameter value

Implements [ArcSec::AttributeValue](#).

5.320.1.3 virtual std::string [ArcSec::X500NameAttribute::getId \(\) \[inline, virtual\]](#)

Get the Attributeld of the <Attribute>

Implements [ArcSec::AttributeValue](#).

5.320.1.4 virtual std::string [ArcSec::X500NameAttribute::getType \(\) \[inline, virtual\]](#)

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

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

- X500NameAttribute.h

5.321 Arc::X509Token Class Reference

Class for manipulating X.509 Token [Profile](#).

```
#include <X509Token.h>
```

Public Types

- enum [X509TokenType](#)

Public Member Functions

- [X509Token](#) (SOAPEnvelope &soap, const std::string &keyfile="")
- [X509Token](#) (SOAPEnvelope &soap, const std::string &certfile, const std::string &keyfile, [X509TokenType](#) token_type=Signature)
- [~X509Token](#) (void)
- [operator bool](#) (void)
- bool [Authenticate](#) (const std::string &cafile, const std::string &capath)
- bool [Authenticate](#) (void)

5.321.1 Detailed Description

Class for manipulating X.509 Token [Profile](#).

This class is for generating/consuming X.509 Token profile. Currently it is used by x509token handler (src/hed/pdc/x509tokensh/) It is not necessary to directly called this class. If we need to use X.509 Token functionality, we only need to configure the x509token handler into service and client.

5.321.2 Member Enumeration Documentation

5.321.2.1 enum Arc::X509Token::X509TokenType

X509TokenType is for distinguishing two types of operation. It is used as the parameter of constructor.

5.321.3 Constructor & Destructor Documentation

5.321.3.1 `Arc::X509Token::X509Token (SOAPEnvelope & soap, const std::string & keyfile = " ")`

Constructor.Parse X509 Token information from SOAP header. X509 Token related information is extracted from SOAP header and stored in class variables. And then it the `X509Token` object will be used for authentication if the tokentype is Signature; otherwise if the tokentype is Encryption, the encrypted soap body will be decrypted and replaced by decrypted message. keyfile is only needed when the `X509Token` is encryption token

5.321.3.2 `Arc::X509Token::X509Token (SOAPEnvelope & soap, const std::string & certfile, const std::string & keyfile, X509TokenType token_type = Signature)`

Constructor. Add X509 Token information into the SOAP header. Generated token contains elements X509 token and signature, and is meant to be used for authentication on the consuming side.

Parameters

<code>soap</code>	The SOAP message to which the X509 Token will be inserted
<code>certfile</code>	The certificate file which will be used to encrypt the SOAP body (if parameter tokentype is Encryption), or be used as <wsse:BinarySecurityToken/> (if parameter tokentype is Signature).
<code>keyfile</code>	The key file which will be used to create signature. Not needed when create encryption.
<code>tokentype</code>	Token type: Signature or Encryption.

5.321.3.3 `Arc::X509Token::~X509Token (void)`

Deconstructor. Nothing to be done except finalizing the xmlsec library.

5.321.4 Member Function Documentation

5.321.4.1 `bool Arc::X509Token::Authenticate (const std::string & cafile, const std::string & capath)`

Check signature by using the certificare information in `X509Token` which is parsed by the constructor, and the trusted certificates specified as one of the two parameters. Not only the signature (in the `X509Token`) itself is checked, but also the certificate which is supposed to check the signature needs to be trused (which means the certificate is issued by the ca certificate from CA file or CA directory). At least one the the two parameters should be set.

Parameters

<code>cafile</code>	The CA file
<code>capath</code>	The CA directory

Returns

true if authentication passes; otherwise false

5.321.4.2 bool Arc::X509Token::Authenticate (void)

Check signature by using the cert information in soap message. Only the signature itself is checked, and it is not guaranteed that the certificate which is supposed to check the signature is trusted.

5.321.4.3 Arc::X509Token::operator bool (void)

Returns true if constructor succeeded

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

- X509Token.h

5.322 Arc::XmlContainer Class Reference

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

- XmlContainer.h

5.323 Arc::XmlDatabase Class Reference

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

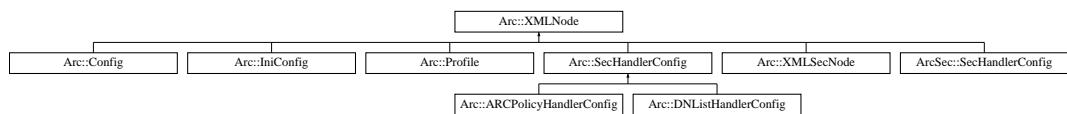
- XmlDatabase.h

5.324 Arc::XmlNode Class Reference

Wrapper for LibXML library Tree interface.

```
#include <XmlNode.h>
```

Inheritance diagram for Arc::XmlNode:



Public Member Functions

- `XMLNode (void)`
- `XMLNode (const XMLNode &node)`
- `XMLNode (const std::string &xml)`
- `XMLNode (const char *xml, int len=-1)`
- `XMLNode (long ptr_addr)`
- `XMLNode (const NS &ns, const char *name)`
- `~XMLNode (void)`
- `void New (XMLNode &node) const`
- `void Exchange (XMLNode &node)`
- `void Move (XMLNode &node)`
- `void Swap (XMLNode &node)`
- `operator bool (void) const`
- `bool operator! (void) const`
- `bool operator==(const XMLNode &node)`
- `bool operator!= (const XMLNode &node)`
- `bool Same (const XMLNode &node)`
- `bool operator==(bool val)`
- `bool operator!=(bool val)`
- `bool operator==(const std::string &str)`
- `bool operator!= (const std::string &str)`
- `bool operator==(const char *str)`
- `bool operator!= (const char *str)`
- `XMLNode Child (int n=0)`
- `XMLNode operator[] (const char *name) const`
- `XMLNode operator[] (const std::string &name) const`
- `XMLNode operator[] (int n) const`
- `void operator++ (void)`
- `void operator-- (void)`
- `int Size (void) const`
- `XMLNode Get (const std::string &name) const`
- `std::string Name (void) const`
- `std::string Prefix (void) const`
- `std::string FullName (void) const`
- `std::string Namespace (void) const`
- `void Name (const char *name)`
- `void Name (const std::string &name)`
- `void GetXML (std::string &out_xml_str, bool user_friendly=false) const`
- `void GetXML (std::string &out_xml_str, const std::string &encoding, bool user_friendly=false) const`
- `void GetDoc (std::string &out_xml_str, bool user_friendly=false) const`
- `operator std::string (void) const`
- `XMLNode & operator= (const char *content)`
- `XMLNode & operator= (const std::string &content)`
- `void Set (const std::string &content)`
- `XMLNode & operator= (const XMLNode &node)`

- `XMLNode Attribute (int n=0)`
- `XMLNode Attribute (const char *name)`
- `XMLNode Attribute (const std::string &name)`
- `XMLNode NewAttribute (const char *name)`
- `XMLNode NewAttribute (const std::string &name)`
- `int AttributesSize (void) const`
- `void Namespaces (const NS &namespaces, bool keep=false, int recursion=-1)`
- `NS Namespaces (void)`
- `std::string NamespacePrefix (const char *urn)`
- `XMLNode NewChild (const char *name, int n=-1, bool global_order=false)`
- `XMLNode NewChild (const std::string &name, int n=-1, bool global_order=false)`
- `XMLNode NewChild (const char *name, const NS &namespaces, int n=-1, bool global_order=false)`
- `XMLNode NewChild (const std::string &name, const NS &namespaces, int n=-1, bool global_order=false)`
- `XMLNode NewChild (const XMLNode &node, int n=-1, bool global_order=false)`
- `void Replace (const XMLNode &node)`
- `void Destroy (void)`
- `XMLNodeList Path (const std::string &path)`
- `XMLNodeList XPathLookup (const std::string &xpathExpr, const NS &nsList)`
- `XMLNode GetRoot (void)`
- `XMLNode Parent (void)`
- `bool SaveToFile (const std::string &file_name) const`
- `bool SaveToStream (std::ostream &out) const`
- `bool ReadFromFile (const std::string &file_name)`
- `bool ReadFromStream (std::istream &in)`
- `bool Validate (const std::string &schema_file, std::string &err_msg)`

Protected Member Functions

- `XMLNode (xmlNodePtr node)`

Protected Attributes

- `bool is_owner_`
- `bool is_temporary_`

Friends

- `bool MatchXMLName (const XMLNode &node1, const XMLNode &node2)`
- `bool MatchXMLName (const XMLNode &node, const char *name)`
- `bool MatchXMLName (const XMLNode &node, const std::string &name)`
- `bool MatchXMLNamespace (const XMLNode &node1, const XMLNode &node2)`
- `bool MatchXMLNamespace (const XMLNode &node, const char *uri)`
- `bool MatchXMLNamespace (const XMLNode &node, const std::string &uri)`

5.324.1 Detailed Description

Wrapper for LibXML library Tree interface.

This class wraps XML Node, Document and Property/Attribute structures. Each instance serves as pointer to actual LibXML element and provides convenient (for chosen purpose) methods for manipulating it. This class has no special ties to LibXML library and may be easily rewritten for any XML parser which provides interface similar to LibXML Tree. It implements only small subset of XML capabilities, which is probably enough for performing most of useful actions. This class also filters out (usually) useless textual nodes which are often used to make XML documents human-readable.

5.324.2 Constructor & Destructor Documentation

5.324.2.1 `Arc::XMLNode::XMLNode(xmlNodePtr node) [inline, protected]`

Private constructor for inherited classes Creates instance and links to existing LibXML structure. Acquired structure is not owned by class instance. If there is need to completely pass control of LibXML document to then instance's `is_owner_` variable has to be set to true.

5.324.2.2 `Arc::XMLNode::XMLNode(void) [inline]`

Constructor of invalid node Created instance does not point to XML element. All methods are still allowed for such instance but produce no results.

5.324.2.3 `Arc::XMLNode::XMLNode(const XMLNode & node) [inline]`

Copies existing instance. Underlying XML element is NOT copied. Ownership is NOT inherited. Strictly speaking it shuld be no const here - but that conflicts with C++.

5.324.2.4 `Arc::XMLNode::XMLNode(const std::string & xml)`

Creates XML document structure from textual representation of XML document. Created structure is pointed and owned by constructed instance

5.324.2.5 `Arc::XMLNode::XMLNode(const char * xml, int len = -1)`

Same as previous

5.324.2.6 `Arc::XMLNode::XMLNode(long ptr_addr)`

Copy constructor. Used by language bindigs

5.324.2.7 Arc::XMLNode::XMLNode (const NS & ns, const char * name)

Creates empty XML document structure with specified namespaces. Created XML contains only root element named 'name'. Created structure is pointed and owned by constructed instance

5.324.2.8 Arc::XMLNode::~XMLNode (void)

Destructor Also destroys underlying XML document if owned by this instance

5.324.3 Member Function Documentation

5.324.3.1 XMLNode Arc::XMLNode::Attribute (int n = 0)

Returns list of all attributes of node.

Returns [XMLNode](#) instance reresenting n-th attribute of node.

Referenced by Attribute().

5.324.3.2 XMLNode Arc::XMLNode::Attribute (const char * name)

Returns [XMLNode](#) instance representing first attribute of node with specified by name

5.324.3.3 XMLNode Arc::XMLNode::Attribute (const std::string & name) [inline]

Returns [XMLNode](#) instance representing first attribute of node with specified by name

References Attribute().

5.324.3.4 int Arc::XMLNode::AttributesSize (void) const

Returns number of attributes of node

5.324.3.5 XMLNode Arc::XMLNode::Child (int n = 0)

Returns [XMLNode](#) instance representing n-th child of XML element. If such does not exist invalid [XMLNode](#) instance is returned

5.324.3.6 void Arc::XMLNode::Destroy (void)

Destroys underlying XML element. XML element is unlinked from XML tree and destroyed. After this operation [XMLNode](#) instance becomes invalid

5.324.3.7 void Arc::XMLNode::Exchange (XMLNode & node)

Exchanges XML (sub)trees. Following combinations are possible If either this or node are referring owned XML tree (top level node) then references are simply exchanged. This operation is fast. If both this and node are referring to XML (sub)tree of different documents then (sub)trees are exchanged between documents. If both this and node are referring to XML (sub)tree of same document then (sub)trees are moved inside document. The main reason for this method is to provide effective way to insert one XML document inside another. One should take into account that if any of exchanged nodes is top level it must be also owner of document. Otherwise method will fail. If both nodes are top level owners and/or invalid nodes then this method is identical to [Swap\(\)](#).

5.324.3.8 std::string Arc::XMLNode::FullName (void) const [inline]

Returns prefix:name of XML node

References Name(), and Prefix().

5.324.3.9 XMLNode Arc::XMLNode::Get (const std::string & name) const [inline]

Same as operator[]

References operator[]().

5.324.3.10 void Arc::XMLNode::GetDoc (std::string & out_xml_str, bool user_friendly = false) const

Fills argument with whole XML document textual representation

5.324.3.11 XMLNode Arc::XMLNode::GetRoot (void)

Get the root node from any child node of the tree

5.324.3.12 void Arc::XMLNode::GetXML (std::string & out_xml_str, bool user_friendly = false) const

Fills argument with this instance XML subtree textual representation

5.324.3.13 void Arc::XMLNode::GetXML (std::string & out_xml_str, const std::string & encoding, bool user_friendly = false) const

Fills argument with this instance XML subtree textual representation if the XML subtree is corresponding to the encoding format specified in the argument, e.g. utf-8

5.324.3.14 void Arc::XMLNode::Move (XMLNode & node)

Moves content of this XML (sub)tree to node This opearation is similar to New except that XML (sub)tree to refered by this is destroyed. This method is more effective than combination of [New\(\)](#) and [Destroy\(\)](#) because internally it is optimized not to copy data if not needed. The main purpose of this is to effectively extract part of XML document.

5.324.3.15 std::string Arc::XMLNode::Name (void) const

Returns name of XML node

Referenced by [FullName\(\)](#), and [Name\(\)](#).

5.324.3.16 void Arc::XMLNode::Name (const std::string & name) [inline]

Assigns new name to XML node

References [Name\(\)](#).

5.324.3.17 void Arc::XMLNode::Name (const char * name)

Assigns new name to XML node

5.324.3.18 std::string Arc::XMLNode::Namespace (void) const

Returns namespace URI of XML node

5.324.3.19 std::string Arc::XMLNode::NamespacePrefix (const char * urn)

Returns prefix of specified namespace. Empty string if no such namespace.

5.324.3.20 NS Arc::XMLNode::Namespaces (void)

Returns namespaces known at this node

5.324.3.21 void Arc::XMLNode::Namespaces (const NS & namespaces, bool keep = false, int recursion = -1)

Assigns namespaces of XML document at point specified by this instance. If namespace already exists it gets new prefix. New namespaces are added. It is useful to apply this method to XML being processed in order to refer to it's elements by known prefix. If keep is set to false existing namespace definition residing at this instance and below are removed (default behavior). If recursion is set to positive number then depth of prefix replacement is limited by this number (0 limits it to this node only). For unlimited recursion

use -1. If recursion is limited then value of keep is ignored and existing namespaces are always kept.

5.324.3.22 void Arc::XMLNode::New (XMLNode & node) const

Creates a copy of XML (sub)tree. If object does not represent whole document - top level document is created. 'new_node' becomes a pointer owning new XML document.

5.324.3.23 XMLNode Arc::XMLNode::NewAttribute (const char * name)

Creates new attribute with specified name.

Referenced by NewAttribute().

**5.324.3.24 XMLNode Arc::XMLNode::NewAttribute (const std::string & name)
[inline]**

Creates new attribute with specified name.

References NewAttribute().

**5.324.3.25 XMLNode Arc::XMLNode::NewChild (const char * name, int n = -1, bool
global_order = false)**

Creates new child XML element at specified position with specified name. Default is to put it at end of list. If global order is true position applies to whole set of children, otherwise only to children of same name. Returns created node.

Referenced by NewChild().

**5.324.3.26 XMLNode Arc::XMLNode::NewChild (const std::string & name, int n = -1, bool
global_order = false) [inline]**

Same as [NewChild\(const char*,int,bool\)](#)

References NewChild().

**5.324.3.27 XMLNode Arc::XMLNode::NewChild (const char * name, const NS &
namespaces, int n = -1, bool global_order = false)**

Creates new child XML element at specified position with specified name and namespaces. For more information look at [NewChild\(const char*,int,bool\)](#)

5.324.3.28 **XMLNode Arc::XMLNode::NewChild (const std::string & name, const NS & namespaces, int n = -1, bool global_order = false) [inline]**

Same as [NewChild\(const char*,const NS&,int,bool\)](#)

References [NewChild\(\)](#).

5.324.3.29 **XMLNode Arc::XMLNode::NewChild (const XMLNode & node, int n = -1, bool global_order = false)**

Link a copy of supplied XML node as child. Returns instance refering to new child. XML element is a copy of supplied one but not owned by returned instance

5.324.3.30 **Arc::XMLNode::operator bool (void) const [inline]**

Returns true if instance points to XML element - valid instance

References [is_temporary_](#).

5.324.3.31 **Arc::XMLNode::operator std::string (void) const**

Returns textual content of node excluding content of children nodes

5.324.3.32 **bool Arc::XMLNode::operator! (void) const [inline]**

Returns true if instance does not point to XML element - invalid instance

References [is_temporary_](#).

5.324.3.33 **bool Arc::XMLNode::operator!= (const XMLNode & node) [inline]**

Returns false if 'node' represents same XML element

5.324.3.34 **bool Arc::XMLNode::operator!= (bool val) [inline]**

This operator is needed to avoid ambiguity

5.324.3.35 **bool Arc::XMLNode::operator!= (const std::string & str) [inline]**

This operator is needed to avoid ambiguity

5.324.3.36 **bool Arc::XMLNode::operator!= (const char * str) [inline]**

This operator is needed to avoid ambiguity

5.324.3.37 void Arc::XMLNode::operator++ (void)

Convenience operator to switch to next element of same name. If there is no such node this object becomes invalid.

5.324.3.38 void Arc::XMLNode::operator-- (void)

Convenience operator to switch to previous element of same name. If there is no such node this object becomes invalid.

5.324.3.39 XMLNode& Arc::XMLNode::operator= (const char * content)

Sets textual content of node. All existing children nodes are discarded.

Referenced by operator=(), and Set().

5.324.3.40 XMLNode& Arc::XMLNode::operator= (const XMLNode & node)

Make instance refer to another XML node. Ownership is not inherited. Due to nature of **XMLNode** there should be no const here, but that does not fit into C++.

**5.324.3.41 XMLNode& Arc::XMLNode::operator= (const std::string & content)
[inline]**

Sets textual content of node. All existing children nodes are discarded.

References operator=().

5.324.3.42 bool Arc::XMLNode::operator== (bool val) [inline]

This operator is needed to avoid ambiguity

5.324.3.43 bool Arc::XMLNode::operator== (const XMLNode & node) [inline]

Returns true if 'node' represents same XML element

Referenced by Same().

5.324.3.44 bool Arc::XMLNode::operator== (const char * str) [inline]

This operator is needed to avoid ambiguity

5.324.3.45 bool Arc::XMLNode::operator== (const std::string & str) [inline]

This operator is needed to avoid ambiguity

5.324.3.46 XMLNode Arc::XMLNode::operator[] (const char * *name*) const

Returns [XMLNode](#) instance representing first child element with specified name. Name may be "namespace_prefix:name", "namespace_uri:name" or simply "name". In last case namespace is ignored. If such node does not exist invalid [XMLNode](#) instance is returned. This method should not be marked const because obtaining unrestricted [XMLNode](#) of child element allows modification of underlying XML tree. But in order to keep const in other places non-const-handling is passed to programmer. Otherwise C++ compiler goes nuts.

Referenced by Get(), and operator[]().

**5.324.3.47 XMLNode Arc::XMLNode::operator[] (const std::string & *name*) const
[inline]**

Similar to previous method

References operator[]().

5.324.3.48 XMLNode Arc::XMLNode::operator[] (int *n*) const

Returns [XMLNode](#) instance representing n-th node in sequence of siblings of same name. It's main purpose is to be used to retrieve element in array of children of same name like node["name"][5]. This method should not be marked const because obtaining unrestricted [XMLNode](#) of child element allows modification of underlying XML tree. But in order to keep const in other places non-const-handling is passed to programmer. Otherwise C++ compiler goes nuts.

5.324.3.49 XMLNode Arc::XMLNode::Parent (void)

Get the parent node from any child node of the tree

5.324.3.50 XMLNodeList Arc::XMLNode::Path (const std::string & *path*)

Collects nodes corresponding to specified path. This is a convenience function to cover common use of XPath but without performance hit. Path is made of node_name[/node_name[...]] and is relative to current node. node_names are treated in same way as in operator[]. Returns all nodes which are represented by path.

5.324.3.51 std::string Arc::XMLNode::Prefix (void) const

Returns namespace prefix of XML node

Referenced by FullName().

5.324.3.52 `bool Arc::XMLNode::ReadFromFile (const std::string & file_name)`

Read XML document from file and associate it with this node

5.324.3.53 `bool Arc::XMLNode::ReadFromStream (std::istream & in)`

Read XML document from stream and associate it with this node

5.324.3.54 `void Arc::XMLNode::Replace (const XMLNode & node)`

Makes a copy of supplied XML node and makes this instance refere to it

5.324.3.55 `bool Arc::XMLNode::Same (const XMLNode & node) [inline]`

Returns true if 'node' represents same XML element - for bindings

References operator==().

5.324.3.56 `bool Arc::XMLNode::SaveToFile (const std::string & file_name) const`

Save string representation of node to file

5.324.3.57 `bool Arc::XMLNode::SaveToStream (std::ostream & out) const`

Save string representation of node to stream

5.324.3.58 `void Arc::XMLNode::Set (const std::string & content) [inline]`

Same as operator=. Used for bindings.

References operator=().

5.324.3.59 `int Arc::XMLNode::Size (void) const`

Returns number of children nodes

5.324.3.60 `void Arc::XMLNode::Swap (XMLNode & node)`

Swaps XML (sub)trees to this this and node refer. For XML subtrees this method is not anyhow different then using combination `XMLNode tmp=*`this; `*this=node;` `node=tmp;` But in case of either this or node owning XML document ownership is swapped too. And this is a main purpose of [Swap\(\)](#) method.

```
5.324.3.61 bool Arc::XMLNode::Validate ( const std::string & schema_file, std::string & err_msg )
```

Remove all eye-candy information leaving only informational parts * void Purify(void);
XML schema validation against the schema file defined as argument

```
5.324.3.62 XMLNodeList Arc::XMLNode::XPathLookup ( const std::string & xpathExpr, const NS & nsList )
```

Uses xPath to look up the whole xml structure, Returns a list of [XMLNode](#) points. The xpathExpr should be like "//xx:child1/" which indicates the namespace and node that you would like to find; The nsList is the namespace the result should belong to (e.g. xx="uri:test"). [Query](#) is run on whole XML document but only the elements belonging to this XML subtree are returned.

5.324.4 Friends And Related Function Documentation

```
5.324.4.1 bool MatchXMLName ( const XMLNode & node1, const XMLNode & node2 ) [friend]
```

Returns true if underlying XML elements have same names

```
5.324.4.2 bool MatchXMLName ( const XMLNode & node, const std::string & name ) [friend]
```

Returns true if 'name' matches name of 'node'. If name contains prefix it's checked too

```
5.324.4.3 bool MatchXMLName ( const XMLNode & node, const char * name ) [friend]
```

Returns true if 'name' matches name of 'node'. If name contains prefix it's checked too

```
5.324.4.4 bool MatchXMLNamespace ( const XMLNode & node1, const XMLNode & node2 ) [friend]
```

Returns true if underlying XML elements belong to same namespaces

```
5.324.4.5 bool MatchXMLNamespace ( const XMLNode & node, const std::string & uri ) [friend]
```

Returns true if 'namespace' matches 'node's namespace.

5.324.4.6 `bool MatchXMLNamespace (const XMLNode & node, const char * uri)`
[friend]

Returns true if 'namespace' matches 'node's namespace.

5.324.5 Field Documentation

5.324.5.1 `bool Arc::XMLNode::is_owner_ [protected]`

If true node is owned by this instance - hence released in destructor. Normally that may be true only for top level node of XML document.

5.324.5.2 `bool Arc::XMLNode::is_temporary_ [protected]`

This variable is for future

Referenced by operator `bool()`, and operator `!()`.

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

- `XMLNode.h`

5.325 Arc::XMLNodeContainer Class Reference

```
#include <XMLNode.h>
```

Public Member Functions

- `XMLNodeContainer (void)`
- `XMLNodeContainer (const XMLNodeContainer &)`
- `XMLNodeContainer & operator= (const XMLNodeContainer &)`
- `void Add (const XMLNode &)`
- `void Add (const std::list< XMLNode > &)`
- `void AddNew (const XMLNode &)`
- `void AddNew (const std::list< XMLNode > &)`
- `int Size (void) const`
- `XMLNode operator[] (int)`
- `std::list< XMLNode > Nodes (void)`

5.325.1 Detailed Description

Container for multiple `XMLNode` elements

5.325.2 Constructor & Destructor Documentation

5.325.2.1 `Arc::XMLNodeContainer::XMLNodeContainer (void)`

Default constructor

5.325.2.2 `Arc::XMLNodeContainer::XMLNodeContainer (const XMLNodeContainer &)`

Copy constructor. Add nodes from argument. Nodes owning XML document are copied using [AddNew\(\)](#). Not owning nodes are linked using [Add\(\)](#) method.

5.325.3 Member Function Documentation

5.325.3.1 `void Arc::XMLNodeContainer::Add (const XMLNode &)`

Link XML subtree refered by node to container. XML tree must be available as long as this object is used.

5.325.3.2 `void Arc::XMLNodeContainer::Add (const std::list< XMLNode > &)`

Link multiple XML subtrees to container.

5.325.3.3 `void Arc::XMLNodeContainer::AddNew (const XMLNode &)`

Copy XML subtree referenced by node to container. After this operation container refers to independent XML document. This document is deleted when container is destroyed.

5.325.3.4 `void Arc::XMLNodeContainer::AddNew (const std::list< XMLNode > &)`

Copy multiple XML subtrees to container.

5.325.3.5 `std::list<XMLNode> Arc::XMLNodeContainer::Nodes (void)`

Returns all stored nodes.

5.325.3.6 `XMLNodeContainer& Arc::XMLNodeContainer::operator= (const XMLNodeContainer &)`

Same as copy constructor with current nodes being deleted first.

5.325.3.7 `XMLNode Arc::XMLNodeContainer::operator[](int)`

Returns n-th node in a store.

5.325.3.8 `int Arc::XMLNodeContainer::Size(void) const`

Return number of referred/stored nodes.

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

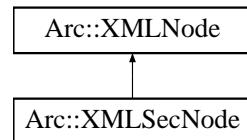
- `XMLNode.h`

5.326 Arc::XMLSecNode Class Reference

Extends `XMLNode` class to support XML security operation.

```
#include <XMLSecNode.h>
```

Inheritance diagram for Arc::XMLSecNode:



Public Member Functions

- `XMLSecNode (XMLNode &node)`
- `void AddSignatureTemplate (const std::string &id_name, const SignatureMethod sign_method, const std::string &incl_namespaces="")`
- `bool SignNode (const std::string &privkey_file, const std::string &cert_file)`
- `bool VerifyNode (const std::string &id_name, const std::string &ca_file, const std::string &ca_path, bool verify_trusted=true)`
- `bool EncryptNode (const std::string &cert_file, const SymEncryptionType encrypt_type)`
- `bool DecryptNode (const std::string &privkey_file, XMLNode &decrypted_node)`

5.326.1 Detailed Description

Extends `XMLNode` class to support XML security operation.

All `XMLNode` methods are exposed by inheriting from `XMLNode`. `XMLSecNode` itself does not own node, instead it uses the node from the base class `XMLNode`.

5.326.2 Constructor & Destructor Documentation

5.326.2.1 `Arc::XMLSecNode::XMLSecNode (XMLNode & node)`

Create a object based on an `XMLNode` instance.

5.326.3 Member Function Documentation

5.326.3.1 `void Arc::XMLSecNode::AddSignatureTemplate (const std::string & id_name, const SignatureMethod sign_method, const std::string & incl_namespaces = " ")`

Add the signature template for later signing.

Parameters

<code>id_name</code>	The identifier name under this node which will be used for the <Signature> to refer to.
<code>sign_method</code>	The sign method for signing. Two options now, RSA_SHA1, DSA_SHA1

5.326.3.2 `bool Arc::XMLSecNode::DecryptNode (const std::string & privkey_file, XMLNode & decrypted_node)`

Decrypt the <xenc:EncryptedData/> under this node, the decrypted node will be output in the second argument of DecryptNode method. And the <xenc:EncryptedData/> under this node will be removed after decryption.

Parameters

<code>privkey_file</code>	The private key file, which is used for decrypting
<code>decrypted_node</code>	Output the decrypted node

5.326.3.3 `bool Arc::XMLSecNode::EncryptNode (const std::string & cert_file, const SymEncryptionType encr_type)`

Encrypt this node, after encryption, this node will be replaced by the encrypted node

Parameters

<code>cert_file</code>	The certificate file, the public key parsed from this certificate is used to encrypted the symmetric key, and then the symmetric key is used to encrypted the node
<code>encr_type</code>	The encryption type when encrypting the node, four option in SymEncryptionType
<code>verify_trusted</code>	Verify trusted certificates or not. If set to false, then only the signature will be checked (by using the public key from KeyInfo).

5.326.3.4 bool Arc::XMLSecNode::SignNode (const std::string & *privkey_file*, const std::string & *cert_file*)

Sign this node (identified by *id_name*).

Parameters

<i>privkey_file</i>	The private key file. The private key is used for signing
<i>cert_file</i>	The certificate file. The certificate is used as the <KeyInfo> part of the <Signature>; <KeyInfo> will be used for the other end to verify this <Signature>
<i>incl_namespaces</i>	InclusiveNamespaces for Transform in Signature

5.326.3.5 bool Arc::XMLSecNode::VerifyNode (const std::string & *id_name*, const std::string & *ca_file*, const std::string & *ca_path*, bool *verify_trusted* = true)

Verify the signature under this node

Parameters

<i>id_name</i>	The id of this node, which is used for identifying the node
<i>ca_file</i>	The CA file which used as trusted certificate when verify the certificate in the <KeyInfo> part of <Signature>
<i>ca_path</i>	The CA directory; either <i>ca_file</i> or <i>ca_path</i> should be set.

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

- XMLSecNode.h

Index

- ~BrokerLoader
 - Arc::BrokerLoader, 72
- ~Counter
 - Arc::Counter, 98
- ~DTRCallback
 - DataStaging::DTRCallback, 144
- ~Database
 - Arc::Database, 119
- ~IntraProcessCounter
 - Arc::IntraProcessCounter, 190
- ~JobControllerLoader
 - Arc::JobControllerLoader, 205
- ~JobDescriptionParserLoader
 - Arc::JobDescriptionParserLoader, 211
- ~Loader
 - Arc::Loader, 222
- ~Logger
 - Arc::Logger, 227
- ~MCCLoader
 - Arc::MCCLoader, 243
- ~Message
 - Arc::Message, 248
- ~PayloadRaw
 - Arc::PayloadRaw, 269
- ~PayloadStream
 - Arc::PayloadStream, 276
- ~Plexer
 - Arc::Plexer, 289
- ~Query
 - Arc::Query, 303
- ~Run
 - Arc::Run, 313
- ~SAMLToken
 - Arc::SAMLToken, 321
- ~SOAPMessage
 - Arc::SOAPMessage, 335
- ~SubmitterLoader
 - Arc::SubmitterLoader, 358
- ~TargetRetrieverLoader
 - Arc::TargetRetrieverLoader, 367
- ~URL
 - Arc::URL, 382
- ~URLLocation
 - Arc::URLLocation, 391
- ~WSAEndpointReference
 - Arc::WSAEndpointReference, 430
- ~X509Token
 - Arc::X509Token, 458
- ~XMLNode
 - Arc::XMLNode, 463
- Abandon
 - Arc::Run, 313
- Acquire
 - Arc::DelegationConsumer, 129
 - Arc::InformationContainer, 183
- acquire
 - Arc::FileLock, 169
- acquireDelegation
 - Arc::ClientX509Delegation, 87
- Action
 - Arc::WSAHeader, 433
- Add
 - Arc::MessageContext, 254
 - Arc::XMLNodeContainer, 473
- add
 - Arc::Adler32Sum, 52
 - Arc::CheckSum, 75
 - Arc::CheckSumAny, 78
 - Arc::CRC32Sum, 105
 - Arc::MD5Sum, 245
 - Arc::MessageAttributes, 250
 - Arc::SoftwareRequirement, 346, 347
- add_dtr
 - DataStaging::DTRLList, 147
- AddBartender
 - Arc::UserConfig, 398
- AddCADir
 - Arc::BaseConfig, 67
- AddCAFfile
 - Arc::BaseConfig, 67
- AddCertExtObj

Arc::Credential, 110
 AddCertificate
 Arc::BaseConfig, 67
 AddChain
 Arc::VOMSTrustList, 429
 addDestination
 Arc::Logger, 228
 addDestinations
 Arc::Logger, 228
 AddExtension
 Arc::Credential, 110, 111
 AddHTTPOption
 Arc::URL, 382
 AddIndexServer
 Arc::TargetGenerator, 360
 AddJob
 Arc::TargetGenerator, 360, 361
 AddLDAPAttribute
 Arc::URL, 383
 AddLocation
 Arc::URL, 383
 AddMetaDataOption
 Arc::URL, 383
 AddNew
 Arc::XMLNodeContainer, 473
 AddOption
 Arc::URL, 383
 AddOverlay
 Arc::BaseConfig, 67
 AddPluginsPath
 Arc::BaseConfig, 67
 addPolicy
 ArcSec::Evaluator, 155
 ArcSec::Policy, 297
 AddPrivateKey
 Arc::BaseConfig, 68
 AddProxy
 Arc::BaseConfig, 68
 AddRegex
 Arc::VOMSTrustList, 429
 addRegistrar
 Arc::InfoRegisterContainer, 180
 addRequestItem
 ArcSec::Request, 308
 Address
 Arc::WSAEndpointReference, 431
 AddSecHandler
 Arc::ClientSOAP, 85
 Arc::MCC, 237
 Arc::Service, 331
 AddService
 Arc::TargetGenerator, 361
 addService
 Arc::InfoRegisterContainer, 180
 Arc::InfoRegistrar, 182
 AddServices
 Arc::UserConfig, 398, 399
 AddSignatureTemplate
 Arc::XMLSecNode, 475
 AddTarget
 Arc::TargetGenerator, 361
 addVOMSAC
 Arc, 32
 AfterFork
 Arc::Run, 313
 allocated
 Arc::PayloadRawBuf, 271
 allocated_
 Arc::WSRF, 437
 ApplicationEnvironments
 Arc::ExecutionTarget, 163
 ApplyToConfig
 Arc::UserConfig, 399
 approveCSR
 Arc::OAuthConsumer, 264
 Arc::SAML2SSOHTTPClient, 318
 Arc, 19
 addVOMSAC, 32
 AttrConstIter, 30
 AttrIter, 30
 AttrMap, 31
 BUSY_ERROR, 32
 ContentFromPayload, 33
 CreateThreadFunction, 33
 createVOMSAC, 33
 CredentialLogger, 43
 DirCreate, 33
 DirDelete, 33
 EnvLockUnwrap, 33
 EnvLockWrap, 34
 escape_hex, 31
 escape_octal, 31
 escape_chars, 34
 escape_type, 31
 FileCopy, 34
 FileCreate, 34
 FileDelete, 34
 FileLink, 34
 FileRead, 35
 FileReadLink, 35

FileStat, 35
final_xmlsec, 35
GENERIC_ERROR, 32
get_cert_str, 35
get_key_from_certfile, 35
get_key_from_certstr, 35
get_key_from_keyfile, 35
get_key_from_keystore, 36
get_node, 36
get_plugin_instance, 31
getCredentialProperty, 36
GUID, 36
init_xmlsec, 36
inttostr, 36, 37
istring_to_level, 37
load_key_from_certfile, 38
load_key_from_certstr, 38
load_key_from_keyfile, 38
load_trusted_cert_file, 38
load_trusted_cert_str, 38
load_trusted_certs, 38
LogFormat, 31
LogLevel, 31
MatchXMLName, 38, 39
MatchXMLNamespace, 39
OpenSSLInit, 39
operator<<, 39
parseVOMSAC, 39, 40
PARSING_ERROR, 32
passphrase_callback, 41
plugins_table_name, 43
PROTOCOL_RECOGNIZED_ERROR, 32
SESSION_CLOSE, 32
STATUS_OK, 32
StatusKind, 32
string, 41
strtoint, 41
thread_stacksize, 43
TimeStamp, 42
TmpDirCreate, 42
TmpFileCreate, 42
UNKNOWN_SERVICE_ERROR, 32
uri_encode, 42
VOMSDecode, 42
WSAFault, 32
WSAFaultAssign, 42
WSAFaultExtract, 43
WSAFaultInvalidAddressingHeader, 32
WSAFaultUnknown, 32
Arc::Adler32Sum, 51
add, 52
end, 52
print, 53
scan, 53
start, 53
Arc::ApplicationEnvironment, 56
Arc::ApplicationType, 56
Arc::ArcLocation, 56
GetPlugins, 57
Init, 57
Arc::ARCPolicyHandlerConfig, 57
Arc::ArcVersion, 58
Arc::AttributelIterator, 59
AttributelIterator, 60
current_, 62
end_, 62
hasMore, 60
key, 61
MessageAttributes, 62
operator*, 61
operator++, 61
operator->, 61
Arc::AutoPointer, 66
Arc::Base64, 66
Arc::BaseConfig, 66
AddCADir, 67
AddCAFfile, 67
AddCertificate, 67
AddOverlay, 67
AddPluginsPath, 67
AddPrivateKey, 68
AddProxy, 68
GetOverlay, 68
MakeConfig, 68
Arc::Broker, 69
GetBestTarget, 70
PossibleTargets, 71
PreFilterTargets, 70
SortTargets, 70
Arc::BrokerLoader, 71
~BrokerLoader, 72
BrokerLoader, 72
GetBrokers, 72
load, 72
Arc::BrokerPluginArgument, 72
Arc::ByteArray, 73
Arc::CertEnvLocker, 74
operator PluginsFactory *, 74

Arc::CheckSum, 75
 add, 75
 end, 76
 print, 76
 scan, 76
 start, 77
Arc::CheckSumAny, 77
 add, 78
 end, 78
 FileChecksum, 78
 print, 79
 scan, 79
 start, 79
Arc::CIStringValue, 80
 CIStringValue, 81
 equal, 81
 operator bool, 81
Arc::ClassLoader, 81
Arc::ClassLoaderPluginArgument, 82
Arc::ClientHTTP, 82
Arc::ClientHTTPwithSAML2SSO, 83
 ClientHTTPwithSAML2SSO, 83
 process, 83
Arc::ClientInterface, 83
Arc::ClientSOAP, 84
 AddSecHandler, 85
 ClientSOAP, 85
 GetEntry, 85
 Load, 85
 process, 85
Arc::ClientSOAPwithSAML2SSO, 85
 ClientSOAPwithSAML2SSO, 86
 process, 86
Arc::ClientTCP, 86
Arc::ClientX509Delegation, 87
 acquireDelegation, 87
 ClientX509Delegation, 87
 createDelegation, 88
Arc::Config, 90
 Config, 91
 getFileName, 91
 parse, 91
 print, 91
 save, 91
 setFileName, 92
Arc::ConfusaCertHandler, 92
 ConfusaCertHandler, 92
 createCertRequest, 92
 getCertRequestB64, 92
Arc::ConfusaParserUtils, 93
 destroy_doc, 93
 evaluate_path, 93
 extract_body_information, 93
 get_doc, 94
 handle_redirect_step, 94
 urlencode, 94
 urlencode_params, 94
Arc::CountedPointer, 94
Arc::Counter, 95
 ~Counter, 98
 cancel, 98
 changeExcess, 98
 changeLimit, 98
 Counter, 98
 extend, 99
 getCounterTicket, 99
 getCurrentTime, 100
 getExcess, 100
 getExpirationReminder, 100
 getExpiryTime, 101
 getLimit, 101
 getValue, 101
 IDType, 97
 reserve, 101
 setExcess, 102
 setLimit, 102
Arc::CounterTicket, 103
 cancel, 104
 CounterTicket, 104
 extend, 104
 isValid, 104
Arc::CRC32Sum, 105
 add, 105
 end, 106
 print, 106
 scan, 106
 start, 106
Arc::Credential, 107
 AddCertExtObj, 110
 AddExtension, 110, 111
 Credential, 109, 110
 GenerateEECRequest, 111
 GenerateRequest, 111, 112
 GetCAName, 112
 GetCert, 112
 GetCertNumofChain, 112
 GetCertReq, 112
 GetDN, 112
 GetEndTime, 112
 GetExtension, 112

getFormat, 112
GetIdentityName, 113
GetIssuerName, 113
GetLifeTime, 113
GetPrivKey, 113
GetProxyPolicy, 113
GetPubKey, 113
GetStartTime, 113
GetType, 113
GetVerification, 113
InitProxyCertInfo, 113
InquireRequest, 114
IsCredentialsValid, 114
IsValid, 114
.LogError, 114
OutputCertificate, 114
OutputCertificateChain, 115
OutputPrivatekey, 115
OutputPublickey, 115
SelfSignEECRequest, 115
SetLifeTime, 115
SetProxyPolicy, 116
SetStartTime, 116
SignEECRequest, 116
SignRequest, 116, 117
STACK_OF, 117
Arc::CredentialError, 117
 CredentialError, 117
Arc::CredentialStore, 118
Arc::Database, 118
 ~Database, 119
 close, 119
 connect, 119
 Database, 119
 enable_ssl, 119
 isconnected, 120
 shutdown, 120
Arc::DBranch, 128
Arc::DelegationConsumer, 128
 Acquire, 129
 Backup, 129
 DelegationConsumer, 129
 Generate, 129
 ID, 129
 .LogError, 129
 Request, 130
 Restore, 130
Arc::DelegationConsumerSOAP, 130
 DelegateCredentialsInit, 131
 DelegatedToken, 131
DelegationConsumerSOAP, 131
UpdateCredentials, 131
Arc::DelegationContainerSOAP, 131
 context_lock_, 133
 DelegateCredentialsInit, 132
 DelegatedToken, 132
 max_duration_, 133
 max_size_, 133
 max_usage_, 133
 UpdateCredentials, 132
Arc::DelegationProvider, 133
 Delegate, 134
 DelegationProvider, 134
Arc::DelegationProviderSOAP, 134
 DelegateCredentialsInit, 135, 136
 DelegatedToken, 136
 DelegationProviderSOAP, 135
 ID, 136
 UpdateCredentials, 136
Arc::DiskSpaceRequirementType, 138
 CacheDiskSpace, 138
 DiskSpace, 138
 SessionDiskSpace, 138
Arc::DlItem, 139
Arc::DlItemString, 139
Arc::DNListHandlerConfig, 139
Arc::EnvLockWrapper, 152
Arc::ExecutableType, 160
Arc::ExecutionTarget, 160
 ApplicationEnvironments, 163
 ComputingShareName, 163
 ExecutionTarget, 161
 FreeSlotsWithDuration, 163
 GetSubmitter, 161
 MaxDiskSpace, 163
 MaxMainMemory, 163
 MaxVirtualMemory, 163
 OperatingSystem, 164
 operator=, 162
 Print, 162
 SaveToStream, 162
 Update, 162
Arc::ExpirationReminder, 164
 getExpiryTime, 165
 getReservationID, 165
 operator<, 165
Arc::FileAccess, 165
 copy, 167
 geterrno, 167
 mkdirp, 167

mkstemp, 167
 open, 167
 opendir, 167
 setuid, 167
 Arc::FileAccess::header_t, 177
 Arc::FileLock, 167
 acquire, 169
 check, 169
 FileLock, 168
 release, 169
 Arc::FileType, 170
 Checksum, 170
 Arc::FinderLoader, 170
 Arc::GlobusResult, 175
 Arc::GLUE2, 175
 Arc::GSSCredential, 176
 Arc::HakaClient, 176
 processConsent, 176
 processIdP2Confusa, 176
 processIdPLogin, 176
 Arc::HTTPClientInfo, 177
 Arc::InfoCache, 177
 InfoCache, 177
 Arc::InfoCacheInterface, 178
 Get, 178
 Arc::InfoFilter, 178
 Filter, 179
 InfoFilter, 179
 Arc::InfoRegister, 180
 Arc::InfoRegisterContainer, 180
 addRegistrar, 180
 addService, 180
 removeService, 181
 Arc::InfoRegisters, 181
 InfoRegisters, 181
 Arc::InfoRegistrar, 182
 addService, 182
 registration, 182
 Arc::InformationContainer, 182
 Acquire, 183
 Assign, 183
 doc_, 184
 Get, 184
 InformationContainer, 183
 Arc::InformationInterface, 184
 Get, 185
 InformationInterface, 185
 lock_, 185
 Arc::InformationRequest, 185
 InformationRequest, 186
 SOAP, 186
 Arc::InformationResponse, 187
 InformationResponse, 187
 Result, 187
 Arc::IniConfig, 187
 Arc::initializeCredentialsType, 188
 Arc::IntraProcessCounter, 189
 ~IntraProcessCounter, 190
 cancel, 190
 changeExcess, 190
 changeLimit, 191
 extend, 191
 getExcess, 191
 getLimit, 192
 getValue, 192
 IntraProcessCounter, 190
 reserve, 192
 setExcess, 193
 setLimit, 193
 Arc::ISIS_description, 193
 Arc::IString, 194
 Arc::Job, 194
 Job, 195
 operator=, 195
 Print, 195
 ReadAllJobsFromFile, 195
 ReadJobIDsFromFile, 196
 RemoveJobsFromFile, 197
 SaveToStream, 197
 ToXML, 198
 WriteJobIDsToFile, 198
 WriteJobIDToFile, 198
 WriteJobsToFile, 199, 200
 WriteJobsToTruncatedFile, 200
 Arc::JobController, 201
 Cat, 202
 Migrate, 203
 PrintJobStatus, 203
 SaveJobStatusToStream, 204
 Arc::JobControllerLoader, 204
 ~JobControllerLoader, 205
 GetJobControllers, 205
 JobControllerLoader, 205
 load, 205
 Arc::JobControllerPluginArgument, 206
 Arc::JobDescription, 206
 GetSourceLanguage, 207
 operator bool, 207
 OtherAttributes, 210
 Parse, 207, 208

Print, 208
SaveToStream, 209
UnParse, 209
Arc::JobDescriptionParser, 210
Arc::JobDescriptionParserLoader, 210
 ~JobDescriptionParserLoader, 211
 GetJobDescriptionParsers, 211
 JobDescriptionParserLoader, 211
 load, 212
Arc::JobDescriptionParserLoader::iterator, 194
Arc::JobIdentificationType, 212
Arc::JobState, 212
 IsFinished, 213
Arc::JobSupervisor, 213
 Cancel, 215
 Clean, 215
 Get, 216
 GetJobControllers, 217
 JobSupervisor, 214
 Kill, 217
 Migrate, 217
 Renew, 218
 Resubmit, 219
 Resume, 220
Arc::LoadableModuleDescription, 221
Arc::Loader, 221
 ~Loader, 222
 factory_, 222
 Loader, 222
Arc::LogDestination, 223
 LogDestination, 223
Arc::LogFile, 224
 log, 225
 LogFile, 224, 225
 setBackups, 225
 setMaxSize, 225
 setReopen, 225
Arc::Logger, 226
 ~Logger, 227
 addDestination, 228
 addDestinations, 228
 getDestinations, 228
 getRootLogger, 228
 getThreshold, 228
 Logger, 227
 msg, 228, 229
 setThreadContext, 229
 setThreshold, 229
 setThresholdForDomain, 229, 230
Arc::LoggerContext, 230
Arc::LoggerFormat, 230
Arc::LogMessage, 231
 getLevel, 232
 Logger, 232
 LogMessage, 231
 operator<<, 232
 setIdentifier, 232
Arc::LogStream, 233
 log, 234
 LogStream, 233, 234
Arc::MCC, 236
 AddSecHandler, 237
 logger, 238
 MCC, 237
 Next, 237
 next_, 238
 process, 237
 ProcessSecHandlers, 237
 sechandlers_, 238
 Unlink, 237
Arc::MCC_Status, 238
 getExplanation, 239
 getKind, 239
 getOrigin, 239
 isOk, 240
 MCC_Status, 239
 operator bool, 240
 operator std::string, 240
Arc::MCCConfig, 241
 MakeConfig, 241
Arc::MCCHandle, 241
 process, 242
Arc::MCCLoader, 242
 ~MCCLoader, 243
 MCCLoader, 243
Arc::MCCPluginArgument, 244
Arc::MD5Sum, 244
 add, 245
 end, 245
 print, 245
 scan, 246
 start, 246
Arc::MemoryAllocationException, 246
Arc::Message, 246
 ~Message, 248
 Attributes, 248
 Auth, 248
 AuthContext, 248
 Context, 248

Message, 247, 248
 operator=, 249
 Payload, 249
Arc::MessageAttributes, 249
 add, 250
 attributes_, 252
 count, 251
 get, 251
 getAll, 251
 MessageAttributes, 250
 remove, 251
 removeAll, 252
 set, 252
Arc::MessageAuth, 252
 Export, 253
 Filter, 253
Arc::MessageAuthContext, 254
Arc::MessageContext, 254
 Add, 254
Arc::MessageContextElement, 255
Arc::MessagePayload, 255
Arc::ModuleDesc, 256
Arc::ModuleManager, 256
 find, 257
 findLocation, 257
 load, 257
 makePersistent, 257
 ModuleManager, 257
 reload, 258
 setCfg, 258
 unload, 258
Arc::MultiSecAttr, 258
 Export, 259
 operator bool, 259
Arc::MySQLDatabase, 259
 close, 260
 connect, 260
 enable_ssl, 260
 isconnected, 260
 shutdown, 261
Arc::MySQLQuery, 261
 execute, 261
 get_array, 262
 get_num_columns, 262
 get_num_rows, 262
 get_row, 262
 get_row_field, 263
Arc::NotificationType, 263
Arc::NS, 263
Arc::OAuthConsumer, 263
 approveCSR, 264
OAuthConsumer, 264
 parseDN, 264
 processLogin, 265
 pushCSR, 265
 storeCert, 265
Arc::OpenIdpClient, 265
 processConsent, 266
 processIdP2Confusa, 266
 processIdPLogin, 266
Arc::OptionParser, 266
Arc::PathIterator, 267
 operator bool, 268
 operator*, 268
 operator++, 268
 operator--, 268
PathIterator, 267
 Rest, 268
Arc::PayloadRaw, 268
 ~PayloadRaw, 269
 Buffer, 269
 BufferPos, 269
 BufferSize, 269
 Content, 270
 Insert, 270
 PayloadRaw, 269
 Size, 270
 Truncate, 270
Arc::PayloadRawBuf, 271
 allocated, 271
 length, 271
 size, 271
Arc::PayloadRawInterface, 271
 Buffer, 272
 BufferPos, 272
 BufferSize, 272
 Content, 273
 Insert, 273
 Size, 273
 Truncate, 273
Arc::PayloadSOAP, 274
 PayloadSOAP, 274, 275
Arc::PayloadStream, 275
 ~PayloadStream, 276
 Get, 276
 handle_, 278
 Limit, 276
 operator bool, 277
 PayloadStream, 276
 Pos, 277

Put, 277
seekable_, 278
Size, 278
Timeout, 278
Arc::PayloadStreamInterface, 278
 Get, 279, 280
 Limit, 280
 operator bool, 280
 Pos, 280
 Put, 280, 281
 Size, 281
 Timeout, 281
Arc::PayloadWSRF, 281
 PayloadWSRF, 282
Arc::Period, 284
 GetPeriod, 285
 istr, 285
 operator std::string, 285
 operator<, 285
 operator<=, 285
 operator>, 285
 operator>=, 286
 operator=, 285
 operator==, 285
 Period, 284
 SetPeriod, 286
Arc::Plexer, 288
 ~Plexer, 289
 logger, 290
 Next, 290
 Plexer, 289
 process, 290
Arc::PlexerEntry, 290
Arc::Plugin, 291
Arc::PluginArgument, 292
 get_factory, 293
 get_module, 293
Arc::PluginDesc, 293
Arc::PluginDescriptor, 293
Arc::PluginsFactory, 294
 FilterByKind, 295
 load, 295
 PluginsFactory, 295
 report, 295
 scan, 295
 TryLoad, 295
Arc::PrintF, 300
Arc::PrintFBase, 300
Arc::Profile, 302
Arc::Query, 302
 ~Query, 303
 execute, 303
 get_array, 304
 get_num_columns, 304
 get_num_rows, 304
 get_row, 304, 305
 get_row_field, 305
 Query, 303
 Arc::Range, 305
 Arc::Register_Info_Type, 305
 Arc::RegisteredService, 305
 RegisteredService, 306
 Arc::RegularExpression, 306
 match, 307
 Arc::ResourceSlotType, 311
 Arc::ResourcesType, 311
 Arc::Run, 312
 ~Run, 313
 Abandon, 313
 AfterFork, 313
 AssignStderr, 313
 AssignStdin, 313
 AssignStdout, 314
 AssignWorkingDirectory, 314
 CloseStderr, 314
 CloseStdin, 314
 CloseStdout, 314
 KeepStderr, 314
 KeepStdin, 314
 KeepStdout, 314
 Kill, 314
 operator bool, 314
 ReadStderr, 315
 ReadStdout, 315
 Result, 315
 Run, 313
 Running, 315
 Start, 315
 Wait, 315
 WriteStdin, 315
 Arc::SAML2LoginClient, 316
 findSimpleSAMLInstallation, 316
 processLogin, 317
 SAML2LoginClient, 316
 Arc::SAML2SSOHTTPClient, 317
 approveCSR, 318
 parseDN, 318
 processConsent, 318
 processIdP2Confusa, 318
 processIdPLogin, 318

processLogin, 318
 pushCSR, 318
 storeCert, 319
Arc::SAMLToken, 319
 ~SAMLToken, 321
 Authenticate, 321, 322
 operator bool, 322
 SAMLToken, 321
 SAMLVersion, 320
Arc::ScalableTime, 322
Arc::ScalableTime< int >, 322
Arc::SecAttr, 324
 Export, 325
 get, 325
 getAll, 325
 Import, 326
 operator bool, 326
 operator==, 326
Arc::SecAttrFormat, 326
Arc::SecAttrValue, 327
 operator bool, 327
 operator==, 328
Arc::SecHandlerConfig, 328
Arc::Service, 330
 AddSecHandler, 331
 getID, 331
 logger, 332
 ProcessSecHandlers, 332
 RegistrationCollector, 332
 sechandlers_, 332
 Service, 331
Arc::ServicePluginArgument, 332
Arc::SharedMutex, 333
Arc::SimpleCondition, 333
 broadcast, 333
 lock, 333
 reset, 333
 signal, 334
 signal_nonblock, 334
 unlock, 334
 wait, 334
 wait_nonblock, 334
Arc::SimpleCounter, 334
 wait, 334
Arc::SOAPMessage, 335
 ~SOAPMessage, 335
 Attributes, 336
 Payload, 336
 SOAPMessage, 335
Arc::Software, 336
 ComparisonOperator, 338
 ComparisonOperatorEnum, 338
 convert, 339
 empty, 340
 EQUAL, 338
 getFamily, 340
 getName, 340
 getVersion, 340
 GREATERTHAN, 338
 GREATERTHANOREQUAL, 338
 LESSTHAN, 338
 LESSTHANOREQUAL, 338
 NOTEQUAL, 338
 operator std::string, 340
 operator<, 341
 operator<<, 344
 operator<=, 342
 operator>, 342
 operator>=, 343
 operator(), 341
 operator==, 342
 Software, 338, 339
 toString, 343
 VERSIONTOKENS, 344
Arc::SoftwareRequirement, 344
 add, 346, 347
 clear, 347
 empty, 347
 getComparisonOperatorList, 347
 getSoftwareList, 348
 isResolved, 348
 isSatisfied, 348, 349
 operator=, 350
 selectSoftware, 350, 351
 SoftwareRequirement, 345, 346
Arc::Submitter, 356
 GetTestJob, 356
 Migrate, 357
 Submit, 357
Arc::SubmitterLoader, 357
 ~SubmitterLoader, 358
 GetSubmitters, 358
 load, 358
 SubmitterLoader, 358
Arc::SubmitterPluginArgument, 359
Arc::TargetGenerator, 359
 AddIndexServer, 360
 AddJob, 360, 361
 AddService, 361
 AddTarget, 361

FoundJobs, 362
FoundTargets, 362
GetExecutionTargets, 362
GetJobs, 362
GetTargets, 362
ModifyFoundTargets, 363
PrintTargetInfo, 363
RetrieveExecutionTargets, 363
RetrieveJobs, 364
SaveTargetInfoToStream, 364
ServiceCounter, 364
TargetGenerator, 360
Arc::TargetRetriever, 365
 GetExecutionTargets, 366
 GetJobs, 366
 GetTargets, 366
 TargetRetriever, 365
Arc::TargetRetrieverLoader, 367
 ~TargetRetrieverLoader, 367
 GetTargetRetrievers, 367
 load, 368
 TargetRetrieverLoader, 367
Arc::TargetRetrieverPluginArgument, 368
Arc::ThreadDataItem, 368
 Attach, 369
 Dup, 370
 Get, 370
 ThreadDataItem, 369
Arc::ThreadInitializer, 370
Arc::ThreadRegistry, 370
 WaitForExit, 371
 WaitOrCancel, 371
Arc::Time, 371
 GetFormat, 372
 GetTime, 372
 operator std::string, 372
 operator<, 373
 operator<=, 373
 operator>, 374
 operator>=, 374
 operator+, 373
 operator-, 373
 operator=, 373
 operator==, 373
 SetFormat, 374
 SetTime, 374
 str, 374
 Time, 372
Arc::TimedMutex, 375
Arc::URL, 379
 ~URL, 382
 AddHTTPOption, 382
 AddLDAPAttribute, 383
 AddLocation, 383
 AddMetaDataOption, 383
 AddOption, 383
 BaseDN2Path, 383
 ChangeFullPath, 383
 ChangeHost, 383
 ChangeLDAPFilter, 384
 ChangeLDAPScope, 384
 ChangePath, 384
 ChangePort, 384
 ChangeProtocol, 384
 CommonLocOption, 384
 CommonLocOptions, 384
 commonlocoptions, 388
 ConnectionURL, 384
 FullPath, 384
 FullPathURIEncoded, 385
 fullstr, 385
 Host, 385
 host, 389
 HTTPOption, 385
 HTTPOptions, 385
 httpoptions, 389
 ip6addr, 389
 IsSecureProtocol, 385
 LDAPAttributes, 385
 Idapattributes, 389
 LDAPFilter, 385
 Idapfilter, 389
 LDAPScope, 385
 Idapscope, 389
 Locations, 386
 locations, 389
 MetaDataOption, 386
 MetaDataOptions, 386
 metadataoptions, 389
 operator bool, 386
 operator<, 386
 operator<<, 388
 operator==, 386
 Option, 386
 Options, 386
 OptionString, 387
 ParseOptions, 387
 ParsePath, 387
 Passwd, 387
 passwd, 389

Path, 387
 path, 389
 Path2BaseDN, 387
 plainstr, 387
 Port, 387
 port, 390
 Protocol, 387
 protocol, 390
 RemoveHTTPOption, 387
 RemoveMetaDataOption, 388
 RemoveOption, 388
 Scope, 382
 str, 388
 URL, 382
 urloptions, 390
 Username, 388
 username, 390
 valid, 390
 Arc::URLLocation, 390
 ~URLLocation, 391
 fullstr, 392
 Name, 392
 name, 392
 str, 392
 URLLocation, 391
 Arc::User, 392
 Arc::UserConfig, 392
 AddBartender, 398
 AddServices, 398, 399
 ApplyToConfig, 399
 ARCUSERDIRECTORY, 423
 Bartender, 400
 Broker, 400, 401
 CACertificatePath, 402
 CACertificatesDirectory, 403
 CertificateLifeTime, 403, 404
 CertificatePath, 404, 405
 ClearRejectedServices, 405
 ClearSelectedServices, 405, 406
 CredentialsFound, 406
 DEFAULT_BROKER, 424
 DEFAULT_TIMEOUT, 424
 DEFAULTCONFIG, 424
 EXAMPLECONFIG, 424
 GetRejectedServices, 406
 GetSelectedServices, 407
 GetUser, 407
 IdPName, 408
 InitializeCredentials, 408
 JobDownloadDirectory, 410
 JobListFile, 410, 411
 KeyPassword, 411, 412
 KeyPath, 412, 413
 KeySize, 413
 LoadConfigurationFile, 414
 operator bool, 415
 OverlayFile, 416
 Password, 416, 417
 ProxyPath, 417, 418
 SaveToFile, 418
 SetUser, 418
 SLCS, 419
 StoreDirectory, 419, 420
 SYSCONFIG, 424
 SYSCONFIGARCLOC, 425
 Timeout, 420
 UserConfig, 396, 397
 UserName, 421
 UtilsDirPath, 421, 422
 Verbosity, 422
 VOMSESPath, 423
 Arc::UsernameToken, 425
 Authenticate, 426, 427
 operator bool, 427
 PasswordType, 426
 Username, 427
 UsernameToken, 426
 Arc::UserSwitch, 427
 Arc::VOMSACInfo, 428
 Arc::VOMSTrustList, 428
 AddChain, 429
 AddRegex, 429
 VOMSTrustList, 428, 429
 Arc::WSAEndpointReference, 429
 ~WSAEndpointReference, 430
 Address, 431
 MetaData, 431
 operator XMLNode, 431
 operator=, 431
 ReferenceParameters, 431
 WSAEndpointReference, 430
 Arc::WSAHeader, 431
 Action, 433
 Check, 433
 FaultTo, 433
 From, 433
 header_allocated_, 435
 MessageID, 433
 NewReferenceParameter, 433
 operator XMLNode, 433

ReferenceParameter, 434
RelatesTo, 434
RelationshipType, 434
ReplyTo, 434
To, 434
WSAHeader, 432
Arc::WSRF, 435
 allocated_, 437
 operator bool, 436
 set_namespaces, 436
 SOAP, 436
 valid_, 437
 WSRF, 436
Arc::WSRFBaseFault, 437
 set_namespaces, 438
 WSRFBaseFault, 438
Arc::WSRFResourceUnavailableFault, 438
Arc::WSRFResourceUnknownFault, 438
Arc::WSRP, 439
 set_namespaces, 441
 WSRP, 441
Arc::WSRPDeleteResourceProperties, 441
Arc::WSRPDeleteResourcePropertiesRequest, 441
Arc::WSRPDeleteResourcePropertiesRequestFailedFa-
 442
Arc::WSRPDeleteResourcePropertiesResponse, 442
Arc::WSRPFault, 443
 WSRPFault, 443
Arc::WSRPGetMultipleResourcePropertiesRe-
 444
Arc::WSRPGetMultipleResourcePropertiesRe-
 444
Arc::WSRPGetPropertyDocumentRequest, 444
Arc::WSRPGetPropertyDocumentResponse, 445
Arc::WSRPGetPropertyRequest, 445
Arc::WSRPInsertResourceProperties, 446
Arc::WSRPInsertResourcePropertiesRequest, 446
Arc::WSRPInsertResourcePropertiesRequestFailed-
 447
Arc::WSRPInsertResourcePropertiesResponse, 447
Arc::WSRPIInvalidModificationFault, 448
Arc::WSRPIInvalidResourcePropertyQNameFault,
 448
Arc::WSRPModifyResourceProperties, 449
Arc::WSRPPutResourcePropertyDocumentRequest,
 449
Arc::WSRPPutResourcePropertyDocumentResponse,
 450
Arc::WSRPQueryResourcePropertiesRequest,
 450
Arc::WSRPQueryResourcePropertiesResponse,
 450
Arc::WSRPResourcePropertyChangeFailure,
 451
 WSRPResourcePropertyChangeFail-
 ure, 451
Arc::WSRPSetResourcePropertiesRequest,
 452
Arc::WSRPSetResourcePropertiesResponse,
 452
Arc::WSRPSetResourcePropertyRequestFailedFault,
 452
Arc::WSRPUnableToModifyResourcePropertyFault,
 453
Arc::WSRPUnableToPutResourcePropertyDocumentFault,
 453
Arc::WSRPUpdateResourceProperties, 454
Arc::WSRPUpdateResourcePropertiesRequest,
 454
Arc::WSRPUpdateResourcePropertiesRequestFailedFault,
 455
Arc::WSRPUpdateResourcePropertiesResponse,
 455
Arc::X509Token, 457
 ~X509Token, 458
Arc::Authenticate, 458, 459
 operator bool, 459
Arc::X509Token, 457, 458
 X509TokenType, 457
Arc::XmlContainer, 459
Arc::XmlDatabase, 459
Arc::XmlNode, 459
 ~XmlNode, 463
Attribute, 463
AttributesSize, 463
Child, 463
Destroy, 463
Exchange, 463
FullName, 464
Get, 464
GetDoc, 464

GetRoot, 464
 GetXML, 464
 is_owner_, 472
 is_temporary_, 472
 MatchXMLName, 471
 MatchXMLNamespace, 471
 Move, 464
 Name, 465
 Namespace, 465
 NamespacePrefix, 465
 Namespaces, 465
 New, 466
 NewAttribute, 466
 NewChild, 466, 467
 operator bool, 467
 operator std::string, 467
 operator++, 467
 operator--, 468
 operator=, 468
 operator==, 468
 Parent, 469
 Path, 469
 Prefix, 469
 ReadFromFile, 469
 ReadFromStream, 470
 Replace, 470
 Same, 470
 SaveToFile, 470
 SaveToStream, 470
 Set, 470
 Size, 470
 Swap, 470
 Validate, 470
 XMLNode, 462
 XPathLookup, 471
Arc::XMLNodeContainer, 472
 Add, 473
 AddNew, 473
 Nodes, 473
 operator=, 473
 Size, 474
 XMLNodeContainer, 473
Arc::XMLSecNode, 474
 AddSignatureTemplate, 475
 DecryptNode, 475
 EncryptNode, 475
 SignNode, 475
 VerifyNode, 476
 XMLSecNode, 475
ArcCredential, 43
 CERT_TYPE_CA, 44
 CERT_TYPE_EEC, 44
 CERT_TYPE_GSI_2_LIMITED_PROXY, 45
 CERT_TYPE_GSI_2_PROXY, 45
 CERT_TYPE_GSI_3_IMPERSONATION_PROXY, 44
 CERT_TYPE_GSI_3_INDEPENDENT_PROXY, 45
 CERT_TYPE_GSI_3_LIMITED_PROXY, 45
 CERT_TYPE_GSI_3_RESTRICTED_PROXY, 45
 CERT_TYPE_RFC_ANYLANGUAGE_PROXY, 45
 CERT_TYPE_RFC_IMPERSONATION_PROXY, 45
 CERT_TYPE_RFC_INDEPENDENT_PROXY, 45
 CERT_TYPE_RFC_LIMITED_PROXY, 45
 CERT_TYPE_RFC_RESTRICTED_PROXY, 45
 certType, 44
ArcCredential::ACACI, 49
ArcCredential::ACATHOLDER, 49
ArcCredential::ACATTR, 49
ArcCredential::ACATTRIBUTE, 49
ArcCredential::ACC, 49
ArcCredential::ACCERTS, 50
ArcCredential::ACDIGEST, 50
ArcCredential::ACFORM, 50
ArcCredential::ACFULLATTRIBUTES, 50
ArcCredential::ACHOLDER, 50
ArcCredential::ACIETFATTR, 50
ArcCredential::ACINFO, 51
ArcCredential::ACIS, 51
ArcCredential::ACSEQ, 51
ArcCredential::ACTARGET, 51
ArcCredential::ACTARGETS, 51
ArcCredential::ACVAL, 51
ArcCredential::cert_verify_context, 74
ArcCredential::PROXYCERTINFO_st, 302
ArcCredential::PROXPOLICY_st, 302
ArcSec::AlgFactory, 54
 createAlg, 54
ArcSec::AnyURIAttribute, 55
 encode, 55
 equal, 55
 getId, 55

getType, 55
ArcSec::ArcPeriod, 57
ArcSec::Attr, 58
ArcSec::AttributeFactory, 58
ArcSec::AttributeProxy, 62
 getAttribute, 63
ArcSec::AttributeValue, 63
 encode, 64
 equal, 64
 getId, 64
 getType, 64
ArcSec::Attrs, 65
ArcSec::AuthzRequest, 65
ArcSec::AuthzRequestSection, 65
ArcSec::BooleanAttribute, 68
 encode, 69
 equal, 69
 getId, 69
 getType, 69
ArcSec::CombiningAlg, 89
 combine, 89
 getAlgId, 89
ArcSec::DateAttribute, 126
 encode, 126
 equal, 126
 getId, 126
 getType, 126
ArcSec::DateTimeAttribute, 127
 encode, 127
 equal, 127
 getId, 127
 getType, 127
ArcSec::DenyOverridesCombiningAlg, 137
 combine, 137
 getAlgId, 138
ArcSec::DurationAttribute, 151
 encode, 152
 equal, 152
 getId, 152
 getType, 152
ArcSec::EqualFunction, 152
 evaluate, 153
 getFunctionName, 153
ArcSec::EvalResult, 154
ArcSec::EvaluationCtx, 154
 EvaluationCtx, 154
ArcSec::Evaluator, 154
 addPolicy, 155
 evaluate, 156
 getAlgFactory, 157
 getAttrFactory, 157
 getFnFactory, 157
 getName, 157
 setCombiningAlg, 157
ArcSec::EvaluatorContext, 158
 operator AlgFactory *, 158
 operator AttributeFactory *, 158
 operator FnFactory *, 158
ArcSec::EvaluatorLoader, 158
 getEvaluator, 159
 getPolicy, 159
 getRequest, 159, 160
ArcSec::FnFactory, 170
 createFn, 171
ArcSec::Function, 171
 evaluate, 172
ArcSec::GenericAttribute, 174
 encode, 174
 equal, 174
 getId, 175
 getType, 175
ArcSec::InRangeFunction, 188
 evaluate, 188
ArcSec::MatchFunction, 234
 evaluate, 235
 getFunctionName, 235
ArcSec::OrderedCombiningAlg, 266
ArcSec::PDP, 282
ArcSec::PDPConfigContext, 283
ArcSec::PDPPPluginArgument, 283
ArcSec::PeriodAttribute, 286
 encode, 286
 equal, 287
 getId, 287
 getType, 287
ArcSec::PermitOverridesCombiningAlg, 287
 combine, 288
 getAlgId, 288
ArcSec::Policy, 296
 addPolicy, 297
 eval, 297
 getEffect, 297
 getEvalName, 297
 getEvalResult, 297
 getName, 297
 make_policy, 297
 Policy, 296, 297
 setEvalResult, 298
 setEvaluatorContext, 298
 ArcSec::PolicyParser, 298

parsePolicy, 299
 ArcSec::PolicyStore, 299
 PolicyStore, 299
 ArcSec::PolicyStore::PolicyElement, 298
 ArcSec::Request, 307
 addRequestItem, 308
 getEvalName, 308
 getName, 308
 getRequestItems, 309
 make_request, 309
 Request, 308
 setAttributeFactory, 309
 setRequestItems, 309
 ArcSec::RequestAttribute, 309
 duplicate, 310
 RequestAttribute, 310
 ArcSec::RequestItem, 310
 RequestItem, 310
 ArcSec::RequestTuple, 311
 ArcSec::Response, 311
 ArcSec::ResponseItem, 311
 ArcSec::ResponseList, 312
 ArcSec::SecHandler, 328
 ArcSec::SecHandlerConfig, 329
 ArcSec::SecHandlerPluginArgument, 329
 ArcSec::Security, 330
 ArcSec::Source, 352
 Source, 352
 ArcSec::SourceFile, 353
 ArcSec::SourceURL, 353
 ArcSec::StringAttribute, 355
 encode, 355
 equal, 355
 getId, 355
 getType, 355
 ArcSec::TimeAttribute, 374
 encode, 375
 equal, 375
 getId, 375
 getType, 375
 ArcSec::X500NameAttribute, 456
 encode, 456
 equal, 456
 getId, 456
 getType, 456
 ARCUSERDIRECTORY
 Arc::UserConfig, 423
 Assign
 Arc::InformationContainer, 183
 AssignStderr

Arc::Run, 313
 AssignStdin
 Arc::Run, 313
 AssignStdout
 Arc::Run, 314
 AssignWorkingDirectory
 Arc::Run, 314
 Attach
 Arc::ThreadDataItem, 369
 AttrConstIter
 Arc, 30
 Attribute
 Arc::XMLNode, 463
 Attributelterator
 Arc::Attributelterator, 60
 Attributes
 Arc::Message, 248
 Arc::SOAPMessage, 336
 attributes_
 Arc::MessageAttributes, 252
 AttributesSize
 Arc::XMLNode, 463
 AttrIter
 Arc, 30
 AttrMap
 Arc, 31
 Auth
 Arc::Message, 248
 AuthContext
 Arc::Message, 248
 Authenticate
 Arc::SAMLToken, 321, 322
 Arc::UsernameToken, 426, 427
 Arc::X509Token, 458, 459
 Backup
 Arc::DelegationConsumer, 129
 Bartender
 Arc::UserConfig, 400
 BaseDN2Path
 Arc::URL, 383
 broadcast
 Arc::SimpleCondition, 333
 Broker
 Arc::UserConfig, 400, 401
 BrokerLoader
 Arc::BrokerLoader, 72
 Buffer
 Arc::PayloadRaw, 269
 Arc::PayloadRawInterface, 272

BufferPos
 Arc::PayloadRaw, 269
 Arc::PayloadRawInterface, 272

BufferSize
 Arc::PayloadRaw, 269
 Arc::PayloadRawInterface, 272

BUSY_ERROR
 Arc, 32

CACertificatePath
 Arc::UserConfig, 402

CACertificatesDirectory
 Arc::UserConfig, 403

CACHE_ALREADY_PRESENT
 DataStaging, 46

CACHE_CHECKED
 DataStaging::DTRStatus, 150

CACHE_DOWNLOADED
 DataStaging, 46

CACHE_ERROR
 DataStaging::DTRErrorStatus, 146

CACHE_LOCKED
 DataStaging, 46

CACHE_NOT_USED
 DataStaging, 47

CACHE_PROCESSED
 DataStaging::DTRStatus, 151

CACHE_RENEW
 DataStaging, 46

CACHE_SKIP
 DataStaging, 47

CACHE_WAIT
 DataStaging::DTRStatus, 150

CACHEABLE
 DataStaging, 46

CacheDiskSpace
 Arc::DiskSpaceRequirementType, 138

CacheState
 DataStaging, 46

calculate_shares
 DataStaging::TransferShares, 378

Cancel
 Arc::JobSupervisor, 215

cancel
 Arc::Counter, 98
 Arc::CounterTicket, 104
 Arc::IntraProcessCounter, 190

CANCELLED
 DataStaging::DTRStatus, 150

CANCELLED_FINISHED
 DataStaging::DTRStatus, 150

Cat
 Arc::JobController, 202

CERT_TYPE_CA
 ArcCredential, 44

CERT_TYPE_EEC
 ArcCredential, 44

CERT_TYPE_GSI_2_LIMITED_PROXY
 ArcCredential, 45

CERT_TYPE_GSI_2_PROXY
 ArcCredential, 45

CERT_TYPE_GSI_3_IMPERSONATION_PROXY
 ArcCredential, 44

CERT_TYPE_GSI_3_INDEPENDENT_PROXY
 ArcCredential, 45

CERT_TYPE_GSI_3_LIMITED_PROXY
 ArcCredential, 45

CERT_TYPE_GSI_3_RESTRICTED_PROXY
 ArcCredential, 45

CERT_TYPE_RFC_ANYLANGUAGE_PROXY
 ArcCredential, 45

CERT_TYPE_RFC_IMPERSONATION_PROXY
 ArcCredential, 45

CERT_TYPE_RFC_INDEPENDENT_PROXY
 ArcCredential, 45

CERT_TYPE_RFC_LIMITED_PROXY
 ArcCredential, 45

CERT_TYPE_RFC_RESTRICTED_PROXY
 ArcCredential, 45

CertificateLifeTime
 Arc::UserConfig, 403, 404

CertificatePath
 Arc::UserConfig, 404, 405

certType
 ArcCredential, 44

changeExcess
 Arc::Counter, 98
 Arc::IntraProcessCounter, 190

ChangeFullPath
 Arc::URL, 383

ChangeHost
 Arc::URL, 383

ChangeLDAPFilter
 Arc::URL, 384

ChangeLDAPScope
 Arc::URL, 384

changeLimit
 Arc::Counter, 98
 Arc::IntraProcessCounter, 191

ChangePath
 Arc::URL, 384
 ChangePort
 Arc::URL, 384
 ChangeProtocol
 Arc::URL, 384
 Check
 Arc::WSAHeader, 433
 check
 Arc::FileLock, 169
 CHECK_CACHE
 DataStaging::DTRStatus, 150
 CHECKING_CACHE
 DataStaging::DTRStatus, 150
 Checksum
 Arc::FileType, 170
 Child
 Arc::XMLNode, 463
 CStringValue
 Arc::CStringValue, 81
 Clean
 Arc::JobSupervisor, 215
 clear
 Arc::SoftwareRequirement, 347
 ClearRejectedServices
 Arc::UserConfig, 405
 ClearSelectedServices
 Arc::UserConfig, 405, 406
 ClientHTTPwithSAML2SSO
 Arc::ClientHTTPwithSAML2SSO, 83
 ClientSOAP
 Arc::ClientSOAP, 85
 ClientSOAPwithSAML2SSO
 Arc::ClientSOAPwithSAML2SSO, 86
 ClientX509Delegation
 Arc::ClientX509Delegation, 87
 close
 Arc::Database, 119
 Arc::MySQLDatabase, 260
 CloseStderr
 Arc::Run, 314
 CloseStdin
 Arc::Run, 314
 CloseStdout
 Arc::Run, 314
 combine
 ArcSec::CombiningAlg, 89
 ArcSec::DenyOverridesCombiningAlg, copy
 137
 ArcSec::PermitOverridesCombiningAlg,
 288
 CommClosed
 DataStaging::DataDeliveryComm, 123
 CommExited
 DataStaging::DataDeliveryComm, 123
 CommFailed
 DataStaging::DataDeliveryComm, 123
 CommInit
 DataStaging::DataDeliveryComm, 123
 CommNoError
 DataStaging::DataDeliveryComm, 123
 CommonLocOption
 Arc::URL, 384
 CommonLocOptions
 Arc::URL, 384
 commonlocoptions
 Arc::URL, 388
 CommStatusType
 DataStaging::DataDeliveryComm, 123
 CommTimeout
 DataStaging::DataDeliveryComm, 123
 ComparisonOperator
 Arc::Software, 338
 ComparisonOperatorEnum
 Arc::Software, 338
 ComputingShareName
 Arc::ExecutionTarget, 163
 Config
 Arc::Config, 91
 ConfusaCertHandler
 Arc::ConfusaCertHandler, 92
 connect
 Arc::Database, 119
 Arc::MySQLDatabase, 260
 ConnectionURL
 Arc::URL, 384
 Content
 Arc::PayloadRaw, 270
 Arc::PayloadRawInterface, 273
 ContentFromPayload
 Arc, 33
 Context
 Arc::Message, 248
 context_lock_
 Arc::DelegationContainerSOAP, 133
 convert
 Arc::Software, 339
 Arc:: FileAccess, 167

count
 Arc::MessageAttributes, 251

Counter
 Arc::Counter, 98

CounterTicket
 Arc::CounterTicket, 104

createAlg
 ArcSec::AlgFactory, 54

createCertRequest
 Arc::ConfusaCertHandler, 92

createDelegation
 Arc::ClientX509Delegation, 88

createFn
 ArcSec::FnFactory, 171

CreateThreadFunction
 Arc, 33

createVOMSAC
 Arc, 33

Credential
 Arc::Credential, 109, 110

CredentialError
 Arc::CredentialError, 117

CredentialLogger
 Arc, 43

CredentialsFound
 Arc::UserConfig, 406

current_
 Arc::Attributelteator, 62

Database
 Arc::Database, 119

DataDeliveryComm
 DataStaging::DataDeliveryComm, 123

DataStaging, 45
 CACHE_ALREADY_PRESENT, 46
 CACHE_DOWNLOADED, 46
 CACHE_LOCKED, 46
 CACHE_NOT_USED, 47
 CACHE_RENEW, 46
 CACHE_SKIP, 47
 CACHEABLE, 46
 CacheState, 46
 NON_CACHEABLE, 46

DataStaging::CacheParameters, 73

DataStaging::DataDelivery, 120
 receiveDTR, 121

DataStaging::DataDeliveryComm, 121
 CommClosed, 123
 CommExited, 123
 CommFailed, 123

 CommInit, 123
 CommNoError, 123
 CommStatusType, 123
 CommTimeout, 123
 DataDeliveryComm, 123
 PullStatus, 123

 DataStaging::DataDeliveryComm::Status, 354

 DataStaging::DataDeliveryCommHandler, 124

 DataStaging::DataDeliveryLocalComm, 124

 DataStaging::DataDeliveryRemoteComm, 125

 DataStaging::DTR, 140
 DTR, 142
 registerCallback, 143
 reset, 143
 set_error_status, 143

 DataStaging::DTRCallback, 144
 ~DTRCallback, 144
 receiveDTR, 144

 DataStaging::DTRErrorStatus, 145
 CACHE_ERROR, 146
 DTRErrorLocation, 146
 DTRErrorStatusType, 146
 ERROR_DESTINATION, 146
 ERROR_SOURCE, 146
 ERROR_TRANSFER, 146
 ERROR_UNKNOWN, 146
 INTERNAL_LOGIC_ERROR, 146
 INTERNAL_PROCESS_ERROR, 146
 LOCAL_FILE_ERROR, 146
 NO_ERROR_LOCATION, 146
 NONE_ERROR, 146
 PERMANENT_REMOTE_ERROR, 146
 SELF_REPLICATION_ERROR, 146
 STAGING_TIMEOUT_ERROR, 146
 TEMPORARY_REMOTE_ERROR, 146
 TRANSFER_SPEED_ERROR, 146

 DataStaging::DTRLList, 146
 add_dtr, 147
 delete_dtr, 147
 dumpState, 147
 filter_dtrs_by_job, 148
 filter_dtrs_by_next_receiver, 148
 filter_dtrs_by_owner, 148
 filter_dtrs_by_status, 148
 filter_pending_dtrs, 148

 DataStaging::DTRStatus, 149
 CACHE_CHECKED, 150
 CACHE_PROCESSED, 151
 CACHE_WAIT, 150
 CANCELLED, 150

CANCELLED_FINISHED, 150
 CHECK_CACHE, 150
 CHECKING_CACHE, 150
 DONE, 150
 DTRStatusType, 150
 ERROR, 150
 NEW, 150
 NULL_STATE, 151
 PRE_CLEAN, 150
 PRE_CLEANED, 150
 PRE_CLEANING, 150
 PROCESS_CACHE, 150
 PROCESSING_CACHE, 151
 QUERY_REPLICA, 150
 QUERYSING_REPLICA, 150
 REGISTER_REPLICA, 150
 REGISTERING_REPLICA, 151
 RELEASE_REQUEST, 150
 RELEASING_REQUEST, 151
 REPLICA_QUERIED, 150
 REPLICA_REGISTERED, 151
 REQUEST_RELEASED, 151
 RESOLVE, 150
 RESOLVED, 150
 RESOLVING, 150
 STAGE_PREPARE, 150
 STAGED_PREPARED, 150
 STAGING_PREPARING, 150
 STAGING_PREPARING_WAIT, 150
 TRANSFER, 150
 TRANSFER_WAIT, 150
 TRANSFERRED, 151
 TRANSFERRING, 150
 TRANSFERRING_CANCEL, 151
 DataStaging::Generator, 173
 receiveDTR, 174
 DataStaging::Processor, 300
 receiveDTR, 301
 start, 301
 stop, 301
 DataStaging::Scheduler, 322
 receiveDTR, 323
 start, 323
 stop, 324
 DataStaging::TransferParameters, 376
 max_inactivity_time, 376
 min_average_bandwidth, 376
 min_current_bandwidth, 376
 DataStaging::TransferShares, 377
 calculate_shares, 378
 decrease_number_of_slots, 378
 decrease_transfer_share, 378
 GROUP, 378
 increase_transfer_share, 378
 NONE, 378
 ROLE, 378
 ShareType, 378
 USER, 378
 VO, 378
 decrease_number_of_slots
 DataStaging::TransferShares, 378
 decrease_transfer_share
 DataStaging::TransferShares, 378
 DecryptNode
 Arc::XMLSecNode, 475
 DEFAULT_BROKER
 Arc::UserConfig, 424
 DEFAULT_TIMEOUT
 Arc::UserConfig, 424
 DEFAULTCONFIG
 Arc::UserConfig, 424
 Delegate
 Arc::DelegationProvider, 134
 DelegateCredentialsInit
 Arc::DelegationConsumerSOAP, 131
 Arc::DelegationContainerSOAP, 132
 Arc::DelegationProviderSOAP, 135, 136
 DelegatedToken
 Arc::DelegationConsumerSOAP, 131
 Arc::DelegationContainerSOAP, 132
 Arc::DelegationProviderSOAP, 136
 DelegationConsumer
 Arc::DelegationConsumer, 129
 DelegationConsumerSOAP
 Arc::DelegationConsumerSOAP, 131
 DelegationProvider
 Arc::DelegationProvider, 134
 DelegationProviderSOAP
 Arc::DelegationProviderSOAP, 135
 delete_dtr
 DataStaging::DTRLList, 147
 Destroy
 Arc::XMLNode, 463
 destroy_doc
 Arc::ConfusaParserUtils, 93
 DirCreate
 Arc, 33
 DirDelete
 Arc, 33
 DiskSpace

Arc::DiskSpaceRequirementType, 138 EnvLockWrap
doc_ Arc, 34
Arc::InformationContainer, 184 EQUAL
DONE Arc::Software, 338
DataStaging::DTRStatus, 150 equal
DTR Arc::CIStringValue, 81
DataStaging::DTR, 142 ArcSec::AnyURIAttribute, 55
DTRErrorLocation ArcSec::AttributeValue, 64
DataStaging::DTRErrorStatus, 146 ArcSec::BooleanAttribute, 69
DTRErrorStatusType ArcSec::DateAttribute, 126
DataStaging::DTRErrorStatus, 146 ArcSec::DateTimeAttribute, 127
DTRStatusType ArcSec::DurationAttribute, 152
DataStaging::DTRStatus, 150 ArcSec::GenericAttribute, 174
dumpState ArcSec::PeriodAttribute, 287
DataStaging::DTRLList, 147 ArcSec::StringAttribute, 355
Dup ArcSec::TimeAttribute, 375
Arc::ThreadDataItem, 370 ArcSec::X500NameAttribute, 456
duplicate ERROR
ArcSec::RequestAttribute, 310 DataStaging::DTRStatus, 150
empty ERROR_DESTINATION
Arc::Software, 340 DataStaging::DTRErrorStatus, 146
Arc::SoftwareRequirement, 347 ERROR_SOURCE
enable_ssl DataStaging::DTRErrorStatus, 146
Arc::Database, 119 ERROR_TRANSFER
Arc::MySQLDatabase, 260 DataStaging::DTRErrorStatus, 146
encode ERROR_UNKNOWN
ArcSec::AnyURIAttribute, 55 DataStaging::DTRErrorStatus, 146
ArcSec::AttributeValue, 64 escape_hex
ArcSec::BooleanAttribute, 69 Arc, 31
ArcSec::DateAttribute, 126 escape_octal
ArcSec::DateTimeAttribute, 127 Arc, 31
ArcSec::DurationAttribute, 152 escape_chars
ArcSec::GenericAttribute, 174 Arc, 34
ArcSec::PeriodAttribute, 286 escape_type
ArcSec::StringAttribute, 355 Arc, 31
ArcSec::TimeAttribute, 375 eval
ArcSec::X500NameAttribute, 456 ArcSec::Policy, 297
EncryptNode evaluate
Arc::XMLSecNode, 475 ArcSec::EqualFunction, 153
end ArcSec::Evaluator, 156
Arc::Adler32Sum, 52 ArcSec::Function, 172
Arc::CheckSum, 76 ArcSec::InRangeFunction, 188
Arc::CheckSumAny, 78 ArcSec::MatchFunction, 235
Arc::CRC32Sum, 106 evaluate_path
Arc::MD5Sum, 245 Arc::ConfusaParserUtils, 93
end_ EvaluationCtx
Arc::Attributelterator, 62 ArcSec::EvaluationCtx, 154
EnvLockUnwrap EXAMPLECONFIG
Arc, 33 Arc::UserConfig, 424
Exchange

Arc::XMLNode, 463
 execute
 Arc::MySQLQuery, 261
 Arc::Query, 303
 ExecutionTarget
 Arc::ExecutionTarget, 161
 Export
 Arc::MessageAuth, 253
 Arc::MultiSecAttr, 259
 Arc::SecAttr, 325
 extend
 Arc::Counter, 99
 Arc::CounterTicket, 104
 Arc::IntraProcessCounter, 191
 extract_body_information
 Arc::ConfusaParserUtils, 93

 factory_
 Arc::Loader, 222
 FaultTo
 Arc::WSAHeader, 433
 FileChecksum
 Arc::CheckSumAny, 78
 FileCopy
 Arc, 34
 FileCreate
 Arc, 34
 FileDelete
 Arc, 34
 FileLink
 Arc, 34
 FileLock
 Arc::FileLock, 168
 FileRead
 Arc, 35
 FileReadLink
 Arc, 35
 FileStat
 Arc, 35
 Filter
 Arc::InfoFilter, 179
 Arc::MessageAuth, 253
 filter_dtrs_by_job
 DataStaging::DTRLList, 148
 filter_dtrs_by_next_receiver
 DataStaging::DTRLList, 148
 filter_dtrs_by_owner
 DataStaging::DTRLList, 148
 filter_dtrs_by_status
 DataStaging::DTRLList, 148
 filter_pending_dtrs
 DataStaging::DTRLList, 148
 FilterByKind
 Arc::PluginsFactory, 295
 final_xmlsec
 Arc, 35
 find
 Arc::ModuleManager, 257
 findLocation
 Arc::ModuleManager, 257
 findSimpleSAMLInstallation
 Arc::SAML2LoginClient, 316
 FoundJobs
 Arc::TargetGenerator, 362
 FoundTargets
 Arc::TargetGenerator, 362
 FreeSlotsWithDuration
 Arc::ExecutionTarget, 163
 From
 Arc::WSAHeader, 433
 FullName
 Arc::XMLNode, 464
 FullPath
 Arc::URL, 384
 FullPathURIEncoded
 Arc::URL, 385
 fullstr
 Arc::URL, 385
 Arc::URLLocation, 392

 GDS20::GDS20Service, 172
 process, 172
 Generate
 Arc::DelegationConsumer, 129
 GenerateEECRequest
 Arc::Credential, 111
 GenerateRequest
 Arc::Credential, 111, 112
 GENERIC_ERROR
 Arc, 32
 Get
 Arc::InfoCacheInterface, 178
 Arc::InformationContainer, 184
 Arc::InformationInterface, 185
 Arc::JobSupervisor, 216
 Arc::PayloadStream, 276
 Arc::PayloadStreamInterface, 279, 280
 Arc::ThreadDataItem, 370
 Arc::XMLNode, 464
 get

Arc::MessageAttributes, 251
Arc::SecAttr, 325
get_array
 Arc::MySQLQuery, 262
 Arc::Query, 304
get_cert_str
 Arc, 35
get_doc
 Arc::ConfusaParserUtils, 94
get_factory
 Arc::PluginArgument, 293
get_key_from_certfile
 Arc, 35
get_key_from_certstr
 Arc, 35
get_key_from_keyfile
 Arc, 35
get_key_from_keystr
 Arc, 36
get_module
 Arc::PluginArgument, 293
get_node
 Arc, 36
get_num_columns
 Arc::MySQLQuery, 262
 Arc::Query, 304
get_num_rows
 Arc::MySQLQuery, 262
 Arc::Query, 304
get_plugin_instance
 Arc, 31
get_row
 Arc::MySQLQuery, 262
 Arc::Query, 304, 305
get_row_field
 Arc::MySQLQuery, 263
 Arc::Query, 305
getAlgFactory
 ArcSec::Evaluator, 157
getalgId
 ArcSec::CombiningAlg, 89
 ArcSec::DenyOverridesCombiningAlg, getEvalResult
 138
 ArcSec::PermitOverridesCombiningAlg, getEvaluator
 288
getAll
 Arc::MessageAttributes, 251
 Arc::SecAttr, 325
getAttrFactory
 ArcSec::Evaluator, 157
getAttribute
 ArcSec::AttributeProxy, 63
GetBestTarget
 Arc::Broker, 70
GetBrokers
 Arc::BrokerLoader, 72
GetCName
 Arc::Credential, 112
GetCert
 Arc::Credential, 112
GetCertNumOfChain
 Arc::Credential, 112
GetCertReq
 Arc::Credential, 112
getCertRequestB64
 Arc::ConfusaCertHandler, 92
getComparisonOperatorList
 Arc::SoftwareRequirement, 347
getCounterTicket
 Arc::Counter, 99
getCredentialProperty
 Arc, 36
getCurrentTime
 Arc::Counter, 100
getDestinations
 Arc::Logger, 228
GetDN
 Arc::Credential, 112
GetDoc
 Arc::XMLNode, 464
getEffect
 ArcSec::Policy, 297
GetEndTime
 Arc::Credential, 112
GetEntry
 Arc::ClientSOAP, 85
geterrno
 Arc::FileAccess, 167
getEvalName
 ArcSec::Policy, 297
 ArcSec::Request, 308
getExcess
 Arc::Counter, 100
 Arc::IntraProcessCounter, 191
GetExecutionTargets
 Arc::TargetGenerator, 362

Arc::TargetRetriever, 366
getExpirationReminder
 Arc::Counter, 100
getExpiryTime
 Arc::Counter, 101
 Arc::ExpirationReminder, 165
getExplanation
 Arc::MCC_Status, 239
GetExtension
 Arc::Credential, 112
getFamily
 Arc::Software, 340
getFileName
 Arc::Config, 91
getFnFactory
 ArcSec::Evaluator, 157
GetFormat
 Arc::Time, 372
getFormat
 Arc::Credential, 112
getFunctionName
 ArcSec::EqualFunction, 153
 ArcSec::MatchFunction, 235
getId
 Arc::Service, 331
getId
 ArcSec::AnyURIAttribute, 55
 ArcSec::AttributeValue, 64
 ArcSec::BooleanAttribute, 69
 ArcSec::DateAttribute, 126
 ArcSec::DateTimeAttribute, 127
 ArcSec::DurationAttribute, 152
 ArcSec::GenericAttribute, 175
 ArcSec::PeriodAttribute, 287
 ArcSec::StringAttribute, 355
 ArcSec::TimeAttribute, 375
 ArcSec::X500NameAttribute, 456
GetIdentityName
 Arc::Credential, 113
GetIssuerName
 Arc::Credential, 113
GetJobControllers
 Arc::JobControllerLoader, 205
 Arc::JobSupervisor, 217
GetJobDescriptionParsers
 Arc::JobDescriptionParserLoader, 211
GetJobs
 Arc::TargetGenerator, 362
 Arc::TargetRetriever, 366
getKind

Arc::MCC_Status, 239
getLevel
 Arc::LogMessage, 232
GetLifeTime
 Arc::Credential, 113
getLimit
 Arc::Counter, 101
 Arc::IntraProcessCounter, 192
getName
 Arc::Software, 340
 ArcSec::Evaluator, 157
 ArcSec::Policy, 297
 ArcSec::Request, 308
getOrigin
 Arc::MCC_Status, 239
GetOverlay
 Arc::BaseConfig, 68
GetPeriod
 Arc::Period, 285
GetPlugins
 Arc::ArcLocation, 57
getPolicy
 ArcSec::EvaluatorLoader, 159
GetPrivKey
 Arc::Credential, 113
GetProxyPolicy
 Arc::Credential, 113
GetPubKey
 Arc::Credential, 113
GetRejectedServices
 Arc::UserConfig, 406
getRequest
 ArcSec::EvaluatorLoader, 159, 160
getRequestItems
 ArcSec::Request, 309
getReservationID
 Arc::ExpirationReminder, 165
GetRoot
 Arc::XMLNode, 464
getRootLogger
 Arc::Logger, 228
GetSelectedServices
 Arc::UserConfig, 407
getSoftwareList
 Arc::SoftwareRequirement, 348
GetSourceLanguage
 Arc::JobDescription, 207
GetStartTime
 Arc::Credential, 113
GetSubmitter

Arc::ExecutionTarget, 161
GetSubmitters
 Arc::SubmitterLoader, 358
GetTargetRetrievers
 Arc::TargetRetrieverLoader, 367
GetTargets
 Arc::TargetGenerator, 362
 Arc::TargetRetriever, 366
GetTestJob
 Arc::Submitter, 356
getThreshold
 Arc::Logger, 228
GetTime
 Arc::Time, 372
GetType
 Arc::Credential, 113
getType
 ArcSec::AnyURIAttribute, 55
 ArcSec::AttributeValue, 64
 ArcSec::BooleanAttribute, 69
 ArcSec::DateAttribute, 126
 ArcSec::DateTimeAttribute, 127
 ArcSec::DurationAttribute, 152
 ArcSec::GenericAttribute, 175
 ArcSec::PeriodAttribute, 287
 ArcSec::StringAttribute, 355
 ArcSec::TimeAttribute, 375
 ArcSec::X500NameAttribute, 456
 GetUser
 Arc::UserConfig, 407
getValue
 Arc::Counter, 101
 Arc::IntraProcessCounter, 192
GetVerification
 Arc::Credential, 113
getVersion
 Arc::Software, 340
GetXML
 Arc::XMLNode, 464
GREATERTHAN
 Arc::Software, 338
GREATERTHANOREQUAL
 Arc::Software, 338
GROUP
 DataStaging::TransferShares, 378
GUID
 Arc, 36
handle_
 Arc::PayloadStream, 278
handle_redirect_step
 Arc::ConfusaParserUtils, 94
hasMore
 Arc::AttributelIterator, 60
header_allocated_
 Arc::WSAHeader, 435
Host
 Arc::URL, 385
host
 Arc::URL, 389
HTTPOption
 Arc::URL, 385
HTTPOptions
 Arc::URL, 385
httpoptions
 Arc::URL, 389
ID
 Arc::DelegationConsumer, 129
 Arc::DelegationProviderSOAP, 136
IdPName
 Arc::UserConfig, 408
IDType
 Arc::Counter, 97
Import
 Arc::SecAttr, 326
increase_transfer_share
 DataStaging::TransferShares, 378
InfoCache
 Arc::InfoCache, 177
InfoFilter
 Arc::InfoFilter, 179
InfoRegisters
 Arc::InfoRegisters, 181
InformationContainer
 Arc::InformationContainer, 183
InformationInterface
 Arc::InformationInterface, 185
InformationRequest
 Arc::InformationRequest, 186
InformationResponse
 Arc::InformationResponse, 187
Init
 Arc::ArcLocation, 57
init_xmlsec
 Arc, 36
InitializeCredentials
 Arc::UserConfig, 408
InitProxyCertInfo
 Arc::Credential, 113

InquireRequest
 Arc::Credential, 114

Insert
 Arc::PayloadRaw, 270
 Arc::PayloadRawInterface, 273

INTERNAL_LOGIC_ERROR
 DataStaging::DTRErrorStatus, 146

INTERNAL_PROCESS_ERROR
 DataStaging::DTRErrorStatus, 146

IntraProcessCounter
 Arc::IntraProcessCounter, 190

inttostr
 Arc, 36, 37

ip6addr
 Arc::URL, 389

is_owner_
 Arc::XMLNode, 472

is_temporary_
 Arc::XMLNode, 472

isconnected
 Arc::Database, 120
 Arc::MySQLDatabase, 260

IsCredentialsValid
 Arc::Credential, 114

IsFinished
 Arc::JobState, 213

isOk
 Arc::MCC_Status, 240

isResolved
 Arc::SoftwareRequirement, 348

isSatisfied
 Arc::SoftwareRequirement, 348, 349

IsSecureProtocol
 Arc::URL, 385

istr
 Arc::Period, 285

istring_to_level
 Arc, 37

isValid
 Arc::Credential, 114

isValid
 Arc::CounterTicket, 104

Job
 Arc::Job, 195

JobControllerLoader
 Arc::JobControllerLoader, 205

JobDescriptionParserLoader
 Arc::JobDescriptionParserLoader, 211

JobDownloadDirectory

Arc::UserConfig, 410

JobListFile
 Arc::UserConfig, 410, 411

JobSupervisor
 Arc::JobSupervisor, 214

KeepStderr
 Arc::Run, 314

KeepStdin
 Arc::Run, 314

KeepStdout
 Arc::Run, 314

key
 Arc::AttributIterator, 61

KeyPassword
 Arc::UserConfig, 411, 412

KeyPath
 Arc::UserConfig, 412, 413

KeySize
 Arc::UserConfig, 413

Kill
 Arc::JobSupervisor, 217
 Arc::Run, 314

LDAPAttributes
 Arc::URL, 385

Idapattributes
 Arc::URL, 389

LDAPFilter
 Arc::URL, 385

Idapfilter
 Arc::URL, 389

LDAPScope
 Arc::URL, 385

Idapscope
 Arc::URL, 389

length
 Arc::PayloadRawBuf, 271

LESSTHAN
 Arc::Software, 338

LESSTHANOREQUAL
 Arc::Software, 338

Limit
 Arc::PayloadStream, 276
 Arc::PayloadStreamInterface, 280

Load
 Arc::ClientSOAP, 85

load
 Arc::BrokerLoader, 72
 Arc::JobControllerLoader, 205

Arc::JobDescriptionParserLoader, 212 LogLevel
Arc::ModuleManager, 257 Arc, 31
Arc::PluginsFactory, 295 LogMessage
Arc::SubmitterLoader, 358 Arc::LogMessage, 231
Arc::TargetRetrieverLoader, 368 LogStream
load_key_from_certfile Arc::LogStream, 233, 234
 Arc, 38
load_key_from_certstr Arc, 38
load_key_from_keyfile Arc, 38
load_trusted_cert_file Arc, 38
load_trusted_cert_str Arc, 38
load_trusted_certs Arc, 38
LoadConfigurationFile Arc::UserConfig, 414
Loader Arc::Loader, 222
LOCAL_FILE_ERROR DataStaging::DTRErrorStatus, 146
Locations Arc::URL, 386
locations Arc::URL, 389
lock Arc::SimpleCondition, 333
lock_ Arc::InformationInterface, 185
log Arc::LogFile, 225
 Arc::LogStream, 234
LogDestination Arc::LogDestination, 223
.LogError Arc::Credential, 114
 Arc::DelegationConsumer, 129
LogFile Arc::LogFile, 224, 225
LogFormat Arc, 31
Logger Arc::Logger, 227
 Arc::LogMessage, 232
logger Arc::MCC, 238
 Arc::Plexer, 290
 Arc::Service, 332
make_policy ArcSec::Policy, 297
make_request ArcSec::Request, 309
MakeConfig Arc::BaseConfig, 68
 Arc::MCCConfig, 241
makePersistent Arc::ModuleManager, 257
match Arc::RegularExpression, 307
MatchXMLName Arc, 38, 39
 Arc::XMLNode, 471
MatchXMLNamespace Arc, 39
 Arc::XMLNode, 471
max_duration_ Arc::DelegationContainerSOAP, 133
max_inactivity_time DataStaging::TransferParameters, 376
max_size_ Arc::DelegationContainerSOAP, 133
max_usage_ Arc::DelegationContainerSOAP, 133
MaxDiskSpace Arc::ExecutionTarget, 163
MaxMainMemory Arc::ExecutionTarget, 163
MaxVirtualMemory Arc::ExecutionTarget, 163
MCC Arc::MCC, 237
MCC_Status Arc::MCC_Status, 239
MCCLoader Arc::MCCLoader, 243
Message Arc::Message, 247, 248
MessageAttributes Arc::AttributeIterator, 62
 Arc::MessageAttributes, 250
MessageID

Arc::WSAHeader, 433
MetaData
 Arc::WSAEndpointReference, 431
MetaDataOption
 Arc::URL, 386
MetaDataOptions
 Arc::URL, 386
metadataoptions
 Arc::URL, 389
Migrate
 Arc::JobController, 203
 Arc::JobSupervisor, 217
 Arc::Submitter, 357
min_average_bandwidth
 DataStaging::TransferParameters, 376
min_current_bandwidth
 DataStaging::TransferParameters, 376
mkdirp
 Arc::FileAccess, 167
mkstemp
 Arc::FileAccess, 167
ModifyFoundTargets
 Arc::TargetGenerator, 363
ModuleManager
 Arc::ModuleManager, 257
Move
 Arc::XMLNode, 464
msg
 Arc::Logger, 228, 229
Name
 Arc::URLLocation, 392
 Arc::XMLNode, 465
name
 Arc::URLLocation, 392
Namespace
 Arc::XMLNode, 465
NamespacePrefix
 Arc::XMLNode, 465
Namespaces
 Arc::XMLNode, 465
NEW
 DataStaging::DTRStatus, 150
New
 Arc::XMLNode, 466
NewAttribute
 Arc::XMLNode, 466
NewChild
 Arc::XMLNode, 466, 467
NewReferenceParameter
 Arc::WSAHeader, 433
Next
 Arc::MCC, 237
 Arc::Plexer, 290
next_
 Arc::MCC, 238
NO_ERROR_LOCATION
 DataStaging::DTRErrorStatus, 146
Nodes
 Arc::XMLNodeContainer, 473
NON_CACHEABLE
 DataStaging, 46
NONE
 DataStaging::TransferShares, 378
NONE_ERROR
 DataStaging::DTRErrorStatus, 146
NOTEQUAL
 Arc::Software, 338
NULL_STATE
 DataStaging::DTRStatus, 151
OAuthConsumer
 Arc::OAuthConsumer, 264
open
 Arc::FileAccess, 167
opendir
 Arc::FileAccess, 167
OpenSSLInit
 Arc, 39
OperatingSystem
 Arc::ExecutionTarget, 164
operator AlgFactory *
 ArcSec::EvaluatorContext, 158
operator AttributeFactory *
 ArcSec::EvaluatorContext, 158
operator bool
 Arc::CStringValue, 81
 Arc::JobDescription, 207
 Arc::MCC_Status, 240
 Arc::MultiSecAttr, 259
 Arc::PathIterator, 268
 Arc::PayloadStream, 277
 Arc::PayloadStreamInterface, 280
 Arc::Run, 314
 Arc::SAMLToken, 322
 Arc::SecAttr, 326
 Arc::SecAttrValue, 327
 Arc::URL, 386
 Arc::UserConfig, 415
 Arc::UsernameToken, 427

Arc::WSRF, 436
Arc::X509Token, 459
Arc::XMLNode, 467
operator FnFactory *
 ArcSec::EvaluatorContext, 158
operator PluginsFactory *
 Arc::ChainContext, 74
operator std::string
 Arc::MCC_Status, 240
 Arc::Period, 285
 Arc::Software, 340
 Arc::Time, 372
 Arc::XMLNode, 467
operator XMLNode
 Arc::WSAEndpointReference, 431
 Arc::WSAHeader, 433
operator<
 Arc::ExpirationReminder, 165
 Arc::Period, 285
 Arc::Software, 341
 Arc::Time, 373
 Arc::URL, 386
operator<<
 Arc, 39
 Arc::LogMessage, 232
 Arc::Software, 344
 Arc::URL, 388
operator<=
 Arc::Period, 285
 Arc::Software, 342
 Arc::Time, 373
operator>
 Arc::Period, 285
 Arc::Software, 342
 Arc::Time, 374
operator>=
 Arc::Period, 286
 Arc::Software, 343
 Arc::Time, 374
operator*
 Arc::Attributeliterator, 61
 Arc::PathIterator, 268
operator()
 Arc::Software, 341
operator+
 Arc::Time, 373
operator++
 Arc::Attributeliterator, 61
 Arc::PathIterator, 268
 Arc::XMLNode, 467
operator->
 Arc::Attributeliterator, 61
operator--
 Arc::PathIterator, 268
 Arc::XMLNode, 468
operator=
 Arc::ExecutionTarget, 162
 Arc::Job, 195
 Arc::Message, 249
 Arc::Period, 285
 Arc::SoftwareRequirement, 350
 Arc::Time, 373
 Arc::WSAEndpointReference, 431
 Arc::XMLNode, 468
 Arc::XMLNodeContainer, 473
operator==
 Arc::Period, 285
 Arc::SecAttr, 326
 Arc::SecAttrValue, 328
 Arc::Software, 342
 Arc::Time, 373
 Arc::URL, 386
 Arc::XMLNode, 468
Option
 Arc::URL, 386
Options
 Arc::URL, 386
OptionString
 Arc::URL, 387
OtherAttributes
 Arc::JobDescription, 210
OutputCertificate
 Arc::Credential, 114
OutputCertificateChain
 Arc::Credential, 115
OutputPrivatekey
 Arc::Credential, 115
OutputPublickey
 Arc::Credential, 115
OverlayFile
 Arc::UserConfig, 416
Parent
 Arc::XMLNode, 469
Parse
 Arc::JobDescription, 207, 208
parse
 Arc::Config, 91

parseDN
 Arc::OAuthConsumer, 264
 Arc::SAML2SSOHTTPClient, 318
 ParseOptions
 Arc::URL, 387
 ParsePath
 Arc::URL, 387
 parsePolicy
 ArcSec::PolicyParser, 299
 parseVOMSAC
 Arc, 39, 40
 PARsing_Error
 Arc, 32
 passphrase_callback
 Arc, 41
 Passwd
 Arc::URL, 387
 passwd, 267
 Arc::URL, 389
 Password
 Arc::UserConfig, 416, 417
 PasswordType
 Arc::UsernameToken, 426
 Path
 Arc::URL, 387
 Arc::XMLNode, 469
 path
 Arc::URL, 389
 Path2BaseDN
 Arc::URL, 387
 PathIterator
 Arc::PathIterator, 267
 Payload
 Arc::Message, 249
 Arc::SOAPMessage, 336
 PayloadRaw
 Arc::PayloadRaw, 269
 PayloadSOAP
 Arc::PayloadSOAP, 274, 275
 PayloadStream
 Arc::PayloadStream, 276
 PayloadWSRF
 Arc::PayloadWSRF, 282
 Period
 Arc::Period, 284
 PERMANENT_REMOTE_ERROR
 DataStaging::DTRErrorStatus, 146
 plainstr
 Arc::URL, 387
 Plexer
 Arc::Plexer, 289
 plugins_table_name
 Arc, 43
 PluginsFactory
 Arc::PluginsFactory, 295
 Policy
 ArcSec::Policy, 296, 297
 PolicyStore
 ArcSec::PolicyStore, 299
 Port
 Arc::URL, 387
 port
 Arc::URL, 390
 Pos
 Arc::PayloadStream, 277
 Arc::PayloadStreamInterface, 280
 PossibleTargets
 Arc::Broker, 71
 PRE_CLEAN
 DataStaging::DTRStatus, 150
 PRE_CLEANED
 DataStaging::DTRStatus, 150
 PRE_CLEANING
 DataStaging::DTRStatus, 150
 PreFilterTargets
 Arc::Broker, 70
 Prefix
 Arc::XMLNode, 469
 Print
 Arc::ExecutionTarget, 162
 Arc::Job, 195
 Arc::JobDescription, 208
 print
 Arc::Adler32Sum, 53
 Arc::CheckSum, 76
 Arc::CheckSumAny, 79
 Arc::Config, 91
 Arc::CRC32Sum, 106
 Arc::MD5Sum, 245
 PrintJobStatus
 Arc::JobController, 203
 PrintTargetInfo
 Arc::TargetGenerator, 363
 process
 Arc::ClientHTTPwithSAML2SSO, 83
 Arc::ClientSOAP, 85
 Arc::ClientSOAPwithSAML2SSO, 86
 Arc::MCC, 237
 Arc::MCCIInterface, 242
 Arc::Plexer, 290

GDS20::GDS20Service, 172
PROCESS_CACHE
 DataStaging::DTRStatus, 150
processConsent
 Arc::HakaClient, 176
 Arc::OpenIdpClient, 266
 Arc::SAML2SSOHTTPClient, 318
processIdP2Confusa
 Arc::HakaClient, 176
 Arc::OpenIdpClient, 266
 Arc::SAML2SSOHTTPClient, 318
processIdPLogin
 Arc::HakaClient, 176
 Arc::OpenIdpClient, 266
 Arc::SAML2SSOHTTPClient, 318
PROCESSING_CACHE
 DataStaging::DTRStatus, 151
processLogin
 Arc::OAuthConsumer, 265
 Arc::SAML2LoginClient, 317
 Arc::SAML2SSOHTTPClient, 318
ProcessSecHandlers
 Arc::MCC, 237
 Arc::Service, 332
Protocol
 Arc::URL, 387
protocol
 Arc::URL, 390
PROTOCOL_RECOGNIZED_ERROR
 Arc, 32
ProxyPath
 Arc::UserConfig, 417, 418
PullStatus
 DataStaging::DataDeliveryComm, 123
pushCSR
 Arc::OAuthConsumer, 265
 Arc::SAML2SSOHTTPClient, 318
Put
 Arc::PayloadStream, 277
 Arc::PayloadStreamInterface, 280, 281
Query
 Arc::Query, 303
QUERY_REPLICA
 DataStaging::DTRStatus, 150
QUERYING_REPLICA
 DataStaging::DTRStatus, 150
ReadAllJobsFromFile
 Arc::Job, 195
 ReadFromFile
 Arc::XMLNode, 469
 ReadFromStream
 Arc::XMLNode, 470
 ReadJobIDsFromFile
 Arc::Job, 196
 ReadStderr
 Arc::Run, 315
 ReadStdout
 Arc::Run, 315
receiveDTR
 DataStaging::DataDelivery, 121
 DataStaging::DTRCallback, 144
 DataStaging::Generator, 174
 DataStaging::Processor, 301
 DataStaging::Scheduler, 323
ReferenceParameter
 Arc::WSAHeader, 434
ReferenceParameters
 Arc::WSAEndpointReference, 431
REGISTER_REPLICA
 DataStaging::DTRStatus, 150
registerCallback
 DataStaging::DTR, 143
RegisteredService
 Arc::RegisteredService, 306
REGISTERING_REPLICA
 DataStaging::DTRStatus, 151
registration
 Arc::InfoRegistrar, 182
RegistrationCollector
 Arc::Service, 332
RelatesTo
 Arc::WSAHeader, 434
RelationshipType
 Arc::WSAHeader, 434
release
 Arc::FileLock, 169
RELEASE_REQUEST
 DataStaging::DTRStatus, 150
RELEASING_REQUEST
 DataStaging::DTRStatus, 151
reload
 Arc::ModuleManager, 258
remove
 Arc::MessageAttributes, 251
removeAll
 Arc::MessageAttributes, 252
RemoveHTTPOption
 Arc::URL, 387

RemoveJobsFromFile
 Arc::Job, 197

RemoveMetaDataTable
 Arc::URL, 388

RemoveOption
 Arc::URL, 388

removeService
 Arc::InfoRegisterContainer, 181

Renew
 Arc::JobSupervisor, 218

Replace
 Arc::XMLNode, 470

REPLICA_QUERIED
 DataStaging::DTRStatus, 150

REPLICA_REGISTERED
 DataStaging::DTRStatus, 151

ReplyTo
 Arc::WSAHeader, 434

report
 Arc::PluginsFactory, 295

Request
 Arc::DelegationConsumer, 130
 ArcSec::Request, 308

REQUEST_RELEASED
 DataStaging::DTRStatus, 151

RequestAttribute
 ArcSec::RequestAttribute, 310

RequestItem
 ArcSec::RequestItem, 310

reserve
 Arc::Counter, 101
 Arc::IntraProcessCounter, 192

reset
 Arc::SimpleCondition, 333
 DataStaging::DTR, 143

RESOLVE
 DataStaging::DTRStatus, 150

RESOLVED
 DataStaging::DTRStatus, 150

RESOLVING
 DataStaging::DTRStatus, 150

Rest
 Arc::PathIterator, 268

Restore
 Arc::DelegationConsumer, 130

Resubmit
 Arc::JobSupervisor, 219

Result
 Arc::InformationResponse, 187
 Arc::Run, 315

Resume
 Arc::JobSupervisor, 220

RetrieveExecutionTargets
 Arc::TargetGenerator, 363

RetrieveJobs
 Arc::TargetGenerator, 364

ROLE
 DataStaging::TransferShares, 378

Run
 Arc::Run, 313

Running
 Arc::Run, 315

Same
 Arc::XMLNode, 470

SAML2LoginClient
 Arc::SAML2LoginClient, 316

SAMLToken
 Arc::SAMLToken, 321

SAMLVersion
 Arc::SAMLToken, 320

save
 Arc::Config, 91

SaveJobStatusToStream
 Arc::JobController, 204

SaveTargetInfoToStream
 Arc::TargetGenerator, 364

SaveToFile
 Arc::UserConfig, 418
 Arc::XMLNode, 470

SaveToStream
 Arc::ExecutionTarget, 162
 Arc::Job, 197
 Arc::JobDescription, 209
 Arc::XMLNode, 470

scan
 Arc::Adler32Sum, 53
 Arc::CheckSum, 76
 Arc::CheckSumAny, 79
 Arc::CRC32Sum, 106
 Arc::MD5Sum, 246
 Arc::PluginsFactory, 295

Scope
 Arc::URL, 382

sechandlers_
 Arc::MCC, 238
 Arc::Service, 332

seekable_
 Arc::PayloadStream, 278

selectSoftware

Arc::SoftwareRequirement, 350, 351
SELF_REPLICATION_ERROR
 DataStaging::DTRErrorStatus, 146
SelfSignEECRequest
 Arc::Credential, 115
Service
 Arc::Service, 331
ServiceCounter
 Arc::TargetGenerator, 364
SESSION_CLOSE
 Arc, 32
SessionDiskSpace
 Arc::DiskSpaceRequirementType, 138
Set
 Arc::XMLNode, 470
set
 Arc::MessageAttributes, 252
set_error_status
 DataStaging::DTR, 143
set_namespaces
 Arc::WSRF, 436
 Arc::WSRFBaseFault, 438
 Arc::WSRP, 441
setAttributeFactory
 ArcSec::Request, 309
setBackups
 Arc::LogFile, 225
setCfg
 Arc::ModuleManager, 258
setCombiningAlg
 ArcSec::Evaluator, 157
setEvalResult
 ArcSec::Policy, 298
setEvaluatorContext
 ArcSec::Policy, 298
setExcess
 Arc::Counter, 102
 Arc::IntraProcessCounter, 193
setFileName
 Arc::Config, 92
SetFormat
 Arc::Time, 374
setIdentifier
 Arc::LogMessage, 232
SetLifeTime
 Arc::Credential, 115
setLimit
 Arc::Counter, 102
 Arc::IntraProcessCounter, 193
setMaxSize
 Arc::LogFile, 225
SetPeriod
 Arc::Period, 286
SetProxyPolicy
 Arc::Credential, 116
setReopen
 Arc::LogFile, 225
setRequestItems
 ArcSec::Request, 309
SetStartTime
 Arc::Credential, 116
setThreadContext
 Arc::Logger, 229
setThreshold
 Arc::Logger, 229
setThresholdForDomain
 Arc::Logger, 229, 230
SetTime
 Arc::Time, 374
setuid
 Arc::FileAccess, 167
SetUser
 Arc::UserConfig, 418
ShareType
 DataStaging::TransferShares, 378
shutdown
 Arc::Database, 120
 Arc::MySQLDatabase, 261
signal
 Arc::SimpleCondition, 334
signal_nonblock
 Arc::SimpleCondition, 334
SignEECRequest
 Arc::Credential, 116
SignNode
 Arc::XMLSecNode, 475
SignRequest
 Arc::Credential, 116, 117
Size
 Arc::PayloadRaw, 270
 Arc::PayloadRawInterface, 273
 Arc::PayloadStream, 278
 Arc::PayloadStreamInterface, 281
 Arc::XMLNode, 470
 Arc::XMLNodeContainer, 474
size
 Arc::PayloadRawBuf, 271
SLCS
 Arc::UserConfig, 419
SOAP

Arc::InformationRequest, 186
 Arc::WSRF, 436
 SOAPMessage
 Arc::SOAPMessage, 335
 Software
 Arc::Software, 338, 339
 SoftwareRequirement
 Arc::SoftwareRequirement, 345, 346
 SortTargets
 Arc::Broker, 70
 Source
 ArcSec::Source, 352
 STACK_OF
 Arc::Credential, 117
 STAGE_PREPARE
 DataStaging::DTRStatus, 150
 STAGED_PREPARED
 DataStaging::DTRStatus, 150
 STAGING_PREPARING
 DataStaging::DTRStatus, 150
 STAGING_PREPARING_WAIT
 DataStaging::DTRStatus, 150
 STAGING_TIMEOUT_ERROR
 DataStaging::DTRErrorStatus, 146
 Start
 Arc::Run, 315
 start
 Arc::Adler32Sum, 53
 Arc::CheckSum, 77
 Arc::CheckSumAny, 79
 Arc::CRC32Sum, 106
 Arc::MD5Sum, 246
 DataStaging::Processor, 301
 DataStaging::Scheduler, 323
 STATUS_OK
 Arc, 32
 StatusKind
 Arc, 32
 stop
 DataStaging::Processor, 301
 DataStaging::Scheduler, 324
 storeCert
 Arc::OAuthConsumer, 265
 Arc::SAML2SSOHTTPClient, 319
 StoreDirectory
 Arc::UserConfig, 419, 420
 str
 Arc::Time, 374
 Arc::URL, 388
 Arc::URLLocation, 392
 string
 Arc, 41
 strtoint
 Arc, 41
 Submit
 Arc::Submitter, 357
 SubmitterLoader
 Arc::SubmitterLoader, 358
 Swap
 Arc::XMLNode, 470
 SYSCONFIG
 Arc::UserConfig, 424
 SYSCONFIGARCLOC
 Arc::UserConfig, 425
 TargetGenerator
 Arc::TargetGenerator, 360
 TargetRetriever
 Arc::TargetRetriever, 365
 TargetRetrieverLoader
 Arc::TargetRetrieverLoader, 367
 TEMPORARY_REMOTE_ERROR
 DataStaging::DTRErrorStatus, 146
 thread_stacksize
 Arc, 43
 ThreadDataItem
 Arc::ThreadDataItem, 369
 Time
 Arc::Time, 372
 Timeout
 Arc::PayloadStream, 278
 Arc::PayloadStreamInterface, 281
 Arc::UserConfig, 420
 TimeStamp
 Arc, 42
 TmpDirCreate
 Arc, 42
 TmpFileCreate
 Arc, 42
 To
 Arc::WSAHeader, 434
 toString
 Arc::Software, 343
 ToXML
 Arc::Job, 198
 TRANSFER
 DataStaging::DTRStatus, 150
 TRANSFER_SPEED_ERROR
 DataStaging::DTRErrorStatus, 146
 TRANSFER_WAIT

DataStaging::DTRStatus, 150
TRANSFERRED
 DataStaging::DTRStatus, 151
TRANSFERRING
 DataStaging::DTRStatus, 150
TRANSFERRING_CANCEL
 DataStaging::DTRStatus, 151
Truncate
 Arc::PayloadRaw, 270
 Arc::PayloadRawInterface, 273
TryLoad
 Arc::PluginsFactory, 295
UNKNOWN_SERVICE_ERROR
 Arc, 32
Unlink
 Arc::MCC, 237
unload
 Arc::ModuleManager, 258
unlock
 Arc::SimpleCondition, 334
UnParse
 Arc::JobDescription, 209
Update
 Arc::ExecutionTarget, 162
UpdateCredentials
 Arc::DelegationConsumerSOAP, 131
 Arc::DelegationContainerSOAP, 132
 Arc::DelegationProviderSOAP, 136
uri_encode
 Arc, 42
URL
 Arc::URL, 382
urlencode
 Arc::ConfusaParserUtils, 94
urlencode_params
 Arc::ConfusaParserUtils, 94
URLLocation
 Arc::URLLocation, 391
urloptions
 Arc::URL, 390
USER
 DataStaging::TransferShares, 378
UserConfig
 Arc::UserConfig, 396, 397
UserName
 Arc::UserConfig, 421
Username
 Arc::URL, 388
 Arc::UsernameToken, 427
 username
 Arc::URL, 390
 UsernameToken
 Arc::UsernameToken, 426
 UtilsDirPath
 Arc::UserConfig, 421, 422
 valid
 Arc::URL, 390
 valid_
 Arc::WSRF, 437
 Validate
 Arc::XMLNode, 470
 Verbosity
 Arc::UserConfig, 422
 VerifyNode
 Arc::XMLSecNode, 476
VERSIONTOKENS
 Arc::Software, 344
VO
 DataStaging::TransferShares, 378
VOMSDecode
 Arc, 42
VOMSESPath
 Arc::UserConfig, 423
VOMSTrustList
 Arc::VOMSTrustList, 428, 429
Wait
 Arc::Run, 315
wait
 Arc::SimpleCondition, 334
 Arc::SimpleCounter, 334
wait_nonblock
 Arc::SimpleCondition, 334
WaitForExit
 Arc::ThreadRegistry, 371
WaitOrCancel
 Arc::ThreadRegistry, 371
WriteJobIDsToFile
 Arc::Job, 198
WriteJobIDToFile
 Arc::Job, 198
WriteJobsToFile
 Arc::Job, 199, 200
WriteJobsToTruncatedFile
 Arc::Job, 200
WriteStdin
 Arc::Run, 315
WSAEndpointReference

Arc::WSAEndpointReference, 430
WSAFault
 Arc, 32
WSAFaultAssign
 Arc, 42
WSAFaultExtract
 Arc, 43
WSAFaultInvalidAddressingHeader
 Arc, 32
WSAFaultUnknown
 Arc, 32
WSAHeader
 Arc::WSAHeader, 432
WSRF
 Arc::WSRF, 436
WSRFBaseFault
 Arc::WSRFBaseFault, 438
WSRP
 Arc::WSRP, 441
WSRPFault
 Arc::WSRPFault, 443
WSRPRResourcePropertyChangeFailure
 Arc::WSRPRResourcePropertyChangeFailure,
 451
X509Token
 Arc::X509Token, 457, 458
X509TokenType
 Arc::X509Token, 457
XMLNode
 Arc::XMLNode, 462
XMLNodeContainer
 Arc::XMLNodeContainer, 473
XMLSecNode
 Arc::XMLSecNode, 475
XPathLookup
 Arc::XMLNode, 471