# Hosting Environment (Daemon) Chain Components

Generated by Doxygen 1.6.1

Wed Oct 24 16:26:14 2012

# Contents

# Chapter 1

# Namespace Index

## 1.1 Namespace List

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

# Chapter 2

# Data Structure Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Data Structure Index

## 3.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 4

# Namespace Documentation

## 4.1 ArcSec Namespace Reference

ArcRequest, Parsing the specified Arc request format.

### Data Structures

- class AllowPDP

  *This PDP always return true (allow).*

- class ArcAuthZ

  *Tests message against list of PDPs.*

- class ArcAlgFactory

  *Algorithm factory class for Arc.*

- class ArcAttributeFactory

  *Attribute factory class for Arc specified attributes.*

- class ArcAttributeProxy

  *Arc specific AttributeProxy class.*

- class ArcRequestTuple

  *RequestTuple, container which includes the.*

- class ArcEvaluationCtx

  *EvaluationCtx, in charge of storing some context information for evaluation, including Request, current time, etc.*

- class ArcEvaluator

  *Execute the policy evaluation, based on the request and policy.*

- class ArcFnFactory

  *Function factory class for Arc specified attributes.*

- class ArcPDP

  *ArcPDP - PDP which can handle the Arc specific request and policy schema.*

- class ArcPolicy

  *ArcPolicy class to parse and operate Arc specific <Policy> node.*

- class ArcRequest
- class ArcRequestItem

  *Container, <Subjects, Actions, Objects, Contexts> tuple.*

- class ArcRule

  *ArcRule class to parse Arc specific <Rule> node.*

- class DelegationPDP
- class DelegationSH
- class DenyPDP

  *This PDP always returns false (deny).*

- class GACLEvaluator
- class GACLPDP
- class GACLPolicy
- class GACLRequest
- class PDPServiceInvoker

  *PDPServiceInvoker - client which will invoke pdpservice.*

- class SAML2SSO_AssertionConsumerSH

  *Implement the funcionality of the Service Provider in SAML2 SSO profile.*

- class SAMLTokenSH

  *Adds WS-Security SAML Token into SOAP Header.*

- class SimpleListPDP

  *Tests X509 subject against list of subjects in file.*

- class UsernameTokenSH

  *Adds WS-Security Username Token into SOAP Header.*

- class X509TokenSH

  *Adds WS-Security X509 Token into SOAP Header.*

- class AttributeDesignator
- class AttributeSelector
- class XACMLAlgFactory

  *Algorithm factory class for XACML.*

- class XACMLApply
- class XACMLAttributeFactory

  *Attribute factory class for XACML specified attributes.*

- class XACMLAttributeProxy

*XACML specific AttributeProxy class.*

- class XACMLCondition

  *XACMLCondition class to parse and operate XACML specific <Condition> node.*

- class XACMLEvaluationCtx

  *EvaluationCtx, in charge of storing some context information for evaluation, including Request, current time, etc.*

- class XACMLEvaluator

  *Execute the policy evaluation, based on the request and policy.*

- class XACMLFnFactory

  *Function factory class for XACML specified attributes.*

- class XACMLPDP

  *XACMLPDP - PDP which can handle the XACML specific request and policy schema.*

- class XACMLPolicy

  *XACMLPolicy class to parse and operate XACML specific <Policy> node.*

- class XACMLRequest
- class XACMLRule

  *XACMLRule class to parse XACML specific <Rule> node.*

- class XACMLTargetMatch
- class XACMLTargetMatchGroup
- class XACMLTargetSection
- class XACMLTarget

  *XACMLTarget class to parse and operate XACML specific <Target> node.*

## Typedefs

- typedef std::pair< AttributeValue ∗, Function ∗ > Match
- typedef std::list< Match > AndList
- typedef std::list< AndList > OrList

## 4.1.1 Detailed Description

ArcRequest, Parsing the specified Arc request format. XACMLRequest, Parsing the xacml request format.

## 4.1.2 Typedef Documentation

### 4.1.2.1 typedef std::list<Match> ArcSec::AndList

AndList - include items inside one <Subject> (or <Resource> <Action> <Condition>). "And" relationship means the request should satisfy all of the items <Subject> <SubFraction type="X500DN">/O=Grid/OU=KnowARC/CN=XYZ</SubFraction>

<SubFraction type="ShibName">urn:mace:shibboleth:examples</SubFraction> </Subject> "Or" relationship meand the request should satisfy any of the items <Subjects> <Subject type="X500DN">/O=Grid/OU=KnowARC/CN=ABC</Subject> <Subject type="VOMSAttribute">/vo.knowarc/usergroupA</Subject> <Subject> <SubFraction type="X500DN">/O=Grid/OU=KnowARC/CN=XYZ</SubFraction> <SubFraction type="ShibName">urn:mace:shibboleth:examples</SubFraction> </Subject> <GroupIdRef location="./subjectgroup.xml">subgrpexample1</GroupIdRef> </Subjects>

#### 4.1.2.2 typedef std::pair<AttributeValue∗, Function∗> ArcSec::Match

Pair Match include the AttributeValue object in <Rule> and the Function which is used to handle the AttributeValue, default function is "Equal", if some other function is used, it should be explicitly specified, e.g. Subject Type="string" Function="Match">/vo.knowarc/usergroupA</Subject> Subjects> example inside <Rule>: <Subjects> <Subject type="X500Name">/O=NorduGrid/OU=UIO/CN=test</Subject> <Subject type="string">/vo.knowarc/usergroupA</Subject> <Subject> <SubFraction type="string">/O=Grid/OU=KnowARC/CN=XYZ</SubFraction> <SubFraction type="string">urn:mace:shibboleth:examples</SubFraction> </Subject> <GroupIdRef location="./subjectgroup.xml">subgrpexample1</GroupIdRef> </Subjects>

# Chapter 5

# Data Structure Documentation

## 5.1 ArcSec::AllowPDP Class Reference

This PDP always return true (allow).

```
#include <AllowPDP.h>
```

### 5.1.1 Detailed Description

This PDP always return true (allow).

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

- AllowPDP.h

## 5.2 ArcSec::ArcAlgFactory Class Reference

Algorithm factory class for Arc.

`#include <ArcAlgFactory.h>`

### Public Member Functions

- virtual CombiningAlg ∗ createAlg (const std::string &type)

### 5.2.1 Detailed Description

Algorithm factory class for Arc.

### 5.2.2 Member Function Documentation

#### 5.2.2.1 virtual CombiningAlg∗ ArcSec::ArcAlgFactory::createAlg (const std::string & *type*) `[virtual]`

return a Alg object according to the "CombiningAlg" attribute in the <Policy> node; The ArcAlgFactory itself will release the Alg objects

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

- ArcAlgFactory.h

## 5.3 ArcSec::ArcAttributeFactory Class Reference

Attribute factory class for Arc specified attributes.

```
#include <ArcAttributeFactory.h>
```

### Public Member Functions

- virtual AttributeValue ∗ createValue (const Arc::XMLNode &node, const std::string &type)

### 5.3.1 Detailed Description

Attribute factory class for Arc specified attributes.

### 5.3.2 Member Function Documentation

#### 5.3.2.1 virtual AttributeValue∗ ArcSec::ArcAttributeFactory::createValue (const Arc::XMLNode & *node*, const std::string & *type*) `[virtual]`

creat a AttributeValue according to the value in the XML node and the type; It should be the caller to release the AttributeValue Object

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

- ArcAttributeFactory.h

## 5.4 ArcSec::ArcAttributeProxy< TheAttribute > Class Template Reference

Arc specific AttributeProxy class.

`#include <ArcAttributeProxy.h>`

### Public Member Functions

- virtual AttributeValue ∗ getAttribute (const Arc::XMLNode &node)

### 5.4.1 Detailed Description

**template<class TheAttribute> class ArcSec::ArcAttributeProxy< TheAttribute >**

Arc specific AttributeProxy class.

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

- ArcAttributeProxy.h

## 5.5 ArcSec::ArcAuthZ Class Reference

Tests message against list of PDPs.

`#include <ArcAuthZ.h>`

### Data Structures

- class **PDPDesc**

### Public Member Functions

- virtual bool Handle (Arc::Message ∗msg) const

### Protected Member Functions

- bool MakePDPs (Arc::XMLNode cfg)

### 5.5.1 Detailed Description

Tests message against list of PDPs. This class implements SecHandler interface. It's Handle() method runs provided Message instance against all PDPs specified in configuration. If any of PDPs returns positive result Handle() return true, otherwise false. This class is the main entry for configuring authorization, and could include different PDP configured inside.

### 5.5.2 Member Function Documentation

#### 5.5.2.1 virtual bool ArcSec::ArcAuthZ::Handle (Arc::Message ∗ *msg*) const `[virtual]`

Get authorization decision

#### 5.5.2.2 bool ArcSec::ArcAuthZ::MakePDPs (Arc::XMLNode *cfg*) `[protected]`

Create PDP according to conf info

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

- ArcAuthZ.h

## 5.6 ArcSec::ArcEvaluationCtx Class Reference

EvaluationCtx, in charge of storing some context information for evaluation, including Request, current time, etc.

```
#include <ArcEvaluationCtx.h>
```

### Public Member Functions

- ArcEvaluationCtx (Request ∗request)
- virtual void split ()

### 5.6.1 Detailed Description

EvaluationCtx, in charge of storing some context information for evaluation, including Request, current time, etc.

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 ArcSec::ArcEvaluationCtx::ArcEvaluationCtx (Request ∗ *request*)

Construct a new EvaluationCtx based on the given request

### 5.6.3 Member Function Documentation

#### 5.6.3.1 virtual void ArcSec::ArcEvaluationCtx::split () `[virtual]`

Convert/split one RequestItem ( one tuple <SubList, ResList, ActList, CtxList>) into a few <Subject, Resource, Action, Context> tuples. The purpose is for evaluation. The evaluator will evaluate each RequestTuple one by one, not the RequestItem because it includes some independent <Subject, Resource, Action, Context>s and the evaluator should deal with them independently.

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

- ArcEvaluationCtx.h

# 5.7 ArcSec::ArcEvaluator Class Reference

Execute the policy evaluation, based on the request and policy.

`#include <ArcEvaluator.h>`

## Public Member Functions

- virtual Response ∗ evaluate (Request ∗request)

## 5.7.1 Detailed Description

Execute the policy evaluation, based on the request and policy.

## 5.7.2 Member Function Documentation

### 5.7.2.1 virtual Response∗ ArcSec::ArcEvaluator::evaluate (Request ∗ *request*) `[virtual]`

Evaluate the request based on the policy information inside PolicyStore

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

- ArcEvaluator.h

# 5.8 ArcSec::ArcFnFactory Class Reference

Function factory class for Arc specified attributes.

```
#include <ArcFnFactory.h>
```

## Public Member Functions

- virtual Function ∗ createFn (const std::string &type)

## 5.8.1 Detailed Description

Function factory class for Arc specified attributes.

## 5.8.2 Member Function Documentation

### 5.8.2.1 virtual Function∗ ArcSec::ArcFnFactory::createFn (const std::string & *type*) `[virtual]`

return a Function object according to the "Function" attribute in the XML node; The ArcFnFactory itself will release the Function objects

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

- ArcFnFactory.h

## 5.9 ArcSec::ArcPDP Class Reference

ArcPDP - PDP which can handle the Arc specific request and policy schema.

```
#include <ArcPDP.h>
```

### 5.9.1 Detailed Description

ArcPDP - PDP which can handle the Arc specific request and policy schema.

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

- ArcPDP.h

## 5.10 ArcSec::ArcPolicy Class Reference

ArcPolicy class to parse and operate Arc specific <Policy> node.

`#include <ArcPolicy.h>`

### Public Member Functions

- ArcPolicy (Arc::PluginArgument ∗parg)
- ArcPolicy (const Arc::XMLNode node, Arc::PluginArgument ∗parg)
- ArcPolicy (const Arc::XMLNode node, EvaluatorContext ∗ctx, Arc::PluginArgument ∗parg)
- virtual void make_policy ()

### 5.10.1 Detailed Description

ArcPolicy class to parse and operate Arc specific <Policy> node.

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 ArcSec::ArcPolicy::ArcPolicy (Arc::PluginArgument ∗ *parg*)

Constructor

#### 5.10.2.2 ArcSec::ArcPolicy::ArcPolicy (const Arc::XMLNode *node*, Arc::PluginArgument ∗ *parg*)

Constructor

#### 5.10.2.3 ArcSec::ArcPolicy::ArcPolicy (const Arc::XMLNode *node*, EvaluatorContext ∗ *ctx*, Arc::PluginArgument ∗ *parg*)

Constructor

### 5.10.3 Member Function Documentation

#### 5.10.3.1 virtual void ArcSec::ArcPolicy::make_policy () `[virtual]`

Parse XMLNode, and construct the low-level Rule object

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

- ArcPolicy.h

## 5.11 ArcSec::ArcRequest Class Reference

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

- ArcRequest.h

# 5.12 ArcSec::ArcRequestItem Class Reference

Container, <Subjects, Actions, Objects, Contexts> tuple.

```
#include <ArcRequestItem.h>
```

## 5.12.1 Detailed Description

Container, <Subjects, Actions, Objects, Contexts> tuple. Specified ArcRequestItem which can parse Arc request formate

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

- ArcRequestItem.h

## 5.13 ArcSec::ArcRequestTuple Class Reference

RequestTuple, container which includes the.

```
#include <ArcEvaluationCtx.h>
```

### 5.13.1 Detailed Description

RequestTuple, container which includes the.

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

- ArcEvaluationCtx.h

## 5.14 ArcSec::ArcRule Class Reference

ArcRule class to parse Arc specific <Rule> node.

```
#include <ArcRule.h>
```

### 5.14.1 Detailed Description

ArcRule class to parse Arc specific <Rule> node.

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

- ArcRule.h

## 5.15 ArcSec::AttributeDesignator Class Reference

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

- AttributeDesignator.h

## 5.16 ArcSec::AttributeSelector Class Reference

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

- AttributeSelector.h

## 5.17 ArcSHCLegacy::AuthUser Class Reference

### Data Structures

- class **group_t**
- struct **source_t**

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

- auth.h

## 5.18 ArcSHCLegacy::AuthVO Class Reference

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

- auth.h

## 5.19 ArcSHCLegacy::ConfigParser Class Reference

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

- ConfigParser.h

## 5.20 ArcMCCTLS::ConfigTLSMCC Class Reference

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

- ConfigTLSMCC.h

## 5.21 Arc::DataPointARC Class Reference

```
#include <DataPointARC.h>
```

### 5.21.1 Detailed Description

Provides an interface to the Chelonia storage system developed by ARC.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointARC.h

## 5.22 Arc::DataPointFile Class Reference

```
#include <DataPointFile.h>
```

### 5.22.1 Detailed Description

This class allows access to the regular local filesystem through the same interface as is used for remote storage on the grid.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointFile.h

## 5.23 Arc::DataPointGFAL Class Reference

```
#include <DataPointGFAL.h>
```

### 5.23.1 Detailed Description

Provides access to the gLite Grid File Access Library through ARC's API. Only rfio:// protocol is supported at the moment.

Notes on env variables:

- If LFC plugin is installed LFC_HOST must be set (lcgutils bug 322)
- If SRM is used LCG_GFAL_INFOSYS must be set to BDII host:port

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointGFAL.h

## 5.24 Arc::DataPointGridFTP Class Reference

```
#include <DataPointGridFTP.h>
```

### Data Structures

- class **CBArg**

### 5.24.1 Detailed Description

GridFTP is essentially the FTP protocol with GSI security. This class uses libraries from the Globus Toolkit. It can also be used for regular FTP.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointGridFTP.h

## 5.25 Arc::DataPointHTTP Class Reference

```
#include <DataPointHTTP.h>
```

### 5.25.1 Detailed Description

This class allows access through HTTP to remote resources. HTTP over SSL (HTTPS) and HTTP over GSI (HTTPG) are also supported.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointHTTP.h

## 5.26 Arc::DataPointLDAP Class Reference

```
#include <DataPointLDAP.h>
```

### 5.26.1 Detailed Description

LDAP is used in grids mainly to store information about grid services or resources rather than to store data itself. This class allows access to LDAP data through the same interface as other grid resources.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointLDAP.h

## 5.27 Arc::DataPointLFC Class Reference

```
#include <DataPointLFC.h>
```

### 5.27.1 Detailed Description

The LCG File Catalog (LFC) is a replica catalog developed by CERN. It consists of a hierarchical namespace of grid files and each filename can be associated with one or more physical locations.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointLFC.h

## 5.28 Arc::DataPointRLS Class Reference

```
#include <DataPointRLS.h>
```

### 5.28.1 Detailed Description

The Replica Location Service (RLS) is a replica catalog developed by Globus. It maps filenames in a flat namespace to one or more physical locations, and can also store meta-information on each file. This class uses the Globus Toolkit libraries for accessing RLS.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointRLS.h

## 5.29 Arc::DataPointSRM Class Reference

```
#include <DataPointSRM.h>
```

### 5.29.1 Detailed Description

The Storage Resource Manager (SRM) protocol allows access to data distributed across physical storage through a unified namespace and management interface. PrepareReading() or PrepareWriting() must be used before reading or writing a physical file.

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointSRM.h

## 5.30 Arc::DataPointXrootd Class Reference

```
#include <DataPointXrootd.h>
```

### 5.30.1 Detailed Description

xrootd is a protocol for data access across large scale storage clusters. More information can be found at `http://xrootd.slac.stanford.edu/`

This class is a loadable module and cannot be used directly. The DataHandle class loads modules at runtime and should be used instead of this.

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

- DataPointXrootd.h

## 5.31 ArcMCCTLSSec::DelegationCollector Class Reference

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

- DelegationCollector.h

## 5.32 ArcMCCTLSSec::DelegationMultiSecAttr Class Reference

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

- DelegationSecAttr.h

## 5.33 ArcSec::DelegationPDP Class Reference

```
#include <DelegationPDP.h>
```

### 5.33.1 Detailed Description

DeleagtionPDP - PDP which can handle the Arc specific request and policy provided as identity delegation policy.

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

- DelegationPDP.h

## 5.34 ArcMCCTLSSec::DelegationSecAttr Class Reference

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

- DelegationSecAttr.h

## 5.35 ArcSec::DelegationSH Class Reference

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

- DelegationSH.h

## 5.36 ArcSec::DenyPDP Class Reference

This PDP always returns false (deny).

```
#include <DenyPDP.h>
```

### 5.36.1 Detailed Description

This PDP always returns false (deny).

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

- DenyPDP.h

# 5.37 ArcSec::GACLEvaluator Class Reference

## Public Member Functions

- virtual Response ∗ evaluate (Request ∗request)

## 5.37.1 Member Function Documentation

### 5.37.1.1 virtual Response∗ ArcSec::GACLEvaluator::evaluate (Request ∗ *request*) `[virtual]`

Evaluate the request based on the policy information inside PolicyStore

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

- GACLEvaluator.h

## 5.38 ArcSec::GACLPDP Class Reference

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

- GACLPDP.h

## 5.39 ArcSec::GACLPolicy Class Reference

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

- GACLPolicy.h

## 5.40 ArcSec::GACLRequest Class Reference

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

- GACLRequest.h

# 5.41 Arc::LDAPQuery Class Reference

```
#include <LDAPQuery.h>
```

## Public Member Functions

- LDAPQuery (const std::string &ldaphost, int ldapport, int timeout, bool anonymous=true, const std::string &usersn="")
- ∼LDAPQuery ()
- bool Query (const std::string &base, const std::string &filter="(objectclass=∗)", const std::list< std::string > &attributes=std::list< std::string >(), URL::Scope scope=URL::subtree)
- bool Result (ldap_callback callback, void ∗ref)

## 5.41.1 Detailed Description

LDAPQuery class; querying of LDAP servers.

## 5.41.2 Constructor & Destructor Documentation

### 5.41.2.1 Arc::LDAPQuery::LDAPQuery (const std::string & *ldaphost*, int *ldapport*, int *timeout*, bool *anonymous* = `true`, const std::string & *usersn* = `""`)

Constructs a new LDAPQuery object and sets connection options. The connection is first established when calling Query.

### 5.41.2.2 Arc::LDAPQuery::∼LDAPQuery ()

Destructor. Will disconnect from the ldapserver if still connected.

## 5.41.3 Member Function Documentation

### 5.41.3.1 bool Arc::LDAPQuery::Query (const std::string & *base*, const std::string & *filter* = `"(objectclass=∗)"`, const std::list< std::string > & *attributes* = `std::list< std::string >()`, URL::Scope *scope* = `URL::subtree`)

Queries the ldap server.

### 5.41.3.2 bool Arc::LDAPQuery::Result (ldap_callback *callback*, void ∗ *ref*)

Retrieves the result of the query from the ldap-server.

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

- LDAPQuery.h

## 5.42 ArcSHCLegacy::LegacyMap Class Reference

### Data Structures

- class **cfgfile**

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

- LegacyMap.h

## 5.43 ArcSHCLegacy::LegacyPDP Class Reference

### Data Structures

- class **cfgfile**

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

- LegacyPDP.h

## 5.44 ArcSHCLegacy::LegacySecAttr Class Reference

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

- LegacySecAttr.h

## 5.45 ArcSHCLegacy::LegacySecHandler Class Reference

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

- LegacySecHandler.h

## 5.46 Arc::Lister Class Reference

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

- Lister.h

## 5.47 ArcMCCGSI::MCC_GSI_Client Class Reference

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

- MCCGSI.h

## 5.48 ArcMCCGSI::MCC_GSI_Service Class Reference

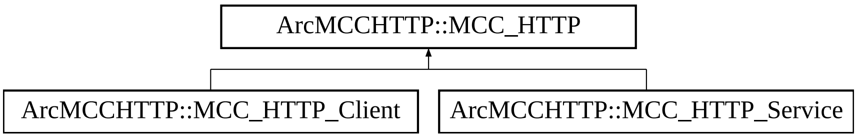
The documentation for this class was generated from the following file:

- MCCGSI.h

## 5.49 ArcMCCHTTP::MCC_HTTP Class Reference

A base class for HTTP client and service MCCs.

`#include <MCCHTTP.h>`Inheritance diagram for ArcMCCHTTP::MCC_HTTP::



### 5.49.1 Detailed Description

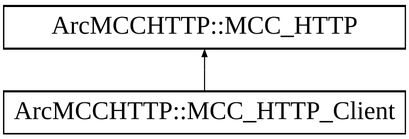
A base class for HTTP client and service MCCs. This is a base class for HTTP client and service MCCs. It provides some common functionality for them, i.e. so far only a logger.

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

- MCCHTTP.h

## 5.50 ArcMCCHTTP::MCC_HTTP_Client Class Reference

`#include <MCCHTTP.h>`Inheritance diagram for ArcMCCHTTP::MCC_HTTP_Client::



### 5.50.1 Detailed Description

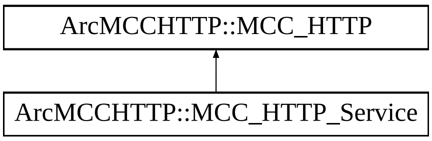
This class is a client part of HTTP MCC. It accepts PayloadRawInterface payload and uses it as body to generate HTTP request. Request is passed to next MCC as PayloadRawInterface type of payload. Returned PayloadStreamInterface payload is parsed into HTTP response and it's body is passed back to calling MCC as PayloadRawInerface. Attributes of request/input message of type HTTP:name are translated into HTTP header with corresponding 'name's. Special attributes HTTP:METHOD and HTTP:ENDPOINT specify method and URL in HTTP request. If not present meathod and URL are taken from configuration. In output/response message following attributes are present: HTTP:CODE - response code of HTTP HTTP:REASON - reason string of HTTP response HTTP:name - all 'name' attributes of HTTP header.

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

- MCCHTTP.h

## 5.51 ArcMCCHTTP::MCC_HTTP_Service Class Reference

`#include <MCCHTTP.h>`Inheritance diagram for ArcMCCHTTP::MCC_HTTP_Service::



### 5.51.1 Detailed Description

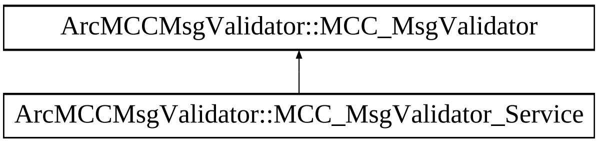
This class implements MCC to processes HTTP request. On input payload with PayloadStreamInterface is expected. HTTP message is read from stream ans it's body is converted into PayloadRaw and passed to next MCC. Returned payload of PayloadRawInterface type is treated as body part of returning PayloadHTTP. Generated HTTP response is sent though stream passed in input payload. During processing of request/input message following attributes are generated: HTTP:METHOD - HTTP method e.g. GET, PUT, POST, etc. HTTP:ENDPOINT - URL taken from HTTP request ENDPOINT - global attribute equal to HTTP:ENDPOINT HTTP:RANGESTART - start of requested byte range HTTP:RANGEEND - end of requested byte range (inclusive) HTTP:name - all 'name' attributes of HTTP header. Attributes of response message of HTTP:name type are translated into HTTP header with corresponding 'name's.

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

- MCCHTTP.h

## 5.52 ArcMCCMsgValidator::MCC_MsgValidator Class Reference

Inheritance diagram for ArcMCCMsgValidator::MCC_MsgValidator::

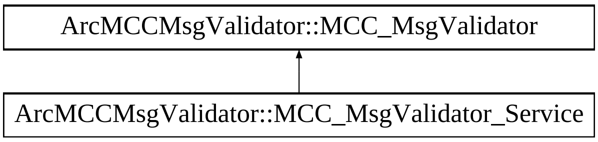

ArcMCCMsgValidator::MCC_MsgValidator
ArcMCCMsgValidator::MCC_MsgValidator_Service

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

- MCCMsgValidator.h

## 5.53 ArcMCCMsgValidator::MCC_MsgValidator_Service Class Reference

Inheritance diagram for ArcMCCMsgValidator::MCC_MsgValidator_Service::


ArcMCCMsgValidator::MCC_MsgValidator
ArcMCCMsgValidator::MCC_MsgValidator_Service

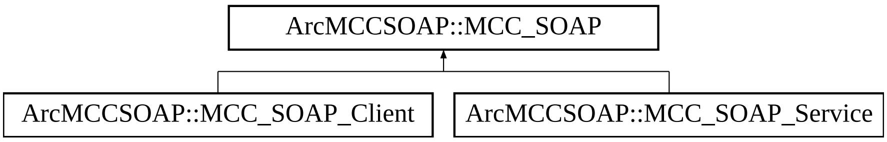
The documentation for this class was generated from the following file:

- MCCMsgValidator.h

## 5.54 ArcMCCSOAP::MCC_SOAP Class Reference

A base class for SOAP client and service MCCs.

`#include <MCCSOAP.h>`Inheritance diagram for ArcMCCSOAP::MCC_SOAP::



### 5.54.1 Detailed Description

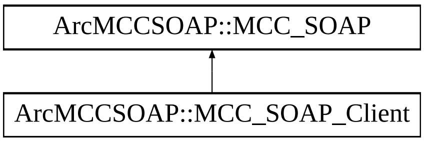
A base class for SOAP client and service MCCs. This is a base class for SOAP client and service MCCs. It provides some common functionality for them, i.e. so far only a logger.

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

- MCCSOAP.h

## 5.55 ArcMCCSOAP::MCC_SOAP_Client Class Reference

Inheritance diagram for ArcMCCSOAP::MCC_SOAP_Client::

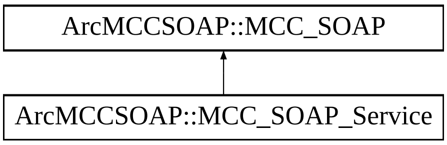


The documentation for this class was generated from the following file:

- MCCSOAP.h

## 5.56 ArcMCCSOAP::MCC_SOAP_Service Class Reference

`#include <MCCSOAP.h>`Inheritance diagram for ArcMCCSOAP::MCC_SOAP_Service::



### 5.56.1 Detailed Description

This MCC parses SOAP message from input payload. On input payload with PayloadRawInterface is expected. It's converted into PayloadSOAP and passed next MCC. Returned PayloadSOAP is converted into PayloadRaw and returned to calling MCC.

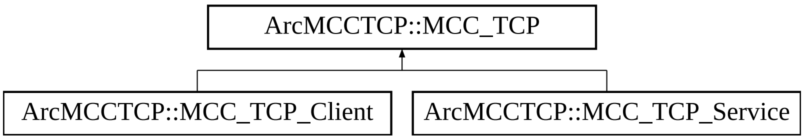
The documentation for this class was generated from the following file:

- MCCSOAP.h

## 5.57 ArcMCCTCP::MCC_TCP Class Reference

A base class for TCP client and service MCCs.

`#include <MCCTCP.h>`Inheritance diagram for ArcMCCTCP::MCC_TCP::



### 5.57.1 Detailed Description

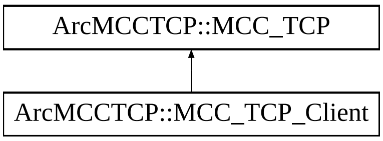
A base class for TCP client and service MCCs. This is a base class for TCP client and service MCCs. It provides some common functionality for them, i.e. so far only a logger.

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

- MCCTCP.h

## 5.58 ArcMCCTCP::MCC_TCP_Client Class Reference

`#include <MCCTCP.h>`Inheritance diagram for ArcMCCTCP::MCC_TCP_Client::



### 5.58.1 Detailed Description

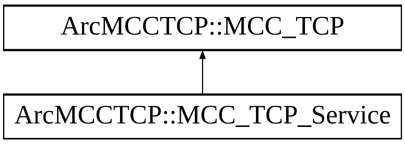
This class is MCC implementing TCP client. Upon creation it connects to specified TCP post at specified host. process() method accepts PayloadRawInterface type of payload. Content of payload is sent over TCP socket. It returns PayloadStreamInterface payload for previous MCC to read response.

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

- MCCTCP.h

## 5.59 ArcMCCTCP::MCC_TCP_Service Class Reference

`#include <MCCTCP.h>`Inheritance diagram for ArcMCCTCP::MCC_TCP_Service::



### Data Structures

- class **mcc_tcp_exec_t**
- class **mcc_tcp_handle_t**

### Public Member Functions

- MCC_TCP_Service (Config ∗cfg, PluginArgument ∗parg)

### 5.59.1 Detailed Description

This class is MCC implementing TCP server. Upon creation this object binds to specified TCP ports and listens for incoming TCP connections on dedicated thread. Each connection is accepted and dedicated thread is created. Then that thread is used to call process() method of next MCC in chain. That method is passed payload implementing PayloadStreamInterface. On response payload with PayloadRawInterface is expected. Alternatively called MCC may use provided PayloadStreamInterface to send it's response back directly. During processing of request this MCC generates following attributes: TCP:HOST - IP address of interface to which local TCP socket is bound TCP:PORT - port number to which local TCP socket is bound TCP:REMOTEHOST - IP address from which connection is accepted TCP:REMOTEPORT - TCP port from which connection is accepted TCP:ENDPOINT - URL-like representation of remote connection - ://HOST:PORT ENDPOINT - global attribute equal to TCP:ENDPOINT

### 5.59.2 Constructor & Destructor Documentation

#### 5.59.2.1 ArcMCCTCP::MCC_TCP_Service::MCC_TCP_Service (Config ∗ *cfg*, PluginArgument ∗ *parg*)

executing function for connection thread

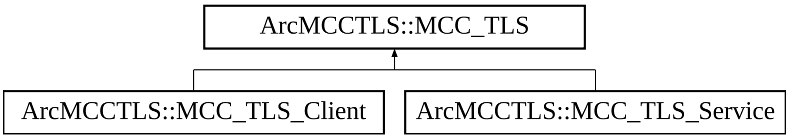
The documentation for this class was generated from the following file:

- MCCTCP.h

## 5.60 ArcMCCTLS::MCC_TLS Class Reference

A base class for TLS client and service MCCs.

`#include <MCCTLS.h>`Inheritance diagram for ArcMCCTLS::MCC_TLS::



### 5.60.1 Detailed Description

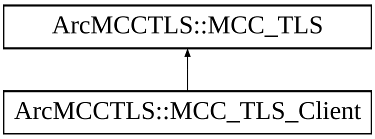
A base class for TLS client and service MCCs. This is a base class for TLS client and service MCCs. It provides some common functionality for them.

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

- MCCTLS.h

## 5.61 ArcMCCTLS::MCC_TLS_Client Class Reference

`#include <MCCTLS.h>`Inheritance diagram for ArcMCCTLS::MCC_TLS_Client::



### 5.61.1 Detailed Description

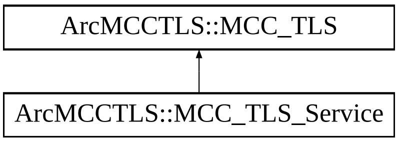
This class is MCC implementing TLS client.

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

- MCCTLS.h

## 5.62 ArcMCCTLS::MCC_TLS_Service Class Reference

`#include <MCCTLS.h>`Inheritance diagram for ArcMCCTLS::MCC_TLS_Service::



### 5.62.1 Detailed Description

This MCC implements TLS server side functionality. Upon creation this object creats SSL_CTX object and configures SSL_CTX object with some environment information about credential. Because we cannot know the "socket" when the creation of MCC_TLS_Service/MCC_TLS_Client object (not like MCC_TCP_Client, which can creat socket in the constructor method by using information in configuration file), we can only creat "ssl" object which is binded to specified "socket", when MCC_HTTP_Client calls the process() method of MCC_TLS_Client object, or MCC_TCP_Service calls the process() method of MCC_TLS_Service object. The "ssl" object is embeded in a payload called PayloadTLSSocket.

The process() method of MCC_TLS_Service is passed payload implementing PayloadStreamInterface and the method returns empty PayloadRaw payload in "outmsg". The ssl object is created and bound to Stream payload when constructing the PayloadTLSSocket in the process() method.

During processing of message this MCC generates attribute TLS:PEERDN which contains Distinguished Name of remoote peer.

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

- MCCTLS.h

## 5.63 ArcMCCGSI::PayloadGSIStream Class Reference

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

- PayloadGSIStream.h

## 5.64 ArcMCCHTTP::PayloadHTTP Class Reference

```
#include <PayloadHTTP.h>
```

### Public Member Functions

- PayloadHTTP (PayloadStreamInterface &stream, bool own=false)
- PayloadHTTP (const std::string &method, const std::string &url, PayloadStreamInterface &stream)
- PayloadHTTP (const std::string &method, const std::string &url)
- PayloadHTTP (int code, const std::string &reason, PayloadStreamInterface &stream, bool head_response=false)
- PayloadHTTP (int code, const std::string &reason, bool head_response=false)
- virtual const std::string & Attribute (const std::string &name)
- virtual const std::multimap< std::string, std::string > & Attributes (void)
- virtual void Attribute (const std::string &name, const std::string &value)
- virtual bool Flush (void)
- virtual void Body (PayloadRawInterface &body, bool ownership=true)

### Protected Member Functions

- bool readline (std::string &line)
- bool read (char *buf, int64_t &size)
- bool read_header (void)
- bool get_body (void)

### Protected Attributes

- PayloadStreamInterface * stream_
- bool stream_own_
- PayloadRawInterface * rbody_
- PayloadStreamInterface * sbody_
- bool body_own_
- std::string uri_
- int version_major_
- int version_minor_
- std::string method_
- int code_
- std::string reason_
- int64_t length_
- int64_t end_
- chunked_t chunked_
- int64_t chunk_size_
- std::multimap< std::string, std::string > attributes_

### 5.64.1 Detailed Description

This class implements parsing and generation of HTTP messages. It implements only subset of HTTP/1.1 and also provides an PayloadRawInterface for including as payload into Message passed through MCC chains.

## 5.64.2 Constructor & Destructor Documentation

### 5.64.2.1 ArcMCCHTTP::PayloadHTTP::PayloadHTTP (PayloadStreamInterface & *stream*, bool *own* = `false`)

Constructor - creates object by parsing HTTP request or response from stream. Supplied stream is associated with object for later use. If own is set to true then stream will be deleted in destructor. Because stream can be used by this object during whole lifetime it is important not to destroy stream till this object is deleted.

### 5.64.2.2 ArcMCCHTTP::PayloadHTTP::PayloadHTTP (const std::string & *method*, const std::string & *url*, PayloadStreamInterface & *stream*)

Constructor - creates HTTP request to be sent through stream. HTTP message is not sent yet.

### 5.64.2.3 ArcMCCHTTP::PayloadHTTP::PayloadHTTP (const std::string & *method*, const std::string & *url*)

Constructor - creates HTTP request to be rendered through Raw interface.

### 5.64.2.4 ArcMCCHTTP::PayloadHTTP::PayloadHTTP (int *code*, const std::string & *reason*, PayloadStreamInterface & *stream*, bool *head_response* = `false`)

Constructor - creates HTTP response to be sent through stream. HTTP message is not sent yet.

### 5.64.2.5 ArcMCCHTTP::PayloadHTTP::PayloadHTTP (int *code*, const std::string & *reason*, bool *head_response* = `false`)

Constructor - creates HTTP response to be rendered through Raw interface.

## 5.64.3 Member Function Documentation

### 5.64.3.1 virtual void ArcMCCHTTP::PayloadHTTP::Attribute (const std::string & *name*, const std::string & *value*) `[virtual]`

Adds HTTP header attribute 'name' = 'value'

### 5.64.3.2 virtual const std::string& ArcMCCHTTP::PayloadHTTP::Attribute (const std::string & *name*) `[virtual]`

Returns HTTP header attribute with specified name. Empty string if no such attribute.

### 5.64.3.3 virtual const std::multimap<std::string,std::string>& ArcMCCHTTP::PayloadHTTP::Attributes (void) `[virtual]`

Returns all HTTP header attributes.

#### 5.64.3.4 virtual void ArcMCCHTTP::PayloadHTTP::Body (PayloadRawInterface & *body*, bool *ownership* = `true`) `[virtual]`

Assign HTTP body. Assigned object is not copied. Instead it is remembered and made available through Raw interface. If 'ownership' is true then passed object is treated as being owned by this instance and destroyed in destructor.

#### 5.64.3.5 virtual bool ArcMCCHTTP::PayloadHTTP::Flush (void) `[virtual]`

Send created object through associated stream. If there is no stream associated then HTTP specific data is inserted into Raw buffers of this object. In last case this operation should not be repeated till content of buffer is completely rewritten.

#### 5.64.3.6 bool ArcMCCHTTP::PayloadHTTP::get_body (void) `[protected]`

Read Body of HTTP message and attach it to inherited PayloadRaw object

#### 5.64.3.7 bool ArcMCCHTTP::PayloadHTTP::read (char ∗ *buf*, int64_t & *size*) `[protected]`

Read up to 'size' bytes from stream_

#### 5.64.3.8 bool ArcMCCHTTP::PayloadHTTP::read_header (void) `[protected]`

Read HTTP header and fill internal variables

#### 5.64.3.9 bool ArcMCCHTTP::PayloadHTTP::readline (std::string & *line*) `[protected]`

Read from stream till

### 5.64.4 Field Documentation

#### 5.64.4.1 std::multimap<std::string,std::string> ArcMCCHTTP::PayloadHTTP::attributes_ `[protected]`

true if conection should not be closed after response

#### 5.64.4.2 bool ArcMCCHTTP::PayloadHTTP::body_own_ `[protected]`

associated HTTP Body stream if any (to avoid copying to own buffer)

#### 5.64.4.3 int64_t ArcMCCHTTP::PayloadHTTP::chunk_size_ `[protected]`

chunked encoding parsing state

#### 5.64.4.4 chunked_t ArcMCCHTTP::PayloadHTTP::chunked_ `[protected]`

Logical end of content computed from Content-Range

#### 5.64.4.5 int ArcMCCHTTP::PayloadHTTP::code_ [protected]

HTTP method being used or requested

#### 5.64.4.6 int64_t ArcMCCHTTP::PayloadHTTP::end_ [protected]

Content-length of HTTP message

#### 5.64.4.7 int64_t ArcMCCHTTP::PayloadHTTP::length_ [protected]

HTTP reason being sent or supplied

#### 5.64.4.8 std::string ArcMCCHTTP::PayloadHTTP::method_ [protected]

minor number of HTTP version - must be 0 or 1

#### 5.64.4.9 PayloadRawInterface* ArcMCCHTTP::PayloadHTTP::rbody_ [protected]

if true stream_ is owned by this

#### 5.64.4.10 std::string ArcMCCHTTP::PayloadHTTP::reason_ [protected]

HTTP code being sent or supplied

#### 5.64.4.11 PayloadStreamInterface* ArcMCCHTTP::PayloadHTTP::sbody_ [protected]

associated HTTP Body buffer if any (to avoid copying to own buffer)

#### 5.64.4.12 PayloadStreamInterface* ArcMCCHTTP::PayloadHTTP::stream_ [protected]

true if whole content of HTTP body was fetched and stored in buffers. Otherwise only header was fetched and part of body in tbuf_ and rest is to be read through stream_.

#### 5.64.4.13 bool ArcMCCHTTP::PayloadHTTP::stream_own_ [protected]

stream used to comminicate to outside

#### 5.64.4.14 std::string ArcMCCHTTP::PayloadHTTP::uri_ [protected]

if true body_ is owned by this

#### 5.64.4.15 int ArcMCCHTTP::PayloadHTTP::version_major_ [protected]

URI being contacted

#### 5.64.4.16 int ArcMCCHTTP::PayloadHTTP::version_minor_ [protected]

major number of HTTP version - must be 1

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

- PayloadHTTP.h

## 5.65 ArcMCCTCP::PayloadTCPSocket Class Reference

```
#include <PayloadTCPSocket.h>
```

### Public Member Functions

- PayloadTCPSocket (const char ∗hostname, int port, int timeout, Logger &logger)
- PayloadTCPSocket (const std::string &endpoint, int timeout, Logger &logger)
- PayloadTCPSocket (int s, int timeout, Logger &logger)
- PayloadTCPSocket (PayloadTCPSocket &s)
- PayloadTCPSocket (PayloadTCPSocket &s, Logger &logger)

### 5.65.1 Detailed Description

This class extends PayloadStream with TCP socket specific features

### 5.65.2 Constructor & Destructor Documentation

#### 5.65.2.1 ArcMCCTCP::PayloadTCPSocket::PayloadTCPSocket (const char ∗ *hostname*, int *port*, int *timeout*, Logger & *logger*)

Constructor - connects to TCP server at specified hostname:port

#### 5.65.2.2 ArcMCCTCP::PayloadTCPSocket::PayloadTCPSocket (const std::string & *endpoint*, int *timeout*, Logger & *logger*)

Constructor - connects to TCP server at specified endpoint - hostname:port

#### 5.65.2.3 ArcMCCTCP::PayloadTCPSocket::PayloadTCPSocket (int *s*, int *timeout*, Logger & *logger*) `[inline]`

Constructor - creates object of already connected socket. Socket is NOT closed in destructor.

#### 5.65.2.4 ArcMCCTCP::PayloadTCPSocket::PayloadTCPSocket (PayloadTCPSocket & *s*) `[inline]`

Copy constructor - inherits socket of copied object. Socket is NOT closed in destructor.

#### 5.65.2.5 ArcMCCTCP::PayloadTCPSocket::PayloadTCPSocket (PayloadTCPSocket & *s*, Logger & *logger*) `[inline]`

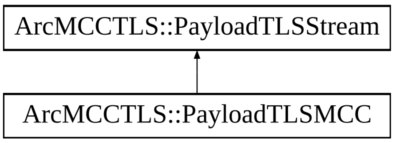
Copy constructor - inherits handle of copied object. Handle is NOT closed in destructor.

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

- PayloadTCPSocket.h

## 5.66 ArcMCCTLS::PayloadTLSMCC Class Reference

Inheritance diagram for ArcMCCTLS::PayloadTLSMCC::



### Public Member Functions

- PayloadTLSMCC (MCCInterface *mcc, const ConfigTLSMCC &cfg, Logger &logger)
- PayloadTLSMCC (PayloadStreamInterface *stream, const ConfigTLSMCC &cfg, Logger &logger)
- PayloadTLSMCC (PayloadTLSMCC &stream)

### 5.66.1 Constructor & Destructor Documentation

#### 5.66.1.1 ArcMCCTLS::PayloadTLSMCC::PayloadTLSMCC (MCCInterface * *mcc*, const ConfigTLSMCC & *cfg*, Logger & *logger*)

Constructor - creates ssl object which is bound to next MCC. This instance must be used on client side. It obtains Stream interface from next MCC dynamically.

#### 5.66.1.2 ArcMCCTLS::PayloadTLSMCC::PayloadTLSMCC (PayloadStreamInterface * *stream*, const ConfigTLSMCC & *cfg*, Logger & *logger*)

Constructor - creates ssl object which is bound to stream. This constructor to be used on server side. Provided stream is NOT destroyed in destructor.

#### 5.66.1.3 ArcMCCTLS::PayloadTLSMCC::PayloadTLSMCC (PayloadTLSMCC & *stream*)

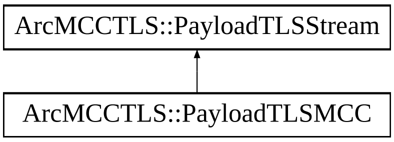
Copy constructor with new logger. Created object shares same SSL objects but does not destroy them in destructor. Main instance must be destroyed after all copied ones.

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

- PayloadTLSMCC.h

## 5.67 ArcMCCTLS::PayloadTLSStream Class Reference

`#include <PayloadTLSStream.h>`Inheritance diagram for ArcMCCTLS::PayloadTLSStream::



### Public Member Functions

- PayloadTLSStream (Logger &logger, SSL ∗ssl=NULL)
- virtual ∼PayloadTLSStream (void)
- X509 ∗ GetPeerCert (void)
- STACK_OF (X509)∗GetPeerChain(void)
- X509 ∗ GetCert (void)

### Protected Attributes

- SSL ∗ ssl_

### 5.67.1 Detailed Description

Implemetation of PayloadStreamInterface for SSL handle.

### 5.67.2 Constructor & Destructor Documentation

#### 5.67.2.1 ArcMCCTLS::PayloadTLSStream::PayloadTLSStream (Logger & *logger*, SSL ∗ *ssl* = NULL)

Constructor. Attaches to already open handle. Handle is not managed by this class and must be closed by external code.

#### 5.67.2.2 virtual ArcMCCTLS::PayloadTLSStream::∼PayloadTLSStream (void) [virtual]

Destructor.

### 5.67.3 Member Function Documentation

#### 5.67.3.1 X509∗ ArcMCCTLS::PayloadTLSStream::GetCert (void)

Get local certificate from associated ssl. Obtained X509 object is owned by this instance and becomes invalid after destruction.

#### 5.67.3.2 X509∗ ArcMCCTLS::PayloadTLSStream::GetPeerCert (void)

Get peer certificate from the established ssl. Obtained X509 object is owned by this instance and becomes invalid after destruction. Still obtained has to be freed at end of usage.

#### 5.67.3.3 ArcMCCTLS::PayloadTLSStream::STACK_OF (X509)

Get chain of peer certificates from the established ssl. Obtained X509 object is owned by this instance and becomes invalid after destruction.

### 5.67.4 Field Documentation

#### 5.67.4.1 SSL∗ ArcMCCTLS::PayloadTLSStream::ssl_ `[protected]`

Timeout for read/write operations

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

- PayloadTLSStream.h

## 5.68 ArcSec::PDPServiceInvoker Class Reference

PDPServiceInvoker - client which will invoke pdpservice.

```
#include <PDPServiceInvoker.h>
```

### 5.68.1 Detailed Description

PDPServiceInvoker - client which will invoke pdpservice.

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

- PDPServiceInvoker.h

## 5.69 ArcSec::SAML2SSO_AssertionConsumerSH Class Reference

Implement the funcionality of the Service Provider in SAML2 SSO profile.

```
#include <SAML2SSO_AssertionConsumerSH.h>
```

### 5.69.1 Detailed Description

Implement the funcionality of the Service Provider in SAML2 SSO profile.

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

- SAML2SSO_AssertionConsumerSH.h

## 5.70 ArcSec::SAMLTokenSH Class Reference

Adds WS-Security SAML Token into SOAP Header.

```
#include <SAMLTokenSH.h>
```

### 5.70.1 Detailed Description

Adds WS-Security SAML Token into SOAP Header.

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

- SAMLTokenSH.h

## 5.71 ArcSec::SimpleListPDP Class Reference

Tests X509 subject against list of subjects in file.

```
#include <SimpleListPDP.h>
```

### 5.71.1 Detailed Description

Tests X509 subject against list of subjects in file. This class implements PDP interface. It's isPermitted() method compares X590 subject of requestor obtained from TLS layer (TLS:PEERDN) to list of subjects (ne per line) in external file. Locations of file is defined by 'location' attribute of PDP caonfiguration. Returns true if subject is present in list, otherwise false.

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

- SimpleListPDP.h

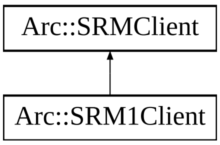
## 5.72 ArcSHCLegacy::SimpleMap Class Reference

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

- simplemap.h

## 5.73 Arc::SRM1Client Class Reference

Inheritance diagram for Arc::SRM1Client::



### Public Member Functions

- SRMReturnCode ping (std::string &, bool=true)
- SRMReturnCode getSpaceTokens (std::list< std::string > &, const std::string &="")
- SRMReturnCode getRequestTokens (std::list< std::string > &, const std::string &="")
- SRMReturnCode requestBringOnline (SRMClientRequest &)
- SRMReturnCode requestBringOnlineStatus (SRMClientRequest &)
- SRMReturnCode mkDir (SRMClientRequest &)
- SRMReturnCode checkPermissions (SRMClientRequest &)
- SRMReturnCode getTURLs (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode getTURLsStatus (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode putTURLs (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode putTURLsStatus (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode releaseGet (SRMClientRequest &req)
- SRMReturnCode releasePut (SRMClientRequest &req)
- SRMReturnCode release (SRMClientRequest &req)
- SRMReturnCode abort (SRMClientRequest &req)
- SRMReturnCode info (SRMClientRequest &req, std::map< std::string, std::list< struct SRMFileMetaData > > &metadata)
- SRMReturnCode info (SRMClientRequest &req, std::list< struct SRMFileMetaData > &metadata)
- SRMReturnCode remove (SRMClientRequest &req)
- SRMReturnCode copy (SRMClientRequest &req, const std::string &source)

### 5.73.1 Member Function Documentation

#### 5.73.1.1 SRMReturnCode Arc::SRM1Client::abort (SRMClientRequest & *req*) `[virtual]`

Called in the case of failure during transfer or releasePut. Releases all TURLs involved in the transfer.

**Parameters:**

*req* The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.2 SRMReturnCode Arc::SRM1Client::checkPermissions (SRMClientRequest & *req*) `[inline, virtual]`

Check permissions for the SURL in the request using the current credentials. req The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.3 SRMReturnCode Arc::SRM1Client::copy (SRMClientRequest & *req*, const std::string & *source*) `[virtual]`

Copy a file between two SRM storages.

**Parameters:**

***req*** The request object

***source*** The source SURL

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.4 SRMReturnCode Arc::SRM1Client::getRequestTokens (std::list< std::string > & *tokens*, const std::string & *description* = "") `[inline, virtual]`

Returns a list of request tokens for the user calling the method which are still active requests, or the tokens corresponding to the token description, if given.

**Parameters:**

***tokens*** The list filled by the service

***description*** The user request description, which can be specified when the request is created

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.5 SRMReturnCode Arc::SRM1Client::getSpaceTokens (std::list< std::string > & *tokens*, const std::string & *description* = "") `[inline, virtual]`

Find the space tokens available to write to which correspond to the space token description, if given. The list of tokens is a list of numbers referring to the SRM internal definition of the spaces, not user-readable strings.

**Parameters:**

***tokens*** The list filled by the service

***description*** The space token description

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.6 SRMReturnCode Arc::SRM1Client::getTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*) `[virtual]`

If the user wishes to copy a file from somewhere, getTURLs() is called to retrieve the transport URL(s) to copy the file from. It may be used synchronously or asynchronously, depending on the synchronous property of the request object. In the former case it will block until the TURLs are ready, in the latter case it will return after making the request and getTURLsStatus() must be used to poll the request status if it was not completed.

**Parameters:**

***req*** The request object

***urls*** A list of TURLs filled by the method

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.7 SRMReturnCode Arc::SRM1Client::getTURLsStatus (SRMClientRequest & *req*, std::list< std::string > & *urls*) `[inline, virtual]`

In the case where getTURLs was called asynchronously and the request was not completed, this method should be called to poll the status of the request. getTURLs must be called before this method and the request object must have ongoing request status.

**Parameters:**

***req*** The request object. Status must be ongoing.

***urls*** A list of TURLs filled by the method if the request completed successfully

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.8 SRMReturnCode Arc::SRM1Client::info (SRMClientRequest & *req*, std::list< struct SRMFileMetaData > & *metadata*) `[virtual]`

Returns information on a file stored in an SRM, such as file size, checksum and estimated access latency. If a directory is listed with recursion >= 1 then the list in metadata will contain the content of the directory.

**Parameters:**

***req*** The request object

***metadata*** A list of structs filled with file information.

**Returns:**

SRMReturnCode specifying outcome of operation

**See also:**

SRMFileMetaData

Implements Arc::SRMClient.

#### 5.73.1.9 SRMReturnCode Arc::SRM1Client::info (SRMClientRequest & *req*, std::map< std::string, std::list< struct SRMFileMetaData > > & *metadata*) `[virtual]`

Returns information on a file or files (v2.2 and higher) stored in SRM, such as file size, checksum and estimated access latency. If a directory or directories is listed with recursion >= 1 then the list mapped to each SURL in metadata will contain the content of the directory or directories.

**Parameters:**

***req*** The request object

***metadata*** A map mapping each SURL in the request to a list of structs filled with file information. If a SURL is missing from the map it means there was some problem accessing it.

**Returns:**

SRMReturnCode specifying outcome of operation

**See also:**

SRMFileMetaData

Implements Arc::SRMClient.

#### 5.73.1.10 SRMReturnCode Arc::SRM1Client::mkDir (SRMClientRequest & *req*) `[inline, virtual]`

Make required directories for the SURL in the request

**Parameters:**

***req*** The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.11 SRMReturnCode Arc::SRM1Client::ping (std::string & *version*, bool *report_error* = true) [inline, virtual]

Find out the version supported by the server this client is connected to. Since this method is used to determine which client version to instantiate, we may not want to report an error to the user, so setting report_error to false suppresses the error message.

**Parameters:**

*version* The version returned by the server

*report_error* Whether an error should be reported

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.12 SRMReturnCode Arc::SRM1Client::putTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*) [virtual]

If the user wishes to copy a file to somewhere, putTURLs() is called to retrieve the transport URL(s) to copy the file to. It may be used synchronously or asynchronously, depending on the synchronous property of the request object. In the former case it will block until the TURLs are ready, in the latter case it will return after making the request and putTURLsStatus() must be used to poll the request status if it was not completed.

**Parameters:**

*req* The request object

*urls* A list of TURLs filled by the method

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.13 SRMReturnCode Arc::SRM1Client::putTURLsStatus (SRMClientRequest & *req*, std::list< std::string > & *urls*) [inline, virtual]

In the case where putTURLs was called asynchronously and the request was not completed, this method should be called to poll the status of the request. putTURLs must be called before this method and the request object must have ongoing request status.

**Parameters:**

*req* The request object. Status must be ongoing.

*urls* A list of TURLs filled by the method if the request completed successfully

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.14 SRMReturnCode Arc::SRM1Client::release (SRMClientRequest & *req*) `[virtual]`

Used in SRM v1 only. Called to release files after successful transfer.

**Parameters:**

***req*** The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.15 SRMReturnCode Arc::SRM1Client::releaseGet (SRMClientRequest & *req*) `[virtual]`

Should be called after a successful copy from SRM storage.

**Parameters:**

***req*** The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.16 SRMReturnCode Arc::SRM1Client::releasePut (SRMClientRequest & *req*) `[virtual]`

Should be called after a successful copy to SRM storage.

**Parameters:**

***req*** The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.17 SRMReturnCode Arc::SRM1Client::remove (SRMClientRequest & *req*) `[virtual]`

Delete a file physically from storage and the SRM namespace.

**Parameters:**

***req*** The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.18 SRMReturnCode Arc::SRM1Client::requestBringOnline (SRMClientRequest & *req*) `[inline, virtual]`

Submit a request to bring online files. If the synchronous property of the request object is false, this operation is asynchronous and the status of the request can be checked by calling requestBringOnlineStatus() with the request token in req which is assigned by this method. If the request is synchronous, this operation blocks until the file(s) are online or the timeout specified in the SRMClient constructor has passed.

**Parameters:**

***req*** The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implements Arc::SRMClient.

#### 5.73.1.19 SRMReturnCode Arc::SRM1Client::requestBringOnlineStatus (SRMClientRequest & *req*) `[inline, virtual]`

Query the status of a request to bring files online. The SURLs map of the request object is updated if the status of any files in the request has changed. requestBringOnline() but be called before this method.

**Parameters:**

***req*** The request object to query the status of

**Returns:**

SRMReturnCode specifying outcome of operation

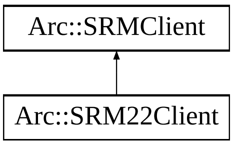
Implements Arc::SRMClient.

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

- SRM1Client.h

## 5.74 Arc::SRM22Client Class Reference

Inheritance diagram for Arc::SRM22Client::



### Public Member Functions

- SRM22Client (const UserConfig &usercfg, const SRMURL &url)
- ~SRM22Client ()
- SRMReturnCode ping (std::string &version, bool report_error=true)
- SRMReturnCode getSpaceTokens (std::list< std::string > &tokens, const std::string &description="")
- SRMReturnCode getRequestTokens (std::list< std::string > &tokens, const std::string &description="")
- SRMReturnCode getTURLs (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode getTURLsStatus (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode putTURLs (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode putTURLsStatus (SRMClientRequest &req, std::list< std::string > &urls)
- SRMReturnCode requestBringOnline (SRMClientRequest &req)
- SRMReturnCode requestBringOnlineStatus (SRMClientRequest &req)
- SRMReturnCode info (SRMClientRequest &req, std::map< std::string, std::list< struct SRMFileMetaData > > &metadata)
- SRMReturnCode info (SRMClientRequest &req, std::list< struct SRMFileMetaData > &metadata)
- SRMReturnCode releaseGet (SRMClientRequest &req)
- SRMReturnCode releasePut (SRMClientRequest &req)
- SRMReturnCode release (SRMClientRequest &)
- SRMReturnCode abort (SRMClientRequest &req)
- SRMReturnCode remove (SRMClientRequest &req)
- SRMReturnCode copy (SRMClientRequest &req, const std::string &source)
- SRMReturnCode mkDir (SRMClientRequest &req)
- SRMReturnCode checkPermissions (SRMClientRequest &req)

### 5.74.1 Constructor & Destructor Documentation

#### 5.74.1.1 Arc::SRM22Client::SRM22Client (const UserConfig & *usercfg*, const SRMURL & *url*)

Constructor

#### 5.74.1.2 Arc::SRM22Client::~SRM22Client ()

Destructor

## 5.74.2 Member Function Documentation

### 5.74.2.1 SRMReturnCode Arc::SRM22Client::abort (SRMClientRequest & *req*) `[virtual]`

Abort request. Called after any failure in the data transfer or putDone calls

Implements Arc::SRMClient.

### 5.74.2.2 SRMReturnCode Arc::SRM22Client::checkPermissions (SRMClientRequest & *req*) `[virtual]`

Call srmCheckPermission

Implements Arc::SRMClient.

### 5.74.2.3 SRMReturnCode Arc::SRM22Client::copy (SRMClientRequest & *req*, const std::string & *source*) `[virtual]`

Implemented in pull mode, ie the endpoint defined in the request object performs the copy.

Implements Arc::SRMClient.

### 5.74.2.4 SRMReturnCode Arc::SRM22Client::getRequestTokens (std::list< std::string > & *tokens*, const std::string & *description* = `""`) `[virtual]`

Use srmGetRequestTokens to return a list of spaces available

Implements Arc::SRMClient.

### 5.74.2.5 SRMReturnCode Arc::SRM22Client::getSpaceTokens (std::list< std::string > & *tokens*, const std::string & *description* = `""`) `[virtual]`

Use srmGetSpaceTokens to return a list of spaces available

Implements Arc::SRMClient.

### 5.74.2.6 SRMReturnCode Arc::SRM22Client::getTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*) `[virtual]`

Get a list of TURLs for the given SURL. Uses srmPrepareToGet and waits until file is ready (online and pinned) if the request is synchronous. If not it returns after making the request. Although a list is returned, SRMv2.2 only returns one TURL per SURL.

Implements Arc::SRMClient.

### 5.74.2.7 SRMReturnCode Arc::SRM22Client::getTURLsStatus (SRMClientRequest & *req*, std::list< std::string > & *urls*) `[virtual]`

Uses srmStatusOfGetRequest to query the status of the given request.

Implements Arc::SRMClient.

#### 5.74.2.8 SRMReturnCode Arc::SRM22Client::info (SRMClientRequest & *req*, std::list< struct SRMFileMetaData > & *metadata*) [virtual]

Use srmLs to get info on the given SURL. Info on each file or content of directory is put in a list of metadata structs

Implements Arc::SRMClient.

#### 5.74.2.9 SRMReturnCode Arc::SRM22Client::info (SRMClientRequest & *req*, std::map< std::string, std::list< struct SRMFileMetaData > > & *metadata*) [virtual]

Use srmLs to get info on the given SURLs. Info on each file or content of directory is put in a list of metadata structs.

Implements Arc::SRMClient.

#### 5.74.2.10 SRMReturnCode Arc::SRM22Client::mkDir (SRMClientRequest & *req*) [virtual]

Call srmMkDir

Implements Arc::SRMClient.

#### 5.74.2.11 SRMReturnCode Arc::SRM22Client::ping (std::string & *version*, bool *report_error* = true) [virtual]

Get the server version from srmPing

Implements Arc::SRMClient.

#### 5.74.2.12 SRMReturnCode Arc::SRM22Client::putTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*) [virtual]

Retrieve TURLs which a file can be written to. Uses srmPrepareToPut and waits until a suitable TURL has been assigned if the request is synchronous. If not it returns after making the request. Although a list is returned, SRMv2.2 only returns one TURL per SURL.

Implements Arc::SRMClient.

#### 5.74.2.13 SRMReturnCode Arc::SRM22Client::putTURLsStatus (SRMClientRequest & *req*, std::list< std::string > & *urls*) [virtual]

Uses srmStatusOfPutRequest to query the status of the given request.

Implements Arc::SRMClient.

#### 5.74.2.14 SRMReturnCode Arc::SRM22Client::release (SRMClientRequest &) [inline, virtual]

Not used in this version of SRM

Implements Arc::SRMClient.

#### 5.74.2.15 SRMReturnCode Arc::SRM22Client::releaseGet (SRMClientRequest & *req*) [virtual]

Release files that have been pinned by srmPrepareToGet using srmReleaseFiles. Called after successful file transfer or failed prepareToGet.

Implements Arc::SRMClient.

#### 5.74.2.16 SRMReturnCode Arc::SRM22Client::releasePut (SRMClientRequest & *req*) [virtual]

Mark a put request as finished. Called after successful file transfer or failed prepareToPut.

Implements Arc::SRMClient.

#### 5.74.2.17 SRMReturnCode Arc::SRM22Client::remove (SRMClientRequest & *req*) [virtual]

Delete by srmRm or srmRmDir

Implements Arc::SRMClient.

#### 5.74.2.18 SRMReturnCode Arc::SRM22Client::requestBringOnline (SRMClientRequest & *req*) [virtual]

Call srmBringOnline with the SURLs specified in req.

Implements Arc::SRMClient.

#### 5.74.2.19 SRMReturnCode Arc::SRM22Client::requestBringOnlineStatus (SRMClientRequest & *req*) [virtual]

Call srmStatusOfBringOnlineRequest and update req with any changes.

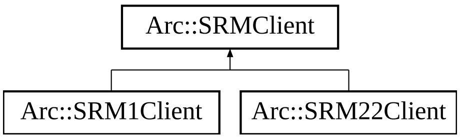
Implements Arc::SRMClient.

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

- SRM22Client.h

## 5.75 Arc::SRMClient Class Reference

`#include <SRMClient.h>`Inheritance diagram for Arc::SRMClient::



### Public Member Functions

- virtual ~SRMClient ()
- std::string getVersion () const
- virtual SRMReturnCode ping (std::string &version, bool report_error=true)=0
- virtual SRMReturnCode getSpaceTokens (std::list< std::string > &tokens, const std::string &description="")=0
- virtual SRMReturnCode getRequestTokens (std::list< std::string > &tokens, const std::string &description="")=0
- virtual SRMReturnCode getTURLs (SRMClientRequest &req, std::list< std::string > &urls)=0
- virtual SRMReturnCode getTURLsStatus (SRMClientRequest &req, std::list< std::string > &urls)=0
- virtual SRMReturnCode requestBringOnline (SRMClientRequest &req)=0
- virtual SRMReturnCode requestBringOnlineStatus (SRMClientRequest &req)=0
- virtual SRMReturnCode putTURLs (SRMClientRequest &req, std::list< std::string > &urls)=0
- virtual SRMReturnCode putTURLsStatus (SRMClientRequest &req, std::list< std::string > &urls)=0
- virtual SRMReturnCode releaseGet (SRMClientRequest &req)=0
- virtual SRMReturnCode releasePut (SRMClientRequest &req)=0
- virtual SRMReturnCode release (SRMClientRequest &req)=0
- virtual SRMReturnCode abort (SRMClientRequest &req)=0
- virtual SRMReturnCode info (SRMClientRequest &req, std::map< std::string, std::list< struct SRMFileMetaData > > &metadata)=0
- virtual SRMReturnCode info (SRMClientRequest &req, std::list< struct SRMFileMetaData > &metadata)=0
- virtual SRMReturnCode remove (SRMClientRequest &req)=0
- virtual SRMReturnCode copy (SRMClientRequest &req, const std::string &source)=0
- virtual SRMReturnCode mkDir (SRMClientRequest &req)=0
- virtual SRMReturnCode checkPermissions (SRMClientRequest &req)=0

### Static Public Member Functions

- static SRMClient * getInstance (const UserConfig &usercfg, const std::string &url, bool &timedout)

### Protected Member Functions

- SRMClient (const UserConfig &usercfg, const SRMURL &url)
- SRMReturnCode process (const std::string &action, PayloadSOAP *request, PayloadSOAP **response)

## Protected Attributes

- std::string service_endpoint
- MCCConfig cfg
- ClientSOAP ∗ client
- NS ns
- SRMImplementation implementation
- time_t user_timeout
- std::string version

## Static Protected Attributes

- static Logger logger

## 5.75.1 Detailed Description

A client interface to the SRM protocol. Instances of SRM clients are created by calling the getInstance() factory method. One client instance can be used to make many requests to the same server (with the same protocol version), but not multiple servers.

## 5.75.2 Constructor & Destructor Documentation

### 5.75.2.1 virtual Arc::SRMClient::∼SRMClient () `[virtual]`

Destructor

## 5.75.3 Member Function Documentation

### 5.75.3.1 virtual SRMReturnCode Arc::SRMClient::abort (SRMClientRequest & *req*) `[pure virtual]`

Called in the case of failure during transfer or releasePut. Releases all TURLs involved in the transfer.

**Parameters:**

***req*** The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

### 5.75.3.2 virtual SRMReturnCode Arc::SRMClient::checkPermissions (SRMClientRequest & *req*) `[pure virtual]`

Check permissions for the SURL in the request using the current credentials. req The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.3 virtual SRMReturnCode Arc::SRMClient::copy (SRMClientRequest & *req*, const std::string & *source*) `[pure virtual]`

Copy a file between two SRM storages.

**Parameters:**

***req*** The request object

***source*** The source SURL

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.4 static SRMClient∗ Arc::SRMClient::getInstance (const UserConfig & *usercfg*, const std::string & *url*, bool & *timedout*) `[static]`

Create an SRMClient instance. The instance will be a SRM v2.2 client unless another version is explicitly given in the url.

**Parameters:**

***usercfg*** The user configuration.

***url*** A SURL. A client connects to the service host derived from this SURL. All operations with a client instance must use SURLs with the same host as this one.

***timedout*** Whether the connection timed out

**Returns:**

A pointer to an instance of SRMClient is returned, or NULL if it was not possible to create one.

#### 5.75.3.5 virtual SRMReturnCode Arc::SRMClient::getRequestTokens (std::list< std::string > & *tokens*, const std::string & *description* = `""`) `[pure virtual]`

Returns a list of request tokens for the user calling the method which are still active requests, or the tokens corresponding to the token description, if given.

**Parameters:**

***tokens*** The list filled by the service

***description*** The user request description, which can be specified when the request is created

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.6 virtual SRMReturnCode Arc::SRMClient::getSpaceTokens (std::list< std::string > & *tokens*, const std::string & *description* = "") `[pure virtual]`

Find the space tokens available to write to which correspond to the space token description, if given. The list of tokens is a list of numbers referring to the SRM internal definition of the spaces, not user-readable strings.

**Parameters:**

***tokens*** The list filled by the service

***description*** The space token description

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.7 virtual SRMReturnCode Arc::SRMClient::getTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*) `[pure virtual]`

If the user wishes to copy a file from somewhere, getTURLs() is called to retrieve the transport URL(s) to copy the file from. It may be used synchronously or asynchronously, depending on the synchronous property of the request object. In the former case it will block until the TURLs are ready, in the latter case it will return after making the request and getTURLsStatus() must be used to poll the request status if it was not completed.

**Parameters:**

***req*** The request object

***urls*** A list of TURLs filled by the method

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.8 virtual SRMReturnCode Arc::SRMClient::getTURLsStatus (SRMClientRequest & *req*, std::list< std::string > & *urls*) `[pure virtual]`

In the case where getTURLs was called asynchronously and the request was not completed, this method should be called to poll the status of the request. getTURLs must be called before this method and the request object must have ongoing request status.

**Parameters:**

***req*** The request object. Status must be ongoing.

***urls*** A list of TURLs filled by the method if the request completed successfully

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

### 5.75.3.9 std::string Arc::SRMClient::getVersion () const `[inline]`

Returns the version of the SRM protocol used by this instance

References version.

### 5.75.3.10 virtual SRMReturnCode Arc::SRMClient::info (SRMClientRequest & *req*, std::list< struct SRMFileMetaData > & *metadata*) `[pure virtual]`

Returns information on a file stored in an SRM, such as file size, checksum and estimated access latency. If a directory is listed with recursion >= 1 then the list in metadata will contain the content of the directory.

**Parameters:**

***req*** The request object

***metadata*** A list of structs filled with file information.

**Returns:**

SRMReturnCode specifying outcome of operation

**See also:**

SRMFileMetaData

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

### 5.75.3.11 virtual SRMReturnCode Arc::SRMClient::info (SRMClientRequest & *req*, std::map< std::string, std::list< struct SRMFileMetaData > > & *metadata*) `[pure virtual]`

Returns information on a file or files (v2.2 and higher) stored in SRM, such as file size, checksum and estimated access latency. If a directory or directories is listed with recursion >= 1 then the list mapped to each SURL in metadata will contain the content of the directory or directories.

**Parameters:**

***req*** The request object

***metadata*** A map mapping each SURL in the request to a list of structs filled with file information. If a SURL is missing from the map it means there was some problem accessing it.

**Returns:**

SRMReturnCode specifying outcome of operation

**See also:**

SRMFileMetaData

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

### 5.75.3.12 virtual SRMReturnCode Arc::SRMClient::mkDir (SRMClientRequest & *req*) `[pure virtual]`

Make required directories for the SURL in the request

**Parameters:**

*req* The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.13 virtual SRMReturnCode Arc::SRMClient::ping (std::string & *version*, bool *report_error* = true) [pure virtual]

Find out the version supported by the server this client is connected to. Since this method is used to determine which client version to instantiate, we may not want to report an error to the user, so setting report_error to false suppresses the error message.

**Parameters:**

*version* The version returned by the server

*report_error* Whether an error should be reported

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.14 virtual SRMReturnCode Arc::SRMClient::putTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*) [pure virtual]

If the user wishes to copy a file to somewhere, putTURLs() is called to retrieve the transport URL(s) to copy the file to. It may be used synchronously or asynchronously, depending on the synchronous property of the request object. In the former case it will block until the TURLs are ready, in the latter case it will return after making the request and putTURLsStatus() must be used to poll the request status if it was not completed.

**Parameters:**

*req* The request object

*urls* A list of TURLs filled by the method

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.15 virtual SRMReturnCode Arc::SRMClient::putTURLsStatus (SRMClientRequest & *req*, std::list< std::string > & *urls*) [pure virtual]

In the case where putTURLs was called asynchronously and the request was not completed, this method should be called to poll the status of the request. putTURLs must be called before this method and the request object must have ongoing request status.

**Parameters:**

*req* The request object. Status must be ongoing.

*urls* A list of TURLs filled by the method if the request completed successfully

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.16 virtual SRMReturnCode Arc::SRMClient::release (SRMClientRequest & *req*) `[pure virtual]`

Used in SRM v1 only. Called to release files after successful transfer.

**Parameters:**

*req* The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.17 virtual SRMReturnCode Arc::SRMClient::releaseGet (SRMClientRequest & *req*) `[pure virtual]`

Should be called after a successful copy from SRM storage.

**Parameters:**

*req* The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.18 virtual SRMReturnCode Arc::SRMClient::releasePut (SRMClientRequest & *req*) `[pure virtual]`

Should be called after a successful copy to SRM storage.

**Parameters:**

*req* The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.19 virtual SRMReturnCode Arc::SRMClient::remove (SRMClientRequest & *req*) `[pure virtual]`

Delete a file physically from storage and the SRM namespace.

**Parameters:**

*req* The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.20 virtual SRMReturnCode Arc::SRMClient::requestBringOnline (SRMClientRequest & *req*) `[pure virtual]`

Submit a request to bring online files. If the synchronous property of the request object is false, this operation is asynchronous and the status of the request can be checked by calling requestBringOnlineStatus() with the request token in req which is assigned by this method. If the request is synchronous, this operation blocks until the file(s) are online or the timeout specified in the SRMClient constructor has passed.

**Parameters:**

*req* The request object

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

#### 5.75.3.21 virtual SRMReturnCode Arc::SRMClient::requestBringOnlineStatus (SRMClientRequest & *req*) `[pure virtual]`

Query the status of a request to bring files online. The SURLs map of the request object is updated if the status of any files in the request has changed. requestBringOnline() but be called before this method.

**Parameters:**

*req* The request object to query the status of

**Returns:**

SRMReturnCode specifying outcome of operation

Implemented in Arc::SRM1Client, and Arc::SRM22Client.

### 5.75.4 Field Documentation

#### 5.75.4.1 std::string Arc::SRMClient::service_endpoint `[protected]`

URL of the service endpoint, eg httpg://srm.host.org:8443/srm/managerv2 All SURLs passed to methods must correspond to this endpoint.

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

- SRMClient.h

## 5.76 Arc::SRMClientRequest Class Reference

Class to represent a SRM request.

```
#include <SRMClientRequest.h>
```

### Public Member Functions

- SRMClientRequest (const std::list< std::string > &urls) throw (SRMInvalidRequestException)
- SRMClientRequest (const std::string &url="", const std::string &id="") throw (SRMInvalidRequestException)
- std::string surl () const
- int waiting_time () const
- void finished_success ()
- void finished_partial_success ()
- void finished_error ()
- void finished_abort ()
- void wait (int t=0)
- void cancelled ()
- SRMRequestStatus status () const

### 5.76.1 Detailed Description

Class to represent a SRM request. It may be used for multiple operations, for example calling getTURLs() sets the request token in the request object (for a v2.2 client) and then same object is passed to releaseGet().

### 5.76.2 Constructor & Destructor Documentation

#### 5.76.2.1 Arc::SRMClientRequest::SRMClientRequest (const std::list< std::string > & *urls*) throw (SRMInvalidRequestException) `[inline]`

Creates a request object with multiple SURLs. The URLs here are in the form srm://srm.host.org/path/to/file

#### 5.76.2.2 Arc::SRMClientRequest::SRMClientRequest (const std::string & *url* = `""`, const std::string & *id* = `""`) throw (SRMInvalidRequestException) `[inline]`

Creates a request object with a single SURL. The URL here is in the form srm://srm.host.org/path/to/file

### 5.76.3 Member Function Documentation

#### 5.76.3.1 int Arc::SRMClientRequest::waiting_time () const `[inline]`

Get waiting time. A waiting time of zero means no estimate was given by the remote service.

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

- SRMClientRequest.h

## 5.77 SRMFileInfo Class Reference

```
#include <SRMInfo.h>
```

### 5.77.1 Detailed Description

Info about a particular entry in the SRM info file

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

- SRMInfo.h

## 5.78 Arc::SRMFileMetaData Struct Reference

SRM-related file metadata.

```
#include <SRMClient.h>
```

### 5.78.1 Detailed Description

SRM-related file metadata.

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

- SRMClient.h

## 5.79 SRMInfo Class Reference

```
#include <SRMInfo.h>
```

### 5.79.1 Detailed Description

Represents SRM info stored in file. A combination of host and SRM version make a unique entry.

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

- SRMInfo.h

## 5.80 Arc::SRMInvalidRequestException Class Reference

General exception to represent a bad SRM request.

```
#include <SRMClientRequest.h>
```

### 5.80.1 Detailed Description

General exception to represent a bad SRM request.

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

- SRMClientRequest.h

# 5.81 SRMURL Class Reference

## Public Member Functions

- SRMURL (std::string url)
- const std::string & Endpoint (void) const
- void SetSRMVersion (const std::string &version)
- std::string FileName (void) const
- std::string ContactURL (void) const
- std::string BaseURL (void) const
- std::string ShortURL (void) const
- std::string FullURL (void) const
- bool PortDefined ()

## 5.81.1 Constructor & Destructor Documentation

### 5.81.1.1 SRMURL::SRMURL (std::string *url*)

Examples shown for functions below assume the object was initiated with srm://srm.ndgf.org/pnfs/ndgf.org/data/atlas/disk/user/user.mlassnig.dataset.1/dummyfile3

## 5.81.2 Member Function Documentation

### 5.81.2.1 std::string SRMURL::BaseURL (void) const

eg srm://srm.ndgf.org:8443/srm/managerv2?SFN=

### 5.81.2.2 std::string SRMURL::ContactURL (void) const

eg httpg://srm.ndgf.org:8443/srm/managerv2

### 5.81.2.3 const std::string& SRMURL::Endpoint (void) const `[inline]`

eg /srm/managerv2

### 5.81.2.4 std::string SRMURL::FileName (void) const `[inline]`

eg pnfs/ndgf.org/data/atlas/disk/user/user.mlassnig.dataset.1/dummyfile3

### 5.81.2.5 std::string SRMURL::FullURL (void) const

eg srm://srm.ndgf.org:8443/srm/managerv2?SFN=pnfs/ndgf.org/data/atlas/disk/user/user.mlassnig.dataset.1/dummyfile3

### 5.81.2.6 bool SRMURL::PortDefined () `[inline]`

Was the port number given in the constructor?

#### 5.81.2.7 void SRMURL::SetSRMVersion (const std::string & *version*)

Possible values of version are "1" and "2.2"

#### 5.81.2.8 std::string SRMURL::ShortURL (void) const

eg srm://srm.ndgf.org:8443/pnfs/ndgf.org/data/atlas/disk/user/user.mlassnig.dataset.1/dummyfile3

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

- SRMURL.h

## 5.82 ArcSHCLegacy::UnixMap Class Reference

### Data Structures

- struct **source_t**
- class **unix_user_t**

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

- unixmap.h

## 5.83 ArcSec::UsernameTokenSH Class Reference

Adds WS-Security Username Token into SOAP Header.

```
#include <UsernameTokenSH.h>
```

### 5.83.1 Detailed Description

Adds WS-Security Username Token into SOAP Header.

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

- UsernameTokenSH.h

# 5.84 ArcSHCLegacy::voms Struct Reference

```
#include <auth.h>
```

## Data Fields

- std::string server
- std::string voname
- std::vector< voms_attrs > attrs

## 5.84.1 Detailed Description

VOMS data

## 5.84.2 Field Documentation

### 5.84.2.1 std::vector<voms_attrs> ArcSHCLegacy::voms::attrs

User's characteristics

### 5.84.2.2 std::string ArcSHCLegacy::voms::server

The VOMS server DN, as from its certificate

### 5.84.2.3 std::string ArcSHCLegacy::voms::voname

The name of the VO to which the VOMS belongs

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

- auth.h

# 5.85 ArcSHCLegacy::voms_attrs Struct Reference

```
#include <auth.h>
```

## Data Fields

- std::string group
- std::string role
- std::string cap

## 5.85.1 Detailed Description

VOMS attributes

## 5.85.2 Field Documentation

### 5.85.2.1 std::string ArcSHCLegacy::voms_attrs::cap

user's capability

### 5.85.2.2 std::string ArcSHCLegacy::voms_attrs::group

user's group

### 5.85.2.3 std::string ArcSHCLegacy::voms_attrs::role

user's role

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

- auth.h

# 5.86 ArcSec::X509TokenSH Class Reference

Adds WS-Security X509 Token into SOAP Header.

```
#include <X509TokenSH.h>
```

## 5.86.1 Detailed Description

Adds WS-Security X509 Token into SOAP Header.

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

- X509TokenSH.h

## 5.87 ArcSec::XACMLAlgFactory Class Reference

Algorithm factory class for XACML.

`#include <XACMLAlgFactory.h>`

### Public Member Functions

- virtual CombiningAlg ∗ createAlg (const std::string &type)

### 5.87.1 Detailed Description

Algorithm factory class for XACML.

### 5.87.2 Member Function Documentation

#### 5.87.2.1 virtual CombiningAlg∗ ArcSec::XACMLAlgFactory::createAlg (const std::string & *type*) `[virtual]`

return a Alg object according to the "CombiningAlg" attribute in the <Policy> node; The XACMLAlgFactory itself will release the Alg objects

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

- XACMLAlgFactory.h

## 5.88 ArcSec::XACMLApply Class Reference

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

- XACMLApply.h

## 5.89 ArcSec::XACMLAttributeFactory Class Reference

Attribute factory class for XACML specified attributes.

```
#include <XACMLAttributeFactory.h>
```

### Public Member Functions

- virtual AttributeValue ∗ createValue (const Arc::XMLNode &node, const std::string &type)

### 5.89.1 Detailed Description

Attribute factory class for XACML specified attributes.

### 5.89.2 Member Function Documentation

#### 5.89.2.1 virtual AttributeValue∗ ArcSec::XACMLAttributeFactory::createValue (const Arc::XMLNode & *node*, const std::string & *type*) `[virtual]`

creat a AttributeValue according to the value in the XML node and the type; It should be the caller to release the AttributeValue Object

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

- XACMLAttributeFactory.h

# 5.90 ArcSec::XACMLAttributeProxy< TheAttribute > Class Template Reference

XACML specific AttributeProxy class.

```
#include <XACMLAttributeProxy.h>
```

## Public Member Functions

- virtual AttributeValue $*$ getAttribute (const Arc::XMLNode &node)

### 5.90.1 Detailed Description

**template<class TheAttribute> class ArcSec::XACMLAttributeProxy< TheAttribute >**

XACML specific AttributeProxy class.

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

- XACMLAttributeProxy.h

# 5.91 ArcSec::XACMLCondition Class Reference

XACMLCondition class to parse and operate XACML specific <Condition> node.

```
#include <XACMLCondition.h>
```

## Public Member Functions

- XACMLCondition (Arc::XMLNode &node, EvaluatorContext ∗ctx)

## 5.91.1 Detailed Description

XACMLCondition class to parse and operate XACML specific <Condition> node.

## 5.91.2 Constructor & Destructor Documentation

### 5.91.2.1 ArcSec::XACMLCondition::XACMLCondition (Arc::XMLNode & *node*, EvaluatorContext ∗ *ctx*)

Constructor -

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

- XACMLCondition.h

## 5.92 ArcSec::XACMLEvaluationCtx Class Reference

EvaluationCtx, in charge of storing some context information for evaluation, including Request, current time, etc.

```
#include <XACMLEvaluationCtx.h>
```

### Public Member Functions

- XACMLEvaluationCtx (Request ∗request)

### 5.92.1 Detailed Description

EvaluationCtx, in charge of storing some context information for evaluation, including Request, current time, etc.

### 5.92.2 Constructor & Destructor Documentation

#### 5.92.2.1 ArcSec::XACMLEvaluationCtx::XACMLEvaluationCtx (Request ∗ *request*)

Construct a new EvaluationCtx based on the given request

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

- XACMLEvaluationCtx.h

## 5.93 ArcSec::XACMLEvaluator Class Reference

Execute the policy evaluation, based on the request and policy.

```
#include <XACMLEvaluator.h>
```

### Public Member Functions

- virtual Response ∗ evaluate (Request ∗request)

### 5.93.1 Detailed Description

Execute the policy evaluation, based on the request and policy.

### 5.93.2 Member Function Documentation

#### 5.93.2.1 virtual Response∗ ArcSec::XACMLEvaluator::evaluate (Request ∗ *request*) [virtual]

Evaluate the request based on the policy information inside PolicyStore

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

- XACMLEvaluator.h

## 5.94 ArcSec::XACMLFnFactory Class Reference

Function factory class for XACML specified attributes.

```
#include <XACMLFnFactory.h>
```

### Public Member Functions

- virtual Function ∗ createFn (const std::string &type)

### 5.94.1 Detailed Description

Function factory class for XACML specified attributes.

### 5.94.2 Member Function Documentation

#### 5.94.2.1 virtual Function∗ ArcSec::XACMLFnFactory::createFn (const std::string & *type*) [virtual]

return a Function object according to the "Function" attribute in the XML node; The XACMLFnFactory itself will release the Function objects

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

- XACMLFnFactory.h

# 5.95 ArcSec::XACMLPDP Class Reference

XACMLPDP - PDP which can handle the XACML specific request and policy schema.

```
#include <XACMLPDP.h>
```

## 5.95.1 Detailed Description

XACMLPDP - PDP which can handle the XACML specific request and policy schema.

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

- XACMLPDP.h

# 5.96 ArcSec::XACMLPolicy Class Reference

XACMLPolicy class to parse and operate XACML specific <Policy> node.

```
#include <XACMLPolicy.h>
```

## Public Member Functions

- XACMLPolicy (Arc::PluginArgument ∗parg)
- XACMLPolicy (const Arc::XMLNode node, Arc::PluginArgument ∗parg)
- XACMLPolicy (const Arc::XMLNode node, EvaluatorContext ∗ctx, Arc::PluginArgument ∗parg)
- virtual void make_policy ()

## 5.96.1 Detailed Description

XACMLPolicy class to parse and operate XACML specific <Policy> node.

## 5.96.2 Constructor & Destructor Documentation

### 5.96.2.1 ArcSec::XACMLPolicy::XACMLPolicy (Arc::PluginArgument ∗ *parg*)

Constructor

### 5.96.2.2 ArcSec::XACMLPolicy::XACMLPolicy (const Arc::XMLNode *node*, Arc::PluginArgument ∗ *parg*)

Constructor

### 5.96.2.3 ArcSec::XACMLPolicy::XACMLPolicy (const Arc::XMLNode *node*, EvaluatorContext ∗ *ctx*, Arc::PluginArgument ∗ *parg*)

Constructor -

## 5.96.3 Member Function Documentation

### 5.96.3.1 virtual void ArcSec::XACMLPolicy::make_policy () `[virtual]`

Parse XMLNode, and construct the low-level Rule object

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

- XACMLPolicy.h

## 5.97 ArcSec::XACMLRequest Class Reference

### Public Member Functions

- virtual const char ∗ getEvalName () const
- virtual const char ∗ getName () const

### 5.97.1 Member Function Documentation

#### 5.97.1.1 virtual const char∗ ArcSec::XACMLRequest::getEvalName () const [inline, virtual]

Get the name of corresponding evaulator

#### 5.97.1.2 virtual const char∗ ArcSec::XACMLRequest::getName (void) const [inline, virtual]

Get the name of this request

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

- XACMLRequest.h

# 5.98 ArcSec::XACMLRule Class Reference

XACMLRule class to parse XACML specific <Rule> node.

```
#include <XACMLRule.h>
```

## 5.98.1 Detailed Description

XACMLRule class to parse XACML specific <Rule> node.

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

- XACMLRule.h

# 5.99 ArcSec::XACMLTarget Class Reference

XACMLTarget class to parse and operate XACML specific <Target> node.

```
#include <XACMLTarget.h>
```

## Public Member Functions

- XACMLTarget (Arc::XMLNode &node, EvaluatorContext ∗ctx)

## 5.99.1 Detailed Description

XACMLTarget class to parse and operate XACML specific <Target> node.

## 5.99.2 Constructor & Destructor Documentation

### 5.99.2.1 ArcSec::XACMLTarget::XACMLTarget (Arc::XMLNode & *node*, EvaluatorContext ∗ *ctx*)

Constructor -

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

- XACMLTarget.h

## 5.100 ArcSec::XACMLTargetMatch Class Reference

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

- XACMLTarget.h

## 5.101 ArcSec::XACMLTargetMatchGroup Class Reference

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

- XACMLTarget.h

## 5.102 ArcSec::XACMLTargetSection Class Reference

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

- XACMLTarget.h

# Index