Hosting Environment (Daemon) Chain Components

Generated by Doxygen 1.5.7.1

Tue Apr 28 11:10:45 2009

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 **DelegationCollector**
- class **DelegationSecAttr**
- class **DelegationMultiSecAttr**
- 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 **ArcEvaluator**

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

- class **ArcFnFactory**

  Function factory class for Arc specified attributes.

- class **ArcPDP**

  ***ArcPDP** (p. 18) - PDP which can handle the Arc specific request and policy schema.*

- class **ArcPolicy**

__ArcPolicy__ (p. 20) class to parse and operate Arc specific <Policy> node.

- class **ArcRequest**
- class **ArcRequestItem**

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

- class **ArcRule**

  __ArcRule__ (p. 22) class to parse Arc specific <Rule> node.

- class **CountPDP**

  __CountPDP__ (p. 23) - PDP which can handle the Arc specific request and policy schema.

- class **DelegationPDP**
- class **DelegationSH**
- class **DenyPDP**

  This PDP always returns false (deny).

- class **GACLEvaluator**
- class **GACLPDP**
- class **GACLPolicy**
- class **GACLRequest**
- class **ArcPDPServiceInvoker**

  __ArcPDPServiceInvoker__ (p. 19) - client which will invoke pdpservice.

- class **SAML2SSO_AssertionConsumerSH**

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

- 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 **XACMLPolicy**

  __XACMLPolicy__ (p. 59) class to parse and operate XACML specific <Policy> node.

- class **XACMLRequest**
- class **XACMLRule**

  __XACMLRule__ (p. 60) class to parse XACML specific <Rule> node.

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

  __XACMLTarget__ (p. 61) 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>

4.1.2.3 typedef std::list<AndList> ArcSec::OrList

OrList - include items inside one <Subjects> (or <Resources> <Actions> <Conditions>).

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** (p. 12) 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.

5.4.2 Member Function Documentation

5.4.2.1 template<class TheAttribute > AttributeValue ∗ ArcSec::ArcAttributeProxy< TheAttribute >::getAttribute (const Arc::XMLNode & *node*) `[inline, virtual]`

Implementation of getAttribute method.

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)

Protected Member Functions

- bool **MakePDPs** (Arc::Config ∗cfg)

5.5.1 Detailed Description

Tests message against list of PDPs.

This class implements SecHandler interface. It's **Handle()** (p. 15) method runs provided Message instance against all PDPs specified in configuration. If any of PDPs returns positive result **Handle()** (p. 15) 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*) `[virtual]`

Get authorization decision

5.5.2.2 bool ArcSec::ArcAuthZ::MakePDPs (Arc::Config ∗ *cfg*) `[protected]`

Create PDP according to conf info

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

- ArcAuthZ.h

5.6 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)

5.6.1 Detailed Description

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

5.6.2 Member Function Documentation

5.6.2.1 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

5.7 ArcSec::ArcFnFactory Class Reference

Function factory class for Arc specified attributes.

`#include <ArcFnFactory.h>`

Public Member Functions

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

5.7.1 Detailed Description

Function factory class for Arc specified attributes.

5.7.2 Member Function Documentation

5.7.2.1 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** (p. 17) itself will release the Function objects

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

- ArcFnFactory.h

5.8 ArcSec::ArcPDP Class Reference

ArcPDP (p. 18) - PDP which can handle the Arc specific request and policy schema.

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

5.8.1 Detailed Description

ArcPDP (p. 18) - PDP which can handle the Arc specific request and policy schema.

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

- ArcPDP.h

5.9 ArcSec::ArcPDPServiceInvoker Class Reference

ArcPDPServiceInvoker (p. 19) - client which will invoke pdpservice.

```
#include <ArcPDPServiceInvoker.h>
```

5.9.1 Detailed Description

ArcPDPServiceInvoker (p. 19) - client which will invoke pdpservice.

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

- ArcPDPServiceInvoker.h

5.10 ArcSec::ArcPolicy Class Reference

ArcPolicy (p. 20) class to parse and operate Arc specific <Policy> node.

```
#include <ArcPolicy.h>
```

Public Member Functions

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

5.10.1 Detailed Description

ArcPolicy (p. 20) class to parse and operate Arc specific <Policy> node.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 ArcSec::ArcPolicy::ArcPolicy (void)

Constructor

5.10.2.2 ArcSec::ArcPolicy::ArcPolicy (const Arc::XMLNode *node*)

Constructor

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

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::ArcRequestItem Class Reference

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

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

5.11.1 Detailed Description

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

Specified **ArcRequestItem** (p. 21) which can parse Arc request formate

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

- ArcRequestItem.h

5.12 ArcSec::ArcRule Class Reference

ArcRule (p. 22) class to parse Arc specific <Rule> node.

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

5.12.1 Detailed Description

ArcRule (p. 22) class to parse Arc specific <Rule> node.

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

- ArcRule.h

5.13 ArcSec::CountPDP Class Reference

CountPDP (p. 23) - PDP which can handle the Arc specific request and policy schema.

```
#include <CountPDP.h>
```

5.13.1 Detailed Description

CountPDP (p. 23) - PDP which can handle the Arc specific request and policy schema.

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

- CountPDP.h

5.14 ArcSec::DelegationPDP Class Reference

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

5.14.1 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

5.15 ArcSec::DenyPDP Class Reference

This PDP always returns false (deny).

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

5.15.1 Detailed Description

This PDP always returns false (deny).

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

- DenyPDP.h

5.16 Arc::LDAPQuery Class Reference

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

Public Member Functions

- **LDAPQuery** (const std::string &ldaphost, int ldapport, bool anonymous=true, const std::string &usersn="", int timeout=TIMEOUT)
- ∼**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)

5.16.1 Detailed Description

LDAPQuery (p. 26) class; querying of LDAP servers.

5.16.2 Constructor & Destructor Documentation

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

Constructs a new **LDAPQuery** (p. 26) object and sets connection options. The connection is first established when calling Query.

5.16.2.2 Arc::LDAPQuery::∼LDAPQuery ()

Destructor. Will disconnect from the ldapserver if still connected.

5.16.3 Member Function Documentation

5.16.3.1 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.

5.16.3.2 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:

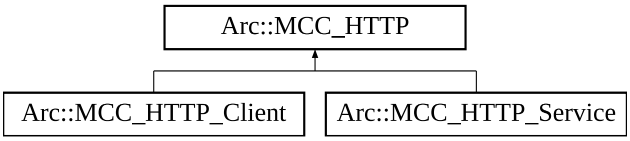
- LDAPQuery.h

5.17 Arc::MCC_HTTP Class Reference

A base class for HTTP client and service MCCs.

```
#include <MCCHTTP.h>
```

Inheritance diagram for Arc::MCC_HTTP::



5.17.1 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.

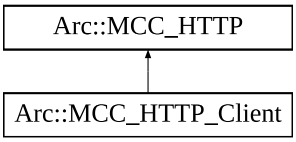
The documentation for this class was generated from the following file:

- MCCHTTP.h

5.18 Arc::MCC_HTTP_Client Class Reference

```
#include <MCCHTTP.h>
```

Inheritance diagram for Arc::MCC_HTTP_Client::



5.18.1 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.

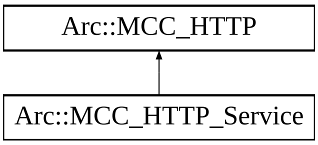
The documentation for this class was generated from the following file:

- MCCHTTP.h

5.19 Arc::MCC_HTTP_Service Class Reference

`#include <MCCHTTP.h>`

Inheritance diagram for Arc::MCC_HTTP_Service::



5.19.1 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** (p. 38). 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:

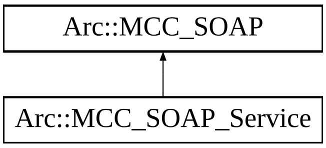
- MCCHTTP.h

5.20 Arc::MCC_SOAP Class Reference

A base class for SOAP client and service MCCs.

`#include <MCCSOAP.h>`

Inheritance diagram for Arc::MCC_SOAP::



5.20.1 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.

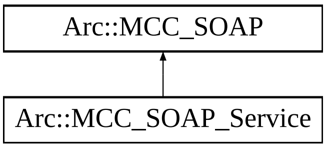
The documentation for this class was generated from the following file:

- MCCSOAP.h

5.21 Arc::MCC_SOAP_Service Class Reference

```
#include <MCCSOAP.h>
```

Inheritance diagram for Arc::MCC_SOAP_Service::



5.21.1 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:

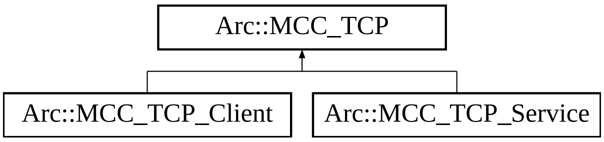
- MCCSOAP.h

5.22 Arc::MCC_TCP Class Reference

A base class for TCP client and service MCCs.

```
#include <MCCTCP.h>
```

Inheritance diagram for Arc::MCC_TCP::



5.22.1 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.

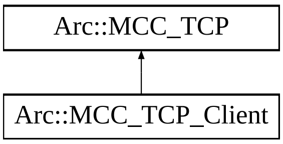
The documentation for this class was generated from the following file:

- MCCTCP.h

5.23 Arc::MCC_TCP_Client Class Reference

```
#include <MCCTCP.h>
```

Inheritance diagram for Arc::MCC_TCP_Client::



5.23.1 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.

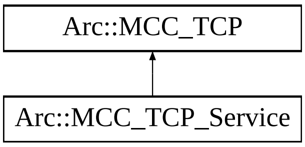
The documentation for this class was generated from the following file:

- MCCTCP.h

5.24 Arc::MCC_TCP_Service Class Reference

```
#include <MCCTCP.h>
```

Inheritance diagram for Arc::MCC_TCP_Service::



Data Structures

- class **mcc_tcp_exec_t**

Public Member Functions

- **MCC_TCP_Service** (Config ∗cfg)

5.24.1 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

5.24.2 Constructor & Destructor Documentation

5.24.2.1 Arc::MCC_TCP_Service::MCC_TCP_Service (Config ∗ *cfg*)

executing function for connection thread

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

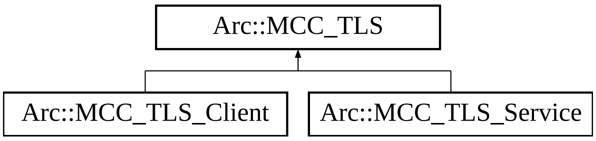
- MCCTCP.h

5.25 Arc::MCC_TLS Class Reference

A base class for TLS client and service MCCs.

```
#include <MCCTLS.h>
```

Inheritance diagram for Arc::MCC_TLS::



5.25.1 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.

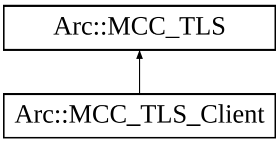
The documentation for this class was generated from the following file:

- MCCTLS.h

5.26 Arc::MCC_TLS_Client Class Reference

```
#include <MCCTLS.h>
```

Inheritance diagram for Arc::MCC_TLS_Client::



5.26.1 Detailed Description

This class is MCC implementing TLS client.

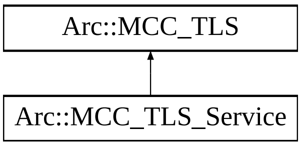
The documentation for this class was generated from the following file:

- MCCTLS.h

5.27 Arc::MCC_TLS_Service Class Reference

```
#include <MCCTLS.h>
```

Inheritance diagram for Arc::MCC_TLS_Service::



5.27.1 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** (p. 33), 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** (p. 28) calls the process() method of **MCC_TLS_Client** (p. 36) object, or **MCC_TCP_Service** (p. 34) calls the process() method of **MCC_TLS_Service** (p. 37) object. The "ssl" object is embeded in a payload called PayloadTLSSocket.

The process() method of **MCC_TLS_Service** (p. 37) 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

5.28 Arc::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)
- **PayloadHTTP** (int code, const std::string &reason)
- virtual const std::string & **Attribute** (const std::string &name)
- virtual const std::map< 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, int &size)
- bool **parse_header** (void)
- bool **get_body** (void)

Protected Attributes

- PayloadStreamInterface ∗ **stream_**
- bool **stream_own_**
- PayloadRawInterface ∗ **body_**
- bool **body_own_**
- std::string **uri_**
- int **version_major_**
- int **version_minor_**
- std::string **method_**
- int **code_**
- std::string **reason_**
- int **length_**
- bool **chunked_**
- bool **keep_alive_**
- std::map< std::string, std::string > **attributes_**

5.28.1 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.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 Arc::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.

5.28.2.2 Arc::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.

5.28.2.3 Arc::PayloadHTTP::PayloadHTTP (const std::string & *method*, const std::string & *url*)

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

5.28.2.4 Arc::PayloadHTTP::PayloadHTTP (int *code*, const std::string & *reason*, PayloadStreamInterface & *stream*)

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

5.28.2.5 Arc::PayloadHTTP::PayloadHTTP (int *code*, const std::string & *reason*)

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

5.28.3 Member Function Documentation

5.28.3.1 virtual void Arc::PayloadHTTP::Attribute (const std::string & *name*, const std::string & *value*) `[virtual]`

Sets HTTP header attribute 'name' to 'value'

5.28.3.2 virtual const std::string& Arc::PayloadHTTP::Attribute (const std::string & *name*) `[virtual]`

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

5.28.3.3 virtual const std::map<std::string,std::string>& Arc::PayloadHTTP::Attributes (void) `[virtual]`

Returns all HTTP header attributes.

5.28.3.4 virtual void Arc::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.

5.28.3.5 virtual bool Arc::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.

5.28.3.6 bool Arc::PayloadHTTP::get_body (void) [protected]

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

5.28.3.7 bool Arc::PayloadHTTP::parse_header (void) [protected]

Read HTTP header and fill internal variables

5.28.3.8 bool Arc::PayloadHTTP::read (char ∗ *buf*, int & *size*) [protected]

Read up to 'size' bytes from stream_

5.28.3.9 bool Arc::PayloadHTTP::readline (std::string & *line*) [protected]

Read from stream till

5.28.4 Field Documentation

5.28.4.1 std::map<std::string,std::string> Arc::PayloadHTTP::attributes_ [protected]

true if conection should not be closed after response

5.28.4.2 PayloadRawInterface∗ Arc::PayloadHTTP::body_ [protected]

if true stream_ is owned by this

5.28.4.3 bool Arc::PayloadHTTP::body_own_ [protected]

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

5.28.4.4 bool Arc::PayloadHTTP::chunked_ [protected]

Content-length of HTTP message

5.28.4.5 int Arc::PayloadHTTP::code_ [protected]

HTTP method being used or requested

5.28.4.6 bool Arc::PayloadHTTP::keep_alive_ [protected]

true if content is chunked

5.28.4.7 int Arc::PayloadHTTP::length_ [protected]

HTTP reason being sent or supplied

5.28.4.8 std::string Arc::PayloadHTTP::method_ [protected]

minor number of HTTP version - must be 0 or 1

5.28.4.9 std::string Arc::PayloadHTTP::reason_ [protected]

HTTP code being sent or supplied

5.28.4.10 PayloadStreamInterface* Arc::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_.

5.28.4.11 bool Arc::PayloadHTTP::stream_own_ [protected]

stream used to comminicate to outside

5.28.4.12 std::string Arc::PayloadHTTP::uri_ [protected]

if true body_ is owned by this

5.28.4.13 int Arc::PayloadHTTP::version_major_ [protected]

URI being contacted

5.28.4.14 int Arc::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

5.29 Arc::PayloadTCPSocket Class Reference

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

Public Member Functions

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

5.29.1 Detailed Description

This class extends PayloadStream with TCP socket specific features

5.29.2 Constructor & Destructor Documentation

5.29.2.1 Arc::PayloadTCPSocket::PayloadTCPSocket (const char ∗ *hostname*, int *port*, Logger & *logger*)

Constructor - connects to TCP server at specified hostname:port

5.29.2.2 Arc::PayloadTCPSocket::PayloadTCPSocket (const std::string *endpoint*, Logger & *logger*)

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

5.29.2.3 Arc::PayloadTCPSocket::PayloadTCPSocket (int *s*, Logger & *logger*) `[inline]`

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

5.29.2.4 Arc::PayloadTCPSocket::PayloadTCPSocket (PayloadTCPSocket & *s*) `[inline]`

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

5.29.2.5 Arc::PayloadTCPSocket::PayloadTCPSocket (PayloadStream & *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

5.30 Arc::PayloadTLSStream Class Reference

```
#include <PayloadTLSStream.h>
```

Inherited by Arc::PayloadTLSMCC.

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_**

5.30.1 Detailed Description

Implemetation of PayloadStreamInterface for SSL handle.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 Arc::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.

5.30.2.2 virtual Arc::PayloadTLSStream::∼PayloadTLSStream (void) `[virtual]`

Destructor.

5.30.3 Member Function Documentation

5.30.3.1 X509∗ Arc::PayloadTLSStream::GetCert (void)

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

5.30.3.2 X509∗ Arc::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.

5.30.3.3 Arc::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.

5.30.4 Field Documentation

5.30.4.1 SSL∗ Arc::PayloadTLSStream::ssl_ `[protected]`

Timeout for read/write operations

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

- PayloadTLSStream.h

5.31 ArcSec::SAML2SSO_AssertionConsumerSH Class Reference

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

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

5.31.1 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

5.32 ArcSec::SimpleListPDP Class Reference

Tests X509 subject against list of subjects in file.

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

5.32.1 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

5.33 SRMClient Class Reference

`#include <SRMClient.h>`

Inherited by SRM1Client, and SRM22Client.

Public Member Functions

- bool **connect** (void)
- bool **disconnect** (void)
- virtual ∼**SRMClient** ()
- void **Timeout** (int t)
- std::string **getVersion** ()
- virtual SRMReturnCode **ping** (std::string &**version**, bool report_error=true)=0
- virtual SRMReturnCode **getSpaceTokens** (std::list< std::string > &tokens, std::string description="")=0
- virtual SRMReturnCode **getRequestTokens** (std::list< std::string > &tokens, std::string description="")=0
- virtual bool **getTURLs** (**SRMClientRequest** &req, std::list< std::string > &urls)=0
- virtual SRMReturnCode **requestBringOnline** (**SRMClientRequest** &req)=0
- virtual SRMReturnCode **requestBringOnlineStatus** (**SRMClientRequest** &req)=0
- virtual bool **putTURLs** (**SRMClientRequest** &req, std::list< std::string > &urls, unsigned long long size=0)=0
- virtual bool **releaseGet** (**SRMClientRequest** &req)=0
- virtual bool **releasePut** (**SRMClientRequest** &req)=0
- virtual bool **release** (**SRMClientRequest** &req)=0
- virtual bool **abort** (**SRMClientRequest** &req)=0
- virtual bool **info** (**SRMClientRequest** &req, std::list< struct **SRMFileMetaData** > &metadata, const int recursive=0)=0
- virtual bool **remove** (**SRMClientRequest** &req)=0
- virtual bool **copy** (**SRMClientRequest** &req, const std::string &source)=0

Static Public Member Functions

- static **SRMClient** ∗ **getInstance** (std::string url, time_t timeout=300, SRMVersion srm_version=SRM_VNULL)

Protected Attributes

- std::string **service_endpoint**
- Arc::HTTPSClientSOAP ∗ **csoap**
- SRMImplementation **implementation**
- std::string **version**

Static Protected Attributes

- static time_t **request_timeout**
- static Arc::Logger **logger**

5.33.1 Detailed Description

A client interface to the SRM protocol. Instances of SRM clients are created by calling the **getInstance()** (p. 49) 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.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 virtual SRMClient::∼SRMClient () `[inline, virtual]`

empty destructor

5.33.3 Member Function Documentation

5.33.3.1 virtual bool 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:

True on success, false on failure

5.33.3.2 bool SRMClient::connect (void) `[inline]`

Establish a connection to the service

References csoap.

5.33.3.3 virtual bool 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:

True on success, false on failure

5.33.3.4 bool SRMClient::disconnect (void) `[inline]`

Disconnect from the service and destroy the connection

References csoap.

5.33.3.5 static SRMClient* SRMClient::getInstance (std::string *url*, time_t *timeout* = `300`, SRMVersion *srm_version* = `SRM_VNULL`) `[static]`

Returns an **SRMClient** (p. 47) instance with the required protocol version. This must be used to create **SRMClient** (p. 47) instances. Specifying a version explicitly forces creation of a client with that version.

Parameters:

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.

version If this is non-NULL a client instance of this version is returned.

5.33.3.6 virtual SRMReturnCode SRMClient::getRequestTokens (std::list< std::string > & *tokens*, 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:

True on success, false on failure

5.33.3.7 virtual SRMReturnCode SRMClient::getSpaceTokens (std::list< std::string > & *tokens*, 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:

True on success, false on failure

5.33.3.8 virtual bool SRMClient::getTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*) `[pure virtual]`

If the user wishes to copy a file from somewhere, **getTURLs()** (p. 49) is called to retrieve the transport URL to copy the file from.

Parameters:

req The request object

urls A list of TURLs filled by the method

Returns:

True on success, false on failure

5.33.3.9 std::string SRMClient::getVersion () [inline]

Returns the version of the SRM protocol used by this instance

References version.

5.33.3.10 virtual bool SRMClient::info (SRMClientRequest & *req*, std::list< struct SRMFileMetaData > & *metadata*, const int *recursive* = 0) [pure virtual]

Returns information on a file or files (v2.2 and higher) stored in an SRM, such as file size, checksum and estimated access latency.

Parameters:

req The request object

metadata A list of structs filled with file information

recursive The level of recursion into sub directories

Returns:

True on success, false on failure

See also:

SRMFileMetaData (p. 56)

5.33.3.11 virtual SRMReturnCode 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 supresses 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

5.33.3.12 virtual bool SRMClient::putTURLs (SRMClientRequest & *req*, std::list< std::string > & *urls*, unsigned long long *size* = 0) [pure virtual]

If the user wishes to copy a file to somewhere, **putTURLs()** (p. 51) is called to retrieve the transport URL to copy the file to.

Parameters:

req The request object

urls A list of TURLs filled by the method

size The size of the file

Returns:

True on success, false on failure

5.33.3.13 virtual bool SRMClient::release (SRMClientRequest & *req*) [pure virtual]

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

Parameters:

req The request object

Returns:

True on success, false on failure

5.33.3.14 virtual bool SRMClient::releaseGet (SRMClientRequest & *req*) [pure virtual]

Should be called after a successful copy from SRM storage.

Parameters:

req The request object

Returns:

True on success, false on failure

5.33.3.15 virtual bool SRMClient::releasePut (SRMClientRequest & *req*) [pure virtual]

Should be called after a successful copy to SRM storage.

Parameters:

req The request object

Returns:

True on success, false on failure

5.33.3.16 virtual bool SRMClient::remove (SRMClientRequest & *req*) [pure virtual]

Delete a file physically from storage and the SRM namespace.

Parameters:

req The request object

Returns:

True on success, false on failure

5.33.3.17 virtual SRMReturnCode SRMClient::requestBringOnline (SRMClientRequest & *req*) [pure virtual]

Submit a request to bring online files. This operation is asynchronous and the status of the request can be checked by calling **requestBringOnlineStatus()** (p. 52) with the request token in req which is assigned by this method.

Parameters:

req The request object

Returns:

SRMReturnCode specifying outcome of operation

5.33.3.18 virtual SRMReturnCode SRMClient::requestBringOnlineStatus (SRMClientRequest & *req*) [pure virtual]

Query the status of a request to bring files online. The SURLs map is updated if the status of any files in the request has changed.

Parameters:

req The request object to query the status of

Returns:

SRMReturnCode specifying outcome of operation

5.33.3.19 void SRMClient::Timeout (int *t*) [inline]

set the request timeout

References request_timeout.

5.33.4 Field Documentation

5.33.4.1 Arc::HTTPSClientSOAP∗ SRMClient::csoap [protected]

SOAP client object

Referenced by connect(), and disconnect().

5.33.4.2 SRMImplementation SRMClient::implementation `[protected]`

The implementation of the server

5.33.4.3 Arc::Logger SRMClient::logger `[static, protected]`

Logger

5.33.4.4 time_t SRMClient::request_timeout `[static, protected]`

Timeout for requests to the SRM service

Referenced by Timeout().

5.33.4.5 std::string SRMClient::service_endpoint `[protected]`

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

5.33.4.6 std::string SRMClient::version `[protected]`

The version of the SRM protocol used

Referenced by getVersion().

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

- SRMClient.h

5.34 SRMClientRequest Class Reference

`#include <SRMClient.h>`

Public Member Functions

- **SRMClientRequest** (std::list< std::string > urls) throw (SRMInvalidRequestException)
- **SRMClientRequest** (std::string url="", std::string id="") throw (SRMInvalidRequestException)
- void **request_id** (int id)
- void **request_token** (char *token)
- void **file_ids** (std::list< int > ids)
- void **space_token** (std::string token)
- std::list< std::string > **surls** ()
- void **surl_statuses** (std::string surl, SRMFileLocality locality)
- void **surl_failures** (std::string surl, std::string reason)
- void **waiting_time** (int wait_time)
- void **finished_success** ()
- void **long_list** (bool list)

5.34.1 Detailed Description

Class to represent a request which 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().

5.34.2 Constructor & Destructor Documentation

5.34.2.1 SRMClientRequest::SRMClientRequest (std::list< std::string > *urls*) throw (SRMInvalidRequestException) `[inline]`

Creates a request object with multiple SURLs. The URLs here are in the form srm://srm.ndgf.org/pnfs/ndgf.org/data/atlas/disk/user/user.mlassnig.dataset.1/dummyfile3

5.34.2.2 SRMClientRequest::SRMClientRequest (std::string *url* = "", std::string *id* = "") throw (SRMInvalidRequestException) `[inline]`

Creates a request object with a single SURL. The URL here are in the form srm://srm.ndgf.org/pnfs/ndgf.org/data/atlas/disk/user/user.mlassnig.dataset.1/dummyfile3

5.34.3 Member Function Documentation

5.34.3.1 void SRMClientRequest::file_ids (std::list< int > *ids*) `[inline]`

set and get file id list

5.34.3.2 void SRMClientRequest::finished_success () `[inline]`

set and get status of request

5.34.3.3 void SRMClientRequest::long_list (bool *list*) `[inline]`

set and get long list flag

5.34.3.4 void SRMClientRequest::request_id (int *id*) `[inline]`

set and get request id

5.34.3.5 void SRMClientRequest::request_token (char ∗ *token*) `[inline]`

set and get request token

5.34.3.6 void SRMClientRequest::space_token (std::string *token*) `[inline]`

set and get space token

5.34.3.7 void SRMClientRequest::surl_failures (std::string *surl*, std::string *reason*) `[inline]`

set and get surl failures

5.34.3.8 void SRMClientRequest::surl_statuses (std::string *surl*, SRMFileLocality *locality*) `[inline]`

set and get surl statuses

5.34.3.9 std::list<std::string> SRMClientRequest::surls () `[inline]`

get SURLs

5.34.3.10 void SRMClientRequest::waiting_time (int *wait_time*) `[inline]`

set and get waiting time

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

- SRMClient.h

5.35 SRMFileMetaData Struct Reference

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

5.35.1 Detailed Description

File metadata

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

- SRMClient.h

5.36 ArcSec::UsernameTokenSH Class Reference

Adds WS-Security Username Token into SOAP Header.

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

5.36.1 Detailed Description

Adds WS-Security Username Token into SOAP Header.

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

- UsernameTokenSH.h

5.37 ArcSec::X509TokenSH Class Reference

Adds WS-Security X509 Token into SOAP Header.

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

5.37.1 Detailed Description

Adds WS-Security X509 Token into SOAP Header.

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

- X509TokenSH.h

5.38 ArcSec::XACMLPolicy Class Reference

XACMLPolicy (p. 59) class to parse and operate XACML specific <Policy> node.

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

Public Member Functions

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

5.38.1 Detailed Description

XACMLPolicy (p. 59) class to parse and operate XACML specific <Policy> node.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 ArcSec::XACMLPolicy::XACMLPolicy (Arc::XMLNode & *node*, EvaluatorContext ∗ *ctx*)

Constructor -

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

- XACMLPolicy.h

5.39 ArcSec::XACMLRule Class Reference

XACMLRule (p. 60) class to parse XACML specific <Rule> node.

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

5.39.1 Detailed Description

XACMLRule (p. 60) class to parse XACML specific <Rule> node.

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

- XACMLRule.h

5.40 ArcSec::XACMLTarget Class Reference

XACMLTarget (p. 61) class to parse and operate XACML specific <Target> node.

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

Public Member Functions

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

5.40.1 Detailed Description

XACMLTarget (p. 61) class to parse and operate XACML specific <Target> node.

5.40.2 Constructor & Destructor Documentation

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

Constructor -

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

- XACMLTarget.h

Index