Functional Specification for Sailfin Security

Author(s): Ramesh.Parthasarathy@Sun.COM,Venugopal Rao K

Contributors : Binod.Pg@Sun.COM, kristoffer.gronowski@ericsson.com, Ronald.Monzillo@Sun.COM

Version:1.0

Table of Contents

1 Introduction

Security framework in Sailfin is responsible for ensuring all configured security requirements are met before a SIP message is delivered to the SIP application. Security requirements to be supported by Sailfin is based on JSR 289 specification.

2 Functional Requirements

#	Description	Source	Milestone
1	Digest authentication implementation RFC 2617	JSR 289	M1
2	Support for P-asserted identity header RFC3325.	JSR 289	M2 & M3
3	Integration of RFC 2617 with Web container		M1
4	SIP declarative security and Programmatic security(isUserInRole(),runAs(),getRemoteUser(),getUserPrincipal()).	JSR 289	M1 & M2
5	SIPs, TLS	JSR 289	M1

Please refer to Appendix A for detailed synopsis of requirements from RFC 3325 and RFC 3261

3 Design Overview

<Discuss the core concepts and design. Provide conceptual diagrams, if they would be helpful. Show how this sub-system/feature co-exists with other sub-systems. You may write 1-4 pages (can be shorter or longer). This section is should be a map to navigate well documented code!>

3.1 Usecases :

3.1.1 Digest Authentication to Proxy with SSL

In this scenario Client( ) authenticates with the Proxy using Digest Password and the transport is secured using SSL. The user must be in a **“Manager”** role to invoke any specified method.

Client:

The client uses AuthInfo API to pass the username and password along with the realm when 407 is returned.

```
String realm = sipServeltResponse.getChallengeRealm();
AuthInfo authInfo = sipFactory.createAuthInfo();
authInfo.addAuthInfo(statusCode, realm,userName, password);

SipServletRequest origSReq = sipServletResponse.getSipRequest();

SipServletRequest newSReq
=B2buaHelper.createRequest(origSReq,sipServletResponse.getSession(
),null);

sipServletRequest.addAuthHeader(sipServletResponse,authInfo);
newSReq.send();
```

Server:

The server is configured for Digest authentication with appropriate realm and the configuration is as follows.

```
<security-constraint>
    <display-name>UserConstraint1</display-name>
    <auth-constraint>
        <description/>
        <role-name>manager</role-name>
    </auth-constraint>
    <user-data-constraint>
        <description>CONFIDENTIAL</description>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
    <proxy-authenticate/>
    <resource-collection>
        <resource-name>Resource</resource-name>
        <description/>
        <servlet-name>ProxyServlet</servlet-name>
        <sip-method>INVITE</sip—method>
        <sip-method>REGISTER</sip--method>
    </resource-collection>
</security-constraint>

<login-config>
    <auth-method>DIGEST</auth-method>
    <realm-name>sun-proxy-india.com</realm-name>
</login-config>
```

the **sun-sip.xml** will have the following configuration.

```
<security-role-mapping>
  <role-name>Manager</role-name>
  <group-name>AuthorizedManagers</group-name>
</security-role-mapping>
```

3.1.2 Digest Authentication with P-Asserted Identity.

In this scenario Client authenticates with the Proxy using Digest Password and the transport is secured using SSL. The user must be in a **“Manager”** role to invoke any specified method. The Proxy in turn forwards the request to two other proxy servers Proxy A and Proxy B. Proxy A is part of the trust domain where as proxy B is not. The user has Privacy header set.

Originating Proxy will not send P-Asserted Identity header to Proxy B but will forward P-Asserted identity header to Proxy A.

3.2 Authentication

JSR 289 requires us to support Digest Authentication and P-Asserted Identity. This section of the document describes design and implementation details of Digest and P-Asserted Identity.

3.2.1 Digest Authentication.

Sailfin implementation of Digest authentication is based on RFC 2617. When a SIP servlet is configured for Digest authentication, Sailfin will challenge the client with www-Authenticate or Proxy-Authenticate header. Sailfin will send the following arguments in the challenge.

Sample for 401:

```
WWW-Authenticate: Digest

Nonce=”<generated nonce .....>”

qop=”auth,auth-int”

realm=”<configured realm name>”

algorithm=”MD5”

opaque=””
```

Sailfin will support handling of 401(Unauthorized) and 407 (proxy authenticate required) messages by requesting the application developer to provide the authentication information. Realm name which is required by the user to provide appropriate authentication information is exposed to the application via **`SipServletResponse.getChallengeRealm().`** As part of sailfin effort we have also enabled Digest Authentication in WebContainer.

3.2.1.1 Digest Authentication API's

- **DigestAlgorithmParameter :**

  This interface represents parameters that are used in Digest calculation.

  Methods:

```
public byte[] getDelimiter();

public byte[] getValue() ;

public String getAlgorithm();

public String getName();
```

getDelimiter method returns the delimiter to be used for digest calculation.

getValue returns the parameter value.

getAlgorithm returns the algorithm to be used for digest calculation. eg: MD5 getName returns the name of the algorithm parameter.

- ## NestedDigestAlgoParam :

This interface represents a group of authentication parameters interface along with DigestAlgorithmParameter represents various parameters like nonce,cnonce,qop,nonce-count which form the data part of the Http digest calculation. Based on the **qop** value the format method returns the appropriate representation of data part.

When qop =auth

```
   unq(nonce-value)”:”H(Method ":" digest-uri-value)
```

when qop = auth-int

```
 unq(nonce-value)":" nc-value":" unq(cnonce-value)":" unq(qop-
value)":" H(Method ":" digest-uri-value ":" H(entity-body))
```

Methods :

```
 public AlgorithmParameterSpec[] getNestedParams();
```

getNestedParams method returns a array of AlgorithmParameterSpec eg: an array of DigestAlgorithmParameters.

- ## Key :

This Interface represents the key used in the digest calculation. The password from the back end is obtained using this Key.

```
   username+”:”+realm-name+”:”+”user-password”.
```

Methods:

```
   public java.lang.String getRealmName();

   public java.lang.String getUsername();
```

getRealmName method returns the realm name the user is associated with.

getUsername method returns the user name .

- **DigestParameterGenerator:**

  DigestParameterGenerator abstract class supports creation of DigestAlgorithmParamter objects from different sources eg: HttpServletRequest , SipServletRequest etc.

  Methods :

```
public abstract DigestAlgorithmParameter[]
generateParameters(AlgorithmParameterSpec value)  throws
InvalidAlgorithmParameterException;
```

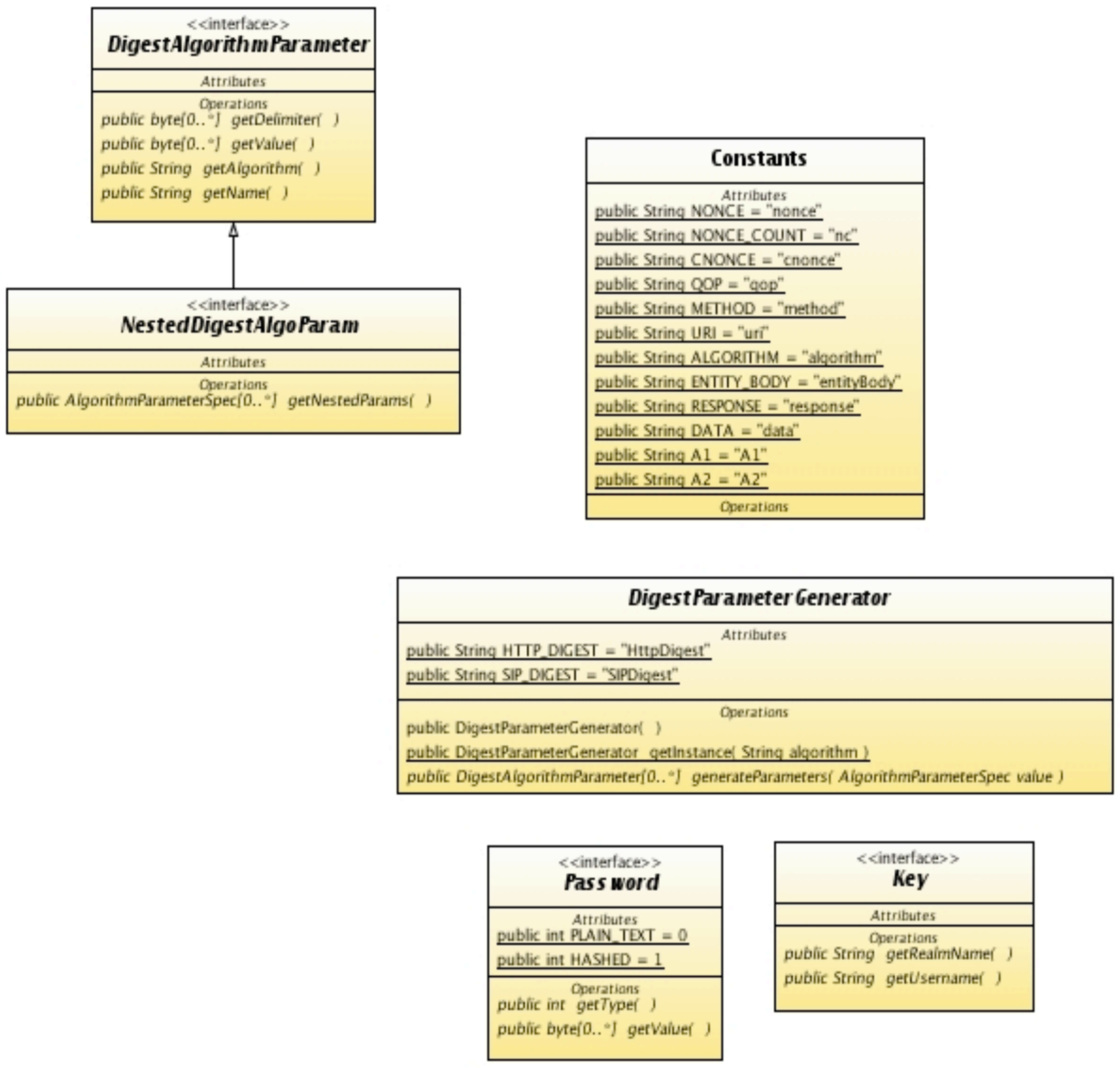

<<interface>>
DigestAlgorithmParameter
Attributes
Operations
public byte[0..*] getDelimiter( )
public byte[0..*] getValue( )
public String getAlgorithm( )
public String getName( )
<<interface>>
NestedDigestAlgoParam
Attributes
Operations
public AlgorithmParameterSpec[0..*] getNestedParams( )
Constants
Attributes
public String NONCE = "nonce"
public String NONCE_COUNT = "nc"
public String CNONCE = "cnonce"
public String QOP = "qop"
public String METHOD = "method"
public String URI = "uri"
public String ALGORITHM = "algorithm"
public String ENTITY_BODY = "entityBody"
public String RESPONSE = "response"
public String DATA = "data"
public String A1 = "A1"
public String A2 = "A2"
Operations
DigestParameterGenerator
Attributes
public String HTTP_DIGEST = "HttpDigest"
public String SIP_DIGEST = "SIPDigest"
Operations
public DigestParameterGenerator( )
public DigestParameterGenerator getInstance(String algorithm)
public DigestAlgorithmParameter[0..*] generateParameters(AlgorithmParameterSpec value)
<<interface>>
Password
Attributes
public int PLAIN_TEXT = 0
public int HASHED = 1
Operations
public int getType( )
public byte[0..*] getValue( )
<<interface>>
Key
Attributes
Operations
public String getRealmName( )
public String getUsername( )

Illustration 1: Digest Interfaces

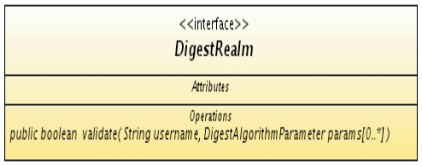
3.2.1.2 Digest Authentication Realm.

Realms which plan to support Digest authentication will have to implement DigestRealm interface. This interfaces defines validate method which accepts the DigestAlgorithmParameters along with the username. The server digest object will not have password and is the responsibility of the provider to configure realm to provide the same. The realm can provide a plain text password or pre calculated digest password.

Methods :

```
public boolean validate(String username,DigestAlgorithmParameter
params[]);
```

Illustration 2: Digest Realm


<<interface>>
DigestRealm
Attributes
Operations
public boolean validate(String username, DigestAlgorithmParameter params[0..*])

3.2.2 P-Asserted Identity Authentication

The P-Asserted-Identity(RFC 3325) header field is used among trusted SIP entities (typically intermediaries) to carry the identity of the authenticated user sending SIP messages. In order to prevent identity information to be sent to non trusted SIP entities and to prevent use of identity information from messages received from non trusted SIP entities, Sailfin uses two forms of configuration to determine trusted SIP entities. This configuration is reference by applications using a property named `“trust-id-ref”`.

3.2.2.1 Configuring Trust Domains

Trust among domains/hosts(on n/w) is achieved in Sailfin using static configuration of trusted hosts.

1. **Static Configuration :** Hosts that are trusted needs to be configured on the client and server side. “Security-service” element in domain.xml is the parent element for identity-assertion-trust.

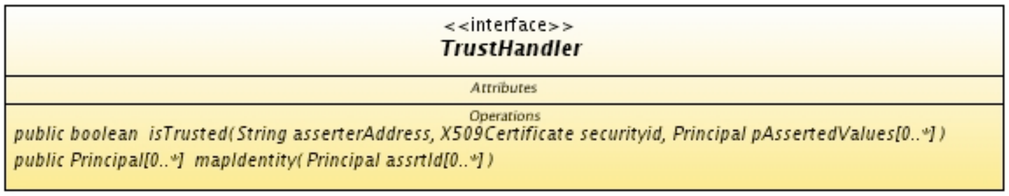
```
<identity-assertion-trust trust-id=”sun-config”>*
   <trusted-entity>*
      <ip-address></ip-address>
      <domain-name><domain-name>*
   </trusted-entity>
   <trust-handler classname =””/>
</identity-assertion-trust>
```

```
<!ELEMENT security-service
 (auth-realm+, jacc-provider+, audit-module*, message-security-
config*, identy-assertion-trust*, property*)>

 <!--
   identity-assertion-trust represents trust domain configuration
   information as per RFC 3325.
 -->

<!ELEMENT identity-assertion-trust ((trusted-entity* | trust-
handler))>

<!--
  trust-id is a identifier used to uniquely identify this
  identity-assertion-trust element.
-->
<!ATTLIST identity-assertion-trust trust-id CDATA #REQUIRED>

<!--
 trusted-entity represents network entities(in the form of
 ipaddress/domain) that are to be trusted. This information may
 be different for incoming and outgoing messages.
-->

<!ELEMENT trusted-entity (ip-address, domain-name?, principal?)>

<!--
 trusted-as with value intermediate represents configuration
```

```
  information for incoming messages,if it has value destination
  then configuration under trusted-entity is applied to outgoing
  messages.
-->
<!ATTLIST trusted-entity trusted-as (intermediate|destination)
#IMPIED>

<!--
   trust-handler users can provide custom implementation to
   determine trust(as per RFC 3325) and to convert user identity
   to a format recognized by the system. Attribute class-name
   specifies the implementation class name . The class will have
   to implement com.sun.enterprise.security.trust.TrustHandler
   interface.
-->

<!ELEMENT trust-handler EMPTY>
<!ATTLIST trust-handler class-name CDATA #REQUIRED>

<!--trusted-ip-address  identified the trusted host on the
network.eg : 129.169.223.2 -->

<!ELEMENT ip-address (#PCDATA)>

<!--trusted-domain-name   identifies the trusted host on the
network using domain names. eg: sun.com, cisco.com. All hosts
from sun.com domain are trusted.
-->
<!ELEMENT domain-name (#PCDATA)>
```

3.2.2.2 Trust Handler Interface

Interface enables developers to plugin custom validation of asserting entity (eg: host) and identity of the asserter (eg: X509Certificate used in the SSL connection).



```
public boolean  isTrusted(String asserterAddress, X509Certificate securityid, Principal pAssertedValues[0..*] )
public Principal[0..*]  mapIdentity( Principal assrtId[0..*] )
```

Illustration 3: Trust Handler

If the auth-method is P-Asserted and implementation of `TrustHandler` is configured then the container will invoke `isTrusted` method followed by `mapIdentity` method. `mapIdentity` method is only invoked if `isTrusted` returns true. Refer to Appendix A for more interface and javadoc of `TrustHandler`.

3.2.3 Implementation

3.2.3.1 Enabling Digest Authentication in WebContainer.

In the webcontainer HttpDigest authentication is enabled by making following changes.

a)Add

`DIGEST=org.apache.catalina.authenticator.DigestAuthenticator`

to

`appserv-webtier/src/java/org/apache/catalina/startup/Authenticators.properties` file.

b) `com.sun.enterprise.security.auth.LoginContextDriver.login(DigestCredentials digestCred)` will create JAAS LoginContext and invoke login method.

c)`DigestRealmBase` class is an extension to `IASRealm` class which has the default implementation of DigestValidation.

d) A new JASS Login module has been introduced to handle Digest based authentication. Any
login module that needs to support Digest authentication should extend `DigestLoginModule` and implement the abstract method `getGroups` shown below. `Illustration 4` shows `JDBCLoginModule` extending from `DigestLoginModule`.

```
      protected Enumeration getGroups(String username)
```

e)`$AS_HOME/domains/domain/config/login.conf` file is the place holder for `LoginModule` configuration.

Illustration 5 shows what is explained above using a sequence diagram

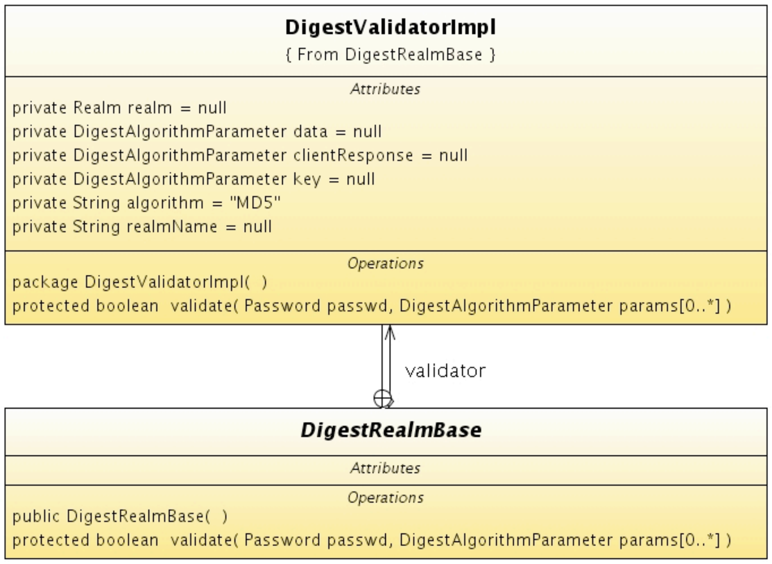
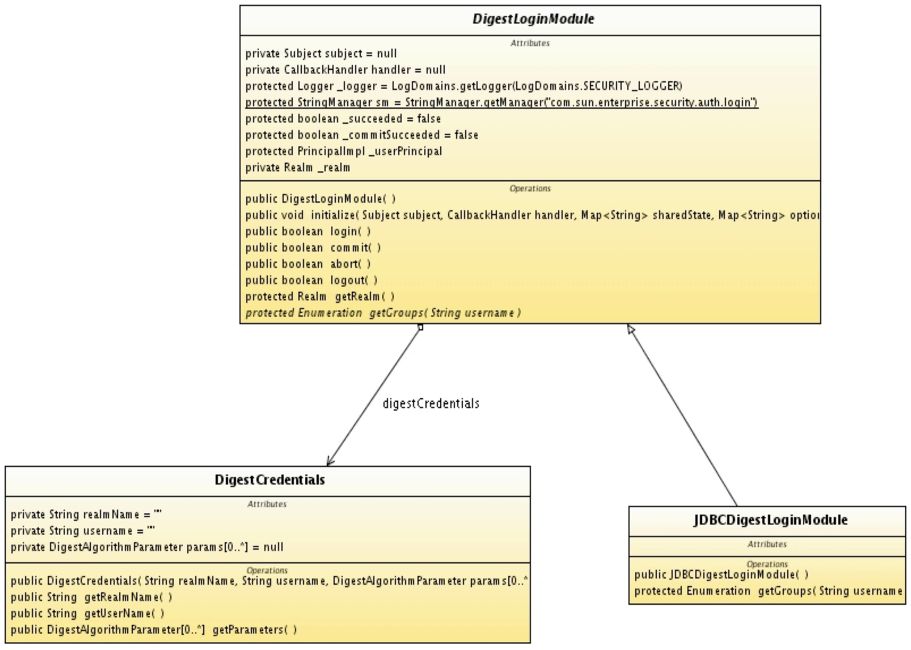

DigestValidatorImpl
{ From DigestRealmBase }
Attributes
private Realm realm = null
private DigestAlgorithmParameter data = null
private DigestAlgorithmParameter clientResponse = null
private DigestAlgorithmParameter key = null
private String algorithm = "MD5"
private String realmName = null
Operations
package DigestValidatorImpl( )
protected boolean validate(Password passwd, DigestAlgorithmParameter params[0..*])
validator
DigestRealmBase
Attributes
Operations
public DigestRealmBase( )
protected boolean validate(Password passwd, DigestAlgorithmParameter params[0..*])

Illustration 4: DigestRealmBase



```
DigestLoginModule
private Subject subject = null
private CallbackHandler handler = null
protected Logger _logger = LogDomains.getLogger(LogDomains.SECURITY_LOGGER)
protected StringManager sm = StringManager.getManager("com.sun.enterprise.security.auth.login")
protected boolean _succeeded = false
protected boolean _commitSucceeded = false
protected PrincipalImpl _userPrincipal
private Realm _realm

public DigestLoginModule( )
public void initialize( Subject subject, CallbackHandler handler, Map<String> sharedState, Map<String> optio
public boolean login( )
public boolean commit( )
public boolean abort( )
public boolean logout( )
protected Realm getRealm( )
protected Enumeration getGroups( String username )

DigestCredentials
private String realmName = ""
private String username = ""
private DigestAlgorithmParameter params[0..*] = null

public DigestCredentials( String realmName, String username, DigestAlgorithmParameter params[0..*
public String getRealmName( )
public String getUserName( )
public DigestAlgorithmParameter[0..*] getParameters( )

JDBCDigestLoginModule
public JDBCDigestLoginModule( )
protected Enumeration getGroups( String username
```

Illustration 5: DigestLoginModule

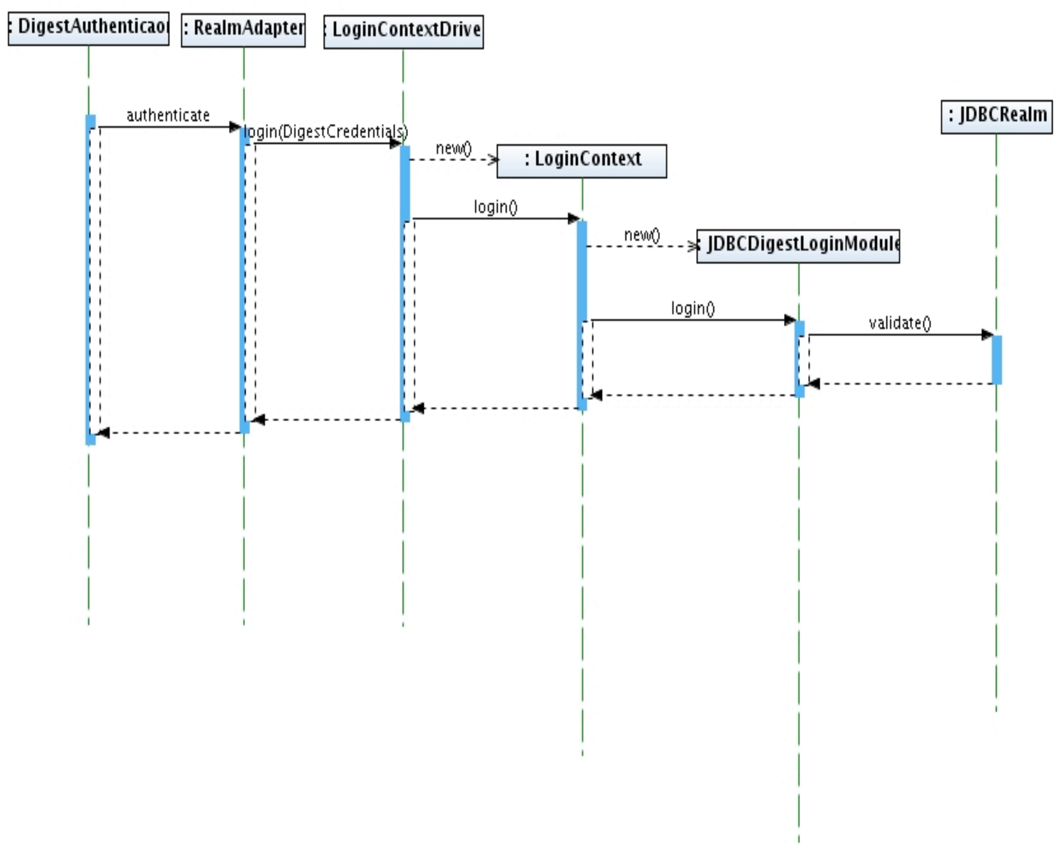


Illustration 6: Http authentication Sequence diagram

3.2.3.2 Enabling Digest Authentication in SIP Container.

SIP Container shares the LoginModules and Realms used by the Glassfish. Authentication code is plugged into the `invoke(SipServletRequestImpl request, SipApplicationRouterInfo info)` method of the `ServletDispacther.java.`

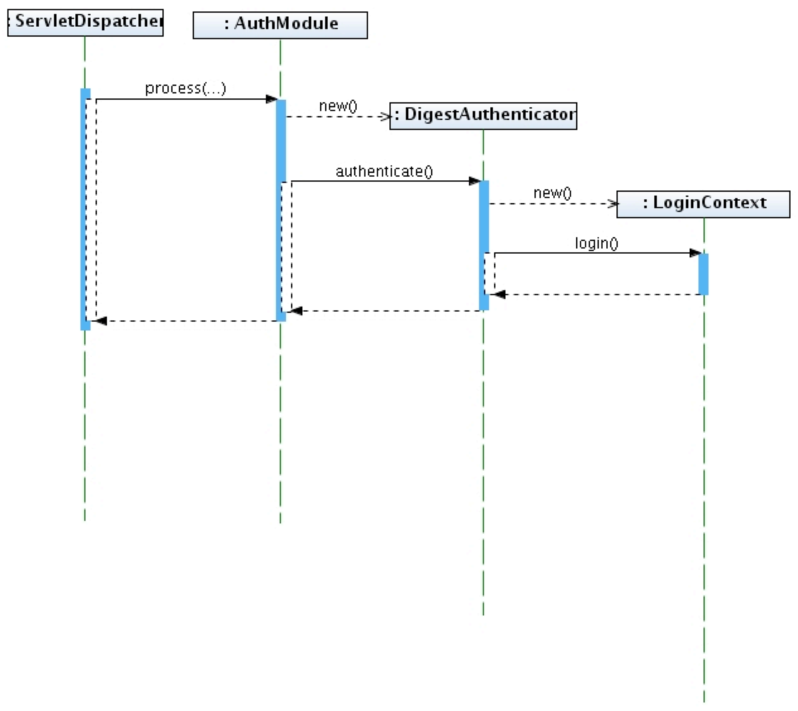


Illustration 7: SIP Authentication

3.3 Authorization

3.3.1 Policy Framework

JACC (Java Autthorization Contract for Containers) framework in Glassfish will be used by sailfin module. The policy context identifier will be computed as follows.

for standalone modules file.

```
/<application module identifier>/<application module identifier>
```

in case of enterprise applications

```
/< application module identifier>/<module archive-uri>
```

SipResourcePermission,SipUserDataPermission and SipRoleRefPermission classes are added to support SIP resource authorization.

➢ SipResourcePermission

Class for SIPServlet resource permissions
.
package : org.jvnet.glassfish.comms.security.authorize

Methods :

```
String getName();
String getActions();
boolean implies(Permission perm);
```

getName() method returns the SIP Servlet name
getActions method returns string representation of actions of this SiPResourcePermission.
implies() checks if the argument permission is implied by this SiPResourcePermission.
Returns true if

a)if permission argument is instanceof SiPResourcePermission
b)If name of the permission argument is equal to the name of this permission.
c)if actions of this permission object is empty.
d) if the actions of the permission argument is a subset of actions of this permission.

➢ SipUserDataPermission

class for SIPServlet UserDataPermissions. The behaviour of this class is similar to WebUserDataPermission class

package : org.jvnet.glassfish.comms.security.authorize

Methods:

```
String getName();
String getActions();
boolean implies(Permission perm);
```

getName() method returns the SIP Servlet name
getActions method returns string representation of actions of this SiPUserDataPermission.
implies() checks if the argument permission is implied by this SiPUserDataPermission.
Returns true if

a)if permission argument is instanceof SiPUserDataPermission
b)If name of the permission argument is equal to the name of this permission.
c)if the actions of the permission argument is a subset of actions of this permission.
d)transport type of this permission equals to NONE or is equal to the transportType of the permission argument.

- **SipRoleRefPermission**

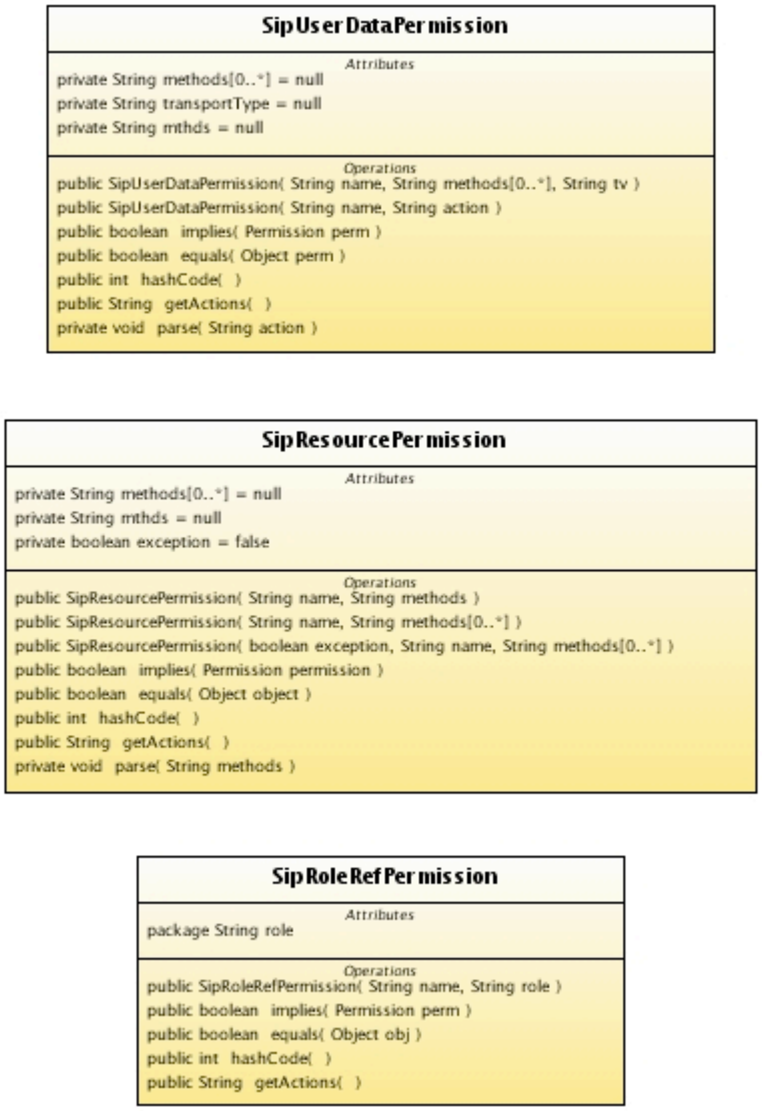
class represents mapping of roles to a SIP servlet. The behaviour of this class is similar to WebRoleRefPermission class

package : org.jvnet.glassfish.comms.security.authorize

Methods:

```
String getName();
String getActions();
boolean implies(Permission perm);
```

getName() method returns the SIP Servlet name
getActions method returns string representation of actions of this SipRoleRefPermission.
implies() checks if the argument permission is implied by this SipRoleRefPermission.
Returns true if
a)argument is instanceof SipResourcePermission class
b)name and role references of the permission argument match with name and role references of this object.



```
SipUserDataPermission
Attributes
private String methods[0..*] = null
private String transportType = null
private String mthds = null
Operations
public SipUserDataPermission( String name, String methods[0..*], String tv )
public SipUserDataPermission( String name, String action )
public boolean  implies( Permission perm )
public boolean  equals( Object perm )
public int  hashCode(  )
public String  getActions(  )
private void  parse( String action )
```

```
SipResourcePermission
Attributes
private String methods[0..*] = null
private String mthds = null
private boolean exception = false
Operations
public SipResourcePermission( String name, String methods )
public SipResourcePermission( String name, String methods[0..*] )
public SipResourcePermission( boolean exception, String name, String methods[0..*] )
public boolean  implies( Permission permission )
public boolean  equals( Object object )
public int  hashCode(  )
public String  getActions(  )
private void  parse( String methods )
```

```
SipRoleRefPermission
Attributes
package String role
Operations
public SipRoleRefPermission( String name, String role )
public boolean  implies( Permission perm )
public boolean  equals( Object obj )
public int  hashCode(  )
public String  getActions(  )
```

Illustration 8: Sip Permission classes

For every incoming message following steps are performed

- Resolve the servlets to be invoked.
- Check if UserDataConstraint has been configured and the connection has met the requirements.
- if UserDataConstraint has not been met then send a redirect response 3xx with corrected SIP Url.

- Check if resource-contraint has been configured for the method and the servlet to be invoked.
- if a resource constrain has been configured and if principal is not present in the Session return 401/407 response.
- if the request has Authentication information invoke authentication module.
- if the request has P-Asserted Identity and the authentication method configured is P-Asserted then perform trust validation.
- If principal is present then validate if the principal is authorized to invoke the servlet by calling AuthModule.hasResourcePermission(..);

3.4 Exported Interfaces.

Following interfaces are exposed to the users wanted to provide customised LoginModule and Realm implementations for Digest Authentication.

Class Name/Interface Name	Package
DigestAlgorithmParameter	com.sun.enterprise.security.auth.digest.api
DigestParameterGenerator	com.sun.enterprise.security.auth.digest.api
Key	com.sun.enterprise.security.auth.digest.api
DigestCredentials	com.sun.enterprise.security.auth.digest.api

4 Quality and Availability

<How do you handle availability concerns? Does this feature introduce any new failure modes? List testing and failure scenarios that quality team needs to worry about.>

Refer to Functional Requirements and Appendix A for more information.

5 Performance

<How do you want performance team to measure this sub-system? Any micro benchmarks necessary?Any goals? Anticipated scalability limits or goals?>

6 Management and Monitoring

<Describe how performance, management status, and diagnostic information is exposed. How does this feature handle dynamic configuration changes?>

6.1 Formal Interfaces

<How is this feature(s) configured by administrator? Does it introduce new commands or modify existing ones? Show syntax of expected administrative commands and response codes. What is the schema/syntax for new configuration in domain.xml? Show the DTD snippets later in this section. What are their default values? What are the validation rules? Think about the stability level for each of the above. Are you expecting that the proposed design will change?]

7 Packaging, Files, and Location

<Does this feature add new jar files or extend existing ones? Where are they located?>

8 Documentation Requirements

<List the required documentation to support this product feature.>

9 Open Issues

1)Identity plugin points to create policy file during deployment and application load time.

10 Appendix A:

10.1 Requirements summary from RFC 3325 and RFC 3261

Digest Authentication.

1. Container should have the ability to identify realms that support digest authentication.
2. Realms that support digest authentication should
   1. Nonce cache
   2. Nonce timeout interval
3. Support for qop=auth-int
4. Ability reject headers with Authentication mechanism other than Digest
   1. Eg: Request for Basic authentication should be rejected
5. Ability to use realm string as the protection domain.
6. Realm strings created should be globally unique(use hostname/domain name).
7. Realm name should be presentable to user i,e readable.

When UAC

1. Container should have the ability to deliver realm value received from the challenge to the client
2. Increment CSEQ header when resubmitting requests after receiving 401/407 response.
3. Ability to handle multiple **www-Authenticate** and **Proxy Authenticate** requests having same or different realms.

When UAS
Container

1. should not challenge CANCEL requests.
2. Should respond with 401 response and appropriate header fields shown below when no auth information is found.

When Proxy
Container

1. should respond with 407 error code when authentication is required and authentication headers are not found.
2. MUST not consume Authorization header filed if the realm value does not match the realm configured for proxy processing the message. This is usually the case when multiple proxies are present in the chain.
3. CSeq header must be incremented
4. Ability to aggregate multiple security challenges into a single response to the UA Client.
5. If authentication is configured do not proxy the request unless valid credentials are provided.

Challenge 1)

```
 WWW-Authenticate: Digest
              realm="biloxi.com",
              qop="auth,auth-int",
              nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
              opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Authorization response 2)

```
      Authorization: Digest username="bob",
              realm="biloxi.com",
              nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
              uri="sip:bob@biloxi.com",
              qop=auth,
              nc=00000001,
              cnonce="0a4f113b",
              response="6629fae49393a05397450978507c4ef1",
              opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Processing of P-Asserted Identity by Container for Proxy applications.

1. Accept **P-Asserted Identity** from trusted entities and treat the information as if the server authenticated the principal.
2. P-Asserted Identity should be a SIP/SIPS/TEL URI
3. Should be able to remove P-Asserted Identity information when received from non trusted nodes.
4. Should remove P-Asserted Identity information when forwarding request to non trusted nodes based on the privacy (id) element.
5. Container should have the ability to determine trusted and non trusted nodes. i,e ability to configure trust domains and associate policies to the domain.
6. Ability to reject messages with appropriate error codes when Identity cannot be validated/found in the requested realm.
7. Ability to process **P-Preferred Identity**
8. P-Preferred Identity to be sent only when sender is a UA and receiver is a trusted Proxy node.
9. Ability to process **"id"** (privacy) header element.
10. When forwarding requests without privacy header element the decision to forward P-Asserted Identity should be based on domain specific configuration,see #5.

Processing of P-Asserted Identity by Container for User agent applications.

1. Ability to use P-Asserted Identity information for authorization purposes.
2. Should be able to forward Identity information to Non Trusted nodes.
3. Should not use P-Asserted Identity information received from Non Trusted nodes.

Configuration Requirements:

1. Ability to configure P-Asserted Realms and Digest Realms
2. Ability to configure Trusted domains per application.
3. Ability to configure id attribute value.

10.2 Trust Handler Interface

```
import java.security.Principal;

import java.security.cert.X509Certificate;


/**

 *

 * Enables developers to provide custom implementation to enable sip containers

 * to determine if a network entity can be trusted and also enables formatting of

 * P-Asserted-Identity values.

 * eg: "Cullen Jennings" <sip:fluffy@cisco.com> value can be mapped/formatted to
 *      "CullenJ".
 */


public interface TrustHandler {


   /**
    * determines if the container can trust the network entity from which we received the message with P-
Asserted-Identity
    * header. This method also validates if the identity that was used to secure(eg: SSL) the message is trusted.
    *
    * @param pAssertedValues P-Asserted-Identity header values
    *@param trustedAs  value intermediate represents configuration information for incoming messages,
    *   if it has value destination then configuration under trusted-entity is applied to outgoing messages.
    * @param asserterAddress ipaddress/hostname of the network entity from which we received the SIP
message
    * with P-Asserted-Identity header. Inorder to accept/use the values in P-Asserted-Identity
    * header the network entity should be a trusted.
    * @param securityid is the asserting security identity, if a secure connection is used then this
    * would be the java.security.cert.X509Certificate, else null.
    * @return true if we trust the networtid and the securityid.
    */
   public boolean isTrusted(String asserterAddress,String trustedAs,X509Certificate securityid,Principal []
pAssertedValues);
   /**
    *
    * converts values in to a format understood by the container.
    *
    * @param assrtId P-Asserted-Identity values.
    * @return P-Asserted-Identity values in a format understood by the container.
    */
   public Principal [] mapIdentity(Principal [] assrtId);
}
```