| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SaslServer.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/security/sasl/SaslException.html)   [**NEXT CLASS**](http://docs.google.com/javax/security/sasl/SaslServerFactory.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/sasl/SaslServer.html)    [**NO FRAMES**](http://docs.google.com/SaslServer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **javax.security.sasl**

Interface SaslServer

public interface **SaslServer**

Performs SASL authentication as a server.

A server such an LDAP server gets an instance of this class in order to perform authentication defined by a specific SASL mechanism. Invoking methods on the SaslServer instance generates challenges according to the SASL mechanism implemented by the SaslServer. As the authentication proceeds, the instance encapsulates the state of a SASL server's authentication exchange.

Here's an example of how an LDAP server might use a SaslServer. It first gets an instance of a SaslServer for the SASL mechanism requested by the client:

SaslServer ss = Sasl.createSaslServer(mechanism,  
 "ldap", myFQDN, props, callbackHandler);

It can then proceed to use the server for authentication. For example, suppose the LDAP server received an LDAP BIND request containing the name of the SASL mechanism and an (optional) initial response. It then might use the server as follows:

while (!ss.isComplete()) {  
 try {  
 byte[] challenge = ss.evaluateResponse(response);  
 if (ss.isComplete()) {  
 status = ldap.sendBindResponse(mechanism, challenge, SUCCESS);  
 } else {  
 status = ldap.sendBindResponse(mechanism, challenge,   
 SASL\_BIND\_IN\_PROGRESS);  
 response = ldap.readBindRequest();  
 }  
 } catch (SaslException e) {  
 status = ldap.sendErrorResponse(e);  
 break;  
 }  
 }  
 if (ss.isComplete() && status == SUCCESS) {  
 String qop = (String) sc.getNegotiatedProperty(Sasl.QOP);  
 if (qop != null   
 && (qop.equalsIgnoreCase("auth-int")   
 || qop.equalsIgnoreCase("auth-conf"))) {  
  
 // Use SaslServer.wrap() and SaslServer.unwrap() for future  
 // communication with client  
 ldap.in = new SecureInputStream(ss, ldap.in);  
 ldap.out = new SecureOutputStream(ss, ldap.out);  
 }  
 }

**Since:** 1.5 **See Also:**[Sasl](http://docs.google.com/javax/security/sasl/Sasl.html), [SaslServerFactory](http://docs.google.com/javax/security/sasl/SaslServerFactory.html)

| **Method Summary** | |
| --- | --- |
| void | [**dispose**](http://docs.google.com/javax/security/sasl/SaslServer.html#dispose())()            Disposes of any system resources or security-sensitive information the SaslServer might be using. |
| byte[] | [**evaluateResponse**](http://docs.google.com/javax/security/sasl/SaslServer.html#evaluateResponse(byte%5B%5D))(byte[] response)            Evaluates the response data and generates a challenge. |
| [String](http://docs.google.com/java/lang/String.html) | [**getAuthorizationID**](http://docs.google.com/javax/security/sasl/SaslServer.html#getAuthorizationID())()            Reports the authorization ID in effect for the client of this session. |
| [String](http://docs.google.com/java/lang/String.html) | [**getMechanismName**](http://docs.google.com/javax/security/sasl/SaslServer.html#getMechanismName())()            Returns the IANA-registered mechanism name of this SASL server. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getNegotiatedProperty**](http://docs.google.com/javax/security/sasl/SaslServer.html#getNegotiatedProperty(java.lang.String))([String](http://docs.google.com/java/lang/String.html) propName)            Retrieves the negotiated property. |
| boolean | [**isComplete**](http://docs.google.com/javax/security/sasl/SaslServer.html#isComplete())()            Determines whether the authentication exchange has completed. |
| byte[] | [**unwrap**](http://docs.google.com/javax/security/sasl/SaslServer.html#unwrap(byte%5B%5D,%20int,%20int))(byte[] incoming, int offset, int len)            Unwraps a byte array received from the client. |
| byte[] | [**wrap**](http://docs.google.com/javax/security/sasl/SaslServer.html#wrap(byte%5B%5D,%20int,%20int))(byte[] outgoing, int offset, int len)            Wraps a byte array to be sent to the client. |

| **Method Detail** |
| --- |

### getMechanismName

[String](http://docs.google.com/java/lang/String.html) **getMechanismName**()

Returns the IANA-registered mechanism name of this SASL server. (e.g. "CRAM-MD5", "GSSAPI").

**Returns:**A non-null string representing the IANA-registered mechanism name.

### evaluateResponse

byte[] **evaluateResponse**(byte[] response)  
 throws [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html)

Evaluates the response data and generates a challenge. If a response is received from the client during the authentication process, this method is called to prepare an appropriate next challenge to submit to the client. The challenge is null if the authentication has succeeded and no more challenge data is to be sent to the client. It is non-null if the authentication must be continued by sending a challenge to the client, or if the authentication has succeeded but challenge data needs to be processed by the client. isComplete() should be called after each call to evaluateResponse(),to determine if any further response is needed from the client.

**Parameters:**response - The non-null (but possibly empty) response sent by the client. **Returns:**The possibly null challenge to send to the client. It is null if the authentication has succeeded and there is no more challenge data to be sent to the client. **Throws:** [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html) - If an error occurred while processing the response or generating a challenge.

### isComplete

boolean **isComplete**()

Determines whether the authentication exchange has completed. This method is typically called after each invocation of evaluateResponse() to determine whether the authentication has completed successfully or should be continued.

**Returns:**true if the authentication exchange has completed; false otherwise.

### getAuthorizationID

[String](http://docs.google.com/java/lang/String.html) **getAuthorizationID**()

Reports the authorization ID in effect for the client of this session. This method can only be called if isComplete() returns true.

**Returns:**The authorization ID of the client. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this authentication session has not completed

### unwrap

byte[] **unwrap**(byte[] incoming,  
 int offset,  
 int len)  
 throws [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html)

Unwraps a byte array received from the client. This method can be called only after the authentication exchange has completed (i.e., when isComplete() returns true) and only if the authentication exchange has negotiated integrity and/or privacy as the quality of protection; otherwise, an IllegalStateException is thrown.

incoming is the contents of the SASL buffer as defined in RFC 2222 without the leading four octet field that represents the length. offset and len specify the portion of incoming to use.

**Parameters:**incoming - A non-null byte array containing the encoded bytes from the client.offset - The starting position at incoming of the bytes to use.len - The number of bytes from incoming to use. **Returns:**A non-null byte array containing the decoded bytes. **Throws:** [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html) - if incoming cannot be successfully unwrapped. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the authentication exchange has not completed, or if the negotiated quality of protection has neither integrity nor privacy

### wrap

byte[] **wrap**(byte[] outgoing,  
 int offset,  
 int len)  
 throws [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html)

Wraps a byte array to be sent to the client. This method can be called only after the authentication exchange has completed (i.e., when isComplete() returns true) and only if the authentication exchange has negotiated integrity and/or privacy as the quality of protection; otherwise, a SaslException is thrown.

The result of this method will make up the contents of the SASL buffer as defined in RFC 2222 without the leading four octet field that represents the length. offset and len specify the portion of outgoing to use.

**Parameters:**outgoing - A non-null byte array containing the bytes to encode.offset - The starting position at outgoing of the bytes to use.len - The number of bytes from outgoing to use. **Returns:**A non-null byte array containing the encoded bytes. **Throws:** [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html) - if outgoing cannot be successfully wrapped. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the authentication exchange has not completed, or if the negotiated quality of protection has neither integrity nor privacy.

### getNegotiatedProperty

[Object](http://docs.google.com/java/lang/Object.html) **getNegotiatedProperty**([String](http://docs.google.com/java/lang/String.html) propName)

Retrieves the negotiated property. This method can be called only after the authentication exchange has completed (i.e., when isComplete() returns true); otherwise, an IllegalStateException is thrown.

**Parameters:**propName - the property **Returns:**The value of the negotiated property. If null, the property was not negotiated or is not applicable to this mechanism. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this authentication exchange has not completed

### dispose

void **dispose**()  
 throws [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html)

Disposes of any system resources or security-sensitive information the SaslServer might be using. Invoking this method invalidates the SaslServer instance. This method is idempotent.

**Throws:** [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html) - If a problem was encountered while disposing the resources.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SaslServer.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/security/sasl/SaslException.html)   [**NEXT CLASS**](http://docs.google.com/javax/security/sasl/SaslServerFactory.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/sasl/SaslServer.html)    [**NO FRAMES**](http://docs.google.com/SaslServer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).