| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SaslClient.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/Sasl.html)   [**NEXT CLASS**](http://docs.google.com/javax/security/sasl/SaslClientFactory.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/sasl/SaslClient.html)    [**NO FRAMES**](http://docs.google.com/SaslClient.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 SaslClient

public interface **SaslClient**

Performs SASL authentication as a client.

A protocol library such as one for LDAP gets an instance of this class in order to perform authentication defined by a specific SASL mechanism. Invoking methods on the SaslClient instance process challenges and create responses according to the SASL mechanism implemented by the SaslClient. As the authentication proceeds, the instance encapsulates the state of a SASL client's authentication exchange.

Here's an example of how an LDAP library might use a SaslClient. It first gets an instance of a SaslClient:

SaslClient sc = Sasl.createSaslClient(mechanisms,  
 authorizationId, protocol, serverName, props, callbackHandler);

It can then proceed to use the client for authentication. For example, an LDAP library might use the client as follows:

// Get initial response and send to server  
 byte[] response = (sc.hasInitialResponse() ? sc.evaluateChallenge(new byte[0]) :  
 null);  
 LdapResult res = ldap.sendBindRequest(dn, sc.getName(), response);  
 while (!sc.isComplete() &&   
 (res.status == SASL\_BIND\_IN\_PROGRESS || res.status == SUCCESS)) {  
 response = sc.evaluateChallenge(res.getBytes());  
 if (res.status == SUCCESS) {  
 // we're done; don't expect to send another BIND  
 if (response != null) {  
 throw new SaslException(  
 "Protocol error: attempting to send response after completion");  
 }  
 break;  
 }  
 res = ldap.sendBindRequest(dn, sc.getName(), response);  
 }  
 if (sc.isComplete() && res.status == SUCCESS) {  
 String qop = (String) sc.getNegotiatedProperty(Sasl.QOP);  
 if (qop != null   
 && (qop.equalsIgnoreCase("auth-int")   
 || qop.equalsIgnoreCase("auth-conf"))) {  
  
 // Use SaslClient.wrap() and SaslClient.unwrap() for future  
 // communication with server  
 ldap.in = new SecureInputStream(sc, ldap.in);  
 ldap.out = new SecureOutputStream(sc, ldap.out);  
 }  
 }

If the mechanism has an initial response, the library invokes evaluateChallenge() with an empty challenge and to get initial response. Protocols such as IMAP4, which do not include an initial response with their first authentication command to the server, initiates the authentication without first calling hasInitialResponse() or evaluateChallenge(). When the server responds to the command, it sends an initial challenge. For a SASL mechanism in which the client sends data first, the server should have issued a challenge with no data. This will then result in a call (on the client) to evaluateChallenge() with an empty challenge.

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

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

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

### getMechanismName

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

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

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

### hasInitialResponse

boolean **hasInitialResponse**()

Determines whether this mechanism has an optional initial response. If true, caller should call evaluateChallenge() with an empty array to get the initial response.

**Returns:**true if this mechanism has an initial response.

### evaluateChallenge

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

Evaluates the challenge data and generates a response. If a challenge is received from the server during the authentication process, this method is called to prepare an appropriate next response to submit to the server.

**Parameters:**challenge - The non-null challenge sent from the server. The challenge array may have zero length. **Returns:**The possibly null reponse to send to the server. It is null if the challenge accompanied a "SUCCESS" status and the challenge only contains data for the client to update its state and no response needs to be sent to the server. The response is a zero-length byte array if the client is to send a response with no data. **Throws:** [SaslException](http://docs.google.com/javax/security/sasl/SaslException.html) - If an error occurred while processing the challenge or generating a response.

### isComplete

boolean **isComplete**()

Determines whether the authentication exchange has completed. This method may be called at any time, but typically, it will not be called until the caller has received indication from the server (in a protocol-specific manner) that the exchange has completed.

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

### 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 server. 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 server.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 server. 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.

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 non-null property name. **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 SaslClient might be using. Invoking this method invalidates the SaslClient 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/SaslClient.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/Sasl.html)   [**NEXT CLASS**](http://docs.google.com/javax/security/sasl/SaslClientFactory.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/sasl/SaslClient.html)    [**NO FRAMES**](http://docs.google.com/SaslClient.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).