| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GSSContext.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/org/ietf/jgss/ChannelBinding.html)   [**NEXT CLASS**](http://docs.google.com/org/ietf/jgss/GSSCredential.html) | [**FRAMES**](http://docs.google.com/index.html?org/ietf/jgss/GSSContext.html)    [**NO FRAMES**](http://docs.google.com/GSSContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#4d34og8) |

## **org.ietf.jgss**

Interface GSSContext

public interface **GSSContext**

This interface encapsulates the GSS-API security context and provides the security services that are available over the context. Security contexts are established between peers using locally acquired credentials. Multiple contexts may exist simultaneously between a pair of peers, using the same or different set of credentials. GSS-API functions in a manner independent of the underlying transport protocol and depends on its calling application to transport the tokens that are generated by the security context between the peers.

If the caller instantiates the context using the default GSSManager instance, then the Kerberos v5 GSS-API mechanism is guaranteed to be available for context establishment. This mechanism is identified by the Oid "1.2.840.113554.1.2.2" and is defined in RFC 1964.

Before the context establishment phase is initiated, the context initiator may request specific characteristics desired of the established context. Not all underlying mechanisms support all characteristics that a caller might desire. After the context is established, the caller can check the actual characteristics and services offered by that context by means of various query methods. When using the Kerberos v5 GSS-API mechanism offered by the default GSSManager instance, all optional services will be available locally. They are mutual authentication, credential delegation, confidentiality and integrity protection, and per-message replay detection and sequencing. Note that in the GSS-API, message integrity is a prerequisite for message confidentiality.

The context establishment occurs in a loop where the initiator calls [initSecContext](http://docs.google.com/org/ietf/jgss/GSSContext.html#initSecContext(byte%5B%5D,%20int,%20int)) and the acceptor calls [acceptSecContext](http://docs.google.com/org/ietf/jgss/GSSContext.html#acceptSecContext(byte%5B%5D,%20int,%20int)) until the context is established. While in this loop the initSecContext and acceptSecContext methods produce tokens that the application sends over to the peer. The peer passes any such token as input to its acceptSecContext or initSecContext as the case may be.

During the context establishment phase, the [isProtReady](http://docs.google.com/org/ietf/jgss/GSSContext.html#isProtReady()) method may be called to determine if the context can be used for the per-message operations of [wrap](http://docs.google.com/org/ietf/jgss/GSSContext.html#wrap(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) and [getMIC](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMIC(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)). This allows applications to use per-message operations on contexts which aren't yet fully established.

After the context has been established or the isProtReady method returns true, the query routines can be invoked to determine the actual characteristics and services of the established context. The application can also start using the per-message methods of [wrap](http://docs.google.com/org/ietf/jgss/GSSContext.html#wrap(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) and [getMIC](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMIC(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) to obtain cryptographic operations on application supplied data.

When the context is no longer needed, the application should call [dispose](http://docs.google.com/org/ietf/jgss/GSSContext.html#dispose()) to release any system resources the context may be using.

A security context typically maintains sequencing and replay detection information about the tokens it processes. Therefore, the sequence in which any tokens are presented to this context for processing can be important. Also note that none of the methods in this interface are synchronized. Therefore, it is not advisable to share a GSSContext among several threads unless some application level synchronization is in place.

Finally, different mechanism providers might place different security restrictions on using GSS-API contexts. These will be documented by the mechanism provider. The application will need to ensure that it has the appropriate permissions if such checks are made in the mechanism layer.

The example code presented below demonstrates the usage of the GSSContext interface for the initiating peer. Different operations on the GSSContext object are presented, including: object instantiation, setting of desired flags, context establishment, query of actual context flags, per-message operations on application data, and finally context deletion.

// Create a context using default credentials   
 // and the implementation specific default mechanism  
 GSSManager manager ...  
 GSSName targetName ...  
 GSSContext context = manager.createContext(targetName, null, null,  
 GSSContext.INDEFINITE\_LIFETIME);  
   
 // set desired context options prior to context establishment  
 context.requestConf(true);  
 context.requestMutualAuth(true);  
 context.requestReplayDet(true);  
 context.requestSequenceDet(true);  
   
 // establish a context between peers  
  
 byte []inToken = new byte[0];  
  
 // Loop while there still is a token to be processed  
  
 while (!context.isEstablished()) {  
  
 byte[] outToken   
 = context.initSecContext(inToken, 0, inToken.length);  
   
 // send the output token if generated  
 if (outToken != null)  
 sendToken(outToken);  
   
 if (!context.isEstablished()) {  
 inToken = readToken();  
 }  
   
 // display context information  
 System.out.println("Remaining lifetime in seconds = "   
 + context.getLifetime());  
 System.out.println("Context mechanism = " + context.getMech());  
 System.out.println("Initiator = " + context.getSrcName());  
 System.out.println("Acceptor = " + context.getTargName());  
   
 if (context.getConfState())  
 System.out.println("Confidentiality (i.e., privacy) is available");  
   
 if (context.getIntegState())  
 System.out.println("Integrity is available");  
   
 // perform wrap on an application supplied message, appMsg,  
 // using QOP = 0, and requesting privacy service  
 byte [] appMsg ...  
   
 MessageProp mProp = new MessageProp(0, true);  
   
 byte []tok = context.wrap(appMsg, 0, appMsg.length, mProp);  
   
 sendToken(tok);  
   
 // release the local-end of the context  
 context.dispose();

**Since:** 1.4

| **Field Summary** | |
| --- | --- |
| static int | [**DEFAULT\_LIFETIME**](http://docs.google.com/org/ietf/jgss/GSSContext.html#DEFAULT_LIFETIME)            A lifetime constant representing the default context lifetime. |
| static int | [**INDEFINITE\_LIFETIME**](http://docs.google.com/org/ietf/jgss/GSSContext.html#INDEFINITE_LIFETIME)            A lifetime constant representing indefinite context lifetime. |

| **Method Summary** | |
| --- | --- |
| byte[] | [**acceptSecContext**](http://docs.google.com/org/ietf/jgss/GSSContext.html#acceptSecContext(byte%5B%5D,%20int,%20int))(byte[] inToken, int offset, int len)            Called by the context acceptor upon receiving a token from the peer. |
| void | [**acceptSecContext**](http://docs.google.com/org/ietf/jgss/GSSContext.html#acceptSecContext(java.io.InputStream,%20java.io.OutputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) inStream, [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream)            Called by the context acceptor to process a token from the peer using streams. |
| void | [**dispose**](http://docs.google.com/org/ietf/jgss/GSSContext.html#dispose())()            Releases any system resources and cryptographic information stored in the context object and invalidates the context. |
| byte[] | [**export**](http://docs.google.com/org/ietf/jgss/GSSContext.html#export())()            Exports this context so that another process may import it.. |
| boolean | [**getAnonymityState**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getAnonymityState())()            Determines if the context initiator is anonymously authenticated to the context acceptor. |
| boolean | [**getConfState**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getConfState())()            Determines if data confidentiality is available over the context. |
| boolean | [**getCredDelegState**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getCredDelegState())()            Determines if credential delegation is enabled on this context. |
| [GSSCredential](http://docs.google.com/org/ietf/jgss/GSSCredential.html) | [**getDelegCred**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getDelegCred())()            Obtains the credentials delegated by the context initiator to the context acceptor. |
| boolean | [**getIntegState**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getIntegState())()            Determines if data integrity is available over the context. |
| int | [**getLifetime**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getLifetime())()            Determines what the remaining lifetime for this context is. |
| [Oid](http://docs.google.com/org/ietf/jgss/Oid.html) | [**getMech**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMech())()            Determines what mechanism is being used for this context. |
| byte[] | [**getMIC**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMIC(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp))(byte[] inMsg, int offset, int len, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Returns a token containing a cryptographic Message Integrity Code (MIC) for the supplied message, for transfer to the peer application. |
| void | [**getMIC**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMIC(java.io.InputStream,%20java.io.OutputStream,%20org.ietf.jgss.MessageProp))([InputStream](http://docs.google.com/java/io/InputStream.html) inStream, [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Uses streams to produce a token containing a cryptographic MIC for the supplied message, for transfer to the peer application. |
| boolean | [**getMutualAuthState**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMutualAuthState())()            Determines if mutual authentication is enabled on this context. |
| boolean | [**getReplayDetState**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getReplayDetState())()            Determines if replay detection is enabled for the per-message security services from this context. |
| boolean | [**getSequenceDetState**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getSequenceDetState())()            Determines if sequence checking is enabled for the per-message security services from this context. |
| [GSSName](http://docs.google.com/org/ietf/jgss/GSSName.html) | [**getSrcName**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getSrcName())()            Returns the name of the context initiator. |
| [GSSName](http://docs.google.com/org/ietf/jgss/GSSName.html) | [**getTargName**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getTargName())()            Returns the name of the context acceptor. |
| int | [**getWrapSizeLimit**](http://docs.google.com/org/ietf/jgss/GSSContext.html#getWrapSizeLimit(int,%20boolean,%20int))(int qop, boolean confReq, int maxTokenSize)            Used to determine limits on the size of the message that can be passed to wrap. |
| byte[] | [**initSecContext**](http://docs.google.com/org/ietf/jgss/GSSContext.html#initSecContext(byte%5B%5D,%20int,%20int))(byte[] inputBuf, int offset, int len)            Called by the context initiator to start the context creation phase and process any tokens generated by the peer's acceptSecContext method. |
| int | [**initSecContext**](http://docs.google.com/org/ietf/jgss/GSSContext.html#initSecContext(java.io.InputStream,%20java.io.OutputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) inStream, [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream)            Called by the context initiator to start the context creation phase and process any tokens generated by the peer's acceptSecContext method using streams. |
| boolean | [**isEstablished**](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished())()            Used during context establishment to determine the state of the context. |
| boolean | [**isInitiator**](http://docs.google.com/org/ietf/jgss/GSSContext.html#isInitiator())()            Determines if this is the context initiator. |
| boolean | [**isProtReady**](http://docs.google.com/org/ietf/jgss/GSSContext.html#isProtReady())()            Determines if the context is ready for per message operations to be used over it. |
| boolean | [**isTransferable**](http://docs.google.com/org/ietf/jgss/GSSContext.html#isTransferable())()            Determines if the context is transferable to other processes through the use of the [export](http://docs.google.com/org/ietf/jgss/GSSContext.html#export()) method. |
| void | [**requestAnonymity**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestAnonymity(boolean))(boolean state)            Requests that the initiator's identity not be disclosed to the acceptor. |
| void | [**requestConf**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestConf(boolean))(boolean state)            Requests that data confidentiality be enabled for the wrap method. |
| void | [**requestCredDeleg**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestCredDeleg(boolean))(boolean state)            Requests that the initiator's credentials be delegated to the acceptor during context establishment. |
| void | [**requestInteg**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestInteg(boolean))(boolean state)            Requests that data integrity be enabled for the wrap and getMICmethods. |
| void | [**requestLifetime**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestLifetime(int))(int lifetime)            Requests a lifetime in seconds for the context. |
| void | [**requestMutualAuth**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestMutualAuth(boolean))(boolean state)            Requests that mutual authentication be done during context establishment. |
| void | [**requestReplayDet**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestReplayDet(boolean))(boolean state)            Requests that replay detection be enabled for the per-message security services after context establishemnt. |
| void | [**requestSequenceDet**](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestSequenceDet(boolean))(boolean state)            Requests that sequence checking be enabled for the per-message security services after context establishemnt. |
| void | [**setChannelBinding**](http://docs.google.com/org/ietf/jgss/GSSContext.html#setChannelBinding(org.ietf.jgss.ChannelBinding))([ChannelBinding](http://docs.google.com/org/ietf/jgss/ChannelBinding.html) cb)            Sets the channel bindings to be used during context establishment. |
| byte[] | [**unwrap**](http://docs.google.com/org/ietf/jgss/GSSContext.html#unwrap(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp))(byte[] inBuf, int offset, int len, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Used to process tokens generated by the wrap method on the other side of the context. |
| void | [**unwrap**](http://docs.google.com/org/ietf/jgss/GSSContext.html#unwrap(java.io.InputStream,%20java.io.OutputStream,%20org.ietf.jgss.MessageProp))([InputStream](http://docs.google.com/java/io/InputStream.html) inStream, [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Uses streams to process tokens generated by the wrap method on the other side of the context. |
| void | [**verifyMIC**](http://docs.google.com/org/ietf/jgss/GSSContext.html#verifyMIC(byte%5B%5D,%20int,%20int,%20byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp))(byte[] inToken, int tokOffset, int tokLen, byte[] inMsg, int msgOffset, int msgLen, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Verifies the cryptographic MIC, contained in the token parameter, over the supplied message. |
| void | [**verifyMIC**](http://docs.google.com/org/ietf/jgss/GSSContext.html#verifyMIC(java.io.InputStream,%20java.io.InputStream,%20org.ietf.jgss.MessageProp))([InputStream](http://docs.google.com/java/io/InputStream.html) tokStream, [InputStream](http://docs.google.com/java/io/InputStream.html) msgStream, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Uses streams to verify the cryptographic MIC, contained in the token parameter, over the supplied message. |
| byte[] | [**wrap**](http://docs.google.com/org/ietf/jgss/GSSContext.html#wrap(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp))(byte[] inBuf, int offset, int len, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Applies per-message security services over the established security context. |
| void | [**wrap**](http://docs.google.com/org/ietf/jgss/GSSContext.html#wrap(java.io.InputStream,%20java.io.OutputStream,%20org.ietf.jgss.MessageProp))([InputStream](http://docs.google.com/java/io/InputStream.html) inStream, [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream, [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)            Applies per-message security services over the established security context using streams. |

| **Field Detail** |
| --- |

### DEFAULT\_LIFETIME

static final int **DEFAULT\_LIFETIME**

A lifetime constant representing the default context lifetime. This value is set to 0.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.ietf.jgss.GSSContext.DEFAULT_LIFETIME)

### INDEFINITE\_LIFETIME

static final int **INDEFINITE\_LIFETIME**

A lifetime constant representing indefinite context lifetime. This value must is set to the maximum integer value in Java - [Integer.MAX\_VALUE](http://docs.google.com/java/lang/Integer.html#MAX_VALUE).

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.ietf.jgss.GSSContext.INDEFINITE_LIFETIME)

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

### initSecContext

byte[] **initSecContext**(byte[] inputBuf,  
 int offset,  
 int len)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Called by the context initiator to start the context creation phase and process any tokens generated by the peer's acceptSecContext method. This method may return an output token which the application will need to send to the peer for processing by its acceptSecContext method. The application can call [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) to determine if the context establishment phase is complete on this side of the context. A return value of false from isEstablished indicates that more tokens are expected to be supplied to initSecContext. Upon completion of the context establishment, the available context options may be queried through the get methods.

Note that it is possible that the initSecContext method return a token for the peer, and isEstablished return true also. This indicates that the token needs to be sent to the peer, but the local end of the context is now fully established.

Some mechanism providers might require that the caller be granted permission to initiate a security context. A failed permission check might cause a [SecurityException](http://docs.google.com/java/lang/SecurityException.html) to be thrown from this method.

**Parameters:**inputBuf - token generated by the peer. This parameter is ignored on the first call since no token has been received from the peer.offset - the offset within the inputBuf where the token begins.len - the length of the token. **Returns:**a byte[] containing the token to be sent to the peer. null indicates that no token is generated. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN), [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC), [GSSException.NO\_CRED](http://docs.google.com/org/ietf/jgss/GSSException.html#NO_CRED), [GSSException.CREDENTIALS\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CREDENTIALS_EXPIRED), [GSSException.BAD\_BINDINGS](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_BINDINGS), [GSSException.OLD\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#OLD_TOKEN), [GSSException.DUPLICATE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DUPLICATE_TOKEN), [GSSException.BAD\_NAMETYPE](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_NAMETYPE), [GSSException.BAD\_MECH](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MECH), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### initSecContext

int **initSecContext**([InputStream](http://docs.google.com/java/io/InputStream.html) inStream,  
 [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Called by the context initiator to start the context creation phase and process any tokens generated by the peer's acceptSecContext method using streams. This method may write an output token to the OutpuStream, which the application will need to send to the peer for processing by its acceptSecContext call. Typically, the application would ensure this by calling the [flush](http://docs.google.com/java/io/OutputStream.html#flush()) method on an OutputStream that encapsulates the connection between the two peers. The application can determine if a token is written to the OutputStream from the return value of this method. A return value of 0 indicates that no token was written. The application can call [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) to determine if the context establishment phase is complete on this side of the context. A return value of false from isEstablished indicates that more tokens are expected to be supplied to initSecContext. Upon completion of the context establishment, the available context options may be queried through the get methods.

Note that it is possible that the initSecContext method return a token for the peer, and isEstablished return true also. This indicates that the token needs to be sent to the peer, but the local end of the context is now fully established.

The GSS-API authentication tokens contain a definitive start and end. This method will attempt to read one of these tokens per invocation, and may block on the stream if only part of the token is available. In all other respects this method is equivalent to the byte array based [initSecContext](http://docs.google.com/org/ietf/jgss/GSSContext.html#initSecContext(byte%5B%5D,%20int,%20int)).

Some mechanism providers might require that the caller be granted permission to initiate a security context. A failed permission check might cause a [SecurityException](http://docs.google.com/java/lang/SecurityException.html) to be thrown from this method.

The following example code demonstrates how this method might be used:

InputStream is ...  
 OutputStream os ...  
 GSSContext context ...  
  
 // Loop while there is still a token to be processed  
  
 while (!context.isEstablished()) {  
  
 context.initSecContext(is, os);  
  
 // send output token if generated  
 os.flush();  
 }

**Parameters:**inStream - an InputStream that contains the token generated by the peer. This parameter is ignored on the first call since no token has been or will be received from the peer at that point.outStream - an OutputStream where the output token will be written. During the final stage of context establishment, there may be no bytes written. **Returns:**the number of bytes written to the OutputStream as part of the token to be sent to the peer. A value of 0 indicates that no token needs to be sent. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN), [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC), [GSSException.NO\_CRED](http://docs.google.com/org/ietf/jgss/GSSException.html#NO_CRED), [GSSException.CREDENTIALS\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CREDENTIALS_EXPIRED), [GSSException.BAD\_BINDINGS](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_BINDINGS), [GSSException.OLD\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#OLD_TOKEN), [GSSException.DUPLICATE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DUPLICATE_TOKEN), [GSSException.BAD\_NAMETYPE](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_NAMETYPE), [GSSException.BAD\_MECH](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MECH), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### acceptSecContext

byte[] **acceptSecContext**(byte[] inToken,  
 int offset,  
 int len)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Called by the context acceptor upon receiving a token from the peer. This method may return an output token which the application will need to send to the peer for further processing by its initSecContext call.

The application can call [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) to determine if the context establishment phase is complete for this peer. A return value of false from isEstablished indicates that more tokens are expected to be supplied to this method. Upon completion of the context establishment, the available context options may be queried through the get methods.

Note that it is possible that acceptSecContext return a token for the peer, and isEstablished return true also. This indicates that the token needs to be sent to the peer, but the local end of the context is now fully established.

Some mechanism providers might require that the caller be granted permission to accept a security context. A failed permission check might cause a [SecurityException](http://docs.google.com/java/lang/SecurityException.html) to be thrown from this method.

The following example code demonstrates how this method might be used:

byte[] inToken;  
 byte[] outToken;  
 GSSContext context ...  
  
 // Loop while there is still a token to be processed  
  
 while (!context.isEstablished()) {  
 inToken = readToken();  
 outToken = context.acceptSecContext(inToken, 0,  
 inToken.length);  
 // send output token if generated  
 if (outToken != null)  
 sendToken(outToken);  
 }

**Parameters:**inToken - token generated by the peer.offset - the offset within the inToken where the token begins.len - the length of the token. **Returns:**a byte[] containing the token to be sent to the peer. null indicates that no token is generated. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN), [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC), [GSSException.NO\_CRED](http://docs.google.com/org/ietf/jgss/GSSException.html#NO_CRED), [GSSException.CREDENTIALS\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CREDENTIALS_EXPIRED), [GSSException.BAD\_BINDINGS](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_BINDINGS), [GSSException.OLD\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#OLD_TOKEN), [GSSException.DUPLICATE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DUPLICATE_TOKEN), [GSSException.BAD\_MECH](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MECH), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### acceptSecContext

void **acceptSecContext**([InputStream](http://docs.google.com/java/io/InputStream.html) inStream,  
 [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Called by the context acceptor to process a token from the peer using streams. It may write an output token to the OutputStream, which the application will need to send to the peer for processing by its initSecContext method. Typically, the application would ensure this by calling the [flush](http://docs.google.com/java/io/OutputStream.html#flush()) method on an OutputStream that encapsulates the connection between the two peers. The application can call [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) to determine if the context establishment phase is complete on this side of the context. A return value of false from isEstablished indicates that more tokens are expected to be supplied to acceptSecContext. Upon completion of the context establishment, the available context options may be queried through the get methods.

Note that it is possible that acceptSecContext return a token for the peer, and isEstablished return true also. This indicates that the token needs to be sent to the peer, but the local end of the context is now fully established.

The GSS-API authentication tokens contain a definitive start and end. This method will attempt to read one of these tokens per invocation, and may block on the stream if only part of the token is available. In all other respects this method is equivalent to the byte array based [acceptSecContext](http://docs.google.com/org/ietf/jgss/GSSContext.html#acceptSecContext(byte%5B%5D,%20int,%20int)).

Some mechanism providers might require that the caller be granted permission to accept a security context. A failed permission check might cause a [SecurityException](http://docs.google.com/java/lang/SecurityException.html) to be thrown from this method.

The following example code demonstrates how this method might be used:

InputStream is ...  
 OutputStream os ...  
 GSSContext context ...  
  
 // Loop while there is still a token to be processed  
  
 while (!context.isEstablished()) {  
  
 context.acceptSecContext(is, os);  
  
 // send output token if generated  
 os.flush();  
 }

**Parameters:**inStream - an InputStream that contains the token generated by the peer.outStream - an OutputStream where the output token will be written. During the final stage of context establishment, there may be no bytes written. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN), [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC), [GSSException.NO\_CRED](http://docs.google.com/org/ietf/jgss/GSSException.html#NO_CRED), [GSSException.CREDENTIALS\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CREDENTIALS_EXPIRED), [GSSException.BAD\_BINDINGS](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_BINDINGS), [GSSException.OLD\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#OLD_TOKEN), [GSSException.DUPLICATE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DUPLICATE_TOKEN), [GSSException.BAD\_MECH](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MECH), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### isEstablished

boolean **isEstablished**()

Used during context establishment to determine the state of the context.

**Returns:**true if this is a fully established context on the caller's side and no more tokens are needed from the peer.

### dispose

void **dispose**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Releases any system resources and cryptographic information stored in the context object and invalidates the context.

**Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### getWrapSizeLimit

int **getWrapSizeLimit**(int qop,  
 boolean confReq,  
 int maxTokenSize)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Used to determine limits on the size of the message that can be passed to wrap. Returns the maximum message size that, if presented to the wrap method with the same confReq and qop parameters, will result in an output token containing no more than maxTokenSize bytes.

This call is intended for use by applications that communicate over protocols that impose a maximum message size. It enables the application to fragment messages prior to applying protection.

GSS-API implementations are recommended but not required to detect invalid QOP values when getWrapSizeLimit is called. This routine guarantees only a maximum message size, not the availability of specific QOP values for message protection.

**Parameters:**qop - the level of protection wrap will be asked to provide.confReq - true if wrap will be asked to provide privacy, false otherwise.maxTokenSize - the desired maximum size of the token emitted by wrap. **Returns:**the maximum size of the input token for the given output token size **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.BAD\_QOP](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_QOP), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### wrap

byte[] **wrap**(byte[] inBuf,  
 int offset,  
 int len,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Applies per-message security services over the established security context. The method will return a token with the application supplied data and a cryptographic MIC over it. The data may be encrypted if confidentiality (privacy) was requested.

The MessageProp object is instantiated by the application and used to specify a QOP value which selects cryptographic algorithms, and a privacy service to optionally encrypt the message. The underlying mechanism that is used in the call may not be able to provide the privacy service. It sets the actual privacy service that it does provide in this MessageProp object which the caller should then query upon return. If the mechanism is not able to provide the requested QOP, it throws a GSSException with the BAD\_QOP code.

Since some application-level protocols may wish to use tokens emitted by wrap to provide "secure framing", implementations should support the wrapping of zero-length messages.

The application will be responsible for sending the token to the peer.

**Parameters:**inBuf - application data to be protected.offset - the offset within the inBuf where the data begins.len - the length of the datamsgProp - instance of MessageProp that is used by the application to set the desired QOP and privacy state. Set the desired QOP to 0 to request the default QOP. Upon return from this method, this object will contain the the actual privacy state that was applied to the message by the underlying mechanism. **Returns:**a byte[] containing the token to be sent to the peer. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.BAD\_QOP](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_QOP), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### wrap

void **wrap**([InputStream](http://docs.google.com/java/io/InputStream.html) inStream,  
 [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Applies per-message security services over the established security context using streams. The method will return a token with the application supplied data and a cryptographic MIC over it. The data may be encrypted if confidentiality (privacy) was requested. This method is equivalent to the byte array based [wrap](http://docs.google.com/org/ietf/jgss/GSSContext.html#wrap(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) method.

The application will be responsible for sending the token to the peer. Typically, the application would ensure this by calling the [flush](http://docs.google.com/java/io/OutputStream.html#flush()) method on an OutputStream that encapsulates the connection between the two peers.

The MessageProp object is instantiated by the application and used to specify a QOP value which selects cryptographic algorithms, and a privacy service to optionally encrypt the message. The underlying mechanism that is used in the call may not be able to provide the privacy service. It sets the actual privacy service that it does provide in this MessageProp object which the caller should then query upon return. If the mechanism is not able to provide the requested QOP, it throws a GSSException with the BAD\_QOP code.

Since some application-level protocols may wish to use tokens emitted by wrap to provide "secure framing", implementations should support the wrapping of zero-length messages.

**Parameters:**inStream - an InputStream containing the application data to be protected. All of the data that is available in inStream is used.outStream - an OutputStream to write the protected message to.msgProp - instance of MessageProp that is used by the application to set the desired QOP and privacy state. Set the desired QOP to 0 to request the default QOP. Upon return from this method, this object will contain the the actual privacy state that was applied to the message by the underlying mechanism. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.BAD\_QOP](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_QOP), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### unwrap

byte[] **unwrap**(byte[] inBuf,  
 int offset,  
 int len,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Used to process tokens generated by the wrap method on the other side of the context. The method will return the message supplied by the peer application to its wrap call, while at the same time verifying the embedded MIC for that message.

The MessageProp object is instantiated by the application and is used by the underlying mechanism to return information to the caller such as the QOP, whether confidentiality was applied to the message, and other supplementary message state information.

Since some application-level protocols may wish to use tokens emitted by wrap to provide "secure framing", implementations should support the wrapping and unwrapping of zero-length messages.

**Parameters:**inBuf - a byte array containing the wrap token received from peer.offset - the offset where the token begins.len - the length of the tokenmsgProp - upon return from the method, this object will contain the applied QOP, the privacy state of the message, and supplementary information stating if the token was a duplicate, old, out of sequence or arriving after a gap. **Returns:**a byte[] containing the message unwrapped from the input token. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN), [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC), [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### unwrap

void **unwrap**([InputStream](http://docs.google.com/java/io/InputStream.html) inStream,  
 [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Uses streams to process tokens generated by the wrap method on the other side of the context. The method will return the message supplied by the peer application to its wrap call, while at the same time verifying the embedded MIC for that message.

The MessageProp object is instantiated by the application and is used by the underlying mechanism to return information to the caller such as the QOP, whether confidentiality was applied to the message, and other supplementary message state information.

Since some application-level protocols may wish to use tokens emitted by wrap to provide "secure framing", implementations should support the wrapping and unwrapping of zero-length messages.

The format of the input token that this method reads is defined in the specification for the underlying mechanism that will be used. This method will attempt to read one of these tokens per invocation. If the mechanism token contains a definitive start and end this method may block on the InputStream if only part of the token is available. If the start and end of the token are not definitive then the method will attempt to treat all available bytes as part of the token.

Other than the possible blocking behaviour described above, this method is equivalent to the byte array based [unwrap](http://docs.google.com/org/ietf/jgss/GSSContext.html#unwrap(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) method.

**Parameters:**inStream - an InputStream that contains the wrap token generated by the peer.outStream - an OutputStream to write the application message to.msgProp - upon return from the method, this object will contain the applied QOP, the privacy state of the message, and supplementary information stating if the token was a duplicate, old, out of sequence or arriving after a gap. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN), [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC), [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### getMIC

byte[] **getMIC**(byte[] inMsg,  
 int offset,  
 int len,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Returns a token containing a cryptographic Message Integrity Code (MIC) for the supplied message, for transfer to the peer application. Unlike wrap, which encapsulates the user message in the returned token, only the message MIC is returned in the output token.

Note that privacy can only be applied through the wrap call.

Since some application-level protocols may wish to use tokens emitted by getMIC to provide "secure framing", implementations should support derivation of MICs from zero-length messages.

**Parameters:**inMsg - the message to generate the MIC over.offset - offset within the inMsg where the message begins.len - the length of the messagemsgProp - an instance of MessageProp that is used by the application to set the desired QOP. Set the desired QOP to 0 in msgProp to request the default QOP. Alternatively pass in null for msgProp to request the default QOP. **Returns:**a byte[] containing the token to be sent to the peer. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.BAD\_QOP](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_QOP), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### getMIC

void **getMIC**([InputStream](http://docs.google.com/java/io/InputStream.html) inStream,  
 [OutputStream](http://docs.google.com/java/io/OutputStream.html) outStream,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Uses streams to produce a token containing a cryptographic MIC for the supplied message, for transfer to the peer application. Unlike wrap, which encapsulates the user message in the returned token, only the message MIC is produced in the output token. This method is equivalent to the byte array based [getMIC](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMIC(byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) method. Note that privacy can only be applied through the wrap call.

Since some application-level protocols may wish to use tokens emitted by getMIC to provide "secure framing", implementations should support derivation of MICs from zero-length messages.

**Parameters:**inStream - an InputStream containing the message to generate the MIC over. All of the data that is available in inStream is used.outStream - an OutputStream to write the output token to.msgProp - an instance of MessageProp that is used by the application to set the desired QOP. Set the desired QOP to 0 in msgProp to request the default QOP. Alternatively pass in null for msgProp to request the default QOP. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.BAD\_QOP](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_QOP), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### verifyMIC

void **verifyMIC**(byte[] inToken,  
 int tokOffset,  
 int tokLen,  
 byte[] inMsg,  
 int msgOffset,  
 int msgLen,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Verifies the cryptographic MIC, contained in the token parameter, over the supplied message.

The MessageProp object is instantiated by the application and is used by the underlying mechanism to return information to the caller such as the QOP indicating the strength of protection that was applied to the message and other supplementary message state information.

Since some application-level protocols may wish to use tokens emitted by getMIC to provide "secure framing", implementations should support the calculation and verification of MICs over zero-length messages.

**Parameters:**inToken - the token generated by peer's getMIC method.tokOffset - the offset within the inToken where the token begins.tokLen - the length of the token.inMsg - the application message to verify the cryptographic MIC over.msgOffset - the offset in inMsg where the message begins.msgLen - the length of the message.msgProp - upon return from the method, this object will contain the applied QOP and supplementary information stating if the token was a duplicate, old, out of sequence or arriving after a gap. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN) [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC) [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED) [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### verifyMIC

void **verifyMIC**([InputStream](http://docs.google.com/java/io/InputStream.html) tokStream,  
 [InputStream](http://docs.google.com/java/io/InputStream.html) msgStream,  
 [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html) msgProp)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Uses streams to verify the cryptographic MIC, contained in the token parameter, over the supplied message. This method is equivalent to the byte array based [verifyMIC](http://docs.google.com/org/ietf/jgss/GSSContext.html#verifyMIC(byte%5B%5D,%20int,%20int,%20byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) method. The MessageProp object is instantiated by the application and is used by the underlying mechanism to return information to the caller such as the QOP indicating the strength of protection that was applied to the message and other supplementary message state information.

Since some application-level protocols may wish to use tokens emitted by getMIC to provide "secure framing", implementations should support the calculation and verification of MICs over zero-length messages.

The format of the input token that this method reads is defined in the specification for the underlying mechanism that will be used. This method will attempt to read one of these tokens per invocation. If the mechanism token contains a definitive start and end this method may block on the InputStream if only part of the token is available. If the start and end of the token are not definitive then the method will attempt to treat all available bytes as part of the token.

Other than the possible blocking behaviour described above, this method is equivalent to the byte array based [verifyMIC](http://docs.google.com/org/ietf/jgss/GSSContext.html#verifyMIC(byte%5B%5D,%20int,%20int,%20byte%5B%5D,%20int,%20int,%20org.ietf.jgss.MessageProp)) method.

**Parameters:**tokStream - an InputStream containing the token generated by the peer's getMIC method.msgStream - an InputStream containing the application message to verify the cryptographic MIC over. All of the data that is available in msgStream is used.msgProp - upon return from the method, this object will contain the applied QOP and supplementary information stating if the token was a duplicate, old, out of sequence or arriving after a gap. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.DEFECTIVE\_TOKEN](http://docs.google.com/org/ietf/jgss/GSSException.html#DEFECTIVE_TOKEN) [GSSException.BAD\_MIC](http://docs.google.com/org/ietf/jgss/GSSException.html#BAD_MIC) [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED) [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### export

byte[] **export**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Exports this context so that another process may import it.. Provided to support the sharing of work between multiple processes. This routine will typically be used by the context-acceptor, in an application where a single process receives incoming connection requests and accepts security contexts over them, then passes the established context to one or more other processes for message exchange.

This method deactivates the security context and creates an interprocess token which, when passed to [GSSManager.createContext](http://docs.google.com/org/ietf/jgss/GSSManager.html#createContext(byte%5B%5D)) in another process, will re-activate the context in the second process. Only a single instantiation of a given context may be active at any one time; a subsequent attempt by a context exporter to access the exported security context will fail.

The implementation may constrain the set of processes by which the interprocess token may be imported, either as a function of local security policy, or as a result of implementation decisions. For example, some implementations may constrain contexts to be passed only between processes that run under the same account, or which are part of the same process group.

The interprocess token may contain security-sensitive information (for example cryptographic keys). While mechanisms are encouraged to either avoid placing such sensitive information within interprocess tokens, or to encrypt the token before returning it to the application, in a typical GSS-API implementation this may not be possible. Thus the application must take care to protect the interprocess token, and ensure that any process to which the token is transferred is trustworthy.

Implementations are not required to support the inter-process transfer of security contexts. Calling the [isTransferable](http://docs.google.com/org/ietf/jgss/GSSContext.html#isTransferable()) method will indicate if the context object is transferable.

Calling this method on a context that is not exportable will result in this exception being thrown with the error code [GSSException.UNAVAILABLE](http://docs.google.com/org/ietf/jgss/GSSException.html#UNAVAILABLE).

**Returns:**a byte[] containing the exported context **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.UNAVAILABLE](http://docs.google.com/org/ietf/jgss/GSSException.html#UNAVAILABLE), [GSSException.CONTEXT\_EXPIRED](http://docs.google.com/org/ietf/jgss/GSSException.html#CONTEXT_EXPIRED), [GSSException.NO\_CONTEXT](http://docs.google.com/org/ietf/jgss/GSSException.html#NO_CONTEXT), [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[GSSManager.createContext(byte[])](http://docs.google.com/org/ietf/jgss/GSSManager.html#createContext(byte%5B%5D))

### requestMutualAuth

void **requestMutualAuth**(boolean state)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests that mutual authentication be done during context establishment. This request can only be made on the context initiator's side and it has to be done prior to the first call to initSecContext.

Not all mechanisms support mutual authentication and some mechanisms might require mutual authentication even if the application doesn't. Therefore, the application should check to see if the request was honored with the [getMutualAuthState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMutualAuthState()) method.

**Parameters:**state - a boolean value indicating whether mutual authentication shouls be used or not. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getMutualAuthState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getMutualAuthState())

### requestReplayDet

void **requestReplayDet**(boolean state)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests that replay detection be enabled for the per-message security services after context establishemnt. This request can only be made on the context initiator's side and it has to be done prior to the first call to initSecContext. During context establishment replay detection is not an option and is a function of the underlying mechanism's capabilities.

Not all mechanisms support replay detection and some mechanisms might require replay detection even if the application doesn't. Therefore, the application should check to see if the request was honored with the [getReplayDetState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getReplayDetState()) method. If replay detection is enabled then the [MessageProp.isDuplicateToken](http://docs.google.com/org/ietf/jgss/MessageProp.html#isDuplicateToken()) and [MessageProp.isOldToken](http://docs.google.com/org/ietf/jgss/MessageProp.html#isOldToken()) methods will return valid results for the MessageProp object that is passed in to the unwrap method or the verifyMIC method.

**Parameters:**state - a boolean value indicating whether replay detection should be enabled over the established context or not. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getReplayDetState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getReplayDetState())

### requestSequenceDet

void **requestSequenceDet**(boolean state)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests that sequence checking be enabled for the per-message security services after context establishemnt. This request can only be made on the context initiator's side and it has to be done prior to the first call to initSecContext. During context establishment sequence checking is not an option and is a function of the underlying mechanism's capabilities.

Not all mechanisms support sequence checking and some mechanisms might require sequence checking even if the application doesn't. Therefore, the application should check to see if the request was honored with the [getSequenceDetState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getSequenceDetState()) method. If sequence checking is enabled then the [MessageProp.isDuplicateToken](http://docs.google.com/org/ietf/jgss/MessageProp.html#isDuplicateToken()), [MessageProp.isOldToken](http://docs.google.com/org/ietf/jgss/MessageProp.html#isOldToken()), [MessageProp.isUnseqToken](http://docs.google.com/org/ietf/jgss/MessageProp.html#isUnseqToken()), and [MessageProp.isGapToken](http://docs.google.com/org/ietf/jgss/MessageProp.html#isGapToken()) methods will return valid results for the MessageProp object that is passed in to the unwrap method or the verifyMIC method.

**Parameters:**state - a boolean value indicating whether sequence checking should be enabled over the established context or not. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getSequenceDetState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getSequenceDetState())

### requestCredDeleg

void **requestCredDeleg**(boolean state)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests that the initiator's credentials be delegated to the acceptor during context establishment. This request can only be made on the context initiator's side and it has to be done prior to the first call to initSecContext. Not all mechanisms support credential delegation. Therefore, an application that desires delegation should check to see if the request was honored with the [getCredDelegState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getCredDelegState()) method. If the application indicates that delegation must not be used, then the mechanism will honor the request and delegation will not occur. This is an exception to the general rule that a mechanism may enable a service even if it is not requested.

**Parameters:**state - a boolean value indicating whether the credentials should be delegated or not. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getCredDelegState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getCredDelegState())

### requestAnonymity

void **requestAnonymity**(boolean state)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests that the initiator's identity not be disclosed to the acceptor. This request can only be made on the context initiator's side and it has to be done prior to the first call to initSecContext. Not all mechanisms support anonymity for the initiator. Therefore, the application should check to see if the request was honored with the [getAnonymityState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getAnonymityState()) method.

**Parameters:**state - a boolean value indicating if the initiator should be authenticated to the acceptor as an anonymous principal. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getAnonymityState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getAnonymityState())

### requestConf

void **requestConf**(boolean state)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests that data confidentiality be enabled for the wrap method. This request can only be made on the context initiator's side and it has to be done prior to the first call to initSecContext. Not all mechanisms support confidentiality and other mechanisms might enable it even if the application doesn't request it. The application may check to see if the request was honored with the [getConfState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getConfState()) method. If confidentiality is enabled, only then will the mechanism honor a request for privacy in the [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html#MessageProp(int,%20boolean)) object that is passed in to the wrap method.

Enabling confidentiality will also automatically enable integrity.

**Parameters:**state - a boolean value indicating whether confidentiality should be enabled or not. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getConfState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getConfState()), [getIntegState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getIntegState()), [requestInteg(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestInteg(boolean)), [MessageProp](http://docs.google.com/org/ietf/jgss/MessageProp.html)

### requestInteg

void **requestInteg**(boolean state)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests that data integrity be enabled for the wrap and getMICmethods. This request can only be made on the context initiator's side and it has to be done prior to the first call to initSecContext. Not all mechanisms support integrity and other mechanisms might enable it even if the application doesn't request it. The application may check to see if the request was honored with the [getIntegState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getIntegState()) method.

Disabling integrity will also automatically disable confidentiality.

**Parameters:**state - a boolean value indicating whether integrity should be enabled or not. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getIntegState()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getIntegState())

### requestLifetime

void **requestLifetime**(int lifetime)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Requests a lifetime in seconds for the context. This method can only be called on the context initiator's side and it has to be done prior to the first call to initSecContext.

The actual lifetime of the context will depend on the capabilites of the underlying mechanism and the application should call the [getLifetime](http://docs.google.com/org/ietf/jgss/GSSContext.html#getLifetime()) method to determine this.

**Parameters:**lifetime - the desired context lifetime in seconds. Use INDEFINITE\_LIFETIME to request an indefinite lifetime and DEFAULT\_LIFETIME to request a default lifetime. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[getLifetime()](http://docs.google.com/org/ietf/jgss/GSSContext.html#getLifetime())

### setChannelBinding

void **setChannelBinding**([ChannelBinding](http://docs.google.com/org/ietf/jgss/ChannelBinding.html) cb)  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Sets the channel bindings to be used during context establishment. This method can be called on both the context initiator's and the context acceptor's side, but it must be called before context establishment begins. This means that an initiator must call it before the first call to initSecContext and the acceptor must call it before the first call to acceptSecContext.

**Parameters:**cb - the channel bindings to use. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### getCredDelegState

boolean **getCredDelegState**()

Determines if credential delegation is enabled on this context. It can be called by both the context initiator and the context acceptor. For a definitive answer this method must be called only after context establishment is complete. Note that if an initiator requests that delegation not be allowed the [requestCredDeleg](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestCredDeleg(boolean)) method will honor that request and this method will return false on the initiator's side from that point onwards.

**Returns:**true if delegation is enabled, false otherwise.**See Also:**[requestCredDeleg(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestCredDeleg(boolean))

### getMutualAuthState

boolean **getMutualAuthState**()

Determines if mutual authentication is enabled on this context. It can be called by both the context initiator and the context acceptor. For a definitive answer this method must be called only after context establishment is complete. An initiator that requests mutual authentication can call this method after context completion and dispose the context if its request was not honored.

**Returns:**true if mutual authentication is enabled, false otherwise.**See Also:**[requestMutualAuth(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestMutualAuth(boolean))

### getReplayDetState

boolean **getReplayDetState**()

Determines if replay detection is enabled for the per-message security services from this context. It can be called by both the context initiator and the context acceptor. For a definitive answer this method must be called only after context establishment is complete. An initiator that requests replay detection can call this method after context completion and dispose the context if its request was not honored.

**Returns:**true if replay detection is enabled, false otherwise.**See Also:**[requestReplayDet(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestReplayDet(boolean))

### getSequenceDetState

boolean **getSequenceDetState**()

Determines if sequence checking is enabled for the per-message security services from this context. It can be called by both the context initiator and the context acceptor. For a definitive answer this method must be called only after context establishment is complete. An initiator that requests sequence checking can call this method after context completion and dispose the context if its request was not honored.

**Returns:**true if sequence checking is enabled, false otherwise.**See Also:**[requestSequenceDet(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestSequenceDet(boolean))

### getAnonymityState

boolean **getAnonymityState**()

Determines if the context initiator is anonymously authenticated to the context acceptor. It can be called by both the context initiator and the context acceptor, and at any time. **On the initiator side, a call to this method determines if the identity of the initiator has been disclosed in any of the context establishment tokens that might have been generated thus far by initSecContext. An initiator that absolutely must be authenticated anonymously should call this method after each call to initSecContext to determine if the generated token should be sent to the peer or the context aborted.** On the acceptor side, a call to this method determines if any of the tokens processed by acceptSecContext thus far have divulged the identity of the initiator.

**Returns:**true if the context initiator is still anonymous, false otherwise.**See Also:**[requestAnonymity(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestAnonymity(boolean))

### isTransferable

boolean **isTransferable**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Determines if the context is transferable to other processes through the use of the [export](http://docs.google.com/org/ietf/jgss/GSSContext.html#export()) method. This call is only valid on fully established contexts.

**Returns:**true if this context can be exported, false otherwise. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### isProtReady

boolean **isProtReady**()

Determines if the context is ready for per message operations to be used over it. Some mechanisms may allow the usage of the per-message operations before the context is fully established.

**Returns:**true if methods like wrap, unwrap, getMIC, and verifyMIC can be used with this context at the current stage of context establishment, false otherwise.

### getConfState

boolean **getConfState**()

Determines if data confidentiality is available over the context. This method can be called by both the context initiator and the context acceptor, but only after one of [isProtReady](http://docs.google.com/org/ietf/jgss/GSSContext.html#isProtReady()) or [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) return true. If this method returns true, so will [getIntegState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getIntegState())

**Returns:**true if confidentiality services are available, false otherwise.**See Also:**[requestConf(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestConf(boolean))

### getIntegState

boolean **getIntegState**()

Determines if data integrity is available over the context. This method can be called by both the context initiator and the context acceptor, but only after one of [isProtReady](http://docs.google.com/org/ietf/jgss/GSSContext.html#isProtReady()) or [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) return true. This method will always return true if [getConfState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getConfState()) returns true.

**Returns:**true if integrity services are available, false otherwise.**See Also:**[requestInteg(boolean)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestInteg(boolean))

### getLifetime

int **getLifetime**()

Determines what the remaining lifetime for this context is. It can be called by both the context initiator and the context acceptor, but for a definitive answer it should be called only after [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) returns true.

**Returns:**the remaining lifetime in seconds**See Also:**[requestLifetime(int)](http://docs.google.com/org/ietf/jgss/GSSContext.html#requestLifetime(int))

### getSrcName

[GSSName](http://docs.google.com/org/ietf/jgss/GSSName.html) **getSrcName**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Returns the name of the context initiator. This call is valid only after one of [isProtReady](http://docs.google.com/org/ietf/jgss/GSSContext.html#isProtReady()) or [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) return true.

**Returns:**a GSSName that is an MN containing the name of the context initiator. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)**See Also:**[GSSName](http://docs.google.com/org/ietf/jgss/GSSName.html)

### getTargName

[GSSName](http://docs.google.com/org/ietf/jgss/GSSName.html) **getTargName**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Returns the name of the context acceptor. This call is valid only after one of [isProtReady](http://docs.google.com/org/ietf/jgss/GSSContext.html#isProtReady()) or [isEstablished](http://docs.google.com/org/ietf/jgss/GSSContext.html#isEstablished()) return true.

**Returns:**a GSSName that is an MN containing the name of the context acceptor. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### getMech

[Oid](http://docs.google.com/org/ietf/jgss/Oid.html) **getMech**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Determines what mechanism is being used for this context. This method may be called before the context is fully established, but the mechanism returned may change on successive calls in the negotiated mechanism case.

**Returns:**the Oid of the mechanism being used **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### getDelegCred

[GSSCredential](http://docs.google.com/org/ietf/jgss/GSSCredential.html) **getDelegCred**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Obtains the credentials delegated by the context initiator to the context acceptor. It should be called only on the context acceptor's side, and once the context is fully established. The caller can use the method [getCredDelegState](http://docs.google.com/org/ietf/jgss/GSSContext.html#getCredDelegState()) to determine if there are any delegated credentials.

**Returns:**a GSSCredential containing the initiator's delegated credentials, or null is no credentials were delegated. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

### isInitiator

boolean **isInitiator**()  
 throws [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html)

Determines if this is the context initiator. This can be called on both the context initiator's and context acceptor's side.

**Returns:**true if this is the context initiator, false if it is the context acceptor. **Throws:** [GSSException](http://docs.google.com/org/ietf/jgss/GSSException.html) - containing the following major error codes: [GSSException.FAILURE](http://docs.google.com/org/ietf/jgss/GSSException.html#FAILURE)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GSSContext.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/org/ietf/jgss/ChannelBinding.html)   [**NEXT CLASS**](http://docs.google.com/org/ietf/jgss/GSSCredential.html) | [**FRAMES**](http://docs.google.com/index.html?org/ietf/jgss/GSSContext.html)    [**NO FRAMES**](http://docs.google.com/GSSContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#4d34og8) |

[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).