



## CHAPTER 22

# *org.ietf.jgss*

### Package **org.ietf.jgss**

**Java 1.4**

This package is a Java binding of the Generic Security Services (GSS) API defined by the Internet Engineering Task Force (IETF). GSS is an API that allows two network peers to mutually authenticate each other, and to ensure the integrity and confidentiality of the data they exchange. GSS is a “generic” API in that it is designed to be general enough to work on top of any “security mechanism” that supports authentication, encryption, and data integrity protection. One such mechanism is Kerberos, and Sun’s JGSS implementation currently supports only the Kerberos mechanism. Sun’s implementation is integrated with the JAAS API defined by the `javax.security.auth` package and its sub-packages. Clients and servers that want to use JGSS must first perform a JAAS login to ensure that the required Kerberos credentials are available to the JGSS implementation.

Begin studying this package with `GSSManager`, which is the central factory class for the API. Understanding and using this package requires substantial knowledge of the GSS API and of Kerberos. Because these topics are well beyond the scope of this reference, no attempt is made to fully document the classes in this package. For more information, please read the relevant RFC documents published by the IETF:

- The GSS API is defined in a language-independent way by RFC 2743: <http://www.ietf.org/rfc/rfc2743.txt>.
- The Java binding of the GSS API is defined by RFC 2853: <http://www.ietf.org/rfc/rfc2853.txt>.
- The Kerberos Network Authentication Service is defined by RFC 1510: <http://www.ietf.org/rfc/rfc1510.txt>.
- A Kerberos security mechanism for GSS is defined by RFC 1964: <http://www.ietf.org/rfc/rfc1964.txt>.

In addition to these primary sources, Sun includes JGSS documentation and tutorials in its documentation bundle for Java 1.4. You can find them at: <http://java.sun.com/j2se/1.4/docs/guide/security/jgss/tutorials/index.html>.

*Interfaces:*

```
public interface GSSContext;
public interface GSSCredential extends Cloneable;
public interface GSSName;
```

*Classes:*

```
public class ChannelBinding;
public abstract class GSSManager;
public class MessageProp;
public class Oid;
```

*Exception:*

```
public class GSSException extends Exception;
```

**ChannelBinding****Java 1.4****org.ietf.jgss**

A ChannelBinding object is used to provide additional constraints, such as the Internet addresses of the client and server, upon the establishment of a security context.

```
public class ChannelBinding {
    // Public Constructors
    public ChannelBinding(byte[ ] appData);
    public ChannelBinding(java.net.InetAddress initAddr, java.net.InetAddress acceptAddr, byte[ ] appData);
    // Public Instance Methods
    public java.net.InetAddress getAcceptorAddress();
    public byte[ ] getApplicationData();
    public java.net.InetAddress getInitiatorAddress();
    // Public Methods Overriding Object
    public boolean equals(Object obj);
    public int hashCode();
}
```

*Passed To:* GSSContext.setChannelBinding()

**GSSContext****Java 1.4****org.ietf.jgss**

This interface represents the “security context” within which authentication, encryption, and data integrity protection take place. A GSSContext object is obtained from a GSSManager factory. Before a GSSContext can be used to exchange data with a network peer, the security context must first be “established” (this typically involves mutual authentication) in a “context establishment loop” in which the client repeatedly calls initSecContext() and the server repeatedly calls acceptSecContext(), until the isEstablished() method returns true. Once a context has been established, you can use wrap() to apply security services (such as encryption) to an array of bytes or to a stream and can use unwrap() to reverse this process. You can also use getMIC() and verifyMIC() to generate and verify a “message integrity code” to protect the integrity of data that is transmitted in the clear.

```
public interface GSSContext {
    // Public Constants
    public static final int DEFAULT_LIFETIME; =0
    public static final int INDEFINITE_LIFETIME; =2147483647
```

## GSSContext

```
// Property Accessor Methods (by property name)
public abstract boolean getAnonymityState();
public abstract boolean getConfState();
public abstract boolean getCredDelegState();
public abstract GSSCredential getDelegCred() throws GSSEException;
public abstract boolean isEstablished();
public abstract boolean isInitiator() throws GSSEException;
public abstract boolean getIntegState();
public abstract int getLifetime();
public abstract Oid getMech() throws GSSEException;
public abstract boolean getMutualAuthState();
public abstract boolean isProtReady();
public abstract boolean getReplayDetState();
public abstract boolean getSequenceDetState();
public abstract GSSName getSrcName() throws GSSEException;
public abstract GSSName getTargName() throws GSSEException;
public abstract boolean isTransferable() throws GSSEException;

// Public Instance Methods
public abstract void acceptSecContext(java.io.InputStream inStream, java.io.OutputStream outStream)
    throws GSSEException;
public abstract byte[] acceptSecContext(byte[] inToken, int offset, int len) throws GSSEException;
public abstract void dispose() throws GSSEException;
public abstract byte[] export() throws GSSEException;
public abstract void getMIC(java.io.InputStream inStream, java.io.OutputStream outStream,
    MessageProp msgProp) throws GSSEException;
public abstract byte[] getMIC(byte[] inMsg, int offset, int len, MessageProp msgProp) throws GSSEException;
public abstract int getWrapSizeLimit(int qop, boolean confReq, int maxTokenSize) throws GSSEException;
public abstract int initSecContext(java.io.InputStream inStream, java.io.OutputStream outStream)
    throws GSSEException;
public abstract byte[] initSecContext(byte[] inputBuf, int offset, int len) throws GSSEException;
public abstract void requestAnonymity(boolean state) throws GSSEException;
public abstract void requestConf(boolean state) throws GSSEException;
public abstract void requestCredDeleg(boolean state) throws GSSEException;
public abstract void requestInteg(boolean state) throws GSSEException;
public abstract void requestLifetime(int lifetime) throws GSSEException;
public abstract void requestMutualAuth(boolean state) throws GSSEException;
public abstract void requestReplayDet(boolean state) throws GSSEException;
public abstract void requestSequenceDet(boolean state) throws GSSEException;
public abstract void setChannelBinding(ChannelBinding cb) throws GSSEException;
public abstract void unwrap(java.io.InputStream inStream, java.io.OutputStream outStream,
    MessageProp msgProp) throws GSSEException;
public abstract byte[] unwrap(byte[] inBuf, int offset, int len, MessageProp msgProp) throws GSSEException;
public abstract void verifyMIC(java.io.InputStream tokStream, java.io.InputStream msgStream,
    MessageProp msgProp) throws GSSEException;
public abstract void verifyMIC(byte[] inToken, int tokOffset, int tokLen, byte[] inMsg, int msgOffset, int msgLen,
    MessageProp msgProp) throws GSSEException;
public abstract void wrap(java.io.InputStream inStream, java.io.OutputStream outStream, MessageProp msgProp)
    throws GSSEException;
public abstract byte[] wrap(byte[] inBuf, int offset, int len, MessageProp msgProp) throws GSSEException;
}
```

*Returned By:* GSSManager.createContext()

**GSSCredential**

Java 1.4

org.ietf.jgss

cloneable

This interface defines a high-level API to a generic security credential, such as a Kerberos ticket or a Kerberos key.



```

public interface GSSCredential extends Cloneable {
// Public Constants
    public static final int ACCEPT_ONLY;                =2
    public static final int DEFAULT_LIFETIME;          =0
    public static final int INDEFINITE_LIFETIME;        =2147483647
    public static final int INITIATE_AND_ACCEPT;        =0
    public static final int INITIATE_ONLY;              =1
// Public Instance Methods
    public abstract void add(GSSName name, int initLifetime, int acceptLifetime, Oid mech, int usage)
        throws GSSException;
    public abstract void dispose() throws GSSException;
    public abstract boolean equals(Object another);
    public abstract Oid[] getMechs() throws GSSException;
    public abstract GSSName getName() throws GSSException;
    public abstract GSSName getName(Oid mech) throws GSSException;
    public abstract int getRemainingAcceptLifetime(Oid mech) throws GSSException;
    public abstract int getRemainingInitLifetime(Oid mech) throws GSSException;
    public abstract int getRemainingLifetime() throws GSSException;
    public abstract int getUsage() throws GSSException;
    public abstract int getUsage(Oid mech) throws GSSException;
    public abstract int hashCode();
}
  
```

*Passed To:* GSSManager.createContext()

*Returned By:* GSSContext.getDelegCred(), GSSManager.createCredential()

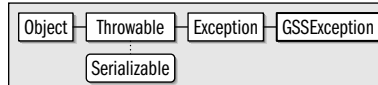
**GSSException**

Java 1.4

org.ietf.jgss

serializable checked

This exception class is used to signal all exceptions in the org.ietf.jgss package. `getMajor()` returns a GSS error code, which should be one of the integer constants defined by this class. `getMajorString()` returns an error message that corresponds to that error code. `getMinor()` and `getMinorString()` return an error code and error string that are specific to the underlying security mechanism.



```

public class GSSException extends Exception {
// Public Constructors
    public GSSException(int majorCode);
    public GSSException(int majorCode, int minorCode, String minorString);
// Public Constants
    public static final int BAD_BINDINGS;                =1
    public static final int BAD_MECH;                    =2
    public static final int BAD_MIC;                      =6
    public static final int BAD_NAME;                     =3
    public static final int BAD_NAME_TYPE;                =4
}
  
```

## *GSSException*

```
public static final int BAD_QOP; =14
public static final int BAD_STATUS; =5
public static final int CONTEXT_EXPIRED; =7
public static final int CREDENTIALS_EXPIRED; =8
public static final int DEFECTIVE_CREDENTIAL; =9
public static final int DEFECTIVE_TOKEN; =10
public static final int DUPLICATE_ELEMENT; =17
public static final int DUPLICATE_TOKEN; =19
public static final int FAILURE; =11
public static final int GAP_TOKEN; =22
public static final int NAME_NOT_MN; =18
public static final int NO_CONTEXT; =12
public static final int NO_CRED; =13
public static final int OLD_TOKEN; =20
public static final int UNAUTHORIZED; =15
public static final int UNAVAILABLE; =16
public static final int UNSEQ_TOKEN; =21
// Public Instance Methods
public int getMajor();
public String getMajorString();
public int getMinor();
public String getMinorString();
public void setMinor(int minorCode, String message);
// Public Methods Overriding Throwable
public String getMessage();
public String toString();
}
```

*Thrown By:* Too many methods to list.

## **GSSManager**

**Java 1.4**

**org.ietf.jgss**

This is the central factory class of the `org.ietf.jgss` package. Obtain the default `GSSManager` with the static `getInstance()` method or by instantiating some vendor-supplied subclass. Call `getMechs()` to query the security mechanisms supported by the manager. (The default `GSSManager` supplied by Sun supports only the Kerberos mechanism.) Call `createContext()` to create a `GSSContext`. Call `createCredential()` to create a `GSSCredential`. Call `createName()` to create a `GSSName`.

```
public abstract class GSSManager {
// Public Constructors
public GSSManager();
// Public Class Methods
public static GSSManager getInstance();
// Public Instance Methods
public abstract void addProviderAtEnd(java.security.Provider p, Oid mech) throws GSSException;
public abstract void addProviderAtFront(java.security.Provider p, Oid mech) throws GSSException;
public abstract GSSContext createContext(GSSCredential myCred) throws GSSException;
public abstract GSSContext createContext(byte[] interProcessToken) throws GSSException;
public abstract GSSContext createContext(GSSName peer, Oid mech, GSSCredential myCred, int lifetime)
throws GSSException;
public abstract GSSCredential createCredential(int usage) throws GSSException;
public abstract GSSCredential createCredential(GSSName name, int lifetime, Oid mech, int usage)
throws GSSException;
public abstract GSSCredential createCredential(GSSName name, int lifetime, Oid[] mechs, int usage)
throws GSSException;
}
```

```

public abstract GSSName createName(String nameStr, Oid nameType) throws GSSEException;
public abstract GSSName createName(byte[] name, Oid nameType) throws GSSEException;
public abstract GSSName createName(String nameStr, Oid nameType, Oid mech) throws GSSEException;
public abstract GSSName createName(byte[] name, Oid nameType, Oid mech) throws GSSEException;
public abstract Oid[] getMechs();
public abstract Oid[] getMechsForName(Oid nameType);
public abstract Oid[] getNamesForMech(Oid mech) throws GSSEException;
}

```

*Returned By:* GSSManager.getInstance()

## GSSName

Java 1.4

org.ietf.jgss

A GSSName represents an entity, such as a user or a network server, that participates in a GSS session. The constants defined by this class represent the allowed set of “name types.”

```

public interface GSSName {
// Public Constants
    public static final Oid NT_ANONYMOUS;
    public static final Oid NT_EXPORT_NAME;
    public static final Oid NT_HOSTBASED_SERVICE;
    public static final Oid NT_MACHINE_UID_NAME;
    public static final Oid NT_STRING_UID_NAME;
    public static final Oid NT_USER_NAME;
// Public Instance Methods
    public abstract GSSName canonicalize(Oid mech) throws GSSEException;
    public abstract boolean equals(Object another);
    public abstract boolean equals(GSSName another) throws GSSEException;
    public abstract byte[] export() throws GSSEException;
    public abstract Oid getStringNameType() throws GSSEException;
    public abstract int hashCode();
    public abstract boolean isAnonymous();
    public abstract boolean isMN();
    public abstract String toString();
}

```

*Passed To:* GSSCredential.add(), GSSManager.{createContext(), createCredential()}, GSSName.equals()

*Returned By:* GSSContext.{getSrcName(), getTargName()}, GSSCredential.getName(), GSSManager.createName(), GSSName.canonicalize()

## MessageProp

Java 1.4

org.ietf.jgss

This utility class is used with the wrap(), unwrap(), getMIC() and verifyMIC() methods of GSSContext to pass and return properties related to the message or message integrity code.

```

public class MessageProp {
// Public Constructors
    public MessageProp(boolean privState);
    public MessageProp(int qop, boolean privState);
// Property Accessor Methods (by property name)
    public boolean isDuplicateToken();
    public boolean isGapToken();
}

```

## *MessageProp*

```
public int getMinorStatus();
public String getMinorString();
public boolean isOldToken();
public boolean getPrivacy();
public void setPrivacy(boolean privState);
public int getQOP();
public void setQOP(int qop);
public boolean isUnseqToken();
// Public Instance Methods
public void setSupplementaryStates(boolean duplicate, boolean old, boolean unseq, boolean gap,
int minorStatus, String minorString);
}
```

*Passed To:* GSSContext.{getMIC(), unwrap(), verifyMIC(), wrap()}

## **Oid**

**Java 1.4**

**org.ietf.jgss**

An Oid is a universal “object identifier” based on a hierarchical numbering system. The Oid class is used in this package to uniquely identify security mechanisms and name types. The Oid for the Kerberos security mechanism is 1.2.840.113554.1.2.2.

```
public class Oid {
// Public Constructors
public Oid(String strOid) throws GSSException;
public Oid(java.io.InputStream derOid) throws GSSException;
public Oid(byte[] data) throws GSSException;
// Public Instance Methods
public boolean containedIn(Oid[] oids);
public byte[] getDER() throws GSSException;
// Public Methods Overriding Object
public boolean equals(Object other);
public int hashCode();
public String toString();
}
```

*Passed To:* Too many methods to list.

*Returned By:* GSSContext.getMech(), GSSCredential.getMechs(), GSSManager.{getMechs(), getMechsForName(), getNamesForMech()}, GSSName.getStringNameType()

*Type Of:* GSSName.{NT\_ANONYMOUS, NT\_EXPORT\_NAME, NT\_HOSTBASED\_SERVICE, NT\_MACHINE\_UID\_NAME, NT\_STRING\_UID\_NAME, NT\_USER\_NAME}