NETWORK WORKING GROUP
INTERNET-DRAFT
Expires: June 24, 2005

J. Luciani
Novell, Inc.

December 22, 2004

GSS-API V2: Java & C# Bindings
draft-ietf-kitten-gssapi-rfc2853-update-for-csharp-00

Status of this Memo

Copyright Notice

Abstract

The Generic Security Services Application Program Interface (GSS-API) offers application programmers uniform access to security services atop a variety of underlying cryptographic mechanisms. This document proposes an update to Generic Security Service API Version 2: Java Bindings [RFC2853], to include C# bindings.

The proposed updates are documented as additions to be merged into section 4 of RFC 2853.

Table of Contents

1. Introduction

This document specifies modifications to RFC 2853, Generic Security Service API Version 2: Java Bindings, that will allow it to also document C# bindings for GSS-API V2.

The C# language has recently gained much popularity with the advent of the .NET and the Mono frameworks. The C# GSS-API bindings aim to allow C# application developers to leverage the security services of the API from within those frameworks.

The design goal of the C# GSS-API was to adhere to the definition of the Java GSS-API as much as possible to leverage the work that has been done on it and to ease the transition of Java application developers to the C# environment. The following section describes additions that when merged with the contents of RFC 2853 should result in a document that covers both the Java and C# bindings of GSS-API [RFC2743].

2.0 Additions to Section 4 of RFC 2853

2.1 New Section 4.17 - Title: C# Modifications

This section describes the language dependent modifications necessary to implement the interface in C#.

2.2 New Section 4.17.1 - Title: C# Assembly Name

The C# namespace is org.ietf.gss. See section 4.17.5 for an example.

2.3 New Section 4.17.2 - Title: C# Class Definitions

All class definitions & methods remain the same as specified in the Java bindings.

2.4 New Section 4.17.3 - Title: C# Data Types

All data types remain the same.

2.5 New Section 4.17.4 - Title: C# Exception Handling

All exception codes remain the same as specified in the Java bindings. However, C# does not have a 'throws' statement. Therefore,

method prototypes do not include the exception type. For example,

Java method prototype :

```
   public abstract GSSName createName(String nameStr, Oid nameType)
      throws GSSException;
```

Equivalent C# method prototype :

```
   public abstract GSSName createName(String nameStr, Oid nameType);
```

C# does implement the throw and catch keywords, for example:

```
   public class GSSName createName(String nameStr, Oid nameType)
   {
      int majorCode = 0;
      ...

      majorCode = validateParms(nameStr, nameType);

      if (majorCode)
         throw new GSSException(majorCode);

      ...
   }
```


2.6 New Section 4.17.5: Title: C# Example Code

Client example :

```
using ietf.org.gss;

class GssapiClient
{
   private static TcpClient client;
   private static NetworkStream stream;

        static void Main(string[] args)
        {
                Connect("127.0.0.1", "message from client");

        try
        {
```

```
            GSSManager manager = GSSManager.getInstance();

            Oid krb5Mechanism = new Oid("1.2.840.113554.1.2.2");
            Oid krb5PrincipalNameType = new Oid("1.2.840.113554.1.2.2.1");

            // Optionally Identify who the client wishes to be
            // GSSName name = manager.createName("test@gsserver",
          //                                   GSSName.NT_USER_NAME);

            // Obtain default credential
            GSSCredential userCreds =
           manager.createCredential(GSSCredential.INITIATE_ONLY);
            GSSName name = userCreds.getName(krb5PrincipalNameType);

            Console.WriteLine(
           "Just acquired credentials for " + name.toString());

            int acceptLife =
           userCreds.getRemainingAcceptLifetime(new Oid("2.3.4"));
            int initLife =
           userCreds.getRemainingInitLifetime(new Oid("1..3."));
            int remLife =
           userCreds.getRemainingLifetime();
            int usage =
           userCreds.getUsage();

            GSSName namea = userCreds.getName();
            Oid[] oa = userCreds.getMechs();
```



```
         // Instantiate and initialize a security context that will be
         // established with the server
            GSSContext context = manager.createContext(name,
                                                    krb5Mechanism,
                                                    userCreds,
                                                    GSSContext.DEFAULT_LIFETI

            userCreds.dispose();

            // Optionally Set Context Options, must be done
```

```
          // before iniSecContext call.
               context.requestMutualAuth(true);
               context.requestConf(true);
               context.requestInteg(true);
               context.requestSequenceDet(true);
               context.requestCredDeleg(true);

               MemoryStream ins = new MemoryStream();
               MemoryStream outs = new MemoryStream();

               // loop until context is setup and no more tokens to receive
               while (!context.isEstablished())
               {
               outs = new MemoryStream();
                  context.initSecContext(ins, outs);

                  // send token if present
                  if (outs.Length > 0)
                  {
                          Console.WriteLine("Sending token...");
                          sendToken(outs);
                  }

                  // check if we should expect more tokens
                  if (context.isEstablished())
                          break;

                  // another token expected from peer
                  Console.WriteLine(
                "Still expecting another token from server...");
                  ins = recvToken();
               }

               //
               // display context information
               //
```



```
               // Did the server authenticate back to client?
               Console.WriteLine("\n{0} Mutual Authentication",
               context.getMutualAuthState() ? "Using" : "Not using");
```

```
  Console.WriteLine("Credentials were delegated = "
  + context.getCredDelegState());
  Console.WriteLine("Remaining lifetime in seconds = "
     + context.getLifetime());
  Console.WriteLine("Context mechanism = " + context.getMech());
  Console.WriteLine("Initiator = "
+ context.getSrcName().toString());
  Console.WriteLine("Acceptor = "
+ context.getTargName().toString());
  Console.WriteLine("Confidentiality (i.e., privacy)
is {0}available",
  context.getConfState() ? "" : "not ");
  Console.WriteLine("Integrity is {0}available",
  context.getIntegState() ? "" : "not ");
  Console.WriteLine("Is initiator = " + context.isInitiator());
  Console.WriteLine("Is transferable = "
+ context.isTransferable());
  Console.WriteLine("Is protReady = "
+ context.isProtReady());
  Console.WriteLine("ReplayDetState = " +
  context.getReplayDetState());
  Console.WriteLine("SequenceDetState = " +
  context.getSequenceDetState());

  // perform wrap on an application supplied message
  // using QOP = 0, and requesting privacy service

  MessageProp msgProp = new MessageProp(0, true);
  byte [] message =
System.Text.Encoding.ASCII.GetBytes("Hello GSS-API!");
  byte [] token =
System.Text.Encoding.ASCII.GetBytes("tok");

  // Byte aray method is equivalent to stream method
  //byte []token = context.wrap(message,
                              0,
                              appMsg.length,
                              msgProp);
  //sendToken(token);

  ins = new MemoryStream();
  outs = new MemoryStream();
  ins.Write(token, 0, token.Length);
  context.getMIC(ins, outs, msgProp);
  sendToken(outs);
```

```
            outs = new MemoryStream();
            outs.Write(message, 0, message.Length);
            sendToken(outs);

            ins = new MemoryStream();
            outs = new MemoryStream();
            ins.Write(message, 0, message.Length);
            context.wrap(ins, outs, msgProp);
            sendToken(outs);

         // Optionally export context to another thead
            GSSContext ctx = manager.createContext(context.export());
            Console.WriteLine("New context isTransferable = "
          + ctx.isTransferable());
            Console.WriteLine("New context isInitiator = "
          + ctx.isInitiator());
            Console.WriteLine("New context protReady = "
          + ctx.isProtReady());
            Console.WriteLine("New context srcName = "
          + ctx.getSrcName().toString());
            Console.WriteLine("New context targName = "
          + ctx.getTargName().toString());

            // release the local-end of the context
            ctx.dispose();

            stream.Close();
            Console.WriteLine("Leaving...");
         }
         catch (GSSException e)
         {
            Console.WriteLine(e.getMessage());
            Console.WriteLine(e.StackTrace);
         }
      }
```

3. IANA Considerations

This document has no actions for IANA.

4. Acknowledgments

The author would like to thank the following:

Corby Morris who wrote the original version of this document and is the creator of the C# GSS-API bindings.

Jeff Altman for his support and suggestions.

Kabat, J. and Upadhyay, M. for writing the Generic Security Service API Version 2 : Java Bindings specification [RFC2743] that constitutes the basis of this work.

Funding for the RFC Editor function is currently provided by the Internet Society.

5. Normative References

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[RFC2853] Kabat, J. and Upadhyay, M., "Generic Security Service API Version 2 : Java Bindings", RFC 2853, June 2000.

6. Authors' Addresses

Juan Carlos Luciani
Novell, Inc.
1800 South Novell Place
Provo, Utah 84606
US

EMail: jluciani@novell.com

7. Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

8. Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED

WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

9. Copyright Statement