Java and .NET Code Samples for SAP Logon Ticket Verification

Applies To:

NetWeaver Security (Ticket verification library SAPSSOEXT)

Summary

The example demonstrates the usage of ticket verification in external non-SAP components. These two code samples should help to give an overview only for JAVA and .NET.

By: Markus Strehle

Company and Title: SAP NetWeaver Security and Identity Management

Date: 24 Feb 2005

JAVA Example

```
import java.io.*;
/**
 * (C) Copyright 2000-2005 SAP AG Walldorf
 *
 * Author:  SAP AG, Security Development
 *
 * SAP AG DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
 * INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO
 * EVENT SHALL SAP AG BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL
 * DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR
 * PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
 * ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
 * OF THIS SOFTWARE.
 *
 * This class provides wrapper functionality for SSO2Ticket
 * (SAP Logon Ticket) in Java.
 *
 * @version 1.1 2005
 *
 */

public class SSO2Ticket
{
    public static final int ISSUER_CERT_SUBJECT = 0;
    public static final int ISSUER_CERT_ISSUER = 1;
    public static final int ISSUER_CERT_SERIALNO = 2;

    private static boolean initialized = false;
    public static String SECLIBRARY ;
    public static String SSO2TICKETLIBRARY = "sapssoext";

    static {
        if (System.getProperty("os.name").startsWith("Win"))  {
            SECLIBRARY = "sapsecu.dll";
        } else {
```



```
            SECLIBRARY = "libsapsecu.so";
        }
        try {
            System.loadLibrary(SSO2TICKETLIBRARY);
            System.out.println("SAPSSOEXT loaded.");
        } catch (Throwable e) {
            System.out.println (
              "Error during initialization of SSO2TICKET:\n" + e.getMessage());
        }
        System.out.println("static part ends.\n");
    }


    /**
     * Initialization
     *
     * @param seclib location of ssf-implemenation
     *
     * @return true/false whether initailisation was ok
     */
    private static native synchronized boolean init(String seclib);

    /**
     * Returns internal version.
     *
     * @return version
     */
    public static native synchronized String getVersion();

    /**
     * eval ticket
     *
     * @param ticket        the ticket
     * @param pab           location of pab
     * @param pab_password  password for access the pab
     *
     * @return [0] = (String)user, [1] = (String)sysid, [2] = (String)client , [3] =
(byte[])certificate
     *
     */
    public static native synchronized Object[] evalLogonTicket(
        String ticket,
        String pab,
        String pab_password)
        throws Exception;


    /**
     * Parse certificate
     * @param cert                  Certificate received from evalLogonTicket
     * @param info_id       One of the requst id´s
     *
     * @return Info string from certificate
```

```
 *
 */
public static native synchronized String parseCertificate(
    byte[] cert,
    int info_id);

public static void main(String[] args) throws Exception
{
        byte[] certificate;
        String ticket;
        String pab;
        String ssf_library;

        try {
              // plausi check
              if(getCommandParam(args,"-i") == null)
              {
                    PrintHelp();
                    return;
              }

          System.out.println("Start SSO2TICKET main");
          System.out.println("-------------- test version --------------");
          String version =SSO2Ticket.getVersion();
          System.out.println("Version of SAPSSOEXT: " + version);
        // read ticket into a String
        ticket = getTicket(getCommandParam(args,"-i"));
        // get PAB (public key) of issuing system
        pab = getFullFilePath(getCommandParam(args,"-p"));
              // init sapsecu library
              ssf_library = getCommandParam(args,"-L");
              if(ssf_library==null)
                 ssf_library = SECLIBRARY;

              if( !init(ssf_library)) {
                    System.out.println (
                     "Could not load library: " + ssf_library);
                    return;
              }
              // evaluate the ticket
        Object o[] = evalLogonTicket(
                      ticket, pab!=null?pab:"SAPdefault" , null);


        // print out all parameters received from SAPSSOEXT
        PrintResults((String)o[0],
                           (String)o[1],
                           (String)o[2],
                     parseCertificate((byte[])o[3],ISSUER_CERT_SUBJECT),
  parseCertificate((byte[])o[3],ISSUER_CERT_ISSUER),
                                    ticket);

    } catch (Exception e) {
```

```
            System.out.println(e);
        } catch (Throwable te) {
              System.out.println(te);
        }
    }

      // print the parameters from ticket
      static void PrintResults(String user, String sysid, String client,
      String subject, String issuer, String ticket)
      {
            System.out.println("***********************************************");
            System.out.println(" Output of program:");
            System.out.println("***********************************************");
            System.out.println("\n");
            System.out.println("The ticket\n\n" + ticket + "\n");
            System.out.println("was successfully validated.");
            System.out.println("User     : " + user);
            System.out.println("Ident of ticket issuing system:");
            System.out.println("Sysid    : " + sysid);
            System.out.println("Client   : " + client);
            System.out.println("Certificate data of issuing system:");
            System.out.println("Subject  : " + subject);
            System.out.println("Issuer   : " + issuer);
            System.out.println("\n");
      }

      // read the ticket string from a File
      public static String getTicket(String filename)
      throws FileNotFoundException
      {
            try {
               BufferedReader in = new BufferedReader(new FileReader(filename));
               String str;
               StringBuffer strBuffer = new StringBuffer();
               while ((str = in.readLine()) != null) {
                  strBuffer.append(str);
               }
               in.close();
               return strBuffer.toString();
            }
            catch (Exception e)
            {
                  // Let the user know what went wrong.
                  System.out.println("The file could not be read:");
                  System.out.println(e.getMessage());
                  throw new FileNotFoundException(
                          "File "+ filename +" could not be read");
            }

      }

      // parse the arguments for an option
      static String getCommandParam(String[] args, String option)
```

```
	{
		for(int i=0; i<args.length; i++)
		{
			if(args[i].equals(option) && args.length > i+1)
			{
				return args[i+1];
			}
		}
		return null;
	}

	// print help to console
	static void PrintHelp()
	{
		System.out.println("   java SSO2Ticket -i <ticket_file> [-L <SSF_LIB>]");
		System.out.println("   [-p <file containing public key>]");
	}

	// get the full path to a file
	static String getFullFilePath(String filename)
	throws FileNotFoundException
	{
		if(filename==null)
		   return null;
		String path;
		File file = new File(filename);

		if( file.getAbsolutePath().toLowerCase().indexOf(".pse") > 0 )
		{
			path = file.getAbsolutePath();
		}
		else
		{
			path = file.getAbsolutePath() + ".pse";
		}
		if( ! new File(path).exists() )
			throw new FileNotFoundException("File "+ filename +" does not
exists");
		return path;
	}
}
```

C# Example

```
/**
 *  (C) Copyright 2000-2005 SAP AG Walldorf
 *
 * Author:  SAP AG, Security Development
 *
 * SAP AG DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
 * INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO
 * EVENT SHALL SAP AG BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL
 * DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR
```

```
 * PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
 * ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
 * OF THIS SOFTWARE.
 *
 */
using System;
using System.IO;
using System.Reflection;

namespace sapssoext
{
      /// <summary>
      ///  Example class for SAPSSOEXT library implemented in CSharp (C#).
      ///
      ///  Compile this class with .Net compiler:
      ///     csc ssosample.cs
      ///
      ///  This class performs the calls via Reflection because no further
      ///  references have to be set, but the COM component in SAPSSOEXT must
      ///  be registered to the Windows registry:
      ///  1) ensure that "regsvr32 sapssoext.dll" component registration was done
      ///  2) if a null library property (Cryptlib) is passed, the environment
      ///  variable SSF_LIB is taken.
      /// </summary>
      class SSO2Ticket
      {
        // Constant definitions for
        // ParseCertificate function
        const long ISSUER_CERT_SUBJECT =0;
        const long ISSUER_CERT_ISSUER  =1;
        const long ISSUER_CERT_SERIALNO=2;
            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main(string[] args)
            {
            // plausi check
            if(getCommandParam(args,"-i") == null)
            {
                PrintHelp();
                return;
            }
            Object[] RetArray;
            Type     MyType;
            Object   MyObj;
            string   subject;
            string   issuer;
            string   ticket;

            try
            {
                // get type from registry and initiate an instance of it
```

```
                MyType = Type.GetTypeFromProgID("SAPSSOEXT.SSO2Ticket",true);
                MyObj = Activator.CreateInstance(MyType);
                // read the ticket from a File
                ticket= getTicket(getCommandParam(args,"-i"));
                // build parameters for fuction calls
                Object[] seclib = { getCommandParam(args,"-L") };
                Object[] parms =  { ticket, getFullFilePath(getCommandParam(args,"-
p")), null };
                // invoke the first call: set the property CryptLib
                MyType.InvokeMember("CryptLib",
                    System.Reflection.BindingFlags.SetProperty,null,MyObj,seclib);
                // invoke the main method to check the ticket
                RetArray = (Object[])MyType.InvokeMember("EvalLogonTicket",
                    System.Reflection.BindingFlags.InvokeMethod,null,MyObj,parms);
                // build parameters for cert parser
                Object[] parms1 = { RetArray[3], ISSUER_CERT_SUBJECT };
                // invoke ParseCertificat
                subject = (string)MyType.InvokeMember("ParseCertificate",
                    System.Reflection.BindingFlags.InvokeMethod,null,MyObj,parms1);
                // build parameters for cert parser
                Object[] parms2 = { RetArray[3], ISSUER_CERT_ISSUER };
                // invoke ParseCertificat
                issuer = (string)MyType.InvokeMember("ParseCertificate",
                    System.Reflection.BindingFlags.InvokeMethod,null,MyObj,parms2);

                // Finally print out all parameter from the ticket:
                //       RetArray(0) is the user name
                //       RetArray(1) is the client of the issuing system
                //       RetArray(2) is the id of the issuing system
                //       RetArray(3) is the X.509 certificate of the issuing system
                //       The "certificate" object is a Base64 (PEM) encoded X.509
certificate.
                //
                PrintResults((string)RetArray[0],
                    (string)RetArray[1],
                    (string)RetArray[2],
                    subject,
                    issuer,
                    ticket);
            } // error related to COM
            catch(System.Runtime.InteropServices.COMException ex)
            {
                if (ex.ErrorCode == -2147221005)
                {
                    Console.WriteLine(
            "Register object SAPSSOEXT.SSO2Ticket:\nregsvr32 sapssoext.dll");
                }
                else
                {
                    Console.WriteLine(ex.Message);
                }
            } // the inner exception means the error within SAPSSOEXT
            catch(System.Reflection.TargetInvocationException fex)
```

```
        {
            Console.WriteLine(fex.InnerException.Message);
        } // catch the rest of possible exceptions
        catch(Exception exp)
        {
            Console.WriteLine(exp.Message);
        }
        }

    // print the parameters from ticket
    static void PrintResults(string user, string sysid, string client,
        string subject, string issuer, string ticket)
    {

        Console.WriteLine("***********************************************");
        Console.WriteLine(" Output of program:");

        Console.WriteLine("***********************************************");
        Console.Write("\n");
        Console.Write("The ticket\n\n" + ticket + "\n");
        Console.WriteLine("was successfully validated.");
        Console.WriteLine("User      : " + user);
        Console.WriteLine("Ident of ticket issuing system:");
        Console.WriteLine("Sysid     : " + sysid);
        Console.WriteLine("Client    : " + client);
        Console.WriteLine("Certificate data of issuing system:");
        Console.WriteLine("Subject   : " + subject);
        Console.WriteLine("Issuer    : " + issuer);
        Console.Write("\n");
    }

    // read the ticket string from a File
    public static String getTicket(string filename)
    {
        try
        {
            // Create an instance of StreamReader to read from a file.
            // The using statement also closes the StreamReader.
            using (StreamReader sr = new StreamReader(filename))
            {
                String line = sr.ReadToEnd();
                return line;
            }
        }
        catch (Exception e)
        {
            // Let the user know what went wrong.
            Console.WriteLine("The file could not be read:");
            Console.WriteLine(e.Message);
            throw new FieldAccessException(
              "File "+ filename +" could not be read",e);
        }
```

```
        }

        // parse the arguments for an option
        public static String getCommandParam(string[] args, string option)
        {
            for(int i=0; i<args.Length; i++)
            {
                if(args[i].Equals(option) && args.Length > i+1)
                {
                    return args[i+1];
                }
            }
            return null;
        }

        // print help to console
        public static void PrintHelp()
        {
            Console.WriteLine("   ssosample -i <ticket_file> [-L <SSF_LIB>]");
            Console.WriteLine("   [-p <file containing public key>]");
        }

        // get the full path to a file
        public static String getFullFilePath(string filename)
        {
            String path;

            if( Path.HasExtension(filename) )
            {
                path = Path.GetFullPath(filename);
            }
            else
            {
                path = Path.GetFullPath(filename + ".pse");
            }
            if( ! File.Exists(path) )
                throw new FileNotFoundException(
                  "File "+ filename +" does not exists", filename);
            return path;
        }
    }
}
```

Disclaimer & Liability Notice

This document may discuss sample coding, which does not include official interfaces and therefore is not supported. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing of the code and methods suggested here, and anyone using these methods, is doing it under his/her own responsibility.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of the technical article, including any liability resulting from incompatibility between the content of the technical article and the materials and services offered by SAP. You agree that you will not hold SAP responsible or liable with respect to the content of the Technical Article or seek to do so.