October 2006

Sample Application: SSO with a .NET-based Web Service Client using SAP Logon Tickets

Summary

It is shown how to develop a .NET based Web services client that uses Windows Integrated Authentication and SAP logon tickets for Single Sign On to a SAP NetWeaver Web service. After the .NET client has successfully authenticated against a SAP NetWeaver portal with Windows Integrated Authentication using SPNego's Login Module it is issued a SAP logon ticket. The .NET client can receive the SAP logon ticket that is technically a HTTP cookie by creating a new instance of the class *HTTPWebRequest*. The SAP logon ticket is stored in the *CookieContainer* property of the *HTTPWebRequest* class and afterwards assigned that to the *CookieContainer* property of the proxy class before calling the XML Web service method. As a result SSO is achieved also for subsequent Web services Calls to an SAP NetWeaver based Web service as long as the SAP logon ticket remains valid.

Applies to

- SAP NetWeaver 2004 SP 15 and SAP NetWeaver 2004s SP 6 and higher
- mySAP Business Suite
- Microsoft Active Directory 2000 and 2003
- Microsoft Visual Studio 2005

Contact

For feedback or questions you can contact the Collaboration Technology Support Center via the .NET Technologies forum in the .NET interoperability area of SDN. Please check the .NET interoperability area in SDN for any updates or further information.
http://www.sdn.sap.com/irj/sdn/developerareas/dotnet

Author Bio

André Fischer works at **SAP AG** in the Strategic Alliance Microsoft Team. He is also a member of the Collaboration Technology Support Center – Microsoft (CTSC – MS) that addresses various kinds of interoperability topics regarding SAP and Microsoft solutions. In addition, André has lent his talents as an SAP technology consultant for the last eight years, and has gained significant experience in both the SAP and the Microsoft solution stack. In the last two years, André has also specialized in single sign-on, SAP active directory integration, SAP Exchange Infrastructure BizTalk integration and knowledge management Microsoft Windows integration.

Contents

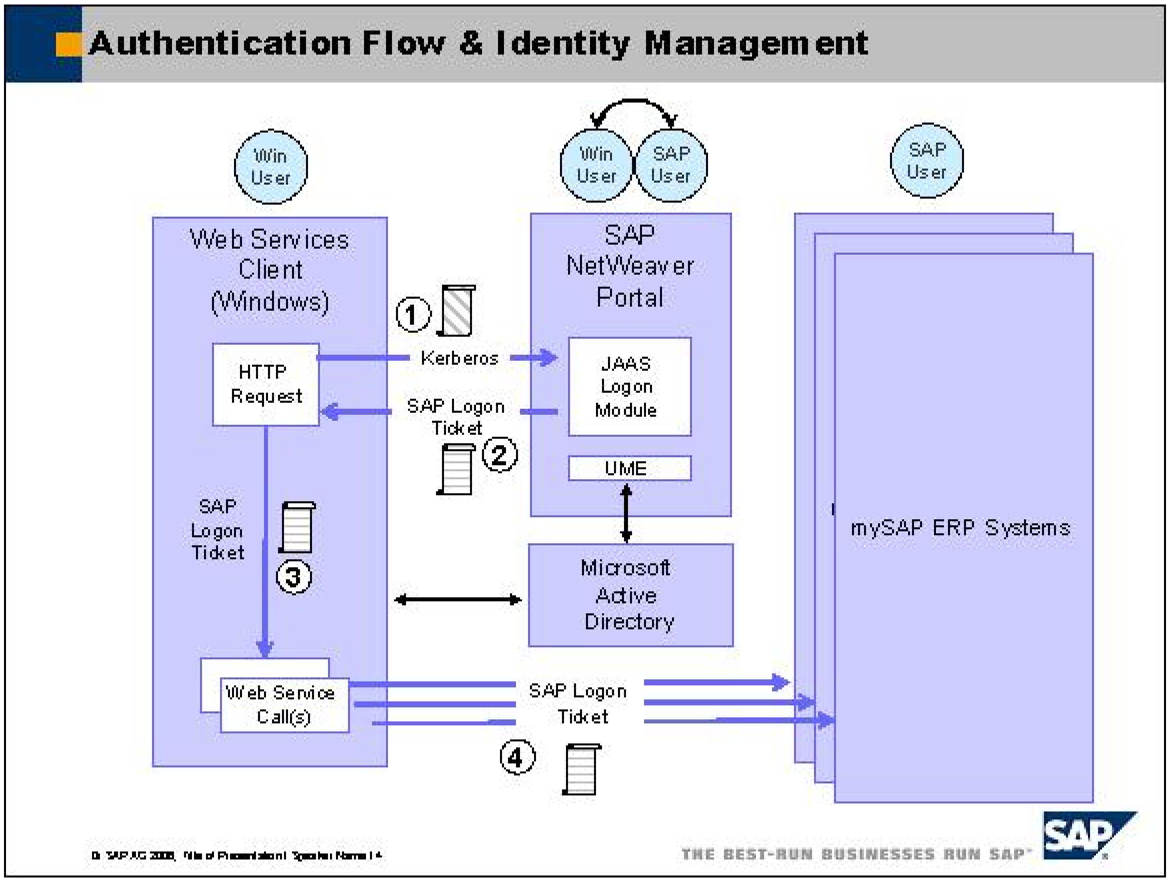
Summary 1
Applies to 1
Contact 1
Author Bio 1
Contents 3
Introduction 4
The sample application 5
- **Class SAPLogonTicket 5**
  - Property: public bool ShowTicket 5
  - Property: public *CookieContainer CookieContainer* 5
  - Method: public bool GetTicket 5
- **Class GetFlights 6**
  - Retry Code 6

Running the application 7
Configuration steps in the SAP components 7
- **SAP NetWeaver Portal 7**
  - Configuration of the SPNegoLoginModule 7
  - Exporting the public key certificate 8
  - Configuring the SAP logon ticket lifetime 8
- **mySAP ERP 8**
  - Web service Creation Wizard 8
  - Configuring the Web service to accept SAP logon tickets 9
  - Transaction STRUSTSSO2 9

Limitations 12
Outlook 12
Related Content 13
Appendix: Sample Application Source Code 14

Introduction

In a recent whitepaper Single Sign-On of Windows-based Web service Clients using SAP logon tickets we described the concept how windows based Web services clients can achieve Single Sign-On using Windows Integrated Authentication and SAP logon tickets. While the first paper provided an overview about the technical concept we will now get our hands dirty and show how to develop a .NET based application using C#. The process to use the SAP NetWeaver Portal as a ticket issuing system for SAP logon tickets is shown in the following picture:



In the first step **(1)** the .NET application performs a HTTP request (not a Web service call) to the URL of the portal server. As in the case of browser based access the authentication used is Integrated Windows Authentication by using the Kerberos Ticket of the logged on user. The .NET framework offers the class *HTTPWebRequest* that enable the user to interact directly with servers using HTTP. It is possible to use Windows Integrated Authentication for client authentication with *HTTPWebRequest*. Once the .NET application is successfully authenticated, it is issued with a SAP logon ticket. The property *CookieContainer* of the class *HTTPWebRequest* contains the cookies associated with a request and thus contains the SAP logon ticket after the user has authenticated successfully using his or hers windows credentials **(2)**.
For the Web services Call an object based on the class *HttpWebClientProtocol* is used. This class also has a *CookieContainer* Property that represents the cookies for a Web

services client. Therefore it is possible to copy the *CookieContainer* of the HTTP request that contains the SAP logon ticket to the Web services Client **(3)**.
Since this all takes place in the memory of the application the user's authentication information is therefore no longer available to services after the user closes the application.
However it can be reused by the application for subsequent Web services Calls as long as the application is active and the SAP logon ticket is valid **(4)**.

The sample application

The business case of our sample application is a clerk that works in a travel agency. He or she has to check for available flights to different destinations several times a day thereby calling a Web service that is published by an ABAP SAP NetWeaver application server. To keep the example simple from a coding perspective a console application is used. The sample application takes the URL of the ticket issuing portal as a parameter. When the application starts it acquires the SAP logon ticket from the portal and stores it in the property of an object that is based on the custom developed class *SAPLogonTicket*. The Cookie Container of the object SAPLogonTicket is copied to the Cookie Container of the Web service proxy thus allowing SSO.
The *Main* procedure performs calls to a Web service that are enclosed in an endless loop. After the flight availability has been checked the application waits for user input to either perform another search using an additional Web service call or to exit the application. Since the SAP logon ticket has been added to the *CookieContainer* Property of the Web service proxy this credentials can be used for authentication of any subsequent Web services call.
Therefore we are able to simulate the normal usage of an office application such as Microsoft Outlook where the user opens the client in the morning and does not close it until he leaves the office in the evening.

Class SAPLogonTicket

The application contains a public class *SAPLogonTicket* that performs the authentication at the ticket issuing portal. This task is encapsulated in a separate public class because it must be possible to obtain a new SAP logon ticket if the life time of the SAP logon ticket used by the client application is exceeded.

Property: public bool ShowTicket

If the public property ShowTicket is set to `true` the console application prints out the content of the SAP logon ticket.

Property: public *CookieContainer CookieContainer*

The public property *CookieContainer* which is of type *CookieContainer* is used to store the SAP logon ticket. Since the property is public it can be accessed in any other class of the program.

Method: public bool GetTicket

The public method GetTicket gets the URL of the ticket issuing portal as a parameter. It performs a HTTP request using an instance of the *HTTPWebRequest* class. If the *CookieContainer* of the http request is not empty it is assigned to the property *CookieContainer* of the Class SAPLogonTicket.

The http request uses Windows Integrated Authentication using the following coding:

```
request.Credentials = CredentialCache.DefaultCredentials;
```

If the *SPNegoLoginModule* is not yet configured in the portal yet you can still test SSO to SAP Web services if you provide hard coded credentials for the portal (username and password) in the meantime.

```
request.Credentials = new NetworkCredential("username", "pw");
```

If the authentication was successful and a logon ticket was issued the ticket is assigned to the public property *CookieContainer* (see above).

```
this.CookieContainer = request.CookieContainer;
```

Class GetFlights

The class GetFlights contains the *Main* method of our sample application. It takes the URL of the ticket issuing SAP portal as a parameter. The coding starts with the creation of objects based on the custom developed class *SAPLogonTicket* (see above) and a Web services proxy to call the SAP NetWeaver Web service.

Retry Code

The HTTP request and the Web services call throw a *WebException* when errors occur while accessing the Internet resource.
If the SAP logon ticket that is used by the Web services client is not valid any more the following Error will occur: *"The request failed with* HTTP *status 401: Unauthorized."*
To handle such exceptions the Web services call and the initial http request that acquires the SAP logon ticket is performed inside a *try* block. The question that arises is how to branch back to the *try* block from the *catch* block if the situation has occurred that the SAP logon ticket has become invalid?
Though exceptions in the common sense are by nature unrecoverable the failure of the Web services authentication caused by an invalid SAP logon ticket because it's lifetime has exceeded is not. The only action the Web services client has to perform is to acquire a new SAP logon ticket by re-authentication against the SAP portal.
This is a case where the much-maligned *goto* statement is useful. By defining a label called *GetSAPLogonTicket:* just before the try block it is possible to jump back to the initial authentication step. The maximum number of *attempts* to get a new SAP logon ticket is specified by the variable *maxAttempts*.
In the coding I added a line that can be used to simulate an invalid SAP logon ticket. If the if-statement `if ( attempts > 0)` is active the Cookie Container will not be transferred to the Web services proxy with the first attempt. An exception will be thrown since the Web services client cannot authenticate against the Web service.

```
//if ( attempts > 0)

myProxy.CookieContainer = myTicket.CookieContainer;
```

If the Web services call was successful the list of available flights is shown to the user. The programme continues to ask the user if another destination should be checked or whether the programme should be terminated.

Running the application

Running the console application gives the following output :

```
Specify the URL to obtain the SSO2 Ticket.
Trying the following URL:
http://sapportal.mycompany.com:50000/irj/portal instead (Y/N) ?
y
Connect to http://sapportal.mycompany.com:50000/irj/portal

HTTP Return Code OK

MYSAPSSO2=AjExMDAgAB5wb3J0YWw6RDA0MTYxNUBNU0NUU0MuU0FQLkNPUlCIABNiYXNpY2F1dGhlbn
RpY2F0aW9uAQAHRDA0MTYxNQIAAzAwMAMAA1AwMwQADDIwMDYxMDI2MDk0NQUABAAAAAgKAAdEMDQxNj
E1%2FwD1MIHyBgkqhkiG9w0BBwKggeQwgeECAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3DQEHATGBwTCB
vgIBATATMA4xDDAKBgNVBAMTA1AwMwIBADAJBgUrDgMCGgUAoF0wGAYJKoZIhvcNAQkDMQsGCSqGSIb3
DQEHATAcBgkqhkiG9w0BCQUxDxcNMDYxMDI2MDk0NTM0WjAjBgkqhkiG9w0BCQQxFgQUEUhuDg%2F%2F
wZG7C4gr6RVjhLuh0jswCQYHKoZIzjgEAwQvMC0CFQDBCaLqtCNfGslY2KJRyGw4vgfKqAIUQdB4uHry
fyiHMR9PKm2AzkUSvGk%3D

got ticket

Check another destination?
Please enter name of destination
or 'N' to cancel:
new york

Calling Web service
1 flight found

-------------------------------------
AIRLINE = Lufthansa
Flight = LH 0400
FROM = FRANKFURT
TO = NEW YORK
Departure = 1995-02-28 10:10:00
Arrival = 1995-02-28 11:34:00
-------------------------------------

Check another destination?
Please enter name of destination
or 'N' to cancel:
san francisco

Calling Web service
1 flight found

-------------------------------------
AIRLINE = Lufthansa
Flight = LH 0454
FROM = FRANKFURT
TO = SAN FRANCISCO
Departure = 1995-11-17 10:10:00
Arrival = 1995-11-17 12:30:00
-------------------------------------
Check another destination?
Please enter name of destination
or 'N' to cancel:
```

Configuration steps in the SAP components

We don’t want to duplicate the content of the SAP NetWeaver Online Help. However we want to point you to the resources needed and give an overview about the most important configuration steps.

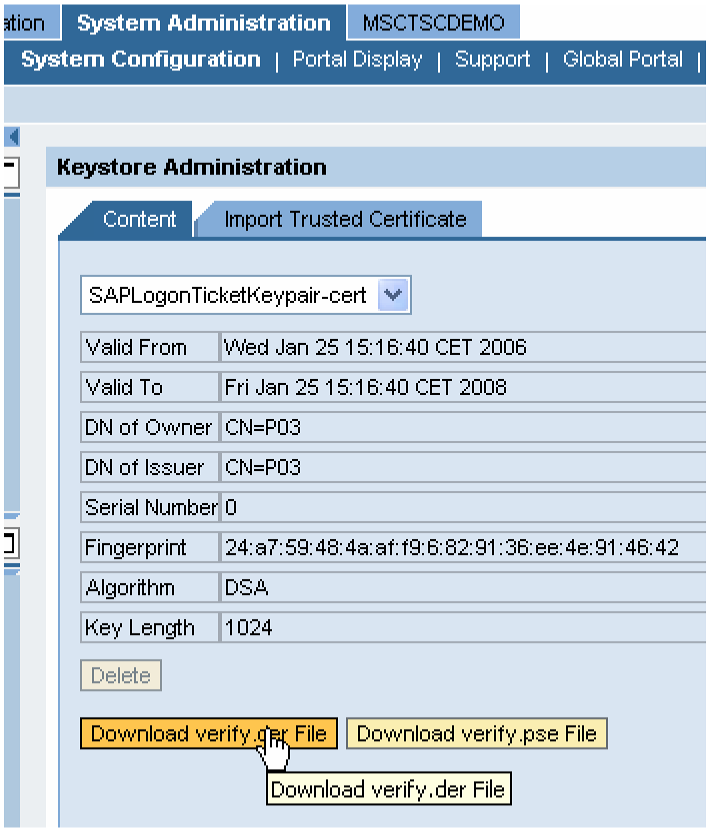
SAP NetWeaver Portal

Configuration of the SPNegoLoginModule

The portal has to support windows integrated authentication. The configuration of the SPNegoLoginModule is out of scope of this whitepaper. Please refer to the documentation available in the SAP Online that are mentioned in *Related Content*.

Exporting the public key certificate

The keystore administration tool only contains *TicketKeystore* which contains the private and public key of the Portal Server and its certificate. To access the keystore administration too in the portal, choose *System Administration → SystemConfiguration → Keystore Administration.*



SAPLogonTicketKeypair-cert

Valid From	Wed Jan 25 15:16:40 CET 2006
Valid To	Fri Jan 25 15:16:40 CET 2008
DN of Owner	CN=P03
DN of Issuer	CN=P03
Serial Number	0
Fingerprint	24:a7:59:48:4a:af:f9:6:82:91:36:ee:4e:91:46:42
Algorithm	DSA
Key Length	1024

Configuring the SAP logon ticket lifetime

During the development you may want to test the behaviour of the application when the lifetime of a ticket is exceeded. The lifetime of the SAP logon tickets issued by a portal server can be configured using the parameter *login.ticket_lifetime*. This parameter contains the number of hours that the logon ticket is valid. You can also have values of the form *hh:mm*, thus allowing you to specify a login lifetime that spans only several minutes.
Alternatively you can remove the ticket programmatically when the client tries to call the Web service for the first time. If the target system is accessible the status code of the http-response will be the same as if the lifetime of the ticket is expired.

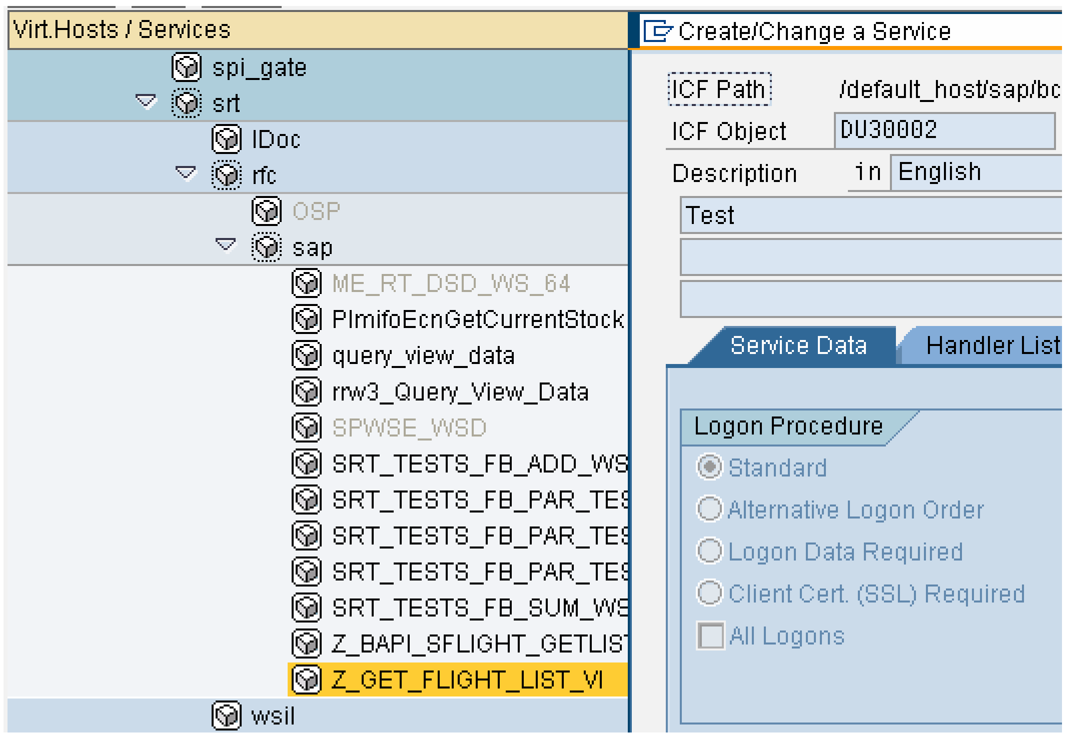
mySAP ERP

Web service Creation Wizard

As described in the SAP Online Help Creating a Web service we published the function module BAPI_SFLIGHT_GETLIST as a Web service. There is also a new Web services tutorial available in SDN.

Configuring the Web service to accept SAP logon tickets

In the transaction SICF you can display and change the Web server settings for your Web service. You will find your Web service beneath *default_host/sap/bc/srt/rfc/sap*



Virt.Hosts / Services

- spi_gate
- srt
  - IDoc
  - rfc
    - OSP
    - sap
      - ME_RT_DSD_WS_64
      - PlmifoEcnGetCurrentStock
      - query_view_data
      - rrw3_Query_View_Data
      - SPWSE_WSD
      - SRT_TESTS_FB_ADD_WS
      - SRT_TESTS_FB_PAR_TES
      - SRT_TESTS_FB_PAR_TES
      - SRT_TESTS_FB_PAR_TES
      - SRT_TESTS_FB_SUM_WS
      - Z_BAPI_SFLIGHT_GETLIS
      - Z_GET_FLIGHT_LIST_VI
  - wsil

Create/Change a Service

ICF Path /default_host/sap/bc
ICF Object DU30002
Description in English
Test

Service Data | Handler List

Logon Procedure
- [✓ Standard]
- Alternative Logon Order
- Logon Data Required
- Client Cert. (SSL) Required
- All Logons

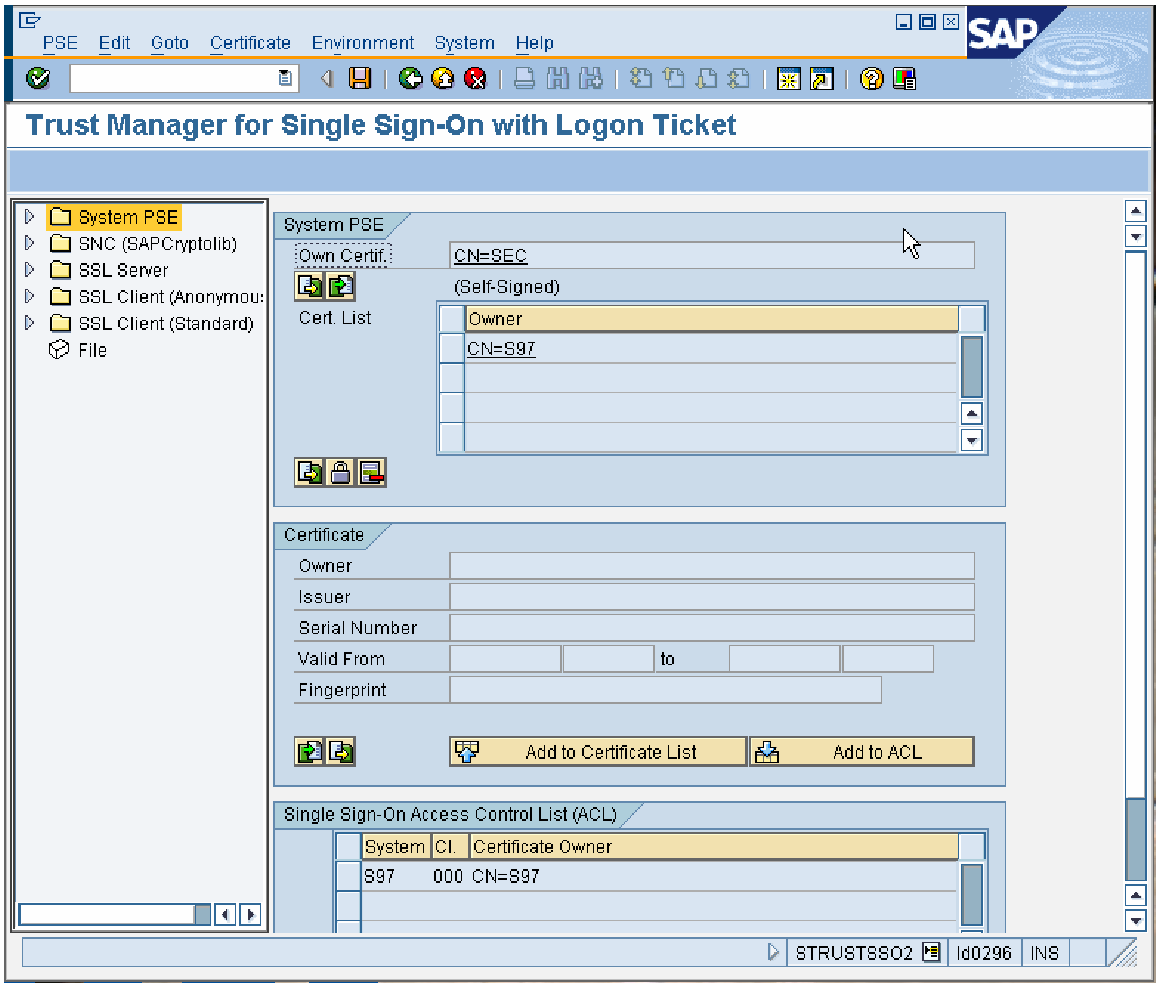
If the default option *Standard* is chosen for the Logon Procedure multiple checks are run in a specified order. These are in particular:

1. Fields Authentication (logon using HTTP fields)
2. SSO Authentication (logon using Single Sign-On)
3. Basic Authentication
4. SAP Authentication (logon using SAP user and password)
5. Certificate Authentication (logon using a client certificate)
6. Service Authentication (logon using the anonymous logon data stored in the service)

You thus have nothing special to configure so that SAP logon tickets can be used to authenticate against the Web service.

Transaction STRUSTSSO2

Since the ABAP Web AS should accept logon tickets of the portal server it must have access to the issuing server's public-key certificate so that it can verify the digital signature provided with the ticket. Therefore the public certificate of the ticket issuing SAP NetWeaver Portal has to be exported from its key store as described above and has to be imported into all systems of the SAP system landscape.



Trust Manager for Single Sign-On with Logon Ticket

System PSE
SNC (SAPCryptolib)
SSL Server
SSL Client (Anonymou:
SSL Client (Standard)
File

System PSE

Own Certif. CN=SEC

(Self-Signed)

Cert. List

Owner
CN=S97

Certificate

Owner	
Issuer	
Serial Number	
Valid From	to
Fingerprint	

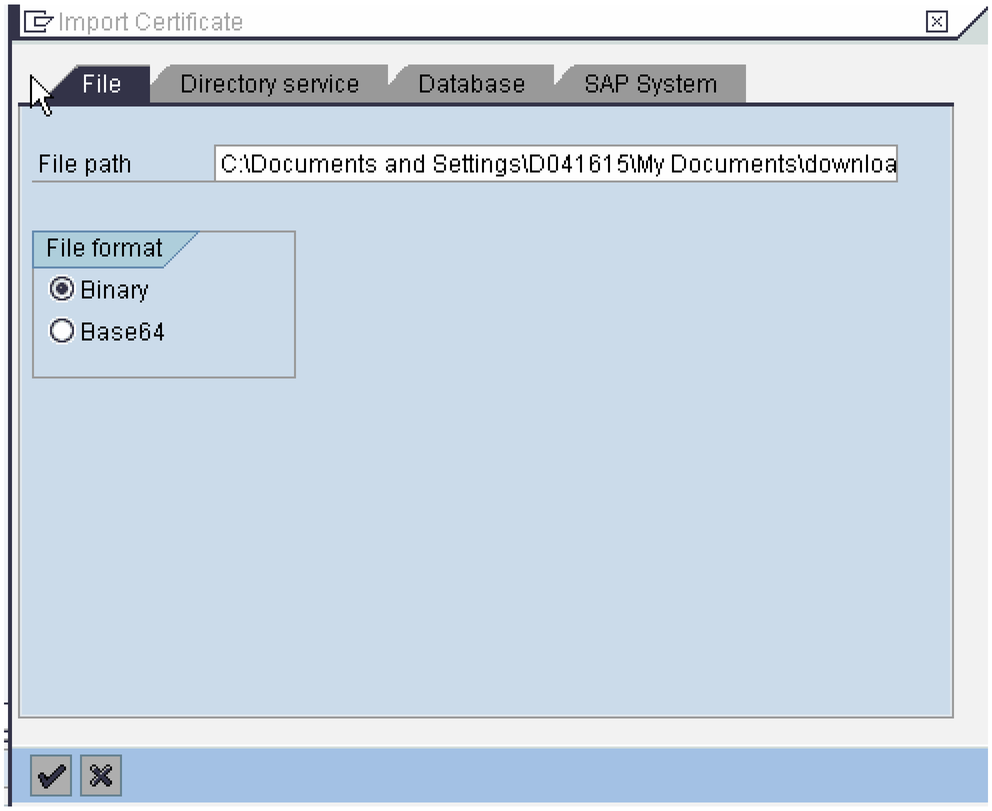
Add to Certificate List Add to ACL

Single Sign-On Access Control List (ACL)

System	Cl.	Certificate Owner
S97	000	CN=S97

STRUSTSSO2 ld0296 INS

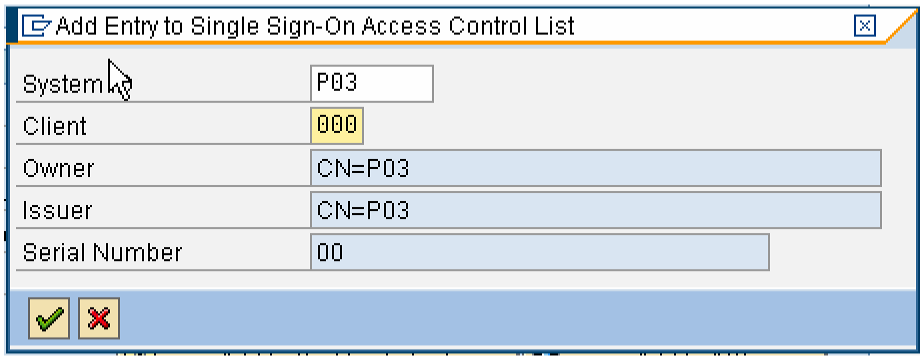
Select the *der-File* and upload the certificate. It is not imported with this step.



File path: C:\Documents and Settings\D041615\My Documents\downloa

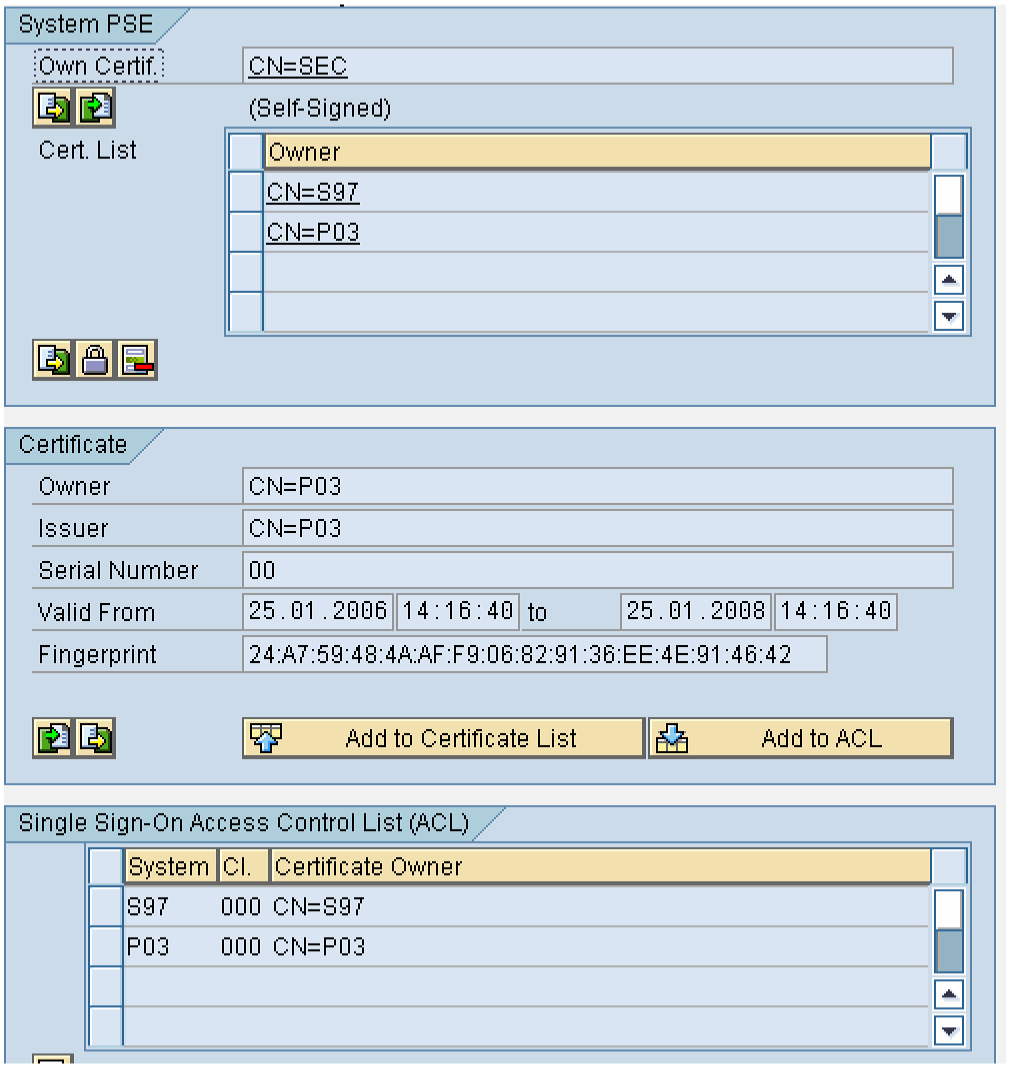
File format: (● Binary) ○ Base64

After the der-File has been specified the content has to be uploaded. Therefore press the buttons *Add to Certificate List* and *Add to ACL*. If *Add to ACL* is pressed the following dialog pops up:



System	P03
Client	000
Owner	CN=P03
Issuer	CN=P03
Serial Number	00

As a result the PSE now contains the following entries:



System PSE

Own Certif.	CN=SEC
	(Self-Signed)

Cert. List

Owner
CN=S97
CN=P03

Certificate

Owner	CN=P03
Issuer	CN=P03
Serial Number	00
Valid From	25.01.2006 14:16:40 to 25.01.2008 14:16:40
Fingerprint	24:A7:59:48:4A:AF:F9:06:82:91:36:EE:4E:91:46:42

Single Sign-On Access Control List (ACL)

System	Cl.	Certificate Owner
S97	000	CN=S97
P03	000	CN=P03

Don't forget to save the System PSE.

Limitations

Logon tickets are only sent to SAP NetWeaver Application Servers that are located in the same DNS domain as the SAP NetWeaver Portal Server that issued the ticket. Therefore SAP logon tickets are not well suited for cross domain scenarios.

Outlook

Upcoming major releases of both the SAP and the Microsoft technology stacks will allow interoperability based on SAML support for Web services technology. It will be possible to update a Web services scenario build on existing technologies as described in this whitepaper easily using the upcoming advanced Web services standards.

Related Content

- Web services Security Interoperability with SAP NetWeaver and Microsoft .NET Part I , SDN
- Web services Security Interoperability with SAP NetWeaver and Microsoft .NET Part II , SDN
- SAP Online Help: Using Kerberos Authentication for Single Sign-On
- Single Sign-On of Windows-based Web service Clients using SAP logon tickets
- SAP Online Help: Creating a Web service
- Web services tutorial, SDN
- Inside C#, Second Edition , Microsoft Press

Appendix: Sample Application Source Code

```
using System;
using System.Net;

namespace TestSSO
{


    class SAPLogonTicket
    {

        public SAPLogonTicket()
        {
            this.ShowTicket = false;
        }

       // protected string ticketissuerURL;

        public bool ShowTicket;

        public bool GetTicket(string ticketissuerURL)
        {
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(ticketissuerURL);
            request.CookieContainer = new CookieContainer();

            // Set some reasonable limits on resources used by this request
            request.MaximumAutomaticRedirections = 4;
            request.MaximumResponseHeadersLength = 4;

            // Set credentials to use for this request.
            // if no SPNego Login Module is configured
            // basic authentication can be used
            // as a workaround

            //request.Credentials = new NetworkCredential("username", "pw");
            request.Credentials = CredentialCache.DefaultCredentials;

            Console.WriteLine("Connect to " + ticketissuerURL);
            HttpWebResponse response = (HttpWebResponse)request.GetResponse();
            Console.WriteLine("HTTP Return Code {0}", response.StatusCode);

            this.CookieContainer = request.CookieContainer;

            if (ShowTicket == true )
            {
                foreach (Cookie cook in response.Cookies)
                {
                    if (cook.Name == "MYSAPSSO2")
                    {
                        Console.WriteLine(cook.Name + "=" + cook.Value);
                    }
                 }
             }

             if (CookieContainer != null)
                 return true;
             else
                 return false;
        }

        public CookieContainer CookieContainer;

    }


    /// <summary>
    /// Summary description for Class1.
    /// </summary>
    class GetFlights
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            //
            // TODO: Add code to start application here
            //

            string myURL = "http://sapportal.mycompany.com:50000/irj/portal";
            int numFlights = 0;
            int attempts = 0;
            int maxAttempts = 2;

            // Create Object for SAP Logon Tickets
            SAPLogonTicket myTicket = new SAPLogonTicket();
            myTicket.ShowTicket = true;

            // Create a proxy for the web service
            BAPI_FLIGHT_GETLIST.Z_GET_FLIGHT_LIST_VIService myProxy = new
            BAPI_FLIGHT_GETLIST.Z_GET_FLIGHT_LIST_VIService();

            // Create a Cookie Container for the WebService
            myProxy.CookieContainer = new CookieContainer();

            // define parameters
            BAPI_FLIGHT_GETLIST.BAPISFLDRA[] dateRange = new
            BAPI_FLIGHT_GETLIST.BAPISFLDRA[0];
            BAPI_FLIGHT_GETLIST.BAPISFLDST destinationFrom = new
            BAPI_FLIGHT_GETLIST.BAPISFLDST();
            BAPI_FLIGHT_GETLIST.BAPISFLDST destinationTo = new
            BAPI_FLIGHT_GETLIST.BAPISFLDST();
            BAPI_FLIGHT_GETLIST.BAPIPAREX[] extensionIn = new
            BAPI_FLIGHT_GETLIST.BAPIPAREX[0];
            BAPI_FLIGHT_GETLIST.BAPIPAREX[] extensionOut = new
            BAPI_FLIGHT_GETLIST.BAPIPAREX[0];
            BAPI_FLIGHT_GETLIST.BAPISFLDAT[] flightList = new
            BAPI_FLIGHT_GETLIST.BAPISFLDAT[0];
            BAPI_FLIGHT_GETLIST.BAPIRET2[] bapiReturn = new
            BAPI_FLIGHT_GETLIST.BAPIRET2[0];

            // initialize parameters
            destinationFrom.AIRPORTID = "";
            destinationFrom.CITY = "";
            destinationFrom.COUNTR = "";
            destinationFrom.COUNTR_ISO = "";
            destinationTo.AIRPORTID = "";
            destinationTo.CITY = "";
            destinationTo.COUNTR = "";
            destinationTo.COUNTR_ISO = "";



            if (args == null || args.Length != 1)
            {
                Console.WriteLine("Specify the URL to obtain the SSO2 Ticket.");
                Console.WriteLine("Trying the following URL:");
                Console.WriteLine(myURL + " instead (Y/N) ?");
                string response = Console.ReadLine();
                response = response.ToUpper();
                if (response != "Y") Environment.Exit(1);
             }

            else
            {
                myURL = args[0];
            }


            GetSAPLogonTicket:

            try
            {
                // get SAP Logon Ticket

                // The Cookie Container that contains the SAP Logon Ticket
                // is transferred to the Cookie Container of the WebService Request
                // create Cookie Container to store SAP Logon Ticket

                if (myTicket.GetTicket(myURL) == true)
                {
                    Console.WriteLine("got ticket");

                    //activate the following if statement
                    //to simulate an invalid SAP Logon Ticket
                    //since the SAP Logon Ticket than will only
                    //be transferred with the second attempt

                    //if ( attempts > 0)
                    myProxy.CookieContainer = myTicket.CookieContainer;
                }
                else
                    Console.WriteLine("did not get ticket");

                // set SAP credentials with username and password to test web service
                // if no SAP Logon Ticket issuing system is available
                // myProxy.Credentials = _
                // new System.Net.NetworkCredential("testsapuser", "secret");

                // endless loop until user enters "N"
                while(true)
                {
                    Console.WriteLine("Check another destination?");
                    Console.WriteLine("Please enter name of destination");
                    Console.WriteLine("or 'N' to cancel:");

                    string response = Console.ReadLine();
                    response = response.ToUpper();
                    if (response == "N")
                        Environment.Exit(1);
                    else
                        destinationTo.CITY = response;

                // call SAP Web Service
                Console.WriteLine("Calling Web Service");

                myProxy.BAPI_FLIGHT_GETLIST("",
                ref dateRange,
                destinationFrom,
                destinationTo,
                ref extensionIn,
                ref extensionOut,
                ref flightList,
                20,
                true,
                ref bapiReturn);

                if (flightList != null)
                {
                    numFlights = flightList.GetLength(0);

                    if (numFlights == 1)
                    {
                        Console.WriteLine("{0} flight found",
                        flightList.Length.ToString());
                        Console.WriteLine();
                    }
                    else
                    {
                        Console.WriteLine("{0} flights found",
                        flightList.Length.ToString());
                        Console.WriteLine();
                    }

                    for (int i = 0; i < numFlights; i++)
                    {
                        Console.WriteLine("-------------------------------------");
                        Console.WriteLine("AIRLINE = {0} ", flightList[i].AIRLINE);
                        Console.WriteLine("Flight = {0} {1}", flightList[i].AIRLINEID,
                                          flightList[i].CONNECTID);
                        Console.WriteLine("FROM = {0} ", flightList[i].CITYFROM);
                        Console.WriteLine("TO = {0} ", flightList[i].CITYTO);
                        Console.WriteLine("Departure = {0} {1}",
                                          flightList[i].FLIGHTDATE,
                                          flightList[i].DEPTIME);
                        Console.WriteLine("Arrival = {0} {1}", flightList[i].ARRDATE,
                                          flightList[i].ARRTIME);
                    }

                    Console.WriteLine("-------------------------------------");
                }
                else
                    Console.WriteLine("No flights found");
            }

            }

            catch (WebException e)
            {
                // When the response to an Internet request indicates an error,
                // WebRequest.GetResponse sets the Status property
                // to WebExceptionStatus.ProtocolError and provides
                // the WebResponse that contains the error message in the
                // Response property of the WebException that was thrown.
                // The application can examine the WebResponse to determine
                // the actual error.

                Console.WriteLine("WebException: " + e.Message);

                if (e.Status == WebExceptionStatus.ProtocolError)
                {
                    if (++attempts < maxAttempts)
                    {
                        Console.WriteLine("Status Code : {0}",
                        ((HttpWebResponse)e.Response).StatusCode);

                        Console.WriteLine("Status Description : {0}",
                        ((HttpWebResponse)e.Response).StatusDescription);

                        Console.WriteLine("Attempt Number {0}. Retrying ...", attempts);
                        goto GetSAPLogonTicket;
                    }
                    else
                    {
                        Console.WriteLine("Authentication using " +
                                          "SAP Logon Ticket failed.");
                        Console.WriteLine("after {0} attempts!", maxAttempts);
                    }

                }

            }
```

```
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.Message);
            }

            // waiting for input so that Command Prompt does not close
            Console.ReadKey();

        }
    }
}
```