

LINEE GUIDA PER L'INTEGRAZIONE CON ORACLE LOGIN SERVER

PARTE 2: APPLICAZIONI ASP.NET

Ver. 1.0

Data	Versione	Descrizione	Cap. /Sez. modificati
novembre 2003	1.0	Nascita del documento	tutti

INDICE

1	Introduzione	4
2	Architettura Applicativa.....	5
2.1	Schema operativo procedura di login da un'applicazione ASP.NET	5
3	Linee guida di programmazione	7
3.1	Prerequisiti applicazione ASP.NET da integrare.....	7
3.2	Configurazione di un sito ASP.NET con Forms Authentication.....	7
3.3	Parametri di configurazione	7
3.4	Pagina di Login della Partner Application ASP.NET (Login.aspx)	8
3.5	Pagina di verifica del token ed autenticazione (VerifyToken.aspx)	9
3.6	Accesso alla Funzione SSOSERVER_URL del login server	10
3.7	Accesso alla Funzione SSOSERVER_TOKEN del login server.....	12
4	Package per l'integrazione Asp.NET/SSO	14

1 Introduzione

Il presente documento deve essere considerato una specifica tecnica del documento “Linee guida per l’integrazione con Oracle Login Server – parte 1: applicazioni java”. Riassumiamo brevemente alcuni punti chiave esposti nel documento di riferimento.

Con il termine *partner application* si intende un’applicazione che delega al login server la funzionalità di autenticazione. E’ compito dell’applicazione verificare se l’utente è già autenticato e nel caso contrario innescare il flusso di autenticazione.

La logica di autenticazione e del repository di utenti e password non è compito della partner application che delega tali funzioni al login server.

Scopo del presente allegato è fornire una serie di linee guida per l’integrazione di “partner application” sviluppate con tecnologia Microsoft ASP.NET in ambito MEF.

Si precisa che il termine SSO usato nel seguito indica la funzionalità di logon unico, mentre i termini “Login server” e “Single Sign On (SSO) Server” individuano i prodotti software Oracle nei 2 diversi release che realizzano la funzionalità di logon unico.

2 Architettura Applicativa

I paragrafi seguenti descrivono il processo che caratterizza il SSO in ambito MEF.

2.1 Schema operativo procedura di login da un'applicazione ASP.NET

Nel seguente paragrafo viene descritto il flusso operativo del meccanismo di autenticazione di una partner application sviluppata in tecnologia Microsoft ASP.NET, secondo quanto riportato nel documento di riferimento (par. 3.1.2 – L'utente accede tramite il portale alla Partner Application 2).

Nella figura che segue è rappresentato lo schema operativo del flusso dove sono state riportate principalmente le interazioni tra il browser e l'applicazione ASP.NET e quelle tra il login server e l'applicazione ASP.NET; sono state tralasciate le interazioni tra il login server ed il browser già descritte nel documento di riferimento:

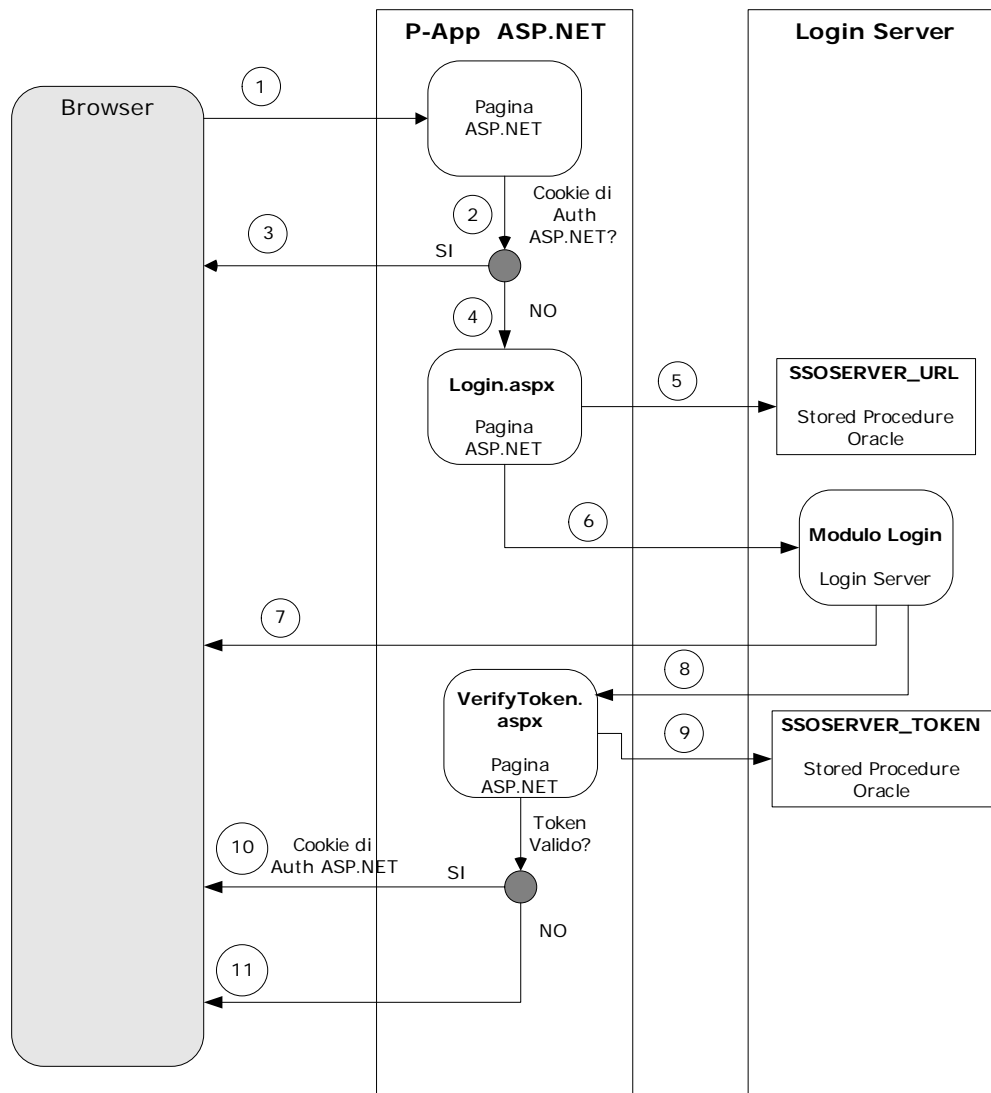


Figura 1: Architettura applicativa

Il processo può essere così descritto:

1. l'utente, utilizzando un generico browser, si collega ad una qualsiasi delle pagine dell'applicazione ASP.NET per la quale è richiesta l'autenticazione;
2. l'applicazione ASP.NET, configurata per l'utilizzo della Forms Authentication, verifica tramite il security provider la presenza e la validità del cookie di autenticazione;
3. **se il cookie esiste ed è valido** all'utente viene visualizzata la pagina richiesta. Ciò vuol dire che l'utente è stato già autenticato durante una fase precedente e può continuare la sua normale attività all'interno dell'applicazione.
4. **se il cookie non esiste oppure non è valido** l'utente viene rediretto sulla pagina di login dell'applicazione che, nel nostro esempio, è stata denominata *login.aspx*. Tale pagina, priva di User Interface, è l'anello di congiunzione tra la partner application ed il sistema di login server. Il colloquio tra l'applicazione ASP.NET ed il login server avviene in una modalità mista composta di: chiamate HTTP (get) e l'esecuzione di Stored Procedure presenti nel Database Oracle del login server.
5. la pagina *login.aspx* interroga, tramite la stored procedure **SSOServer_URL**, il login server il quale genera e restituisce l'URL che fa riferimento al modulo di autenticazione del login server;
6. la pagina *login.aspx* dirige l'utente sull'URL ricavata nel passo precedente;
7. il login server richiede all'utente le credenziali necessarie per procedere alla fase di autenticazione come descritto nel documento di riferimento;
8. il Login server, dopo aver riconosciuto l'utente, provvede a ridirigere l'utente su una pagina dell'applicazione ASP.NET chiamante, che nell'esempio è stata denominata *verifytoken.aspx*. Un Token di avvenuta autenticazione viene passato come parametro della pagina.
9. la pagina *verifytoken.aspx* verifica, tramite la stored procedure **SSOSERVER_TOKEN**, la validità del Token e, solo se valido, ottiene la UserId dell'utente autenticato;
10. **se il token è valido** l'applicazione ASP.NET autentica l'utente restituendo il cookie di autenticazione al browser e visualizzando la pagina richiesta al passo 1;
11. **se il token non è valido** l'applicazione ASP.NET non autentica l'utente per cui non è possibile accedere alle pagine

Una volta conclusa con successo la fase di autenticazione l'utente ha a disposizione:

- il cookie di autenticazione dell'applicazione ASP.NET;
- il cookie di Single Sign On del login server.

Nessuna ulteriore autenticazione sarebbe richiesta se l'utente volesse accedere ad una qualsiasi delle altre partner application.

3 Linee guida di programmazione

In questa sezione vengono fornite le linee guida di programmazione e configurazione per integrare un'applicazione ASP.NET con il sistema di login server.

Viene inoltre fornito del codice di esempio (.NET - C#) con l'obiettivo di agevolare l'implementazione del flusso di autenticazione descritto nella sezione precedente.

3.1 Prerequisiti applicazione ASP.NET da integrare

Affinchè sia possibile utilizzare questa modalità di integrazione la partner application deve soddisfare i seguenti requisiti:

- sia scritta utilizzando il .NET Framework;
- sia un'applicazione ASP.NET che utilizzi la modalità di autenticazione "Forms Authentication"
- sia abilitata a dialogare con il login server via HTTP;
- l'application server dell'applicazione ASP.NET deve poter accedere al database del login server.

3.2 Configurazione di un sito ASP.NET con Forms Authentication

E' necessario effettuare i seguenti passi per configurare l'utilizzo della forms authentication per l'applicazione ASP.NET :

- abilitare la modalità Forms nella sezione "authentication" del file web.config della propria applicazione. Qui di seguito viene mostrato un esempio di come configurare opportunamente il file web.config

```
<authentication mode="Forms">
    <forms name="NETSSO" loginUrl="login.aspx"
        timeout="30" path="/" protection="All">
    </forms>
</authentication>
<authorization>
    <deny users="?" />
</authorization>
```

- inserire nell'applicazione la pagina login.aspx fornita nell'esempio nella root dell'applicazione;
- inserire nell'applicazione la pagina verifytoken.aspx fornita nell'esempio.
- consentire l'accesso anonimo alle due pagine login.aspx e verifytoken.aspx.

Attivando la modalità di autenticazione *Forms Authentication*, come mostrato nel riquadro, la pagina login.aspx viene invocata tutte le volte che un utente non ancora autenticato cerca di accedere ad una risorsa che richiede l'autenticazione.

3.3 Parametri di configurazione

Affinchè l'applicazione possa accedere al database Oracle del login server è necessario configurare opportunamente la stringa di connessione al database. La stringa di connessione al DB Oracle è configurabile tramite un parametro nella sezione appsetting (Application Setting) del file web.config così come segue:

```
<configuration>
  <appSettings>
    <add key="dsnOLoginServer"
value="Provider=OraOLEDB.Oracle.1;Password=ssosdk;Persist Security
Info=True;User ID=ssosdk;Data Source=consip_dst01.tesoro.it"/>
  </appSettings>
</configuration>
```

Figura 2: Parametri di configurazione

3.4 Pagina di Login della Partner Application ASP.NET (Login.aspx)

Tale pagina viene invocata dal security provider ogni qual volta è necessario verificare le credenziali dell'utente ovvero quando:

- un utente non autenticato, sprovvisto di cookie di autenticazione, cerca di accedere ad una risorsa che richiede l'autenticazione;
- il cookie di autenticazione viene invalidato o modificato;
- la sessione è scaduta per cui è necessario autenticarsi nuovamente.

Al caricamento della pagina viene invocato il metodo **getLoginServerURLProc** dell'oggetto di business **SCIOPNET_TX.clsLoginServer** che ritorna l'URL del login server.

Se il metodo restituisce il valore booleano true viene effettuata una redirectione al login server, per permettere all'utente di inserire le proprie credenziali a fronte delle quali verrà riconosciuto. Nel caso contrario viene restituito un messaggio di errore.

Qui di seguito viene riportato il codice della pagina login.aspx utilizzata come anello di congiunzione tra il login server e l'applicazione ASP.NET.

```
using System.Configuration;

private void Page_Load(object sender, System.EventArgs e)
{
    string retURL = Request["ReturnURL"];
    string sLoginSSOURL;
    string sErrore;
    string Connessione =
ConfigurationSettings.AppSettings["dsnOLoginServer"];

    SCIOPNET_TX.clsLoginServer objLoginServer= new
    SCIOPNET_TX.clsLoginServer(Connessione);

    if (objLoginServer.getLoginServerURLProc(retURL, "", out sLoginSSOURL,
    out sErrore))
    {
        Response.Redirect(sLoginSSOURL);
    }
}
```



```
}  
else  
{  
Response.Write("Errore " + sErrore);  
}  
}
```

Figura 3: Parte del codice della pagina login.aspx

3.5 Pagina di verifica del token ed autenticazione (VerifyToken.aspx)

Tale pagina viene invocata dal login server dopo aver eseguito la fase di riconoscimento dell'utente; il login server provvede a valorizzare il parametro HTTP **urlc** con un token crittografato che identifica l'utente autenticato.

Al caricamento della pagina viene invocato il metodo **AuthLoginServerProc** dell'oggetto di business **SCIOPNET_TX.clsLoginServer**, il cui codice sarà descritto successivamente, il quale preso in input il token ne verifica la validità e restituisce la UserID dell'utente.

Se il metodo restituisce il valore booleano **true** viene invocato il metodo **SetAuthCookie** dell'oggetto **FormsAuthentication**, fornito dal .NET Framework, il quale provvede a settare il cookie di autenticazione e, quindi, ad autenticare l'utente che sarà ridirezionato sulla pagina inizialmente richiesta.

Se il metodo restituisce il valore booleano **false** viene restituito un messaggio di errore.

```
using System.Configuration;  
using System.Web.Security;  
  
private void Page_Load(object sender, System.EventArgs e)  
{  
    string Token = Request["urlc"];  
    string UserID;  
    string ReturnURL;  
    bool Auth;  
    string Errore;  
  
    // Metodo VerifyToken su login server IN Token OUT Auth, UserID  
    string Connessione =  
        ConfigurationSettings.AppSettings["dsnOLoginServer"];  
    clsLoginServer Obj = new clsLoginServer(Connessione);  
  
    Auth=Obj.AuthLoginServerProc(Token, out UserID, out ReturnURL, out  
    Errore);  
  
    if (Auth)  
    {  
        //Se Auth  
        FormsAuthentication.SetAuthCookie(UserID, false);  
        Response.Redirect("http://" +  
            Request.ServerVariables["SERVER_NAME"] + ReturnURL.Trim());  
    }  
}
```

```
else
{
    Response.Write("errore!!" + Errore);
}
```

Figura 4: Parte del codice della pagina verifytoken.aspx

3.6 Accesso alla Funzione SSOSERVER_URL del login server

Come descritto precedentemente, durante la fase di autenticazione è necessario interfacciare dall'applicazione ASP.NET la stored procedure **SSOSever_URL** il cui compito è quello di generare e restituire l'URL che fa riferimento al modulo di autenticazione del login server. Qui di seguito viene descritta la stored procedure, i parametri di **input** sono:

- *p_app_name* – *varchar2(200)*: contiene la chiave con cui viene effettuata la ricerca all'interno del Login Server(prodotta in fase di registrazione della partner application.)
- *requested_url* – *varchar2(200)*: contiene l'URL della partner application che l'utente ha richiesto originariamente;
- *cancel_url* – *varchar2(200)*: contiene l'URL della partner application a cui deve essere ridiretto l'utente nel caso in cui selezioni il tasto "cancel" durante la fase di autenticazione.
- I parametri di **output** sono:
- *p_gen_redirect_url* – *varchar2(200)*: contiene l'URL del login sever a cui deve essere ridirezionato l'utente, tale URL viene generata dinamicamente e contiene un token crittografato;
- *p_error* – *varchar2(200)*: contiene eventuali messaggi di errore restituiti dal login server, nel caso in cui non ci siano stati errori la procedura restituisce "TRUE".

Qui di seguito è riportato il codice di esempio del metodo **getLoginServerURLProc** contenuto nella classe .NET **clsLoginServer** che accede al database Oracle utilizzando il provider OLEDB. Il metodo viene richiamato dalla pagina login.aspx.

```
public bool getLoginServerURLProc(
    string ReturnURL,
    string CancelURL,
    out string SSURL ,
    out string Errore)
{
    string dsn = _Connessione;

    OleDbCommand myCommand = new OleDbCommand();
    OleDbDataReader objDataReader;
    myCommand.Connection = new OleDbConnection(dsn);
    myCommand.CommandType=CommandType.StoredProcedure;
    myCommand.CommandText="ms_ssosdk.SSOSERVER_URL" ;

    myCommand.Connection.Open();

    OleDbParameter ParRequestedURL = new OleDbParameter("@requested_url" ,
        OleDbType.VarChar, 255);
    ParRequestedURL.Value = ReturnURL;
    myCommand.Parameters.Add(ParRequestedURL);

    OleDbParameter ParCancelURL = new OleDbParameter ("@cancel_url",
        OleDbType.VarChar, 255);
    ParCancelURL.Value = CancelURL;
    myCommand.Parameters.Add(ParCancelURL);

    OleDbParameter ParUrlGen = new OleDbParameter("@p_gen_redirect_url",
        OleDbType.VarChar,1000);
    ParUrlGen.Direction=ParameterDirection.Output;
    myCommand.Parameters.Add(ParUrlGen);

    OleDbParameter ParErrore = new OleDbParameter("@p_error", OleDbType.VarChar, 255);
    ParErrore.Direction=ParameterDirection.Output;
    myCommand.Parameters.Add(ParErrore);

    int numRows=myCommand.ExecuteNonQuery();

    SSURL=myCommand.Parameters["@p_gen_redirect_url"].Value.ToString();
    Errore=myCommand.Parameters["@p_error"].Value.ToString();

    if (Errore.Trim()=="TRUE")
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Figura 5: Parte del codice del metodo getLoginServerURLProc

3.7 Accesso alla Funzione SSOSERVER_TOKEN del login server

Come descritto precedentemente, durante la fase di autenticazione è necessario interfacciare dall'applicazione ASP.NET la stored procedure **SSOServer_TOKEN** il cui compito è quello di verificare il token ed eventualmente restituire la UserID dell'utente autenticato. Qui di seguito viene descritta la stored procedure, i parametri di **input** sono:

- *p_app_name* – *varchar2(200)*: contiene la chiave con cui viene effettuata la ricerca all'interno del Login Server(prodotta in fase di registrazione della partner application.)
- *urlc* – *varchar2(200)*: contiene il token fornito dal login server ;
- I parametri di **output** sono:
- *p_sso_user_name* – *varchar2(200)*: contiene la userID dell'utente; se il token non è valido viene restituita una stringa vuota;
- *p_groups* – *varchar2(200)*: contiene il gruppo o i gruppi a cui appartiene l'utente separati dal carattere(:); se il token non è valido viene restituita una stringa vuota;
- *p_url_requested* – *varchar2(200)*: contiene l'URL della partner application che l'utente ha richiesto originariamente;
- *p_error* – *varchar2(200)*: contiene eventuali messaggi di errore restituiti dal login server, nel caso in cui non ci siano stati errori la procedura restituisce "TRUE".

Qui di seguito è riportato il codice di esempio del metodo **AuthLoginServerProc** contenuto nella classe .NET **clsLoginServer** che accede al database Oracle utilizzando il provider OLEDB.

Il metodo viene richiamato dalla pagina verifytoken.aspx.

```
public bool AuthLoginServerProc(String SSOToken, out String User, out
String RequestedUrl, out String Errore)
{
    bool authenticated = false;
    string dsn = _Connessione;

    OleDbCommand myCommand = new OleDbCommand();
    OleDbDataReader objDataReader;

    myCommand.Connection = new OleDbConnection(dsn);
    myCommand.CommandType=CommandType.StoredProcedure;
    myCommand.CommandText="ms_ssosdk.PARSE_SSOSERVER_TOKEN" ;
    myCommand.Connection.Open();

    OleDbParameter ParToken = new OleDbParameter("@urlc" ,
    OleDbType.VarChar,1000);
    ParToken.Value = SSOToken;
    myCommand.Parameters.Add(ParToken);

    OleDbParameter ParUserName = new OleDbParameter
    ("@p_sso_user_name",OleDbType.VarChar,100);
    ParUserName.Direction=ParameterDirection.Output;
    myCommand.Parameters.Add(ParUserName);

    OleDbParameter ParUrlRequested = new OleDbParameter
    ("@p_url_requested",OleDbType.VarChar,255);
```

```
ParUrlRequested.Direction=ParameterDirection.Output;
myCommand.Parameters.Add(ParUrlRequested);

OleDbParameter ParErrore = new OleDbParameter
("@p_error",OleDbType.VarChar,255);
ParErrore.Direction=ParameterDirection.Output;
myCommand.Parameters.Add(ParErrore);

int numRows=myCommand.ExecuteNonQuery();

User =
myCommand.Parameters["@p_sso_user_name"].Value.ToString().Trim();

RequestedUrl =
myCommand.Parameters["@p_url_requested"].Value.ToString().Trim();

Errore = myCommand.Parameters["@p_error"].Value.ToString().Trim();

if (Errore.Trim()=="TRUE")
{
    authenticated=true;
}
else
{
    authenticated=false;
}

return authenticated;
}
```

Figura 6: Parte del codice del metodo AuthLoginServerProc

4 Package per l'integrazione Asp.NET/SSO

Di seguito viene elencato lo specs ed il body del package necessario all'integrazione tra Asp.NET ed il componente SSO di Oracle Portal.

Si precisa che tale codice costituisce un esempio e viene qui allegato "as is".

Package Specs:

```
CREATE OR REPLACE PACKAGE MS_SSOSDK IS

    --g_listener_token VARCHAR2(1000) := 'sciop_net';
    --g_requested_url VARCHAR2(1000) :=
    'http://scsii429.scsii.tesoro.it/PartnerApplASPNET';
    --g_cancel_url VARCHAR2(1000) := 'http://scsii429.scsii.tesoro.it/';
    g_string_success VARCHAR2(10) := 'TRUE';
    g_separator VARCHAR2(1) := ':';

FUNCTION GET_GROUPS (user IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2;

-- RESTITUISCE LA URL DEL LOGIN SERVER
-- La procedura data una requested url
-- restituisce la url del login server con
-- il parametro site2pstoretoken che contiene
-- l'informazione della requested url
--
-- *****
-- INPUT
-- * requested_url => url da chiamare dopo il login
-- * cancel_url => url nel caso utente clicca cancel
-- OUTPUT
-- * p_gen_redirect_url => url del Login Server (compreso di parametro
site2pstoretoken)
-- * p_error => Eventuale errore (TRUE se tutto e' ok)
-- *****
--
--
PROCEDURE SSOSERVER_URL
(
    p_app_name IN VARCHAR2
    ,requested_url IN VARCHAR2 --DEFAULT g_listener_token
    ,cancel_url IN VARCHAR2 --DEFAULT g_cancel_url
    ,p_gen_redirect_url OUT VARCHAR2
    ,p_error OUT VARCHAR2
);
-- RESTITUISCE USERNAME DATO IL TOKEN
-- La procedura dato il token generato del Login Server
-- restituisce l'utente e la url richiesta (quella che
-- ha invocato il Login Server)
--
-- *****
-- INPUT
-- * urlc => Token Generato dal Login Server
-- OUTPUT
-- * p_sso_user_name => nome utente loggato
-- * p_groups => lista dei gruppi separati da ':'
-- * p_url_requested => url richiesta (url che ha invocato il Login Server)
-- * p_error => Eventuale errore (TRUE se tutto e' ok)
-- *****
--
--
--
```

```
PROCEDURE PARSE_SSOSERVER_TOKEN
(
    p_app_name IN VARCHAR2,
    urlc IN VARCHAR2,
    p_sso_user_name OUT VARCHAR2,
    p_groups OUT VARCHAR2,
    p_url_requested OUT VARCHAR2,
    p_error OUT VARCHAR2
);

/*
FUNCTION GET_SSOSERVER_URL
(
    requested_url IN VARCHAR2 DEFAULT g_listener_token
    ,cancel_url IN VARCHAR2 DEFAULT g_cancel_url
)
RETURN VARCHAR2;

FUNCTION GET_SSOSERVER_USER
(
    urlc IN VARCHAR2
)
RETURN VARCHAR2;

FUNCTION GET_SSOSERVER_URL_REQUESTED
(
    urlc IN VARCHAR2
)
RETURN VARCHAR2;
*/

END; -- Package spec
/
```

Package Body:

```
CREATE OR REPLACE PACKAGE BODY MS_SSOSDK IS

FUNCTION TABLE_TO_STRING (arr IN portal30.wwsec_api.idarray )
RETURN VARCHAR2 IS
str VARCHAR2(4000) := '';
j integer :=1;
BEGIN
LOOP
    str:=portal30.wwsec_api.group_name(arr(j)) || g_separator || str;
    j:=j+1;
END LOOP;
RETURN str;

EXCEPTION
    WHEN OTHERS THEN
        RETURN str ;
END TABLE_TO_STRING;

FUNCTION GET_GROUPS (user IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2 IS
```

```
l_groupid_array portal30.wwsec_api.idarray;
j integer;
str varchar2(4000);
BEGIN
l_groupid_array := portal30.wwsec_api.user_in_groups(p_user_name => user);
str:=table_to_string(l_groupid_array);
RETURN str;
EXCEPTION
    WHEN OTHERS THEN
        RETURN str;
END GET_GROUPS;

PROCEDURE SSOSERVER_URL
(
    p_app_name      IN VARCHAR2
    ,requested_url  IN VARCHAR2 --DEFAULT g_listener_token
    ,cancel_url     IN VARCHAR2 --DEFAULT g_cancel_url
    ,p_gen_redirect OUT VARCHAR2
    ,p_error        OUT VARCHAR2
) IS
BEGIN
    p_gen_redirect_url := --
'http://sdpart4.tesoro.it/pls/portal30_sso_col/portal30_sso.wwsso_app_admin.ls_1
    login';
    wwsec_sso_enabler.generate_redirect
    (
        p_lsnr_token    => p_app_name,
        p_url_requested => requested_url,
        p_url_cancel    => cancel_url
    );
    p_error:=MS_SSOSDK.g_string_success;
EXCEPTION
    WHEN no_data_found OR
        wwsec_sso_enabler.CONFIG_MISSING_EXCEPTION THEN
        p_error:= 'Error in application: missing application registration
information';
    WHEN others THEN
        p_error:= 'Error in application:' || sqlerrm;
END SSOSERVER_URL;

PROCEDURE PARSE_SSOSERVER_TOKEN
(
    p_app_name IN VARCHAR2,
    urlc IN VARCHAR2,
    p_sso_user_name OUT VARCHAR2,
    p_groups OUT VARCHAR2,
    p_url_requested OUT VARCHAR2,
    p_error OUT VARCHAR2
) IS

    l_sso_user_name      VARCHAR2(1000);
    l_ip_address        VARCHAR2(1000);
    l_sso_time_remaining VARCHAR2(1000);
    l_site_time_stamp   VARCHAR2(1000);
    l_url_requested     VARCHAR2(1000);
BEGIN
    wwsec_sso_enabler.parse_url_cookie
    (
        p_lsnr_token    => p_app_name,
        p_enc_url_cookie => urlc,
        p_sso_username  => l_sso_user_name,
        p_ipaddr        => l_ip_address,
        p_sso_timerremaining => l_sso_time_remaining,
        p_site_timestamp => l_site_time_stamp,
        p_url_requested => l_url_requested
    );
    p_sso_user_name := l_sso_user_name;
```



```

p_groups := MS_SSOSDK.GET_GROUPS(l_sso_user_name);
p_url_requested := l_url_requested;
p_error := MS_SSOSDK.g_string_success;
EXCEPTION
  WHEN wwsec_sso_enabler.CONFIG_MISSING_EXCEPTION THEN
    p_error:= 'Enabler configuration missing : ' || sqlerrm;
  WHEN wwsec_sso_enabler.UNSUPPORTED_VERSION_EXCEPTION THEN
    p_error:= 'SSO Server version not supported: ' || sqlerrm;
  WHEN wwsec_sso_enabler.IPADDR_ERROR_EXCEPTION THEN
    p_error:= 'IP address mismatch. ' || sqlerrm;
  WHEN wwsec_sso_enabler.COOKIE_EXPIRED_EXCEPTION THEN
    p_error:= 'Authentication token send by Login Server has expired ' ||
sqlerrm;
  WHEN OTHERS THEN
    p_error:= 'Unknown error: ' || sqlerrm;
END PARSE_SSOSERVER_TOKEN;

/*
FUNCTION GET_SSOSERVER_URL
(
  requested_url IN VARCHAR2 DEFAULT g_listener_token
, cancel_url   IN VARCHAR2 DEFAULT g_cancel_url
)
RETURN VARCHAR2 IS
  l_gen_redirect_url VARCHAR2(32000);
BEGIN
  l_gen_redirect_url :=
    wwsec_sso_enabler.generate_redirect
  (
    p_lsnr_token    => g_listener_token,
    p_url_requested => requested_url,
    p_url_cancel    => cancel_url
  );
  -- http.p('<br>g_listener_token:""/|g_listener_token|/"<br>'); --mario
  -- http.p('<br>g_requested_url:""/|g_requested_url|/"<br>'); --mario
  -- http.p('<br>l_gen_redirect_url:""/|l_gen_redirect_url|/"<br>'); --mario
  RETURN l_gen_redirect_url;
EXCEPTION
  WHEN no_data_found OR
    wwsec_sso_enabler.CONFIG_MISSING_EXCEPTION THEN
    RETURN 'Error in application: missing application registration
information';
  WHEN others THEN
    RETURN 'Error in application:' || sqlerrm;
END GET_SSOSERVER_URL;

FUNCTION GET_SSOSERVER_USER
(
  urlc IN VARCHAR2
)
RETURN VARCHAR2 IS
  l_sso_user_name      VARCHAR2(1000);
  l_ip_address         VARCHAR2(1000);
  l_sso_time_remaining VARCHAR2(1000);
  l_site_time_stamp    VARCHAR2(1000);
  l_url_requested      VARCHAR2(1000);
BEGIN
  --http.p('<br>c g_listener_token "/|g_listener_token|/"<br>');
  --http.p('<br>c urlc "/|urlc|/"<br>');
  wwsec_sso_enabler.parse_url_cookie
  (
    p_lsnr_token      => g_listener_token,
    p_enc_url_cookie  => urlc,
    p_sso_username    => l_sso_user_name,
    p_ipaddr          => l_ip_address,
    p_sso_timeremaining => l_sso_time_remaining,

```

```
        p_site_timestamp      => l_site_time_stamp,
        p_url_requested        => l_url_requested
    );
    --http.p('<br>c l_sso_user_name '||l_sso_user_name||'<br>');
    RETURN l_sso_user_name;
EXCEPTION
    WHEN wwsec_sso_enabler.CONFIG_MISSING_EXCEPTION THEN
        RETURN 'Enabler configuration missing : ' || sqlerrm;
    WHEN wwsec_sso_enabler.UNSUPPORTED_VERSION_EXCEPTION THEN
        RETURN 'SSO Server version not supported: ' || sqlerrm;
    WHEN wwsec_sso_enabler.IPADDR_ERROR_EXCEPTION THEN
        RETURN 'IP address mismatch. ' || sqlerrm;
    WHEN wwsec_sso_enabler.COOKIE_EXPIRED_EXCEPTION THEN
        RETURN 'Authentication token send by Login Server has expired ' ||
sqlerrm;
    WHEN OTHERS THEN
        RETURN 'Unknown error: ' || sqlerrm;
END GET_SSOSERVER_USER;

FUNCTION GET_SSOSERVER_URL_REQUESTED
(
    urlc IN VARCHAR2
)
RETURN VARCHAR2 IS

    l_sso_user_name          VARCHAR2(1000);
    l_ip_address             VARCHAR2(1000);
    l_sso_time_remaining     VARCHAR2(1000);
    l_site_time_stamp        VARCHAR2(1000);
    l_url_requested          VARCHAR2(1000);
BEGIN
    wwsec_sso_enabler.parse_url_cookie
    (
        p_lsnr_token          => g_listener_token,
        p_enc_url_cookie      => urlc,
        p_sso_username        => l_sso_user_name,
        p_ipaddr              => l_ip_address,
        p_sso_timerremaining   => l_sso_time_remaining,
        p_site_timestamp      => l_site_time_stamp,
        p_url_requested        => l_url_requested
    );
    RETURN l_url_requested;
EXCEPTION
    WHEN wwsec_sso_enabler.CONFIG_MISSING_EXCEPTION THEN
        RETURN 'Enabler configuration missing : ' || sqlerrm;
    WHEN wwsec_sso_enabler.UNSUPPORTED_VERSION_EXCEPTION THEN
        RETURN 'SSO Server version not supported: ' || sqlerrm;
    WHEN wwsec_sso_enabler.IPADDR_ERROR_EXCEPTION THEN
        RETURN 'IP address mismatch. ' || sqlerrm;
    WHEN wwsec_sso_enabler.COOKIE_EXPIRED_EXCEPTION THEN
        RETURN 'Authentication token send by Login Server has expired ' ||
sqlerrm;
    WHEN OTHERS THEN
        RETURN 'Unknown error: ' || sqlerrm;
END GET_SSOSERVER_URL_REQUESTED;
*/

END MS_SSOSDK;
/
```