Allegato 9: Specifiche tecniche per l'utilizzo dei sistemi di autenticazione centralizzata

1 Considerazioni generali

Questo documento ha lo scopo di descrivere le possibili soluzioni centralizzate da adottare per implementare l'autenticazione nelle applicazioni ospitate sulle infrastrutture della Regione Emilia-Romagna:

- Sistema di Identity & Access management (IAM),
- Soluzione custom.

Sistema di Identity & Access management (IAM)

2 Gestione degli utenti di un'applicazione con il sistema di Identity Management

L'applicazione deve delegare al sistema di Identity la gestione degli utenti. L'identity ha bisogno di oggetti chiamati "connettori" che gli permettano di interagire con il repository degli utenti utilizzato dall'applicazione (es. una tabella utenti su un db). In linea generale esistono due tipologie di connettori:

- connettori standard,
- connettori custom.

I primi sono connettori già implementati da Sun. In questo caso non è necessaria alcuna fase di sviluppo e si può passare alla fase di integrazione. Per informazioni circa i connettori standard si rimanda alla documentazione di prodotto e in particolare al documento **Sun Java System Identity Manager 7.1 Resources Reference**.
Nel secondo caso, è invece necessario sviluppare dei metodi **network-enable** (API, Store Procedure, Web Services) che devono essere esposti per l'integrazione con il sistema di IdM.
Tali metodi sono normalmente un sottoinsieme dei seguenti, in funzione delle necessità:

- createUser, crea un account sul target
- deleteUser, cancella un account sul target
- getUser, resituisce la vista di un utente sul target
- getAccountIterator, restituisce un iteratore sull'accoundID degli utenti sul target
- update, aggiorna i dati di un utente sul target (compresa la password)
- enable, abilita l'utente sul target
- disable, disabilita l'utente sul target
- listProfile, effettua la lista di tutti i profili del target
- test, testa il funzionamento del servizio

Dei metodi precedenti dovranno inoltre essere definiti i parametri di input e di output in base al target da integrare (alcuni target potranno avere dei parametri differenti).
Particolare attenzione va al campo password, nel caso in cui nel target che si vuole integrare essa sia gestita.
Qualora infatti si debba gestire la password sul target, deve essere indicato al gruppo di IdM l'algoritmo di cifratura utilizzato.

2.1 Use case per la scelta dei connettori

Di seguito vengono presentati alcuni use case di esempio per la scelta del connettore da implementare.

- In caso di integrazione di target con utenti gestiti su un'unica tabella di DB, in cui non sia possibile sviluppare delle Store Procedure, esiste un connettore specifico "Database Table" che effettuerà delle operazioni di Insert, Update e Delete direttamente sul database.
- In caso di integrazione di target tramite Store Procedure, deve essere utilizzato il connettore "Scripted JDBC" opportunamente customizzato (già utilizzato per le applicazioni Vetrina Sostenibilità, FTPS, ACollab e Atti).
- In caso di integrazione di target con connettori non presenti nel documento IDM_Resource_Reference, sarà possibile svilupparne di nuovi utilizzando il Sun Resource Extension Facility Kit (fornito nella Directory /REF del prodotto) che rappresenta la guida per creare connettori custom. In questa Directory sono presenti anche codici di esempio e tool per la creazione di nuovi connettori. Tale REF Kit rappresenta una modalità di sviluppo del connettore a basso livello.

3 Accesso alle web application

Per poter accedere ad una Web Application protetta da un sistema di Web SSO, l'utente deve prima aver effettuato il Logon al sistema di autenticazione.
Effettuato il Logon Primario, l'utente può accedere alle Web Application esposte.

Ad ogni richiesta di accesso vengono ripetuti i seguenti passi:

1. La richiesta viene intercettata dal Reverse Proxy
2. L'Agent del Reverse Proxy verifica la presenza di una sessione autenticata per l'utente. Per il mantenimento della sessione viene utilizzato un cookie volatile sul client. Nel caso non esista una sessione associata alla richiesta, l'utente viene diretto verso il modulo di autenticazione del sistema di Web SSO centralizzato.
3. Il sistema di Web SSO autentica l'utente e restituisce le informazioni all'Agent che a sua volta trasmette le informazioni necessarie alla web application dell'aderente utilizzando l'header http.

3.1 Requisiti di progettazione delle interfacce

Il passaggio dei parametri tra il reverse proxy e l'applicazione avviene sfruttando i meccanismi standard del Web e cioè gli header del protocollo HTTP, quindi la Web Application deve essere in grado di gestire gli header HTTP.
Nel caso in cui l'applicazione voglia filtrare ulteriormente gli accessi può prelevare gli attributi dell'utente dall'http header ed effettuare la profilazione applicativa.

3.2 Formato parametri

I parametri passati nell'header HTTP alla web application esposta servono ad identificare ed a caratterizzare l'originatore della richiesta.
Il parametro necessariamente presente nell'header HTTP dovrà essere il seguente:

Parametro	Significato
USERNAME	Identificativo utente (identifica l'originatore della richiesta).

Potranno essere aggiunti ulteriori parametri da passare nell'http header. Questi parametri sono prelevati dagli attributi dell'utente.
I parametri elencati saranno presenti nell'header HTTP di ogni richiesta. La modalità di accesso alle variabili dell'header sono legate al linguaggio utilizzato per la creazione e la gestione delle pagine

web. Al termine del presente documento sono riportati esempi di accesso a queste variabili in JSP, PERL, ASP.NET, ASP.

3.3 Vincoli

Sono riportati di seguito i vincoli a cui le Web Application devono attenersi per poter essere protette dal sistema di Web SSO.

3.3.1 Identificazione e Autenticazione dell'utente

La web application esposta non deve richiedere il login agli utenti che accedono, in quanto il processo di identificazione e autenticazione viene già effettuato nella fase di Logon Primario che l'utente effettua sul sistema di Web SSO. In particolare, l'applicazione non deve richiedere l'immissione esplicita da parte dell'utente di un username e di una password, ma può utilizzare le informazioni di identificazione dell'utente contenute nell'header HTTP di ogni richiesta

3.3.2 Autorizzazione dell'utente

Il processo di autorizzazione all'accesso da parte dell'utente alla web application esposta è effettuato dal Reverse Proxy, che abilita o impedisce l'accesso a singole Applicazioni in base a policy prestabilite.
E' facoltà dell'applicazione di estendere il processo di autorizzazione svolto dal Reverse-Proxy, utilizzando le informazioni contenute nell'header HTTP di ogni richiesta per realizzare nuove regole di autorizzazione.

3.3.3 Uso di Cookie

Il meccanismo di Logon e le verifiche effettuate dal Reverse Proxy si basano sullo scambio di *cookie* con la Postazione di Lavoro dell'utente che necessariamente deve permettere l'utilizzo di *cookie*.

3.3.4 Convenzioni sui nomi dei domini

Una web application esposta dalla RER potrà essere accessibile, sia da Internet che dalla rete interna, mediante una URL così strutturata:

https://applicazioni.regione.emilia-romagna.it/<*percorso-applicazione*>/

dove <*percorso-applicazione*> è il *path* (al limite costituito da un singolo nome) scelto per identificare la web application (è ammesso il passaggio di parametri nella URL).

3.3.5 Convenzioni sui nomi delle web application

Qualora un servizio esponga più di una web application, gli URL corrispondenti si differenziano solo relativamente alla componente <*percorso applicazione*>.
Esempio:

https://<dominio>/<percorso applicazione -1>/

https://<dominio>/<percorso applicazione -2>/

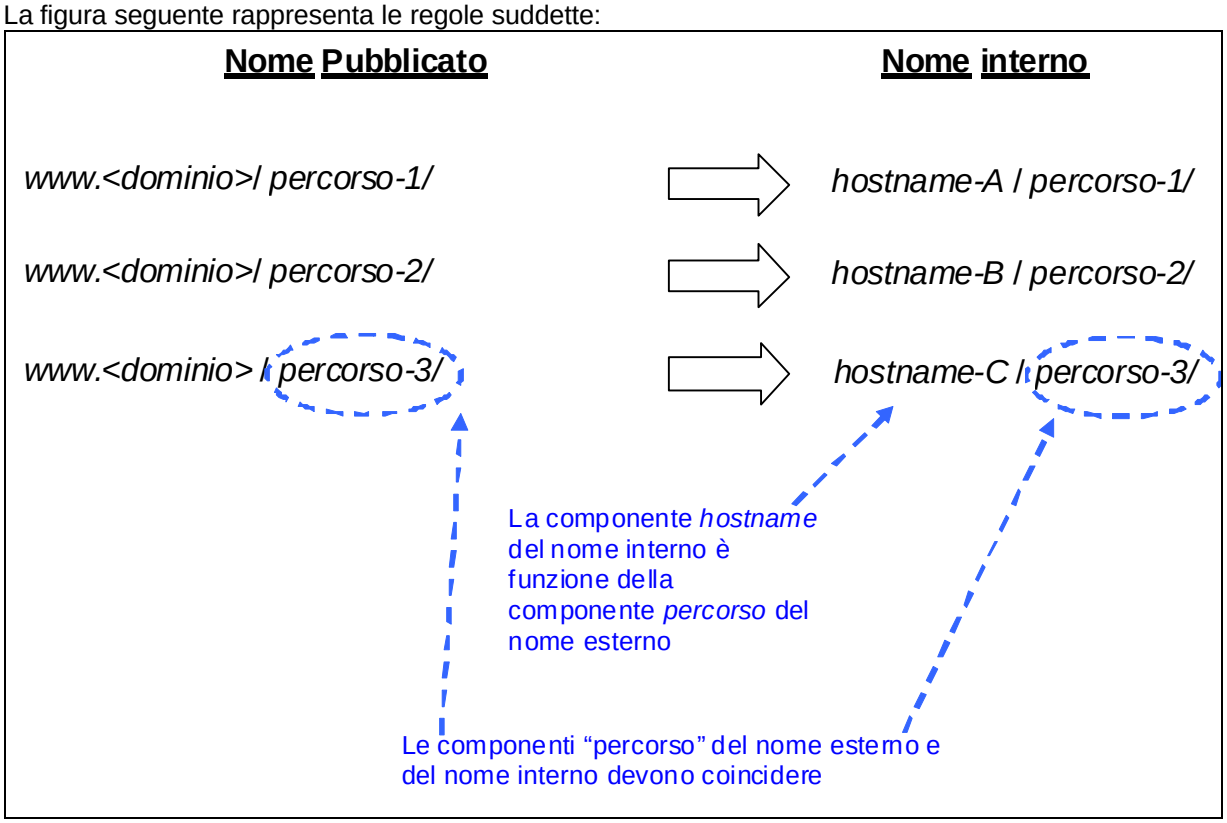
https://<dominio><percorso applicazione -3>/

3.3.6 Regole d'instradamento del Reverse Proxy

La convenzione sui nomi degli URL si riferisce all'esposizione delle web application, ovvero al modo in cui tali applicazioni sono esposte tramite Reverse Proxy e non implica che la medesima convenzione debba necessariamente essere adottata internamente all'applicativo.
Internamente all'applicativo, le web application possono risiedere su uno o più host. La corrispondenza tra il "nome esterno" e il "nome interno" dell'applicazione viene effettuata dal Reverse Proxy tramite le *regole di instradamento*. Tali regole consentono di collocare le proprie web application sui server della rete interna, svincolandosi dall'*hostname* con cui sono visibili.

Nell'instradamento, il path dell'applicazione (cioè la porzione dell'URL che viene dopo l'*hostname*) del "nome esterno" deve coincidere con il path del "nome interno".

La figura seguente rappresenta le regole suddette:



E' ammesso il passaggio di parametri nell'URL, se previsto dalla web application, mentre non è ammesso utilizzare i parametri per identificare la web application esposta.

Esempio di utilizzo di URL non è ammesso:
https:// www.<dominio>/entrypoint?applicazione=applicazione1

3.3.7 Accessibilità dell'applicazione tramite proxy

Si descrivono di seguito i vincoli che la web application dell'Aderente deve rispettare per poter essere esposta attraverso il Reverse Proxy (requisiti di *proxability*).
La web application da esporre non deve contenere riferimenti assoluti alle proprie risorse, ma solo puntamenti relativi.
In altri termini le eventuali risorse referenziate all'interno dell'applicazione (quali., ad esempio, immagini o link ad altre pagine) devono essere indirizzate tramite **URL relativi**, ovvero URL in cui viene esplicitata solamente la componente *path* senza le componenti *protocollo* ed *hostname*. Oltre agli URL anche **i PATH devono essere relativi**, ovvero non devono iniziare con il carattere "***/***".

Ad esempio:

URL ASSOLUTI (NON UTILIZZABILI)	URL RELATIVI (DA UTILIZZARE)
http(s)://hostname/logo.gif	logo.gif
http(s)://hostname/subdir1/index.html	subdir1/index.html **NOTA :** non è possibule utilizzare URL relativi del tipo **/**subdir1/index.html, i quali, pur essendo URL relativi (non vengono infatti indicati protocollo ed hostname), sono comunque PATH assoluti.

Inoltre, ogni singola Web Application deve prevedere un unico punto di ingresso da cui si diramano i diversi sottoservizi.
Non sono quindi consentiti collegamenti a sottoservizi non appartenenti all'albero che ha come radice la URL di ingresso della Web Application: ad esempio, supponendo che l'URL di "ingresso" della Web Application *sia* ***http(s)://hostnameX/directoryY***, non è consentito il collegamento a pagine che non risiedano sotto il path *directoryY* quali ***http(s)://hostnameX/directoryZ/pageJ.jsp*** (sempre che l'applicazione che ha come "ingresso" della WebApplication l'URL ***http(s)://hostnameX/directoryZ*** non sia stata esposta a sua volta).

3.3.8 Sicurezza

Ogni applicazione dovrà accettare solo le richieste pervenute dall'indirizzo IP del reverse proxy. Il controllo dovrà essere effettuato in tutti i casi a livello applicativo e, ove possibile, a livello sistemistico (con un firewall o sul web server). L'applicazione dovrà riconoscere l'utente, tramite i parametri passati nell'http Header, solo nel caso la richiesta gli venga inoltrata dal reverse proxy, in tutti gli altri casi l'applicazione non dovrà permettere l'accesso all'utente.

3.3.9 Gestione del logout applicativo

Nel caso in cui l'applicazione abbia creato una sessione al momento dell'autenticazione dell'utente, il logout dall'applicazione deve invalidare la sessione applicativa.
In ogni caso l'utente deve essere reindirizzato alla lista delle applicazioni al seguente link relativo:

- ./index.php

3.3.10 Definizione delle regole di autorizzazione

La regola implicita di autorizzazione degli accessi a una web application esposta sul dominio RER prevede la negazione di ogni accesso. Pertanto, ogni abilitazione deve essere espressamente dichiarata al sottosistema di controllo accessi tramite la formulazione di opportune regole.
Ogni singola regola prevede la specifica di:

1. il nome della risorsa oggetto della regola di abilitazione, dove per 'risorsa' si intende una web application o una sua porzione (sottoalbero o singola pagina);
2. l'elenco dei gruppi applicativi (nell'ambito di quelli definiti dalla RER) abilitati ad accedere alla risorsa di cui al punto precedente;
3. le operazioni ammesse (get e post).

Il nome della risorsa (*Resource Name*) è espresso mediante una *regular expression* che può contenere *wildcards*.

Esempi di nomi risorsa:

https:// www.<dominio>/applicazione1/*	Tutta l'applicazione: 'applicazione1'
https://www.<dominio>/applicazione1/home.htm	La sola pagina 'home.htm' dell'applicazione 'applicazione1'
https://www.<dominio>/applicazione1/consult/*	Tutte le risorse all'interno del sottoalbero 'consult' dell'applicazione 'applicazione1'

Ogni volta che un utente accede ad una *web application* esposta sul dominio RER tramite il *Reverse Proxy*, le URL delle pagine chiamate vengono confrontate con questa *regular expression*. In caso di corrispondenza, viene consentito l'accesso solo se il gruppo dell'originatore della richiesta è stato abilitato per la risorsa identificata e se l'operazione (get o post) è ammessa.
Nel caso vi siano applicazioni con redirezione implicita su una pagina di default è necessario aggiungere una nuova regola dedicata. (ad esempio https:// www.<dominio>/applicazione2).

4 Modulo di raccolta informazioni

Concordemente con quanto espresso sopra, al fine di impostare le configurazioni del *Reverse Proxy*, le regole di autorizzazione, ecc., è necessario per il sottosistema di *Access Management* conosca un insieme di informazioni che dovranno essere fornite dai responsabili applicativi, che verranno raccolte mediante il modulo riportato di seguito.
Sulla base dei dati in esso raccolti verranno configurati il *reverse proxy* (interno ed esterno) e le regole di autorizzazione centralizzate del sistema di *Access Management*.

DATO	VALORE
Nome applicativo	
Referente tecnico	
Path relativo	
Path assoluto	
Note	

Indicazioni per la compilazione:

- **Nome applicativo:** nome dell'applicativo per esteso (esempio: Vetrina della sostenibilità)
- **Referente tecnico:** riferimenti della persona indicata come referente tecnico per l'integrazione delle web application (esempio: Marco Rossi, telefono 051-999999, mail xxx@xx.xx)
- **Path relativo:** nome breve dell'applicativo, utilizzato per formare l'URL di accesso alla web application sul portale delle applicazioni[1] (esempio: VetrinaSostenibilita)
- **Path assoluto:** URL completo della web application da mappare sull'Access Management (esempio: http://wwwservizi.regione.emilia-romagna.it/VetrinaSostenibilita).
- **Note:** eventuali note

5 Esempi di utilizzo degli attributi dell'http Header

Esempio di JSP:

```
<html>
<head><title>Recupero Username e Dominio </title></head>
<body>
<%
String userid = request.getHeader("username");
String gruppo = request.getHeader("domain");

out.println("<BR> UserID  ->  " + username);
out.println("<BR> Dominio  ->  " + domain);
```

[1] https://applicazioni.regione.emilia-romagna.it

```
%>
</body>
</html>
```

Esempio di PERL:

```
#!/usr/bin/env perl
#
# Programma di test
#

print "Content-type: text/html\n\n" ;
print "<html><head><title>Recupero Username e Dominio></title></head>\n";

print "<table border=1 bordercolor=black>";
foreach $e (%ENV)
{
	if ($e eq "HTTP_USERNAME" || $e eq "HTTP_DOMAIN")
	{
		print "<tr>";
		print "<td>$e </td><td>$ENV{$e}</td>";
		print "</tr>";
	}
}
print "</table></body></html>\n";
```

Esempio di ASP.NET:

```
string username = string.Format(@"{0}\{1}",
                    Request.Headers["Domain"],
                    Request.Headers["Username"]
                  );

Response.Write(username);
```

Esempio di ASP:

```
username = Request.ServerVariables("HTTP_Domain") & "\" &
           Request.ServerVariables("HTTP_Username")

Response.Write username
```

Sistema di autenticazione centralizzata custom

Questo sezione ha lo scopo di descrivere la soluzione custom per implementare l'autenticazione nelle applicazioni ospitate sulle infrastrutture della Regione Emilia-Romagna (in seguito applicazioni *client*). In particolare sono presenti due domini "Active Directory", uno contenente gli account degli utenti regionali[2] (dominio intranet) e l'altro contenente account di utenti esterni alla regione (dominio extranet), e un sistema centralizzato di autenticazione (utilizzabile per entrambi i domini).

Ci sono due possibilità per sfruttare i servizi di questo sistema centralizzato:

- appoggiarsi per il login e la modifica della password all'applicazione web di autenticazione centralizzata disponibile all'url https://wwwservizi.regione.emilia-romagna.it/AutenticazioneCentralizzata (l'integrazione con l'applicazione *client* avviene tramite "browser redirection")
- richiamare i servizi web che implementano i metodi necessari per l'autenticazione disponibili all'url https://wwwservizi.regione.emilia-romagna.it/WebServices/AutenticazioneCentralizzata/Authentication.asmx

NB: in entrambi i casi ci si occupa solo dell'autenticazione dell'utente e non delle autorizzazioni (ad esempio profilature degli utenti, etc…) che rimangono a carico dell'applicazione *client*.
L'autenticazione consiste nella verifica dell'esistenza dell'utente nei domini "Active Directory" e nella verifica dell'appartenenza ad un gruppo di dominio definito a priori per ciascuna applicazione *client*.
Visto che le credenziali d'accesso sono gestite completamente dal sistema di autenticazione centralizzata, è necessario eliminare dal database dell'applicazione *client* ogni campo che contiene password.
Deve invece rimanere ogni campo relativo allo username, al fine di associare gli eventuali profili autorizzativi. Tali campi possono essere al massimo di 40 caratteri e devono contenere sia il dominio che il nome account separati dal backslash (ad esempio EXTRARER\sempronio.tizio per gli utenti esterni e RERSDM\Cognome_N per gli utenti regionali)

Per garantire che il servizio di autenticazione sia usato solo da chi ne ha diritto e per garantire a chi lo usa che sia proprio il servizio di autenticazione della Regione a rispondere, si sfrutta il sistema del *salted hash*. Ad ogni applicazione *client*, i gestori del sistema assegnano un identificativo univoco dell'applicazione (in seguito *IDApplicazione*) ed un codice segreto (il *salt*) noto solo all'applicazione *client* e al sistema di autenticazione. Il chiamante aggiunge al messaggio il *salted hash* (ovvero l'hash della concatenazione tra messaggio e salt). Il ricevente ricalcola il *salted hash* del messaggio e lo confronta con quello ricevuto. Se sono uguali, allora è garantita l'autenticità del mittente e l'integrità del messaggio.
L'algoritmo di hashing scelto è l'MD5.

Riassumendo, per poter sfruttare i servizi del sistema centralizzato è necessario farne richiesta al Servizio SIIR (Sistema informativo-informatico regionale) che provvederà a:

- fornire l'identificativo univoco dell'applicazione *client,*
- fornire il codice segreto dell'applicazione *client,*
- creare i gruppi di dominio (rersdm e extrarer) associati all'applicazione client, illustrando le modalità operative di gestione degli utenti[3] nell'Active Directory.

Nella richiesta è necessario indicare il referente regionale dell'applicazione e, nel caso in cui ci si appoggi per il login all'applicazione web di autenticazione centralizzata, è necessario fornire la url della pagina dell'applicazione client (in seguito pagina definita a priori) a cui l'applicazione di autenticazione centralizzata reindirizza il browser dopo aver autenticato l'utente.

[2] Si intendono "regionali" anche i collaboratori, consulenti, ecc…
[3] Almeno fino a quando non ci sarà la possibilità di crearli in modo indipendente mediante opportuna interfaccia

6 Redirect all'applicativo web generalizzato di autenticazione

Nelle applicazioni client per le quali non importa una propria pagina di autenticazione, è possibile effettuare un redirect all'applicativo web generalizzato di autenticazione disponibile all'url https://wwwservizi.regione.emilia-romagna.it/AutenticazioneCentralizzata/Default.aspx.

Si visualizza una maschera standard di login in cui si deve specificare il tipo di utente (dominio), lo username e la password (alternativamente vi è un apposito link per utenti già autenticati sul dominio, ovvero che hanno fatto logon sulla propria workstation con le credenziali regionali).

I parametri da passare, nella querystring, alla pagina Default.aspx sono:

- *IDApplicazione* (stringa) (obbligatorio)
- *dominio* (stringa) (facoltativo): se valorizzato, si imposta automaticamente nel form il corrispondente campo. Valori possibili: "RERSDM", "EXTRARER" a seconda che l'utente sia regionale o esterno. **NB:** se il dominio è vuoto l'applicativo cerca di estrapolarlo dallo username assumendo la sintassi ***nomedominio\nomeutente***.
- *username* (stringa) (facoltativo): se valorizzato, si imposta automaticamente nel form il corrispondente campo
- *customInfo* (stringa) (facoltativo): verrà restituita dall'applicazione di autenticazione centralizzata alla *pagina definita a priori* così come gli è stata inviata. E' a disposizione dell'applicazione *client* per tenere traccia di qualsiasi tipo di informazione durante la fase di autenticazione (che è appunto esterna all'applicazione *client* stessa). Ad esempio: se l'applicazione *client* vuole che a fronte dell'avvenuta autenticazione, il controllo ritorni alla pagina che era stata richiesta inizialmente dall'utente, si può mettere in *customInfo* l'URL della pagina da richiamare. Naturalmente il redirect a questa pagina è a carico della *pagina definita a priori*.
- *saltedHash* (stringa) (obbligatorio)

L'applicazione di autenticazione generalizzata, dopo aver controllato l'integrità del messaggio, tenta di verificare le credenziali specificate nel form.
Se le credenziali sono scorrette, l'account è scaduto, bloccato o disabilitato, allora si visualizza un messaggio di login non riuscito.
Se le credenziali sono corrette, l'utente è autorizzato ad accedere all'applicazione, non è il primo accesso e la password non è scaduta, allora si eseguirà redirect alla *pagina definita a priori* indicandogli:

- *username* (stringa) dell'utente autenticato (nel formato ***nomedominio\nomeutente*)**.
- *validity* (stringa): data e ora massima di validità della richiesta, ovvero una data/ora oltre la quale non è più attendibile, per la pagina richiamata dall'applicativo centralizzato, l'informazione che l'utente è stato autenticato (ciò per tutelarsi di sniffing dell'url richiamato). Il formato di *validity è* AAAAMMGGhhmmss[4]
- *customInfo* (stringa)
- *saltedHash* (stringa)

La pagina dell'applicazione *client* invocata dal servizio centralizzato riceve i parametri sopra indicati. Dopo aver verificato l'integrità del messaggio e la validità della richiesta[5] potrà operare normalmente (come quando aveva in locale la pagine di richiesta delle credenziali: potrà quindi creare il proprio cookie di autenticazione o token di autenticazione, ecc…).

Se invece le credenziali sono corrette e l'utente è autorizzato ad accedere all'applicazione, ma è il primo accesso oppure la password è scaduta, allora si rimanda al form di modifica password: https://wwwservizi.regione.emilia-romagna.it/AutenticazioneCentralizzata/ModificaPassword.aspx (che

[4] AAAA: anno a quattro cifre; MM, GG, hh, mm, ss: rispettivamente mese, giorno, ora, minuti secondi a 2 cifre
[5] Controllare che la data/ora attuale sia inferiore o uguale alla data di validità ricevuta (*validity*)

è comunque sempre richiamabile, da chiunque e in qualunque momento, per cambiare la password) pre-impostando il dominio e lo username. Dopo la modifica della password si procede come nel caso precedente.

NB: nel caso si dovessero avere difficoltà a calcolare l'hash MD5 è disponibile un web service che lo fa (vedere sotto).

6.1 Modifica della password

È sempre possibile richiamare la pagina di modifica della password (da qualsiasi applicazione o sito, indipendentemente che si tratti di una delle applicazioni che sfruttano l'autenticazione centralizzata. L'url della pagina da richiamare è:

https://wwwservizi.regione.emilia-romagna.it/AutenticazioneCentralizzata/ModificaPassword.aspx

e i parametri da passare sono:

- *dominio* (stringa) (facoltativo): se valorizzato, si imposta automaticamente nel form il corrispondente campo. Valori possibili: "RERSDM", "EXTRARER" a seconda che l'utente sia regionale o esterno. **NB:** se il dominio è vuoto l'applicativo cerca di estrapolarlo dallo username assumendo la sintassi ***nomedominio\nomeutente***.
- *username* (stringa) (facoltativo): se valorizzato, si imposta automaticamente nel form il corrispondente campo
- *tornaA* (stringa) (facoltativo): se valorizzato, nella pagina di modica password viene creato un link con testo "Torna alla pagina precedente" che punta alla pagina indicata nel parametro *tornaA*

7 Chiamata dei servizi web di Autenticazione

Le applicazioni web che vogliono gestire l'autenticazione localmente (senza appoggiarsi all'applicativo centralizzato, su cui non si possono eseguire personalizzazioni di grafica o altro), possono richiamare i seguenti servizi web:

- *Authentication* che espone i metodi *Login*, *LoginIntranet*, *CambioPassword* disponibile all'url https://wwwservizi.regione.emilia-romagna.it/WebServices/AutenticazioneCentralizzata/Authentication.asmx
- *HashHelper* che espone i metodi *ComputeHash*, *ComputeSaltedHash* disponibile all'url https://wwwservizi.regione.emilia-romagna.it/WebServices/AutenticazioneCentralizzata/HashHelper.asmx

NB: le applicazioni *client* che richiamano tali servizi web devono utilizzare il protocollo *https*.

7.1 Authentication, metodo "Login"

Consente di verificare se un certo utente può accedere ad una certa applicazione.
Riceve in input i parametri:

- *IDApplicazione* (stringa)
- *Dominio* (stringa), valori possibili: "RERSDM", "EXTRARER" a seconda che l'utente sia regionale o esterno
- *Username* (stringa)
- *Password* (stringa)
- *SaltedHash* (stringa)

Restituisce in output uno dei seguenti valori:

- NonRiuscito

- Riuscito
- LoginFallito
- PasswordScaduta
- AccountDisabilitato
- PasswordDaCambiare
- AccountBloccato
- AccountScaduto

7.2 Authentication, metodo "LoginIntranet"

Consente di verificare se un utente, autenticato sul dominio, può accedere ad una certa applicazione.
Riceve in input i parametri:

- *IDApplicazione* (stringa)
- *Dominio* (stringa), valori possibili: "RERSDM", "EXTRARER" a seconda del dominio su cui l'utente si è loggato
- *Username* (stringa)
- *SaltedHash* (stringa)

Restituisce in output uno dei seguenti valori:

- NonRiuscito
- Riuscito
- LoginFallito
- PasswordScaduta
- AccountDisabilitato
- PasswordDaCambiare
- AccountBloccato
- AccountScaduto

7.3 Authentication, metodo "CambioPassword"

Consente di modificare la password di un utente.
Riceve in input i parametri:

- *IDApplicazione* (stringa)
- *Dominio* (stringa), valori possibili: "RERSDM", "EXTRARER" a seconda che l'utente sia regionale o esterno
- *Username* (stringa)
- *OldPassword* (stringa)
- *NewPassword* (stringa)
- *SaltedHash* (stringa)

Restituisce in output uno dei seguenti valori:

- NonRiuscito
- Riuscito
- VecchiaPasswordSbagliata
- PasswordNonSoddisfaIRequisiti
- AccountRestriction
- PasswordNonModificabile
- AccountBloccato
- AccountScaduto

7.4 HashHelper, metodo ComputeHash

Consente di calcolare l'hash in formato MD5 di una stringa ricevuta in input.

Riceve in input il parametro

- *value* (stringa)

Restituisce in output l'hash di *value.*

7.5 HashHelper, metodo ComputeSaltedHash

Consente di calcolare l'hash di una sequenza di parametri salati con un codice privato.

Riceve in input il parametro
- *salt* (stringa), il codice privato
- *parametri (*stringa), messaggio da "salare"

Restituisce in output il *salted hash*, ovvero l'hash della stringa ottenuta concatenando i vari parametri e mettendo in coda il *salt.*

8 Libreria di utilità per applicazioni ASP

In caso si sviluppi in linguaggio ASP è possibile sfruttare una "libreria" di funzioni, definite in una pagina ASP, che rendono molto semplici l'utilizzo dei servizi dell'infrastruttura dell'autenticazione centralizzata.
Tali funzioni utilizzano implicitamente i valori delle variabili *RCRER_IDApplicazione* e *RCRER_codicePrivato* che vanno pertanto impostate (ma non dichiarate, sono dichiarate nella libreria) nella specifica applicazione *client* e sono rispettivamente l'identificativo univoco e il codice segreto dell'applicazione *client*.

La pagina ASP *AutenticazioneCentralizzata.asp* (che si trova nella directory virutale "/includes" dei server web presenti in Regione) va inclusa nelle applicazioni che intendono utilizzare le funzioni in essa definite. Le funzioni disponibili sono elencate qui sotto.

8.1 Funzioni utili in caso di redirect all'Autenticazione Centralizzata

VaiALoginCentralizzatoConCustomInfo (customInfo) / VaiALoginCentralizzato

Esegue il redirect all'url restituito da HRefLoginCentralizzato(customInfo).
Vedi "HRefLoginCentralizzato (customInfo)"

Utilizzare una o l'altra a seconda che sia necessario o meno tenere traccia di informazioni.

VaiALoginCentralizzatoConCustomInfo (customInfo)

Esegue il redirect all'url restituito da HRefLoginCentralizzato(customInfo).
vedi "HRefLoginCentralizzato (customInfo)"

HRefLoginCentralizzato (customInfo)

Restituisce l'url su cui ridirezionare l'applicazione client per il login.
Tale url è comprensivo dei parametri che si aspetta il sistema di autenticazione centralizzata, ovvero
- *IDApplicazione*
- *customInfo*
- *saltedHash*

Vedi "Redirect all'applicativo web generalizzato di autenticazione"

MessaggioUtenteNonAutorizzatoConCustomInfo (customInfo) / MessaggioUtenteNonAutorizzato

Restituisce il messaggio da visualizzare nel caso in cui l'utente sia stato autenticato ma non disponga della autorizzazione necessaria per accedere (ad esempio se nel database non è presente alcun utente *username*). La stringa restituita è comprensiva anche del link al *Login*.

Utilizzare una o l'altra a seconda che sia necessario o meno tenere traccia di alcune informazioni.

IsSalaturaOK (byref username)

Si recuperano i parametri che l'applicazione centralizzata ha passato all'applicazione *client*. Si ricalcola il *salted hash* in base ai parametri ricevuti e lo si confronta con quello ricevuto. Se sono uguali, la funzione restituisce true altrimenti si visualizza "Messaggio Corrotto" e la funzione restituisce false.

IsLatenzaOK

Si controlla che la data attuale non sia superiore a *validity*. La funzione restituisce true o false a seconda che la richiesta sia ancora valida o meno. In particolare se la richiesta non è valida, allora si ridireziona il browser alla pagina di login.

IsAccessoOK (byref username)

Restituisce true se la salatura è corretta (*IsSalaturaOK*) e la richiesta è ancora valida (*IsLatenzaOK*). Se una delle due condizioni non è verificata restituisce false.

HashHelper_ComputeHash (value)

Restituisce l'hash di *value*.

HashHelper_ComputeSaltedHash(salt, parametri)

Restituisce il saltedhash della stringa *parametri* salato con il codice privato *salt*

8.2 Funzioni utili in caso di richiamo dei servizi Web

Authentication_Login (domain, username, password)

Calcola il salted hash dei parametri ricevuti (*HashHelper_ComputeSaltedHash*) e richiama il metodo *Login* del servizio web di *Authentication*. Restituisce l'esito del login (vedi "Authentication, metodo "Login"").

Authentication_LoginIntranet (domain, username)

Calcola il salted hash dei parametri ricevuti (*HashHelper_ComputeSaltedHash*) e richiama il metodo *LoginIntranet* del servizio web di *Authentication*. Restituisce l'esito del login (vedi "Authentication, metodo "LoginIntranet"")

IsLoginRiuscito (esitoLogin)

Restituisce true o false se *esitoLogin* è uguale a Riuscito

IsLoginPasswordScaduta (esitoLogin)

Restituisce true o false se *esitoLogin* è uguale a PasswordScaduta

IsLoginPasswordDaCambiare (esitoLogin)

Restituisce true o false se *esitoLogi*n è uguale a PasswordDaCambiare

Authentication_CambioPassword (domain, username, oldPassword, newPassword)

Richiama il metodo *CambioPassword* del servizio web di *Authentication* e restituisce l'esito del cambio password (vedi "Authentication, metodo "CambioPassword"").

IsCambioPasswordRiuscito (esitoCambioPassword)

Restituisce true o false se *esitoCambioPassword* è uguale a Riuscito

IsCambioPasswordVecchiaPasswordSbagliata (esitoCambioPassword)

Restituisce true o false se *esitoCambioPassword* è uguale a VecchiaPasswordSbagliata.

IsCambioPasswordPasswordNonSoddisfaRequisiti (esitoCambioPassword)

Restituisce true o false se *esitoCambioPassword* è uguale a PasswordNonSoddisfaRequisiti.

8.3 Esempio di utilizzo delle funzioni in caso di Redirect all'autenticazione centralizzata

Supponiamo che lo scenario dell'applicazione *client* sia il seguente:

- *Default.ASP*: è la prima pagina caricata (per intenderci quella che normalmente conterrebbe il form di login)
- *Login.ASP*: è la pagina che viene richiamata dall'applicazione di autenticazione centralizzata (per intenderci quella che sarebbe l'action del form di login della *default*.ASP)

Entrambe includono la pagina *Global.ASP* che contiene le funzioni di utilità comuni all'applicazione.

Per utilizzare le funzioni descritte nel paragrafo "Funzioni utili in caso di redirect all'Autenticazione Centralizzata", occorre procedere, per ciascuna pagina, nel seguente modo.

Global.ASP

Includere il file /includes/AutenticazioneCentralizzata.asp e valorizzare le variabili *RCRER_IDApplicazione* e *RCRER_codicePrivato*.
In pratica inserire le seguenti istruzioni:

```
<!-- include virtual="/includes/AutenticazioneCentralizzata.ASP" -- >
<%
RCRER_IDApplicazione = "ApplicazioneX"
RCRER_codicePrivato = "swu55kdjfk16grte"
%>
```

Default.asp

Ridirezionare il browser all'applicazione centralizzata richiamando *VaiALoginCentralizzato*.
In pratica le istruzioni della pagina diventerebbero:

```
<!- - include file="global.ASP" - - >
<%
VaiALoginCentralizzato
%>
```

In questo caso particolare il browser richiamerebbe il seguente url:

http://.....?IDApplicazione=ApplicazioneX&customInfo=&saltedHash=jkquF+X6+c29hoJC04lGDw==[6]

NB: nel caso in cui si voglia tenere traccia di alcune informazioni, utilizzare VaiALoginCentralizzatoConCustomInfo passando le informazioni necessarie. Tali informazioni

[6] È il salted hash dei parametri passati nel querystring (in questo caso della stringa "*ApplicazioneX*" - unico parametro non vuoto) salati con il codice privato dell'applicazione

saranno restituite, così come sono state passate, alla pagina Login.ASP (vedi sotto).
Ad esempio, volendo passare il nome di una pagina asp a cui la Login.ASP dovrebbe ridirigersi, si avrebbe:

```
<!- - include file="global.ASP" - - >
<%
VaiALoginCentralizzato "torna_a=menu.asp"
%>
```

e in questo caso particolare il browser richiamerebbe il seguente url:

http://.....?IDApplicazione=ApplicazioneX&customInfo=torna_a=menu.asp&saltedHash=lFr1E+65zr6uneh7hhXCPg==[7]

Login.asp

Definire la variabile *username* se già non è stata definita nel global.ASP
Verificare che la salatura e che il periodo di validità siano corretti richiamando *IsAccessoOK* passando *username* che è passato per riferimento e impostato dalla chiamata stessa.

In pratica le istruzioni della pagina diventerebbero:

```
<!- - include file="global.ASP" - - >
<%
Dim username
If IsAccessoOK (username) Then
  'qui ci va quello che si farebbe normalmente:
    'creazione del cookie di autenticazione,
      'impostazione delle variabili di sessione,
        'individuazione del profilo, redirect al menu, etc….

          'In particolare se nel database non è presente alcun
  'utente username, è possibile richiamare
    'MessaggioUtenteNonAutorizzato
End If
%>
```

N.B. In questo costrutto *If – Then* non è presente l'*Else* perchè è la funzione *IsAccessoOK* che si occupa di visualizzare un messaggio o ridirezionare alla pagina di login se il messaggio era corrotto o se la richiesta non era più valida.

8.4 Esempio di utilizzo delle funzioni in caso di richiamo dei servizi web

Supponiamo che lo scenario dell'applicazione *client* sia il seguente:

- *Default.ASP*: è la prima pagina caricata, quella che contiene il form di login
- *Login.ASP*: è l'action di *Default.ASP*

Entrambe includono la pagina *Global.ASP* che contiene le funzioni di utilità comuni all'applicazione.

Per utilizzare le funzioni descritte nel paragrafo Funzioni utili in caso di richiamo dei servizi Web, occorre procedere, per ciascuna pagina, nel seguente modo.

Global.asp

Includere il file e valorizzare le variabili *RCRER_IDApplicazione* e *RCRER_codicePrivato*.

[7] È il salted hash dei parametri passati nel querystring (in questo caso della stringa "*ApplicazioneXtorna_a=menu.asp"*) salati con il codice privato dell'applicazione

In pratica inserire la seguente inclusione di file:

```
<!- - include virtual="/includes/AutenticazioneCentralizzata.ASP" - - >
<%
RCRER_IDApplicazione = "ApplicazioneX"
RCRER_codicePrivato = "swu55kdjfk16grte"
%>
```

Default.ASP

Rimane inalterata.

Login.ASP

Richiamare la funzione *authentication_Login* passando quanto impostato nel form di login.

Se il login è riuscito (*IsLoginRiuscito* è true) e la password è da cambiare (*IsLoginPasswordScaduta* e *IsLoginPasswordDaCambiare* sono true), allora richiamare la pagina per modificare la password.

Diversamente procedere normalmente a seconda che il login sia riuscito (senza necessità di cambiare la password) oppure non riuscito.

In pratica le istruzioni diventerebbero:

```
<!- - include file="global.ASP" - - >
<%
Dim Dominio, Username, Password
Dim esitoLogin
Dominio  = Trim(Request.Form("Dominio"))
Username = Trim(Request.Form("Username"))
Password = Trim(Request.Form("Password"))

esitoLogin = authentication_Login (dominio, username, password)
If IsLoginRiuscito (esitoLogin) Then
     If IsLoginPasswordScaduta(esitoLogin) or
      IsLoginPasswordDaCambiare(esitoLogin) Then
        'qui ci va il redirect alla pagina per modificare la password

   Else
        'qui ci va quello che si farebbe normalmente:
      'creazione del cookie di autenticazione,
      'impostazione delle variabili di sessione,
      'individuazione del profilo, redirect al menu, etc….
End If
Else
   'qui si deve visualizzare il messaggio di login non riuscito
End If
%>
```

9 Classe di utilità per applicazioni ASP.NET

Analogamente a quanto fatto per le pagine ASP è disponibile una classe che rende molto semplici l'utilizzo dei servizi dell'infrastruttura dell'autenticazione centralizzata nel caso di applicazioni web

ASP.NET che utilizzano la Forms Authentication. Per quanto riguarda i webservices non è stato definito niente in quanto con .NET già sufficiente quello che fornisce il framework per il "consumo" di servizi web.

La classe è definita nell'assembly RER.Tools il suo nome è:

RER.Tools.AutenticazioneCentralizzataHelper

Tale classe ha due costruttori a cui passare l'IDApplicazione e il codice privato (che rappresentano rispettivamente l'identificativo univoco e il codice segreto dell'applicazione *client*). I due costruttori differiscono solo per il parametro customInfo che può essere specificato o meno.

Questi tre attributi: IDApplicazione, CodicePrivato e CustomInfo sono anche accessibili tramite omonime proprietà pubbliche dell'istanza della classe.

Sono poi definiti i seguenti metodi (da usare per invocare l'applicazione centralizzata):

- **VaiALogin(), VaiALogin(string customInfo)**: che esegue il redirect all'applicazione centralizzata di login.
- **UrlLogin(), UrlLogin(string customInfo)**: che restuisce l'url su cui ridirezionare l'applicazione client per il login.

Sono infine definite queste altre tre proprietà (da usare quando si deve interpretare l'esito del login, nella *pagina definita a priori*, per intendersi):

- **UtenteAutenticato**: invoca le due proprietà sottostanti e se tutto ok, restituisce l'utente autenticato. Se invece la "salatura" non è corretta lancia l'eccezione *RER.Tools.AutenticazioneCentralizzataHelper.MessaggioCorrottoException,* se invece è la "latenza" a non essere accettabile, esegue automaticamente un redirect alla pagina di login.
- **IsSalaturaOk**: si recuperano i parametri che l'applicazione centralizzata ha passato all'applicazione client. Si ricalcola il salted hash in base ai parametri ricevuti e lo si confronta con quello ricevuto. Se sono uguali, la funzione restituisce true altrimenti restituisce false.
- **IsLatenzaOk**: controlla che la data attuale non sia superiore a *validity*. La funzione restituisce true o false a seconda che la richiesta sia ancora valida o meno.

Per comodità la classe espone anche una proprietà *static* che contiene l'url per la pagina di modifica password.

9.1 Esempio di utilizzo

Definire nella classe Global (del global.asax) le due costanti IDApplicazione e CodicePrivato:

```
public class Global : System.Web.HttpApplication
{

        public const string IDApplicazione = "yyyyyyyyyyyyyyy";
        public const string CodicePrivato = "xxxxxxxxxxxxxx";

        // ...
}
```

Nella pagina di Login (quella indicata nella configurazione della Forms Authentication) adattare il seguente codice per l'evento Page_Load nel caso IsPostBack sia falso (il caso in cui è true non si verifica mai, in quanto la pagina di login non ha una interfaccia propria):

```
private void Page_Load(object sender, System.EventArgs e)
{
   if (!IsPostBack)
   {
      AutenticazioneCentralizzataHelper autenticazioneCentralizzataHelper =
         new AutenticazioneCentralizzataHelper(
            Global.IDApplicazione, Global.CodicePrivato);

     if (Request.QueryString["Username"] == null)
          autenticazioneCentralizzataHelper.VaiALogin();
     else
     {
          string username = autenticazioneCentralizzataHelper.UtenteAutenticato;

          // qui mettere il codice per la creazione dell cookie di
          // autenticazione e eventuale gestione dell'autorizzazione
     }
   }
}
```