SardegnaIT
Società in house per lo sviluppo
e la gestione dei progetti e dei servizi
di e-government

REGIONE AUTÒNOMA DE SARDIGNA
REGIONE AUTONOMA DELLA SARDEGNA

[IdM-RAS]

Sistema per la gestione delle identità digitali della Regione Sardegna

Identity Management System – IdM RAS

Manuale di integrazione

Data documento:	19.03.2013
File:	Allegato 1 - Modalita di integrazione sistema di Identity Management IDM RAS.doc
Tipo documento:	Manuale tecnico per l'integrazione del sistema
Versione:	01.00

Redazione:	Sardegna IT
Rilascio:	Sardegna IT
Divulgazione:	Documento pubblico

[IdM-RAS]
Sistema per la gestione delle identità digitali della Regione Sardegna
Identity Management System – IdM RAS

Manuale di integrazione

Autori

Autori del documento	*Struttura di riferimento*	*Parte del documento*	*Note*
Sardegna IT	Servizio per lo Sviluppo del Software	Tutte	-

Storia delle modifiche

Vers.Rev	*Data*	*Autore*	*Descrizione modifiche*
1.0	19.09.2013	SardegnaIT	Rilascio documento versione consolidata

Acronimi

Acronimo	*Definizione*
IdM-RAS	Piattaforma di Identity management della Regione Sardegna
CNS	Carta Nazionale dei servizi
IdP	Identity Provider
SAML	Security Assertion Markup Language

Sommario

1. Documenti di riferimento

[1] Progetto ICAR – Interoperabilità e Cooperazione Applicativa tra le Regioni. http://www.progettoicar.it

[2] OASIS Security Assertion Markup Language 2.0 - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[3] OASIS Security Services (SAML) TC, Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 marzo 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[4] OASIS Security Services (SAML) TC, Metadata for the OASIS Security Assertion Markup Language (SAML)V2.0, OASIS Standard, 15 marzo 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[5] Internet2 OpenSAML 2.0 - https://spaces.internet2.edu/display/OpenSAML/Home

[6] Progetto ICAR, Sistema Federato Interregionale di Autenticazione: Modello Architetturale di Riferimento v1.0, 3 Aprile 2007. http://www.progettoicar.it/GetMedia.aspx?id=04c341a0-00e6-4b9c-9fbe-4fddcbda07be&s=0&at=1

[7] Progetto ICAR, Sistema Federato Interregionale di Autenticazione: Specifica Interfacce Interne e Progettazione Reference Implementation, 11 Maggio 2008.

2. Introduzione

2.1. Scopi e Contenuti

Il presente documento ha come scopo quello di illustrare brevemente il sistema di autenticazione della Regione Autonoma Sardegna (d'ora in poi IdM-RAS) e le modalità da seguire per realizzare l'interfacciamento con un generico sistema informativo regionale.

3. Architettura del sistema

L'architettura di alto livello di IdM-RAS è raffigurata nella seguente Figura 3-1, nella quale sono evidenziati in tre box a contorno grigio i tre domini logici afferenti rispettivamente ad un utente del sistema che intende accedere ad uno dei servizi erogati da un determinato fornitore nell'ambito di RAS, il fornitore identificato come un generico sistema informativo di RAS ed infine l'infrastruttura di autenticazione vera e propria.

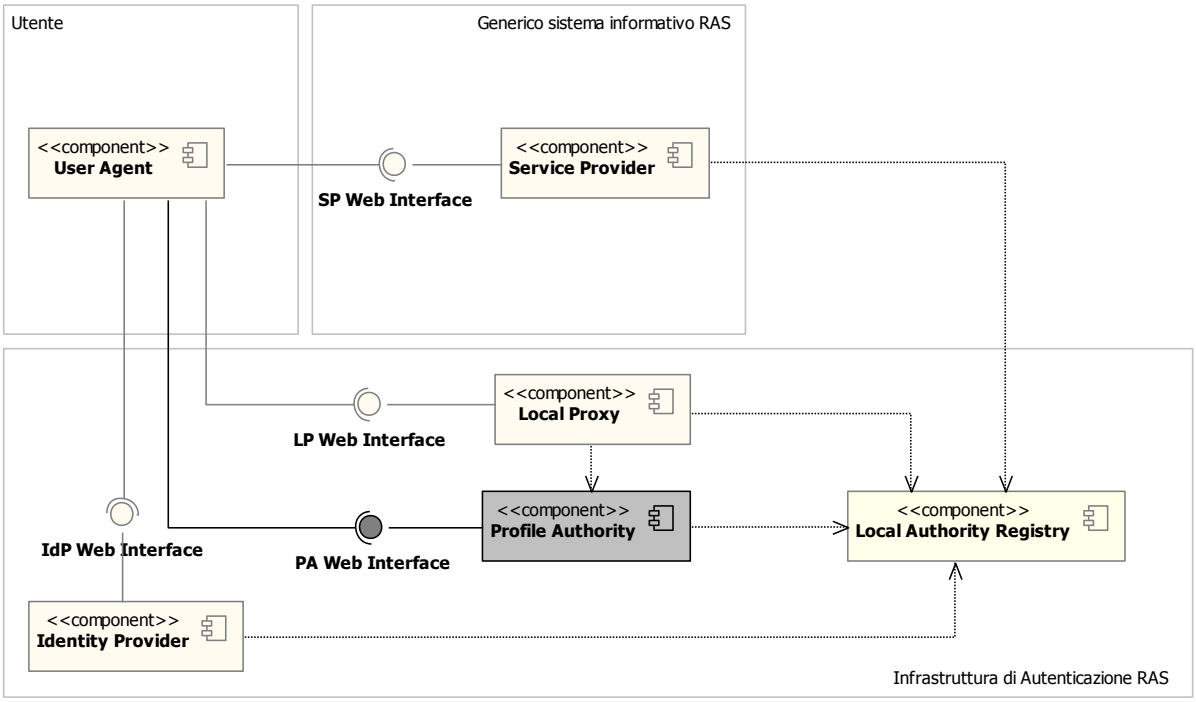

Figura 3-1 : Diagramma architetturale di alto livello dell'infrastruttura di autenticazione RAS

Si assume che l'utente operi mediante un qualunque user agent di tipo web browser, abilitato cioè ad effettuare richieste HTTP e HTTPS a risorse in rete. Infatti, i tre soggetti principali che operano nei domini di erogazione dei servizi e di autenticazione, cioè il Service Provider, il Local Proxy e l'Identity Provider, espongono ciascuno un'interfaccia web per ricevere richieste e fornire le relative risposte.

3.1. Service Provider

Scendendo ad un livello di dettaglio maggiore, con il termine "Service Provider" si intende identificare un certo sistema informativo di RAS che è stato integrato con l'infrastruttura di autenticazione. Tale integrazione è fatta mediante l'aggiunta di due componenti denominati "SP Request Handler" e "Assertion Consumer Service" al sistema informativo in oggetto. L'architettura interna di un Service Provider integrato con IdM-RAS è rappresentata in Figura 3-2. Si noti che il sistema informativo con tutti i relativi servizi erogati viene indicato in figura con l'accezione "Servizio Applicativo".

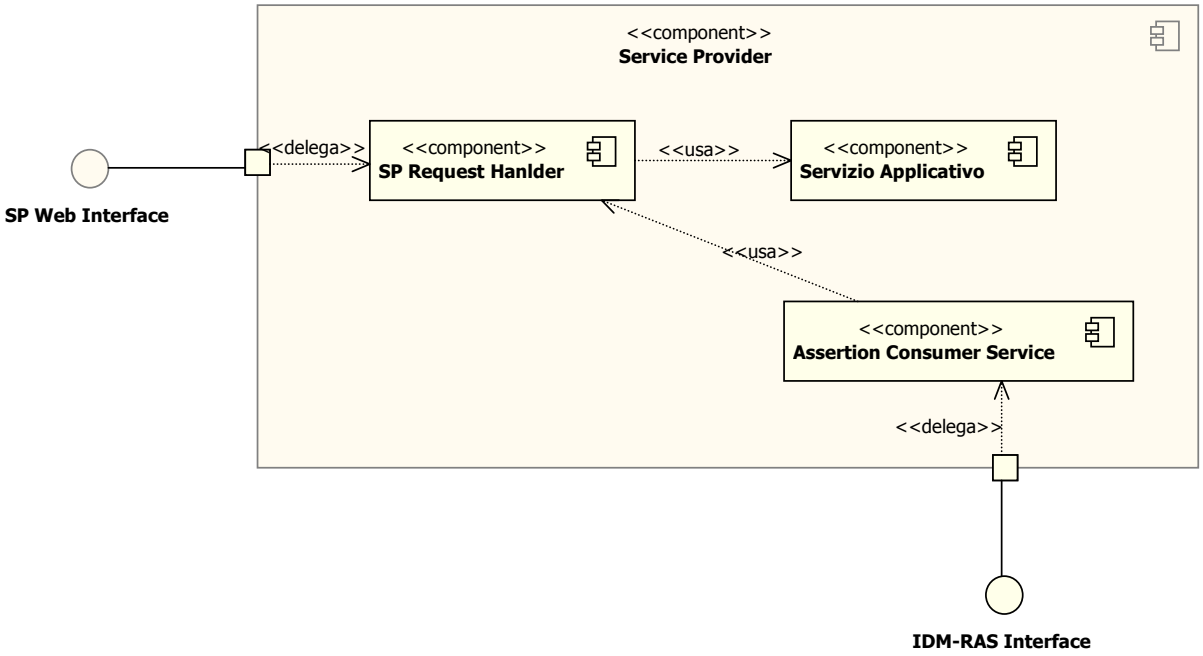

Figura 3-2 - Architettura del Service Provider

Il componente SP Request Handler consiste in un filtro in grado di intercettare le richieste su protocollo HTTP(S) in arrivo da parte del browser dell'utente e dirette ad uno dei servizi erogati dal Service Provider con il quale è integrato e attivare il processo di autenticazione mediante dialogo con IdM-RAS, se necessario. Il componente Assertion Consumer Service è invece preposto alla ricezione dei messaggi di risposta provenienti da IdM-RAS al termine del processo di autenticazione.

3.2. Local proxy

Il Local Proxy ha anzitutto il ruolo di ricevente delle richieste di autenticazione provenienti dai vari Service Provider integrati con l'infrastruttura di autenticazione. Esso rappresenta infatti, per tutti i Service Provider ad esso afferenti, il gateway verso il sistema di autenticazione. Come tale, esamina le richieste di autenticazione arrivate, controllando la firma digitale apposta e operando altri controlli logici. Il Local Proxy propone quindi all'utente l'elenco degli Identity Provider disponibili (al momento uno solo, nell'infrastruttura di RAS, ma altri possono essere aggiunti in qualunque momento) tra i quali viene richiesto di scegliere quello desiderato ove effettuare l'autenticazione. Il Local Proxy infine rimanda l'utente all'Identity Provider scelto dal quale riceverà un messaggio XML contenente i dettagli sull'autenticazione avvenuta.

3.3. Profile Authority

Il componente Profile Authority, responsabile della gestione dei profili degli utenti e del dialogo con le Attribute Authority è stato rappresentato in grigio perché, nonostante faccia parte logicamente dell'infrastruttura, così come definita originariamente dal progetto ICAR, nell'attuale implementazione di IdM-RAS esso non è compreso e potrà essere aggiunto in un secondo momento, riconfigurando opportunamente gli altri componenti che dovranno interagire con esso (Local Proxy e Identity Provider). Questa riconfigurazione interna non andrà ad impattare con il funzionamento dei service provider già integrati.

3.4. Identity Provider

L'Identity Provider è il componente con il quale ciascun utente che accede ai servizi di un Service Provider integrato con l'infrastruttura di autenticazione si trova ad interagire durante la fase di autenticazione. L'Identity Provider è infatti l'unica entità che ha titolo a richiedere all'utente le proprie credenziali così da riconoscerlo ed emettere il corrispondente messaggio XML contenente un'asserzione di autenticazione con i dettagli sulle modalità con le quali essa è avvenuta. Al fine di interagire con l'utente, ciascun Identity Provider deve essere dotato di un'interfaccia HTTPS per stabilire un canale di comunicazione sicuro sul quale veicolare le credenziali.

Nell'infrastruttura IdM-RAS è presente un componente con funzioni di Identity Provider il quale offre, in aggiunta, la funzionalità di registrazione di nuovi utenti. Inoltre ciascun utente in possesso delle credenziali può accedere all'Identity Provider per modificare i dati relativi alla propria registrazione. La manutenzione delle credenziali è possibile anche da parte di un operatore, mediante una console di gestione separata, non fruibile da parte degli utenti finali per motivi di sicurezza. In Figura 3-3 è raffigurato il componente Identity Provider esploso nelle sue funzionalità logiche principali.

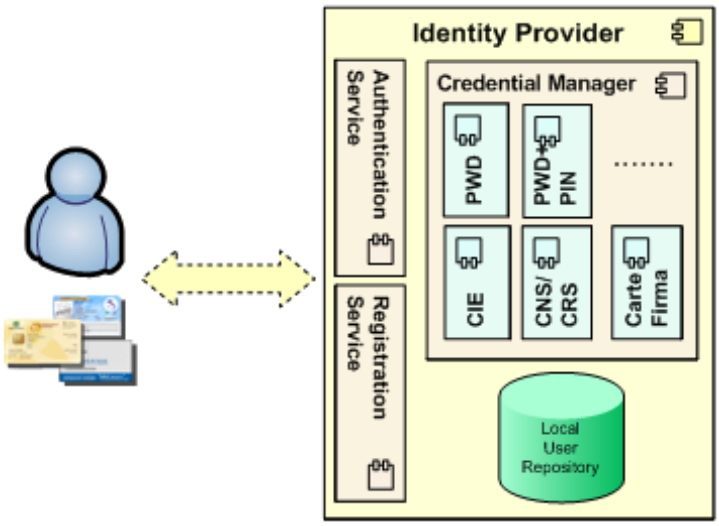

Figura 3-3 - Il componente di registrazione e autenticazione

Come è possibile vedere, le due funzionalità principali, quella di autenticazione e quella di registrazione costituiscono il punto di ingresso per gli utenti che si presentano con smart-card. L'espletamento dell'una o dell'altra funzionalità è portato a termine avvalendosi dei servizi di gestione delle credenziali (password, pin, smart-card) e accedendo al repository locale ove sono memorizzate tutte le utenze attive.

Mediante la *procedura di registrazione* vengono attribuite all'utente le credenziali utilizzabili per accedere ai servizi online. La procedura di registrazione si rende necessaria per poter predisporre il contesto sistemico e applicativo di supporto alla gestione di ogni potenziale utente, cioè per creare il *Profilo Utente*. Dopo essersi registrato l'utente potrà accedere ai servizi cui ha titolo utilizzando le proprie credenziali per l'autenticazione, con modalità che dipendono dal servizio richiesto. Gli utenti che dispongono di strumenti di identificazione validi a livello nazionale o regionale (CIE/CNS/CRS) potranno sempre utilizzarli in fase di autenticazione in sostituzione delle credenziali associate al profilo utente e definite in fase di registrazione.

La fase di autenticazione degli utenti è gestita dall'Identity Provider attraverso diverse modalità che potranno essere utilizzate dall'utente sulla base dei requisiti richiesti per accedere al servizio selezionato:

- *Autenticazione debole*: username + password (assegnati in fase di registrazione)
- *Autenticazione intermedia :* username+password+PIN (assegnati in fase di registrazione)
- *Autenticazione forte*: smartcard (CIE/CNS/Carte di firma con certificato di autenticazione)

Il componente architetturale principale dell'Identity Provider è costituito dall'AuthenticationManager il quale è in grado di realizzare una vera autenticazione con due meccanismi: username+password e smart-card. Tale componente accede ad una parte di "back-end" che gestisce l'accesso ad un repository con i dati relativi a tutti gli utenti registrati. Esso è inoltre in grado di offrire la funzionalità di **Single-Sign-On (SSO)**, grazie alla generazione di un cookie nel dominio dell'Identity Provider che viene consegnato al browser dell'utente che lo può riutilizzare entro un certo intervallo predefinito e personalizzabile in fase di configurazione. L'autenticazione mediante smart-card è realizzata mediante un componente in grado di riconoscere alcuni tipi di carte esistenti sul territorio nazionale (es. Carta Nazionale dei Servizi, Carta Regionale dei Servizi Lombardia, carte di firma di Infocamere/Infocert, ecc.) dotate o meno di una struttura di dati personali interna. Il componente può essere esteso a supportare nuovi tipi di carte che dovessero risultare non ancora riconosciute.

3.5. Local Authority Registry

L'Authority Registry è un componente in grado di rispondere a interrogazioni relative ai soggetti interoperanti nell'infrastruttura di autenticazione citati in precedenza; per ciascuno dei soggetti esso memorizza informazioni quali un identificativo, un tipo (es. Local Proxy, Identity Provider) e l'indirizzo (URL) al quale sono pubblicati le informazioni necessarie per il colloquio con essi, raccolte in opportune strutture XML chiamate *metadati*. L'Authority Registry si comporta all'interfaccia allo stesso modo di una generica Attribute Authority.

Nell'infrastruttura IdM-RAS, è presente il componente chiamato Local Authority Registry, il quale si configura come una versione locale del Authority Registry che a regime sarà messo a disposizione dalla federazione ICAR.

3.6. Architettura complessiva

In Figura 3-4 è rappresentata l'architettura complessiva di IdM-RAS con il dettaglio dei sotto-componenti principali per ciascuno dei soggetti interagenti. Non viene rappresentata la Profile Authority definita nel progetto ICAR, in quanto essa non è dispiegata in tale infrastruttura.

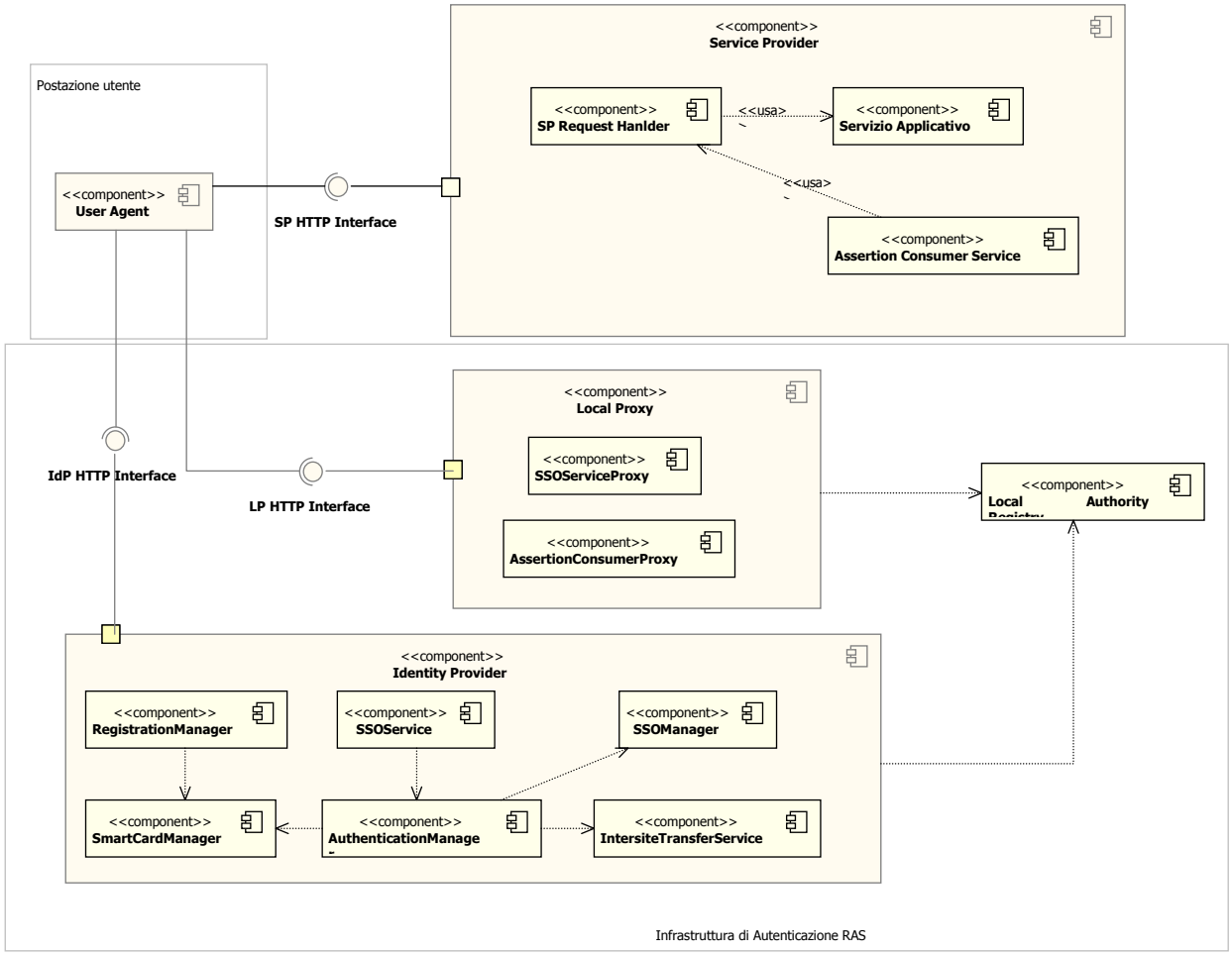

Figura 3-4 - Infrastruttura di autenticazione RAS: architettura complessiva.

4. Integrazione di un servizio informativo regionale

4.1. Interfaccia tra Service Provider e Local Proxy

Come accennato nella descrizione architetturale, un Service Provider è l'entità preposta all'erogazione dei servizi ai quali gli utenti finali possono accedere. I Service Provider coincidono con sistemi informativi da integrare, presso i quali si trovano svariate tipologie di servizi di e-government, per i quali si pone l'esigenza di introdurre una fase preliminare di autenticazione all'accesso. E' frequente il caso in cui servizi pre-esistenti siano caratterizzati ciascuno da un sistema di autenticazione proprietario ed eterogeneo rispetto ai sistemi di autenticazione di altri servizi. L'integrazione di tali servizi con l'infrastruttura di autenticazione oggetto del presente documento consente di realizzare un vero e proprio Service Provider nell'accezione data nel progetto ICAR, e accedere così ad un'infrastruttura omogenea e flessibile, che evita la proliferazione degli account e delle soluzioni ad-hoc realizzate caso per caso.

Nella figura seguente è possibile vedere il flusso che viene seguito per procedere all'autenticazione dell'utente. Per semplificare lo schema i componenti SP Request Handler e Assertion Consumer Service sono stati accorpati in un un unico componente Login Module.

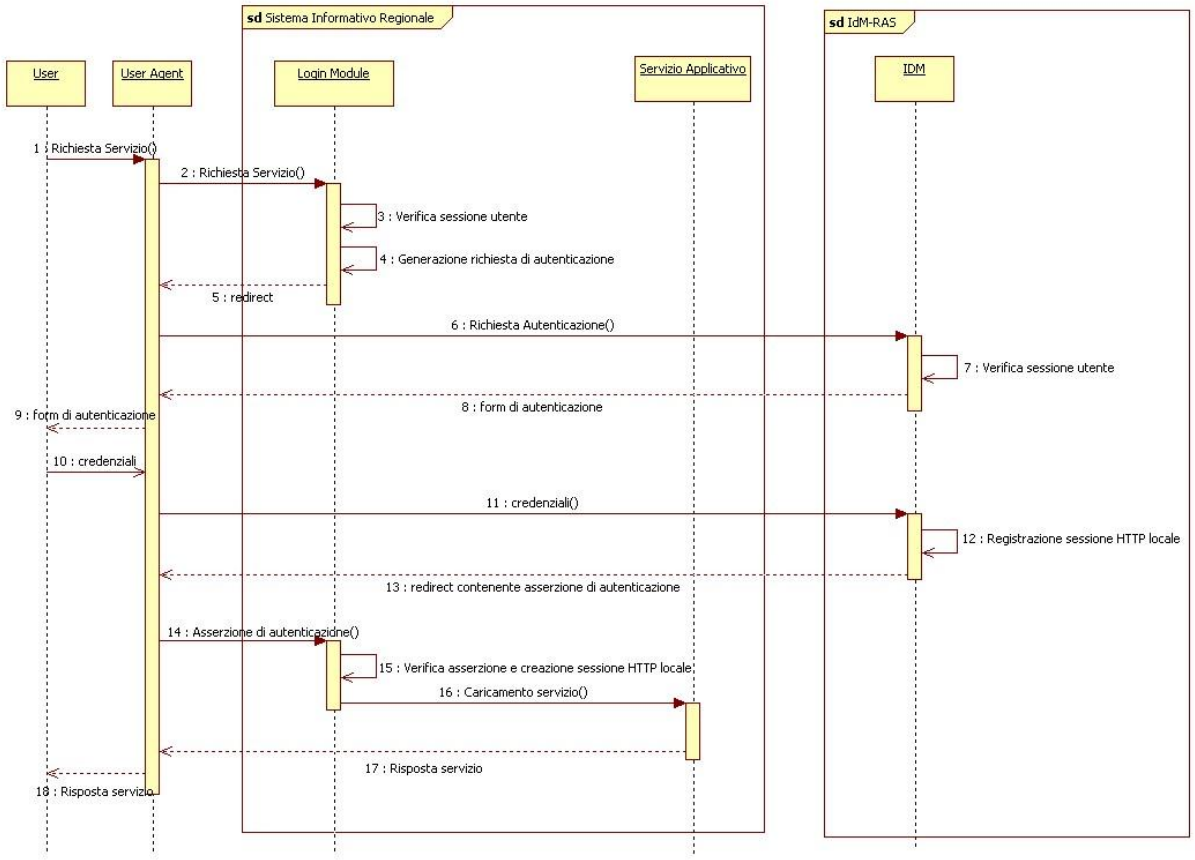

Figura 4-1 Erogazione di un servizio con autenticazione mediante IdM-RAS.

Step	*Descrizione*
1-2	L'utente richiede mediante browser l'accesso ad un servizio erogato da un sistema informativo regionale
3	L' SP Request Handler del Login Module intercetta la richiesta e verifica se l'utente è già autenticato nell'applicazione. Se l'utente è già stato precedentemente riconosciuto si passa al punto 16 altrimenti si passa al punto successivo.
4	Il Login Module genera una richiesta di autenticazione da inviare all'IdM-RAS.
5-6	Il Login Module invia la richiesta mediante HTTP-Redirect o Post.
7	L'IdM-RAS verifica se l'utente è stato riconosciuto precedentemente. Se l'utente è già in sessione si passa al punto 13.
8-9	L'IdM-RAS richiede le credenziali (username e password o smartcard) dell'utente attraverso la pagina di Login
10-11	L'utente fornisce le proprie credenziali.
12	L'IdM-RAS riconosce l'utente e traccia in sessione i dati dell'utente e del servizio richiedente.
13-14	L'IdM-RAS invia all'Assertion Consumer Service del Login Module, tramite Post, l'XML delle asserzioni contenenti i dati dell'utente.
15	L'Assertion Consumer Service del Login Module verifica l'XML contenente le asserzioni e mette a disposizione dei servizi i dati provenienti dall'IdM-RAS
16	Il Login Module procede al caricamento del servizio richiesto dall'utente
17-18	L'utente visualizza le pagine del servizio richiesto.

Tutti i messaggi scambiati tra i service provider e il Local Proxy devono essere firmati ed eventualmente criptati attraverso chiavi asimmetriche dai componenti da cui vengono generati.

4.1.1. Protocolli e binding

Il messaggio con la richiesta di autenticazione può essere inoltrato dal Service Provider al Local Proxy usando il binding HTTP Redirect o il binding HTTP POST. La relativa risposta XML può invece essere inviata dal Local Proxy al Service Provider solo tramite il binding HTTP. Di seguito si riassumono i dettagli della struttura dei messaggi scambiati secondo i vari binding.

4.1.1.1 Struttura dei metadati del Service Provider

Alla ricezione della richiesta di autenticazione proveniente dal Service Provider, il Local Proxy ne verifica anzitutto la firma. A tale scopo il Local Proxy accede ai metadati che devono essere esposti dal Service Provider, all'interno dei quali deve essere presente un certificato contenente la chiave pubblica corrispondente alla chiave privata usata dal Service Provider per firmare i messaggi. I metadati del Service Provider devono inoltre contenere informazioni relative al servizio denominato "AssertionConsumerService" che deve essere presente presso il Service Provider in quanto responsabile di ricevere i messaggi provenienti dall'infrastruttura di autenticazione, e in particolare dal Local Proxy. Di Assertion Consumer Service è necessario specificare il binding utilizzato (deve essere specificato il binding HTTP-POST) e il suo indirizzo URL. Infine, i metadati del Service Provider devono riportare l'elenco delle strutture denominate "AttributeConsumingService", ciascuna delle quali specifica informazioni relative a ciascuno dei servizi erogati da tale Service Provider. Per ciascuno dei servizi è necessario riportare un indice progressivo, il nome e il set di eventuali attributi richiesti.

Chi si occupa di gestire il sistema di IdM-RAS si fa carico di rilasciare e concordare i file contenenti tali metadati.

4.2. Single Log Out

Ogni sistema informativo Regionale collegato al sistema di Identity Management, dopo il riconoscimento dell'utente, crea e mantiene una sessione propria utilizzata per il corretto funzionamento del sistema stesso. Ogni volta che si passa da un sistema all'altro, l'utente viene riconosciuto dal nuovo sistema (funzionalità propria del single sign on) ma la sessione creata nei precedenti viene mantenuta fino alla naturale scadenza o fino alla chiusura del browser.

La funzionalità di Single Log Out permette all'utente, attraverso un unico click, di effettuare la chiusura di tutte le sessioni aperte nelle applicazioni che ha utilizzato. Il SLO dovrà essere scatenato dall'applicazione in cui si trova l'utente, mandando un messaggio formattato secondo determinate specifiche al Local Proxy che si occuperà di contattare gli altri servizi interessati dal log out attraverso un web service che dovrà essere implementato in ogni service provider.

5. Integrazione del Service Provider nell'infrastruttura di autenticazione

Il presente capitolo ha lo scopo di illustrare le modalità secondo cui è possibile integrare un sistema informativo esistente con l'infrastruttura di autenticazione oggetto di questo documento. Nel seguito, in particolare, si affronteranno separatamente le situazioni in cui il sistema informativo in questione, che è visto come un generico Service Provider da parte del sistema di autenticazione, è sviluppato su piattaforma J2EE oppure altre piattaforme.

5.1. Piattaforma J2EE

Questa sezione si riferisce al caso in cui il sistema informativo da integrare, o almeno la parte di esso che si trovi ad interagire con il sistema di autenticazione, sia sviluppato in tecnologia J2EE. E' possibile integrare il proprio sistema con due possibili librerie che possono essere fornite su richiesta:

- spring-security_idm-ras : libreria di integrazione per progetti che sfruttano le librerie spring security versione 2.x.x , 3.0.x e 3.1.x;
- ras-sp-integration-kit : libreria per progetti che non sfruttano le librerie spring security.

5.1.1. RAS-SP-integration-kit - descrizione dei componenti di integrazione

Come illustrato nella sezione architetturale 3.1 e ripreso nella seguente Figura 5-1 Componenti di integrazione con il Service Provider relativa al Service Provider, in esso esistono alcuni componenti fondamentali alla base dell'integrazione con l'infrastruttura IDM-RAS. Tali componenti, descritti nei paragrafi seguenti, rispettano le interfacce descritte nel capitolo precedente per l'invio e la ricezione dei messaggi in standard SAML 2.0. Nell'infrastruttura IDM-RAS dispiegata, tali componenti sono messi a disposizione nel file libreria di nome "ras-sp-integration-kit.jar" che è possibile includere nel CLASSPATH di ogni web application che si intende integrare.

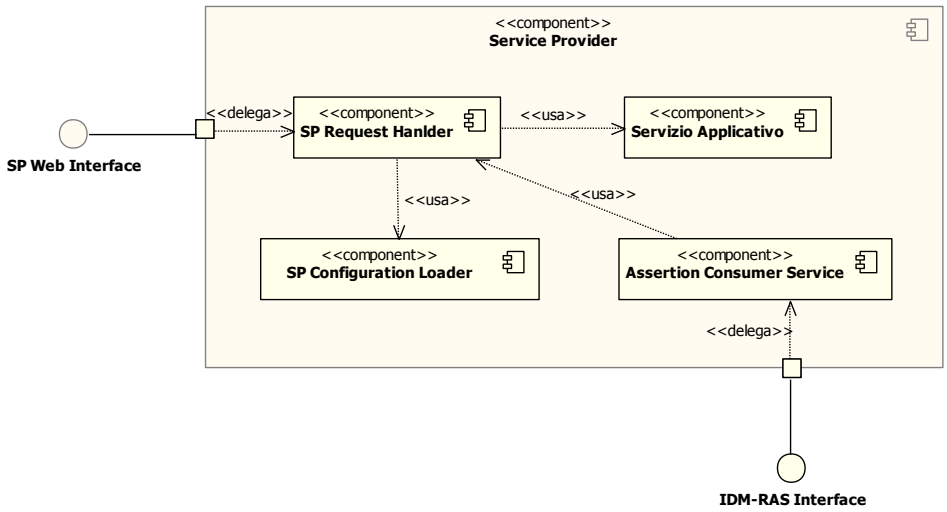

Figura 5-1 Componenti di integrazione con il Service Provider

5.1.1.1 Configurazione del componente SP Request Handler

Come già indicato durante la descrizione architetturale al capitolo 3, IDM-RAS è dotato di un componente detto "SP Request Handler" che deve operare nel contesto web di ciascun Service Provider che si intende integrare con l'infrastruttura di autenticazione. Tale componente è il responsabile di intercettare qualunque richiesta di accesso ai servizi erogati da tale Service Provider, verificare il contesto di sicurezza relativo alla sessione di lavoro ed eventualmente innescare il processo che porta l'utente attivo ad autenticarsi presso un Identity Provider. All'interno dell'infrastruttura di autenticazione, tale componente è realizzato come servlet-filter J2EE che ha accesso alla sessione di lavoro e produce il messaggio SAML di AuthnRequest. Per farlo, esso deve accedere ad un file di configurazione che elenca i dettagli sui requisiti in termini di forza delle credenziali richieste da parte del Service Provider per il servizio richiesto dall'utente.

E' pertanto necessario che il Service Provider venga protetto, relativamente a tutti gli URL dei servizi che esso eroga e che si intendono gestire, tramite tale filtro. Il filtro in oggetto viene configurato preparando un file di configurazione in formato XML come nell'esempio riportato di seguito.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<ServiceProviderConfiguration>
  <ServiceProfiles>
    <ServiceProfile>
      <Name>Servizio 1</Name>
      <Description>Servizio 1</Description>
      <URLPrefix>/servicepage1</URLPrefix>
```

```
      <AuthenticationMethodType>strong</AuthenticationMethodType>
    </ServiceProfile>
    <ServiceProfile>
      <Name>Servizio 2</Name>
      <Description>Servizio 2</Description>
      <URLPrefix>/servicepage2</URLPrefix>
      <AuthenticationMethodType>weak, strong</AuthenticationMethodType>
    </ServiceProfile>
  </ServiceProfiles>
  <AuthenticationMethods>
    <AuthenticationMethod>
      <Type>weak</Type>
      <Method>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </Method>
    </AuthenticationMethod>
    <AuthenticationMethod>
      <Type>strong</Type>
      <Method>urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard</Method>
    </AuthenticationMethod>
  </AuthenticationMethods>
</ServiceProviderConfiguration>
```

Figura 5.2 - Esempio di file di configurazione per il componente SP Request Handler

Il file di configurazione sopra rappresentato è diviso in due sezioni. La prima, delimitata da tag **<ServiceProfiles>**, consente di elencare tutti i servizi erogati dal relativo Service Provider, all'interno di sezioni marcate con il tag **<ServiceProfile>.** Per ogni servizio, ciascuna di tali sezioni indica un nome (tag **<Name>**), una descrizione (tag **<Description>**), il prefisso caratteristico degli URL delle pagine di tale servizio (tag **<URLPrefix>**) ed infine i tipi di metodi di autenticazione supportati, separati da una virgola (tag **<AuthenticationMethodType>**). Quest'ultima informazione è definita nella seconda parte del file, delimintata da tag **<AuthenticationMethods>**, che tratta proprio dei metodi di autenticazione riconosciuti dal Service Provider e che possono quindi essere richiesti da uno dei servizi descritti nella prima parte del file. In particolare, ciascuna delle tipologie di autenticazione supportate è definita in una sezione marcata con il tag **<AuthenticationMethod>.** Per ogni metodo di autenticazione deve essere specificato il tipo (tag **<Type>**), al quale si riferiscono i tag **<AuthenticationMethodType>** della sezione precedente, e uno o più metodi effettivi (tag **<Method>**), così come definiti dalla specifica SAML 2.0. Ciascun Service Provider infatti può definire autonomamente le proprie tipologie di metodi di autenticazione, facendo però corrispondere a ciascuna di esse un insieme di relativi meccanismi in standard SAML 2.0. Nell'esempio sopra riportato sono definite due tipologie di metodi di autenticazione, rispettivamente denominate "weak" e "strong", alle quali corrispondono il meccanismo di autenticazione con username e password, rispettivamente con smart-card. Il primo dei servizi elencati, chiamato "Servizio 1" richiede un'autenticazione di tipo "strong", mentre il secondo di tipo "weak" oppure "strong".[1]

In aggiunta al file di configurazione sopra descritto, il componente SP Request Handler necessita di altre informazioni per creare il messaggio SAML AuthnRequest. In particolare sono necessari i dati relativi a quale tra i servizi richiesti è stato scelto e l'indirizzo al quale inviare i messaggi SAML Response, al termine del processo di autenticazione. Entrambe tali informazioni vengono ricavate da un

[1] Tutti i meccanismi elencati all'interno di ciascuna delle tipologie saranno utilizzati per comporre la struttura "RequestedAuthenticationContext" dei messaggi "SAML AuthnRequest" definiti dalla specifica SAML 2.0.

secondo file, definito dalla specifica SAML 2.0 per ciascuno degli attori interagenti nello scenario di autenticazione. Tale file, detto "metadata file", contiene informazioni che si differenziano secondo il ruolo del soggetto al quale il file è associato. In questo modo, il file di metadati associato ad un soggetto di tipo Service Provider conterrà informazioni diverse dal file di metadati associato ad un Identity Provider e così via. Per maggiori dettagli relativamente ai file di metadati e alla loro struttura si rimanda alla documentazione prodotta dal task INF-3 del progetto ICAR [7] e alla specifica SAML 2.0 [3].

Il componente SP Request Handler accede pertanto al file di metadati definito per il Service Provider, dal quale legge l'indice posizionale della struttura "AttributeConsumingService" relativa al servizio richiesto, e l'URL del servizio al quale risponde il componente Assertion Consumer Service, in grado di ricevere i messaggi di risposta "SAML Response", inviati da IDM-RAS.

5.1.1.2 Assertion Consumer Service

Nell'ambito del sistema di autenticazione dispiegato presso la Regione Autonoma Sardegna è fornito un componente detto "*Assertion Consumer Service*" che deve operare nel contesto di ciascun Service Provider che si intende integrare con l'infrastruttura di autenticazione. Tale componente è il responsabile di ricevere i messaggi SAML Response prodotti dal Local Proxy e destinati al Service Provider. Come illustrato in precedenza, tali messaggi contengono le informazioni relative al processo di autenticazione, in particolare il meccanismo utilizzato per tale processo (es. username e password) ed eventuali altri dati sugli attributi del profilo dell'utente. Il messaggio SAML Response viene convalidato da Assertion Consumer Service che ne verifica la firma e la struttura, e quindi processato, depositando in sessione alcuni Java Bean con le informazioni veicolate dal Local Proxy, che restano quindi accessibili ai servizi erogati dal Service Provider.

In particolare, la sessione di ciascun Service Provider viene inizializzata con una mappa inserita nell'attributo di nome "serviceContextMap", che associa a ciascun URL prefix caratteristico di un servizio scelto dall'utente un'istanza del bean chiamato *AuthenticationSessionBean.* Quest'ultimo contiene tutte le informazioni relative alla fase di avvenuta autenticazione dell'utente, inclusi eventuali attributi del suo profilo, che sono stati reperiti contestualmente alla fase stessa. In dettaglio, le informazioni contenute nei campi di tale bean che possono essere utili ai servizi erogati dal Service Provider sono elencati nella seguente tabella:

Nome campo nel bean	*Tipo del contenuto*	*Descrizione del contenuto*
userID	String	l'identificativo del subject (utente) che ha effettuato l'autenticazione
attributesMap	Map<String,List<String>>	mappa che associa al codice di ogni attributo la lista dei suoi valori (stringhe)
attributesFriendlyNamesMap	Map<String,String>	mappa che associa al codice di ogni attributo un nome descrittivo così come noto al Service Provider
authenticationAssertion	String	asserzione di autenticazione così come ricevuta dal Local Proxy
assertionWallet	AssertionWallet	bean contenente l'insieme delle asserzioni raccolte durante tutto il processo di autenticazione. Il bean AssertionWallet presenta un unico campo "assertionList" di tipo List contenente stringhe
authenticationMethod	String	la tipologia del meccanismo di autenticazione utilizzato, così come nota al Service Provider

5.1.2. Parametri di configurazione del Service Provider

In aggiunta alla preparazione di un file di configurazione così come descritto nelle sezioni precedenti, la configurazione di un Service Provider integrato con IDM-RAS si completa realizzando il relativo file di metadati in conformità a quanto definito dalla specifica SAML 2.0 e con l'impostazione di tutti i parametri di configurazione necessari all'interno del deployment-descriptor della web application ad esso relativa.

Per quanto riguarda il file dei metadati, si rimanda alla documentazione realizzata dal task INF-3 del progetto ICAR [7] e alla specifica SAML 2.0 [4]. In questa sede si vuole sottolineare il fatto che è necessario specificare tante sezioni **`<AttributeConsumingService>`** quanti sono i servizi erogati dal Service Provider. Una di tali sezioni deve essere marcata con l'attributo "isDefault=true" ad indicare

quale tra i servizi elencati è quello al quale ci si deve riferire, nel caso in cui l'URL del servizio scelto dall'utente non comprenda il parametro "serviceIndex" nella query string. Il valore di questo parametro è un numero intero che rappresenta l'indice posizionale del servizio all'interno del file dei metadati.

Relativamente ai rimanenti parametri di configurazione del Service Provider, essi sono elencati all'interno del file deployment descriptor che fa parte della web application che lo realizza.

5.1.2.1 Parametri di contesto

Nella tabella seguente sono elencati i parametri di contesto (context-param) che devono essere presenti nel deployment-descriptor web.xml, con il relativo significato.

Nome del parametro	*Significato*
keystorePath	Percorso, relativo alla web application, del file con il keystore che deve essere utilizzato dai componenti del SP per firmare i messaggi SAML 2.0 prodotti
keystoreAlias	Alias della chiave privata, contenuta nel keystore specificato al punto precedente, da utilizzare per la firma dei messaggi SAML 2.0.
keystorePassword	Password da utilizzare per accedere al keystore e alla chiave privata in esso contenuta. Si assume che la password del keystore e della chiave coincidano.
configurationFile	Percorso, relativo alla web application, del file XML contentente la configurazione dei servizi erogati dal SP, così come descritto alla sezione 5.1.1.1
showDetailInErrorPage	(opzionale) Se viene utilizzata la pagina di errore "error.jsp" fornita insieme al Service Provider dimostrativo, questo parametro indica (valori "true" o "false") se deve essere prodotto un dettaglio dell'errore verificatosi
metadataFile	Percorso, relativo alla web application, del file dei metadati SAML 2.0 relativi al SP.
metadataConnectionTimeout	Valore, espresso in millesimi di secondo, del timeout di attesa durante ogni connessione al servizio HTTP di fornitura metadati esposto dai vari soggetti con i quali il SP si trova ad interagire
entityURLPrefix	E' il prefisso dell'URL della webapp del SP. Come si vede, esso comprende unicamente il protocollo, l'hostname e la porta.
entityID	E' l'identificativo dell'entità-SP che comparirà in tutti i messaggi SAML 2.0.
icar.inf3.error.returnURL	URL assoluto al quale è necessario ritornare a seguito del verificarsi di un messaggio di errore
language	Sigla della lingua richiesta dal SP. Al momento, questo valore deve essere impostato a "it".
authorityRegistryMetadataProviderURL	E' l'URL assoluto del servizio HTTP di fornitura dei metadati esposto dal componente Local Authority Registry.

5.1.2.2 Parametri del filtro di autenticazione

Il deployment-descriptor dichiara obbligatoriamente il componente SP Request Handler che è realizzato dal filtro J2EE la cui classe è "it.cefriel.ras.auth.web.filters.AuthenticationFilter". I suoi parametri sono i seguenti:

Nome del parametro	*Significato*
forwardBinding	E' il binding da utilizzare per l'invio dei messaggi "SAML AuthnRequest". I valori possibili sono "HTTP-POST" e "HTTP-REDIRECT".
returnBinding	E' il binding che deve essere utilizzato da IDM-RAS per l'invio dei messaggi "SAML Response". Deve esssere impostato al valore "HTTP-POST".
localProxyMetadataProviderURL	E' l'URL assoluto del servizio HTTP di fornitura metadati del componente Local Proxy di IDM-RAS.
postAuthnRequestPage	Percorso, relativo alla web application, della pagina JSP da utilizzare per l'invio mediante binding POST dei messaggi "SAML AuthnRequest". Se viene utilizzata la pagina fornita insieme al SP dimostrativo, si può impostare il valore "/resources/PostAuthnRequest_debug.jsp"

Nome del parametro	*Significato*
proxyCount	Valore intero positivo che indica il numero di step consentiti nei quali il messaggio SAML AuthnRequest può essere inoltrato ad altri soggetti. Dato che normalmente tali richieste devono raggiungere l'Identity Provider transitando per il Local Proxy, tale valore deve essere impostato al valore 2.

5.1.2.3 Parametri di Assertion Consumer Service

Il deployment-descriptor dichiara obbligatoriamente il componente Assertion Consumer Service che è realizzato dal filtro J2EE la cui classe è "it.cefriel.ras.auth.web.servlet.AssertionConsumerService". I suoi parametri sono i seguenti:

Nome del parametro	*Significato*
authnFailedPage	Percorso, relativo alla web application, della pagina JSP da utilizzare nel caso la procedura di autenticazione non sia andata a buon fine. Se viene utilizzata la pagina fornita insieme al SP dimostrativo, si può impostare il valore "/resources/authnFailed.jsp"

5.1.2.4 Servizio di pubblicazione dei metadati

All'interno del deployment-descriptor web.xml è necessario obbligatoriamente dichiarare il servizio di pubblicazione via HTTP dei metadati contenuti nel file specificato al parametro di contesto di nome "metadataFile" indicato sopra. Tale servizio è realizzato mediante la servlet "it.cefriel.icar.inf3.web.servlet.MetadataPublisherServlet". L'URL con il quale tale servlet viene mappata deve essere registrato all'interno del file XML di configurazione del componente Local Authority Registry, nella sezione relativa al Service Provider.

5.1.3. Classe per il trattamento dei messaggi e delle asserzioni

Se si vuole integrare la propria applicazione in maniera semplice, per la gestione dei messaggi, è stata sviluppata una classe ad-hoc (it.sardegnait.idmras.samlws.SAMLService su ras-sp-integration-kit.jar). In particolar modo la classe si occupa di generare i messaggi di richiesta da inviare all'IdM-RAS e di verificare i dati presenti sulla risposta ricevuta dall'IdM-RAS.

Le richieste vengono generate attraverso questi metodi :

- *getSAMLPostRequest* : genera e firma l'XML di richiesta di autenticazione da inviare mediante HTTP-POST al Local Proxy;
- *getSAMLPostRequestWithAssertionConsumerURL* : genera e firma l'XML di richiesta di autenticazione da inviare mediante HTTP-POST al Local Proxy impostando come AssertionConsumerURL l'indirizzo specificato nei parametri del metodo;
- *getSAMLRedirectRequestURL* : restituisce l'URL contenente l'XML di autenticazione firmato. L'applicazione dovrà fare un redirect verso questo URL. I parametri da passare sono 3 stringhe:
    - requestedURL : identificativo del servizio chiamante;
    - relayStateID : id della sessione sulla applicazione chiamante;
    - attributeConsumingServiceIndex : sempre a "1";
- *getSAMLRedirectRequestURLWithAssertionConsumerURL* : genera l'url, contenente l'XML di autenticazione firmato, per effettuare il bind via HTTP-Redirect impostando come AssertionConsumerURL l'indirizzo specificato nei parametri del metodo;
- *getSAMLSingleLogoutRequestWithURL* : restituisce l'xml da mandare al servizio di singleLogout. L'xml prodotto dovrà essere mandato mediante binding SOAP. I parametri da passare sono due stringhe:
    - codiceFiscale : codice fiscale dell'utente richiedente;

  - requestedURL : identificativo del servizio chiamante;

- *getSAMLSingleLogoutServiceResponseWithURL* genera la risposta da mandare in seguito alla richiesta di logout da parte dell'IdM-RAS. Il parametro da passare è una stringa:
  - requestedURL : identificativo del servizio chiamante;
- *getSAMLSingleLogoutServiceResponse* : genera e firma l'XML di risposta in seguito ad un single logout inviato, mediante web service, dal local proxy;
- *verifyLogoutRequestWithURL* verifica la firma dei dati della richiesta di logout proveniente dall'idm-ras inviata servizio mediante binding SOAP. Se le verifiche danno esito positivo il metodo restituisce la username(codice fiscale) dell'utente. I parametri da passare sono due stringhe:
  - requestedURL : identificativo del servizio chiamante;
  - logoutRequestXML : XML arrivato dall'IdM-RAS (nel file di esempio si tratta dell'XML che sta all'interno del tag <soap11:Body>);
- *verifyResponseWithURL* : verifica la risposta ricevuta dall'IdM restituendo l'xml con i dati dell'utente o un xml di errore in caso di risposta errata. I parametri da passare sono due stringhe:
  - requestedURL : identificativo del servizio chiamante;
  - samlResponse : xml di risposta ricevuto dall'IdM.

Qui di seguito un XML di esempio:

```
<?xml version="1.0" encoding="UTF-8"?>
<userattributes>
	<attribute name="codiceFiscale">CGNNMO78E06B354P</attribute>
	<attribute name="cognome">Cognome</attribute>
	<attribute name="nome">Nome</attribute>
	<attribute name="nrCivicoResidenza">11</attribute>
	<attribute name="provinciaNascita">CA</attribute>
	<attribute name="telefono">0702929</attribute>
	<attribute name="emailAddress">prova@sardegnait.it</attribute>
	<attribute name="capResidenza">09100</attribute>
	<attribute name="cittaResidenza">Cagliari</attribute>
	<attribute name="sesso">M</attribute>
	<attribute name="provinciaResidenza">CA</attribute>
	<attribute name="statoResidenza">Italia</attribute>
	<attribute name="cellulare">32008271</attribute>
	<attribute name="dataNascita">16/12/1970</attribute>
	<attribute name="indirizzoResidenza">Via Roma</attribute>
	<attribute name="luogoNascita">Cagliari</attribute>
	<attribute name="AuthenticationMethod">
		urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
	</attribute>
</userattributes>
```

- *verifyResponseWithURL :* in base all'URL di richiesta passato come parametro verifica la firma e la scadenza dei dati, e restituisce un XML contenente tutti i dati dell'utente come nell'esempio precedente.

I file di configurazione di questa classe si trovano sotto la cartella conf-SAMLService.

5.1.4. Integrazione con spring security

Per i progetti che utilizzano il modulo spring security è stata creata una libreria che sfrutta l'estensione SAML2 (http://docs.spring.io/spring-security/site/extensions/saml/). E' disponibile per le versioni 2.x.x , 3.0.x e 3.1.x di spring security.

Per la corretta integrazione è necessario modificare nel file xml che riporta le configurazioni del modulo spring security l'entry point di autenticazione sul tag http e inserire tutti i filtri utili alla libreria.Qua sotto un esempio del file per la versione 3 :

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:security="http://www.springframework.org/schema/security"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
            http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- Enable autowiring -->
    <context:annotation-config/>
    <context:component-scan base-package="org.springframework.security.saml"/>
    <security:http entry-point-ref="samlEntryPoint">
        <security:intercept-url pattern="/**" access="IS_AUTHENTICATED_FULLY"/>
        <security:intercept-url pattern="/saml/web/**" filters="none"/>
        <security:intercept-url pattern="/logout.jsp" filters="none"/>
        <security:intercept-url pattern="/login.jsp" filters="none"/>
        <security:intercept-url pattern="/favicon.ico" filters="none"/>
        <security:custom-filter before="PRE_AUTH_FILTER" ref="metadataFilter"/>
        <security:custom-filter position="PRE_AUTH_FILTER" ref="samlEntryPoint"/>
        <security:custom-filter after="BASIC_AUTH_FILTER" ref="samlProcessingFilter"/>
        <security:custom-filter after="LOGOUT_FILTER" ref="samlLogoutFilter"/>
        <security:custom-filter before="LOGOUT_FILTER" ref="samlLogoutProcessingFilter"/>
        <security:session-management>
            <security:concurrency-control max-sessions="1" session-registry-
ref='sessionRegistry'/>
        </security:session-management>
    </security:http>

    <bean id="sessionRegistry"
class="org.springframework.security.core.session.SessionRegistryImpl"/>

    <!-- Handler deciding where to redirect user after successful login -->
    <bean id="successRedirectHandler"
class="org.springframework.security.web.authentication.SavedRequestAwareAuthenticationSuccessHa
ndler">
        <property name="defaultTargetUrl" value="/"/>
    </bean>

    <!--
    Use the following for interpreting RelayState coming from unsolicited response as redirect
URL:
    <bean id="successRedirectHandler"
class="org.springframework.security.saml.SAMLRelayStateSuccessHandler">
       <property name="defaultTargetUrl" value="/" />
    </bean>
    -->

    <!-- Handler for successful logout -->
    <bean id="successLogoutHandler"
class="org.springframework.security.web.authentication.logout.SimpleUrlLogoutSuccessHandler">
        <property name="defaultTargetUrl" value="/"/>
    </bean>

    <!-- Register authentication manager with SAML provider -->
    <security:authentication-manager alias="authenticationManager">
        <security:authentication-provider ref="samlAuthenticationProvider"/>
```

```
    </security:authentication-manager>

    <!-- Logger for SAML messages and events -->
    <bean id="samlLogger" class="org.springframework.security.saml.log.SAMLDefaultLogger"/>

    <!-- Central storage of cryptographic keys -->
    <bean id="keyManager" class="org.springframework.security.saml.key.JKSKeyManager">
        <constructor-arg value="classpath:security/samlKeystore.jks"/>
        <constructor-arg type="java.lang.String" value="nalle123"/>
        <constructor-arg>
            <map>
                <entry key="apollo" value="nalle123"/>
            </map>
        </constructor-arg>
        <constructor-arg type="java.lang.String" value="apollo"/>
    </bean>

    <!-- Entry point to initialize authentication, default values taken from properties file --
>
    <bean id="samlEntryPoint" class="org.springframework.security.saml.SAMLEntryPoint">
        <property name="filterSuffix" value="/saml/login"/>
        <!-- OPTIONAL property: In case idpSelectionPath property is not set the user will be
redirected to the default IDP -->
        <property name="idpSelectionPath" value="/WEB-INF/security/idpSelection.jsp"/>
        <property name="defaultProfileOptions">
            <bean class="org.springframework.security.saml.websso.WebSSOProfileOptions">
                <property name="includeScoping" value="true"/>
                <property name="assertionConsumerIndex" value="1"/>
                <property name="allowCreate" value="true"/>
                <property name="nameID" value="urn:oasis:names:tc:SAML:2.0:nameid-
format:transient"/>
                <property name="authnContexts">
                    <list>
<value>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</value>
                        <value>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</value>
                    </list>
                </property>
            </bean>
        </property>
    </bean>

    <!-- OPTIONAL bean: The filter is waiting for connections on URL suffixed with filterSuffix
and presents SP metatdata there -->
    <bean id="metadataFilter"
class="org.springframework.security.saml.metadata.MetadataDisplayFilter">
        <property name="filterSuffix" value="/saml/metadata"/>
    </bean>

    <!-- Class is capable of generating SP metadata describing the currently running
environnment -->
    <bean id="metadataGenerator"
class="org.springframework.security.saml.metadata.MetadataGenerator">
    </bean>

    <!-- IDP Metadata configuration - paths to metadata of IDPs in circle of trust is here -->
    <!-- Do no forget to call iniitalize method on providers -->

    <bean id="metadata"
class="org.springframework.security.saml.metadata.CachingMetadataManager">
        <constructor-arg>
            <list>
                 <bean
class="org.springframework.security.saml.metadata.ExtendedMetadataDelegate">

                    <constructor-arg>
                        <bean
class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                            <constructor-arg>
                                <value type="java.io.File">classpath:security/metadata-
test.customize.it.xml</value>
                            </constructor-arg>
```

```
                        <property name="parserPool" ref="parserPool"/>
                    </bean>
                </constructor-arg>

                <constructor-arg>
                    <bean
class="org.springframework.security.saml.metadata.ExtendedMetadata">
                    </bean>
                </constructor-arg>
            </bean>
            <bean class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider">
                <!-- URL containing the metadata -->
                <constructor-arg>
                    <value type="java.lang.String">https://idm.regione.sardegna.it/ras-
lp/MetadataPublisherServlet</value>
                </constructor-arg>

                <!-- Timeout for metadata loading in ms -->
                <constructor-arg>
                    <value type="int">5000</value>
                </constructor-arg>
                <property name="parserPool" ref="parserPool"/>
            </bean>
        </list>
    </constructor-arg>

    <!-- OPTIONAL used when one of the metadata files contains information about this
service provider -->
    <property name="hostedSPName" value="test.customize.it"/>
    <!-- OPTIONAL property: can tell the system which IDP should be used for authenticating
user by default. -->
    <property name="defaultIDP" value="https://idm.staging.regione.sardegna.it/ras-lp"/>
</bean>

<!-- SAML Authentication Provider responsible for validating of received SAML messages -->
<bean id="samlAuthenticationProvider"
class="org.springframework.security.saml.SAMLAuthenticationProvider">
    <!-- OPTIONAL property: can be used to store/load user data after login -->
    <!--
    <property name="userDetails" ref="bean" />
    -->
</bean>

<!-- Provider of default SAML Context -->
<bean id="contextProvider"
class="org.springframework.security.saml.context.SAMLContextProviderImpl"/>

<!-- Override default authentication processing filter with the one processing SAML
messages -->
<bean id="samlProcessingFilter"
class="org.springframework.security.saml.SAMLProcessingFilter">
    <property name="authenticationManager" ref="authenticationManager"/>
    <property name="authenticationSuccessHandler" ref="successRedirectHandler"/>
</bean>

<!-- Logout handler terminating local session -->
<bean id="logoutHandler"
 class="org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler">

    <property name="invalidateHttpSession" value="false"/>
</bean>

<!-- Override default logout processing filter with the one processing SAML messages -->
<bean id="samlLogoutFilter"
class="it.sardegnait.springframework.security.saml.SAMLLogoutFilter">
    <constructor-arg ref="successLogoutHandler"/>
    <constructor-arg ref="logoutHandler"/>
    <constructor-arg ref="logoutHandler"/>
</bean>
```

```
    <!-- Filter processing incoming logout messages -->
    <!-- First argument determines URL user will be redirected to after successful global
logout -->
    <bean id="samlLogoutProcessingFilter"
class="org.springframework.security.saml.SAMLLogoutProcessingFilter">
        <constructor-arg ref="successLogoutHandler"/>
        <constructor-arg ref="logoutHandler"/>
    </bean>

    <!-- Class loading incoming SAML messages from httpRequest stream -->
    <bean id="processor" class="org.springframework.security.saml.processor.SAMLProcessorImpl">
        <constructor-arg>
            <list>
                <ref bean="redirectBinding"/>
                <ref bean="postBinding"/>
                <ref bean="artifactBinding"/>
                <ref bean="soapBinding"/>
                <ref bean="paosBinding"/>
            </list>
        </constructor-arg>
    </bean>

    <!-- SAML 2.0 Assertion Consumer -->
    <bean id="webSSOprofileConsumer"
class="it.sardegnait.springframework.security.saml.websso.WebSSOProfileConsumerImpl"/>

    <!-- SAML 2.0 Web SSO profile -->
    <bean id="webSSOprofile"
class="it.sardegnait.springframework.security.saml.websso.WebSSOProfileImpl"/>

    <!-- SAML 2.0 ECP profile -->
    <bean id="ecpprofile"
class="org.springframework.security.saml.websso.WebSSOProfileECPImpl"/>

    <!-- SAML 2.0 Logout Profile -->
    <bean id="logoutprofile"
class="it.sardegnait.springframework.security.saml.websso.SingleLogoutProfileImpl"/>
    <!-- Bindings, encoders and decoders used for creating and parsing messages -->
    <bean id="postBinding" class="org.springframework.security.saml.processor.HTTPPostBinding">
        <constructor-arg ref="parserPool"/>
        <constructor-arg ref="velocityEngine"/>
    </bean>
    <bean id="redirectBinding"
class="org.springframework.security.saml.processor.HTTPRedirectDeflateBinding">
        <constructor-arg ref="parserPool"/>
    </bean>
    <bean id="artifactBinding"
class="org.springframework.security.saml.processor.HTTPArtifactBinding">
        <constructor-arg ref="parserPool"/>
        <constructor-arg ref="velocityEngine"/>
        <constructor-arg>
            <bean
class="org.springframework.security.saml.websso.ArtifactResolutionProfileImpl">
                <constructor-arg>
                    <bean class="org.apache.commons.httpclient.HttpClient"/>
                </constructor-arg>
                <property name="processor">
                    <bean id="soapProcessor"
class="org.springframework.security.saml.processor.SAMLProcessorImpl">
                        <constructor-arg ref="soapBinding"/>
                    </bean>
                </property>
            </bean>
        </constructor-arg>
    </bean>

    <bean id="soapBinding"
class="org.springframework.security.saml.processor.HTTPSOAP11Binding">
        <constructor-arg ref="parserPool"/>
    </bean>
    <bean id="paosBinding"
class="org.springframework.security.saml.processor.HTTPPAOS11Binding">
```

```
        <constructor-arg ref="parserPool"/>
    </bean>

    <!-- Initialization of OpenSAML library-->
    <bean class="org.springframework.security.saml.SAMLBootstrap"/>

    <!-- Initialization of the velocity engine -->
    <bean id="velocityEngine" class="org.springframework.security.saml.util.VelocityFactory"
          factory-method="getEngine"/>
    <!-- XML parser pool needed for OpenSAML parsing -->
    <bean id="parserPool" class="org.opensaml.xml.parse.BasicParserPool" scope="singleton"/>
</beans>
```

Al suo interno vengono configurati anche i riferimenti agli ambienti idm, al file dei metadati, al certificato utilizzato per la firma delle asserzioni e il nome del servizio (hostedSPName). Verrà fornita una copia precompilata del file sopra riportato all'inizio del progetto.

Per quanto riguarda il codice, tutti i dati del profilo provenienti dall'IdM vengono memorizzati su una istanza della classe it.sardegnait.springframework.security.saml.SAMLCredential presente all'interno dell'Authentication del ServiceContext. Attraverso il metodo getAttributeFirstValue(<nome campo>) è possibile recuperare i valori. I nomi dei campi possono essere presi dalla classe com.sardegnait.springframework.security.saml.userdetails.UserProfileConstants . E' inoltre possibile sviluppare una classe che implementa UserDetailsService. Questo bean deve essere configurato e passato a userDetailsServiceBuilder all'interno del file di configurazione sopra descritto. Di questa classe verrà richiamato il metodo loadUserByUsername() a cui verrà passato il codice fiscale dell'utente riconosciuto e su cui può essere inserito il codice per la gestione dell'integrazione con i dati dell'applicazione.

Per il logout deve essere richiamato il link <nome_webapp>/saml/logout?local=false. Mettendo il parametro *local* a *true* si effettuerà un logout solo a livello applicativo senza richiamare il servizio di single logout dell'IdM-RAS.

5.2. Altre piattaforme

Per qualsiasi altra piattaforma di sviluppo è possibile utilizzare la stessa classe descritta precedentemente attraverso un web service che espone la stessa.

5.3. Campi IdM-RAS

L'IdM-RAS raccoglie i dati anagrafici di inseriti da ogni utente in fase di registrazione. Nella fase di autenticazione questi dati vengono inviati ai servizi richiedenti. Nella tabella che segue vengono riportati i campi restituiti dall'IdM-RAS:

Nome campo	*Tipo del contenuto (dimensione massima)*	*Descrizione del campo*
codiceFiscale	Stringa(16)	Codice fiscale dell'utente
nome	Stringa(128)	Nome dell'utente
cognome	Stringa(128)	Cognome dell'utente
dataNascita	Data (gg/mm/aaaa)	Data di nascita dell'utente
luogoNascita	Stringa(128)	Comune di nascita dell'utente
provinciaNascita	Stringa(2)	Provincia di nascita dell'utente
sesso	Stringa(1)	Sesso dell'utente
indirizzoResidenza	Stringa(128)	Indirizzo di residenza dell'utente
nrCivicoResidenza	Stringa(10)	Numero civico di residenza dell'utente
cittaResidenza	Stringa(128)	Comune di residenza dell'utente
capResidenza	Stringa(30)	CAP del comune di residenza dell'utente

Nome campo	*Tipo del contenuto (dimensione massima)*	*Descrizione del campo*
provincia	Stringa(2)	Provincia del comune di residenza dell'utente
telefono	Stringa(30)	Numero di telefono dell'utente
cellulare	Stringa(30)	Numero di cellulare dell'utente
emailAddress	Stringa(255)	Indirizzo e-mail dell'utente