AGENDA DIGITALE LOMBARDA

–Specifiche di Interfaccia ai Servizi –

Integrazione Gateway Enti Locali (GEL) tramite Shibboleth

Codice Documento: **LI-SIS-W8B3-GEL#001**
Revisione del Documento: **4**
Data revisione: **10-01-2018**

	Struttura	Nome	Data	Firma
Redatto da:	Innovazione Digitale Sicurezza Applicativa e Progetti Innovativi	Daniele Crespi Alberto Zanini	10-01-2018	
Verificato da:				
Approvato da:				
Emesso da:				

Cronologia delle Revisioni

Revisione	Data	Sintesi delle Modifiche
1.0	12-07-2017	Prima stesura
2.0	31-08-2017	Introdotte nuove informazioni rilasciate nella asserzione
3.0	09-11-2017	Adeguamenti minori (sezioni 2, 3.1 e 3.4)
4.0	10-01-2018	Separazione dei kit di integrazione dedicati all'ambiente di Test ed Esercizio (sezione 3.1); adeguamento dei dati restituiti in asserzione (sezione 3.5)

Limiti di utilizzo del documento
In base alla classificazione del documento.

Indice

1. Introduzione

1.1 Scopo e campo di applicazione documento

Questo documento ha lo scopo di illustrare le interfacce per la comunicazione tra gli attori coinvolti nello scenario per l'accesso ai servizi mediato dal componente denominato Gateway Enti Locali (GEL), facente parte del servizio IdPC (Identity Provider Cittadini di Regione Lombardia).

Tale componente regola la fase di autenticazione degli utenti (cittadini) che, utilizzando un web browser, richiedono i servizi online offerti dai diversi Enti che intendono avvalersi delle funzionalità di IdPC, sia per l'identificazione informatica tramite l'utilizzo della Carta Regionale dei Servizi (CRS) ovvero Tessera Sanitaria – Carta Nazionale dei Servizi (TS-CNS), che tramite il Sistema Pubblico per l'Identità Digitale (SPID).

Nel seguito, dopo una breve presentazione dell'architettura del sistema, verranno forniti i dettagli sulle interfacce fornite e richieste da parte di IdPC nei confronti dei fornitori di servizi esterni con i quali avverrà lo scambio di informazioni, al fine di consentire o negare l'accesso agli utenti

1.2 Riferimenti

	Definizione
[1]	OASIS – "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0" - OASIS Standard, 15March 2015
[2]	OASIS – "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0" - OASIS Standard, 15March 2015

Tabella 1 - Riferimenti

1.3 Acronimi e Definizioni

	Definizione
AgID	Agenzia per l'Italia Digitale
CA	Certification Authority
CAD	Codice dell'Amministrazione Digitale
CN	Common Name
CNS	Carta Nazionale dei Servizi
CRS	Carta Regionale dei Servizi
EELL	Enti Locali
GEL	Gateway Enti Locali
HTTP	Hyper Text Transfer Protocol
IdP	Identity Provider
IdPC	Identity Provider Cittadini di Regione Lombardia
LI, LISPA	Lombardia Informatica
OTP	One Time Password
PEO	Posta Elettronica Ordinaria
RL	Regione Lombardia
SAML	Security Assertion Markup Language
Shibboleth	OpenSource Shibboleth-ServiceProvider (https://shibboleth.net/)
SP	Service Provider
SPID	Sistema Pubblico per l'Identità Digitale
SSL	Secure Socket Layer
TLS	Transport Layer Security
TS-CNS	Tessera Sanitaria – Carta Nazionale dei Servizi
XML	Extensible Markup Language

Tabella 2: Acronimi e definizioni

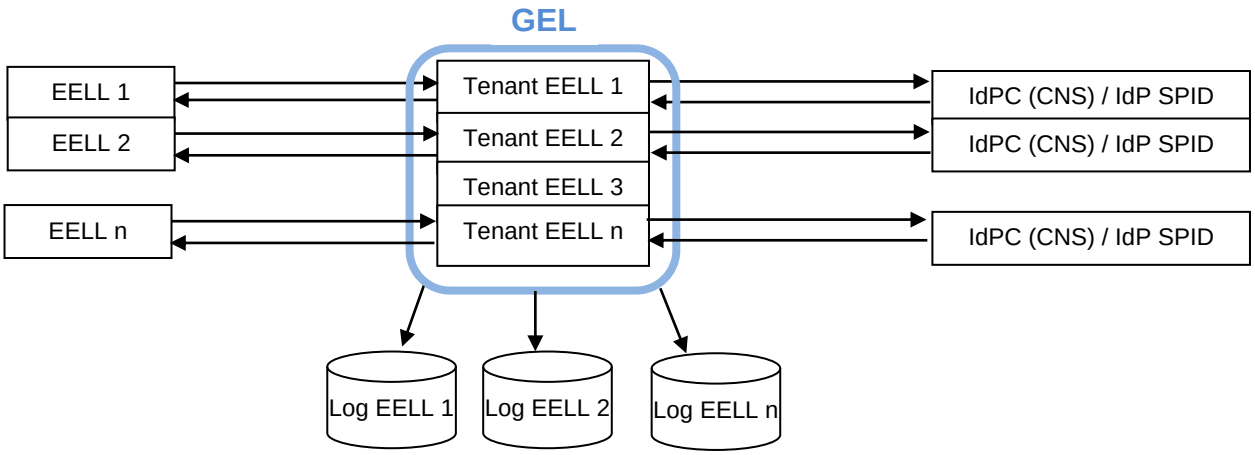
2. Introduzione al servizio GEL

Al fine di supportare l'adesione a SPID degli EELL della Regione Lombardia è stato realizzato un servizio, denominato GEL (Gateway Enti Locali), che è messo a disposizione gratuitamente in modalità SaaS (Software as a Service) presso il Datacenter di Regione Lombardia sito in via Taramelli 26 a Milano.

Il servizio GEL è progettato in architettura "multi-tenant" ovvero in modo che sia possibile creare istanze separate per ogni singolo Ente Locale.

Ogni Ente Locale che intenda avvalersi del servizio GEL sarà quindi autonomo nella possibilità di configurare la propria istanza avvalendosi comunque delle componenti di base comune a tutte le istanze e potendo contare sull'impegno di Regione Lombardia e Lombardia informatica ad adeguare il servizio GEL ad ogni modifica delle regole tecniche emanate da AgID.

La soluzione adottata è descritta sinteticamente di seguito:



Il servizio GEL metterà a disposizione degli enti aderenti una console di gestione con la quale potranno essere svolte tutte le attività di configurazione necessarie. Tramite la stessa console ogni ente potrà prelevare il proprio LOG, previsto dal regolamento attuativo di SPID, per conservarlo attraverso i propri processi di conservazione.

Il servizio GEL è utilizzabile sia dagli enti che hanno già integrato il servizio IdPC, sia da chi non l'ha mai utilizzato ed intende utilizzarlo da ora per adeguarsi alla necessità di identificazione informatica con CNS e SPID prevista dal CAD.

Il servizio GEL espone altresì un tenant di test, condiviso tra tutti gli EELL, agganciato agli IdP SPID di test e fruibile su Internet, per facilitare le attività di integrazione.

Si noti che nel seguito del documento ci si riferirà all'ambiente "di Produzione" anche con la nomenclatura ambiente "di Esercizio".

3. Dettagli tecnici

Questa sezione descrive le operazioni lato SP necessarie per la fruizione degli IdP SPID attraverso il servizio GEL.

L'elemento cardine dell'integrazione è l'avvenuta installazione e configurazione presso il SP di una istanza del prodotto **Shibboleth** (https://shibboleth.net/products/service-provider.html). Si noti che la suite Shibboleth è composta da diversi moduli: per questa specifica integrazione, è necessario e sufficiente il solo modulo "service provider".

Per eventuali approfondimenti su caratteristiche e configurazioni base di questo prodotto, è possibile consultare le sezioni5 e successive.

Per facilitare il percorso di integrazione al servizio GEL, si consiglia di procedere nella lettura del documento in accordo al seguente schema:

- I SP che erano **già integrati** al sistema di autenticazione IdPC di Regione Lombardia tramite la componente **Shibboleth** devono seguire le istruzioni incluse nei capitoli 3 e 4 ;
- I SP che **già integrati** al sistema di autenticazione IdPC di Regione Lombardia tramite la componente "**reference implementation Java**" devono seguire le istruzioni incluse nei capitoli 3, 4, e 7.1 ;
- Ai SP che implementano per la **prima volta** l'integrazione al sistema di autenticazione IdPC è consigliata la lettura dell'intero documento.

3.1 Kit di integrazione al servizio GEL

LISPA fornisce "**kit di integrazione**" specializzati per ogni ambiente reso disponibile all'Ente, quindi un "kit" per l'ambiente di **Integrazione**, ove eseguire i test, ed un "kit" relativo all'ambiente di **Esercizio** vero e proprio.

Il "GEL kit Integrazione" è composto da:

- ✓ il presente documento ;
- ✓ un file (*IdpcGelMetadata_locale_PREIT-internet.xml*) da installare sulla istanza di Shibboleth di visibilità del SP, in accordo alle istruzioni incluse nella sezione 3.4; questo file regola l'ingaggio del tenant di test comune a tutti gli EELL ;
- ✓ un file (*attribute_map.xml*) per facilitare le configurazioni di Shibboleth non già predisposte all'integrazione con IdPC, come illustrato nella sezione 3.4 ;
- ✓ alcune utenze SPID di test afferenti ad almeno un IdP accreditato da AgID ;
- ✓ una coppia di chiavi crittografiche di test (*gel-spid.p12*) da utilizzarsi esclusivamente per i test di integrazione al servizio (tenant di test) ;
- ✓ un "componente aggiuntivo" necessario solamente ai SP precedentemente integrati con il sistema di autenticazione di Regione Lombardia tramite le librerie Java denominate "Reference Implementation".

Il "GEL kit Esercizio" è composto da:

- ✓ un modulo ove specificare alcuni dati relativi all'Ente aderente ;
- ✓ un file (*IdpcGelMetadata_locale.xml*) da installare sulla istanza di Shibboleth di visibilità del SP, in accordo alle istruzioni incluse nella sezione 3.4; questo file regola l'ingaggio del tenant di Produzione dell'EELL ;
- ✓ un tool per la decifratura del file di log ;
- ✓ un tool per la firma dei metadati del SP ;
- ✓ un manuale d'uso della console web dedicata agli EELL.

3.2 Chiavi crittografiche per l'ambiente di Esercizio

Ogni SP deve richiedere, per l'ambiente di Produzione, quantità crittografiche personalizzate, che sono utilizzate per molteplici finalità:

- garantiscono la firma delle richieste di autenticazione verso GEL ;
- nella loro componente pubblica, sono utilizzate per la cifratura dei log di pertinenza del SP ;

- sono utilizzate per la firma del metadata specifico del SP, che va indirizzato ad AgID nell'ambito del processo di abilitazione dell'Ente.

Le chiavi crittografiche sono richieste tramite accesso autenticato ad una particolare web application realizzata da LISPA a beneficio dei SP. Indirizzo internet (URL) e manuale d'utilizzo di questa web application saranno fornite agli Enti interessati.

Una volta che la Certification Authority ha prodotto il materiale crittografico, la consegna di chiavi e certificati avviene nel seguente modo:

- chiave privata e chiave pubblica (entrambi inclusi in un "contenitore" avente formato .p12) vengono inviati via email (PEO) ad un indirizzo indicato dall'Ente durante la fase di richiesta ;
- la passphrase della chiave privata (indispensabile per sbloccare la stessa e consentirne l'uso) viene inviata via SMS ad un numero di telefono mobile indicato dall'Ente durante la fase di richiesta.

3.3 Set di dati richiesti dal SP

Il SP dovrà scegliere quali dati di identità dell'utente intende richiedere all'IdP SPID che governa il processo di autenticazione. Si noti che questi dati saranno un subset rispetto a quelli rilasciati dal servizio GEL ed illustrati nella sezione 3.5. Le possibili combinazioni di dati sono aggregate in "set" autoconsistenti, tra cui il SP dovrà scegliere. Nella sezione 3.5 sono anche evidenziati i campi *obbligatoriamente* restituiti dagli IdP e quelli *opzionalmente* restituiti.

I "set" ad oggi previsti sono i seguenti:

- Set 0: nome, cognome, codice fiscale, email, codice identificativo SPID
- Set 1: nome, cognome, codice fiscale, email, codice identificativo SPID, sesso, data di nascita, luogo di nascita
- Set 2: nome, cognome, codice fiscale, email, codice identificativo SPID, sesso, data di nascita, luogo di nascita, provincia di nascita
- Set 3: nome, cognome, codice fiscale, email, codice identificativo SPID, sesso, data di nascita, luogo di nascita, provincia di nascita, telefono mobile
- Set 4: nome, cognome, codice fiscale, codice identificativo SPID
- Set 5: nome, cognome, codice fiscale, codice identificativo SPID, ragioneSociale, sedeLegale, partitaIVA

Si ricorda quanto previsto dal "Regolamento recante le modalità attuative per la realizzazione dello SPID":

> ***Art. 27 (Uso degli attributi SPID)***
> *I fornitori di servizi, per verificare le policy di sicurezza relativi all'accesso ai servizi da essi erogati potrebbero avere necessità di informazioni relative ad attributi riferibili ai soggetti richiedenti. Tali policy dovranno essere concepite in modo da richiedere per la verifica il* ***set minimo di attributi pertinenti e non eccedenti*** *le necessità effettive del servizio offerto e mantenuti per il tempo strettamente necessario alla verifica stessa, come previsto dall'articolo 11 del decreto legislativo n. 196 del 2003.*

Si consiglia pertanto di scegliere il Set 4, salvo motivate necessità del servizio.

L'indicazione del set scelto andrà inserita nella richiesta di autenticazione creata dalla componente Shibboleth (vedi sezione successiva).

3.4 Configurazione della componente Shibboleth

L'istanza di Shibboleth utilizzata dal SP per l'integrazione ad IdPC va corredata da alcune informazioni peculiari che consentono l'integrazione al servizio GEL.

Il file *shibboleth2.xml* deve contenere le personalizzazioni descritte nel seguito. Alcuni tag vanno valorizzati in accordo alla struttura dell'applicazione, in questi casi è presente una nota a piè pagina esplicativa. Gli altri valori vanno inseriti senza modificarli. Si noti in particolare che Shibboleth viene istruito per lavorare in **SAML2**.

Si noti che, come previsto dall'infrastruttura di autenticazione SAML2, entrambe le controparti (SP e IdP) devono essere dotate di un "metadata", ed ogni entità deve conoscere il "metadata" della controparte.

La configurazione della componente Shibboleth illustrata in questo paragrafo consente, tra le altre cose, al SP di installare sui propri sistemi il "metadata" dell'IdP (il servizio GEL).

L'operazione speculare, ovvero l'installazione sull'IdP del "metadata" del SP, che è *specifico di ogni EELL*, avviene con procedure automatizzate e mediate dalla console di gestione menzionata nella sezione 2.

```
...
<RequestMapper type="Native">
<RequestMap applicationId="default">
<Host name="HOST_NAME" [1] scheme="https" port="443">
<Path name="URL_PROTETTO" [2] applicationId="ID_APP" [3] authType="shibboleth" requireSession="true" exportAssertion="true"/>
</Host>
</RequestMap>
</RequestMapper>

...

<ApplicationOverride id="ID_APP[4]" policyId="default" entityID="https://idpcgel.crs.lombardia.it/gelmetadata/COD_ISTAT" [5]
REMOTE_USER="cf" signing="true" encryption="false">
<Sessions lifetime="28800" timeout="3600" checkAddress="false" handlerURL="URL_PROTETTO/Shibboleth.sso [6]" handlerSSL="true"
exportLocation="/GetAssertion" exportACL="127.0.0.1" idpHistory="false" idpHistoryDays="7" cookieProps="; path=/; secure; HttpOnly">
<SessionInitiator type="SAML2" Location="/Login" entityID="https://idpcgel.crs.lombardia.it/idpcgel" >
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="idpcgel" Version="2.0" IssueInstant="2012-01-01T00:00:00Z"
AttributeConsumingServiceIndex="4" [7]>
<samlp:NameIDPolicy AllowCreate="true" Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
<samlp:RequestedAuthnContext Comparison="exact" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
<saml:AuthnContextClassRef xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
https://www.spid.gov.it/SpidL2oppurehttps://www.spid.gov.it/SpidL3[8]
</saml:AuthnContextClassRef>
</samlp:RequestedAuthnContext>
<samlp:Scoping ProxyCount="1"></samlp:Scoping>
</samlp:AuthnRequest>
</SessionInitiator>
<md:AssertionConsumerService Location="/SAML2/POST" index="1" Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>
</Sessions>
<CredentialResolver type="File">
<Key password="PASSWORD_P12" format="PKCS12">[9]
<Path>PATH_P12</Path>
<Name>Regione_Lombardia</Name>
</Key>
<Certificate password="PASSWORD_P12" format="PKCS12">
<Path>PATH_P12</Path>
</Certificate>
</CredentialResolver>
<MetadataProvider type="Chaining">
<MetadataProvider type="XML" file="PATH_METADATA_GEL"/>[10]
</MetadataProvider>
</ApplicationOverride>
---
```

1 Personalizzare con hostname del servizio
2 Personalizzare con la URL (o sezione del sito) da proteggere con autenticazione
3 Identificativo dell'applicazione (tipicamente una stringa autoesplicativa)
4 Vedi nota 3; si noti che la URL dell'AssertionConsumer del singolo SP è prodotta automaticamente da Shibboleth ed inserita nella AuthRequest
5 Personalizzare la URL indicando il codice ISTAT dell'Ente Locale. Per l'ingaggio del tenant di test utilizzare la seguente valorizzazione: https://idpcgel.integrazione.lispa.it/gelmetadata/test
6 Path relativo, a partire dalla context root dell'applicazione, dove è situata la risorsa da proteggere (es. /my_site/protected/Shibboleth.sso)
7 Indice che referenzia la sezione del metadata della componente "GEL" dove è contenuto il set di valori di interesse dell'applicazione – si veda paragrafo 3.3
8 Specificare il livello di autenticazione SPID richiesto (può essere L2 o L3 - utilizzare esclusivamente una delle due stringhe indicate)
9 LISPA fornirà al SP una coppia di chiavi, necessarie per la firma delle richieste di autenticazione. In questa sezione andrà configurato il path assoluto del file (.p12) ricevuto via email da LISPA, corredato della relativa password (ricevuta via sms). Utilizzare invece il file "gel-spid.p12" per l'ingaggio del tenant di test (ambiente di Integrazione)
10 LISPA fornirà al SP il file di "metadata" associato alla componente "GEL". Questo file, denominato "IdpcGelMetadata_locale.xml" per l'ambiente di Produzione, va referenziato in path assoluto. Utilizzare invece il file "IdpcGelMetadata_locale_PREIT-internet.xml" per l'ingaggio del tenant di test (ambiente di Integrazione)

3.5 Asserzione di identità rilasciata al SP

L'asserzione di identità rilasciata dalla componente GEL viene "mappata" in *header http* attraverso una operazione di configurazione entro la componente Shibboleth. Si sottolinea che alcune valorizzazioni dei tag sono peculiari nel caso in cui l'asserzione sia nativamente emessa da un IdP SPID:

Campo asserzione IdPC-GEL	Campo asserzione SPID	Note
Dati obbligatori e tipicamente valorizzati		
Nome	Name	
Cognome	familyName	
codice Fiscale	fiscalNumber	Privato del preambolo "TINIT:"
Sesso	Gender	
data Nascita	dateOfBirth	
luogo Nascita	placeOfBirth	Previa transcodifica Belfiore
provincia Nascita	countryOfBirth	
stato Nascita	-	calcolato da codiceFiscale
tipo Autenticazione	*derivato da AuthnContext, solo se status=Success*	Può valere uno dei seguenti: IDPC_AUTHENTICATION_SMARTCARD, IDPC_SPID_L2, IDPC_SPID_L3
identificativo Utente	spidCode	
nome Utente	spidCode	duplicato per compatibilità
livello Verifica	*derivato da AuthnContext, solo se status=Success*	Inserito per compatibilità con integrazioni già in essere. Può valere uno dei seguenti: 50 (se L1), 70 (se L2), 90 (se L3)
Dati tipicamente opzionali		
emailAddress	email	
Cellular	mobilePhone	
ragione Sociale	companyName	
sede Legale	registeredOffice	
partitaIVA	ivaCode	Si noti che gli IdP SPID possono gestire identità di persone giuridiche. I SP interessati devono quindi prevedere questi casi e predisporsi per gestirle
docIdentita	idCard	
scadDocIdentita	expirationDate	
domicilio Fisico	address	
domicilioDigitale	digitalAddress	
CNS_CARTA_REALE	-	Sempre NON_DISPONIBILE
CNS_ISSUER	-	Sempre NON_DISPONIBILE
CNS_SUBJECT	-	Sempre NON_DISPONIBILE
origineDatiUtente	-	SPID, ARCHIVIO_CARTE (nel caso di utilizzo di CRS RL), SMARTCARD (nel caso di utilizzo di CNS)
statoValidazioneProfiloUtente	-	Sempre NON_DISPONIBILE
idComuneRegistrazione	-	Sempre 03 (mantenuto per compatibilità con integrazioni legacy)
indirizzo_IP_corrente	-	IP address della postazione utilizzata dall'utente per l'autenticazione corrente

Al fine di facilitare la configurazione delle istanze di Shibboleth, LISPA fornisce ai SP il file *attribute-map.xml* che consente la copia dei dati di asserzione nei http header.

4. Istanza e procedure di test del GEL

4.1 Piattaforme di test

Al fine di permettere l'esecuzione di test in fase di sviluppo e pre-esercizio, è messa a disposizione degli EELL della Regione Lombardia una istanza di test del GEL già configurata allo scopo.

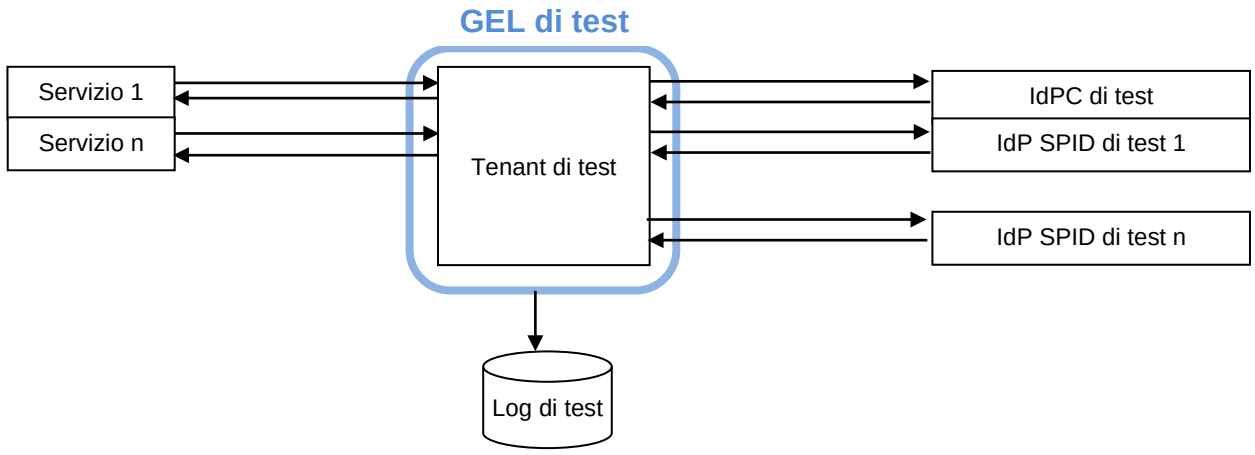
L'istanza di test, funzionalmente equivalente, permette di testare l'integrazione di uno o più servizi con l'istanza IdPC di test e con le istanze di test degli idP SPID che le hanno messe a disposizione.

È importante notare che, trattandosi di circuiti di test e non di circuiti reali, non è possibile utilizzare identità di cittadini reali, bensì nei circuiti di test è possibile unicamente utilizzare:

- Carte CRS/CNS di test (ovvero intestare a cittadini fittizi)
- Identità SPID di test fornite dagli IdP

Lombardia Informatica è in possesso di un certo numero di "credenziali SPID di test" che può distribuire agli Enti.

La soluzione adottata è descritta sinteticamente di seguito:



L'istanza di test del GEL è resa disponibile su Internet ad una URL che verrà comunicata agli Enti interessati.

4.2 Procedura per i test di integrazione

Per utilizzare l'istanza di test, è necessario configurare Shibboleth in accordo con le specifiche indicate nella sezione 3.4 adottando le seguenti accortezze:

- valorizzare **COD_ISTAT** con ***test***;
- specificare autenticazione livello L2 ;
- copiare sul file system ospite, in una locazione a scelta del SP, il file ***gel-spid.p12*** incluso nel "Kit di integrazione" fornito da LISPA ;
- specificare in **PATH_P12** la locazione del file ***gel-spid.p12***;
- valorizzare **PASSWORD_P12** con la stringa ***siss***

A questo punto, la componente Shibboleth è in grado di richiedere autenticazioni di test verso la componente GEL dispiegata allo scopo nel DataCenter di LISPA.

Al fine di validare l'integrazione, si consiglia di:

- effettuare autenticazioni utilizzando credenziali SPID di test ;
- se in possesso di CNS di test, effettuare autenticazioni con smartcard ;
- a valle di ogni autenticazione, e con qualsiasi strumento utilizzato, verificare che i dati valorizzati nell'http header di Shibboleth, ovvero messi a disposizione dell'applicazione, siano coerenti con il set atteso (ad esempio, che siano presenti e correttamente valorizzati: nome, cognome, codice fiscale) ;
- verificare che la selezione del Logout reso disponibile dall'applicazione integrata comporti effettivamente la richiesta di una nuova autenticazione nel momento in cui l'utente prova ad accedere nuovamente ad un'area protetta.

5. Architettura del Reverse Proxy Shibboleth SP

L'infrastruttura di gestione degli accessi a Service Provider realizzata mediante il Reverse Proxy applicativo Shibboleth presenta un'architettura costituita da due sotto-sistemi principali, dei quali il primo è un web server di tipo Apache HTTP Server in versione 2.2.12 o successiva e il secondo è un server Shibboleth SP, anch'esso in versione 2.5.5 o successiva[11]. Tali sotto-sistemi comunicano tra loro attraverso uno specifico modulo di Apache denominato "**mod_shib**", in grado di intercettare l'accesso ad una o più risorse (URL) protette e attivare i componenti di Shibboleth SP per iniziare il processo di autenticazione. Nella figura successiva è illustrata l'architettura complessiva.

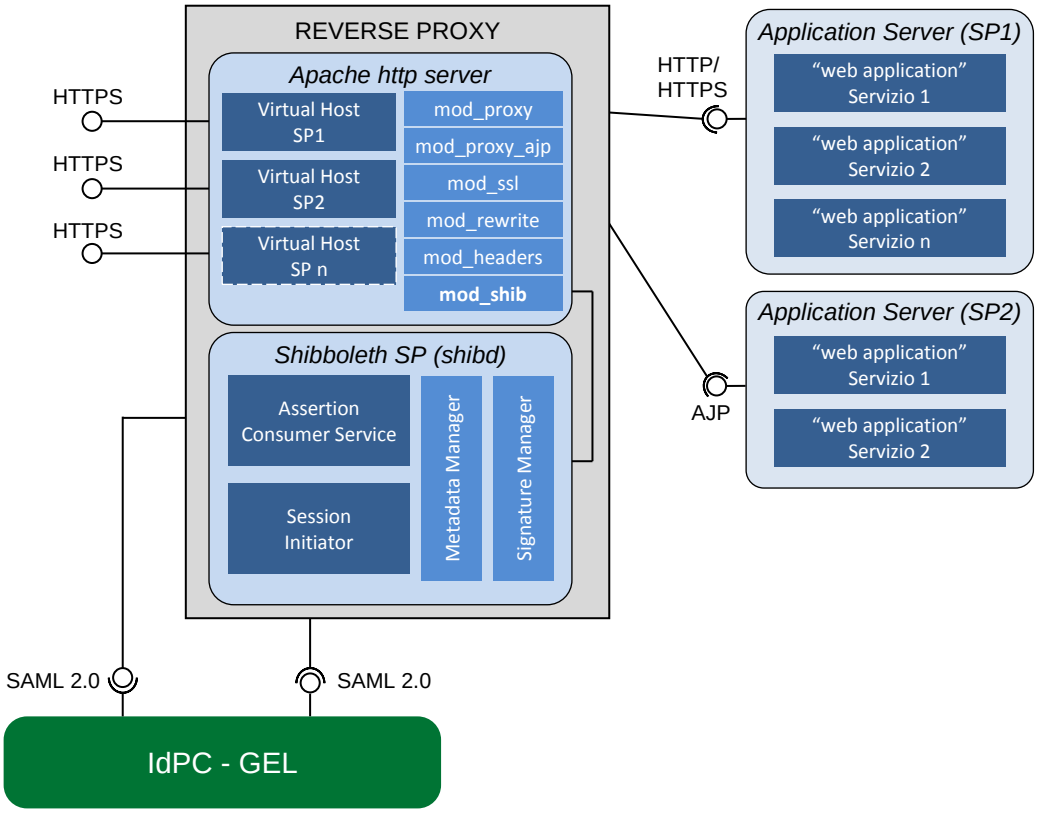


Figura 1 - Architettura del Reverse Proxy e interazioni con SP e IdP

Il Reverse Proxy indicato in figura si caratterizza come entità a se stante che può essere dispiegata indipendentemente dai Service Provider che dovrà proteggere e dai sistemi di autenticazione che si troverà ad utilizzare. In questo modo il Reverse Proxy può essere dispiegato in un contesto di DMZ in modo da essere accessibile da parte dell'utenza che deve accedere ai Service Provider. Questi ultimi possono invece trovarsi in una rete interna, protetta. Attraverso questa architettura, infatti, i Service Provider che sarebbero normalmente accessibili direttamente perché espongono un'interfaccia di tipo HTTP o quelli accessibili su protocollo AJP mediante l'aggiunta di un semplice web server Apache con utilizzo del modulo "mod_proxy_ajp" vengono schermati dalla presenza del web server Apache dispiegato presso il Reverse Proxy, al quale pertanto è importante che giungano tutte le richieste di servizio.

L'accesso diretto ai Service Provider retrostanti deve quindi essere impedito e gli indirizzi originariamente utilizzati per l'accesso devono essere rimappati in modo che corrispondano ad endpoint configurati presso il web server Apache del Reverse Proxy, secondo le modalità che saranno illustrate nel seguito.

Il server Shibboleth SP rappresentato in figura è in grado di operare utilizzando il protocollo SAML 2.0, adattandosi così ad interagire con Identity Provider quali **il sistema di autenticazione IdPC di Regione Lombardia**.

Nel seguito del capitolo saranno descritti in maggiore dettaglio i sottosistemi relativi al web server Apache e al server Shibboleth SP.

[11] Il server Shibboleth SP è in grado di integrarsi anche con il server web Microsoft Internet Information Services (IIS). La trattazione tuttavia farà riferimento al caso dell'integrazione con Apache HTTP Server che è parte costitutive dell'infrastruttura del Reverse Proxy.

5.1 Apache HTTP Server

Allo scopo di gestire l'accesso ai vari Service Provider che si intende proteggere tramite Reverse Proxy, il web server Apache HTTP Server deve essere dotato di alcuni moduli aggiuntivi, le cui funzioni sono illustrate di seguito:

-***mod_proxy***: consente di realizzare la funzione di reverse proxy esponendo gli endpoint URL necessari per la fruizione dei vari servizi applicativi e rimappandoli in modo che siano indirizzati agli specifici Application Server che ospitano i vari Service Provider;

- ***mod_proxy_ajp***: completa le funzionalità del modulo mod_proxy, consentendo di inoltrare le richieste pervenute ad Apache su protocollo HTTP o HTTPS utilizzando il protocollo AJP verso gli Application Server di destinazione;

- ***mod_shib***: consente al web server di proteggere le risorse (URL) predefiniti ed attivare i componenti del server Shibboleth SP per attivare il processo di autenticazione;

- ***mod_ssl***: utilizzato per mettere a disposizione dei client degli end-point HTTPS;

- ***mod_rewrite***: utilizzato per effettuare degli adattamenti sui messaggi SAML generati dal server Shibboleth quando è necessario interagire con Identity Provider conformi alle specifiche SAML.

Il web server inoltre gestisce l'accesso ai Service Provider in modo isolato, l'uno rispetto all'altro, mediante l'impostazione di altrettanti Virtual Host. In tale modo è possibile delimitare e rendere indipendente la configurazione dedicata a gestire gli accessi verso un erogatore di servizi da quelle destinate ai rimanenti ed offrire ad esempio l'accesso in HTTP per uno e in HTTPS per un altro. In questo modo è altresì possibile mantenere gli hostname e le porte originali definiti per l'accesso diretto ai vari Service Provider, prima della loro integrazione con il Reverse Proxy.

5.2 Shibboleth SP

Il server Shibboleth SP, realizzato dal daemon "shibd" si compone dei seguenti elementi:

- **Session Initiator** – questo componente agisce come un filtro di autenticazione: ha infatti il ruolo di intercettare le richieste dirette al servizio applicativo e verificare il contesto di autenticazione per la sessione utente. Nel caso non esista un'autenticazione pregressa riutilizzabile, viene generata una nuova richiesta destinata ad un Identity Provider o altra infrastruttura di autenticazione. Per produrre tale richiesta, il componente accede alla propria configurazione dove sono elencate informazioni quali le modalità di autenticazione richieste e il protocollo SAML da utilizzare. È funzionalmente equivalente alla componente **AccessCheck**;
- **Assertion Consumer Service** – questo componente ha il compito di ricevere i messaggi SAML Response prodotti dall'infrastruttura di autenticazione (GEL) contenenti un'asserzione con uno statement di autenticazione. Assertion Consumer Service estrae i dati utente dal messaggio di Response e li mette a disposizione del servizio applicativo. Al pari del componente Session Initiator, anche Assertion Consumer Service si avvale di servizi di supporto realizzati dai componenti "Metadata Manager" e "Signature Manager". Il primo è utilizzato per accedere ai metadati del soggetto, interno all'infrastruttura di autenticazione, che ha inviato il messaggio SAML Response, allo scopo di verificare la sua firma digitale. Il secondo è acceduto durante il processo di verifica vera e propria della firma. Infine, il componente Assertion Consumer Service è in grado di estrarre i dati relativi all'utente autenticato e metterli a disposizione dell'erogatore in un formato neutro, in particolare attraverso un set di header HTTP appositamente definiti nella configurazione. Le chiavi, con le quali l'Assertion Consumer Service verifica la firma dell'asserzione prodotta dall'IdPC, sono censite all'interno del metadata specificato nel tag <MetadataProvider> del file *shibboleth2.xml*.**Questo file va modificato in modo da censire i certificati digitali utilizzati dal sistema di autenticazione di Regione Lombardia durante l'operazione di firma delle asserzioni. Si consulti a tal proposito la sezione predisposta allo scopo nel documento.**È possibile censire nello stesso metadata entrambe le root, oppure utilizzare due metadata differenti, uno che censisce la chiave reale e uno la chiave virtuale (ad esempio *metadata_idpcrl_reale.xml* e *metadata_idpcrl_virtuale.xml*, piuttosto che un unico *metadata_idpcrl.xml*). Si ricorda che il sistema di autenticazione di Regione Lombardia firma le proprie asserzioni con certificati digitali emessi da CA differenti. È questo il motivo per cui, a livello di Assertion Consumer Service, è possibile "trustare" una sola root (ad esempio per accettare le sole autenticazioni avvenute con CNS reali) piuttosto che entrambe. Si presti particolare attenzione al fatto che **tale configurazione ècomunea tutte le applicazioni** (Service Provider) protetti dalla singola istanza di Shibboleth ;
- **Signature Manager** – È un componente che ha lo scopo di fornire le funzionalità atte a firmare digitalmente i messaggi in standard SAML prodotti dai vari soggetti interoperanti, prima del loro invio. Consente inoltre di verificare la firma dei messaggi SAML a valle della loro ricezione. Ciascun componente produttore di messaggi SAML, pertanto, deve essere dotato di una chiave privata per apporre le firme e di una corrispondente chiave pubblica contenuta in un certificato per consentire agli altri componenti di verificare la firma apposta. Le chiavi pubbliche sono reperibili direttamente consultando i metadati di ciascun soggetto.

- **Metadata Manager** – questo componente ha lo scopo di fornire le funzionalità di accesso ai servizi di pubblicazione delle strutture metadati esposte dai vari componenti dell'infrastruttura di autenticazione. È in grado di interpretare i metadati, distinguendo quelli propri delle varie tipologie di soggetti (es. Identity Provider, Service Provider ecc.) e di estrarre le informazioni utili agli scenari d'interazione. Tale componente realizza anche il servizio di pubblicazione dei metadati contattabile remotamente via HTTP da qualunque altro soggetto.

5.3 Interfacciamento verso soggetti esterni

Come illustrato nella figura precedente, il sistema Reverse Proxy presenta interfacce verso vari tipi di soggetti esterni e in particolare verso i seguenti:

- client dei servizi applicativi che si vogliono proteggere tramite autenticazione
- application server che ospitano i servizi applicativi di cui al punto precedente
- infrastrutture di autenticazione da attivare durante l'accesso ai servizi

Nel seguito vengono illustrate brevemente tali interfacce, per ciascuno dei soggetti elencati. I dettagli della configurazione dei vari sotto-sistemi (Apache HTTP Server e Shibboleth SP) al fine di realizzare tali interfacce saranno forniti nel capitolo successivo.

5.3.1 Interfacce verso i fruitori di servizi applicativi

Nei confronti dei fruitori, o client, di servizi applicativi il Reverse Proxy può offrire interfacce di tipo HTTP o HTTPS, secondo le necessità dello specifico servizio che si intende proteggere mediante autenticazione. Tali interfacce sono realizzate da Apache HTTP Server, e in particolare dai vari Virtual Host configurati al suo interno.

Il generico client deve essere messo in condizioni di indirizzare il Virtual Host opportuno, in funzione del servizio richiesto, ad esempio registrando il nome di dominio relativo al Service Provider e facendolo puntare al Reverse Proxy. Si rimanda di nuovo al capitolo successivo per maggiori dettagli al riguardo.

5.3.2 Interfacce verso gli erogatori di servizi applicativi

Il Reverse Proxy inoltra le richieste, giunte dai vari client dai quali viene acceduto, verso gli opportuni erogatori dei servizi applicativi, dopo che il processo di autenticazione dell'utente richiedente si è concluso positivamente. La comunicazione tra Reverse Proxy e Service Provider può avvenire attraverso i protocolli standard HTTP e HTTPS, qualora l'application server retrostante esponga direttamente un endpoint di tale tipologia.

Tale situazione è tipica di Service Provider preesistenti all'integrazione con il Reverse Proxy e che intendono mantenere la modalità di accesso originaria anche dopo essersi integrati. In alternativa, il Reverse Proxy è in grado di interagire con un Service Provider anche utilizzando il protocollo AJP 1.3, mediante il quale il Service Provider non è direttamente contattabile da client quali normali browser web.

Esempi di Application Server in grado di esporre endpoint AJP sono Apache Tomcat e Jboss. Dalla parte del Reverse Proxy, il Reverse Proxy utilizza Apache HTTP Server con i moduli "mod_proxy" e "mod_proxy_ajp" per supportare rispettivamente i protocolli HTTP/HTTPS e AJP per interagire con i Service Provider a valle.

5.3.3 Interfacce verso le infrastrutture di autenticazione e Identity Provider

Il Reverse Proxy è in grado di interfacciarsi con infrastrutture di autenticazione conformi alle specifiche SAML. Per tale interfacciamento, il Reverse Proxy utilizza il server Shibboleth SP. In particolare per l'invio delle richieste di autenticazione viene coinvolto il componente "Session Initiator", mentre le risposte di autenticazione vengono indirizzate al componente "Assertion Consumer Service".

6. Configurazione del Reverse Proxy

In questo capitolo vengono descritte le fasi di configurazione da svolgere sui componenti del Reverse Proxy al fine di integrare con esso un nuovo Service Provider e proteggerne quindi l'accesso mediante autenticazione. Le attività di configurazione che interessano i diversi tipi di Service Provider supportati saranno invece oggetto del prossimo capitolo.

6.1 Configurazione di Apache HTTP Server

Come presentato nel capitolo precedente, presso l'istanza Apache HTTP Server dispiegata presso il Reverse Proxy sono configurati vari Virtual Host, ciascuno dei quali dedicato all'integrazione di uno specifico Service Provider posto in posizione logica retrostante al Reverse Proxy.

6.1.1 Attività di configurazione generali

Indipendentemente dai vari Virtual Host che dovranno essere creati, Apache HTTP Server necessita di alcune direttive base di configurazione atte a creare le condizioni operative comuni per attivare correttamente il server Shibboleth SP per qualsiasi Service Provider che verrà integrato. Tali direttive di configurazione vengono pertanto impostate una volta sola in fase di configurazione del Reverse Proxy e valgono per tutti i Service Provider che saranno integrati successivamente. In particolare, tali direttive sono le seguenti:

- all'interno del file di configurazione generale di Apache *httpd.conf* impostare il valore della direttiva *ServerName* con il nome dell'host del Reverse Proxy, ad esempio:

```
ServerName SERVICEPROVIDER.DOMAIN.COM
```

- caricare il modulo *mod_headers* per abilitare l'utilizzo degli header HTTP. Shibboleth SP infatti crea degli header HTTP ad-hoc per inviare al servizio applicativo il valore degli attributi del profilo utente ricevuti durante il processo di autenticazione SAML:

```
LoadModule headers_module modules/mod_headers.so
```

- abilitare la direttiva *UseCanonicalName* - se impostata ad **off**, il funzionamento dell'intera architettura è basato esclusivamente sull'utilizzo di IP address fisici e non su URL contenenti nomi logici; se impostata ad **on**, viene consentito l'utilizzo di nomi logici (valorizzazione raccomandata):

```
UseCanonicalName on
```

- caricare il modulo *mod_shib* per abilitare la connessione con il server Shibboleth SP:

```
LoadModule mod_shib <PATH_TO_SHIBBOLETH_SP>/lib/shibboleth/mod_shib_22.so
```

  il percorso su file system del modulo *mod_shib* dipende dall'installazione del server Shibboleth SP fatta. Ad esempio, su sistemi Unix tale percorso potrebbe essere "/usr/lib/shibboleth/mod_shib_22.so";

- definire le seguenti regole di URL rewriting, valide per tutti i Service Provider che si dovranno integrare:

```
RewriteEngine On
RewriteOptions Inherit

RewriteCond%{QUERY_STRING} idpUrl=([^?]+)(\?)([^?]+)&shire=([^&]+)&(.*)target=(.*)&providerId=.*
RewriteRule (.*) %1%2%3&TARGET=%4?target=%6 [R,NE,L]

RewriteCond %{QUERY_STRING} idpUrl=([^?]+)&shire=([^&]+)&(.*)target=(.*)&providerId=.*
RewriteRule (.*) %1?TARGET=%2?target=%4 [R,NE,L]

RewriteLog "rewrite.log"
RewriteLogLevel 3
```

Inoltre, nel Web Server Apache si intendono già caricati i moduli *mod_proxy, mod_proxy_ajp, mod_rewrite* e *mod_ssl* che sono normalmente presenti in molte distribuzioni standard di Apache. Se così non fosse è possibile abilitarli inserendo le apposite direttive "LoadModule" come quelle illustrate in precedenza.

6.1.2 Integrazione di un nuovo Service Provider presso Apache HTTP Server

Dopo aver effettuato gli interventi descritti in precedenza oppure dopo aver verificato che tale configurazione è già attiva, è possibile procedere con le attività mirate ad integrare un nuovo Service Provider.

Ciò comporta pertanto come prima cosa la creazione di un nuovo Virtual Host presso il web server. A questo scopo è possibile aggiungere alla configurazione un frammento, posizionabile in un file separato rispetto al file di configurazione principale (httpd.conf), che deve tuttavia trovarsi nella cartella di nome “conf.d”. Le modalità secondo cui è possibile creare un nuovo Virtual Host dipendono dal fatto che tale Virtual Host sarà acceduto su protocollo HTTP oppure HTTPS; tali alternative sono analizzate nel seguito.

6.1.2.1 Virtual Host su protocollo HTTP

Nel caso di utilizzo del protocollo HTTP è possibile utilizzare la modalità “name-based” per i Virtual Host in Apache, secondo cui all’host fisico del Reverse Proxy vengono assegnati più hostname alternativi, tanti quanti sono quelli ai quali devono essere acceduti i vari Service Provider protetti. È quindi possibile mantenere per il Reverse Proxy un unico indirizzo IP, al quale far corrispondere, su DNS, i vari hostname. Per abilitare tale modalità è necessario che nel file di configurazione principale di Apache HTTP Server (*httpd.conf*) sia riportata la seguente direttiva:

```
NameVirtualHost       *:80
```

Assumendo che venga utilizzata la porta HTTP default (80), in questa modalità il frammento di definizione del Virtual Host è il seguente:

```
<VirtualHost *:80>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog   logs/SERVICEPROVIDER.DOMAIN.COM_error.log
        CustomLog  logs/SERVICEPROVIDER.DOMAIN.COM_access.log

LogLevel warn

ProxyPass         /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPassReverse /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH

ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

<Location /WEBAPP/PROTECTED_PATH>
        AuthType shibboleth
        ShibRequireSession On
        ShibUseHeaders On
        require shibboleth
</Location>

<IfModule mod_alias.c>
       <Location /shibboleth>
              Allow from all
       </Location>
       Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
       Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
</IfModule>

RewriteEngine On
RewriteOptions Inherit

RewriteCond %{REQUEST_METHOD} GET
RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]

RewriteLog " rewrite.log"
RewriteLogLevel 3


</VirtualHost>
```

In esso occorre operare le seguenti sostituzioni:

- sostituire la stringa “SERVICEPROVIDER.DOMAIN.COM” con l’hostname del Service Provider come esposto verso i fruitori esterni;
- sostituire la stringa “WEBAPP” con il nome dell’applicazione presso l’Application Server di destinazione. Nel caso il Service Provider risponda sul path radice sarà sufficiente specificare semplicemente il percorso minimo (“/”);
- sostituire la stringa “PROTO” con il protocollo utilizzato dall’Application Server che ospita il Service Provider da proteggere. I protocolli supportati sono “http”, “https” e “ajp”;
- sostituire la stringa “INTERNAL.SERVICEPROVIDER.DOMAIN.COM” con l’hostname dell’Application Server che ospita il Service Provider da proteggere;
- sostituire la stringa “PORT” con il numero di porta dell’endpoint presso l’Application Server che ospita il Service Provider da proteggere. Naturalmente, è necessario esplicitare il numero di porta soltanto nel caso in cui esso differisca da quello di default per il protocollo;
- sostituire la stringa “INTERNAL_PROTECTED_PATH” con il percorso dell’applicazione del Service Provider da proteggere. Come nel caso della stringa “WEBAPP”, anche in questo caso se sull’Application Server viene utilizzato il percorso radice è sufficiente indicare unicamente la stringa “/”;
- sostituire la stringa “PROTECTED_PATH” con il percorso, presso il Service Provider oggetto dell’integrazione, delle risorse che devono essere protette tramite autenticazione; questo consente di evitare di proteggere qualunque percorso presso il Service Provider, e di specificare solo quelli realmente necessari; qualora gli insiemi di risorse da proteggere fossero più di uno, sarà possibile includere ulteriori elementi <Location> simili a quello riportato, uno per ciascun percorso;
- sostituire la stringa “SHIBBOLETH_INSTALL_PATH” con il percorso assoluto su file system della cartella ove è stato installato il server Shibboleth. Tale percorso è utilizzato per localizzare il logo e il CSS di default dell’installazione Shibboleth. Tali elementi sono visualizzati nelle pagine di errore prodotte dal Reverse Proxy ed è possibile sostituirli con altri personalizzati per ciascun Virtual Host.

Si osserva che entrambe le stringhe WEBAPP e PROTECTED_PATH sopra riportate sono le stesse che è necessario riportare anche nella configurazione del server Shibboleth SP come sarà illustrato nella sez. 6.2.1.

6.1.2.2 Virtual Host su protocollo HTTPS

Considerato che Service Provider deve essere esposto tramite Reverse Proxy su protocollo HTTPS, è necessario disporre della chiave privata e del relativo **certificato** con chiave pubblica, **preferibilmente rilasciato da una Certification Authority Omniroot**, che saranno usati per stabilire la connessione SSL/TLS tra il fruitore del servizio e il Virtual Host presso il Reverse Proxy.

Si noti che, nel caso di Service Provider aderente all'infrastruttura IdPC, è questa la **modalità di esposizione richiesta**, in modo da cifrare - tramite il canale SSL/TLS - i dati di asserzione rilasciati dall'Identity Provider che transitano dal browser utente.

In tale situazione, Apache HTTP Server non può conoscere l’hostname – e quindi il Virtual Host – richiesto dal client fino a che l’handshake SSL non si è completato, e non può completare tale handshake finché non ha presentato il certificato X.509 corrispondente al corretto Virtual Host. Per tale motivo risulta normalmente impossibile per i Virtual Host HTTPS utilizzare la modalità name-based. Esistono tuttavia almeno due modi per mantenere tale modalità, in situazioni particolari, come descritto nel seguito.

Modalità IP-based

Utilizzando la modalità IP-based per definire i Virtual Host, comporta la necessità di assegnare un nuovo indirizzo IP pubblico al Reverse Proxy, a cui far corrispondere mediante DNS l’hostname del Service Provider che deve essere integrato. Il Reverse Proxy si troverebbe così a disporre di tanti indirizzi IP contattabili dai fruitori, quanti sono i Service Provider integrati con esso. In questa modalità il frammento riportato nella sez. 6.1.2.1 deve essere modificato come segue:

```
<VirtualHost IP_SERVICE_PROVIDER:443>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_error_log
        CustomLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_access_log common

LogLevel warn

SSLEngine on
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
SSLCertificateFile SERVICEPROVIDER_SSL_CERTIFICATE_FILE_PATH
SSLCertificateKeyFile SERVICEPROVIDER_SSL_CERTIFICATE_KEY_PATH
SSLVerifyDepth 10
SSLOptions +ExportCertData +StdEnvVars

ProxyPass         /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPassReverse /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH

ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

<Location /WEBAPP/PROTECTED_PATH>
        AuthType shibboleth
        ShibRequireSession On
        ShibUseHeaders On
        require shibboleth
</Location>

<IfModule mod_alias.c>
       <Location /shibboleth>
              Allow from all
       </Location>
       Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
       Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
</IfModule>

RewriteEngine On
RewriteOptions Inherit

RewriteCond %{REQUEST_METHOD} GET
RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]
RewriteLog " rewrite.log"
RewriteLogLevel 3

</VirtualHost>
```

Per il significato dei parametri che devono essere sostituiti in tale frammento si rimanda alla sez.6.1.2.1. Rispetto al frammento già descritto in precedenza, si evidenziano unicamente i seguenti parametri aggiuntivi:

- Sostituire la stringa “IP_SERVICE_PROVDER” con l’indirizzo IP pubblico assegnato al Reverse Proxy e dedicato alle richieste destinate al Service Provider oggetto dell’integrazione; tale indirizzo IP dovrà essere associato, mediante DNS, all’hostname specificato dalla stringa “SERVICEPROVIDER.DOMAIN.COM”;
- sostituire la stringa “SERVICEPROVIDER_SSL_CERTIFICATE_FILE_PATH” con il percorso, relativo alla cartella contenente i file di configurazione di Apache HTTP Server, del file contenente il certificato SSL che deve essere utilizzato per l’endpoint HTTPS del Virtual Host; il certificato deve essere codificato in formato PEM (Base64);
- sostituire la stringa “SERVICEPROVIDER_SSL_CERTIFICATE_KEY_PATH” con il percorso, relativo alla cartella contenente i file di configurazione di Apache HTTP Server, del file contenente la chiave privata che deve essere utilizzata per l’endpoint HTTPS del Virtual Host; la chiave deve essere codificata in formato PEM (Base64).

Modalità name-based con condivisione del dominio

Nel caso in cui i vari Virtual Host HTTPS che si devono creare condividono lo stesso dominio (es. *sp1.domain.com*, *sp2.domain.com*, ecc.), è possibile adottare la modalità *name-based* per i Virtual Host, in modo non dissimile da quanto già presentato nella sezione relativa. Per ovviare al problema presentato sopra, relativamente all’impossibilità per Apache HTTP Server di determinare certificato SSL da utilizzare, corrispondente al Virtual Host corretto, è necessario far uso di certificati di tipo wildcard (nell’esempio, certificati con *CN=*.domain.com*) così da adattarsi a tutti i Virtual Host

dello stesso dominio. In questa modalità, pertanto, il frammento di configurazione di Apache interessato viene modificato come segue:

```
NameVirtualHost *:443
<VirtualHost *:443>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_error_log
        CustomLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_access_log common

LogLevel warn

SSLEngine on
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
SSLCertificateFile SERVICEPROVIDER_WILDCARD_SSL_CERTIFICATE_FILE_PATH
SSLCertificateKeyFile SERVICEPROVIDER_WILDCARD_SSL_CERTIFICATE_KEY_PATH
SSLVerifyDepth 10
SSLOptions +ExportCertData +StdEnvVars

ProxyPass         /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPassReverse /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH

ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

<Location /WEBAPP/PROTECTED_PATH>
        AuthType shibboleth
        ShibRequireSession On
        ShibUseHeaders On
        require shibboleth
</Location>

<IfModule mod_alias.c>
       <Location /shibboleth>
              Allow from all
       </Location>
       Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
       Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
</IfModule>

RewriteEngine On
RewriteOptions Inherit

RewriteCond %{REQUEST_METHOD} GET
RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]
RewriteLog " rewrite.log"
RewriteLogLevel 3

</VirtualHost>
```

Si noti la differenza rispetto al caso precedente, caratterizzata dal fatto che non è necessario esplicitare l'indirizzo IP del server. Per gestire l'handshake SSL iniziale, infatti, Apache utilizza il primo Virtual Host elencato, il quale specifica il certificato wildcard. Successivamente Apache è in grado di selezionare il Virtual Host corretto in base al nome del server specificato nella richiesta HTTP. Essendo la configurazione pressoché identica al caso precedente, non vengono qui dettagliati nuovamente tutti i parametri di configurazione sui quali è possibile intervenire.

Modalità name-based con Server Name Indication (SNI)

In alternativa alle due modalità presentate sopra, ne esiste una terza che fa uso della modalità SNI (Server Name Indication) definita dal documento RFC3546,**Errore. L'origine riferimento non è stata trovata.** che costituisce u n'estensione al protocollo TLS per lo scambio dei nomi dei server in fase di setup delle connessioni sicure su protocollo HTTP. Si osserva che tale modalità non è supportata dalla totalità dei web server esistenti, in particolar modo da quelli realizzati prima dell'introduzione di tali estensioni al protocollo TLS definite dal documento RFC menzionato**Errore. L 'origine riferimento non è stata trovata.**. Nel caso di Apache HTTP Server, è necessario utilizzare una versione 2.2.12 o successiva, compilata con un supporto della libreria OpenSSL 0.9.8f o successiva, nella quale siano state abilitate le estensioni TLS (abilitate per default a partire dalla versione 0.9.8j).

Utilizzando una versione idonea di Apache HTTP Server è possibile definire i Virtual Host per connessioni HTTPS in modalità name-based in modo del tutto simile a quanto descritto per il caso dei Virtual Host per connessioni HTTP. In particolare, il frammento di configurazione di Apache in questo caso è il seguente:

```
NameVirtualHost *:443


<VirtualHost *:443>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_error_log
        CustomLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_access_log common

LogLevel warn

SSLProxyEngine on
SSLEngine on
SSLProtocol +all -SSLv2 -SSLv3
SSLCipherSuite RC4-SHA:AES128-SHA:HIGH:MEDIUM:!aNULL:!MD5G
SSLHonorCipherOrder on
SSLCertificateFile SERVICEPROVIDER_SSL_CERTIFICATE_FILE_PATH
SSLCertificateKeyFile SERVICEPROVIDER_SSL_CERTIFICATE_KEY_PATH
SSLVerifyDepth 10
SSLOptions +ExportCertData +StdEnvVars

ProxyPass         /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPassReverse /WEBAPPPROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH

ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

<Location /WEBAPP/PROTECTED_PATH>
        AuthType shibboleth
        ShibRequireSession On
        ShibUseHeaders On
        require shibboleth
</Location>

<IfModule mod_alias.c>
       <Location /shibboleth>
              Allow from all
       </Location>
       Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
       Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
</IfModule>

RewriteEngine On
RewriteOptions Inherit

RewriteCond %{REQUEST_METHOD} GET
RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]
RewriteLog " rewrite.log"
RewriteLogLevel 3

</VirtualHost>
```

Si noti che le specifiche su `SSLProtocol` e `SSLCipherSuite` sono valorizzate coerentemente con i requisiti di sicurezza oggi necessari. È compito del SP applicare adeguamenti qualora si rendessero necessari con l'evolversi dei protocolli di sicurezza.

6.2 Configurazione del server Shibboleth SP

Il diagramma architetturale presentato sopra presenta la struttura del server Shibboleth SP del quale sono evidenziati i componenti principali che sono oggetto di configurazione. In particolare, l'infrastruttura Shibboleth SP consiste di due componenti principali:

- un componente aggiuntivo per il Web Server (un modulo nel caso di Web Server Apache e un filtro ISAPI nel caso di Web Server IIS)
- un *demone* denominato *shibd* che si occupa della gestione delle sessioni di autenticazione

Il primo di tali componenti (*mod_*shib) è stato già descritto nella sez. 6.1, in merito alla configurazione di Apache HTTP Server. In questa sezione verrà dettagliata la configurazione del secondo componente, che costituisce un processo separato del sistema operativo nel quale opera, di nome *shibd*.

Un'installazione tipica del server Shibboleth SP crea alcune cartelle nel file system come descritto di seguito:

- *bin*: contiene alcune utility da linea di comando utilizzate dal software per firmare i messaggi SAML prodotti e verificare la firma dei messaggi ricevuti, per effettuare interrogazioni sui metadati, ecc.
- *doc*: contiene le licenze del software e le note di rilascio dello stesso
- *etc/shibboleth*: contiene i file di configurazione del software
- *lib*: contiene le librerie da cui dipende il software e, nella directory figlia *shibboleth*, i componenti per i web server
- *sbin*: contiene l'eseguibile del demone *shibd*
- *share/xml*: contiene gli schemi XML dei messaggi SAML e dei file di configurazione del software
- *var*: conterrà i file di log (nella directory figlia *log*) e i file utilizzati durante l'esecuzione del software (nella directory figlia *run*), ad esempio le copie locali dei metadati delle entità interagenti

Le cartelle indicate sopra sono raccolte in un'unica cartella contenitore nel caso di installazione fatta su sistema operativo Microsoft Windows. Nel caso di sistemi Unix/Linux, tali cartelle sono già presenti nell'alberatura standard della maggior parte delle distribuzioni.

La configurazione del software Shibboleth SP si compone di tre step, ai quali corrispondono altrettanti file di configurazione in formato XML presenti nella cartella "*/etc/shibboleth*":

- configurazione dei Virtual Host corrispondenti ai vari Service Provider integrati, delle modalità di autenticazione e dei metadati (file *shibboleth2.xml*);
- configurazione del mapping degli attributi del profilo degli utenti autenticati (file *attribute-map.xml*);
- configurazione delle policy di filtraggio degli attributi del profilo degli utenti autenticati (file *attribute-policy.xml*).

Nelle prossime sezioni verranno dettagliati questi tre step, descrivendo la struttura dei relativi file di configurazione.

6.2.1 File di configurazione shibboleth2.xml

Il file *shibboleth2.xml* rappresenta il nodo centrale della configurazione del Service Provider Shibboleth: con esso è possibile configurare la comunicazione tra il modulo del web server (*mod_shib*) ed il *demone shibd;* in esso, inoltre, sono definiti i singoli servizi applicativi che si desidera proteggere tramite autenticazione SAML, specificando per ciascuno gli endpoint SAML che si vuole esporre all'esterno (i quali rappresenteranno l'interfaccia verso le infrastrutture di autenticazione).

Si rammenta che, in linea generale, l'unica operazione da compiere lato SP è applicare le **personalizzazioni** descritte nella sezione 3.4, lasciando il restante contenuto del file inalterato rispetto a quanto installato dal prodotto in fase di setup. Il presente paragrafo ha carattere informativo qualora l'Ente intendesse sfruttare alcune personalizzazioni avanzate o sfruttare alcune potenzialità insite nel prodotto.

6.2.1.1 Configurazione generale

Il frammento del file inerente la configurazione della comunicazione tra il modulo *mod_shib* e il demone *shibd* è il seguente:

```
<OutOfProcess logger="shibd.logger">
        <!--  <Extensions> ...</Extensions>  -->
</OutOfProcess>
<InProcess logger="native.logger">
        <!--  <Extensions> ...</Extensions>  -->
</InProcess>
<TCPListener address="127.0.0.1"port="1600"acl="127.0.0.1"/>
```

Gli elementi <*OutOfProcess*>e <*InProcess*> configurano rispettivamente il comportamento del demone *shibd* e del modulo *mod_shib* o del filtro ISAPI (a seconda del web server che si sta utilizzando) attraverso l'uso di sotto-elementi <*Extensions*> e <*ISAPI*>, mentre gli attributi *logger* permettono di specificare i file di configurazione dell'infrastruttura di logging per questi componenti.

In caso di utilizzo del web server IIS è necessario configurare l'elemento <*ISAPI*> specificando uno o più sottoelementi <*Site*>, che si riferiscono alle istanze dei siti gestiti da IIS. Si rimanda alla sezione apposita e alla documentazione Shibboleth SP **Errore. L'origine riferimento non è stata trovata.**per maggiori dettagli relativi all'uso del web server M icrosoft IIS.

Ai fini delle attività di integrazione oggetto di questo documento, non è necessario specificare nessun elemento <*Extensions*>. L'elemento <*TCPListener*> permette di specificare l'indirizzo e la porta a cui è connesso il demone *shibd* (queste informazioni sono state specificate in fase di installazione e non necessitano di riconfigurazione).

6.2.1.2 Definizione dei Virtual Host

La definizione di un nuovo Service Provider viene compiuta andando ad operare in due sezioni distinte del file di configurazione *shibboleth.xml*.

Innanzitutto è necessario, tramite l'elemento <*RequestMapper*>, specificare il percorso dei singoli servizi che si desidera configurare, associando ad ognuno di essi un identificativo univoco. Il frammento interessato è mostrato di seguito:

```
<RequestMapper type="Native">
<RequestMap applicationId="default">
<Host name="SERVICEPROVIDER.DOMAIN.COM" scheme="https" port="443">
<Path
name="WEBAPP1/PROTECTED_PATH1"authType="shibboleth"requireSession="true"exportAssertion="true"/>
<Path name="WEBAPP2/PROTECTED_PATH2"applicationId="SERVICE2_ID"
authType="shibboleth"requireSession="true"exportAssertion="true"/>
</Host>
</RequestMap>
</RequestMapper>
```

Il frammento sopra riportato definisce un Virtual Host di tipo HTTPS per il Service Provider con hostname SERVIECPROVIDER.DOMAIN.COM, in ascolto sulla porta standard (443). È possibile variare tali tre parametri per adattare la configurazione a qualunque scenario particolare. Si noti che in questo caso non è necessario fare distinzione tra Virtual Host per connessioni HTTP e Virtual Host per connessioni HTTPS in quanto il front-end è costituito dal server web Apache che riceve le richieste ed è in grado di attivare il Virtual Host corretto, il quale rimanderà a Shibboleth SP individuando già la parte di configurazione interessata.

Come per la configurazione di Apache, è necessario indicare i percorsi protetti da autenticazione per i quali Shibboleth SP può comportarsi in modo diverso relativamente all'attivazione della fase di autenticazione. Nell'esempio, mediante gli elementi <*Path*> definiti all'interno dell'elemento <*Host*>, è possibile definire altrettante risorse (URL) da proteggere mediante autenticazione, presso quel Service Provider.

Dopo aver definito le risorse protette, i dettagli sulle modalità con le quali deve essere attivata la procedura di autenticazione per l'accesso a ciascuna di esse vengono fornite in sezioni successive del file di configurazione. In particolare, se per una direttiva <*Path*> non viene specificato l'attributo "*applicationId*", i dettagli aggiuntivi della configurazione saranno riportati in una sezione denominata <*ApplicationDefaults*>. È possibile definire, per ciascuna risorsa protetta dei parametri personalizzati mediante l'indicazione di un attributo *"applicationId"* sull'elemento <Path>, fornendo come valore il nome di una sezione del file di configurazione, denominata <*ApplicationOverride*>*,* come elemento annidato entro<*ApplicationDefaults*>. Laddove in quest'ultima non siano specificati alcuni dati, verranno utilizzati i valori definiti in <*ApplicationDefaults*>.

Si segnala che è possibile raffinare la risoluzione con cui vengono individuate le risorse protette, includendo uno o più elementi <Path> all'interno di altri elementi <Path> e specificando per essi valori diversi rispetto agli elementi contenitori.

6.2.1.3 Configurazioni delle interfacce SAML

Una volta associati degli identificatori ad ogni servizio, è possibile configurare il comportamento di Shibboleth a fronte delle richieste di risorse protette da autenticazione. Il server Shibboleth attiverà per ciascun nuovo accesso ad una risorsa nell'ambito di una sessione utente, un'interazione con l'infrastruttura di autenticazione o Identity Provider in accordo alla specifica SAML supportata da tale infrastruttura.

Per regolare tale comportamento, al file shibboleth2.xml occorre aggiungere una sezione <ApplicationOverride> per ciascuno dei Service Provider oggetto dell'integrazione, di solito in corrispondenza 1:1 con i Virtual Host come definiti in precedenza. Ogni sezione <ApplicationOverride> deve essere riportata all'interno della sezione <ApplicationDefaults>, dopo tutte le altre definizioni, prima della chiusura di tale sezione.

Ogni sezione <ApplicationOverride> ha la seguente struttura:

```
<ApplicationOverride id="applicationID"
entityID="SP_ENTITY_ID"
REMOTE_USER="saml_attribute_codiceFiscale"
homeURL="HTTPS://SERVICEPROVIDER.DOMAIN.COM/HOMEURL"
signing="true"encryption="false">

<Sessions lifetime="28800"timeout="3600"checkAddress="false"
handlerURL="/WEBAPP/PROTECTED_PATH/Shibboleth.sso"handlerSSL="false"
exportLocation="/GetAssertion"exportACL="127.0.0.1"
idpHistory="false"idpHistoryDays="7">

       <SSO entityID="https://idpcrl.crs.lombardia.it//scauth">
SAML1
</SSO>
       </Sessions>
       <MetadataProvider type="XML" file="[PATH_TO_IDPC_METADATA]/IdPCRL-metadata.xml"/>
       <AttributeExtractor type="XML" validate="true" path="attribute-map-ipdcrl.xml"/>
</ApplicationOverride>
```

In esso occorre operare le seguenti sostituzioni:

- sostituire la stringa "**SP_ENTITY_ID**" con l'identificatore che si vuole associare allo specifico Service Provider; tale identificatore sarà riportato anche nei relativi metadati SAML come entityID;
- sostituire l'URL "**HTTPS://SERVICEPROVIDER.DOMAIN.COM/HOMEURL**" con l'URL della risorsa presso il Service Provider a cui indirizzare l'utente al termine del processo di autenticazione nel caso in cui non sia specificato nessun altro URL valido;
- sostituire la stringa **"/WEBAPP/PROTECTED_PATH**" con il percorso, presso il Service Provider, della risorsa che si intende proteggere**.**

6.2.1.4 Configurazione dell'Identity Provider

In linea generale, è possibile configurare il SP Shibboleth in modo che veicoli le proprie "authentication request" verso un qualsiasi Identity Provider che supporti il protocollo SAML 2.0.

Nell'ambito specifico trattato dal presente documento, l'Identity Provider di riferimento è naturalmente quello di Regione Lombardia (IdPC), ed in particolare il servizio GEL - Gateway Enti Locali.

È necessario configurare Shibboleth in modo che:

1. consenta l'ingaggio dell'opportuno Identity Provider a fronte del tentativo di accesso ad una risorsa protetta;
2. sappia verificare le asserzioni di identità prodotte da IdPC. Tale operazione si compie censendo i **certificati** (e le relative **root**) cui corrispondono le chiavi crittografiche utilizzate da IdPC.

Entrambi i punti sono stati trattati nella sezione 3.4: in particolare, la trust chain di certificazione è contenuta nel file *IdpcGelMetadata_locale.xml*, che viene reso disponibile da LISPA nell'ambito di un "Kit di integrazione" al servizio GEL.

6.2.2 Mappatura degli attributi utente

Il server Shibboleth SP è in grado di ricevere i messaggi SAML prodotti dall'infrastruttura di autenticazione o Identity Provider utilizzato e di estrarre da essi gli attributi relativi agli utenti autenticati. Tali attributi vengono quindi inseriti in altrettanti header HTTP, per consentire al Service Provider di accedervi ed utilizzarli.

Per definire le corrispondenze tra gli attributi presenti nei messaggi SAML e gli header HTTP prodotti, è necessario intervenire sul file "*/etc/shibboleth/attribute-map.xml*", oggetto di questa sezione. In tale file sono definite una serie di regole di mappatura come la seguente:

```
<Attribute name="NOME_ATTRIBUTO_SAML"id="NOME_HEADER_HTTP"nameFormat="SAML_ATTRIBUTE_NAME_FORMAT">
        <AttributeDecoder xsi:type="StringAttributeDecoder"/>
</Attribute>
```

Anche questo file (*attribute-map.xml*) viene fornito nel "Kit di integrazione" distribuito da LISPA in modo che risulti agevole il mapping dei principali attributi provenienti dal sistema di autenticazione.

Gli attributi configurabili sono:

- **name**: indica il nome dell'attributo così come specificato nell'asserzione SAML ricevuta
- **id**: indica il nome che si vuole dare all'header HTTP che conterrà il valore di questo attributo
- **nameFormat**: indica il formato dell'attributo, così come specificato dallo standard SAML 2.0

L'elemento *<AttributeDecoder>* specifica come deve essere interpretato il valore dell'attributo e in generale non deve essere modificato.

A titolo d'esempio si riporta una mappatura dell'attributo relativo al codice fiscale:

```
<Attribute name="codiceFiscale" id="saml_attribute_codicefiscale"
nameFormat="https://idpcrl.crs.lombardia.it//scauth">
        <AttributeDecoder xsi:type="StringAttributeDecoder"/>
</Attribute>
```

6.2.3 Filtraggio degli attributi utente restituiti

A fronte delle definizioni degli attributi mappati come descritto nella sezione apposita, è possibile definire quali tra essi il server Shibboleth utilizzerà effettivamente per popolare gli header HTTP. Di seguito è mostrata una semplice regola, presente nel file *attribute-policy.xml*, che abilita la comunicazione al Service Provider di un attributo, specificando il nome dell'header HTTP corrispondente, indipendentemente dai valori che questo assume:

```
<afp:AttributeFilterPolicy id="releaseToAnyone">
        <afp:PolicyRequirementRule xsi:type="ANY"/>
        <afp:AttributeRule attributeID="NOME_HEADER_HTTP">
                <afp:PermitValueRule xsi:type="ANY"/>
        </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

Per maggiori informazioni sulle regole che possono essere definite, si rimanda alla documentazione in linea del SP Shibboleth.

6.2.4 Casi particolari

6.2.4.1 Configurazione di più applicazioni in un Virtual Host

È possibile proteggere più applicazioni con la stessa istanza di Shibboleth. Per compiere tale operazione, occorre agire sulla stanza <*RequestMapper*> defininendo le varie applicazioni da proteggere ed associando ad ognuna un <*applicationId*> che va poi popolato nella relativa stanza.

Nell'esempio che segue, vengono definiti due applicationId (di cui uno deve sempre essere valorizzato a *default*) per due distinti path. Si noti che il Virtual Host è il medesimo (nell'esempio vale *erogatore.crs.lombardia.it*):

```
<RequestMapper type="Native">
    <RequestMap applicationId="default">
      <Host name="erogatore.crs.lombardia.it" authType="shibboleth" requireSession="true" exportAssertion="true"
scheme="https" port="443">
        <Path name="/protected1" applicationId="erogatore1" authType="shibboleth" requireSession="true"
exportAssertion="true"/>
        <Path name="/erogatore-main/protected2" applicationId="erogatore2" authType="shibboleth"
requireSession="true" exportAssertion="true"/>
      </Host>
    </RequestMap>
</RequestMapper>
```

6.2.4.2 Configurazione di più applicazioni in più Virtual Host

È altresì possibile configurare più applicazioni in differenti Virtual Host. L'accortezza da attuare è replicare la stanza *Host*, come nell'esempio che segue:

```
<RequestMapper type="Native">
    <RequestMap applicationId="default">
<Host name="erogatore.crs.lombardia.it" authType="shibboleth" requireSession="true" exportAssertion="true"
scheme="https" port="443">
          <Path name="/erogatore-servizio/protected" authType="shibboleth" requireSession="true"
exportAssertion="true"/>
      </Host>
      <Host name="faddev.crs.lombardia.it" authType="shibboleth" requireSession="true" exportAssertion="true"
scheme="https" port="443">
          <Path name="/protected" authType="shibboleth" requireSession="true" exportAssertion="true"/>
      </Host>
    </RequestMap>
</RequestMapper>
```

6.2.4.3 Dispiegamento di Shibboleth dietro un bilanciatore o reverse proxy

Se l'istanza di Shibboleth viene dispiegata "dietro" un bilanciatore o reverse proxy che si occupa di chiudere il canale SSL con il browser, vi sono due possibilità di configurazione:

1. si istanzia il listener SSL in Shibboleth (come già descritto nel presente documento) in modo che siano rispettate le regole di "ingaggio" di IdPC che prevedono l'indirizzo della componente *AssertionConsumer* esposto in *https* e non in *http*, **oppure**
2. si effettua una modifica al file di configurazione del Virtual Host di Apache in modo da "istruirlo" del fatto che il dominio che protegge è esposto con protocollo https. Il protocollo SSL viene così "recepito" da IdPC durante l'ingaggio al sistema di autenticazione (parametro "target" della URL di redirezione). Di seguito un esempio di configurazione:

```
Listen dcss.crs.lombardia.it:80

<VirtualHost dcss.crs.lombardia.it:80>

ServerName https://dcss.crs.lombardia.it
ServerAdmin root@fe-dcsanita.cgi.crs.lombardia.it
ErrorLog logs/error_log
TransferLog logs/access_log
LogLevel warn
…
```

6.2.5 Logout da una applicazione

A fronte della volontà di un utente di effettuare il logout da una certa applicazione protetta da una istanza di Shibboleth, l'applicazione stessa dovrà implementare la chiamata alla seguente URL:

http://*applicazione*/protected/Shibboleth.sso/Logout

Tale URL è altresì ricavabile ispezionando il parametro HTTP_SHIB_ASSERTION_01 restituito nell'http-header, e sostituendo le sezioni successive a *Shibboleth.sso* con la sola keyword *Logout*.

L'effetto di tale redirezione del browser sarà la cancellazione del cookie utente entro Shibboleth. Come conseguenza, un successivo tentativo di accesso ad una risorsa protetta entro *applicazione*provocherà un nuovo ingaggio di IdPC per la negoziazione di una nuova autenticazione.

Le applicazioni integrate a Shibboleth sono tenute ad effettuare quanto descritto, in caso contrario l'unica modalità con cui gli utenti potranno effettuare il logout sarà chiudere il browser.

Si noti che ogni applicazione protetta entro Shibboleth ha un cookie di sessione "dedicato", quindi la rimozione di un cookie non interferisce con le altre eventuali sessioni utente attive.

Si segnala che la pagina di "local logout" messa a disposizione da Shibboleth è personalizzabile graficamente, in modo da poterla adeguare al *look-and-feel* della web application che la invoca. Tale possibilità è effettivamente sfruttabile nel solo caso in cui una istanza di Shibboleth protegga una sola applicazione, in quanto la pagina di "local logout" è unica per tutte le applicazioni protette.

Si noti infine che la URL sopra indicata può essere completata con il parametro return per indicare a quale indirizzo portare il browser utente dopo l'avvenuto logout, ad esempio:

http://*applicazione*/protected/Shibboleth.sso/Logout?return=http://www.regione.lombardia.it

7. Integrazione di Service Provider con il Reverse Proxy

Utilizzando il Reverse Proxy descritto nelle sezioni precedenti, è possibile proteggere l'accesso ad un Service Provider tramite autenticazione ottenuta da un Identity Provider come IdPC. Come detto, il server Shibboleth SP mette a disposizione del Service Provider l'insieme degli attributi relativi all'utente autenticato, sotto forma di header HTTP. Per consentire una più agevole integrazione di alcune tipologie di Service Provider sviluppati in tecnologia JEE con il Reverse Proxy, sono stati realizzati alcuni componenti in grado di leggere tali header HTTP e di passare le informazioni in essi contenute al Service Provider nel formato atteso. Tali componenti possono essere installati presso il Service Provider il quale necessiterà pertanto di alcune semplici attività di riconfigurazione per abilitarne il funzionamento. In particolare, in questo documento verranno descritte le configurazioni necessarie per riconfigurare Service Provider precedentemente integrati con l'interfaccia di accesso ai servizi per IdPC, e per servizi non ancora integrati con nessuna infrastruttura di autenticazione.

7.1 Migrazione di Service Provider J2EE integrati con IdPC mediante "Reference Implementation"

Le operazioni da compiere per integrare un servizio precedentemente integrato tramite l'interfaccia di accesso ai servizi con IdPC consistono in una fase preliminare di disabilitazione dei precedenti componenti di autenticazione, seguita da una successiva installazione di nuovi componenti.

7.1.1 Disattivazione dei precedenti componenti di integrazione

Per disattivare i precedenti componenti di integrazione inclusi nell'interfaccia di accesso ai servizi per IdPC è necessario commentare (o rimuovere) le relative sezioni presenti nel deployment descriptor (file "web.xml") del servizio.In particolare è necessario disattivare:

- il filtro di autenticazione

```
<!--
<filter>
        <filter-name>Authentication Filter</filter-name>
        <filter-class>
                it.lisit.idpc.ri.filters.AuthenticationFilter
        </filter-class>
        ...
</filter>
-->
```

- la regola di mapping del filtro

```
<!--
<filter-mapping>
        <filter-name>Authentication Filter</filter-name>
        <url-pattern>/protected/*</url-pattern>
</filter-mapping>
-->
```

- la servlet che implementa il servizio Assertion Consumer Service

```
<!--
<servlet>
        <servlet-name>AssertionConsumerService</servlet-name>
        <display-name>Sirac Assertion Consumer Service</display-name>
        <description>Sirac Assertion Consumer Service</description>
        <servlet-class>
                it.lisit.idpc.ri.web.AssertionConsumerServlet
        </servlet-class>
        ...
</servlet>
-->
```

- la regola di mapping per la servlet

```
<!--
<servlet-mapping>
        <servlet-name>AssertionConsumerService</servlet-name>
        <url-pattern>/AssertionConsumerService</url-pattern>
</servlet-mapping>
-->
```

7.1.2 Configurazione dei nuovi componenti di integrazione

Una volta disattivati i precedenti componenti di integrazione, è possibile installare e configurare il componente di accesso agli header HTTP.

Questo "**componente aggiuntivo**" è **reso disponibile**nel "Kit di integrazione" distribuito da LISPA.

Tale componente aggiuntivo è così strutturato:

- *idpc-sp-integration-shibboleth.jar*, questa libreria contiene un servlet-filter per l'elaborazione del profilo utente post-autenticazione, e va aggiunto alla directory *WEB-INF/lib* dell'applicazione web del servizio. Il suo compito è accedere all'http header e rendere fruibile all'applicazione i dati di asserzione prodotti da IdPC ;
- *web/WEB-INF/classes/resources/shibboleth-sp-config.xml*, questo file contiene alcuni parametri di configurazione del filtro;
- *web/WEB-INF/classes/resources/attribute-mapper.xml,* questo file definisce il mapping tra i nomi degli header HTTP contenenti i valori degli attributi restituiti da Shibboleth SP e il nome degli attributi così come richiesti dal servizio che si sta integrando;
- *web/WEB-INF/lib*, in questa directory sono presenti alcune librerie di supporto, da aggiungere all'omonima directory del servizio.

Le modifiche da compiere al *deployment descriptor* del servizio sono le seguenti:

- aggiunta della dichiarazione del filtro

```
<filter>
        <filter-name>ShibbolethHeaderReaderFilter</filter-name>
        <display-name>ShibbolethHeaderReaderFilter IdPC</display-name>
        <description></description>
        <filter-class>
                it.lisit.idpc.ri.filters.ShibbolethHeaderReaderFilterIdpc
        </filter-class>
        <init-param>
                <param-name>configurationFile</param-name>
                <param-value>resources/shibboleth-sp-config.xml</param-value>
        </init-param>
</filter>
```

- aggiunta della regola di mapping del filtro

```
<filter-mapping>
        <filter-name>ShibbolethHeaderReaderFilter</filter-name>
        <url-pattern>/protected/*</url-pattern>
</filter-mapping>
```

Il file di configurazionedel filtro (*shibboleth-sp-config.xml*) ha la seguente struttura:

```
<ShibExtensionConfiguration>
        <!-- valori possibili:
                1 = accesso agli header HTTP,
                2 = accesso all'asserzione in response -->
        <FilterMode>1</FilterMode>
        <InitParameters>
                <attributesPrefix>saml_attribute_</attributesPrefix>
                <authenticationMethodHeaderName>
                        Shib-Authentication-Method
                </authenticationMethodHeaderName>
                <identifyingAttributeName>codiceFiscale</identifyingAttributeName>
                <assertionURLHeaderName>Shib-Assertion-01</assertionURLHeaderName>
                <attributeMappingConfigurationFilename>
                        resources/attribute-mapper.xml
                </attributeMappingConfigurationFilename>
        </InitParameters>
</ShibExtensionConfiguration>
```

Il parametro di configurazione “FilterMode” permette di specificare due modalità di funzionamento per il servlet-filter:

- modalità “1”: in questa modalità il filtro si limita a leggere gli header HTTP popolati da Shibboleth SP, effettuando eventualmente un mapping tra i nomi di questi header e i nomi degli attributi attesi dal servizio, e a inserire in sessione alcune variabili, descritte nel seguito, contenenti queste informazioni di base;
- modalità “2”: in questa modalità il filtro oltre ad effettuare tutte le operazioni compiute nella modalità “1”, accede all’asserzione di autenticazione SAML ricevuta dal server Shibboleth del Reverse Proxy, andando così ad arricchire le variabili di sessione.

Di seguito è mostrata una tabella riepilogativa con i parametri di configurazione presenti in questo file.

Parametro	**Descrizione**	**Valore**
FilterMode	Indica la modalità operativa del filtro	int (1 o 2)
attributesPrefix	Indica il prefisso utilizzato da Shibboleth SP nella creazione degli header HTTP contenenti il valore degli attributi e definito nel file di configurazione *attribute-map.xml* (sez. 0)	String
authenticationMethodHeaderName	Indica il nome dell’header HTTP speciale in cui è specificato il metodo di autenticazione utilizzato nella fase di autenticazione	String
identifyingAttributeName	Indica il nome dell’attributo contenente l’identificativo dell’utente autenticato	String
assertionURLHeaderName	Indica il nome dell’header HTTP tramite il quale è possibile accedere all’URL del servizio presso il Reverse Proxy che fornisce l’asserzione di autenticazione per l’utente autenticato	String
attributeMappingConfigurationFilename	Specifica il percorso del file che definisce il mapping tra i nomi degli header HTTP e gli attributi richiesti dal servizio	String

Al termine dell’elaborazione, il filtro imposta in sessione alcune variabili contenenti le informazioni estratte dal profilo utente, in particolare:

- `it.idpc.ri.erogatore.authenticated_user`: contiene l’identificativo del soggetto che ha richiesto l’autenticazione
- `it.idpc.ri.erogatore.authenticated_user_data`: contiene un oggetto Bean con gli attributi del profilo utente ricevuto
- `it.lisit.idpc.ri.erogatore.auth.completed`: contiene un valore booleano (`Boolean.TRUE` o `Boolean.FALSE`) ed indica l’esito dell’autenticazione
- `it.lisit.idpc.ri.erogatore.authentication.current.auth.strong.auth.saml.responsebase64`: contiene l’asserzione SAML di autenticazione ricevuta, codificata in Base 64 (solo in modalità “2”)

Queste variabili sono le stesse impostate dall’interfaccia di accesso ai servizi per IdPC; in questo modo la sostituzione dei componenti di integrazione risulta trasparente rispetto alla procedura con cui il servizio applicativo accede alle informazioni ricevute a valle del processo di autenticazione.

Il file *attribute-mapper.xml* fornito con il componente aggiuntivo viene utilizzato per effettuare un mapping tra il nome degli header HTTP restituiti da Shibboleth SP e il nome degli attributi utente utilizzati per popolare il JavaBean inserito in sessione. Tale mapping permette di configurare i nomi degli attributi sulle esigenze dei singoli servizi, senza dover intervenire sulla configurazione di Shibboleth SP.

Nel seguito è fornito un esempio di tale file di mapping:

```
<configuration>
	<AttributeMappings>
		<AttributeMapping
id="codiceFiscaleMapping"attributeName="saml_attribute_codicefiscale"mappedAttributeName="codiceFisc
ale"/>
		<AttributeMapping
id="cognomeMapping"attributeName="saml_attribute_cognome"mappedAttributeName="cognome"/>
		<AttributeMapping
id="nomeMapping"attributeName="saml_attribute_nome"mappedAttributeName="nome"/>
		<AttributeMapping
id="sessoMapping"attributeName="saml_attribute_sesso"mappedAttributeName="sesso"/>
		<AttributeMapping
id="datanascitaMapping"attributeName="saml_attribute_datanascita"mappedAttributeName="dataNascita"/>
	</AttributeMappings>
</configuration>
```

Ogni mapping è definito da un elemento *<AttributeMapping>* in cui l'attributo *id* rappresenta un identificatore univoco del mapping, l'attributo *attributeName* indica il nome dell'header HTTP che si desidera mappare, e l'attributo *mappedAttributeName* indica il nuovo nome che si vuole assegnare all'attributo.

7.2 Configurazione di Service Provider J2EE non ancora integrati con un IdP

In questa sezione vengono descritte le operazioni da compiere per integrare tramite Reverse Proxy Shibboleth un Service Provider non ancora integrato in alcuna infrastruttura di autenticazione.

L'integrazione è neutra rispetto alla tecnologia in cui è realizzato il servizio e **si basa sulla valorizzazione degli HTTP header.**

7.2.1 Configurazione dei componenti di integrazione

Al termine del processo di autenticazione, l'Identity Provider restituisce al reverse proxy Shibboleth, tra gli altri, i seguenti dati:

nome
cognome
codice Fiscale
sesso
dataNascita
userID
origineDatiUtente
statoNascita
luogoNascita
provinciaNascita
cellulare
emailAddress
idComuneRegistrazione
CNS_CARTA_REALE
CNS_SUBJECT
CNS_ISSUER

A sua volta il reverse proxy, setta **nell'header HTTPi dati** relativi all'utente autenticato, coerentemente con il profilo necessario al servizio (in particolare nel caso di autenticazione tramite SPID) e con quanto definito nel file *attribute-map.xml*disponibile nel "Kit di integrazione", ad esempio:codice fiscale, nome, cognome.

Sarà compito dell'integratore implementare il codice per recuperare i dati dall'Header e passarli al Service Provider.

Se l'implementazione verrà eseguita in linguaggio Java, è possibile utilizzare la librerie *idpc-sp-integration-shibboleth.jar*e le altre componenti citate nel relativo paragrafo, facendo ricorso in particolare alla classe *ShibbolethHeaderReaderCommonFilter* e al metodo *getAttributesMapFromHTTPHeaders*, che realizzano l'estrazione degli attributi presenti nell'HTTP header.

Un **esempio** di header http restituito dal reverse proxy a seguito di una avvenuta autenticazione è il seguente (in **grassetto** i dati ricavati dall'asserzione prodotta da IdPC):

HTTP_CACHE_CONTROL:no-cache

HTTP_CONNECTION:Keep-Alive

HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-ms-application, application/vnd.ms-xpsdocument, application/xaml+xml, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */* HTTP_ACCEPT_ENCODING:gzip, deflate

HTTP_ACCEPT_LANGUAGE:it HTTP_COOKIE:ASPSESSIONIDAQCQSTQT=BNILFLNBAPPOJIEAAAAEHLMA; _shibsession_64656661756c74687474703a2f2f666164646576=_7defa4535fe8199f9f63bf279a86a445
HTTP_HOST:faddev HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; GTB6.6; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; .NET CLR 3.0.30618; InfoPath.2; FDM; .NET4.0C)

HTTP_UA_CPU:x86

HTTP_SHIB_SESSION_ID:_7defa4535fe8199f9f63bf279a86a445

HTTP_SHIB_SESSION_INDEX:

HTTP_SHIB_IDENTITY_PROVIDER:https://idpcrl.crs.lombardia.it//scauth
HTTP_SHIB_AUTHENTICATION_METHOD:urn:oasis:names:tc:SAML:1.0:am:HardwareToken
HTTP_SHIB_AUTHENTICATION_INSTANT:2011-02-10T10:39:57.098Z
HTTP_SHIB_AUTHNCONTEXT_CLASS:urn:oasis:names:tc:SAML:1.0:am:HardwareToken
HTTP_SHIB_AUTHNCONTEXT_DECL:

HTTP_SHIB_ASSERTION_COUNT:01

HTTP_EPPN:

HTTP_AFFILIATION:

HTTP_UNSCOPED_AFFILIATION:

HTTP_ENTITLEMENT:

HTTP_ASSURANCE:

HTTP_TARGETED_ID:

HTTP_PERSISTENT_ID:

HTTP_SAML_ATTRIBUTE_CODICEFISCALE:CTTTML78A01F205F

HTTP_SAML_ATTRIBUTE_FIRSTNAME:

HTTP_SAML_ATTRIBUTE_LASTNAME:

HTTP_SAML_ATTRIBUTE_NOME:TREMILACENTONOVANTAQUATTRO

HTTP_SAML_ATTRIBUTE_COGNOME:CITTASISS

HTTP_SAML_ATTRIBUTE_DATANASCITA:01/01/1978

HTTP_SAML_ATTRIBUTE_SESSO:M

HTTP_SHIB_APPLICATION_ID:default

HTTP_REMOTE_USER:
HTTP_SHIB_ASSERTION_01:http://faddev/protected/Shibboleth.sso/GetAssertion?key=_7defa4535fe8199f9f63bf279a86a445&ID=_eafbe0736d8cecd9d424ec42785457ba

HTTP_X_FORWARDED_FOR:10.221.13.40

HTTP_X_FORWARDED_HOST:faddev

HTTP_X_FORWARDED_SERVER:faddev

Di particolare interesse l'attributo **HTTP_SHIB_AUTHENTICATION_METHOD** con cui il Service Provider può determinare il **meccanismo di autenticazione utilizzato dall'utente**, in modo da autorizzare l'effettiva fruizione del servizio solo se l'utente ha utilizzato credenziali coerenti con le proprie prerogative di sicurezza.