Sistema Informativo Socio Sanitario

- SIAU -
Specifiche di Interfaccia Applicativi Utente

IdPC: Reference Implementation dell'Interfaccia di Accesso alle Applicazioni

Codice documento:	**CRS-ISAU-SIAU#77**		
Versione:	**06.3**	**PVCS:**	*13*
Data di emissione:	**27-09-2012**	**Stato:**	**EMESSO**

		Funzione	Nome	Firma
LI	**Redazione**	Architettura Generale	A.Zanini	
	Preverifica	Area Architettura Generale	P.Valenti	
	Preapprovazione	Dipartimento SISS	N.Contardi	

LI	**Verifica**	Governo Sistemi Informativi e Piattaforma CRS	J.Mason	
	Approvazione	Governo Sistemi Informativi e Piattaforma CRS	J.Mason	

LI	**Emissione**	Strategia,Program Management e Business Consulting	F.Sirtori

CRONOLOGIA DELLE VERSIONI	
Num. versione	**Sintesi delle variazioni**
1	Prima versione
2	Descrizione Reference Implementation .NET
3	Introduzione dei riferimenti al runtime della Reference Implementation; precisazioni minori
4	Precisazioni minori
5	Aggiunto parametro “profile” per la scelta di uno quattro profili utente; aggiunto parametro NOME_MACCHINA e NOME_MACCHINA_ASSERTION_CONSUMER; compatibilità sotto ambiente BEA WEBLOGIC
6	Precisazioni minori
7	Introdotte precisazioni sulla modalità di trasferimento dei dati tra le componenti della "reference implementation"
8	Revisionata la modalità di configurazione per l'architettura J2EE e la presenza di due nuovi certificati "root". Rimozione della Reference Implementation J2EE a favore dell'architettura che utilizza l'agente di sicurezza Shibbolteh (cfr CRS-ISAU-SIAU#97).
9	Corretta la modalità di configurazione per l'architettura J2EE, cap. 4.1.2.1 e 4.1.2.2 .
10	Migliorata la descrizione della modalità di configurazione per l'architettura J2EE, cap. 4.1.2.1 e 4.1.2.2 .
11	Adeguamenti minori.
12	Introdotta sezione 4.1.2.10
13	Introdotta precisazione su refuso nomenclatura trust-store (sez. 4.1.1.2)

SOSTITUISCE/MODIFICA
Tutte le versioni precedenti

LIMITI DI UTILIZZO DEL DOCUMENTO
CRS-SISS Stadio 2 La componente è indipendente dalle altre componenti del SISS e quindi non appartiene propriamente a nessuna delle release del SISS.

INDICE DEL DOCUMENTO

1 INTRODUZIONE

1.1 SCOPO E CAMPO DI APPLICAZIONE

Questo documento descrive la procedura di installazione e configurazione della "Reference Implementation" di una web application J2EE in cui siano integrati i servizi di autenticazione forniti da IdPC di Regione Lombardia.

1.2 RIFERIMENTI

- CRS-ISAU-SIAU#76 "Identity Provider Cittadini Regione Lombardia"
- CRS-ISAU-SIAU#78-Reference_Implementation_Java_IdPCRL.zip
- CRS-ISAU-SIAU#97- Integrazione_IdPC_Shibboleth

1.3 ACRONIMI E DEFINIZIONI

CA	Certification Authority
CIE	Carta d'Identità Elettronica
CNS	Carta Nazionale dei Servizi
IdPC-RL	Identity Provider Cittadini – Regione Lombardia
J2EE	Java 2 Enterprise Edition
JSP	Java Server Pages
WAYF	Where Are You From

2 GENERALITA'

IdPC propone un modello architetturale per l'integrazione di servizi centralizzati di autenticazione: è compito dell'erogatore del servizio adeguare la propria struttura applicativa – nel modo meno invasivo possibile – in modo tale da poter essere *compliant* con le specifiche di interfaccia previste.

La Reference Implementation si propone come una implementazione minimale ma completa di tutte le interfacce previste e che rispetti i vincoli funzionali previsti nel documento CRS-ISAU-SIAU#76.

La versione corrente di questo documento descrive la *Reference Implementation* disponibile per ambienti J2EE.

I sorgenti ed il runtime di queste *Reference Implementation* sono rese disponibili nell'archivio:

CRS-ISAU-SIAU#78-Reference_Implementation_Java_IdPCRL.zip

il cui download è liberamente effettuabile dal sito www.crs.lombardia.it .

3 SPECIFICHE DI PROGETTAZIONE ARCHITETTURALE (DAS)

3.1 SOLUZIONE J2EE

3.1.1 Requisiti di progettazione architetturale

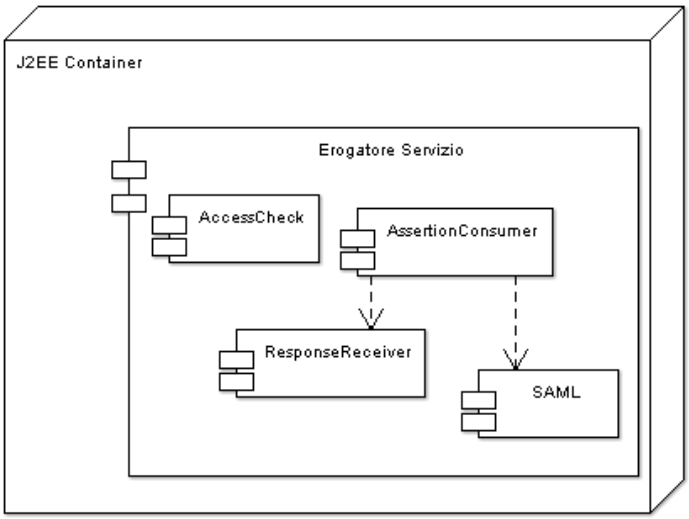
La Reference Implementation è una web application in grado di rispettare il workflow e interfacce previste dal sistema di autenticazione IdPC.
Le componenti architetturali sono state implementate mediante un uso di oggetti Filter, Servlet e pagine jsp.

L'architettura IdPC prevede per l'erogatore la presenza dei seguenti componenti:

- *Access Check*
- *Assertion Consumer*
- *Response Receiver*

La *Reference Implementation* propone un esempio di :

- implementazione dei componenti richiesti secondo specifiche e best practicies del documento SIAU-ISAU#76;
- integrazione con una web application esistente dei suddetti moduli.



3.1.2 Configurazione sistemistica

E' necessario che il J2EE container sia compliant con le specifiche Servlet 2.3 e con JDK 1.4.x o superiori.

E' altresì necessario esporre il servizio di *Assertion Consumer* tramite SSL/TLS: le informazioni che sono contenute nell'asserzione in arrivo da IdPC vengono passate non cifrate; contenendo (potenzialmente) dati sensibili di un profilo utente, è buona norma forzare la cifratura del canale tra il browser ed il modulo.

A tal proposito si consiglia l'adozione di un certificato Omniroot, ovvero emesso da una CA automaticamente ritenuta attendibile dai browser (i.e. Internet Explorer).

Si veda la documentazione del proprio web container per la creazione e configurazione di un certificato SSL.

3.1.3 Configurazione WEB CONTAINER

Il web container necessita di configurazione al fine di poter utilizzare le librerie **OpenSAML** 1.1; in particolare, nel caso in cui la JVM sia una 1.4.x è necessario configurare il run time in modo tale che la JVM utilizzi una versione più recente delle API XML. A tal fine, è sufficiente passare la seguente opzione in fase di inizializzazione della JVM del container:

```
-Djava.endorsed.dirs=<PATH_TO_JAVA_ENDORSED_DIRS>
```

La distribuzione della *Reference Implementation* comprende una directory `endorsed` contenente le librerie (API XML e parser XML); copiare tale directory in un path visibile e accessibile dall'istanza e configurare l'opzione come sopra.

3.2 SOLUZIONE PER AMBIENTI DIVERSI DA J2EE

Per chi utilizzasse ambienti diversi da J2EE (ad es. .Net o PHP) è disponibile una specifica soluzione basata su Reverse Proxy Apache con agente di sicurezza Shibboleth che consente di operare senza installare componenti software nel server applicativo.
Per tale soluzione riferirsi al documento CRS-ISAU-SIAU#97.

La soluzione basata su Reverse Proxy Apache con agente di sicurezza Shibboleth può essere adottata anche per chi opera in ambiente J2EE al fine di mantenere una maggiore modularità della soluzione realizzata.

4 SPECIFICHE DI PROGETTAZIONE FUNZIONALE (DDF)

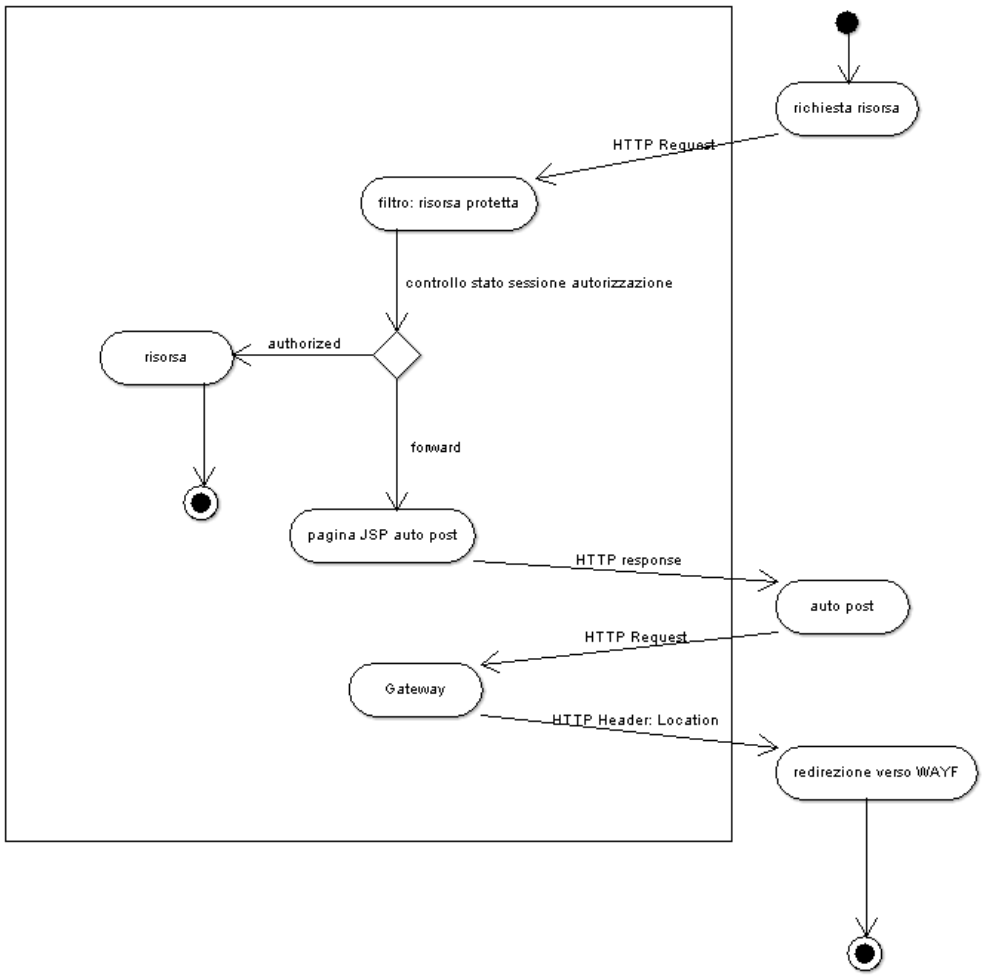
4.1 SOLUZIONE J2EE

4.1.1 Requisiti di progettazione funzionale

4.1.1.1 Access Check

L'Access Check è realizzato mediante un filtro (`it.lisit.idpc.ri.filters.AuthenticationFilter`) che ha il compito di controllare lo stato della sessione di autenticazione del browser: ciò è realizzato mediante il controllo della valorizzazione di un attributo di sessione "`it.lisit.idpc.ri.erogatore.auth.completed`". Nel caso in cui la navigazione dell'utente abbia già impostato tale valore, la richiesta viene passata all'elemento successivo della `FilterChain` configurata per il path della risorsa richiesta. Altrimenti, mediante un forward, l'elaborazione passa ad una pagina JSP (`/WEB-INF/PostAuthRequest.jsp`) che prepara una form autopostante sul browser con la valorizzazione di opportuni campi.

Il filtro it.lisit.idpc.ri.filters.AuthenticationFilter si occupa anche di recuperare i parametri NOME_MACCHINA e NOME_MACCHINA_ASSERTION_CONSUMER impostati nel file IDPC-RI.properties.
Il parametro NOME_MACCHINA corrisponde al nome logico o indirizzo ip su cui è deployata la webapplication, mentre il parametro NOME_MACCHINA_ASSERTION_CONSUMER corrisponse al nome logico o indirizzo ip su cui si trova il servizio AssertionConsumer.



La form invia i dati alla servlet di uscita (`it.lisit.idpc.ri.web.AuthGatewayServlet`) che predispone una redirect verso il ***WAYF*** (ovvero la url di IDPC) di Regione Lombardia.

La servlet it.lisit.idpc.ri.web.AuthGatewayServlet si occupa anche del recupero dell'eventuale parametro "profile" e l' invio verso l' IdPC (vedi sezione 4.1.2.5).

Il filtro protegge tutte e sole le risorse per le quali è stato configurato nel file `WEB-INF/web.xml` della web application; per questo motivo alcune risorse, quali la servlet di gateway e le interfacce dell'*AssertionConsumer* e del *ResponseReceiver* devono non essere protette per non generare dei loop di autenticazione tra le risorse in questione.

Per motivi di generalità e semplicità della presente implementazione, non è stata implementata alcuna metodologia per:
- controllo dello stato di autorizzazione all'accesso di una particolare risorsa ;
- meccanismo fine grained per il controllo della necessaria autenticazione alle risorse della web application.

La soluzione del filtro dà comunque la possibilità di proteggere l'accesso ad alcune risorse di una web application pre-esistente nel modo meno invasivo possibile.

4.1.1.2 Assertion Consumer

L'*AssertionConsumer* è il componente responsabile del controllo dell'asserzione restituita dall'IdPC. La sua interfaccia web è realizzata mediante una servlet (`it.lisit.idpc.ri.web.AssertionConsumerServlet`) che effettua una serie di controlli sull'asserzione SAML veicolata in forma Base64 tramite campo di una richiesta `POST` HTTP.

Il campo `SAMLResponse` viene trasformato in una SAML Response di cui viene controllata:
- l'integrità tramite verifica della firma elettronica apposta dall'IdPC ;
- l'identità del firmatario mediante l'ispezione del certificato allegato alla firma ;
- l'attendibilità della CA che ha rilasciato il certificato del firmatario ;
- l'esito della richiesta di autenticazione ;
- la valorizzazione di attributi.

La verifica dell'identità del firmatario non è puntuale: è sufficiente che il certificato utilizzato per la firma sia stato emesso da una CA trustata dall'applicazione. A tal fine, nella directory `/keystore` della web application è stato preparato un trust store (`trustStoreCompleto.jks` [1]) contenente i root certificate per accettare accessi con carte reali e virtuali (di test) ed un secondo trust store (`trustStoreReale.jks` [2]) contenente i root certificate per accettare accessi con sole carte reali:

Root certificates in `trustStoreCompleto.jks`:

1.
CN = LISIT CA di Servizio
OU = Servizio di certificazione
O = LISIT S.p.A.
C = IT
Identificazione personale: 83 cd 6c dc 6a b7 50 dc 8b 93 92 25 80 1f d0 86 9f 0e 15 70

2.
CN = LISIT CA Servizio di Integrazione
OU = Servizio di certificazione
O = LISIT S.p.A.
C = IT
Identificazione personale: 74 14 3d 0e e8 e0 fc 82 f4 e5 3b 87 b8 dc 29 40 7b c0 1c 2b

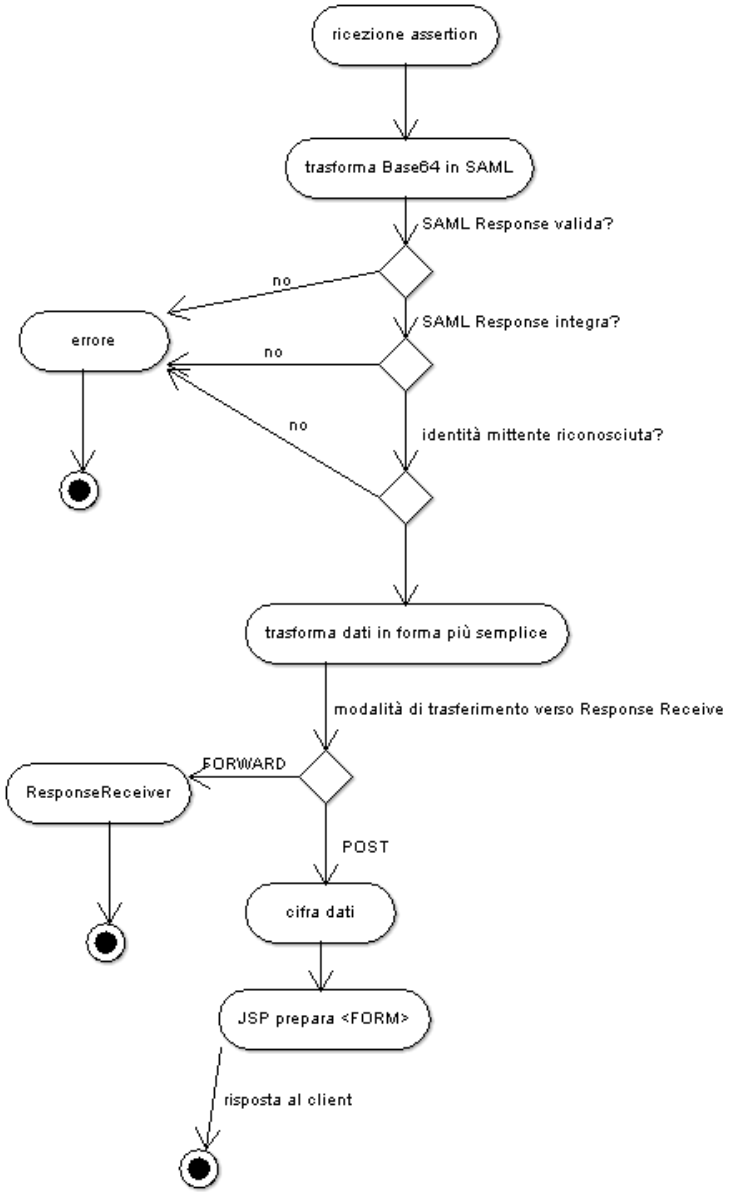
3.
CN = CA di Servizio
OU = Servizio di certificazione
O = Lombardia Informatica S.p.A.
C = IT
Identificazione personale: B2 EB 86 57 9B F8 36 7E E2 8E 5A 72 90 F9 9E 30 16 DB 6E E3

[1] A causa di un refuso durante il confezionamento delle "reference implementation", alcune versioni incorporano un trust store denominato `TrustoreCompleto.jks`. In tal caso, è necessario rinominare il file in `trustStoreCompleto.jks`
[2] A causa di un refuso durante il confezionamento delle "reference implementation", alcune versioni incorporano un trust store denominato `TrustoreReale.jks`. In tal caso, è necessario rinominare il file in `trustStoreReale.jks`

4.
CN = CA di Servizio di Integrazione
OU = Servizio di certificazione
O = Lombardia Informatica S.p.A.
C = IT
Identificazione personale: E2 DB B7 94 46 66 AE 4C 0C A1 C4 93 FC 57 21 61 74 E3 6C 9A

Tale configurazione di default consente di accettare sia asserzioni rilasciate da IdPC-RL e relative ad una CRS/CNS "reale" (CA **1.** e **3.**), sia relativa ad una CRS/CNS "demo" o "di test" (CA **2.** e **4.**).

Il diagramma illustra la sequenza di azioni che l'*Assertion Consumer* è tenuto a compiere.



In caso di fallimento di almeno uno di questi controlli, viene sollevata una eccezione riportante un messaggio esplicativo: sarà compito della pagina di errore preposta alla gestione delle eccezioni di run time visualizzare tale errore.

I dati di profilo contenuti nella SAML Response vengono portati in una forma più semplice per i successivi usi da parte del *Response Receiver* e di tutto l'applicativo.

L'elaborazione passa successivamente al *Response Receiver* tramite **`FORWARD`**.

Nei **soli** casi in cui tale modalità risulti tecnicamente non percorribile (i.e. componenti *AssertionConsumer* e *ResponseReceiver* collocati su server differenti e con tecnologie differenti), è possibile utilizzare la modalità `POST`. In questo caso i dati di asserzione non sono coinvolti in un transito "server-to-server", bensì "passano" dal browser utente.

A prescindere dalla modalità utilizzata, i dati vengono cifrati tramite una chiave di sicurezza conosciuta dalle due componenti (*AssertionConsumer* e *ResponseReceiver*) e nel seguito dettagliata.

4.1.1.3 Response Receiver

Il *ResponseReceiver* è responsabile della integrazione dei dati di profilo ricevuti da IdPC e successive trasformazioni da parte dell'*Assertion Consumer* nel contesto della web application.

Nel caso della *Reference Implementation* vengono valorizzati alcuni attributi della sessione; in particolare:

- `it.lisit.idpc.ri.erogatore.authentication.current.auth.strong.auth.saml.responsebase64` : contiene il Base64 della SAML Response precedentemente ricevuta dall'*Assertion Consumer* ;
- `it.lisit.idpc.ri.erogatore.auth.completed`: valorizzato con un oggetto `Boolean.TRUE`; questo è l'attributo controllato dall'*Access Check* per la verifica dello stato della sessione di autenticazione ;
- `it.idpc.ri.erogatore.authenticated_user_data`: contiene un oggetto Bean con gli attributi di profilo ricevuti.

L'esecuzione prosegue con un redirect del client verso l'indirizzo originario della risorsa precedentemente acceduta.

4.1.2 Configurazione applicativa

Le configurazioni applicative sono contenute nel file `WEB-INF/web.xml` della web application e vengono qui differenziate per domini di interesse.

4.1.2.1 Generazione della chiave simmetrica per la cifratura delle asserzioni tra *AssertionConsumer* e *ResponseReceiver*

Nella Reference Implementation non è rilasciato il file contenente la chiave simmetrica al fine di assicurarsi che ogni integrazione ne generi una propria.

La cartella **lib** della reference impemetation contiene i file di script *GeneratoreChiavi.bat* e *GeneratoreChiavi.sh*, rispettivamente per l'esecuzione in ambiente Window e Linux. Per una corretta esecuzione degli script è necessario sostituire i file **local_policy.jar** e **US_export_policy.jar** consegnate nella *Reference Implementation* nella cartella *jre\lib\security* della JRE installata sulla macchina dove si eseguono gli script. Qualora sulla macchina ospite fossero presenti più JRE di differenti release, si consiglia di indicare esplicitamente (tramite path assoluto) il binario *java.exe* da lanciare, in modo che di essere certi di referenziare la JRE su cui si sono copiati i jar sopra menzionati.

Lo script esegue la generazione di una chiave simmetrica inizializzata con un valore pseudo-casuale che viene memorizzata nel file *responsekeyfile.key* prodotto nella directory di lavoro dello script.

Alla chiave viene successivamente associata una *PassPhrase* utilizzata per alterare la sequenza cifrata, il valore di questo parametro va impostato in sede di configurazione dell'applicazione con l'unico vincolo che sia lo stesso utilizzato da chi cifra e da chi decifra, cioè *AssertionConsumer* e *ResponseReceiver*.

4.1.2.2 Configurazione della chiave simmetrica per la cifratura delle asserzioni tra *AssertionConsumer* e *ResponseReceiver*

La chiave simmetrica deve essere configurata nel file web.xml della web application. E' preferibile una collocazione al di fuori del file .war della web application (file system) in modo di mantenere separati la configurazione sistemistica delle chiavi crittografiche rispetto al contenuto del rilascio applicativo.

Esempio di configurazione:

```
<context-param>
    <param-name>encryptAuthResponseKeyFile</param-name>
    <!--  path relativo o assoluto -->
    <param-value>
        /app/keystore/responsekeyfile.key
    </param-value>
</context-param>

<context-param>
    <param-name>encryptAuthResponseKeyFilePassPhrase</param-name>
    <param-value>p@ssw0rd</param-value>
</context-param>
```

Nel caso in cui, per un errore di configurazione, l'AssertionConsumer e il ResponseReceiver non utilizzino le stesse chiavi di cifra/decifra (o la stessa *PassPhrase*) viene generata un'eccezione del genere:

```
it.lisit.idpc.ri.web.AuthenticationResponseReceiverServlet] - ERROR - doPost() - Impossibile creare Bean contenente i dati di autenticazione
org.apache.xmlbeans.XmlException: Illegal XML character: 0xc
at org.apache.xmlbeans.impl.store.Root.loadXml(Root.java:1053)
 at org.apache.xmlbeans.impl.schema.SchemaTypeLoaderBase.parse(SchemaTypeLoaderBase.java:200)
 at it.people.sirac.authdataholder.xml.AuthenticationResponseDocument$Factory.parse(AuthenticationResponseDocument.java:50)
 at it.lisit.idpc.ri.authdataholder.AuthDataHolderFactory.createAuthDataHolder(AuthDataHolderFactory.java:535)
 at it.lisit.idpc.ri.web.AuthenticationResponseReceiverServlet.doPost(AuthenticationResponseReceiverServlet.java:782)
```

Per una configurazione più modulare in cui il pathname del file chiave non sia "cablato" in web.xml si deve rimuovere il parametro **`encryptAuthResponseKeyFile`** dal file web.xml e configurare il pathname assoluto all'interno del file IDPC-RI.properties (caricato via classloader) tramite la proprietà **encryptAuthResponseKeyFile**.
Analogamente si può esternalizzare anche la *PassPhrase* rimuovendo il parametro **`encryptAuthResponseKeyFilePassPhrase`** dal file web.xml e configurandone il valore all'interno del file IDPC-RI.properties tramite la proprietà **encryptAuthResponseKeyFilePassPhrase**.

4.1.2.3 Web Application

Parametri generali che coinvolgono il comportamento della web application:vengono definiti mediante il tag `<context-param>`:

param-name	param-value
authenticationResponseReceiverServiceTransferMode	Valorizzata a FORWARD (raccomandata): indica la modalità utilizzata nel trasferimento dei dati dall'*Assertion Consumer* al *Response Receiver*
assertionIntendedRecipientURL	Indirizzo URL che deve essere indicato nell'asserzione SAML ricevuta

Ad esempio:

```
<context-param>
       <param-name>
           authenticationResponseReceiverServiceTransferMode
       </param-name>
       <!--  Indicare FORWARD o POST -->
       <param-value>FORWARD</param-value>
</context-param>
```

4.1.2.4 Access Check

Vengono qui elencate le configurazioni che modificano il comportamento dell'*Access Check*.

Configurazione filtro

Il filtro è definito come:

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

Nella versione distribuita è attivo sui seguenti URL pattern:

```
/protected
*.do
```

Parametri di inizializzazione del filtro (<init-param>)

param-name	param-value
SIRACGatewayRedirectURL	Indirizzo URL completo della servlet AuthGatewayServlet
assertionConsumerURL	Indirizzo URL completo dell'Assertion Consumer.
postAuthRequestPage	Pagina JSP per la creazione della form autopostante verso la servlet di gateway

Servlet AuthGateway

```
<servlet>
       <servlet-name>AuthGatewayServlet</servlet-name>
       <display-name>
```

```
            SIRAC Authentication Gateway Servlet
        </display-name>
        <description>SIRAC Authentication Gateway Servlet</description>
        <servlet-class>
            it.lisit.idpc.ri.web.AuthGatewayServlet
        </servlet-class>
        …
```

Parametri di inizializzazione gateway (<init-param>)

param-name	param-value
loginRedirectPageUrl	Indirizzo URL del servizio **IDPC**

4.1.2.5 Assertion Consumer

L'*Assertion Consumer* è implementato mediante una servlet la cui definizione è contenuta in `WEB-INF/web.xml`:

```
<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>
      …
```

Nella versione distribuita, sono definiti i seguenti servlet mapping:

`/AssertionConsumerService`

Parametri di inizializzazione (<init-param>)

param-name	param-value
keystorePath	Percorso del trust store contenente i certificati di CA
keystorePassword	Password per l'accesso al trust store
authenticationResponseReceiverServiceForwardURL	URL della risorsa Response Receiver qualora venga utilizzata la modalità FORWARD
authenticationResponseReceiverServicePOSTURL	URL della risorsa Response Receiver qualora venga utilizzata la modalità POST
postResponsePage	Percorso della pagina JSP per la creazione della form auto postante

4.1.2.6 Response Receiver

Il *Response Receiver* è implementato mediante una servlet la cui definizione è contenuta in `WEB-INF/web.xml`:

```
    <servlet>
        <servlet-name>AuthResponseReceiverService</servlet-name>
        <display-name>
            Authentication Response Receiver Service
        </display-name>
        <description>
            Authentication Response Receiver Service
        </description>
        <servlet-class>
            it.lisit.idpc.ri.web.AuthenticationResponseReceiverServlet
        </servlet-class>
    </servlet>
```

4.1.2.7 Parametro "profile"

E' possibile specificare all'IdPC quale dei quattro "profili" di dati utente utilizzare.
In tabella vengono presentati i quattro profili possibili :

Valore parametro "profile"	Campi Visualizzati
1	solo codice fiscale
2	codice fiscale,nome e cognome
3	codice fiscale,nome,cognome ed e-mail
4	tutti i dati
non valorizzato o valorizzato con valori non concessi	tutti i dati

Per utilizzare uno dei quattro profili è necessario semplicemente aggiungere il sudetto parametro nella chiamata IdPC come mostrato in questo esempio :

```
https://idpcrl.crs.lombardia.it/scauth/SSLAuthServlet?profile=1&TARGET=https://
host-erogatore:port/AssertionConsumer?target=http://host-
erogatore:port/servicePage?serviceParameters...
```

In questo modo arrivati alla pagina di conferma dell'IdPC verrà mostrato all'utente il solo campo codice fiscale, e la saml response inviata alla web application che eroga il servizio conterrà oltre a questo attributo , i soli campi CNS_CARTA_REALE, CNS_ISSUER, CNS_SUBJECT

4.1.2.8 Configurazione IDPC-RI.properties

Il file **IDPC-RI.properties**, è da installare in una directory del classpath dell'application server, contiene alcuni parametri personalizzabili:

- **NOME_MACCHINA :** nome logico o indirizzo ip (più eventuale la porta) su cui è deployata la web application che si desidera proteggere con l'IDPC ;
- **NOME MACCHINA ASSERTION CONSUMER :** nome logico o indirizzo ip(piu eventualmente la porta) su cui si trova il servizio AssertionConsumer ;

ed inoltre, se non presenti nel file web.xml:

- **encryptAuthResponseKeyFile :** path assoluto della chiave di cifratura condivisa tra le componenti AssertionConsumer e ResponseReceiver ;
- **encryptAuthResponseKeyFilePassPhrase :** password che protegge la chiave di cifratura di cui sopra.

4.1.2.9 Segnalazione degli errori

Tutti gli errori di run time della web application sono gestiti dalla pagina `/error.jsp` mediante relativa configurazione nel file `web.xml`.

4.1.2.10 Compatibility issues

La tag library http://java.sun.com/jstl/core , referenziata nella pagina top.jsp, è compatibile con i web server / application server che sono compliant alle servlet sino alla release 2.3 compresa.

Se l'applicazione che integra la Reference Implementation gira in un container "servlet 2.4 compliant", è necessario che effettui la seguente modifica alla pagina top.jsp:

Sostituire la seguente riga:
<%@ taglib uri=http://java.sun.com/jstl/core prefix="c" %>

con:
<%@ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c" %>

Segnalibro FineDoc – Non Cancellare