**Allegato 2 del Capitolato Tecnico**

**INFOCAMERE S.C.P.A.**

# "Servizi di Certification Authority"

*** *** ***

# CAPITOLATO TECNICO - Specifiche tecniche modulo di firma on-line

*** *** ***

# INDICE

**1 Introduzione al documento ..... 3**
1.1 Scopo e campo di applicazione del documento ..... 3
1.2 Termini e definizioni ..... **Errore. Il segnalibro non è definito.**
**2 Schema architetturale ..... 4**
**3 Diagramma di sequenza ..... 5**
**4 Firma e verifica firma ..... 7**
4.1 Verifica firma ..... 7
**5 SPECIFICHE FUNZIONALI SERVER ..... 8**
5.1 Servlet StartSession ..... 8
5.1.1 Input ..... 8
5.1.2 Operazioni eseguite ..... 8
5.1.3 Output ..... 8
5.2 Servlet VerifyService ..... 9
5.2.1 Input ..... 9
5.2.2 Operazioni eseguite ..... 9
5.2.3 Output ..... 9
5.3 Servlet ResultService ..... 9
5.3.1 Input ..... 9
5.3.2 Operazioni eseguite ..... 9
5.3.3 Output ..... 10
5.4 Servlet VerifyDataService ..... 10
5.4.1 Input ..... 10
5.4.2 Operazioni eseguite ..... 11
5.4.3 Output ..... 12
5.4.4 Vincoli/criticità ..... 12
5.5 Policy ..... 12

# 1 Introduzione al documento

## 1.1 Scopo e campo di applicazione del documento

Il documento ha l'obiettivo di descrivere il modulo software integrabile nelle applicazioni Web che consente di effettuare tutte le operazioni di firma on-line in modo automatico.

## 2 Schema architetturale

Il modulo software integrabile nelle applicazioni Web è un prodotto costituito da componenti software che agiscono sia lato client che lato server.

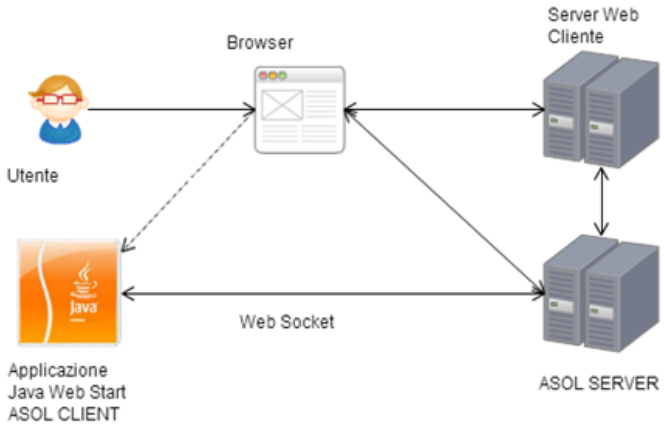
Il prodotto Web è un prodotto modulare costituito da due componenti:

1. Una componente client costituita da un applicazione Java Web Start che consente di effettuare firma PKCS7 e Pades.
2. Una componente server costituita da una WebApplication java che espone servizi di firma e verifica firma e che è in grado di gestire, sulla base di file di configurazione (policy) servizi aggiuntivi quali:
    a. La richiesta di marche temporali;
    b. La verifica delle firme

L'abbinamento dei due componenti consente di eseguire due processi.

1. il processo di "firma e verifica firma" .
2. il processo di "verifica firma".

Di seguito uno schema architetturale:



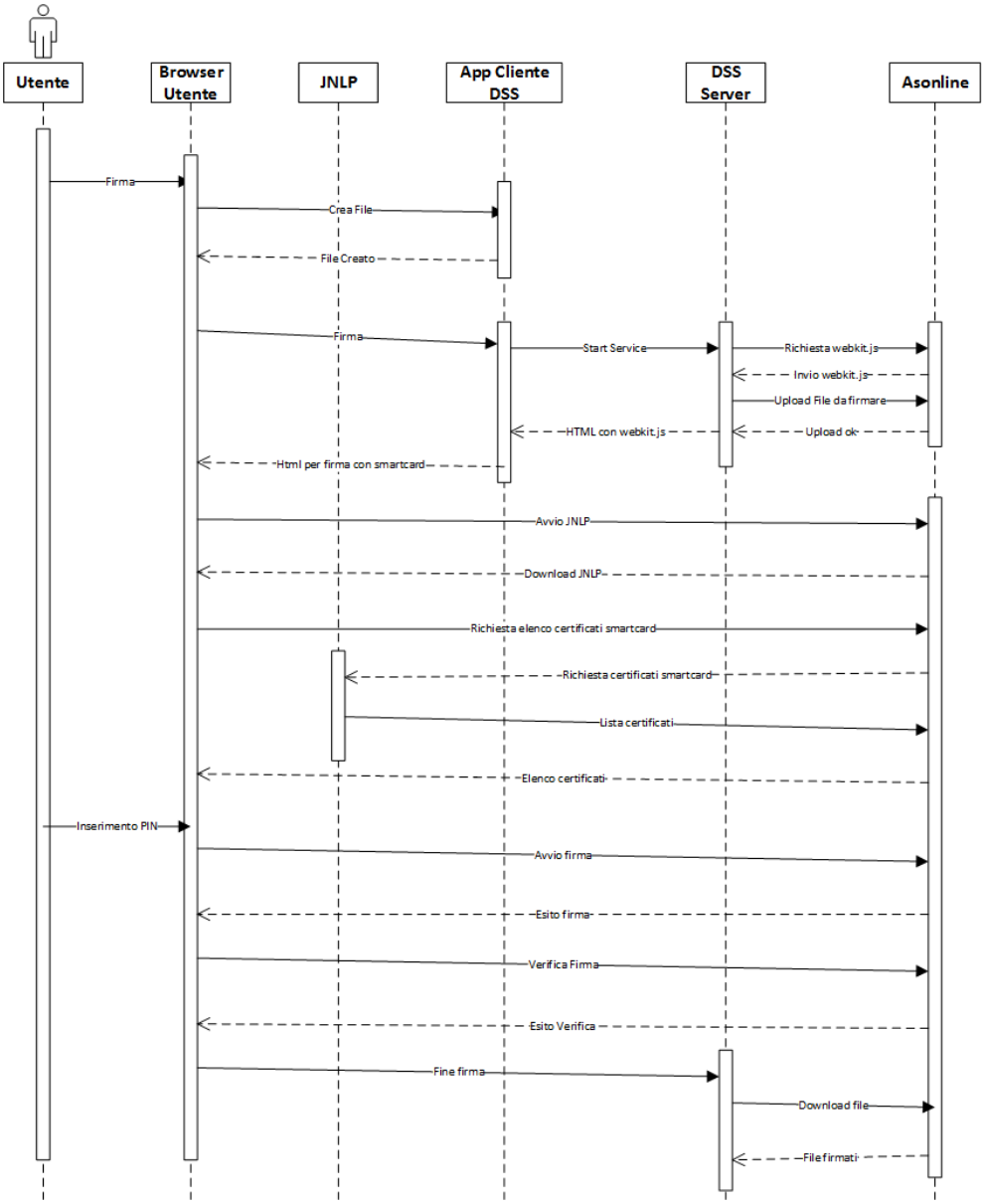

Gli attori nella soluzione sono quattro:

- L'utente che si interfaccia alla soluzione tramite browser,
- L'applicazione java web start
- L'applicazione server
- L'applicazione web del cliente.

Di fatto un cliente che vuole integrare la soluzione all'interno delle sue pagine web deve soltanto includere un javascript generato da ASOL SERVER e richiamarne i metodi.
La libreria javascript si occupa poi di interfacciare i vari componenti: server e client.
La comunicazione tra il componente ASOnline Server e la JNLP avviene tramite protocollo Web Socket, che sfrutta la porta 80 come tutte le richieste internet e quindi limita al massimo i problemi legati a firewall e antivirus.

## 3 Diagramma di sequenza

Di seguito è riportato un esempio del dialogo tra i vari attori per eseguire la procedura di firma:



Una volta preparato il file da firmare, il server cliente potrà inviare il file firmato alla componente Server tramite apposito webservice.

La firma non verrà effettuata tramite soluzione basata su Java Web Start.
Il server ASOnline genererà un apposito javascript che andrà inserito all'interno della pagina web.

La pagina web mostrerà l'interfaccia di scelta certificati e inserimento pin e, una volta selezionata l'operazione di firma, comunicherà tramite il javascript con il componente lato server che a sua volta chiederà al componente lato client (JNLP) di interfacciarsi con la smart-card per effettuare la firma.

Una volta terminata l'operazione, il server cliente potrà scaricare il file firmato tramite apposito webservice.

# 4 Firma e verifica firma

L'operazione di firma e verifica firma avviene in tre fasi corrispondenti all'invocazione di tre servlet distinte.
In una prima fase un client richiama la servlet "StartService" la quale si occupa semplicemente di:

1. Scaricare il documento da firmare se viene specificato un URL e salvarlo sul file system costruendo una complessa alberatura di directory.
2. Fare alcune verifiche di coerenza tra dati passati e policy da utilizzare.

La seconda fase è eseguita dall'applicazione Java Web Start che scarica dal server i file, li firma a seconda della modalità impostata e li rimanda firmati al server in particolare ad una servlet denominata "VerifyService".

Nella terza ed ultima fase un client richiama una servlet denominata "ResultService" che consente di richiedere l'esito dell'operazione, il risultato della verifica firma ed opzionalmente il dato firmato.
Il legame tra le tre servlet avviene attraverso una stringa di sessione detta "SESSIONID" che darà il nome alla directory in cui sono salvati i dati nelle varie operazioni.

## 4.1 Verifica firma

L'operazione di verifica firma è assegnata ad una servlet che si chiama "VerifyDataService" e che prevede lo stesso input della "StartService" in cui però i dati passati in input non sono da interpretarsi come dati da firmare ma dati da verificare.
Questa servlet si limita a verificare il dato, secondo le configurazioni della policy indicata nell'invocazione e produce una risposta in formato XML.

# 5 SPECIFICHE FUNZIONALI SERVER

Il Server Web è costituito da quattro servlet descritte di seguito.

## 5.1 Servlet StartSession

E' la servlet di avvio della sessione.

### 5.1.1 Input

```
<INPUTDATA>
<POLICYID>valore_obbligatorio</POLICYID>
<SESSIONID>valore_obbligatorio</SESSIONID>
<SIGNERCN>valore_opzionale</SIGNERCN>
<PIN>valore_opzionale</PIN>
<DATATOSIGN></DATATOSIGN>
oppure
<URLDATATOSIGN></URLDATATOSIGN>
oppure
<LISTURLDATATOSIGN>
<URL></URL>
<URL></URL>
......
</LISTURLDATATOSIGN>
</INPUTDATA>
```

### 5.1.2 Operazioni eseguite

1. Controllo esistenza policy;
2. Controllo che sessionid sia univoco. Nel caso di mancanza di univocità viene sempre restituito un errore.
3. Controllo coerenza dati:
   - Presenza DATATOSIGN e che dataInput nel file policy sia 1;
   - Presenza URLDATATOSIGN e che dataInput nel file policy sia 2;
   - LISTURLTOSIGN e che dataInput nel file policy sia 2;
4. Creazione cartella con id sessionid e salvataggio del file in questa cartella;
5. Costruzione della pagina HTML con il Javascript necessario alle operazioni di firma
6. Restituzione della struttura XML come dato di output;

### 5.1.3 Output

L'invocazione della servlet conterrà in risposta il seguente XML

```
<DSS-RESPONSE>
```

```
<SESSIONID>valore_session_ID</SESSIONID>
<RETURNCODE>codice_numerico</RETURNCODE>
<RETURNMSG>messaggio_di_testo</RETURNMSG>
<HTMLPAGE>pagina_html_template_elaborata</HTMLPAGE>
</DSS-RESPONSE>
```

Nell'elemento pagina_html_template_elaborata sarà inserita la pagina HTML contenente informazioni necessarie alla firma:

1. Tipo di firma da apporre. Nel caso di firma PaDES eventuali informazioni a corredo tipo firma nome versione, campo invisibile/visibile, name, location ecc.
2. URL del o dei documenti da firmare.
3. Sessionid per le successive invocazioni di verifica.

## 5.2 Servlet VerifyService

### 5.2.1 Input

Dati provenienti dal Javascript, verrà passato il file firmato ed il sessioni id.

### 5.2.2 Operazioni eseguite

1. Con il sessionid viene salvata la firma nella opportuna directory e recuperata la policy.
2. Se sessionid è sconosciuto o inesistente restituisce un errore e termina l'operazione.
3. Viene effettuata la verifica della Se l'operazione fallisce l'operazione di firma viene considerata fallita.
4. Si restituisce l'esito dell'operazione.
5. Con l'esito si costruisce l'XML di risposta da passare alla successiva invocazione.

### 5.2.3 Output

Sarà un URL di redirect da passare al javascript più un esito HTTP che sarà generalmente posto a 200 se è andato tutto a buon fine o a 500 nel caso di gravi errori.

## 5.3 Servlet ResultService

Consente di recuperare i risultati della elaborazione.

### 5.3.1 Input

```
<INPUTDATA>
<SESSIONID>Stringa alfanumerica</SESSIONID>
<ATTACHDATA>Intero</ATTACHDATA>
</INPUTDATA>
```

### 5.3.2 Operazioni eseguite

1. Verifica presenza directory corrispondente a sessionid;
    - Se directory mancante restituisce un errore.
2. Recupera l'XML generato con elaborazione sopra descritta.
    - Se XML mancante restituisce errore.

3. Se richiesto in attachdata viene inserito il o i dati firmati nell'XML.
4. Restituisce XML di risposta al chiamante.

### 5.3.3 Output

```
<DSS-RESPONSE>
<SESSIONID> Stringa alfanumerica</SESSIONID>
<RETURNCODE> Intero </RETURNCODE>
<RETURNMSG> Stringa </ RETURNMSG >
<RESULTVERIFY>
<RESULTDATA>
......
</RESULTDATA>
</RESULTVERIFY>
</DSS-RESPONSE>
```

Se non ci sono stati errori nella elaborazione

```
<RESULTDATA>
<ISSUERDN> </ISSUERDN>
<SUBJECTDN> </SUBJECTDN>
<SUBJECTCN> </SUBJECTCN>
<CERTIFICATO> </CERTIFICATO>
<HASHCERTIFICATO></ HASHCERTIFICATO>
<SIGNEDDATA filename="NomeFileInput.xxx">…</SIGNEDDATA>
<ORIGINALDATA filename="NomeFileInput.xxx">…</ORIGINALDATA>
<SIGNEDDATAFILE filename="NomeFileInput.xxx">…</SIGNEDDATAFILE>
<ESITO filename="NomeFileInput.xxx">……</ESITO>
<MESSAGGIOESITO filename="NomeFileInput.xxx">…</MESSAGGIOESITO>
<DATAORAFIRMA filename="NomeFileInput.xxx">…</DATAORAFIRMA>
<TST filename="NomeFileInput.xxx"> … …</TST>
<REC filename="NomeFileInput.xxx"> ……</REC>
<TOKENLDOC filename="NomeFileInput.xxx"> ……</TOKENLDOC>
</RESULTDATA>
```

## 5.4 Servlet VerifyDataService

### 5.4.1 Input

Come dato di input verrà mantenuto lo stesso file XML pertanto il formato di sarà:

```
<INPUTDATA>
<POLICYID>valore_obbligatorio</POLICYID>
<SESSIONID>valore_ obbligatorio</SESSIONID>
<SIGNERCN>valore_opzionale</SIGNERCN>
<PIN>valore_opzionale</PIN>
<DATATOSIGN></DATATOSIGN>
oppure
<URLDATATOSIGN></URLDATATOSIGN>
oppure
<LISTURLDATATOSIGN>
<URL></URL>
<URL></URL>
……
</LISTURLDATATOSIGN>
</INPUTDATA>
```

N.B Datatosign o Urldatatosign sono da interpretarsi come Datatoverify o Urldatatoverify.

### 5.4.2 Operazioni eseguite

1. Controllo che sessionid non sia esistente. Se esiste restituisce un errore.
2. Controllo esistenza policyId;
3. Controllo coerenza dati:
    1. Presenza DATATOSIGN e che dataInput nel file policy sia 1;
    2. Presenza URLDATATOSIGN e che dataInput nel file policy sia 2;
    3. Presenza LISTURLDATATOSIGN e che dataInput nel file policy sia 2;
4. Creazione cartella con il sessionid e salvataggio del file in questa cartella;
5. Invocazione operazione di verifica secondo quanto stabilito dalla policy o dalle configurazioni di default.
6. Costruzione del XML da inviare in risposta;
7. Restituzione dell’XML.

### 5.4.3 Output

```
<DSS-RESPONSE>
<SESSIONID> stringa alfanumerica </SESSIONID>
<RETURNCODE> intero </RETURNCODE>
<RETURNMSG> Stringa </RETURNMSG>
<VERIFYRESULT>
<VERIFYDATA>
<VALID> true/false </VALID>
<SIGNED> true/false </SIGNED>
<DOCUMENTNAME> nome file/URL di input </DOCUMENTNAME>
<DOCUMENTMSG> stringa </DOCUMENTMSG>
<SIGNATUREINFO>
<ISSUERDN></ISSUERDN>
<SUBJECTDN></SUBJECTDN>
<SUBJECTCN></SUBJECTCN>
<SUBJECTEMAILADDRESS></SUBJECTEMAILADDRESS>
<CERTIFICATE> certificato in base64 </CERTIFICATE>
<VERIFIED> true/false </VERIFIED>
<REVOCATIONCONTROLMETHOD></..>
<REVOCATIONCONTROLTIMESTAMP></..>
<SIGNATURETIME> data/ora di firma o Timestamp</…>
<ISTIMESTAMPED> true/false </ISTIMESTAMPED>
</SIGNATUREINFO>
<SIGNATUREINFO>
<ISSUERDN></ISSUERDN>
<SUBJECTDN></SUBJECTDN>
….
</SIGNATUREINFO>
</VERIFYDATA>
</VERIFYRESULT>
</DSS-RESPONSE>
```

### 5.4.4 Vincoli/criticità

SignatureInfo presenterà l'elenco di tutte le firme e controfirme trovate nel documento.

## 5.5 Policy

La configurazione di ogni tipologia di firma utilizzata da un'applicazione va definita in un profilo (policy).
All'interno deva essere specificato:

- Il nome del profilo di una applicazione.

- Il tipo di dati da firmare (testo, PDF, XML, P7M).
- La modalità di passaggio del file da firmare (da url o come stringa Base64).
- Se è richiesta o meno la verifica di integrità, attendibilità e validità (controllo CRL).
- Come restituire il file firmato (p7m, m7m, pdf,...).

Il file di policy conterrà le seguenti informazioni

```
[Nomi]
Politica=Politica di firma

[Data]
dataType= X
# valori possibili per i Dati Da Firmare
# 3 ==> XML
# 4 ==> PDF
# 5 ==> P7M Dati firmati in formato CADES

dataInput= X
# 1 ==> Direct Stringa B64 in input
# 2 ==> FROM_URL Dati da Url

[Operation]
signType= X
# valori possibili per operazione di imbustamento
# 0 ==> p7m (CADES)
# 1 ==> pdf
# 5 ==> xml (XADES)

timeStamp= X
# valori possibili per operazione di TimeStamp
# 0 ==> non richiedere operazione di TimeStamp
# 1 ==> richiedere operazione di TimeStamp

TSformat=X
# formato del TimeStamp
# 0 ==> DETACHED
# 1 ==> CADES-T per p7m o PADES-T per firma pdf
# 2 ==> TSD

verify= X
```

```
# valori possibili per operazione di verifica firme
# 0 ==> nessuna verifica
# 1 ==> verifica di integrità ed attendibilità
# 2 ==> verifica di integrità, attendibilità e validità (controllo CRL)

FileHtml=FileTemplate.html
ResultPageUrl=http://localhost:8080/DSS/index.html

localTScredentials=X
# valori possibili per il flag
# 0 ==> UTILIZZA CREDENZIALI CONTENUTE NEL CONFIG (DEFAULT)
# 1 ==> UTILIZZA CREDENZIALI CONTENUTE NELLA POLICY

timeStampUser=XXXXXXXX
timeStampPassword=XXXXXXXX

# firma grafica pdf
SignLayout=1;100;100;200;200
SignReason=Test interno
SignLocation=Infocert - 10601 - Test

hideSignatureBox=1
# valori possibili
# 0 ==> MOSTRA BOX DI FIRMA
# 1 ==> NASCONDI BOX DI FIRMA

restituisciDatiOriginali=1
# valori possibili
# 0 ==> NESSUN EFFETTO (DEFAULT)
# 1 ==> NODO <ORIGINALDATA> (V2) O <signedData> (V1)
```