# Guida alla programmazione e integrazione di servizi in OpenSPCoop

# Indice

# Elenco delle figure

# 1 Introduzione

Nella specifica SPCoop, un dominio è definito come il confine di responsabilità di un ente o soggetto amministrativo e racchiude al suo interno tutte le applicazioni da esso gestite.

Il confine applicativo del Dominio è rappresentato dalla Porta di Dominio (PdD), attraverso la quale devono transitare tutte le comunicazioni da e verso il dominio.

Le Porte di Dominio si parlano tra di loro scambiandosi richieste e risposte in un formato parte della specifica SPCoop, denominato busta eGov. Tuttavia il formato della busta non è parlato nativamente dalle applicazioni, pertanto la Porta di Dominio deve anche occuparsi di convertire le richieste applicative nel formato busta eGov.

Facendo riferimento a questa problematica, i compiti della Porta di Dominio vengono solitamente classificati in due componenti:

- il componente di cooperazione che riguarda la comunicazione tra le Porte di Dominio;
- il componente di integrazione che riguarda la comunicazione tra i Servizi Applicativi dell'Ente e la Porta di Dominio.

Il componente di integrazione si differenzia a sua volta in due diversi moduli: la porta delegata e la porta applicativa. In particolare la porta delegata è utilizzata come proxy per l'accesso al servizio destinazione e configurabile attraverso l'apposita interfaccia web come descritto nel documento Manuale Gestione, mentre la porta applicativa deve essere in grado di gestire la consegna dei contenuti delle buste eGov ricevute al corrispondente servizio applicativo interno al dominio destinazione.

Una volta predisposta l'infrastruttura di cooperazione, configurando adeguatamente Registro dei Servizi e Porte di Dominio, i Servizi Applicativi interni al dominio di un Soggetto possono essere abilitati ad interagire con le Porte Delegate e le Porte Applicative definite sulla propria Porta di Dominio.

Ci sono due modalità che è possibile utilizzare per la programmazione dei Servizi Applicativi: la modalità trasparente (Sezione 2) e il servizio di IntegrationManager (Sezione 3). La modalità trasparente è molto più semplice ed elegante da utilizzare, e se ne suggerisce quindi l'uso tutte le volte che questo sia possibile. Nel seguito di questa sezione vediamo come utilizzare le due modalità.

# 2 Modalità d'integrazione trasparente

Questa modalità prevede che il servizio applicativo utilizzi (in caso di porta delegata) o esponga (in caso di porta applicativa) le interfacce applicative native dei servizi così come registrate negli accordi di servizio; in tal caso la Porta di Dominio agisce come un proxy SOAP trasparente con funzionalità di imbustamento e sbustamento eGov dei messaggi applicativi; in questa modalità, gli applicativi potranno continuare ad operare esattamente come se stessero interagendo direttamente con il servizio applicativo dell'altro Ente.

L'invocazione della porta delegata in modalità trasparente può essere realizzata tramite gli strumenti del linguaggio di programmazione nativo del servizio applicativo, utilizzando ad esempio stub creati tramite il proprio ambiente di sviluppo Web Services (ad esempio wsdl2java in CXF), facendo riferimento direttamente al WSDL del servizio destinazione.

In questo caso la principale modifica rispetto all'invocazione dell'effettivo servizio destinazione sarà la URL utilizzata per l'invocazione http, che dovrà essere quella corrispondente alla porta delegata del servizio esposta dalla PdD. Il codice che segue mostra un esempio, usando CXF, di invocazione di una porta delegata usando gli stub del servizio.

```
ComunicazioneVariazioneService ss = new ComunicazioneVariazioneService(wsdlURL, ↩
    SERVICE_NAME);
ComunicazioneVariazione port = ss.getComunicazioneVariazioneInterfaceEndpoint();

org.openspcoop.example.ComunicazioneVariazione_Type notifica = new org.openspcoop.example. ↩
    ComunicazioneVariazione_Type();
notifica.setCF("BBBCCC11F11F111F");
notifica.setCodiceFiscale("DDDFFF22G22G222G");
notifica.setCognome("Rossi");
...

port.notifica(notifica);
```

È sufficiente modificare l'endpoint del servizio specificato nel WSDL con la URL della Porta Delegata da indirizzare, senza ulteriori modifiche al codice applicativo. Non è comunque obbligatorio utilizzare degli stub per l'invocazione del servizio, ma si può utilizzare qualunque altra modalità per la programmazione di web service, ad esempio attraverso un client SAAJ.

```
//Creare la SOAPConnection
SOAPConnectionFactory soapConnFactory = SOAPConnectionFactory.newInstance();
SOAPConnection connection = soapConnFactory.createConnection();

//Creare il SOAPMessage di richiesta
MessageFactory messageFactory = MessageFactory.newInstance();
MimeHeaders mhs = new MimeHeaders();
mhs.addHeader("Content-Type", "text/xml");
mhs.addHeader("SOAPAction", "");
FileInputStream is = new FileInputStream("xmlRequest.xml");
SOAPMessage request = messageFactory.createMessage(mhs,is);

//Settare l'URL dell'endpoint
if(!urlPD.endsWith("/"))
  urlPD = urlPD + "/";

String SOAPUrl = urlPD + PDLocation;

//Spedire la richiesta
connection.call(request, SOAPUrl);

//Chiudere la connessione
connection.close();
```

# 3 Modalità d'integrazione tramite il servizio di IntegrationManager

Questa modalità prevede che il servizio applicativo utilizzi le interfacce di un apposito web service di Integrazione, messo a disposizione dalla Porta di Dominio per la spedizione e/o la ricezione di messaggi applicativi da parte dei servizi applicativi del proprio Dominio di Servizi.

L'interfaccia WSDL completa dell'integration manager è disponibile alla URL *http://www.openspcoop.org/openspcoop/doc/-wsdl/1.1/IntegrationManager.wsdl*;

Nel seguito viene mostrata, a titolo descrittivo, l'interfaccia esposta dal Web Service, espressa in linguaggio Java:

```
interface IntegrationManager {

  SPCoopMessage invocaPortaDelegata(
  String portaDelegata, SPCoopMessage msg)

  SPCoopMessage sendRispostaAsincronaSimmetrica(
  String portaDelegata,SPCoopMessage msg)

  String[] getAllMessagesId()
  String[] getAllMessagesIdByService(
  String tipoServizio, String servizio, String azione)

  String[] getNextMessagesId(int n)
  String[] getNextMessagesIdByService(
  int n, String tipoServizio, String servizio, String azione)

  SPCoopMessage getMessage(String idEGov)
  SPCoopMessage getMessageByReference(String riferimentoMsg)

  void deleteMessage(String idEGov)
  void deleteMessageByReference(String riferimentoMsg)
```

```
  public void deleteAllMessages()
}
```

Il messaggio gestito tramite IntegrationManager viene reso accessibile tramite il tipo `SPCoopMessage`, che contiene varie informazioni relative al messaggio e un array di byte corrispondente al messaggio SOAP vero e proprio. La struttura della classe `SPCoopMessage` è mostrata a titolo descrittivo in linguaggio java nel riquadro seguente.

```
public class SPCoopMessage implements java.io.Serializable {

  public void setMessage(byte [] m)

  public byte[] getMessage()

  public boolean getImbustamento()
  public void setImbustamento(boolean imbustamento)

  public String getIDApplicativo()
  public void setIDApplicativo(String applicativo)

  public String getServizioApplicativo()
  public void setServizioApplicativo(String servizioApplicativo)

  public SPCoopHeaderInfo getSpcoopHeaderInfo()
  public void setSpcoopHeaderInfo(SPCoopHeaderInfo spcoopHeaderInfo)
}
```

Come risulta dalla struttura della classe mostrata nel riquadro precedente, l'oggetto `SPCoopMessage` riferisce un ulteriore oggetto di tipo `SPCoopHeaderInfo` che raccoglie tutte le informazioni relative all'header SPCoop della busta eGov accessibili dai servizi applicativi. La struttura della classe `SPCoopHeaderInfo` è mostrata a titolo descrittivo in linguaggio java nel riquadro seguente.

```
public class SPCoopHeaderInfo implements java.io.Serializable {

  public String getTipoMittente()
  public void setTipoMittente(String type )

  public String getMittente()
  public void setMittente(String m )

  public String getTipoDestinatario()
  public void setTipoDestinatario(String type )

  public String getDestinatario()
  public void setDestinatario(String s )

  public String getServizio()
  public void setServizio(String s )

  public String getTipoServizio()
  public void setTipoServizio(String type )

  public String getAzione()
  public void setAzione(String a )

  public String getID()
  public void setID(String id )

  public String getRiferimentoMessaggio()
  public void setRiferimentoMessaggio(String rif )

  public String getIdCollaborazione()
  public void setIdCollaborazione(String idCollaborazione)
```

```
}
```

### 3.1 Invocazione di una Porta Delegata tramite Integration Manager

Il codice che segue mostra un esempio, in CXF, di invocazione di una porta delegata usando il servizio di IntegrationManager usando gli stub generati con il tool wsdl2java.

```
IntegrationManagerService im = new IntegrationManagerService(new URL("http://localhost ↩
    :8080/openspcoop/IntegrationManager?wsdl"),
                                                                 new QName("http://services.pdd ↩
                                                                     .openspcoop.org", " ↩
                                                                     IntegrationManagerService") ↩
                                                                     );
IntegrationManager imPort = im.getIntegrationManager();

java.lang.String portaDelegata = "ComunicazioneVariazione_PD";

SPCoopMessage msgRequest = new SPCoopMessage();
String xmlRequest = "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/ ↩
    envelope/\" "
                  + "xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" "
                  + "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\">"
                  + "<soapenv:Body>"
                  + "<comunicazioneVariazione xmlns=\"http://www.openspcoop.org/example\"  ↩
                      CF=\"BBBCCC11F11F111F\">"
                  + "<nome>Mario</nome><cognome>Rossi</cognome><codiceFiscale> ↩
                      DDDFFF22G22G222G</codiceFiscale>"
                  + "<nascita>1980-01-01T12:00:00.000Z</nascita><statoCivile>Celibe</ ↩
                      statoCivile></comunicazioneVariazione>"
                  + "</soapenv:Body></soapenv:Envelope>";

msgRequest.setMessage(xmlRequest.getBytes());

try {
  SPCoopMessage msgResponse = imPort.invocaPortaDelegata(portaDelegata, msgRequest);
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  baos.write(msgResponse.getMessage());
  System.out.println("invocaPortaDelegata.result = " + baos.toString());
}
catch (SPCoopException_Exception e) {
  System.out.println("Exception: SPCoopException has occurred.");
  System.out.println(e.toString());
}
```

### 3.2 Accesso ai Messaggi Ricevuti Tramite Integration Manager

Il codice che segue mostra un esempio, in CXF, di accesso ai messaggi ricevuti su una porta applicativa usando il servizio di IntegrationManager.

```
// Inizializzazione punto di accesso al servizio
IntegrationManagerService imService = new IntegrationManagerService(new URL("http:// ↩
    localhost:8080/openspcoop/IntegrationManager?wsdl"),
                                                                 new QName("http://services.pdd ↩
                                                                     .openspcoop.org", " ↩
                                                                     IntegrationManagerService") ↩
                                                                     );
IntegrationManager imPort = imService.getIntegrationManager();
```

```
// Impostazione endpoint e credenziali del servizio
((BindingProvider)imPort).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY ↩
    ,  "http://localhost:8080/openspcoop/IntegrationManager");
((BindingProvider)imPort).getRequestContext().put(BindingProvider.USERNAME_PROPERTY,   ↩
    username);
((BindingProvider)imPort).getRequestContext().put(BindingProvider.PASSWORD_PROPERTY,   ↩
    password);

// Prelievo degli identificatori dei messaggi presenti nella Message Box
String []ids =  imPort.getAllMessagesId();

// Ciclo di lettura, processamento e cancellazione di ogni singolo messaggio
for (int i=0; i&lt;ids.length(); i++ ) {

  SPCoopMessage msg = imPort.getMessage(ids[i]);

  // processa il messaggio ricevuto come byte array ...
  processMessage(msg.getMessage());

  imPort.deleteMessage(ids[i]);
}
```

Nell'esempio precedente viene prima utilizzata l'operazione `getAllMessagesId` per accedere all'elenco dei messaggi ricevuti e quindi l'operazione di `getMessage` per prelevare i singoli messaggi. Dopo il processamento di un messaggio, si procede alla sua rimozione.

Oltre a questi due metodi di base, il servizio di IntegrationManager fornisce ulteriori interfacce per gestire in vario modo i messaggi in arrivo per un SIL. Nel riquadro successivo vengono elencate le varie interfacce utili per l'accesso ai messaggi.

```
String[] getAllMessagesId()
// restituisce gli identificatori di tutti i messaggi disponibili
// in coda per il SIL richiedente

String[] getAllMessagesIdByService(
String tipoServizio, String servizio, String azione)
// restituisce gli identificatori di tutti i messaggi disponibili in
// coda per il SIL corrispondente al servizio ed all'azione
// indicati in input

String[] getNextMessagesId(int n)
// restituisce i primi n identificatori di messaggi disponibili
// in coda per il SIL

String[] getNextMessagesIdByService(
int n, String tipoServizio, String servizio, String azione)
// restituisce i primi n identificatori di messaggi disponibili
// in coda per il SIL corrispondenti al servizio ed all'azione
// indicati in input

SPCoopMessage getMessage(String idEGov)
// restituisce il messaggio identificato dall'identificatore in input

SPCoopMessage getMessageByReference(String riferimentoMsg)
// restituisce il messaggio identificato dall'identificatore
// di riferimentoMessaggio in input

void deleteMessage(String idEGov)
// cancella il messaggio identificato dall'id in input

void deleteMessageByReference(String riferimentoMsg)
// cancella il messaggio identificato dall'id di riferimentoMessaggio
// in input
```

```
public void deleteAllMessages()
// cancella tutti i messaggi in coda per il SIL richiedente
```

# 4 Aspetti specifici di Integrazione nei vari Profili di Collaborazione

In funzione dei singoli profili utilizzati, ci sono delle differenze nelle modalità d'invocazione e nelle risposte restituite al richiedente. Nelle immagini a seguire sono mostrati i messaggi scambiati tra i partecipanti alla comunicazione:

- Le *frecce blu* indicano i messaggi di richiesta per il protocollo HTTP
- Le *frecce rosse* indicano i messaggi di risposta per il protocollo HTTP
- Le *frecce continue* indicano un messaggio con un payload non vuoto
- Le *frecce tratteggiate* indicano un messaggio con un payload vuoto (salvo eccezioni descritte in seguito)

## 4.1 Profilo OneWay

Nel caso del profilo Oneway, la PdD mittente restituisce la risposta al servizio fruitore immediatamente, senza attendere di ricevere una risposta dal servizio applicativo erogatore del servizio.


Figura 1: Profilo Oneway

In questo caso il servizio fruitore riceverà quindi una risposta creata dalla PdD con payload vuoto. Solo nel caso in cui si verifichino problemi nella PdD mittente, il messaggio di risposta al servizio fruitore potrà contenere un SOAP Fault.

## 4.2 Profilo Sincrono

Nel caso di profilo Sincrono, la PdD mittente lascia in attesa il servizio mittente, fino al ricevimento della risposta del servizio erogatore.


Figura 2: Profilo Sincrono

La risposta ricevuta dal servizio fruitore è quindi esattamente corrispondente a quanto prodotto dal servizio erogatore.

## 4.3 Profili Asincroni

I due profili di collaborazione asincroni richiedono invece due interazioni, una richiesta di servizio ed una risposta del servizio che viene ritornata tramite una nuova interazione.

- Nel profilo *Asincrono Simmetrico* il fruitore effettua la richiesta e sarà l'erogatore, in un secondo momento e su una nuova connessione, a avviare la comunicazione per consegnare il messaggio di risposta.
- Nel profilo *Asincrono Asimmetrico* il fruitore effettua la richiesta e sarà sempre lui a richiedere, su una nuova connessione, la risposta all'erogatore.

Entrano quindi in gioco quattro buste per ogni transazione, correlate tra di loro attraverso un comune identificativo eGov. Pertanto i servizi applicativi devono necessariamente scambiare con la Porta di Dominio anche l'informazione relativa all'identificativo eGov, oltre agli usuali contenuti del messaggio applicativo. Per le modalità con cui un servizio applicativo può scambiare l'informazione relativa all'identificatore eGov con la PdD, si rimanda alla Sezione 5.

OpenSPCoop supporta le modalità di interazione sincrona e asincrona tra Servizi Applicativi e PdD in caso di profili asincroni, entrambe compatibili con la specifica SPCoop, in attesa che la specifica sia disambiguata su questo specifico aspetto. Il tipo di modalità utilizzata si può configurare tramite i parametri *ricevuta-asincrona-simmetrica* e *ricevuta-asincrona-asimmetrica* delle porte delegate e applicative.

### 4.3.1 Implementazione Asincrona della Richiesta nei Profili Asincroni

Un primo comportamento della Porta di Dominio consiste nel restituire immediatamente una risposta alle richieste del Servizio Applicativo mittente, in entrambe le interazioni dell'Asincrono Simmetrico. Nel frattempo, in maniera del tutto disaccoppiata dall'interazione con il mittente, si procede all'invio della busta alla PdD del destinatario. Tale PdD replica con una ricevuta SPCoop con SoapBody vuoto, provvedendo anche alla consegna al servizio applicativo destinatario.

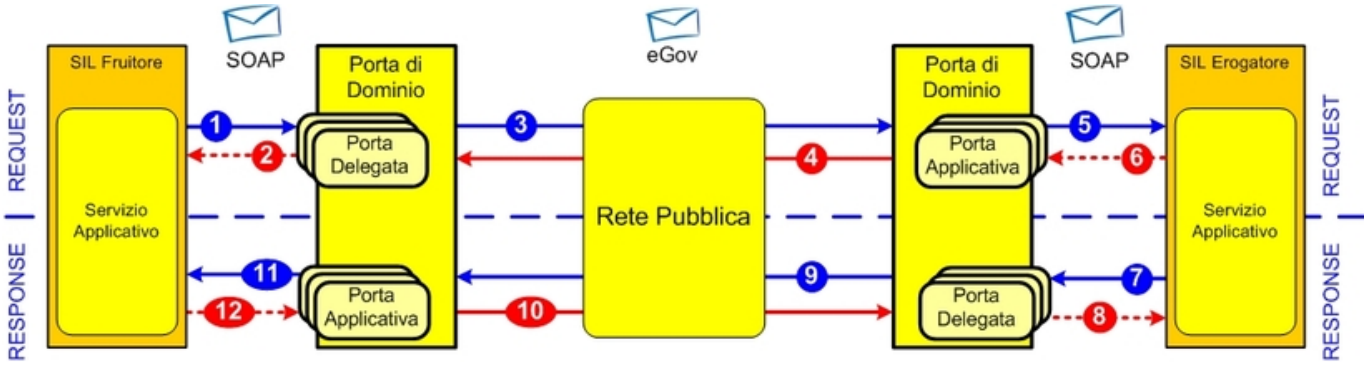


Figura 3: Profilo Asincrono Simmetrico con implementazione Asincrona

Per quanto riguarda il profilo di collaborazione Asincrono Asimmetrico solo la richiesta potrà essere gestita in questo modo

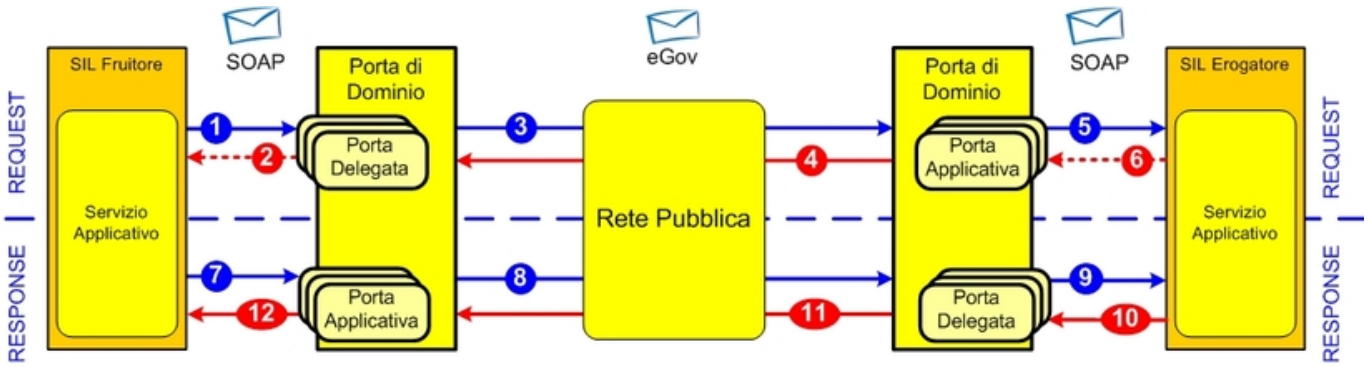


Figura 4: Profilo Asincrono Asimmetrico con implementazione Asincrona

Quindi, dal punto di vista dei servizi applicativi, entrambe le interazioni dell'Asincrono Simmetrico e la richiesta dell'Asincrono Asimmetrico vengono implementate in maniera analoga ad invocazioni del profilo Oneway, eccezion fatta per l'inclusione dell'identificativo di correlazione nelle risposte HTTP. I due servizi applicativi devono implementare queste operazioni in accordo a wsdl:operation che possiedano solo un input (il web service non produce un messaggio di output). Esempio:

```
<wsdl:binding name="ServiceSoapBinding" type="erogatore:Service">

  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

  <wsdl:operation name="actionExample">
    <wsdlsoap:operation soapAction="SoapAction"/>
    <wsdl:input name="richiesta_risposta_richiestaStato">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>

</wsdl:binding>
```

Il servizio applicativo mittente, che invoca una porta delegata, viene quindi immediatamente sbloccato, tramite un messaggio SOAP con body vuoto generato direttamente dalla Porta di Dominio.

#### 4.3.2 Implementazione Sincrona della Richiesta nei Profili Asincroni

Un secondo comportamento consiste invece nel lasciare il richiedente in attesa fino alla ricezione della ricevuta, potendo quindi poi restituire al richiedente il soapBody ottenuto nella ricevuta. In questa implementazione, la porta applicativa della Porta destinataria non ignora la risposta del servizio applicativo, ma la inserisce nella ricevuta stessa.

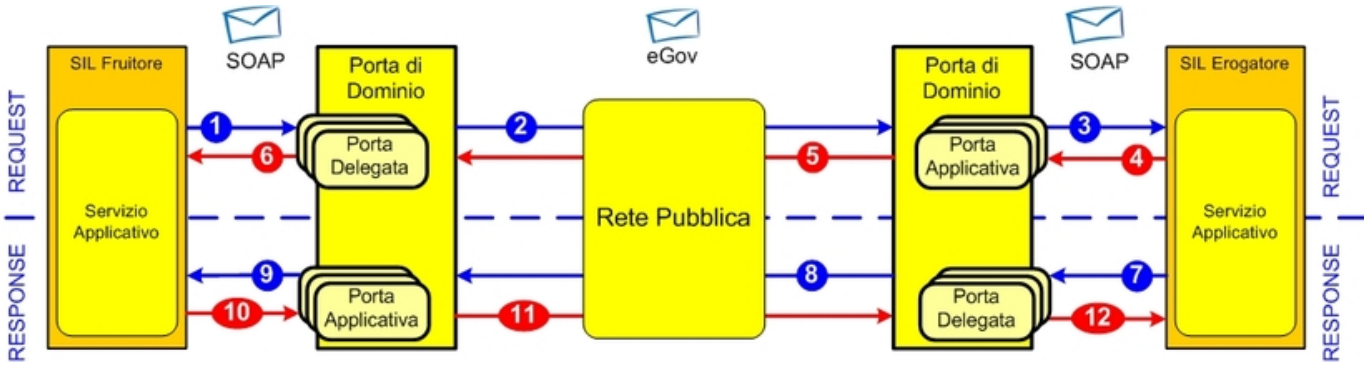


Figura 5: Profilo Asincrono Simmetrico con implementazione Sincrona

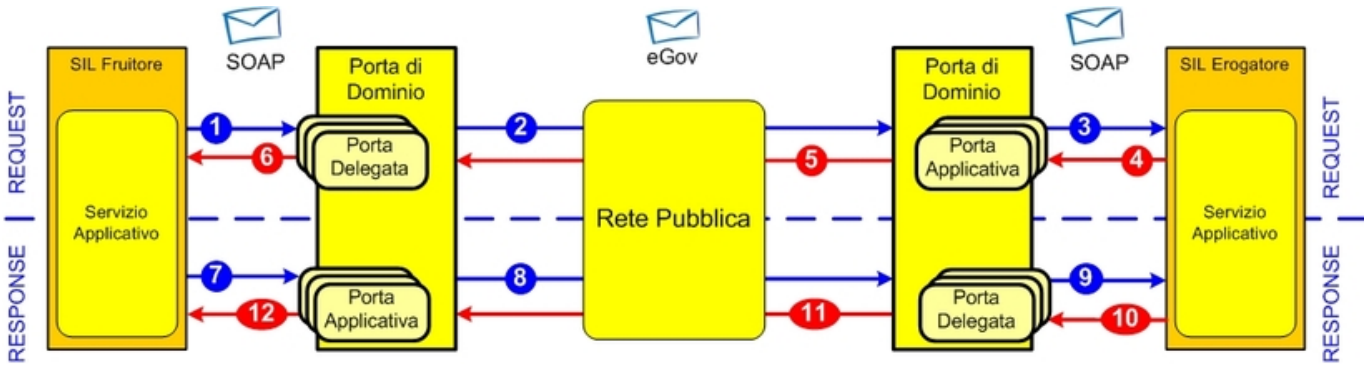


Figura 6: Profilo Asincrono Asimmetrico con implementazione Sincrona

In questo caso, quindi, dal punto del servizio applicativo, tutte le interazioni previste nei profili asincroni vengono implementate in maniera analoga ad invocazioni del profilo Sincrono. I due servizi applicativi devono implementare quindi tutte le operazioni in accordo ad una `wsdl:operation`, che possieda sia un input che un output. Esempio:

```
<wsdl:binding name="ServiceSoapBinding" type="erogatore:Service">

  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

  <wsdl:operation name="actionExample">
    <wsdlsoap:operation soapAction="SoapAction"/>
    <wsdl:input name="richiesta_risposta_richiestaStato">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="ricevuta">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>

</wsdl:binding>
```

Il servizio applicativo mittente, che invoca una porta delegata, rimane quindi in attesa del contenuto applicativo portato nella ricevuta asincrona.

È da notare come i due comportamenti appena esaminati non siano tra loro interoperabili. Ad esempio, nel caso in cui un Accordo di Servizio faccia riferimento ad una operazione del WSDL che possieda sia un input che un output, se la porta del fruitore utilizza la modalità sincrona e la porta dell'erogatore utilizza la modalità asincrona, la PdD del fruitore riceverà una ricevuta con SoapBody vuoto. Il Servizio Applicativo mittente, però, si aspetterà di ricevere la risposta applicativa, in accordo al WSDL del servizio, e potrebbe quindi rifiutare la risposta ricevuta.

---

**Nota**
La differenza di comportamento tra le due implementazioni del profilo Asincrono non si applica al caso della seconda interazione del profilo asincrono asimmetrico. In questo caso, infatti, il servizio applicativo dovrà necessariamente implementare una wsdl:operation che possieda sia un input che un output. La ricevuta asincrona porta quindi sempre il contenuto applicativo del servizio applicativo erogatore invocato per effettuare la richiesta stato, poiché l'output prodotto dal servizio applicativo erogatore corrisponde alla risposta richiesta dal servizio applicativo fruitore (polling dello stato dell'operazione asincrona asimmetrica). Il servizio applicativo fruitore, che invoca una porta delegata relativa ad una richiesta stato asincrona asimmetrica, rimane quindi in attesa del contenuto applicativo portato nella ricevuta asincrona.

---

# 5 Interscambio di informazioni tra Servizi Applicativi e Porte di Dominio

In alcune situazioni, i Servizi Applicativi devono avere una precisa visibilità dei messaggi SPCoop scambiati tra le Porte di Dominio. Una situazione del genere si verifica, ad esempio, quando diversi messaggi SPCoop sono correlati tra loro, come nel caso dell'uso dell'id di collaborazione o dei profili di collaborazione Asincroni, sia simmetrico che asimmetrico.

Per questo, alcune delle informazioni parte dell'header SPCoop della busta eGov possono essere scambiate tra il Servizio Applicativo e la PdD al momento dell'invocazione di una porta delegata o tra la PdD ed il Servizio Applicativo al momento dell'invocazione di una porta applicativa. In particolare le informazioni in questione variano in funzione delle specifiche precipuità degli Accordi di Servizio a cui la busta si riferisce e sono identificate tramite le seguenti keyword:

- SpCoopID
- SPCoopTipoMittente
- SPCoopMittente
- SPCoopTipoDestinatario
- SPCoopDestinatario
- SPCoopTipoServizio
- SPCoopServizio
- SPCoopAzione
- SPCoopRiferimentoMessaggio
- SPCoopIdCollaborazione

Nel caso di uso dell'IntegrationManager, tali informazioni sono accessibili tramite le interfacce di get/set della classe SPCoopHeaderInfo, già mostrata in precedenza.

Nel caso di uso della modalità trasparente, tali informazioni sono invece accessibili tramite quattro diverse modalità:

- come header del trasporto http: in questo caso i nomi degli header saranno uguali alle informazioni sopra elencate;
- come proprietà della QUERY_STRING della URL http invocata: in questo caso il nome della proprietà saranno uguali alle informazioni sopra elencate;
- come informazioni interne ad un'header SOAP di integrazione, appositamente definito per l'interscambio di tali informazioni in OpenSPCoop; in questo caso le informazioni saranno rappresentate in accordo all'xsd dell'header OpenSPCoop di integrazione disponibile alla URL *http://www.openspcoop.org/openspcoop/doc/1.0/integrazione.xsd*
- come informazioni interne ad un'header SOAP definito tramite lo standard *WS-Addressing*; tali informazioni vengono mappate nei seguenti elementi del protocollo WS-Addressing (l'elenco fornisce il nome dell'header WSAddressing e il relativo valore):
  - *To*, http://<SPCoopDestinatario>.spcoop.it/servizi/<SPCoopServizio>
  - *From*, http://[<nomeServizioApplicativoFruitore>.]<SPCoopMittente>.spcoop.it
  - *Action*, http://<SPCoopDestinatario>.spcoop.it/servizi/<SPCoopServizio>/<SPCoopAzione>
  - *MessageId*
    - uuid:<SPCoopID> per un messaggio di risposta, lato PortaDelegata, ritornato al servizio applicativo fruitore o per un messaggio di richiesta, lato PortaApplicativa, diretto al servizio applicativo erogatore
    - uuid:<IDApplicativo> per un messaggio di richiesta, lato PortaDelegata, generato dal servizio applicativo fruitore (Utilizzato per correlazione applicativa)
  - *RelatesTo*, uuid:<SPCoopRiferimentoMessaggio>

Nel seguito di questa sezione sarà mostrato un esempio di codice per la correlazione di due richieste di una transazione asincrona asimmetrica, utilizzando ognuna delle diverse modalità di integrazione appena discusse.

## 5.1 Transazione Asincrona tramite il servizio di IntegrationManager

In questa sezione mostriamo un esempio d'uso dell'Integration Manager per la correlazione delle richieste asincrone. In particolare le informazioni di correlazione vengono gestite tramite i metodi `getID` e `setRiferimentoMessaggio` del servizio di IntegrationManager.

```
SPCoopMessage msg1 = new SPCoopMessage();
msg1.setMessage(soapMessageRichiesta);

// invio richiesta asincrona e prelievo dell'id di correlazione
SPCoopMessage msg1Response = port.invocaPortaDelegata(portaDelegataRichiesta,msg1);
idEgovRichiesta = msg1Response.getID();

// creazione messaggio di richiesta stato
SPCoopMessage msg2 = new SPCoopMessage();
msg2.setMessage(soapMessageRichiestaStato);

// settaggio dell'id di correlazione con la richiesta ed invio richiesta stato
SPCoopHeaderInfo spcoopHeaderInfo2 = new SPCoopHeaderInfo();
spcoopHeaderInfo2.setRiferimentoMessaggio(idEgovRichiesta);
msg2.setSpcoopHeaderInfo(spcoopHeaderInfo2);
SPCoopMessage msg2Response = port.invocaPortaDelegata(portaDelegataRichiestaStato, msg2);
```

## 5.2 Transazione Asincrona tramite header di Trasporto

In questa sezione mostriamo un esempio d'uso degli header di trasporto nella modalità trasparente. In particolare le informazioni di correlazione vengono gestite tramite gli header `SPCoopID` ed `SPCoopRiferimentoMessaggio`.

```
URL url = new URL(UrlPortaDelegataRichiesta);
HttpURLConnection httpCon1 = (HttpURLConnection)url.openConnection();
...
OutputStream out = httpCon1.getOutputStream();
out.write(soapMessageRichiesta);
out.close();
...
String idEgov = httpCon1.getHeaderFields().get("SPCoopID");
...
url = new URL(UrlPortaDelegataRichiestaStato);
HttpURLConnection httpCon2 = (HttpURLConnection)url.openConnection();
...
httpCon2.setRequestProperty("SPCoopRiferimentoMessaggio", idEgov);
OutputStream out = httpCon2.getOutputStream();
out.write(soapMessageRichiestaStato);
out.close();
...
```

## 5.3 Transazione Asincrona tramite QUERY_STRING

In questa sezione mostriamo un esempio d'uso della QUERY_STRING nella modalità trasparente. In particolare le informazioni di correlazione vengono gestite tramite i parametri `SPCoopID` ed `SPCoopRiferimentoMessaggio` delle URL invocate.

```
URL url = new URL(UrlPortaDelegataRichiesta);
HttpURLConnection httpCon1 = (HttpURLConnection)url.openConnection();
...
OutputStream out = httpCon1.getOutputStream();
out.write(soapMessageRichiesta);
out.close();
...
```

```
String idEgov = httpCon1.getHeaderFields().get("SPCoopID");
...
url = new URL( UrlPortaDelegataRichiestaStato + "?SPCoopRiferimentoMessaggio=" + idEgov );

HttpURLConnection httpCon2 = (HttpURLConnection)url.openConnection();
...
OutputStream out = httpCon2.getOutputStream();
out.write(soapMessageRichiestaStato);
out.close();
...
```

### 5.4 Transazione Asincrona tramite Header SOAP

In questa sezione mostriamo un esempio d'uso dell'header SOAP di integrazione di OpenSPCoop nella modalità trasparente. In particolare le informazioni di correlazione vengono gestite tramite gli attributi `SPCoopID` ed `SPCoopRiferimentoMessaggio` dell'header SOAP.

```
URL url = new URL(UrlPortaDelegataRichiesta);
HttpURLConnection httpCon1 = (HttpURLConnection)url.openConnection();
...
OutputStream out1 = httpCon1.getOutputStream();
out1.write(soapMessageRichiesta);
out1.close();
...
InputStream responseStream = httpCon1.getInputStream();
Message response = new Message(responseStream);
responseStream.close();
...
SOAPHeaderElement headerIntegrazione = null;
java.util.Iterator<?> iter =
response.getSOAPHeader().getChildElements();
while( iter.hasNext()  ) {
  SOAPHeaderElement headerElement = (SOAPHeaderElement) iter.next();
  //Controllo Actor
  if(headerElement.getActor().equals("http://www.openspcoop.org/integrazione") ) {
    headerIntegrazione = headerElement;
    break;
  }
}

String idEGov = headerElement.getAttribute("SPCoopID");
...

// Preparazione dell'Header SOAP OpenSPCoop

Name name = new PrefixedQName(
"http://www.openspcoop.org/integrazione", "integrazione", "openspcoop");

SOAPFactory fac = SOAPFactory.newInstance();
SOAPHeaderElement header = (SOAPHeaderElement) fac.createElement(name);

header.setActor("http://www.openspcoop.org/integrazione");
header.addNamespaceDeclaration("SOAP_ENV","http://schemas.xmlsoap.org/soap/envelope/");

// Settaggio riferimentoMessaggio per la correlazione asincrona
header.setAttribute("SPCoopRiferimentoMessaggio", idEGov);

// Aggiunta header al messaggio Soap. Si assume che il Messaggio SOAP
// da spedire sia disponibile nella variabile msgRichiestaStato
```

```
msgRichiestaStato.getHeader().addChildElement(header);

URL urlConnection = new URL(...todo...);
HttpURLConnection httpConn =
(HttpURLConnection) urlConnection.openConnection();

OutputStream out = httpConn.getOutputStream();
axisMsgRichiestaStato.writeTo(out);
out.close();
...
```

## 5.5 Transazione Asincrona tramite Header WSAddressing

In questa sezione mostriamo un esempio d'uso dell'header WS-Addressing di integrazione di OpenSPCoop nella modalità trasparente. In particolare le informazioni di correlazione vengono gestite tramite gli attributi SPCoopID ed SPCoopRiferimentoMessaggio dell'header WSAddressing.

```
          URL url = new URL(UrlPortaDelegataRichiesta);
          HttpURLConnection httpCon1 = (HttpURLConnection)url.openConnection();
          ...
          OutputStream out1 = httpCon1.getOutputStream();
          out1.write(soapMessageRichiesta);
          out1.close();
          ...
          InputStream responseStream = httpCon1.getInputStream();
          Message response = new Message(responseStream);
          responseStream.close();
          ...
          SOAPHeaderElement headerIntegrazione = null;
          java.util.Iterator<?> iter =
          response.getSOAPHeader().examineAllHeaderElements();
          while (iter.hasNext()) {
            SOAPHeaderElement headerElement = (SOAPHeaderElement) iter.next();

            //Controllo Actor
            if ("MessageID".equals(elem.getLocalName()) &&
              "http://www.w3.org/2005/08/addressing".equals(elem.getNamespaceURI()) &&
              "http://www.openspcoop.org/integrazione".equals(elem.getActor())) {
              headerIntegrazione = headerElement;
              break;
            }
          }

          String idEGov_UUID = headerIntegrazione.getValue();
          ...

          // Preparazione dell'Header SOAP OpenSPCoop

          Name nameRelatesTo = new PrefixedQName("http://www.w3.org/2005/08/addressing"," ↩
              RelatesTo","wsa");

          SOAPHeaderElement header =
          new org.apache.axis.message.SOAPHeaderElement(nameRelatesTo);

          header.setActor("http://www.openspcoop.org/integrazione");
          header.setMustUnderstand(false);
          header.addNamespaceDeclaration("SOAP_ENV",
          "http://schemas.xmlsoap.org/soap/envelope/");

          // Settaggio riferimentoMessaggio per la correlazione asincrona
```

```
          header.setValue(idEGov_UUID);

          // Aggiunta header al messaggio Soap. Si assume che il Messaggio Axis
          // da spedire sia disponibile nella variabile axisMsgRichiestaStato
          axisMsgRichiestaStato.getSOAPEnvelope().getHeader().addChildElement(header);

          URL urlConnection = new URL(...URL-PD...);
          HttpURLConnection httpConn =
          (HttpURLConnection) urlConnection.openConnection();

          OutputStream out = httpConn.getOutputStream();
          axisMsgRichiestaStato.writeTo(out);
          out.close();
          ...
```

# 6 La gestione degli errori nell'Interazione con la Porta di Dominio

In funzione del fatto che si usi la modalità di invocazione trasparente o i servizi dell'IntegrationManager, cambia il modo in cui le condizioni di errore vengono restituite al servizio applicativo.

Nel caso si usino i servizi dell'IntegrationManager le condizioni di errore saranno restituite all'interno di una eccezione gestita dal servizio IntegrationManager, il cui formato è rappresentato di seguito in linguaggio java.

```
public class SPCoopException {
  public java.lang.String getCodiceEccezione();
  public void setCodiceEccezione(java.lang.String codiceEccezione);

  public java.lang.String getDescrizioneEccezione();
  public void setDescrizioneEccezione(java.lang.String descrizioneEccezione);

  public java.lang.String getIdentificativoFunzione();
  public void setIdentificativoFunzione(java.lang.String identificativoFunzione);

  public java.lang.String getIdentificativoPorta();
  public void setIdentificativoPorta(java.lang.String identificativoPorta);

  public java.lang.String getOraRegistrazione();
  public void setOraRegistrazione(java.lang.String oraRegistrazione);

  public java.lang.String getTipoEccezione();
  public void setTipoEccezione(java.lang.String tipoEccezione);
}
```

Nel caso in cui si utilizzi la modalità trasparente, sarà invece possibile, in caso di eccezione SOAPFault, testare il campo `FaultActor` per riconoscere e gestire i casi di errore dovuti all'interazione con la porta di dominio (valore OpenSPCoop) da quelli puramente applicativi. Nel frammento di codice seguente vediamo, a titolo di esempio, come gestire questa situazione nel caso specifico del linguaggio java usando Axis 1.4 come web-services engine.

```
try {
  HelloWSServiceLocator locator = new HelloWSServiceLocator();
  locator.setHelloWorldEndpointAddress("http://pdd/openspcoop/PD/GetDate");
  HelloWS port = locator.getHelloWorld();
  String msg = port.getDate();
}
catch (AxisFault e) {
  if ("OpenSPCoop".equals(e.getFaultActor())) {

    System.out.println("Ricevuto Messaggio di Errore Applicativo ["+e.getFaultCode()+"]:");
    System.out.println(e.getFaultString());
```

```
    // Estrazione del dettaglio dell'errore
    Element [] details = e.getFaultDetails();
    if (details!=null) {
      for (int i=0; i<details.length; i++) {
        Element detail = details[i];
        if ("MessaggioDiErroreApplicativo".equals(detail.getLocalName())) {

          // Details, contiene l'xml che forma il messaggio di errore applicativo
          // come definito nella specifica CNIPA (Sistema Pubblico di cooperazione: Porta ↩
              di Dominio v1.0 a pag. 41)
          String xml = XMLUtils.ElementToString(detail);
          System.out.println("Errore Applicativo CNIPA: "+xml);
        }
      }
    }
  } else {

    // Errore applicativo
    System.out.println("Ricevuto SOAPFault applicativo");
    System.out.println("Actor: "+e.getFaultActor());
    System.out.println("Code: "+e.getFaultCode());
    System.out.println("String: "+e.getFaultString());

  }
}
catch (Exception e) {
  System.out.println("ClientError: "+e.getMessage());
  e.printStackTrace();
}
```

Nel riquadro seguente mostriamo un esempio di output generato dal codice di esempio nel caso di errore inviato dalla Porta di Dominio a causa dell'invocazione di una Porta Delegata inesistente.

```
Ricevuto Messaggio di Errore Applicativo da OpenSPCoop [{http://openspcoop.org/errore} ↩
    OPENSPCOOP_ORG_401]:
La porta delegata invocata non esiste location[GetDatecdcdcdcccd] urlInvocazione[ ↩
    GetDatecdcdcdcccd]
Errore Applicativo CNIPA:
<eGov_IT_Ecc:MessaggioDiErroreApplicativo
  xmlns:eGov_IT_Ecc="http://www.cnipa.it/schemas/2003/eGovIT/Exception1_0/" xmlns:xsd="http ↩
      ://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <OraRegistrazione>2010-08-27T17:21:25.195</OraRegistrazione>
  <IdentificativoPorta>OpenSPCoopSPCoopIT</IdentificativoPorta>
  <IdentificativoFunzione>RicezioneContenutiApplicativi_PD</IdentificativoFunzione>
  <Eccezione>
    <EccezioneProcessamento codiceEccezione="OPENSPCOOP_ORG_401" descrizioneEccezione="La ↩
        porta
      delegata invocata non esiste location[GetDatecdcdcdcccd] urlInvocazione[ ↩
          GetDatecdcdcdcccd]"/>
  </Eccezione>
</eGov_IT_Ecc:MessaggioDiErroreApplicativo>
```

Sulla gestione degli errori sono possibili personalizzazioni della Porta di Dominio, come descritto in Integrazione dei Servizi Applicativi e/o nel file *openspcoop.properties* (vedi Guida di Installazione).

Come evidenziato nell'esempio, il particolare codice di eccezione generato dalla Porta di Dominio potrà essere ottenuto tramite il campo `FaultCode` contenuto nel messaggio di SOAP Fault. Gli errori generati dalla Porta di Dominio rientrano in due tipologie. Gli *errori di cooperazione* riguardano i problemi di comunicazione tra Porte di Dominio e sono quindi normati dalla specifica SPCoop. Vediamo nella tabella seguente gli errori di questo tipo.

```
FaultCode Dettaglio di Errore
-----------------------------------------------------------------------------------------
EGOV_IT_001     Formato Busta non corretto
-----------------------------------------------------------------------------------------
EGOV_IT_002     Formato Intestazione non corretto
-----------------------------------------------------------------------------------------
EGOV_IT_003     Formato Corpo non corretto
-----------------------------------------------------------------------------------------
EGOV_IT_100     Errore nel contenuto dell'Intestazione della busta
-----------------------------------------------------------------------------------------
EGOV_IT_101     Identificativo della parte Mittente sconosciuto
-----------------------------------------------------------------------------------------
EGOV_IT_102     Identificativo della parte Destinatario sconosciuto
-----------------------------------------------------------------------------------------
EGOV_IT_103     Profilo di Collaborazione non valido
-----------------------------------------------------------------------------------------
EGOV_IT_104     Identificativo di Collaborazione non valido
-----------------------------------------------------------------------------------------
EGOV_IT_105     Servizio sconosciuto
-----------------------------------------------------------------------------------------
EGOV_IT_106     Azione sconosciuta
-----------------------------------------------------------------------------------------
EGOV_IT_107     Identificatore messaggio non definito
-----------------------------------------------------------------------------------------
EGOV_IT_108     OraRegistrazione messaggio non valida
-----------------------------------------------------------------------------------------
EGOV_IT_109     Riferimento messaggio non definito
-----------------------------------------------------------------------------------------
EGOV_IT_110     Identificatore messaggio non valido
-----------------------------------------------------------------------------------------
EGOV_IT_111     RiferimentoMessaggio non valido
-----------------------------------------------------------------------------------------
EGOV_IT_112     Scadenza messaggio non valida
-----------------------------------------------------------------------------------------
EGOV_IT_113     Profilo di Trasmissione non valido
-----------------------------------------------------------------------------------------
EGOV_IT_114     Sequenza non valida
-----------------------------------------------------------------------------------------
EGOV_IT_115     ListaRiscontri non valida
-----------------------------------------------------------------------------------------
EGOV_IT_116     ListaTrasmissioni non valida
-----------------------------------------------------------------------------------------
EGOV_IT_117     Firma non valida
-----------------------------------------------------------------------------------------
EGOV_IT_118     Riferimento all'allegato non valido
-----------------------------------------------------------------------------------------
EGOV_IT_119     Allegato non presente
-----------------------------------------------------------------------------------------
EGOV_IT_120     Allegato non definito
-----------------------------------------------------------------------------------------
EGOV_IT_200     Errore nelle politiche di sicurezza del Servizio
-----------------------------------------------------------------------------------------
EGOV_IT_201     Mittente non Autorizzato
-----------------------------------------------------------------------------------------
EGOV_IT_202     Firma XML non valida
-----------------------------------------------------------------------------------------
EGOV_IT_203     Firma PKCS#7 non valida
-----------------------------------------------------------------------------------------
EGOV_IT_300     Errore nel processamento del messaggio
-----------------------------------------------------------------------------------------
EGOV_IT_301     Messaggio scaduto
```

| | |
|---|---|
| EGOV_IT_400 | Errore nella gestione della trasparenza temporale |
| EGOV_IT_401 | Trasparenza temporale non supportata |
| EGOV_IT_402 | Trasparenza temporale non gestibile: attributi del Profilo di trasmissione non validi. |

Gli errori di integrazione o relativi ai servizi di base della Porta di Dominio sono specifici di OpenSPCoop e sono elencati nella seguente tabella.

| FaultCode | Dettaglio di Errore |
|---|---|
| OPENSPCOOP_ORG_401 | La porta delegata invocata non esiste |
| OPENSPCOOP_ORG_402 | Autenticazione Fallita |
| OPENSPCOOP_ORG_403 | Pattern Ricerca Porta Delegata Non Valido |
| OPENSPCOOP_ORG_404 | Autorizzazione Fallita |
| OPENSPCOOP_ORG_405 | Servizio SPCoop abbinato alla Porta Delegata Inesistente |
| OPENSPCOOP_ORG_406 | Nessun Messaggio disponibile per il Servizio Applicativo (Integration Manager) |
| OPENSPCOOP_ORG_407 | Messaggio Richiesto Inesistente (Integration Manager) |
| OPENSPCOOP_ORG_408 | Servizio Correlato associato ad un Servizio Asincrono non esistente |
| OPENSPCOOP_ORG_409 | Risposta/RichiestaStato asincrona non correlata ad una precedente richiesta |
| OPENSPCOOP_ORG_410 | Autenticazione richiesta per l'invocazione della Porta Delegata |
| OPENSPCOOP_ORG_411 | Elemento Risposta Asincrona richiesto per l'invocazione della Porta Delegata |
| OPENSPCOOP_ORG_412 | Porta Delegata invocabile solo per riferimento |
| OPENSPCOOP_ORG_413 | Porta Delegata non invocabile per riferimento |
| OPENSPCOOP_ORG_414 | Consegna in ordine utilizzabile sono con profilo Oneway |
| OPENSPCOOP_ORG_415 | Consegna in ordine non utilizzabile per mancanza di dati necessari |
| OPENSPCOOP_ORG_416 | Correlazione Applicativa non riuscita |
| OPENSPCOOP_ORG_417 | Impossibile istanziare un validatore: XSD non valido o mancante |
| OPENSPCOOP_ORG_418 | Validazione del messaggio di richiesta fallita |
| OPENSPCOOP_ORG_419 | Validazione del messaggio di risposta fallita |
| OPENSPCOOP_ORG_420 | Busta E-Gov presente nel messaggio di richiesta |
| OPENSPCOOP_ORG_421 | Il messaggio di richiesta non rispetta il formato SOAP (Integration Manager) |
| OPENSPCOOP_ORG_422 | Imbustamento E-Gov non riuscito per il messaggio di richiesta |

```
OPENSPCOOP_ORG_423   Servizio SPCoop invocato con azione non corretta
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_424   Funzione "Allega Body" non riuscita sul messaggio di richiesta
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_425   Funzione "Scarta Body" non riuscita sul messaggio di richiesta
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_426   Errore di processamento SOAP del messaggio di richiesta
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_427   Impossibile processare header SOAP in messaggio con opzione  ↩
    mustUnderstand
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_428   Autorizzazione basata sul contenuto fallita
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_429   L'header HTTP riporta un Content-Type non previsto in SOAP 1.1
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_430   Envelope con Namespace non previsto in SOAP 1.1
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_500
OPENSPCOOP_ORG_5XX   Porta di Dominio Temporaneamente non Disponibile (Errori Interni del  ↩
    Server)
-------------------------------------------------------------------------------------------
OPENSPCOOP_ORG_500
OPENSPCOOP_ORG_5XX   Porta di Dominio Temporaneamente non Disponibile (Errori Interni del  ↩
    Server)
-------------------------------------------------------------------------------------------
```

I servizi applicativi implementati dovranno quindi gestire tutti gli errori generati dalla porta, trattenendo i messaggi che hanno generato errore per una successiva spedizione.