Guida alla programmazione e integrazione di servizi in OpenSPCoop2

Indice

Elenco delle figure

1 Introduzione

La Porta di Dominio OpenSPCoop2 ha il compito di mediare le comunicazioni tra le applicazioni al fine di garantire il rispetto di un determinato protocollo di cooperazione applicativa. Le applicazioni che necessitano di effettuare comunicazioni secondo il protocollo di cooperazione devono interfacciarsi con la Porta di Dominio.

Le Porte di Dominio comunicano tra loro scambiandosi messaggi secondo il formato previsto dal protocollo di cooperazione adottato. In genere i messaggi scambiati tra le PdD contengono un header specifico che riporta i dati di protocollo. Tuttavia il formato di tali messaggi non è gestito nativamente dalle applicazioni, pertanto la PdD deve anche occuparsi di convertire i messaggi ricevuti dalle applicazioni.

Facendo riferimento a questa problematica, le funzionalità della Porta di Dominio possono essere suddivise in due gruppi:

- *Funzionalità di Cooperazione*: tutto ciò che riguarda la comunicazione tra le Porte di Dominio nel rispetto del protocollo di cooperazione;
- *Funzionalità di Integrazione*: tutto ciò che riguarda la comunicazione tra la Porta di Dominio e i Servizi Applicativi.

In questo documento ci occupiamo di descrivere i meccanismi, messi a disposizione da OpenSPCoop2, per rendere possibile l'integrazione dei Servizi Applicativi alla Porta di Dominio.

I componenti della PdD che gestiscono l'integrazione con i servizi applicativi vengono differenziati in due moduli distinti:

- *Porta Delegata*: utilizzata come proxy dai servizi applicativi che vogliono fruire di un servizio. La configurazione di una porta delegata può essere effettuata tramite la PddConsole sulla base di quanto descritto nel *Manuale di Gestione*.
- *Porta Applicativa*: gestisce la consegna dei contenuti applicativi presenti nei messaggi ricevuti dalla PdD. Dopo aver estratto il messaggio applicativo lo consegna al servizio applicativo destinatario secondo le impostazioni di integrazione che sono state fornite. Anche la porta applicativa può essere configurata tramite la PddConsole in accordo a quanto riportato sul *Manuale di Gestione*.

La configurazione delle porte delegate e applicative viene effettuata dal gestore della PdD in accordo a quelle che sono le modalità di integrazione che sono state implementate sui servizi applicativi. Le modalità di integrazione previste sono di due tipi:

- *Proxy Transparent*: prevede che i servizi utilizzino le interfacce originarie del servizio per comunicare. In questo scenario il ruolo della PdD è per l'appunto quello di un proxy trasparente. Questa modalità di integrazione è molto più semplice ed elegante da utilizzare, e se ne suggerisce quindi l'uso tutte le volte che questo sia possibile.
- *Integration Manager*: prevede che i servizi utilizzino le interfacce di un servizio appositamente esposto dalla PdD che permette di gestire l'integrazione mascherando la presenza del servizio originario.

Nel seguito di questa sezione descriviamo le modalità di utilizzo dei due approcci.

2 Modalità Proxy Transparent

La modalità di integrazione Proxy Transparent prevede che il servizio applicativo utilizzi (in caso di porta delegata) o esponga (in caso di porta applicativa) l'interfaccia originaria del servizio così come definita nel descrittore contenuto nell'accordo di servizio.

In tal caso la Porta di Dominio agisce come un proxy SOAP trasparente con il compito di gestire la creazione e il processamento dell'intestazione di protocollo nei messaggi in transito. In questa modalità, gli applicativi potranno continuare ad operare esattamente come se stessero interagendo direttamente con il servizio applicativo destinatario.

L'invocazione della porta delegata in modalità Proxy Tansparent può essere realizzata tramite gli strumenti del linguaggio di programmazione nativo del servizio applicativo, utilizzando ad esempio gli stub creati tramite il proprio ambiente di sviluppo Web Services (ad esempio wsdl2java in CXF), facendo riferimento direttamente al WSDL del servizio. In questo caso si deve utilizzare l'accorgimento di modificare la URL dell'endpoint del servizio con 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();

ComunicazioneVariazione_Type notifica = new 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, come nel caso seguente.

```
//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();
```

Nota
La URL della Porta Delegata deve essere ricavata seguendo le regole descritte nel Manuale di Gestione alla sezione *3.1 - Formato delle URL di invocazione di una Porta Delegata*

3 Modalità Integration Manager

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.

L'utilizzo della modalità *Integration Manager* in fruizione significa che il client utilizza l'interfaccia di un servizio esposto dalla PdD (al posto dell'interfaccia del servizio da invocare) per invocare una porta delegata.

L'utilizzo della modalità *Integration Manager* in erogazione significa che il servizio applicativo erogatore utilizza l'interfaccia di un servizio esposto dalla PdD per ricevere i messaggi di richiesta pervenuti. Si noti che in questo caso non è il servizio applicativo ad essere contattato dalla PdD per la consegna dei messaggi.

Alla luce di quanto premesso l'Integration Manager della PdD espone due web services:

- *PD*: web service per l'invocazione di una porta delegata. Il descrittore WSDL di questo servizio è disponibile alla URL http://www.openspcoop.org/openspcoop_v3/doc/IM-PD.wsdl
- *MessageBox*: web service per la ricezione dei messaggi presenti nella MessageBox. Il descrittore WSDL di questo servizio è disponibile alla URL http://www.openspcoop.org/openspcoop_v3/doc/IM-MessageBox.wsdl

Il riquadro seguente mostra, a titolo descrittivo, l'interfaccia esposta dal Web Service *PD*, espressa in linguaggio Java:

```
interface PD {

    // invoca una porta delegata correlando il messaggio inviato ad una richiesta  ↩
        precedente (profili asincroni)
    IntegrationManagerMessage invocaPortaDelegataPerRiferimento(
       String portaDelegata,
       IntegrationManagerMessage msg,
       String riferimentoMessaggio
    ) throws IntegrationManagerException_Exception;

    // invoca una porta delegata
    IntegrationManagerMessage invocaPortaDelegata(
       String portaDelegata,
       IntegrationManagerMessage msg
    ) throws IntegrationManagerException_Exception;

    // invoca una porta delegata nel caso di una richiesta correlata per il profilo  ↩
        asincrono asimmetrico
    IntegrationManagerMessage sendRichiestaStatoAsincronaAsimmetrica(
       String portaDelegata,
       IntegrationManagerMessage msg
    ) throws IntegrationManagerException_Exception;

    // invoca una porta delegata nel caso di una richiesta correlata per il profilo  ↩
        asincrono simmetrico
    IntegrationManagerMessage sendRispostaAsincronaSimmetrica(
       String portaDelegata,
       IntegrationManagerMessage msg
    ) throws IntegrationManagerException_Exception;
}
```

Il riquadro seguente mostra, a titolo descrittivo, l'interfaccia esposta dal Web Service *MessageBox*, espressa in linguaggio Java:

```
interface MessageBox {

    // restituisce gli identificativi dei messaggi ricevuti per un dato servizio/azione
    List<String> getAllMessagesIdByService(
        String tipoServizio,
        String servizio,
        String azione
    ) throws IntegrationManagerException_Exception;

    // restituisce il messaggio correlato a quello identificato dal parametro fornito
    IntegrationManagerMessage getMessageByReference(
        String riferimentoMsg
    ) throws IntegrationManagerException_Exception;

    // elimina tutti i messaggi presenti nella Message Box
    void deleteAllMessages() throws IntegrationManagerException_Exception;

    // restituisce gli identificativi di tutti i messaggi presenti nella Message Box
    List<String> getAllMessagesId() throws IntegrationManagerException_Exception;

    // elimina il messaggio correlato a quello identificato dal parametro fornito
```

```
    void deleteMessageByReference(
        String riferimentoMsg
    ) throws IntegrationManagerException_Exception;

    // restituisce il messaggio identificato dal parametro fornito
    IntegrationManagerMessage getMessage(
        String idMessaggio
    ) throws IntegrationManagerException_Exception;

    // restituisce un numero pari al valore "counter" di identificativi a partire dal  ↩
       numero "offset" corrispondenti a messaggi
    // ricevuti per il servizio/azione specificati come parametro
    public List<String> getMessagesIdArrayByService(
        int offset,
        int counter,
        String tipoServizio,
        String servizio,
        String azione
    ) throws IntegrationManagerException_Exception;

    // restituisce i primi "counter" identificativi corrispondenti a messaggi ricevuti
    List<String> getNextMessagesId(
        int counter
    ) throws IntegrationManagerException_Exception;

    // restituisce i primi "counter" identificativi corrispondenti a messaggi ricevuti per  ↩
       il servizio/azione specificati come parametro
    List<String> getNextMessagesIdByService(
        int counter,
        String tipoServizio,
        String servizio,
        String azione
    ) throws IntegrationManagerException_Exception;

    // elimina il messaggio identificato dal parametro fornito
    void deleteMessage(
        String idMessaggio
    ) throws IntegrationManagerException_Exception;

    // restituisce un numero pari al valore "counter" a partire da "offset" di messaggi  ↩
       ricevuti
    List<String> getMessagesIdArray(
        int offset,
        int counter
    ) throws IntegrationManagerException_Exception;
}
```

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

```
public class IntegrationManagerMessage {

    // metodi get/set per l'identificativo di correlazione applicativa
    public String getIDApplicativo()
    public void setIDApplicativo(String value)

    // metodi per indicare se il messaggio deve essere imbustato SOAP dalla PdD
    public boolean isImbustamento()
    public void setImbustamento(boolean value)

    // metodi get/set per impostare o recuperare il contenuto del messaggio come byte array
```

```
    public byte[] getMessage()
    public void setMessage(byte[] value)

    // metodi get/set per impostare o recuperare il nome del servizio applicativo
    public String getServizioApplicativo()
    public void setServizioApplicativo(String value)

    // metodi get/set per impostare o recuperare i dati dell'header di integrazione (dati ↩
        di protocollo)
    public ProtocolHeaderInfo getProtocolHeaderInfo()
    public void setProtocolHeaderInfo(ProtocolHeaderInfo value)

}
```

Come si può notare nella struttura del messaggio è presente un'entità annidata che comprende i dati di protocollo che costituiscono l'header di integrazione e quindi le informazioni scambiate tra la PdD e il servizio applicativo. Tali dati sono contenuti nell'entitè *ProtocolHeaderInfo*.

La struttura dell'entità *ProtocolHeaderInfo* è descritta nel riquadro seguente, a titolo descrittivo, in linguaggio java.

```
public class ProtocolHeaderInfo {

    public String getID()
    public void setID(String value)

    public String getAzione()
    public void setAzione(String value)

    public String getDestinatario()
    public void setDestinatario(String value)

    public String getIdCollaborazione()
    public void setIdCollaborazione(String value)

    public String getMittente()
    public void setMittente(String value)

    public String getRiferimentoMessaggio()
    public void setRiferimentoMessaggio(String value)

    public String getServizio()
    public void setServizio(String value)

    public String getTipoDestinatario()
    public void setTipoDestinatario(String value)

    public String getTipoMittente()
    public void setTipoMittente(String value)

    public String getTipoServizio()
    public void setTipoServizio(String value)
}
```

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 del servizio *PD* generati con il tool wsdl2java.

```
PDService im = new PDService(new URL("http://localhost:8080/openspcoop2/IntegrationManager/ ↩
    PD?wsdl"),
```

```
                                                                    new QName("http://services.pdd ↩
                                                                        .openspcoop.org", " ↩
                                                                        PDService"));
PD imPort = im.getPD();

java.lang.String portaDelegata = "ComunicazioneVariazione_PD";

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

Nota
Per la costruzione della URL di invocazione della porta delegata tramite il servizio *IntegrationManager* si utilizza la seguente sintassi:

```
http://#pdd-hostname#:#pdd-port#/openspcoop2/#protocollo#/IntegrationManager/PD
```

I placeholder delimitati dal carattere '#' dovranno essere sostituiti con i valori specifici del contesto. Se si fa riferimento ad un protocollo supportato nativamente dalla PdD, il placeholder #protocollo# può assumere uno dei seguenti valori:

- spcoop. Il protocollo di cooperazione applicativa definito da AgID per la PA italiana.
- sdi. Il protocollo per la fatturazione elettronica definito da AgID per la PA italiana.
- proxy. Il protocollo trasparente.

Se si omette la stringa del protocollo, la url farà riferimento al protocollo SPCoop.

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 Integration Manager.

```
// Inizializzazione punto di accesso al servizio
MessageBoxService imService = new MessageBoxService(new URL("http://localhost:8080/ ↩
    openspcoop2/IntegrationManager/MessageBox?wsdl"),
```

```
                                                                    new QName("http://services.pdd ↩
                                                                        .openspcoop.org", " ↩
                                                                        MessageBoxService"));
MessageBox imPort = imService.getMessageBox();

// Impostazione endpoint e credenziali del servizio
((BindingProvider)imPort).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY ↩
    ,  "http://localhost:8080/openspcoop2/IntegrationManager/MessageBox");
((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
List<String> ids =  imPort.getAllMessagesId();

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

  IntegrationManagerMessage msg = imPort.getMessage(ids.get(i));

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

  // elimina il messaggio processato
  imPort.deleteMessage(ids.get(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.

Nota

Per la costruzione della URL di invocazione per l'accesso alla MessageBox tramite il servizio *IntegrationManager* si utilizza la seguente sintassi:

```
http://#pdd-hostname#:#pdd-port#/openspcoop2/#protocollo#/IntegrationManager/ ↩
    MessageBox
```

I placeholder delimitati dal carattere '#' dovranno essere sostituiti con i valori specifici del contesto. Se si fa riferimento ad un protocollo supportato nativamente dalla PdD, il placeholder #protocollo# può assumere uno dei seguenti valori:

- spcoop. Il protocollo di cooperazione applicativa definito da AgID per la PA italiana.
- sdi. Il protocollo per la fatturazione elettronica definito da AgID per la PA italiana.
- proxy. Il protocollo trasparente.

Se si omette la stringa del protocollo, la url farà riferimento al protocollo SPCoop.

---

4 Aspetti specifici di Integrazione nei vari Profili di Collaborazione

In funzione del profilo di collaborazione ci sono delle differenze nelle modalità d'invocazione del servizio e nelle risposte restituite al richiedente. Nelle immagini che seguono 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 che il messaggio in transito ha un payload non vuoto
- Le *frecce tratteggiate* indicano che il messaggio in transito ha un payload vuoto (salvo eccezioni descritte in seguito)

4.1 Profilo OneWay (Stateful)

Nel caso del profilo Oneway non è previsto che il servizio invii un messaggio di risposta al chiamante. Se la PdD OpenSPCoop2 è stata configurata in modo da funzionare in modalità *Stateful*, in caso di invocazioni oneway prende in carico le richieste svincolando immediatamente il servizio applicativo chiamante. In questa situazione il servizio applicativo non potrà mai conoscere l'esito della consegna del messaggio inviato.

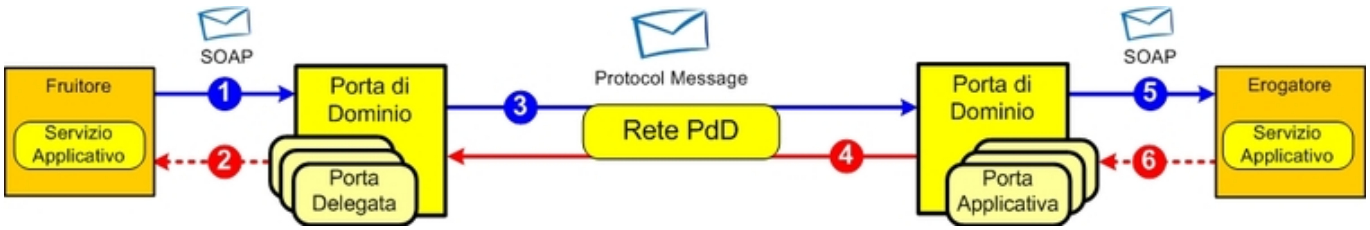

Figura 1: Profilo Oneway (Stateful)

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 SOAPFault.

4.2 Profilo OneWay (Stateless)

Se la PdD OpenSPCoop2 è stata configurata in modo da funzionare in modalità *Stateless*, in caso di invocazioni oneway tiene in attesa il servizio applicativo mittente fino alla ricezione della chiusura HTTP che arriva tramite una comunicazione con payload vuoto.

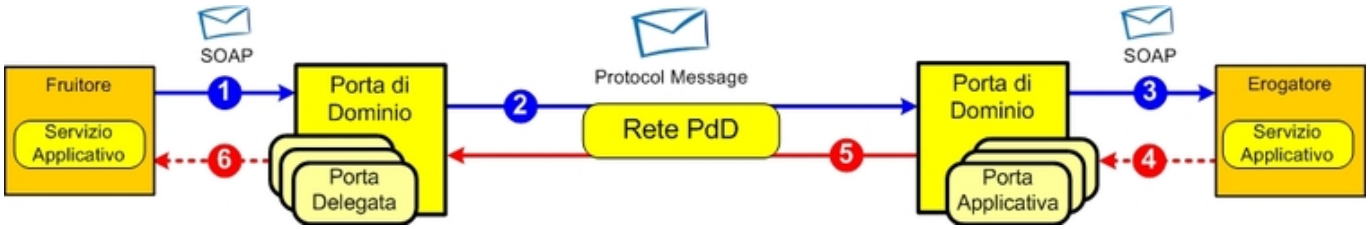

Figura 2: Profilo Oneway (Stateless)

OpenSPCoop2 ha un'opzione di configurazione che consente, anche nel caso del profilo Oneway, di ricevere messaggi di Fault nel caso in cui si verifichino errori durante la comunicazione.

4.3 Profilo Sincrono

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

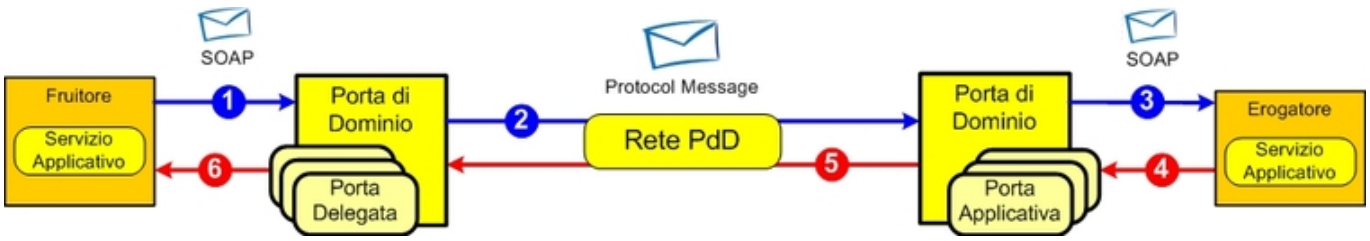

Figura 3: Profilo Sincrono

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

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

OpenSPCoop2 supporta le modalità di interazione sincrona e asincrona tra Servizi Applicativi e PdD in caso di profili asincroni. Il tipo di modalità utilizzata si può configurare tramite i parametri avanzati *ricevuta-asincrona-simmetrica* e *ricevuta-asincrona-asimmetrica* delle porte delegate e applicative.

4.4.1 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 del messaggio alla PdD del destinatario. Tale PdD replica con una "ricevuta" con SoapBody vuoto, provvedendo anche alla consegna al servizio applicativo destinatario.

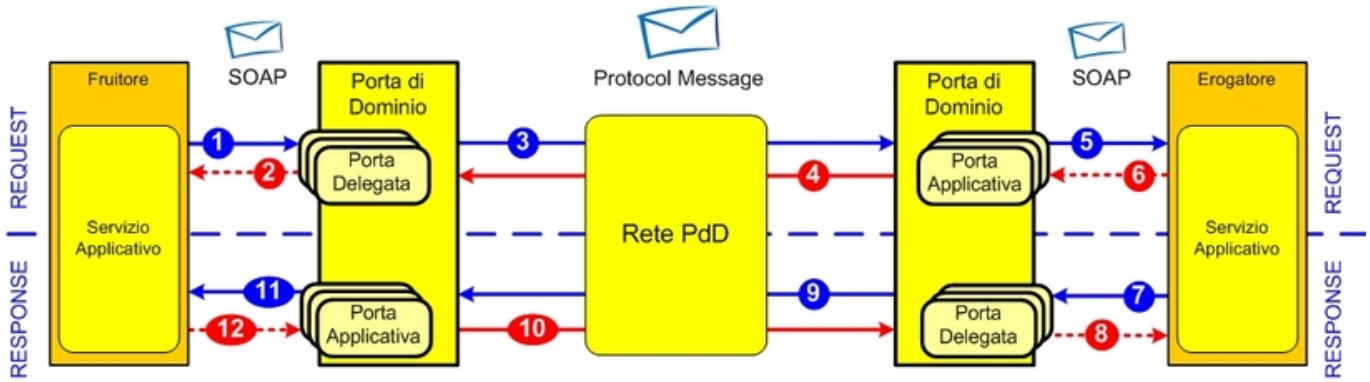

Figura 4: Profilo Asincrono Simmetrico con implementazione Asincrona

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

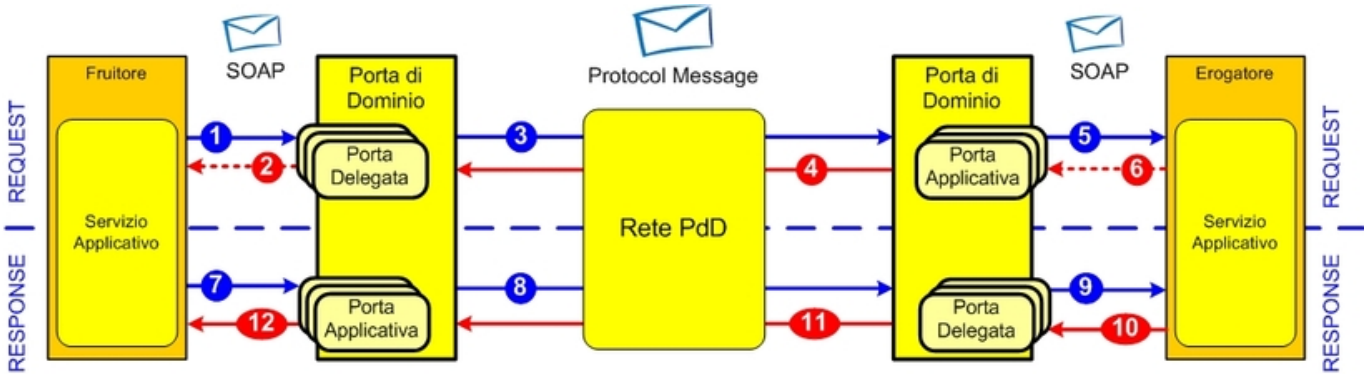

Figura 5: Profilo Asincrono Asimmetrico con implementazione Asincrona

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.

4.4.2 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.

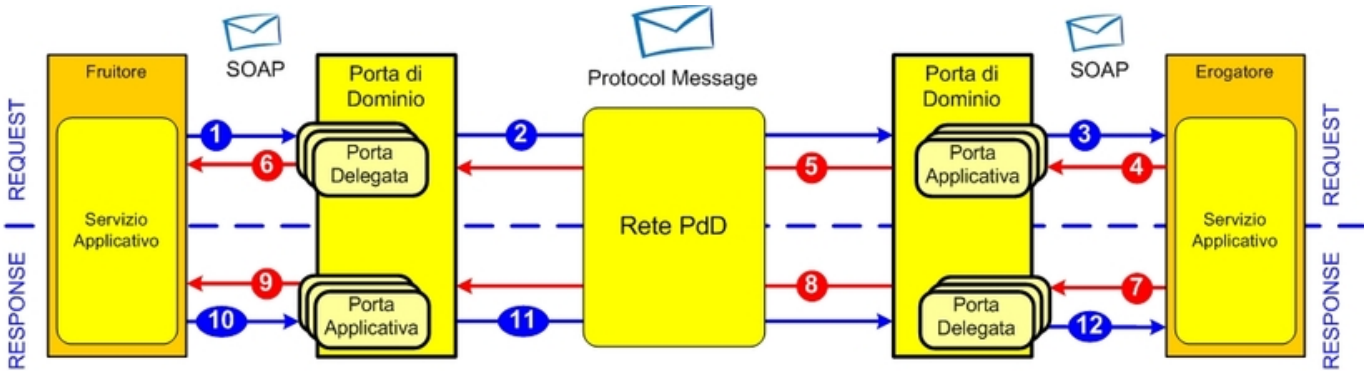

Figura 6: Profilo Asincrono Simmetrico con implementazione Sincrona

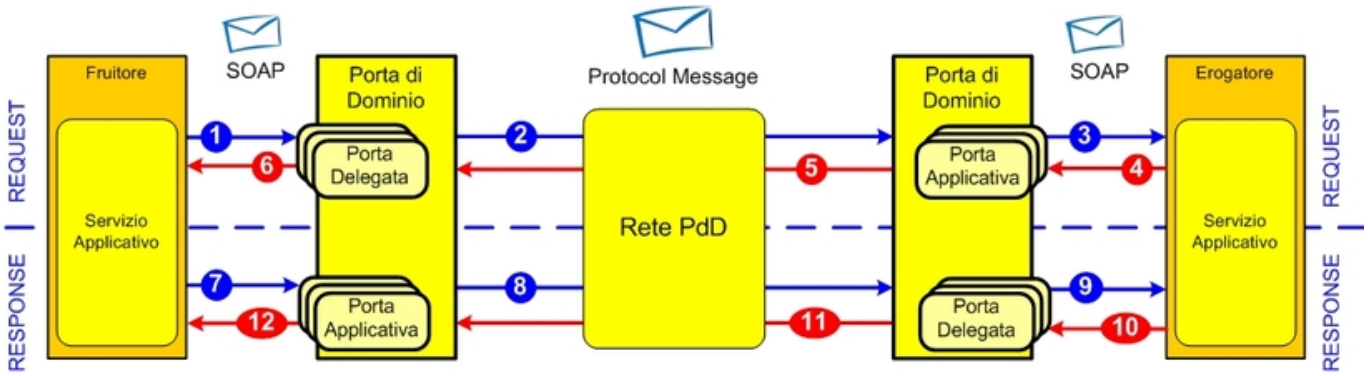

Figura 7: Profilo Asincrono Asimmetrico con implementazione Sincrona

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 dati presenti nell'intestazione di protocollo scambiati tra le Porte di Dominio. Una situazione del genere si verifica, ad esempio, quando diversi messaggi sono correlati tra loro, come nel caso dei profili di collaborazione Asincroni, sia simmetrico che asimmetrico.

Per questo, alcune delle informazioni parte dell'header di protocollo 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 il messaggio si riferisce e sono identificate tramite keyword che variano in base alla modalità con cui vengono accedute.

Se viene utilizzato l'header di trasporto (ad esempio Header HTTP) le keyword per riferire i dati di integrazione sono:

- X-OpenSPCoop2-IdMessaggio
- X-OpenSPCoop2-TipoMittente
- X-OpenSPCoop2-Mittente
- X-OpenSPCoop2-TipoDestinatario
- X-OpenSPCoop2-Destinatario
- X-OpenSPCoop2-TipoServizio
- X-OpenSPCoop2-Servizio
- X-OpenSPCoop2-Azione
- X-OpenSPCoop2-RiferimentoMessaggio
- X-OpenSPCoop2-Collaborazione
- X-OpenSPCoop2-IdApplicativo
- X-OpenSPCoop2-ServizioApplicativo

Se viene utilizzata la query string (e quindi la url) le keyword per riferire i dati di integrazione sono:

- OpenSPCoop2IdMessaggio
- OpenSPCoop2TipoMittente
- OpenSPCoop2Mittente
- OpenSPCoop2TipoDestinatario

- OpenSPCoop2Destinatario
- OpenSPCoop2TipoServizio
- OpenSPCoop2Servizio
- OpenSPCoop2Azione
- OpenSPCoop2RiferimentoMessaggio
- OpenSPCoop2Collaborazione
- OpenSPCoop2IdApplicativo
- OpenSPCoop2ServizioApplicativo

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

Nel caso di uso della modalità *Proxy Transparent*, 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 OpenSPCoop2; in questo caso le informazioni saranno rappresentate in accordo all'xsd dell'header OpenSPCoop2 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 WS-Addressing e il relativo valore):
  - *TO*, http://<tipoSoggettoErogatore>_<nomeSoggettoErogatore>.openspcoop2.org/servizi/<tipoServizio>_<nomeServizio>
  - *FROM*, http://[<nomeServizioApplicativoFruitore>.]<tipoSoggettoFruitore>_<nomeSoggettoFruitore>.openspcoop2.org
  - *Action*, http://<tipoSoggettoErogatore>_<nomeSoggettoErogatore>.openspcoop2.org/servizi/<tipoServizio>_<nomeServizio>/<no
  - *ID*
    - uuid:<IDBusta> 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:<IDBusta> equivalente al riferimento messaggio

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.

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

// invio richiesta asincrona e prelievo dell'id di correlazione
IntegrationManagerMessage msg1Response = port.invocaPortaDelegata(portaDelegataRichiesta, ↩
    msg1);
idRichiesta = msg1Response.getID();
```

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

// settaggio dell'id di correlazione con la richiesta ed invio richiesta stato
ProtocolHeaderInfo protocolHeaderInfo2 = new ProtocolHeaderInfo();
protocolHeaderInfo2.setRiferimentoMessaggio(idRichiesta);
msg2.setProtocolHeaderInfo(protocolHeaderInfo2);
IntegrationManagerMessage msg2Response = port.sendRichiestaStatoAsincronaAsimmetrica( ↩
    portaDelegataRichiestaStato, msg2);
```

5.2 Transazione Asincrona tramite header di Trasporto

In questa sezione mostriamo un esempio d'uso degli header di trasporto nella modalità Proxy Transparent. In particolare le informazioni di correlazione vengono gestite tramite gli header *X-OpenSPCoop2-IdMessaggio* e *X-OpenSPCoop2-RiferimentoMessaggio*.

```
URL url = new URL(UrlPortaDelegataRichiesta);
HttpURLConnection httpCon1 = (HttpURLConnection)url.openConnection();
...
OutputStream out = httpCon1.getOutputStream();
out.write(soapMessageRichiesta);
out.close();
...
String idMessaggio = httpCon1.getHeaderFields().get("X-OpenSPCoop2-IdMessaggio");
...
url = new URL(UrlPortaDelegataRichiestaStato);
HttpURLConnection httpCon2 = (HttpURLConnection)url.openConnection();
...
httpCon2.setRequestProperty("X-OpenSPCoop2-RiferimentoMessaggio", idMessaggio);
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 Proxy Transparent. In particolare le informazioni di correlazione vengono gestite tramite i parametri *OpenSPCoop2IdMessaggio* e *OpenSPCoop2RiferimentoMessaggio* delle URL invocate.

```
URL url = new URL(UrlPortaDelegataRichiesta);
HttpURLConnection httpCon1 = (HttpURLConnection)url.openConnection();
...
OutputStream out = httpCon1.getOutputStream();
out.write(soapMessageRichiesta);
out.close();
...
String idMessaggio = httpCon1.getHeaderFields().get("OpenSPCoop2IdMessaggio");
...
url = new URL( UrlPortaDelegataRichiestaStato + "?OpenSPCoop2RiferimentoMessaggio=" +  ↩
    idMessaggio );

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 OpenSPCoop2 nella modalità Proxy Transparent. In particolare le informazioni di correlazione vengono gestite tramite gli attributi *IdMessaggio* e *RiferimentoMessaggio* 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.openspcoop2.org/core/integrazione") ) {
    headerIntegrazione = headerElement;
    break;
  }
}

String idMessaggio = headerElement.getAttribute("IdMessaggio");
...

// Preparazione dell'Header SOAP OpenSPCoop2

Name name = new PrefixedQName(
"http://www.openspcoop2.org/core/integrazione", "integrazione", "openspcoop2");

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

header.setActor("http://www.openspcoop2.org/core/integrazione");
header.addNamespaceDeclaration("SOAP_ENV","http://schemas.xmlsoap.org/soap/envelope/");

// Settaggio riferimentoMessaggio per la correlazione asincrona
header.setAttribute("RiferimentoMessaggio", idMessaggio);

// 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();
msgRichiestaStato.writeTo(out);
out.close();
...
```

5.5 Transazione Asincrona tramite Header WS-Addressing

In questa sezione mostriamo un esempio d'uso dell'header WS-Addressing di integrazione di OpenSPCoop2 nella modalità Proxy Transparent. In particolare le informazioni di correlazione vengono gestite tramite gli attributi *ID* e *RelatesTo* dell'header WS-Addressing.

```
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 ("ID".equals(elem.getLocalName()) &&
    "http://www.w3.org/2005/08/addressing".equals(elem.getNamespaceURI()) &&
    "http://www.openspcoop2.org/core/integrazione".equals(elem.getActor())) {
    headerIntegrazione = headerElement;
    break;
  }
}

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

// Preparazione dell'Header SOAP OpenSPCoop2

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.openspcoop2.org/core/integrazione");
header.setMustUnderstand(false);
header.addNamespaceDeclaration("SOAP_ENV",
"http://schemas.xmlsoap.org/soap/envelope/");

// Settaggio riferimentoMessaggio per la correlazione asincrona
header.setValue(idMessaggio_UUID);

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

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

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

```
            ...
```

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

La gestione dei casi di errore nelle comunicazioni mediate da Porta di Dominio devono tenere conto di ulteriori situazioni che possono presentarsi rispetto alla situazione di dialogo diretto tra i servizi applicativi. Oltre agli errori conosciuti dai servizi applicativi, e quindi previsti nei descrittori del servizio, denominati *Errori Applicativi*, abbiamo le seguenti tipologie di errori:

- *Errori di Cooperazione*, i casi di errore sollevati dalla Porta di Dominio a seguito di problemi riscontrati nella comunicazione con un'altra Porta di Dominio e quindi legati al protocollo di cooperazione.
- *Errori di Integrazione*, i casi di errore sollevati dalla Porta di Dominio a seguito di problemi riscontrati nella comunicazione con i servizi applicativi, ad esempio per un fallimento nella fase di consegna di un contenuto applicativo.

OpenSPCoop2 consente la totale personalizzazione dei codici e relativi messaggi di errore per ciascun protocollo di cooperazione. Per dettagli sulle modalità di personalizzazione degli errori si consulti il Manuale SDK ed in particolare la documentazione del package *org.openspcoop2.protocol.sdk.config.ITraduttore*.

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

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

```
public class IntegrationManagerException {

    public String getOraRegistrazione()
    public void setOraRegistrazione(String value)

    public String getCodiceEccezione()
    public void setCodiceEccezione(String value)

    public String getDescrizioneEccezione()
    public void setDescrizioneEccezione(String value)

    public String getTipoEccezione()
    public void setTipoEccezione(String value)

    public String getIdentificativoPorta()
    public void setIdentificativoPorta(String value)

    public String getIdentificativoFunzione()
    public void setIdentificativoFunzione(String value)
}
```

Nel caso in cui si utilizzi la modalità Proxy Transparent, 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 "OpenSPCoop2") da quelli puramente applicativi e quindi generati dai servizi applicativi.

Nel frammento di codice seguente vediamo, a titolo di esempio, una possibile gestione dei casi di errore riferita al protocollo SPCoop, utilizzando il linguaggio java e Axis.

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

```
  if ("OpenSPCoop2".equals(e.getFaultActor())) {

    System.out.println("Ricevuto Messaggio di Errore di Cooperazione ["+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())) {
          String xml = XMLUtils.ElementToString(detail);
          System.out.println("Errore Applicativo: "+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 dell'esempio precedente 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:
<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>
```

Nel caso evidenziato dall'esempio, il particolare codice di eccezione generato dalla Porta di Dominio potrà essere ottenuto tramite il campo *FaultCode* contenuto nel messaggio di SOAP Fault.

Vediamo adesso due tabelle che riportano le tipologie di errore gestite da OpenSPCoop2 sia per gli errori di cooperazione che per quelli di integrazione. Si tenga presente che i codici di errore riportati rappresentano una codifica interna e quindi per conoscere il codice di errore ed il messaggio associato si dovrà fare riferimento a quanto previsto nel plugin del protocollo di cooperazione adottato.

6.1 Errori di Cooperazione

```
MITTENTE_SCONOSCIUTO
Il Mittente non risulta registrato nel Registro dei Servizi

MITTENTE_NON_VALIDO
Il Mittente presente nella busta non è valido

TIPO_MITTENTE_NON_VALIDO
Il Tipo del mittente presente nella busta non è valido

DESTINATARIO_SCONOSCIUTO
Il Destinatario non risulta registrato nel Registro dei Servizi

DESTINATARIO_NON_VALIDO
Il Destinatario presente nella busta non è valido

TIPO_DESTINATARIO_NON_VALIDO
Il Tipo del destinatario presente nella busta non è valido

SERVIZIO_SCONOSCIUTO
Il Servizio non risulta registrato nel Registro dei Servizi

SERVIZIO_NON_VALIDO
Il Servizio presente nella busta non è valido

TIPO_SERVIZIO_NON_VALIDO
Il Tipo del servizio presente nella busta non è valido

AZIONE_NON_VALIDA
L'azione presente nella busta non è valida

IDENTIFICATIVO_MESSAGGIO_GIA_PROCESSATO
La busta presenta un identificativo già processato in precedenza

RIFERIMENTO_MESSAGGIO_NON_PRESENTE
La busta non contiene un RiferimentoMessaggio alla precedente busta a cui è correlata ↩
    logicamente come richiede il profilo

RIFERIMENTO_MESSAGGIO_NON_VALIDO
IdentificativoBusta presente nel RiferimentoMessaggio non è valido

MESSAGGIO_SCADUTO
Messaggio scaduto

PROFILO_COLLABORAZIONE_SCONOSCIUTO
Busta con profilo di collaborazione sconosciuto

PROFILO_COLLABORAZIONE_NON_VALIDO
Busta con profilo di collaborazione non valido rispetto al tipo di cooperazione in corso

COLLABORAZIONE_NON_VALIDA
La collaborazione presente nella busta non è valida

COLLABORAZIONE_SCONOSCIUTA
La collaborazione presente nella busta non appartiene a nessuna sessione valida

PROFILO_TRASMISSIONE_CONFERMA_RICEZIONE_NON_PRESENTE
La busta non contiene una richiesta di 'conferma ricezione', nonostante il servizio ↩
    indicato richieda, tramite la definizione dell'accordo nel registro, una consegna ↩
    affidabile
```

```
CONSEGNA_IN_ORDINE_NON_GESTIBILE
La busta non contiene una richiesta di 'consegna in ordine', nonostante il servizio ↩
    indicato richieda, tramite la definizione dell'accordo nel registro, un ordinamento

CONSEGNA_IN_ORDINE_FUORI_SEQUENZA
Riscontrato numero di sequenza diverso da 1, in una busta capostipite di una sequenza

CONSEGNA_IN_ORDINE_TIPO_MITTENTE_NON_VALIDO
Il tipo di mittente non rispetta quello atteso nella gestione della collaborazione con ↩
    consegna in ordine

CONSEGNA_IN_ORDINE_MITTENTE_NON_VALIDO
Il nome del mittente non rispetta quello atteso nella gestione della collaborazione con ↩
    consegna in ordine

CONSEGNA_IN_ORDINE_TIPO_DESTINATARIO_NON_VALIDO
Il tipo di destinatario non rispetta quello atteso nella gestione della collaborazione con ↩
    consegna in ordine

CONSEGNA_IN_ORDINE_DESTINATARIO_NON_VALIDO
Il nome del destinatario non rispetta quello atteso nella gestione della collaborazione con ↩
     consegna in ordine

CONSEGNA_IN_ORDINE_TIPO_SERVIZIO_NON_VALIDO
Il tipo di servizio non rispetta quello atteso nella gestione della collaborazione con ↩
    consegna in ordine

CONSEGNA_IN_ORDINE_SERVIZIO_NON_VALIDO
Il nome del servizio non rispetta quello atteso nella gestione della collaborazione con ↩
    consegna in ordine

CONSEGNA_IN_ORDINE_AZIONE_NON_VALIDA
Il nome dell'azione non rispetta quello atteso nella gestione della collaborazione con ↩
    consegna in ordine

CONSEGNA_IN_ORDINE_COLLABORAZIONE_IN_BUSTA_NON_CAPOSTIPITE_SCONOSCIUTA
Busta non capostipite che richiede funzionalità di consegna in ordine presenta una ↩
    collaborazione non registrata per le funzioni di consegna in ordine

CONSEGNA_IN_ORDINE_NON_SUPPORTATA
Funzionalità di consegna in ordine non supportata

WSSECURITY
È occorso un errore durante la gestione della WSSecurity sul messaggio

AUTORIZZAZIONE_FALLITA
Il Mittente della busta non è autorizzato a fruire del servizio richiesto

ERRORE_GENERICO_PROCESSAMENTO_MESSAGGIO
È occorso un errore durante il processamento del messsaggio

ERRORE_GENERICO_PROTOCOLLO_NON_CORRETTO
È stato rilevata una violazione per quanto concerne le informazioni richieste del ↩
    protocollo in gestione
```

6.2 Errori di Integrazione

```
OPENSPCOOP2_ORG_401
Porta Delegata non esistente
```

```
OPENSPCOOP2_ORG_402
Autenticazione Fallita (sia per credenziali non fornite che per credenziali errate)

OPENSPCOOP2_ORG_403
Dati del servizio (Soggetto, Servizio, Azione) non trovati. Tali dati vengono ricercati ↩
    tramite meccanismi di identificazione dinamica indicati nella porta delegata (ad es. ↩
    contentBased, urlBased, ecc.)

OPENSPCOOP2_ORG_404
Autorizzazione Fallita

OPENSPCOOP2_ORG_405
Servizio abbinato alla PortaDelegata non esistente. Errore che avviene tipicamente quando l ↩
    'identificazione del servizio è dinamica

OPENSPCOOP2_ORG_406
Nessun Messaggio disponibile per il Servizio Applicativo (Accesso alla MessageBox via ↩
    Integration Manager)

OPENSPCOOP2_ORG_407
Messaggio richiesto non esistente (Accesso alla MessageBox via Integration Manager)

OPENSPCOOP2_ORG_408
l'invocazione del servizio indicato nella PortaDelegata genera un errore, poiché non è ↩
    presente nel Registro un servizio/azione correlata per la gestione della risposta ↩
    asincrona

OPENSPCOOP2_ORG_409
l'invocazione della Porta Delegata insieme all'identificativo di correlazione asincrona, ↩
    utilizzato per effettuare la seconda fase dei profili asincroni (richiestaStatoAA o ↩
    rispostaAS), non può essere effettuata dalla PdD poiché non risulta registrata nello ↩
    storico della PdD la prima fase asincrona di richiesta

OPENSPCOOP2_ORG_410
Il servizio indicato nella PortaDelegata risulta possedere un profilo asincrono simmetrico, ↩
     mentre la porta delegata stessa non prevede un meccanismo di autenticazione. Questa ↩
    configurazione è errata e viene segnalata con codice 410. L'identificazione di un ↩
    servizio applicativo fruitore è fondamentale nel profilo asincrono simmetrico per poi ↩
    consegnare la risposta

OPENSPCOOP2_ORG_411
La fruizione di un servizio asincrono simmetrico richiede che il servizio applicativo ↩
    fruitore possieda la configurazione per la spedizione della risposta asincrona. Se tale ↩
    configurazione non viene rilevata viene generato questo errore

OPENSPCOOP2_ORG_412
La porta delegata è stata invocata senza riferimento ad un messaggio precedentemente ↩
    registrato (funzionalità avanzata). Tale modalità di invocazione risulta abilitata sulla ↩
     porta delegata.

OPENSPCOOP2_ORG_413
Errore duale al precedente. La porta delegata è stata invocata per riferimento, ma tale ↩
    modalità di invocazione non risulta abilitata sulla porta delegata stessa

OPENSPCOOP2_ORG_414
La Porta Delegata utilizza un servizio che richiede la consegna in ordine con un profilo di ↩
     collaborazione non compatibile. Tale funzionalità è permessa solamente nel profilo ↩
    oneway

OPENSPCOOP2_ORG_415
```

```
La Porta Delegata utilizza un servizio che richiede la consegna in ordine. Il profilo di ↩
    collaborazione è correttamente oneway, ma non sono presenti altre caratteristiche ↩
    obbligatorie con questa funzionalità (es. confermaRicezione,filtroDuplicati, ↩
    collaborazione)

OPENSPCOOP2_ORG_416
Il processo di Correlazione Applicativa della Richiesta ha generato un errore

OPENSPCOOP2_ORG_417
Il motore di validazione dei contenuti applicativi non è riuscito a creare un ↩
    XSDSchemaCollection valido ai fini della validazione tramite gli schemi presenti nel ↩
    Registro dei Servizi

OPENSPCOOP2_ORG_418
Validazione Applicativa del Messaggio di Richiesta fallita

OPENSPCOOP2_ORG_419
Validazione Applicativa del Messaggio di Risposta fallita

OPENSPCOOP2_ORG_420
È stato riscontrato già un header di protocollo presente nel messaggio di richiesta ↩
    applicativa. L'errore viene generato ad esempio se viene inviato un messaggio contenente ↩
     già un'header eGov al servizio di PortaDelegata del protocollo spcoop

OPENSPCOOP2_ORG_421
Il messaggio di richiesta indicato nella richiesta via IntegrationManager non è un ↩
    messaggio SOAP Valido (Accesso al servizio PD via Integration Manager)

OPENSPCOOP2_ORG_422
Il messaggio di richiesta presente nell'http body della request http (Accesso al servizio ↩
    PDtoSOAP) o il messaggio indicato nella richiesta via IntegrationManager (Accesso al ↩
    servizio PD via Integration Manager) non è utilizzabile tramite la funzionalità di ↩
    Imbustamento per ottenere un messaggio SOAP valido.

OPENSPCOOP2_ORG_423
Servizio invocato con una azione non valida

OPENSPCOOP2_ORG_424
La funzionalità avanzata 'Allega Body' ha generato un errore

OPENSPCOOP2_ORG_425
La funzionalità avanzata 'Scarta Body' ha generato un errore

OPENSPCOOP2_ORG_426
Errore generico che può avvenire durante la gestione della richiesta, dovuto comunque a ↩
    dati forniti nella richiesta stessa (es. Valore SOAPAction scorretto)

OPENSPCOOP2_ORG_427
Errore generato poiché la PdD ha rilevato la presenza di SOAPHeader Element non ↩
    processabili che però richiedono obbligatoriamente il processamento (mustUnderstand=1 e ↩
    actor non presente)

OPENSPCOOP2_ORG_428
Autorizzazione del Contenuto Applicativo Fallita

OPENSPCOOP2_ORG_429
Content-Type Sconosciuto

OPENSPCOOP2_ORG_430
Richiesta che contiene un namespace differente da quello atteso per la versione SOAP ↩
    corrispondente al Content-Type della richiesta
```

```
OPENSPCOOP2_ORG_431
Errore durante la lettura delle credenziali fornite tramite Proxy

OPENSPCOOP2_ORG_432
Contenuto della Richiesta Sconosciuto (es. xml malformato)

OPENSPCOOP2_ORG_433
Richiesta che non contiene l'header Content-Type

OPENSPCOOP2_ORG_434
Il processo di Correlazione Applicativa della Risposta ha generato un errore

OPENSPCOOP2_ORG_435
Richiesta nella porta delegata il Local Forward verso una configurazione non compatibile ↩
    con tale opzione (es. profilo asincrono)

OPENSPCOOP2_ORG_436
Tipo del Soggetto Fruitore non supportato dal Protocollo

OPENSPCOOP2_ORG_437
Tipo del Soggetto Erogatore non supportato dal Protocollo

OPENSPCOOP2_ORG_438
Tipo di Servizio non supportato dal Protocollo

OPENSPCOOP2_ORG_439
Funzionalità non supportato dal Protocollo (es. profiloAsincrono sul protocollo trasparente ↩
    )

OPENSPCOOP2_ORG_440
Contenuto della Risposta Sconosciuto (es. xml malformato)

OPENSPCOOP2_ORG_450
La porta applicativa è inesistente

OPENSPCOOP2_ORG_451
Soggetto inesistente

OPENSPCOOP2_ORG_452
Busta già ricevuta

OPENSPCOOP2_ORG_453
Servizio Applicativo inesistente

OPENSPCOOP2_ORG_454
La risposta del servizio applicativo contiene una busta

OPENSPCOOP2_ORG_516
Connettore utilizzo con errore

OPENSPCOOP2_ORG_517
Il servizio applicativo non ha restituito alcuna risposta (quando attesa)

OPENSPCOOP2_ORG_518
Il servizio applicativo ha restituito un fault come risposta

OPENSPCOOP2_ORG_537
La busta ricevuta è già presente ed attualmente in fase di processamento

OPENSPCOOP2_ORG_538
La richiesta asincrona è ancora in fase di processamento
```

```
OPENSPCOOP2_ORG_539
La ricevuta della richiesta asincrona è attualmente in fase di processamento

OPENSPCOOP2_ORG_5XX
Errore generico durante il processamento del messaggio
```

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