

## Harmonise 2.0

### Specifica tecnica

Versione	Aggiornamento	Stato	Data
0.1	Prima bozza		06-04-2009

Versione: 0.1

Data: **06-04-2009**

ID Documento :

Stato:

Autori: Claudio Prandoni (CPR)

<b>Acronimi e abbreviazioni .....</b>	<b>4</b>
<b>Introduzione .....</b>	<b>5</b>
<b>1. Specifiche di Web Services.....</b>	<b>6</b>
1.1 <i>Specifiche di WSDL.....</i>	6
1.2 <i>Sample client .....</i>	9
<b>2. Specifiche di API di Invio e di Query.....</b>	<b>12</b>
2.1 <i>Specifiche di API di Invio .....</i>	12
2.2 <i>Specifiche dell'API di Invio .....</i>	13
2.3 <i>Esempio del file di query XML: servizio di query euromuse.net.....</i>	14
<b>3. Integrazione del servizio di ricerca di Harmonise in documenti HTML.....</b>	<b>16</b>

## Acronimi e abbreviazioni

API	Application Programming Interface (Interfaccia di programmazione applicativa)
CMS	Content Management System (Sistema di gestione contenuti)
DTD	Document Type Definition (Definizione tipo documento)
HTML	HyperText Markup Language (Linguaggio di marcatura di ipertesto)
HTTP	Hyper Text Transfer Protocol (Protocollo di trasferimento di ipertesto)
IFRAME	Inline Frame (Frame non ancorato)
JAR	Java Archive
JAXB	Java Architecture for XML Binding (Architettura Java per associazione XML)
JAX-RPC	Java API for XML-based RPC (API Java per RPC basata su XML)
JAX-WS	Java API for XML Web Services (API Java per servizi web XML)
RPC	Remote Procedure Call (Chiamata di procedura remota)
SOAP	Simple Object Access Protocol (Semplice protocollo di accesso ad oggetti)
URI	Uniform Resource Identifier (Identificatore uniforme di risorse)
URL	Uniform Resource Locator (Sistema uniforme per l'individuazione delle risorse)
WSDL	Web Services Description Language (Linguaggio di descrizione dei servizi web)
XML	eXtensible Markup Language (Linguaggio di marcatura estensibile)

## Introduzione

Lo scopo di Harmonise consiste nel fornire una soluzione tecnologica che permetta lo scambio di dati tra molti attori, ciascuno dei quali utilizza la propria struttura dei dati.

È stata sviluppata un'interfaccia utente che consente di trasmettere dati ad altri partecipanti e a interrogare dati di altri partecipanti. Per ulteriore dettagli sull'utilizzo del front-end Web, fare riferimento al manuale corrispondente.

L'obiettivo del presente documento consiste nello spiegare come Harmonise può essere utilizzato per scambiare automaticamente dati con altri utenti senza utilizzare l'interfaccia Web.

Il primo capitolo descrive come avvalersi dei Servizi Web di Harmonise per:

T

- a. Trasmettere dati ad altri utenti;
- b. Eseguire query su banche dati di altri utenti per trovare dati specifici.

Il secondo capitolo specifica le API che un partecipante deve implementare per:

- a. Ricevere dati da altri partecipanti attraverso Harmonise, scrivendoli automaticamente nel proprio CMS;
- b. Supportare query che possono essere poste da attori esterni attraverso Harmonise, automaticamente recuperando e trasmettendo dati dal proprio CMS.

Infine l'ultimo capitolo illustra come integrare il servizio di ricerca di Harmonise all'interno delle pagine web esistenti (ossia siti web, portali, ...), grazie all'utilizzo della tecnologia IFRAME.

# 1. Specifiche di Web Services

## 1.1 Specifiche di WSDL

Segue una descrizione di Harmonise Web Services corredata di schemi grafici di WSDL e specifiche XML.

La soluzione tecnologica Harmonise contiene una Interfaccia per Servizi Web `DataExchangeManager` Interface che può essere utilizzata da qualsiasi utente registrato per scambiare dati con tutti gli altri utenti registrati.

Questo Servizio Web consente due operazioni:

- Operazione di invio: può essere chiamata per caricare dati a un partecipante specifico
- Operazione di query: permette di richiedere dati specifici da un partecipante specifico

L'operazione di **invio** utilizza i seguenti parametri di input:

- `sender`: nome del mittente, così come risulta registrato in Harmonise
- `receiver`: nome del ricevente, così come risulta registrato in Harmonise
- `xml_file`: il file XML da inviare (utilizzando il modello di dati del mittente)

Carica e trasmette il file XML di input al destinatario specificato mettendolo nella sua posta in arrivo o introducendolo direttamente nel suo sistema a seconda della configurazione del destinatario. L'operazione restituisce VERO se la trasmissione va a buon fine, altrimenti FALSO, e può sollevare una delle seguenti eccezioni:

- `ACLEException`: se il partecipante non è autorizzato ad eseguire l'operazione di invio richiesta o si verifica un problema durante la verifica dell'autorizzazione
- `ConnectorException`: se si verifica un problema nel collegamento al connettore del ricevente o mittente
- `ReconciliationException`: se si verifica un problema durante la trasformazione dei dati dal formato mittente al formato ricevente
- `IOException`: se si verifica un problema durante la lettura del file XML di input

L'operazione di **query** utilizza i seguenti parametri di input:

- `sender`: nome del mittente della query, così come registrato in Harmonise
- `receiver`: nome del partecipante interrogato, così come registrato in Harmonise
- `maxItems`: numero massimo di risultati ritornati (un numero intero negativo significa che non vi sono restrizioni)
- `xml_query`: il file XML contenente la query da inoltrare (vedere paragrafo 2.3 per trovare un esempio di un file di query XML)

Inoltra il file XML al destinatario specificato eseguendo la query corrispondente e restituendo i risultati al mittente o collocandoli nella sua posta in arrivo o introducendoli direttamente nel suo sistema a seconda della configurazione del mittente. L'operazione restituisce VERO se la query va a buon fine, altrimenti FALSO, e può sollevare le medesime eccezioni dell'operazione di invio:

- `ACLEException`: se il partecipante non è autorizzato ad eseguire l'operazione di query richiesta o si verifica un problema durante il controllo dell'autorizzazione
- `ConnectorException`: se si verifica un problema con la connessione al connettore del ricevitore o mittente
- `ReconciliationException`: se si verifica un problema durante la trasformazione dei risultati della query dal formato ricevente al formato mittente
- `IOException`: se si verifica un problema nella lettura del file di query XML



Lo schema sopra riportato è la rappresentazione grafica degli elementi WSDL di Servizio Web che sono specificati nel file WSDL sotto riportato.

Il file WSDL è strutturato in due parti principali. La prima descrive la definizione dell'interfaccia di servizio astratta, ossia gli elementi `<Types>`, `<Message>`, `<PortType>` e `<Binding>`. La seconda parte descrive i punti finali concreti della definizione dell'implementazione di servizio, ossia l'elemento `<Service>`.

## WSDL specifications

```
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://service.webservices.services.harmonise.org/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://service.webservices.services.harmonise.org/"
name="DataExchangeManagerService">
  <types>
    <xs:schema xmlns:tns="http://service.webservices.services.harmonise.org/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ns1="http://www.w3.org/2005/05/xmlmime"
xmlns:ns2="http://www.w3.org/2005/05/xmlmime"
targetNamespace="http://service.webservices.services.harmonise.org/" version="1.0">
      <xs:element name="ACLEException" type="tns:ACLEException"/>
      <xs:element name="ConnectorException" type="tns:ConnectorException"/>
      <xs:element name="IOException" type="tns:IOException"/>
      <xs:element name="ReconciliationException" type="tns:ReconciliationException"/>
      <xs:element name="query" type="tns:query"/>
      <xs:element name="queryResponse" type="tns:queryResponse"/>
      <xs:element name="send" type="tns:send"/>
      <xs:element name="sendResponse" type="tns:sendResponse"/>
      <xs:complexType name="send">
        <xs:sequence>
          <xs:element name="arg0" type="xs:string" minOccurs="0"/>
          <xs:element name="arg1" type="xs:string" minOccurs="0"/>
          <xs:element name="arg2" type="xs:base64Binary" minOccurs="0"
ns1:expectedContentTypes="application/octet-stream"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="sendResponse">
        <xs:sequence>
          <xs:element name="return" type="xs:boolean"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="ConnectorException">
        <xs:sequence>
          <xs:element name="errorCode" type="xs:int"/>
          <xs:element name="message" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="ACLEException">
        <xs:sequence>
          <xs:element name="errorCode" type="xs:int"/>
          <xs:element name="message" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="ReconciliationException">
```

```

        <xs:sequence>
          <xs:element name="errorCode" type="xs:int"/>
          <xs:element name="message" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="IOException">
        <xs:sequence>
          <xs:element name="message" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="query">
        <xs:sequence>
          <xs:element name="arg0" type="xs:string" minOccurs="0"/>
          <xs:element name="arg1" type="xs:string" minOccurs="0"/>
          <xs:element name="arg2" type="xs:int"/>
          <xs:element name="arg3" type="xs:base64Binary" minOccurs="0"
ns2:expectedContentTypes="application/octet-stream"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="queryResponse">
        <xs:sequence>
          <xs:element name="return" type="xs:boolean"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </types>
  <message name="query">
    <part name="parameters" element="tns:query"/>
  </message>
  <message name="queryResponse">
    <part name="parameters" element="tns:queryResponse"/>
  </message>
  <message name="ACLException">
    <part name="fault" element="tns:ACLException"/>
  </message>
  <message name="ConnectorException">
    <part name="fault" element="tns:ConnectorException"/>
  </message>
  <message name="ReconciliationException">
    <part name="fault" element="tns:ReconciliationException"/>
  </message>
  <message name="IOException">
    <part name="fault" element="tns:IOException"/>
  </message>
  <message name="send">
    <part name="parameters" element="tns:send"/>
  </message>
  <message name="sendResponse">
    <part name="parameters" element="tns:sendResponse"/>
  </message>
  <portType name="DataExchangeManager">
    <operation name="query">
      <input message="tns:query"/>
      <output message="tns:queryResponse"/>
      <fault name="ACLException" message="tns:ACLException"/>
      <fault name="ConnectorException" message="tns:ConnectorException"/>
      <fault name="ReconciliationException" message="tns:ReconciliationException"/>
      <fault name="IOException" message="tns:IOException"/>
    </operation>
    <operation name="send">
      <input message="tns:send"/>
      <output message="tns:sendResponse"/>
      <fault name="ConnectorException" message="tns:ConnectorException"/>
      <fault name="ACLException" message="tns:ACLException"/>
      <fault name="ReconciliationException" message="tns:ReconciliationException"/>
      <fault name="IOException" message="tns:IOException"/>
    </operation>
  </portType>
  <binding name="DataExchangeManagerPortBinding" type="tns:DataExchangeManager">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="query">
      <soap:operation/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>

```

```

        <soap:body use="literal"/>
    </output>
    <fault name="ACLException">
        <soap:fault name="ACLException" use="literal"/>
    </fault>
    <fault name="ConnectorException">
        <soap:fault name="ConnectorException" use="literal"/>
    </fault>
    <fault name="ReconciliationException">
        <soap:fault name="ReconciliationException" use="literal"/>
    </fault>
    <fault name="IOException">
        <soap:fault name="IOException" use="literal"/>
    </fault>
</operation>
<operation name="send">
    <soap:operation/>
    <input>
        <soap:body use="literal"/>
    </input>
    <output>
        <soap:body use="literal"/>
    </output>
    <fault name="ConnectorException">
        <soap:fault name="ConnectorException" use="literal"/>
    </fault>
    <fault name="ACLException">
        <soap:fault name="ACLException" use="literal"/>
    </fault>
    <fault name="ReconciliationException">
        <soap:fault name="ReconciliationException" use="literal"/>
    </fault>
    <fault name="IOException">
        <soap:fault name="IOException" use="literal"/>
    </fault>
</operation>
</binding>
<service name="DataExchangeManagerService">
    <port name="DataExchangeManagerPort" binding="tns:DataExchangeManagerPortBinding">
        <soap:address
location="http://euromuse.ectrldev.com:80/webAccessPortal/DataExchangeManager"/>
    </port>
</service>
</definitions>

```

## 1.2 Sample client

Harmonise Web Services è stato sviluppato utilizzando JAX-WS 2.0, una parte importante della piattaforma Java EE 5. Una successiva release di JAX-RPC 1.1, JAX-WS semplifica l'attività di sviluppo di servizi web utilizzando la tecnologia Java. Risolve alcuni dei problemi di JAX-RPC 1.1 assicurando il supporto di molteplici protocolli quali SOAP 1.1, SOAP 1.2, XML e consentendo di supportare protocolli aggiuntivi rispetto a HTTP. JAX-WS utilizza JAXB 2.0 per l'associazione dei dati e supporta personalizzazioni atte a controllare le interfacce generate di endpoint del servizio. Con tale supporto delle annotazioni, JAX-WS semplifica lo sviluppo dei servizi web e riduce le dimensioni dei file Jar di runtime.

Per accedere a Harmonise, il Servizio Web `DataExchangeManager` descritto al precedente paragrafo, è necessario un programma client. Segue una descrizione dei passaggi da seguire per creare il client:

1. Generare gli elementi portatili necessari per compilare il client
2. Scrivere il client
3. Compilare ed eseguire il client

È possibile utilizzare qualsiasi tipo di tecnologia per sviluppare un tale client (ad es. Axis, Spring WS, ecc...); di seguito descriviamo come crearlo utilizzando la tecnologia JAX-WS.

### 1. Generare gli elementi portatili necessari per compilare il client

Le classi stub del client utilizzabili per richiamare le operazioni dei Servizi Web possono essere generate automaticamente con l'utilità di comando *wsimport*. Tale comando elabora un file WSDL esistente e genera gli elementi portatili necessari per sviluppare le applicazioni di Web Service JAX-WS.

L'utilità *wsimport* può essere lanciata utilizzando lo script della riga di comando *wsimport.sh* (Unix) o *wsimport.bat* (Windows). Unitamente all'utilità è fornito un Ant task per l'utilità *wsimport*. Gli attributi e gli elementi supportati dall'Ant task sono sotto elencati.

```
<wsimport
  wsdl="..."
  destdir="directory for generated class files"
  sourcedestdir="directory for generated source files"
  keep="true|false"
  extension="true|false"
  verbose="true|false"
  version="true|false"
  wsdlLocation="..."
  catalog="catalog file"
  package="package name"
  <binding dir="..." includes="..." />
</wsimport>
```

Attributo	Descrizione	Riga di comando
wsdl	File WSDL	WSDL
destdir	Specifica dove collocare le classi generate di output	-d
sourcedestdir	Specificare dove collocare i file sorgente generati, da mantenere attivo con quest'opzione	-s
keep	Mantiene attivi i file generati con l'opzione sourcedestdir	-keep
verbose	Messaggi di output relativi a quello che sta facendo il compiler	-verbose
binding	Specifica i file di associazione esterni JAX-WS o JAXB	-b
extension	consente le estensioni fornitore (funzionalità non specificata dalla specifica). L'uso delle estensioni può tradursi in applicazioni non portatili oppure può non interagire con altre implementazioni	-extension
wsdllocation	L'URI wsdl URI passata attraverso quest'opzione sarà utilizzata per impostare il valore degli elementi di annotazione @WebService.wsdlLocation e @WebServiceClient.wsdlLocation sulla SEI generata e sull'interfaccia di servizio	-wsdllocation
catalog	Specifica il file di catalogo per risolvere i riferimenti a entità esterne, supporta i formati TR9401, XCatalog, e OASIS XML Catalog. Il tipo ant xmllcatalog può essere inoltre utilizzato per risolvere entità, vedere il campione wsimport_catalog.	-catalog
package	Specifica il pacchetto di destinazione	-p

Gli elementi necessari per sviluppare il client `DataExchangeManager` di Harmonise sono già stati confezionati in un file JAR (`euromuse_webServices-stub.jar`) che è incluso nella release di Harmonise.

## 2. Scrivere il client

La classe seguente, `DataExchangeManagerClient`, fornisce un codice di esempio utilizzabile per sviluppare il client. Richiama le operazioni sia di invio che di query sul servizio distribuito.

### Sample Client

```
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import org.harmonise.services.webservices.client.stub.DataExchangeManager;
import org.harmonise.services.webservices.client.stub.DataExchangeManagerService;

/**
 * Sample client for calling Harmonise push and query web services.
 * Before running the client ("run-client" ant task) it is necessary
 * to generate the stub ("dist-client" ant task).
 */
public class DataExchangeManagerClient {

    private static final String PUSH = "push";
    private static final String QUERY = "query";

    public static void main(String[] args) {
        if(args.length != 4) {
            System.out.println("Error parsing args: != 4 arguments");
            return;
        }
        String sender = args[1];
        String receiver = args[2];
        String xml_file = args[3];
        DataExchangeManagerClient client = new DataExchangeManagerClient();
        if(args[0].equals(DataExchangeManagerClient.PUSH))
            client.doTestSend(sender, receiver, xml_file);
        else if(args[0].equals(DataExchangeManagerClient.QUERY))
            client.doTestQuery(sender, receiver, xml_file);
        else
            System.out.println("Error parsing first arg: 'push' or 'query'");
    }

    public void doTestSend(String sender, String receiver, String xml_file) {
        try {
            DataExchangeManagerService service = new DataExchangeManagerService();
            System.out.println("Retrieving port from service: " + service);
            DataExchangeManager port = service.getDataExchangeManagerPort();
            System.out.println("Invoking the send operation on the port.");
            DataSource source = new FileDataSource(xml_file);
            boolean r = port.send(sender, receiver, new DataHandler(source));
            System.out.println("response: " + r);
        } catch (Exception e) {
            System.err.println("Error in sending the file: " + e.getMessage());
        }
    }

    public void doTestQuery(String sender, String receiver, String xml_query) {
        try {
            DataExchangeManagerService service = new DataExchangeManagerService();
            System.out.println("Retrieving port from service: " + service);
            DataExchangeManager port = service.getDataExchangeManagerPort();
            System.out.println("Invoking the query operation on the port.");
            DataSource source = new FileDataSource(xml_query);
            boolean r = port.query(sender, receiver, -1, new DataHandler(source));
            System.out.println("response: " + r);
        } catch (Exception e) {
            System.err.println("Error in sending the query: " + e.getMessage());
        }
    }
}
```

---



---



---

### 3. **Compilare ed eseguire il client**

Una volta compilato il client utilizzando l'utilità di comando *javac* (utilizzando Ant o riga di comando), è possibile eseguirlo.

Il sample client sopra descritto utilizza come parametri di input:

- *operation*: "push" o "query"
- *sender*: mittente dell'operazione
- *receiver*: ricevente dell'operazione
- *xml\_file*: file da inviare, file XML o query XML

## 2. **Specifiche di API di Invio e di Query**

Al fine di attivare la scrittura automatica dei dati nel CMS e il recupero e l'invio automatico dei dati dal CMS, un partecipante deve implementare due API:

- API di invio
- API di query

Si tratta di due chiamate post del protocollo HTTP da parte di Harmonise per interagire con il sistema proprietario del partecipante.

La **SendAPI** (API di invio) è chiamata da Harmonise per inviare contenuto a un partecipante. Per predefinitone, il contenuto è archiviato nella posta in arrivo del partecipante. Gli utenti interessati a riceverlo e archivarlo direttamente nel proprio CMS devono implementare questo gestore di chiamate HTTP e configurarlo nella propria pagina di configurazione di servizio directory.

La **QueryAPI** (API di query) è chiamata da Harmonise per eseguire ricerche sul contenuto di un partecipante. Gli utenti interessati a supportare query che possono essere poste da attori esterni attraverso Harmonise, con possibilità di recupero e invio automatico di dati dal proprio CMS, devono implementare questo gestore di chiamate HTTP e configurarlo nella propria pagina di configurazione di servizio directory. Questa chiamata dovrebbe restituire la corrispondenza di contenuti che la query ha passato come XML.

### 2.1 **Specifiche di API di Invio**

Il file è inviato da Harmonise al partecipante utilizzando una richiesta post multiparte HTTP standard.

I parametri trasmessi sono:

- *senderid*: id Harmonise del mittente
- *sender*: nome del mittente
- *semail*: e-mail del mittente
- *filename*: nome del file da caricare
- *notes*: alcune annotazioni (facoltativo)
- *xml\_file*: il file XML da caricare (utilizzando il modello di dati del ricevente)

Harmonise chiama l'URL implementata dal ricevente e configurata nella pagina di amministrazione del servizio directory trasmettendo i parametri summenzionati.

La chiamata di Harmonise al partecipante per l'invio di contenuti, dal punto di vista del partecipante, appare come un invio dalla seguente forma:

```

<form method="post"
  action="url implemented by the participant"
  enctype="multipart/form-data">

  Sender id:      <input type="text" name="senderid" value="123">
  Sender name:    <input type="text" name="sender" value="Museum A">
  Sender email:   <input type="text" name="semail" value="museum@a.com">
  File name:     <input type="text" name="filename" value="sample.xml">
  Notes:         <input type="text" name="notes" value="My notes">
  XML File:      <input type="file" name="xml_file">

  <input type="submit" name="import" value="submit">
</form>

```

## 2.2 Specifiche dell'API di Invio

La query è inviata da Harmonise al partecipante utilizzando una richiesta post multiparte HTTP standard.

Un file XML con un esempio del “prodotto” è inviato unitamente ad altri parametri seguendo il cosiddetto approccio Query By Example.

I parametri trasmessi sono are:

- **senderid:** id Harmonise del mittente della query
- **sender:** nome del mittente
- **semail:** e-mail del mittente
- **language:** linguaggio della query
- **limit:** numero massimo di risultati restituiti (una stringa vuota significa “nessuna restrizione”)
- **xml\_query:** il file XML contenente un elemento di esempio utilizzato come “query by example” (per trovare un esempio del file di query XML, vedere il paragrafo 2.3)

Harmonise chiama l'URL implementata dal ricevente e configurata nella pagina d'amministrazione del servizio directory trasmettendo i parametri summenzionati.

Il partecipante interrogato dovrebbe restituire un file XML (utilizzando il proprio modello di dati) contenente tutti gli elementi che corrispondono alla “query by example” passata come parametro.

La chiamata di Harmonise al partecipante per una query sui contenuti, dal punto di vista del partecipante, appare come un invio dalla seguente forma:

```

<form method="post"
  action="url implemented by the participant"
  enctype="multipart/form-data">

  Sender id:      <input type="text" name="senderid" value="123">
  Sender name:    <input type="text" name="sender" value="visiteurope.com">
  Sender email:   <input type="text" name="semail"
                  value="info@visiteurope.com ">
  Language:      <input type="text" name="language" value="en">
  Limit:         <input type="text" name="limit" value="10">
  XML File:      <input type="file" name="xml_query">

  <input type="submit" name="export" value="submit">
</form>

```

Per trovare un esempio di un file XML contenente la query, vedere il paragrafo 2.3.

## 2.3 Esempio del file di query XML: servizio di query *euromuse.net*

Il formato del file XML che descrive la query deve essere specificato dal partecipante che riceve la query, al fine di mappare agevolmente il file di query XML con una reale query al proprio CMS.

Di seguito viene illustrato un esempio di tale file di query XML, ossia quello fornito dal portale *euromuse.net*. La struttura del file XML contenente la query specifica di *euromuse.net* è specificato dalla seguente DTD.

### xml\_query.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT categories (category_id)>
<!ELEMENT category_id (#PCDATA)>
<!ELEMENT date_end (#PCDATA)>
<!ELEMENT date_start (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT exhibition (name, description, date_start, date_end, location,
categories)>
<!ELEMENT exhibitions (exhibition)>
<!ELEMENT location (location_country, location_town)>
<!ELEMENT location_country (#PCDATA)>
<!ELEMENT location_town (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT webservice (exhibitions)>
```

In quest'esempio gli elementi che possono essere oggetto di query sono:

- nome della fiera
- descrizione
- intervallo date (da-a)
- ubicazione (paese, città)
- categoria

Dal punto di vista del partecipante, che in questo caso è *euromuse.net*, il parametro del file `xml_query` della chiamata API di Query appare pertanto come segue:

### xml\_query.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<webservice>
  <exhibitions>
    <exhibition>
      <name></name>
      <description></description>
      <date_start>20090101</date_start>
      <date_end>20091231</date_end>
      <location>
        <location_country>51</location_country>
        <location_town>65</location_town>
      </location>
      <categories>
        <category_id>10</category_id>
```

```
</categories>  
</exhibition>  
</exhibitions>  
</webservice>
```

Ove:

1. i valori di ricerca contenuti nei campi descrittivi (`name`, `description`) vanno interpretati come parole chiave (ossia una query dovrebbe restituire tutte le fiere il cui nome *contiene* il valore specificato nel campo `name/description` e non solo quelli il cui nome è *esattamente* quello) e vanno considerati assieme al parametro `language`
2. `date_start` e `date_end` vanno interpretati come intervallo di date (ossia rispettivamente come “data da” e “data a”)
3. `location_country`, `location_town` e `category_id` sono identificatori attinti dagli elenchi di consultazione di Euromuse (ad es. paese “51” significa “Germania”, città “65” significa “Berlino” e categoria “10” significa “fotografia”).

### 3. Integrazione del servizio di ricerca di Harmonise in documenti HTML

La soluzione tecnica di Harmonise offre anche la possibilità di integrare la funzionalità del servizio di ricerca in pagine web esistenti, quali siti web o portali. Può essere fatto in modo molto semplice grazie all'uso della tecnologia IFRAME.

L'elemento IFRAME definisce un frame non ancorato. Un frame non ancorato o "mobile" ("frame mobile") è un costrutto che incorpora un documento HTML in modo che i dati incorporati siano visualizzati all'interno di una sottofinestra della finestra del browser. Ciò non comporta una inclusione totale: I due documenti restano indipendenti ed entrambi sono trattati come documenti completi anziché essere trattati uno come parte dell'altro.

Fondamentalmente, un elemento iframe ha la seguente forma:

```
<iframe src="URL" more attributes>
alternative content for browsers which do not support iframe
</iframe>
```

I browser che supportano iframe visualizzano il documento riferito dall'URL in una sottofinestra, di norma con barre di scorrimento verticale e/o orizzontale; tali browser ignorano il contenuto dell'elemento iframe (cioè tutto quanto si trova fra il tag di inizio `<iframe...>` e il tag di fine `</iframe>`). I browser che non supportano iframe (o in cui tale supporto è disabilitato) fanno il contrario: elaborano il contenuto come se i tag `<iframe...>` e `</iframe>` non ci fossero.

Quando si utilizzano frame non ancorati, il browser (se li supporta) invia una richiesta al server riferito dall'URL nell'elemento iframe e, dopo avere ottenuto il documento richiesto, lo visualizza all'interno di un frame non ancorato. In tal senso i frame non ancorati sono una questione congiunta browser-server, ma solo il browser ha bisogno di essere specificamente a conoscenza degli iframe; dal punto di vista del server, esiste semplicemente una normale richiesta HTTP di un documento, e il server invia il documento senza avere un'idea (o senza avere necessità di un'idea) di quello che ci farà il browser.

Pertanto, al fine di integrare il servizio di ricerca di Harmonise in una pagina web esistente, è sufficiente collocare l'elemento IFRAME all'interno del documento HTML desiderato specificando l'URL della pagina HTML di Harmonise che contiene il servizio di ricerca come valore dell'attributo IFRAME `src`. Ciò consente di eseguire query utilizzando la soluzione Harmonise e di visualizzare i risultati direttamente nella pagina web in cui è collocato IFRAME.

Segue una breve descrizione dei principali attributi di IFRAME:

- L'attributo `src` di IFRAME fornisce l'ubicazione del contenuto del frame – di norma un documento HTML.
- L'attributo facoltativo `name` specifica il nome del frame non ancorato, consentendo ai link di puntare al frame.
- Il contenuto dell'elemento IFRAME è utilizzato come alternativa per i browser che non sono configurati in modo da visualizzare - o che non supportano - i frame non ancorati. Il contenuto può essere costituito da elementi non ancorati o elementi di blocco, anche se qualsiasi elemento di blocco deve essere consentito all'interno dell'elemento contenente di IFRAME. Ad esempio, un IFRAME all'interno di un `H1` non può contenere un `H2`, ma un IFRAME all'interno di un `DIV` può contenere qualsiasi elemento di blocco.
- L'attributo `longdesc` fornisce l'URI di una lunga descrizione dei contenuti del frame. È particolarmente utile per le descrizioni complete di oggetti incorporati. Si noti che `longdesc` descrive il contenuto del frame, mentre il contenuto dell'elemento IFRAME serve da sostituto quando la risorsa esterna non può essere ancorata.
- Gli attributi `width` e `height` specificano le dimensioni del frame non ancorato in pixel o in termini di percentuale dello spazio disponibile.

- L'attributo `frameborder` specifica se sia necessario tracciare un bordo. Il valore predefinito di 1 si traduce in un bordo mentre un valore di 0 sopprime il bordo. I fogli di stile consentono maggiore flessibilità nel suggerire la presentazione del bordo.
- L'attributo `align` specifica l'allineamento del frame non ancorato. I valori `top`, `middle` e `bottom` specificano la posizione del frame rispetto al contenuto circostante alla propria sinistra e destra. `align=middle` allinea il centro verticale del frame al riferimento corrente. Per centrare il frame orizzontalmente sulla pagina, collocare il frame in un blocco centrato. Gli altri valori di `align`, `left` e `right`, specificano un frame non ancorato (mobile); il frame è collocato sul margine sinistro o destro e il contenuto scorre attorno ad esso. Per collocare il contenuto sotto il frame, utilizzare `<br clear=left|right|all>` quando necessario. Le proprietà `vertical-align` e `float` dei Fogli di Stile CSS presentano metodi più flessibili di allineamento dei frame non ancorati.
- Gli attributi `marginwidth` e `marginheight` definiscono il numero di pixel da utilizzare rispettivamente come margini sinistro/destro e superiore/inferiore, in seno al frame non ancorato. Il valore deve essere un numero intero non negativo.
- L'attributo `scrolling` specifica se sono fornite barre di scorrimento per il frame non ancorato. Il valore predefinito, `auto`, genera barre di scorrimento solo quanto necessario. Il valore `yes` genera barre di scorrimento sempre, e il valore `no` sopprime le barre di scorrimento, anche quando sono necessarie per vedere tutto il contenuto.