檢索系統和數位典藏系統間的通訊協定

郭沛顯　　賴震宇　　林宣華　　何建明

中央研究院資訊科學研究所
臺北市南港區研究院路 2 段 128 號
Tel: 886-2-2788-3799 ext. {1660, 1618, 1657, 1803}
E-Mail: {reno, lawrence, shlin, hoho@iis.sinica.edu.tw}

ABSTRACT

以 SOAP (Simple Object Access Protocol) [9]爲基礎的 XML Web Service[10]是近年來常使用於 HTTP[3]的一種資料交換和服務的方式。SOAP 是一種較爲簡易的 XML[11]資料交換格式，特別適合使用於網路的分散式架構，經由 SOAP/XML 豐富的資料描述能力，透過 HTTP 來存取及交換其所需的資料或服務，不再需要自行定義個別的通訊協定，這正是 Web Service 的好處。本論文以我們所開發的知識文件檢索系統 (KP，Knowledge Portal) 和我們的數位典藏系統 (NDAE，National Digital Archive Environment) [13][14] 爲實驗平台，實作檢索系統與數位典藏系統兩個系統間的資料同步與交換機制，採用 SOAP 的基本設計概念，定義了兩個系統之間的數個基本操作指令和協定。當數位典藏系統有新增或異動時，經由呼叫搜尋引擎所提供的 Web Service (based on SOAP) 將所對應的資料作修正，以達到兩者間的資料同步。

1. 概述

隨著網路的發展，數位內容產業已經逐漸萌芽，政府各單位也一一建置數位典藏內容。爲因應未來各項數位典藏內容之整合運用，整合目錄和檢索機制之建立，是一個相當重要的核心技術問題。不同研究領域或應用領域常常爲各種典藏品定義各種不同的後設資料 (metadata)，爲了整合檢索各種不同的後設資料，必須先要求各種後設資料先對應到一個標準的後設資料格式，例如 IETF 的 Dublin Core [1]。因此，在數位典藏系統中，根據典藏品的欄位或 Dublin Core 欄位來檢索典藏品是必要的功能，如何在檢索系統 (information retrieval systems) [7][12] 與數位典藏系統 (digital archive systems) 兩個開放的異質系統之間交換資料，並使典藏系統的典藏品資料和目錄架構能和檢索系統互相對應與同步化，提供使用者及典藏管理人員快速搜尋和瀏覽相關的資料，是本系統所要達到的目標與成果。爲了讓管理者能彈性修改典藏系統資料對應至檢索欄位的意義，系統還提供 metadata 欄位對應的管理系統，讓管理者設計出符合需求的典藏檢索系統和介面。同時，我們也實做檢索系統與資料庫系統間的資料同步與交換，讓我們的檢索系統能同時支援異質的典藏和資料庫系統的檢索和瀏覽。

2. 系統架構

本系統是介於數位典藏系統和檢索系統之間的溝通橋樑，以下先分別介紹我們所開發的兩個系統：NDAE (National Digital Archive Environment) 和 KP (Knowledge Portal)。

NDAE 是一個開放及分散式的數位典藏系統，其核心主要是對於 metadata 的典藏與管理，提供完整及良好的使用者介面，並支援以 XML 的資料格式與其他數位典藏系統交換典藏品的 metadata 和資料。同時也整合多媒體的管理和後製作，讓典藏品的多媒體數位電子檔可呈現

在網路上。所有典藏品的網頁呈現，都可透過 FTP (File Transfer Protocol) [2] 檔案傳輸協定發行至指定的 Web 站台或其他呈現子系統 (publisher) 等。

KP 的主要架構爲一多欄位的全文檢索系統，其核心包括了資料擷取的網路爬蟲 (crawler) 部份，建立索引的檢索引擎 (index engine)，和提供目錄瀏覽和搜尋服務的目錄/搜尋引擎 (directory/search engine)。KP 在數位典藏的整合系統中，是負責提供使用者快速檢索及呈現導覽的功能。Figure 1 即爲目前數位典藏系統的環境與架構。

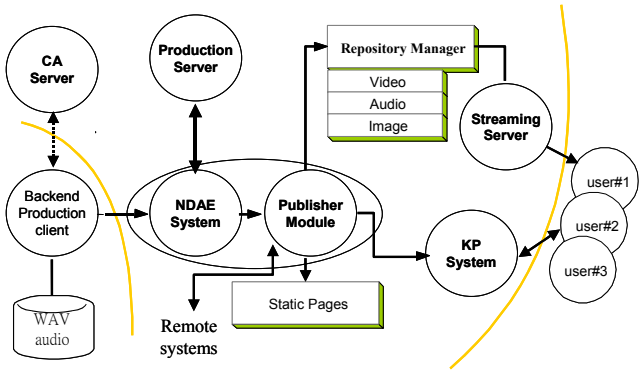

Figure 1. 數位典藏系統環境架構

對於數位典藏典藏品的 metadata 搜尋，我們遵照國際的 Dublin Core 搜尋標準，Dublin Core 提供了 15 個欄位來對典藏品作檢索，NDAE 典藏系統中也提供了將典藏品欄位對應至 Dublin Core 各個欄位的功能，因此在檢索系統的呈現上，也必須提供 Dublin Core 的多欄位資料對應與檢索功能。

我們的同步機制是實作於 NDAE 的發行與呈現子系統 (publisher) 上，同步的主要流程大致如下：

1. 當 NDAE 的發行與呈現子系統在發行時，如果發現典藏品的 metadata 資料有異動，NDAE 會向 KP 送出資料同步的要求。
2. KP 在收到資料需要同步的要求後，會將其送出的 XML metadata 資料解析後，記錄其對應到 Dublin Core 的欄位資訊。
3. 接著再由 KP 的 Crawler 擷取典藏品的呈現網頁，Index Engine 進行分析與建立索引，將典藏品靜態呈現網頁的全文部份及其 Dublin Core 資料索引至資料庫。
4. 透過搜尋的介面，使用者或典藏管理人員便可以透過 Dublin Core 欄位檢索或是全文檢索的方式，搜尋所需要的典藏品。

3. 通訊的方式

目前的檢索系統及典藏系統多爲網路的服務 (Web Service)，而網際網路上最爲通行的通訊協定就是 HTTP (Hyper Text Transfer Protocol)。爲了在兩個以網際網路爲平臺的異質系統之間做簡易的溝通，加上 HTTP 可以不受防火牆限制的方便特性，因此我們選擇了使用 HTTP 來做爲其傳輸資料的通訊協定。

下面將介紹我們定義的通訊協定及 OAI[5]所定義的 metadata 擷取協定 (harvesting protocol)。相同的是，兩者皆使用 XML 來描述典藏品的 metadata 資料，並使用 HTTP 作爲標準的傳輸協定。

3.1 SOAP

HTTP 原本單純用於網頁的傳輸，由於網際網路的發展相當迅速，現在 HTTP 的應用也越來越多，SOAP (Simple Object Access Protocol) 便是一個很好的應用範例。

SOAP 簡單的說就是在 HTTP 上傳輸 XML 的文件 (XML Document)，由於 XML 的彈性及可讀性相當高，且描述 metadata 的能力強大，加上 XML 的資料可以跨平臺使用，這些 XML 的特性及優點對數位典藏系統及檢索系統之間的資料交換是相當有幫助的，因此我們在設計這兩個系統間的通訊同步協定便是建立在 HTTP 及 SOAP 之上。

3.2 OAI-PMH

OAI (Open Archive Initiative) 所制定的 PMH (Protocol for Metadata Harvesting)，同樣是一套基於 XML 的資料交換協定，OAI-PMH[6]是制定數位資料典藏者 (data provider/repository) 與服務提供者 (service provider) 兩者之間的資料

交換協定，較爲不同的是，OAI-PMH 是由服務提供者向資料典藏者提出 HTTP 的 GET 要求，接著資料典藏者會將其要求的 metadata 資料，同樣再以 HTTP 的方式回傳給服務提供者。

和先前 SOAP 通訊協定不同的是，OAI-PMH 在傳送同步資料時，並不將 XML 的資料附加在 HTTP 的資料封包之內，而是將資料打包成 XML 檔案，由服務提供者透過 HTTP 將 XML 檔案取回。

4. 通訊時的動作

接下來我們將介紹以 SOAP 爲基礎的同步通訊協定。數位典藏系統與檢索系統之間的資料同步問題，需要 3 個動作 (operations) 來達成兩個系統資料的同步。

4.1 INSERT

NDAE 典藏系統將典藏品的靜態呈現網頁 (static web page) 發行及公布時，同時告知 KP，將新增的典藏品資料也同步新增至 KP 檢索系統。

4.2 DELETE

NDAE 典藏系統的典藏管理者在刪除典藏品的 metadata 時，也需告知 KP，將資料庫內的相關索引資料刪除。

4.3 UPDATE

NDAE 典藏系統的典藏管理者在更新 metadata 的資料時，如果此 metadata 之前已經由發行與呈現子系統發行過，也需同時告知 KP，將資料庫內的資料更新。

5. 通訊協定的概觀

SOAP 是網路服務所使用的標準交換協定，架構於 HTTP 的網路通訊協定上，並以 XML Document 的資料型態來交換彼此的資料。由於是使用 HTTP，因此 SOAP 可以支援並使用 HTTP 的 GET 或 POST 來取得所需要的 XML Document，端看應用程式如何使用。

因此當我們在設計 NDAE 發行與呈現子系統發與 KP 兩個異質系統的交換協定時，採用了 SOAP 在 HTTP 上傳輸 XML Document 的優點，將需要被檢索的 Dublin Core 15 個欄位資料，以 XML 的表示方式，包裝於 HTTP 的資料傳輸封包內，再由 NDAE 發行與呈現子系統將包裝好的 HTTP 資料以 HTTP 的 POST 方式傳送至 KP。

KP 在收到 request 的 HTTP 資料後，會解析 HTTP 資料內所包含的 XML Document，根據解析完的結果執行相對應的動作：INSERT、DELETE、或 UPDATE。

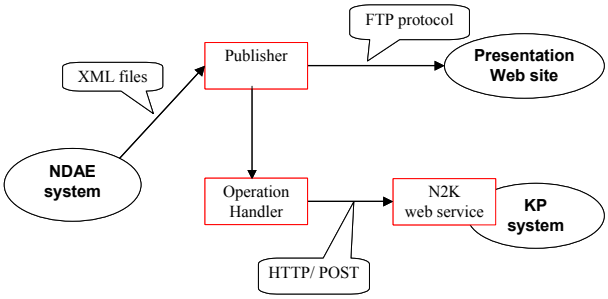

Figure 2. 以 SOAP 爲基礎的同步通訊協定

如Figure 2所示，在 KP 方面，我們提供了 InsertMRec、DeleteMRec 及 DeleteMRec 等 3 個網路服務 (N2K web service)，分別對應到兩個異質系統同步資料時所需的 INSERT、DELETE 及 UPDATE 這 3 個基本操作。接下來我們舉一些實作的範例：

5.1 InsertMRec 的實例

由 NDAE 發行與呈現子系統向 KP 送出 INSERT 的 request，呼叫 InsertMRec 的網路服務，其 HTTP 資料如下：

```
POST /N2K HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=Big5
Content-Length: xxxx
SOAPAction: "http://localhost/N2K"

<?xml version="1.0" encoding="Big5"?>
<soap:Envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<InsertMRec xmlns="http://datf.iis.sinica.edu.tw/">
<MetadataRec CollectionURL="http://foo.com/foo.html" Description="書畫">
<Title>foo</Title>
<Creator>Keyword separated by SPACE</Creator>
. . . (15 Dublin Core elements)
</MetadataRec>
. . . (More metadata elements)
</InsertMRec>
</soap:Body>
</soap:Envelope>
```

資料的第一部份是 HTTP 的標頭 (header)，第二部分就是 SOAP 的 XML 資料。在 soap:Body 的標籤 (tag) 內是這次 request 的 XML 資料，其中 InsertMRec 標籤表示的是要呼叫 InsertMRec 這個網路服務，而 InsertMRec 標籤內包含的是所需要 INSERT 的數筆典藏品 metadata 資料。Metadata 資料則包含在 MetadateRec 的標籤之內，裡面包含的則是有關此筆 metadata 15 個 Dublin Core 要被索引的資訊，如果關鍵詞 (keyword) 有多個以上，那麼關鍵詞與關鍵詞之間將以空白來分隔開。

在 MetadataRec 標籤內有兩個屬性 (attributes)，其中 CollectionURL 屬性表示此 metadata 在發行後其靜態呈現網頁所在的 URL，而 Description 則是對此 metadata 資料的相關描述。

5.2 DeleteMRec 的實例

由 NDAE 發行與呈現子系統向 KP 送出 DELETE 的 request，呼叫 DeleteMRec 的網路服務，其 HTTP 資料如下面的範例所示：

```
POST /N2K HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=Big5
Content-Length: xxxx
SOAPAction: "http://localhost/N2K"

<?xml version="1.0" encoding="Big5"?>
<soap:Envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<DeleteMRec xmlns="http://datf.iis.sinica.edu.tw/">
<MetadataRec CollectionURL="http://foo.com/foo.html" Description="書畫" />
. . . (More metadata elements)
</DeleteMRec>
</soap:Body>
</soap:Envelope>
```

同樣的，資料的第一部份是 HTTP 的標頭，第二部分則是 SOAP 的 XML 資料。在 soap:Body 的標籤內是此 request 的 XML 資料，其中 DeleteMRec 標籤則是呼叫 KP 的 DeleteMRec 網路服務，在 DeleteMRec 標籤內包含的是所需要被 DELETE 的 metadata 資料。需要處理的 metadata 資料一筆則以一個 MetadateRec 的標籤表示，同樣的在 MetadataRec 標籤內有兩個屬性，其中 CollectionURL 屬性表示此 metadata 在發行後其靜態網頁所在的 URL，KP 將會根據此 URL 刪除索引過的資料。而 Description 則是對此 metadata 資料的相關描述。

5.3 UpdateMRec 的實例

由 NDAE 發行與呈現子系統向 KP 送出 UPDATE 的 request，呼叫 UpadteMRec 的網路服務，其 HTTP 資料如下面的範例所示：

```
POST /N2K HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=Big5
Content-Length: xxxx
SOAPAction: "http://localhost/N2K"

<?xml version="1.0" encoding="Big5"?>
<soap:Envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<UpdateMRec xmlns="http://datf.iis.sinica.edu.tw/">
<MetadataRec CollectionURL="http://foo.com/foo.html" Description="書畫">
<Title>foo</Title>
<Creator>Keyword separated by SPACE</Creator>
. . . (15 Dublin Core elements)
</MetadataRec>
. . . (More metadata elements)
</UpdateMRec>
</soap:Body>
</soap:Envelope>
```

同樣的，資料的第一部份是 HTTP 的標頭，第二部分則是 SOAP 的 XML 資料。UPDATE 這部分和 INSERT 的動作相當類似，因此在 soap:Body 標籤內的 XML 資料和 INSERT 時是一樣的。UpdateMRec 標籤呼叫 KP 的 UpdateMRec 網路服務，在 UpdateMRec 標籤內則是更新過後的 metadata 資料。和 INSERT 時相同，metadata 資料則包含在 MetadateRec 的標籤之內，裡面是 metadata 15 個 Dublin Core 更新後的資訊。詳細的 XML 資料的定義，請參閱附錄的 DTD 文件。

6. OAI-PMH

在 OAI-PMH 資料擷取協定中，定義了 6 個資料擷取時可用的指令 (OAI verb)，而在 OAI-PMH 2.0 的規範當中，OAI 建議且規定必須支援 Dublin Core 的欄位描述，下面將介紹 KP 與 OAI repository 同步資料時所用到的 verb。

6.1 ListSets

OAI 規範中建議數位典藏者，如果其典藏的 metadata 資料數量太過龐大，可以將其資料分爲數個 set，而每個 set 有其獨一無二識別碼 (setSpec) 來加以區別。ListSets 這個指令可列出典藏者自行定義的 set 及其分別對應的識別

碼。我們所開發的 KP 全文檢索系統支援整合型的目錄分類瀏覽功能，因此利用 ListSets 我們可以將典藏者所定義的 set 對應到 KP 的目錄架構。

6.2 ListIdentifiers

對於典藏系統內的每一筆典藏資料，同樣必須要有其獨一無二的識別碼 (identifier)，此識別碼的產生則是根據 URI (Uniform Resource Identifier) [4] 所制定的規範。ListIdentifiers 指令可向典藏者要求列出其所有典藏品的唯一識別碼，根據識別碼我們便可擷取此典藏品的相關 metadata 資料。

6.3 GetRecord

GetRecord 指令可根據識別碼，擷取此識別碼所代表典藏品的相關 metadata 資料，而依照 metadataPrefix 參數的設定，OAI-PMH 可提供 Dublin Core、rfc1807 與 oai_marc 等各種不同的 metadata 資料描述格式。

7. 實作 OAI-PMH

我們以 OAI-PMH 的資料擷取協定爲基礎，在 KP 全文檢索系統上實作了一個網路爬蟲程式 (OAI crawler)，負責將 OAI repository 內的典藏品資料，以 Dublin Core 的資料描述方式，對應至 KP 的檢索欄位，再由 Index Engine 根據個別的欄位將之索引。如 Figure 3 所示：

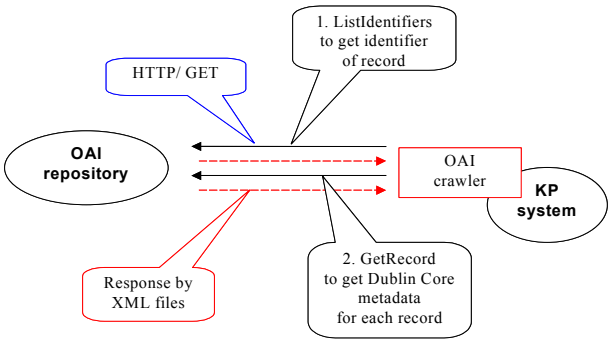

Figure 3. OAI crawler 與 OAI 典藏者的資料同步流程

首先，OAI crawler 會以 HTTP 的 GET 向典藏者送出 ListIdentifiers 的要求，接著 OAI crawler 根據典藏者所傳回的 XML 檔案將之解析後，便可得到典藏品所對應的唯一識別碼，利用此識別碼，OAI crawler 再向典藏者送出 GetRecord 的指令要求，便可得到典藏品的 Dublin Core metadata 描述資料，最後將 Dublin Core 的各個欄位資料對應並儲存到 KP 的資料庫中，便完成資料擷取的動作。當然，再經過 KP 的 Index Engine 將資料索引後，使用者便可以透過 KP 全文檢索系統搜尋並瀏覽典藏者所擁有的典藏品資料。

8. 結論

在這篇論文報告中，是我們在實做 KP 與數位典藏系統資料同步時的一些經驗，我們自行定義的同步通訊協定採用 SOAP 及 HTTP 的優點，再加上使用 Web Service 的連結呼叫，而由典藏系統以 SOAP 的方式向檢索系統提出 Web Service 的要求，在同步的時間上也有即時的優點，當典藏品的資料有異動時，檢索系統馬上可以得知並更新相關的資料，這也是採用 Web Service 的好處之一。

而以 OAI-PMH 的資料擷取同步方式，必須由檢索系統自行向典藏系統擷取所需要的典藏品資料，因此在資料同步上較不具有即時更新的優點。OAI-PMH 定義的 set 及 ListSets 指令，在典藏品的分類與管理上，也可以提供檢索系統對應分類目錄架構時的一種同步方式。

檢索系統與典藏系統間的資料同步方式，當然有許多其他的設計方式，但以 XML 來作爲異質系統間跨平台溝通的方式，是未來的趨勢之一。這篇論文提供了一些實作的經驗，也證明了結合 XML 與 Web Service 的通訊方式將是較爲可行的同步方式。

9. 參考文獻

[1] Dublin Core, http://dublincore.org/.

[2] IETF FTP, "File Transfer Protocol", http://www.ietf.org/rfc/rfc959.txt.

[3] IETF HTTP, "HyperText Transfer Protocol", http://www.ietf.org/rfc/rfc2068.txt.

[4] IETF URI, "Uniform Resource Identifier", http://www.ietf.org/rfc/rfc2396.txt.

[5] OAI, "Open Archives Initiative", http://www.openarchives.org/.

[6] OAI-PMH, "Open Archives Initiative Release Version 2.0 of the Protocol for Metadata Harvesting", http://www.openarchives.org/news/oaiv2press020614.html.

[7] Salton, G., "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer," Addison Wesley, 1989.

[8] W3C HTML, "HyperText Markup Language," http://www.w3.org/MarkUp/.

[9] W3C SOAP, "Simple Object Access Protocol", http://www.w3.org/2000/XP/Group/.

[10] W3C Web Service, http://www.w3.org/2002/WS/.

[11] W3C XML, "Extensible Markup Language," http://www.w3.org/XML/.

[12] William B. Frakes, Ricardo Baeza-Yates, "Information Retrieval: Data Structures & Algorithms," Prentice Hall PTR, 1992.

[13] 范紀文, 何建明, "數位典藏系統與工具──輕鬆建立屬於您的典藏管理系統", PNC 2000 年數位典藏及 TEI 研討會, http://pnclink.org/events/2000dlm/news.html.

[14] 范紀文, 何建明, 李德財, "從典藏資料交換角度探討Metadata設計與標準化問題", 新世紀圖書館與數位博物館趨勢研討會, 2001.11.

10. 附錄

我們的交換協定遵照 XML Document 的規範，因此在 SOAP:body 內的資料，可以用 DTD (Document Type Definition) 標準來定義交換協定內 XML 文件架構與規則，以下是詳細的 DTD 規範。

10.1 InsertMRec 的 DTD

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE InsertMRec [
<!ELEMENT InsertMRec (MetadataRec+)>
<!ELEMENT
MetadataRec (Title?,Creator?,Subject?,Description?,
Publisher?,Contributor?,Date?,Type?,Fotmat?,Identifier?,
Source?,Language?,Relation?,Coverage?,Rights?)>
<!ATTLIST MetadataRec
CollectionURL CDATA #REQUIRED
Description CDATA #IMPLIED>
<!ELEMENT Title (#CDATA)>
<!ELEMENT Creator (#CDATA)>
<!ELEMENT Subject (#CDATA)>
<!ELEMENT Description (#CDATA)>
<!ELEMENT Publisher (#CDATA)>
<!ELEMENT Contributor (#CDATA)>
<!ELEMENT Date (#CDATA)>
<!ELEMENT Type (#CDATA)>
<!ELEMENT Format (#CDATA)>
<!ELEMENT Identifier (#CDATA)>
```

```
<!ELEMENT Source (#CDATA)>
<!ELEMENT Language (#CDATA)>
<!ELEMENT Relation (#CDATA)>
<!ELEMENT Coverage (#CDATA)>
<!ELEMENT Rights (#CDATA)>
]>
```

10.2 DeleteMRec 的 DTD

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE DeleteMRec [
<!ELEMENT DeleteMRec (MetadataRec+)>
<!ATTLIST MetadataRec
CollectionURL CDATA #REQUIRED
Description CDATA #IMPLIED>
]>
```

10.3 UpdateMRec 的 DTD

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE UpdateMRec [
<!ELEMENT UpdateMRec (MetadataRec+)>
<!ELEMENT
MetadataRec (Title?,Creator?,Subject?,Description?
,Publisher?,Contributor?,Date?,Type?,Fotmat?,Identifier?
,Source?,Language?,Relation?,Coverage?,Rights?)>
<!ATTLIST MetadataRec
CollectionURL CDATA #REQUIRED
Description CDATA #IMPLIED>
<!ELEMENT Title (#CDATA)>
<!ELEMENT Creator (#CDATA)>
<!ELEMENT Subject (#CDATA)>
<!ELEMENT Description (#CDATA)>
<!ELEMENT Publisher (#CDATA)>
<!ELEMENT Contributor (#CDATA)>
<!ELEMENT Date (#CDATA)>
<!ELEMENT Type (#CDATA)>
<!ELEMENT Format (#CDATA)>
<!ELEMENT Identifier (#CDATA)>
<!ELEMENT Source (#CDATA)>
<!ELEMENT Language (#CDATA)>
<!ELEMENT Relation (#CDATA)>
<!ELEMENT Coverage (#CDATA)>
<!ELEMENT Rights (#CDATA)>
]>
```