

# Implementering av OAI-PMH i ett bibliotekssystem — Några ord på vägen

Sigfrid Lundberg (*sigfrid.lundberg@lub.lu.se*)  
Biblioteksdirektionen  
Lunds universitet

## SAMMANFATTNING

Syftet med denna text är att ge lite kött på benen om vad OAI-PMH är, vad det kan användas till och hur det skulle kunna integreras i ett bibliotekssystem. Det är inte en ersättning för andra dokument utan skall ses som några ord på vägen för organisationer som planerar att eller är i färd med att implementera OAI-PMH.

### 1. Bakgrund

OAI PMH (Open Archives Initiative Protocol for metadata harvesting) är ett Internet-protokoll som har fått avsevärd uppmärksamhet de senaste åren. De viktigaste anledningarna till detta är måhända:

- Det är enkelt att implementera. Det är en applikation av XML som använder HTTP för transport. Den infrastruktur och den kunskap som krävs för att bygga OAI-applikationer är därmed lättillgängliga.
- Till skillnad från mer generella XML-baserade protokoll, som t ex SOAP, behöver en OAI-klient ingen, eller mycket liten, kunskap om XML. Alla transaktioner sker med enkla HTTP GET- eller POST-förfrågningar. De bäddas aldrig in i någon XML-struktur som behöver parsas på serversidan.

Jag återkommer till protokollet längre fram i denna lilla skrift.

#### 1.1. Uppdraget

Detta dokument skrivs på initiativ från Axiell-Bookit.[AXIELL] Dess syfte är att föra över tankar och erfarenheter från en som arbetat med protokollet till dem som är på väg att påbörja ett sådant arbete. Nedan beskriver jag hur jag uppfattat mitt uppdrag.

Protokollet är redan väl specificerat. Det vore inte meningsfullt att göra det igen. *Däremot* innehåller den formella specifikationen [OAI-SPEC] ett antal "keywords":

*In this document the key words "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional" in bold face are to be interpreted as described in RFC 2119 . An implementation is not conformant if it fails to satisfy one or more of the "must" or "required" level requirements for the protocols it implements.*

RFC 2119 [IETF] är ett språkligt dokument. Det beskriver hur man formulerar *krav* på implementeringar av ett Internet-protokoll. Om man skall skriva en applikation så måste alla *must* uppfyllas. Likaså måste man göra vad som är *required*. Jag kommer att lägga mina krafter på vad som är *optional* och *recommended*.

Vad är bra att ha och vad är umbärligt?

## 1.2. Om detta dokument

Jag har gett texten följande struktur: Först ger jag en allmän bakgrund till OAI-PMH, därefter ger jag en del lösa och ganska personliga tankar kring idén att integrera en OAI-server i en OPAC. Efter dessa, mer allmänna prosaavsnitt går jag in och ger direkta rekommendationer till arbetet med en implementering av OAI inom ramen för Book-IT. Slutligen ger jag en del pekare till programvaror som kan hjälpa implementatören — det är onödigt att återuppfinna hjulet. Jag avslutar med några konklusioner.

## 2. Vad gör OAI-PMH

### 2.1. OAI-PMH är ett protokoll

OAI-PMH är ett protokoll för datorkommunikation. Så är FTP, HTTP och Z39.50. Till skillnad från dessa bygger OAI inte direkt på TCP/IP. Det parasiterar, precis som SOAP och XML-RPC, på HTTP som transportmedium. XML-baserade protokoll har fått en hel del uppmärksamhet under de senaste åren. [PROTO] Till skillnad från SOAP och XML-RPC skickas inte frågan i OAI-PMH i XML, bara svaret.

Syftet med protokollet är att göra det möjligt att kommunicera metadata. Specifikationen talar om *Data Providers* och *Service Providers*:

- Data Providers administer systems that support the OAI-PMH as a means of exposing metadata; and
- Service Providers use metadata harvested via the OAI-PMH as a basis for building value-added services.

I det följande kommer jag att hänvisa till dessa som *OAI-server* respektive *OAI-klient*. OAI-protokollets klient är en robot (en harvester), som skickar förfrågningar till servern som enkla HTTP GET eller POST förfrågningar (en OAI-server skall klara av både GET och POST). Till skillnad från SOAP bygger alltså OAI-PMH i stort på den traditionella arkitekturen för Worldwide Web (Representational State Transfer [REST]).

### 2.2. Hur en OAI-server svarar på frågor

Typiskt definierar man en bas-URL för en OAI-server. Den kan i princip se ut hur som helst, men den leder till ett skript som har tillgång till den databas vari metadata lagras. Frågorna till servern ges som CGI-parametrar till skriptet.

Det är bara ett antal enkla typer av frågor som behöver implementeras. Minns att OAI-PMH inte är ett sökprotokoll (även om databasen som är kopplad till servern förmodligen måste utföra en sökning för att svara på frågan), utan ett protokoll för överföring av metadata från en OAI-Server (Data Provider) till en OAI-klient (en Harvester stationerad hos en Service Provider). Frågorna en OAI-server skall kunna besvara benämns verb. Dessa är sju till antalet (Tabell 1).

Varje verb har sin egen uppsättning med argument. På samma sätt finns det en uppsättning med felmeddelanden som måste implementeras, tillsammans med hantering av vissa responskoder i HTTP. Många programmerare är vana vid att returnera även allvarliga felmeddelanden med statuskoden 200 OK. Det duger till slutanvändare, men inte i en OAI-server.

Table 1. Verb som utnyttjas i OAI-PMH

<b>GetRecord</b>	Används för att fråga om en enskild post i databasen
<b>Identify</b>	En förfrågan till servern om att den skall indentifiera sig.
<b>ListMetadataFormats</b>	Gör det möjligt att fråga en server om dess repertoar av olika metadatasyntaxer. Den syntax man väljer skall vara beskriven med ett XML-schema.[XSD] Det finns olika schema-språk för XML. Flera av dem har fördelar framför XSD, och många har haft svårt att se skönheten i just detta språk. OAI har valt det, och det finns ingen återvändo. I praktiken är det oai_dc som gäller för enkel E-publicering.
<b>ListRecords</b>	Det verb som man använder för mer storskalig skördning (harvesting)
<b>ListIdentifiers</b>	Returnerar enbart huvudet för varje post
<b>ListSets</b>	Returnerar beskrivningar av de kategoriseringar som används. För enkla E-publiceringssystem är det, i praktiken, en lista över systemets ämneskategorier. Det är dock möjligt att utarbeta detta ytterligare, eftersom varje set kan innehålla kompletta metadata för varje samling.

Serversnors resporser till dessa förfrågningar skall kodas i XML, och skall — självklart — validera gentemot de schema man implementerar. Mer specifika överväganden om hur protokollet OAI-PMH bör implementeras ges längre fram.

### 3. Vad kan OAI-PMH göra för nytta i en OPAC?

OAI-PMH är till för att göra metadata i ett digitalt bibliotek (till exempel poster i en bibliografisk databas) synliga i maskinläsbar form. Denna synlighet kan i sig vara något man önskar sig, eller inte önskar sig, av olika skäl.

Innan man börjar integrera OAI i en OPAC-programvara bör man naturligtvis ha klart för sig varför man skulle vilja använda OAI i ett sådant sammanhang. Anledningarna till varför man skulle vilja göra det skiljer sig naturligtvis mellan integrering av en OAI-server och en OAI-klient. Innan jag går in på varför man skulle vilja ha sådana moduler i sin OPAC, kommer jag att ta upp frågan varför man *inte* skulle vilja ha dem. Likaså finns det skäl till varför hela OPACen bör exporteras.

- Är bibliotekets OPAC den rätta hemvisten för dessa metadata? Det är inte självklart att bibliotekskatalogen är den bästa plattformen för publicering från de lokala användarnas perspektiv.

- Det är inte självklart att man har rätt att publicera licensierad metadata i elektronisk form. Det måste finnas gränssnitt för att ange vilka delar av OPACen som skall vara exporterbara med OAI. Sannolikt skall vissa poster inte gå att exportera vidare.
- Vare sig metadata är upphovsrättsligt möjliga att distribuera eller ej, måste man naturligtvis ha en klar uppfattning om omvärlden efterfrågar dem. Bara det som har sitt ursprung i eller är unikt för den lokala bör spridas. Likaså bör en sådan distribution föregås av någon form av kvalitetskontroll.

### 3.1. OAI-server

Om ett biblioteks OPAC är strukturerad så att olika samlingar (vare sig de är elektroniska eller ej) går att söka såväl separat som tillsammans, kan det vara en bra lösning för en mindre organisation att använda sin OPAC som publiceringssystem. Den är sannolikt organisationens mest kapabla bibliografiska söksystem, och är utrustad med ett kapabelt katalogiseringsverktyg.

Enligt min uppfattning är modularisering av databasen i olika samlingar en kritisk egenskap. Distribution av olika rapporter och texter är ett viktigt uppdrag för många organisationer, inom såväl privat som offentlig sektor. Inom högskolevärlden är denna uppgift central. Ett publiceringssystem bör därför definitivt ha många olika användargränssnitt. I vissa av dem skall bara den egna organisationens publikationer synas. Ett av dessa användargränssnitt kan vara OAI-PMH.

### 3.2. OAI-klient

Det sannolikt intressantaste skälet till varför man skulle vilja integrera en OAI-klient (en harvester) i ett bibliotekssystem är att den skulle kunna fungera som ett elektroniskt akquisitionssystem. Om man i en större organisation, exempelvis en högskola, publicerar dokument och metadata enligt en distribuerad modell, kan i så fall OPACen (eller ännu hellre en *samling i den*) fungera som en samkatalog.

Den gängse metoden att uppdatera poster i bibliografiska databaser i svenska bibliotek är att de köps från en större databas, t ex BURK eller Libris. Jag har bara vaga uppfattningar om hur detta går till rent tekniskt, men från vad jag hört använder man en rad olika metoder. Z39.50 update är sällsynt enligt vad jag vet — det lär kunna användas för att föra till den beståndsuppgift i BURK som ett köp av en katalogpost medför. För själva överföringen lär användning av FTP vara det vanligaste.

Det synes mig vara sannolikt att det skulle gå att bygga betydligt mer standardiserade gränssnitt för dessa funktioner ovanpå OAI-PMH. Jag skulle gärna se att detta vore något som beaktades i samband med en övergång till FRBR.[FRBR] Annars fungerar OAI-PMH redan idag utmärkt med MARC 21.

Jag är naturligtvis helt medveten om att idén att inkludera en OAI-klient i Book-IT ligger helt utanför det aktuella projektet.

## 4. Tillgängliga verktyg

Även om det inte alls är en orimlig uppgift att implementera OAI-PMH från grunden för en godtycklig bibliografisk databas, så är det egentligen onödigt att göra det. Det finns mängder med fria programvaror att tillgå. Hur fungerar detta rent juridiskt? Svaret kan inte bli entydigt. Det beror på hur den fria licensen är formulerad. Generellt kan sägas att de så kallade Copyleft-licenserna[GNU] är mycket generösa. Det går mycket väl att *sälja* programvara som är under GNU-

licens. Den populära relationsdatabasen MySQL är en sådan. Likaså går det att distribuera GNU-verktyg med programvara som licensieras enligt andra principer.

OAI har en utmärkt lista med verktyg. [VERKTYG] Alla produkter på listan har någon form av öppen licens (flera är tyvärr mer restriktiva än GNU-licensen). Där finns kompletta system och programmeringsgränssnitt för olika språk och plattformar. Även om man bestämmer sig för att skriva sin applikation från grunden, så utgör dessa verktyg en viktig referens till hur OAI-PMH fungerar. Ett axplock:

- En verklig lättviktslösning är XML-File. [XFile] Utan programmering eller andra "svårigheter" får man en OAI-server genom att spara metadata som XML-filer.
- Ett bibliotek i språket PHP. [OAI-PHP]
- Samma som ovan i språket PERL. Programmeringsgränssnitt för både server och harvester. Går att koppla till befintliga databaser [OAI-PERL]
- OAICat — Java SERVLET som implementerar en OAI-server. Från OCLC [OAICat]

## 5. Några ord på vägen

Källorna till information om hur OAI-PMH skall implementeras är specifikationen och Implementation Guidelines [OAI-IMPL] som är en samling dokument, med bland annat ett för implementatörer av OAI-serverar. Det är ett litet arbete i jämförelse med en implementering av Z39.50. Min utskrift av specifikationen för OAI är 33 sidor medan min utskrift av Z39.50 är på över 500. Arbetet kan ytterligare förenklas av användning av verktyg av det slaget jag beskriver ovan.

I det följande ger jag en guidad tur till dessa dokumentation, och på vägen ger jag lite goda råd. Ibland stannar jag upp inför någon sevärdhet.

OAI-specen består i allt väsentligt två delar: En formatdel (sektionerna 2. Definitions and concepts samt 3. Protocol features) och en protokolldel (sektion 4. Protocol Requests and Responses. Den senare innehåller de verb jag nämnt ovan, och de argument en klient kan ange för vart och ett av dem. Den förra delen beskriver mer protokollet mer generellt samt de olika format som kommer till användning.

### 5.1. Format etc

- **Unik identifierare** (Sekt. 2.4): Varje post måste förses med en unik identifierare. Identifieraren *skall* ha en syntax som motsvarar den av en URI. Den behöver inte vara en OAI-identifierare och den identifierar posten, inte objektet.
- Varje post (Sekt. 2.5) skall levereras i ett specifikt format, vilket består av två delar: ett huvud (**head**) och **metadata**. I huvudet finns ett statusfält som inte är obligatoriskt. Jag skulle varmt anbefalla att detta, om möjligt implementeras. Det innebär att om ett dokument och dess metadata raderas ur katalogen, kommer informationen om denna händelse att föras vidare till OAI-klienten.
- **Set** (Sekt. 2.6): Implementeringen av OAI Sets är frivillig. Problemet med Sets är att det inte finns någon standardisering på området och att den användning av Sets som förekommer idag sker på egen risk. [SETS] Jag anbefaller dock varmt implementering av Sets. Implementeringen av det måste

vara konfigurerbart, så tillvida att det skall gå att redigera de set man använder. Det kan vara möjligt att det går att avgöra en resurs' Set-tillhörighet utifrån dess klassifikation, men det kan lika gärna vara så att katalogisatorerna kommer att (utöver klassifikation) få välja en Set-kategori för resurserna. Den Setstruktur som finns skall kunna exporteras via verbet "listSets"

- **Selective harvesting** (Sekt. 2.7). Hänger samman med Sets (se ovan), och med hantering av datum. Den mest typiska formen av skördning är att man i sin robot ber om metadata publicerad från ett givet datum och klockslag. En serverimplementering kan ange klockslag ner på sekundnoggrannhet. För att vara ärlig har jag aldrig insett varför det skall kunna vara till någon glädje. Man måste dock välja, och det val man gör skall dokumenteras i den information servern lämnar vid en "identify"-förfrågan
- Hela **Protocol features** (Sekt. 3) är viktig läsning. **HTTP Response Format** (Sekt. 3.1.2) är viktigt, eftersom det handlar om HTTP statuskoder som vi internetprogrammerare vanligtvis slarvar med. Avdelningen 3.2 är kanske det mest stimulerande, eftersom det är här delar av XML-maskineriet definieras. **Flow control** (Sekt. 3.5) handlar om hanteringen av själva flöverföringen. Det är viktigt att en server inte överlastas, och att få detta rätt kan vara en viktig överlevnadsfaktor för en belastad server.
- Till Flow control hör det objekt som kallas **resumptionToken**. Servern förmodas skicka poster i lagom munsbitar, och detta lilla objekt berättar för skördaren hur och var och när den kan fortsätta skörda. Att få detta rätt är utomordentligt viktigt.
- Till formatfrågorna hör valet av syntax för metadata. Jag ser det som en självklarhet att en server från början implementerar **oai\_dc**. Det finns andra alternativ, och Marc 21 i XML är ett av dem. Qualified Dublincore kommer med all sannolikhet inom en inte alltför avlägsen framtid, likaså MODS. Standardiseringen här är ett rörligt byte, och man bör skriva sin server så att det i en framtid går lätt att installera nya syntax-moduler.

## 5.2. Protokoll

- OAI-PMHs olika verb avhandlas i Sekt. 4 (se även Tabell 1 i denna pamflett). Notera att man även måste kunna förmedla en rad olika felmeddelanden. Dessa sammanfattas i Sekt. 3.6.
- För verbet **Identify** finns det alternativa "description containers" (Sekt. 4.2). För närvarande är eprints den vanligaste. Jämför Guidelines-dokumentet.
- I responsen för **listSets** (Sekt. 4.3) verkar det bara vara obligatoriskt att ange setSpec. Jag vill gärna ha både setSpec och setName. Detsamma gäller i huvudet (head) för en post.

## 6. Slutsatser

Att ingegrera en OPAC med ett publiceringssystem är en stimulerande tanke. En aspekt att beakta i sammanhanget är att publicering och akquisition är två mycket olika processer. Den förra är kontrollerad av lektörer som typiskt har som uppgift att upprätthålla en förläggares kvalitetspolicy. Den senare processen är till för att tillfredsställa en biblioteksanvändarnas informationsbehov. Innan jag slutar skulle jag vilja ställa frågan om dessa två funktioner går att förena?

Så återstår det nog bara att önska alla lycka till med sina implementeringar av OAI-MPH!

## 7. Tack

Texten har skrivits med stöd från Axiell Book-IT AB. Sara Kjellberg har läst och kommenterat en tidigare version, vilket förbättrat resultatet.

## Referenser

- [AXIELL] <http://www.axiellbookit.com/>
- [OAI-SPEC] <http://www.openarchives.org/OAI/openarchivesprotocol.html>
- [IETF] <http://www.ietf.org/rfc/rfc2119.txt>
- [PROTO] <http://www.xml.com/lpt/a/ws/2000/05/10/protocols/index.html>
- [REST] [http://www.prescod.net/rest/rest\\_vs\\_soap\\_overview/](http://www.prescod.net/rest/rest_vs_soap_overview/)
- [XSD] <http://www.w3.org/XML/Schema>
- [FRBR] <http://www.ifla.org/VII/s13/frbr/frbr.pdf>
- [GNU] <http://www.gnu.org/licenses/licenses.html>
- [VERKTYG] <http://www.openarchives.org/tools/tools.html>
- [XFile] <http://www.dlib.vt.edu/projects/OAI/software/xmlfile/xmlfile.html>
- [OAI-PHP] <http://physnet.uni-oldenburg.de/oai/>
- [OAI-PERL] <http://oai-perl.sourceforge.net/>
- [OAI-Cat] <http://www.oclc.org/research/software/oai/cat.shtm>
- [OAI-IMPL] <http://www.openarchives.org/OAI/2.0/guidelines.htm>
- [SETS] <http://www.openarchives.org/OAI/2.0/guidelines-repository.htm#MinimalImplementation-Sets>