

SRU, MPEG21 DIDL en de resolver bij de KB.

Dit document beschrijft het gebruik, de onderlinge relatie en specifieke KB keuzes m.b.t. SRU, MPEG21 DIDL. Het doel hiervan is dat bij nieuwe projecten de data automatisch zodanig aangeboden worden dat applicaties met een minimum aan voorkennis deze data kunnen vinden en benaderen.

We hebben te maken met

- 1) metadata die we opslaan in KB MDO en indexeren in onze index
- 2) fulltext die we niet apart in KB MDO opslaan maar wel indexeren en waarvoor geen metadata record hoeft te bestaan
- 3) complexe objecten, waarvan het surrogaat (bijvoorbeeld de mpeg21 container) wordt opgeslagen en waarvan delen eventueel geïndexeerd worden en waar wel of niet metadata bij kunnen horen.

Applicaties zonder enige voorkennis zullen zoeken in de index en het veld dc:identificer (zowel in het dc recordschema als het dcx recordschema) gebruiken om het object op te vragen. Zonder voorkennis over mpeg21 zal op basis van dc:identificer de redelijkerwijs meest logische presentatie voor dat specifieke object geboden moeten worden. Dit alles moet onafhankelijk zijn van de specifieke website die voor desbetreffende collectie gemaakt wordt. Uiteraard kunnen eigen websites wel gebruik maken van de kennis over de KB infrastructuur.

SRU

Met SRU kunnen we zoeken en records opvragen. De index (nu Verity) bevat de korte titelpresentatie en wordt beschikbaar gesteld in DC (Dublin Core). Bij de korte titelpresentatie heeft SRU de beschikking over de URL waarmee de data door de index-engine zijn opgehaald. Bij het opvragen van metadata gebruikt SRU deze URL (vgwvdkKey) om de metadata uit KB MDO op te vragen. Deze metadata zijn beschikbaar in DCX en eventuele andere formaten. Bij fulltext hoeven er geen metadata te zijn. Indien aan SRU wel om de metadata wordt gevraagd dan geeft SRU gewoon weer de metadata uit de index terug. Applicaties weten immers niet of er wel of geen KB MDO metadata zijn. SRU moet dat dus wel weten omdat anders de fulltext via het SRU protocol zou teruggegeven worden en dat zou door applicaties niet begrepen worden.

Applicaties zullen altijd het veld dc:identificer gebruiken om het object of de fulltext op te vragen. Indien er geen KB MDO metadata zijn moet dc:identificer dus als veld aanwezig zijn in de data zoals die aan de index worden aangeboden en die door SRU aan de client wordt aangeboden.

SRU moet dus weten welke metadata formaten beschikbaar zijn en omdat deze niet altijd uit KB MDO hoeven te komen (denk aan externe OAI resources) moet SRU ook de URL van de metadata kennen. Omdat deze URL niet gelijk hoeft te zijn aan de URL waarmee de index-engine data ophaalt wordt deze ook als apart veld in de index opgenomen. Dit is de SRUMetadataKey. Bij afwezigheid hiervan gebruikt SRU de vgwvdkKey. De beschikbare metadata formaten staan in het veld recordschemas.

Dus:

vgwvdkKey is voor Verity om de te indexeren data op te halen
dc:identificer dient voor applicaties om het bedoelde object op te halen
SRUMetadataKey dient voor SRU om de metadata op te vragen

MPEG21 en de resolver

De resolver dient om een identifier om te zetten naar een URL. Dit gaat in principe via een concordantietabel. Deze tabel is meestal aanwezig op het systeem waar de data staan (eDepot, SGD, AC etc.) en dan is het voor de resolver voldoende om de identifier te vertalen naar de URL voor het desbetreffende systeem. In andere gevallen is er een één-op-één relatie tussen identifier en filenaam. Voor collecties waarbij dat niet het geval is, moet de resolver een eigen concordantietabel bijhouden.

De mpeg21 dient om van objecten die uit meerdere componenten (files) bestaan de structuur te beschrijven en per component aan te geven, wat de URL van die component is en wat voor soort data in de betreffende component zitten (OCR, image etc.). In principe is dat alles maar er zijn soms redenen om meer gegevens in de mpeg21 op te nemen. Zoals bijv. de oorspronkelijke identifier van de componenten om de relatie met de bron te behouden.

Echter: stel dat men via de search engine een OCR file vindt. Hoe weet men dan dat die file een onderdeel van een samengesteld object is? Dit kan op meerdere manieren opgelost worden. De oplossing waar bij de KB voor gekozen is, is om het opvragen van die componenten via een "mpeg21-service" te laten verlopen met als parameters de identifier van de mpeg21 file (ook surrogaat of container genoemd) en een aantal parameters, die de positie en rol van die gevonden component binnen het object aangeven. Op deze manier blijft de context van een losse component binnen het samengestelde object behouden.

De mpeg21-service kan het complex object presenteren eventueel m.b.v. een stylesheet opgegeven door de client. De client hoeft dus niet perse zelf het complexe object te analyseren. Externe applicaties weten immers niet welk object model gebruikt wordt.

Er zijn vier mogelijkheden:

- 1) de client leest zelf de mpeg21 in
- 2) de client kent de URL-syntax en navigeert via de URL parameters (hiervoor is nog wel enige extra functionaliteit nodig, bijv. om het aantal componenten op te vragen)
- 3) de client laat de presentatie en navigatie aan de service over (indien er geen voorkennis is)
- 4) de client specificeert een stylesheet in de URL

Er is voor gekozen om de resolveerfunctionaliteit en de mpeg21 functionaliteit voor het opvragen van complexe objecten of onderdelen daarvan in één service te combineren: de resolver. De URL om onderdelen van een samengesteld object op te vragen kan immers ook als een persistente URL van de component gezien worden. NB. Door het opnemen van het request gedeelte van de URL voor iedere component in de mpeg21 container op te nemen hoeft de indexeerssoftware deze niet zelf te genereren.

De resolver kan objecten via een forward of een redirect aanbieden. Een redirect heeft vanwege de performance de voorkeur. Een forward is alleen nodig als de resolver gebruikt wordt om de autorisatie af te dwingen (in de toekomst mogelijk het eDepot).

Indexeren van onderdelen van samengestelde objecten

Bij het indexeren van onderdelen van onderdelen van een samengesteld object moet rekening gehouden worden met de functionaliteit die onder SRU besproken is. D.w.z. dat de URL die aan de index engine wordt aangeboden en de URL die als dc:identificer wordt gebruikt en de URL die voor de metadata wordt gebruikt alle drie anders kunnen zijn. Bijvoorbeeld bij het indexeren van losse pagina's van een document of boek kan men kiezen voor:

vgwvdkKey: wijst naar de file die geïndexeerd moet worden. Dit hoeft niet via de resolver

SRUMetadataKey: is afwezig of wijst naar eventuele metadata. In het geval van metadata moet dc:identificer in de metadata hetzelfde zijn als in het dc:identificer veld dat als veld aan de search-engine wordt meegegeven.

dc:identificer: wijst naar de resolver URL met de parameters die de positie van de component binnen het object aangeven.

Indien een volledig document of boek als één geheel geïndexeerd wordt:

vgwvdkKey: wijst naar een service die alle indexeerbare onderdelen opvraagt en gecombineerd als uitvoer biedt.

SRUMetadataKey: is afwezig of wijst naar eventuele metadata. In het geval van metadata moet dc:identificer in de metadata hetzelfde zijn als in het dc:identificer veld dat als veld aan de search-engine wordt meegegeven.

dc:identificer: wijst naar de resolver URL met de identificer van het complexe object als parameter, eventueel aangevuld met een parameter om de presentatie te regelen.

Voor het indexeren van complexe objecten wordt nu gebruik gemaakt van een harvester die de mpeg21 container opvraagt en een stylesheet waarmee deze container getransformeerd wordt naar een formaat dat geschikt is voor de index engine (in het geval van Verity zijn dat BIF-files). In dit stylesheet worden dus alle keuzes vastgelegd m.b.t. wat geïndexeerd wordt (vgwvdkKey), wat als object gezien wordt (dc:identificer) en wat als metadata (SRUMetadataKey) gezien wordt. Dit kan per collectie verschillend zijn. Door deze opzet zitten we niet meer vast aan MPEG21 maar kunnen ook andere datamodellen gebruikt worden. Uiteraard worden deze modellen (nog) niet door de resolver ondersteund. Dit mechanisme is echter wel bruikbaar voor externe complexe objecten waar een eigen lokale resolver bij hoort. Dit zal bijvoorbeeld bij TEL het geval zijn.