NPI Registreringsservice

Guide til Anvendere

Indhold

1 Introduktion

1.1 Formål

NPI Registreringsservice giver dokument-kildesystemer mulighed for at registrere metadata om dokumenter i henhold til IHE-standardens transaktioner ITI-42 og ITI-61.

Dette dokument beskriver anvendelsen af den webservice, der stilles til rådighed af NPI Registreringsservice.

Dokumentet giver en introduktion til det generelle webserviceinterface samt et eksempel på hvorledes den kan anvendes af et dokument-kildesystem.

1.2 Læsevejledning

Dette dokument henvender sig til udviklere og arkitekter der skal anvende NPI Registreringsservice. Forståelse af dette dokument lettes betydeligt, når læseren er fortrolig med IHE ITI-42 og ITI-61 beskrevet i henholdsvis [IHE3] og [IHE_ODD].

Det antages at læseren er bekendt med webservices samt brug af den gode web service (DGWS) og Security Token Service (STS).

Bemærk, at dokumentet er en kladde.

Kode-eksempler er forsøgt holdt korte, bl.a. ved at ofre normale ansvarsfordelinger og gode kode-skikke.

1.3 Dokumenthistorik

Version	Dato	Ansvarlig	Beskrivelse
0.7	11.1.2013	Systematic	Initiel udgave
0.9	27.2.2013	Systematic	Klargjort til afprøvning i demonstrationsprojektet.
0.9a	19.04.2013	Systematic	Udgave til Release Candidate 1

1.4 Definitioner og referencer

Definition	Beskrivelse
IHE	Integrating the Healthcare Enterprise
NPI	Nationalt Patient Indeks
NSI	National Sundheds-IT
NSP	Den nationale service platform (inden for sundheds-IT)
SHAK	Sygehusafdelingsklassifikation
SOR	Sundhedsvæsenets organisationsregister
STS	Security Token Service
XDS	Cross-Enterprise Document Sharing

Alias	Beskrivelse

Alias	Beskrivelse
DGWS 1.0	Den Gode Webservice 1.0
DGWS 1.0.1	Den Gode Webservice 1.0.1
IHE2	IHE Current Technical Framework – Revision 8.0, Vol. 2a: Transactions (ITI1-ITI28) http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_Rev8-0_Vol2a_FT_2011-08-19.pdf
IHE3	IHE Current Technical Framework – Revision 8.0, Vol. 2b: Transactions (ITI29-ITI64) http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_Rev8-0_Vol2b_FT_2011-08-19.pdf
IHE_ODD	IHE IT Infrastructure Technical Framework Supplement – On-Demand Documents – Revision 1.2, 2011-08-19 http://www.ihe.net/Technical_Framework/upload/IHE_ITI_Suppl_On_Demand_Documents_Rev1-2_TI_2011-08-19.pdf
Metadata	NPI Metadata Datamodel (SSE/11734/SDD/0001)
NPI opslag anvenderguide	NPI Service, Guide til anvendere (SSE/11734/PHB/0009)
NPI opslag ws	NPI Service snitfladebeskrivelse (SSE/11734/IFS/0001)
NPI Registreringsservice ws	NPI Registrering web service interface (SSE/11734/IFS/0004)

2 Anvendelse af NPI Registreringsservice

NPI Registreringsservice udstiller operationer til registrering af dokument-metadata i Det Nationale Patient Indeks via en webservice-snitflade beskrevet i [NPI Registreringsservice ws].

Registrering af dokument-metadata foretages under anvendelse af standarden IHE Cross-Enterprise Document Sharing (XDS), der er baseret på ebXML Registry Services (RS) og ebXML Registry Information Model (RIM).

Registreringsoperationen er i IHE XDS-termer IHE Technical Framework's transaktion 42, RegisterDocumentSet-b og transaktion 61, RegisterOnDemandDocumentEntry.

Med henblik på at være baseret på RegisterDocumentSet-b og RegisterOnDemandDocumentEntry i videst muligt omfang, er ændringer til IHE XDS-standardens WSDL minimeret. Derfor er payload til operationerne uændret.

Som beskrevet i [NPI Registreringsservice ws] forventer NPI Registreringsservice, at RegisterDocumentSet-B- og RegisterOnDemandDocumentEntry-kaldet indeholder ekstra headere i form af:

- En Security-header
- En Medcom-header

De kommer af Den Gode Webservice beskrevet i [DGWS 1.0] og [DGWS 1.0.1].

Headerne er tilføjet som implicitte headere, dvs. ved at være beskrevet i WSDL'ens bindings uden at være tilføjet til besked-strukturen for RegisterDocumentSet-b og RegisterOnDemandDocumentEntry.

2.1 Forberedelse til kald af NPI Registreringsservice

Inden kald af NPI Registreringsservice er det kildesystemets ansvar:

- at indhente et ID-kort på mindst sikkerhedsniveau 3 udstedt af STS

Bemærk, at CVR-nummer for organisationen, der anvender kildesystemet, skal være opført på NPI Registreringsservicens whitelist. Tilføjelse på whitelist skal aftales med NPI Registreringsservice systemejer.

2.2 Kald af NPI Registreringsservice

2.2.1 Kald af DocumentRegistry_RegisterDocumentSet-b

Ved kald af operationen DocumentRegistry_RegisterDocumentSet-b på NPI Registreringsservice, skal følgende gennemføres:

1. Der skabes en SubmitObjectsRequest (dvs. payload af DocumentRegistry_RegistryStoredQuery)
2. Security-header populeres med ID-kortet fra STS
3. Medcom-header skabes, herunder:
    a. Skal der tildeles unikt besked-id
    b. Skal der tildeles unikt session-id til Medcom-headerens flowID-element

2.2.2 Kald af DocumentRegistry_RegisterOnDemandDocumentEntry

For kald af operationen DocumentRegistry_RegisterOnDemandDocumentEntry på NPI Registreringsservice gennemføres samme skridt som ved kald af DocumentRegistry_RegisterDocumentSet-b.

3 Eksempler på kald af NPI Registreringsservice

Dette afsnit giver et eksempel på, hvordan denne service kan kaldes. Der er pt. ikke skrevet .NET-eksempelkode på at kalde NPI Service, men der er WSDL'er til rådighed, som man, vha. Visual Studio's webservice wizard kombineret med Seal.NET (se herunder), kan genere en web service consumer ud fra.

3.1 Kildesystem indhenter DGWS ID-kort fra STS

I det følgende skitseres, hvordan et kildesystem udviklet i Java kan benytte Seal.Java til at indhente et STS-signeret ID-kort.

Seal er et open source framework til understøttelse af "Den Gode Webservice", herunder integration til en central identitetsservice (IdP/STS), håndtering af føderationscertifikater mv.

Det findes i udgaver til Java og .Net.

Komponent	Reference
Seal.Java	http://digitaliser.dk/group/374971
Seal.NET	http://digitaliser.dk/group/375117

Indhentning af et STS-signeret ID-kort sker under anvendelse af et virksomhedscertifikat som skitseret ved følgende:

```
Properties properties = SignatureUtil.setupCryptoProviderForJVM();

// Her kan anvendes new SOSITestFederation(properties), hvis test-miljø anvendes
Federation federation = new SOSIFederation(properties);

File keystoreFile = … // read in the keystore file
String keystorePassword = … // obtain the password to the keystore file
CredentialVault vault = new FileBasedCredentialVault(
  properties,
  keystoreFile,
  keystorePassword);

SOSIFactory factory = new SOSIFactory(federation, vault, properties);

String flowId = … // Et unikt flow-id, der sættes i ID-kortet

SystemIDCard systemIDCard = factory.createNewSystemIDCard(
 …); // Her angives forskellige parametre til indhold i ID-kortet

SecurityTokenRequest securityTokenRequest = factory.createNewSecurityTokenRequest();
securityTokenRequest.setIDCard(systemIDCard);
SecurityTokenResponse response = … // Send securityTokenRequest til STS'en og deserialiser svaret
                                   // med factory.deserializeSecurityTokenResponse
…
IDCard signedIDCard = response.getIDCard();
```

Hermed er indhentet et ID-kort signeret af STS'en. I nedenstående eksempler er det antaget, at friske requests til kald af de respektive webservice-operationer er skabt; at det signerede ID-kort er kopieret over i den friske request; at øvrige DGWS-parametre sættes. Dette er i nedenstående beskrevet som en hjælpemetode getValidDGWSHeaders, som anvender antages at lave.

```
DGWSHeaderWrapper signedHeaders = getValidDGWSHeaders(flowId);
```

3.2 Kildesystem skaber SubmitObjectsRequest

I det følgende benyttes en hjælpeklasse SubmitObjectsRequestHelper beskrevet i afsnit 3.4 til skabelse af en SubmitObjectsRequest. SubmitObjectsRequest'en indeholder metadata for et on-demand-dokument, som kildesystemet kan stille til rådighed.

Bemærk, at nærværende kode, der viser princippene for registrering, ikke nødvendigvis bevirker registrering af alle de metadata-informationer, der er krævede. Hvilke metadata-informationer, der kan henholdsvis skal indgå i registrering, skal aftales mellem en dokumentkilde og NSI. Nærværende kode er tillige blot et eksempel på en af de måder IHE-standarden giver mulighed for at registrere metadata.

3.2.1 Skabelse af SubmitObjectsRequest

```
SubmitObjectsRequestHelper requestHelper = new SubmitObjectsRequestHelper();
SubmitObjectsRequest request = requestHelper.create();
```

3.2.2 Skabelse og tilføjelse af XDSDocumentEntry for on-demand dokument

```
// Unik identifikation af dokumentet, der skal erstattes af en konkret værdi.
String documentId = "urn:myorg:mydoc:doc:12345";
String repositoryUniqueId = "1.3.6.1.4.1.21367.2010.1.2.300.1";

ExtrinsicObjectType metadata = requestHelper.addOnDemandDocumentMetadataEntry(
        request
        documentId,
        repositoryUniqueId);
```

Med det følgende registreres, hvilken borger (patient), dokumentet omhandler samt hvilken institution/organisation, der har skabt dokumentet. Organisationen angives her ved brug af et yder-nummer 278467 for en patient med cpr-nummer 1122334455. Begge værdier er kodet/formateret som beskrevet i [Metadata].

```
String authorInstitution = "OrganisationsNavn^^^^^&2.16.208&ISO^^^^Yder=278467";
String patientId = "1122334455^^^&1.3.6.1.4.1.21367.2010.1.2.300&ISO";
requestHelper.addDocumentMetadataAttributes(metadata, authorInstitution, patientId);

// Tilføj øvrige metadata-information på lignende vis
```

Kunne organisationen i stedet identificeres ved SOR- eller SHAK-kode, skulle sidste del af authorInstitution være SOR=*værdi i SOR-klassifikationen* henholdsvis SHAK=*værdi i SHAK-klassifikationen*.

3.2.3 Skabelse og tilføjelse af XDSSubmissionSet

Som del af SubmitObjectsRequest'en skal der være en beskrivelse af den samlede tilføjelse i form af et XDSSubmissionSet. Dette skabes og tilføjes med:

```
RegistryPackageType submissionSet = requestHelper.addSubmissionSet(request);
requestHelper.addSubmissionSetAttributes(submissionSet, patientId);

// Tilføj øvrige metadata-information på lignende vis
```

Det tilføjede XDSDocumentEntry indgår i det skabte XDSSubmissionSet, hvorfor der til SubmitObjectsRequest'en skal tilføjes en association mellem XDSSubmissionSet og XDSDocumentEntry. Dette sker med:

```
requestHelper.addAssociation(request, metadata, submissionSet);
```

3.2.4 Eventuel skabelse og tilføjelse af erstatnings-associationer

Hvis de (nye) dokument-metadata beskrevet ovenfor skal erstatte eksisterende dokument-metadata, da skal der til SubmitObjectsRequest'en tilføjes en association som følger:

```
oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
```

```
AssociationType1 association = objFactory.createAssociationType1();
association.setId(UUID.randomUUID().toString());
association.setAssociationType("urn:ihe:iti:2007:AssociationType:RPLC");
association.setObjectType(
   "urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Association");
association.setSourceObject(metadata.getId()); // Id for nye metadata
association.setTargetObject(toReplaceId); // Id for metadata, der skal erstattes

// Tilføj erstatnings-associationen til request'en
request.getRegistryObjectList().getIdentifiable().add(objFactory.createAssociation(association));
```

Der skal genereres erstatnings-association for hvert enkelt eksisterende dokument-metadata-objekt, der skal erstattes.

Eksisterende dokument-metadata-objekter for et bestemt dokument-id kan fremfindes ved brug af søgning i svaret fra et NPI opslag. NPI opslag er beskrevet i [NPI opslag ws] og [NPI opslag anvenderguide].

3.3 Kildesystem kalder RegisterOnDemandDocumentEntry

I det følgende anvendes det ID-kort, der er indhentet som beskrevet i afsnit 3.1, til at kalde NPI Registreringsservicens operation RegisterOnDemandDocumentEntry. De metadata, der registreres, er skabt som beskrevet i afsnit 3.2.

Oprettelse af service og port:

```
URL wsdl = … // NPI Registreringsservicens.WSDL udpeges
DocumentRegistryService service = new DocumentRegistryService(wsdl,
        new QName("urn:ihe:iti:xds-b:2007", "DocumentRegistry_Service"));
DocumentRegistryPortType port = service.getDocumentRegistryPortSoap();
```

Herefter kan metoden på porten kaldes:

```
SubmitObjectsRequestHelper requestHelper = new SubmitObjectsRequestHelper();
SubmitObjectsRequest request = requestHelper.create();
… // request-strukturen populeres som beskrevet oven for

String flowId = XMLUtil.createNonce(); // Et unikt flow-id, der sættes i ID-kortet, skabes vha. SEAL
DGWSHeaderWrapper signedHeaders = getValidDGWSHeaders(flowId); // se ovenfor

RegistryResponseType response    =
  port.documentRegistryRegisterOnDemandDocumentEntry(
                request,
                signedHeaders.getSecurityHeader(),
                new Holder<Header>(signedHeaders.getMedcomHeader()));
```

3.4 Hjælpeklasse til skabelse af SubmitObjectsRequest

```
package dk.nsi.npiservice.ws;

import java.util.ArrayList;
```

```
import java.util.List;
import java.util.UUID;

import oasis.names.tc.ebxml_regrep.xsd.lcm._3.ObjectFactory;
import oasis.names.tc.ebxml_regrep.xsd.lcm._3.SubmitObjectsRequest;
import oasis.names.tc.ebxml_regrep.xsd.rim._3.AssociationType1;
import oasis.names.tc.ebxml_regrep.xsd.rim._3.ClassificationType;
import oasis.names.tc.ebxml_regrep.xsd.rim._3.ExternalIdentifierType;
import oasis.names.tc.ebxml_regrep.xsd.rim._3.ExtrinsicObjectType;
import oasis.names.tc.ebxml_regrep.xsd.rim._3.LocalizedStringType;
import oasis.names.tc.ebxml_regrep.xsd.rim._3.RegistryPackageType;
import oasis.names.tc.ebxml_regrep.xsd.rim._3.SlotType1;

public class SubmitObjectsRequestHelper {

    public SubmitObjectsRequest create() {
        ObjectFactory objFactory = new ObjectFactory();
        return objFactory.createSubmitObjectsRequest();
    }

    public ExtrinsicObjectType addOnDemandDocumentMetadataEntry(
            SubmitObjectsRequest request,
            String documentId,
            String repositoryUniqueId) {
        ExtrinsicObjectType onDemandDocumentMetadata =
                createOnDemandDocumentMetadataEntry(
                        UUID.randomUUID().toString(),
                        repositoryUniqueId);
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        if (request.getRegistryObjectList() == null) {
            request.setRegistryObjectList(objFactory.createRegistryObjectListType());
        }
        request.getRegistryObjectList().getIdentifiable().add(
                objFactory.createExtrinsicObject(onDemandDocumentMetadata));

        ExternalIdentifierType uniqueIdIdentifier =
                createUniqueIdExternalIdentifierType(documentId, onDemandDocumentMetadata);
        onDemandDocumentMetadata.getExternalIdentifier().add(uniqueIdIdentifier);

        return onDemandDocumentMetadata;
    }

    public void addDocumentMetadataAttributes(ExtrinsicObjectType documentMetadataEntry,
            String authorInstitution, String patientId) {
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        ClassificationType classification = objFactory .createClassificationType();
        classification.setId(UUID.randomUUID().toString());
        classification.setClassificationScheme("urn:uuid:93606bcf-9494-43ec-9b4e-a7748d1a838d");
        classification.setClassifiedObject(documentMetadataEntry.getId());
        SlotType1 authorInstitutionSlot = objFactory.createSlotType1();
        authorInstitutionSlot.setName("authorInstitution");
        authorInstitutionSlot.setValueList(objFactory.createValueListType());
```

```
        authorInstitutionSlot.getValueList().getValue().add(authorInstitution);
        classification.getSlot().add(authorInstitutionSlot);
        documentMetadataEntry.getClassification().add(classification);

        ExternalIdentifierType patientIdIdentifier = objFactory.createExternalIdentifierType();
        patientIdIdentifier.setId(UUID.randomUUID().toString());
        patientIdIdentifier.setIdentificationScheme("urn:uuid:58a6f841-87b3-4a3e-92fd-
a8ffeff98427");
        patientIdIdentifier.setRegistryObject(documentMetadataEntry.getId());
        patientIdIdentifier.setValue(patientId);
        documentMetadataEntry.getExternalIdentifier().add(patientIdIdentifier);
    }

    public RegistryPackageType addSubmissionSet(SubmitObjectsRequest request) {
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        RegistryPackageType submissionSetEntry = objFactory.createRegistryPackageType();
        submissionSetEntry.setId(UUID.randomUUID().toString());
        submissionSetEntry.setObjectType("urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:RegistryPackage");

    request.getRegistryObjectList().getIdentifiable().add(objFactory.createRegistryPackage(submissi
onSetEntry));

        // Classify registry package as XDSSubmissionSet (outside the RegistryPackage)
        ClassificationType classification = objFactory.createClassificationType();
        classification.setClassificationNode("urn:uuid:a54d6aa5-d40d-43f9-88c5-b4633d873bdd"); //
XDSSubmissionSet node value
        classification.setId(UUID.randomUUID().toString());
        classification.setObjectType("urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Classification");
        classification.setClassifiedObject(submissionSetEntry.getId());

    request.getRegistryObjectList().getIdentifiable().add(objFactory.createClassification(classification
));

        return submissionSetEntry;
    }

    public void addSubmissionSetAttributes(RegistryPackageType submissionSetEntry, String
patientId) {
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        ExternalIdentifierType patientIdIdentifier = objFactory.createExternalIdentifierType();
        patientIdIdentifier.setId(UUID.randomUUID().toString());
        patientIdIdentifier.setObjectType("urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExternalIdentifier");
        patientIdIdentifier.setIdentificationScheme("urn:uuid:6b5aea1a-874d-4603-a4bc-
96a0a7b38446");
        patientIdIdentifier.setRegistryObject(submissionSetEntry.getId());
        patientIdIdentifier.setValue(patientId);
        submissionSetEntry.getExternalIdentifier().add(patientIdIdentifier);
    }

    public AssociationType1 addAssociation(SubmitObjectsRequest request,
            ExtrinsicObjectType documentMetadataEntry,
            RegistryPackageType submissionSetEntry) {
```

```
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        AssociationType1 association = objFactory.createAssociationType1();
        association.setId(UUID.randomUUID().toString());
        association.setAssociationType("urn:oasis:names:tc:ebxml-
regrep:AssociationType:HasMember");
        association.setObjectType("urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Association");
        association.setSourceObject(submissionSetEntry.getId());
        association.setTargetObject(documentMetadataEntry.getId());

    request.getRegistryObjectList().getIdentifiable().add(objFactory.createAssociation(association));
        return association;
    }

    public List<AssociationType1> addAssociationsForReplacements(
            SubmitObjectsRequest request,
            ExtrinsicObjectType documentMetadataEntry,
            List<String> extrinsicObjectIdsReplaced) {
        List<AssociationType1> result = new ArrayList<AssociationType1>();
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        for (String extrinsicObjectId : extrinsicObjectIdsReplaced) {
            AssociationType1 association = objFactory.createAssociationType1();
            association.setId(UUID.randomUUID().toString());
            association.setAssociationType("urn:ihe:iti:2007:AssociationType:RPLC");
            association.setObjectType("urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Association");
            association.setSourceObject(documentMetadataEntry.getId());
            association.setTargetObject(extrinsicObjectId);

    request.getRegistryObjectList().getIdentifiable().add(objFactory.createAssociation(association));
            result.add(association);
        }
        return result;
    }

    private ExternalIdentifierType createUniqueIdExternalIdentifierType(
            String documentId, ExtrinsicObjectType metadataObjId) {
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        ExternalIdentifierType uniqueIdIdentifier = objFactory.createExternalIdentifierType();
        uniqueIdIdentifier.setId(UUID.randomUUID().toString());

    uniqueIdIdentifier.setIdentificationScheme(XDSConstantsHelper.XDS_EXTERNAL_IDENTIFIER_U
NIQUE_ID_IDENTIFICATION_SCHEME);
        uniqueIdIdentifier.setRegistryObject(metadataObjId.getId());
        uniqueIdIdentifier.setName(objFactory.createInternationalStringType());
        LocalizedStringType name = objFactory.createLocalizedStringType();
        name.setValue("XDSDocumentEntry.uniqueId");
        uniqueIdIdentifier.getName().getLocalizedString().add(name);
        uniqueIdIdentifier.setValue(documentId);
        return uniqueIdIdentifier;
    }

    private ExtrinsicObjectType createOnDemandDocumentMetadataEntry(
```

```
            String documentId,
            String repositoryUniqueId) {
        return createDocumentMetadataEntry(
                documentId,
                XDSConstantsHelper.XDS_ONDEMANDDOCUMENT_OBJ_TYPE,
                repositoryUniqueId);
    }

    private ExtrinsicObjectType createStableDocumentMetadataEntry(
            String documentId,
            String repositoryUniqueId) {
        return createDocumentMetadataEntry(
                documentId,
                XDSConstantsHelper.XDS_STABLEDOCUMENT_OBJ_TYPE,
                repositoryUniqueId);
    }

    private ExtrinsicObjectType createDocumentMetadataEntry(
            String documentId,
            String documentObjType,
            String repositoryUniqueId) {
        oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory objFactory =
                new oasis.names.tc.ebxml_regrep.xsd.rim._3.ObjectFactory();
        ExtrinsicObjectType extrinsicObjectType = objFactory.createExtrinsicObjectType();
        extrinsicObjectType.setId(documentId);
        extrinsicObjectType.setMimeType("text/xml");
        extrinsicObjectType.setObjectType(documentObjType);

        SlotType1 repositoryUniqueIdSlot = objFactory.createSlotType1();
        repositoryUniqueIdSlot.setName("repositoryUniqueId");
        repositoryUniqueIdSlot.setValueList(objFactory.createValueListType());
        repositoryUniqueIdSlot.getValueList().getValue().add(repositoryUniqueId);
        extrinsicObjectType.getSlot().add(repositoryUniqueIdSlot );
        return extrinsicObjectType;
    }
}
```

3.5 Klientgenerering ud fra WSDL

En webservice-klient kan genereres med udgangspunkt i NPI Service WSDL-filen, se [NPI Registreringsservice ws]. Det er ikke givet, at webservice-klientens proxy-klasser kan generes på baggrund af en deployed service (via ?wsdl-tilføjelse på servicen), da der ikke nødvendigvis er åbnet for sådanne kald.