Nationell Listningstjänst

Bas - Utvecklingsstöd

*skapad för:*

SVR

*skapad av:*

Robert Georén

Version: 1.1

Datum: 2009-09-15

| **Förändringar** | | |
| --- | --- | --- |
| **Vem** | **När** | **Vad** |
| Robert Georén | 2009-05-11 | Skapade dokumentet, version 1.0. |
| Robert Georén | 2009-09-15 | Ändrade i ’göra tjänsteval’ anropet, version 1.1. |
|  |  |  |

Innehåll

[1. Inledning 3](#_Toc240765771)

[2. Arkitektur/Design 3](#_Toc240765772)

[2.1 Introduktion till EN13606-1 3](#_Toc240765773)

[2.2 Nationella anvisningar 4](#_Toc240765774)

[3. Utvecklingsstöd för Anslutningspunkter - Java 4](#_Toc240765775)

[4. Utvecklingsstöd för Konsumenter - Java 5](#_Toc240765776)

[4.1 Generera proxy för tjänsten 5](#_Toc240765777)

[4.2 Hämta tjänsteval 5](#_Toc240765778)

[4.3 Hämta tillgängliga tjänsteutövare 8](#_Toc240765779)

[4.4 Göra tjänsteval 10](#_Toc240765780)

[5. Testning 12](#_Toc240765781)

[6. Referenser 12](#_Toc240765782)

# Inledning

Syftet med dokumentet är att underlätta för systemutvecklare som ska utveckla en Anslutningspunkt eller Konsument för den nationella listningstjänsten.

# Arkitektur/Design

## Introduktion till EN13606-1

En13606-1 är ett dataformat som används för att utbyta klinisk data mellan informationssystem rörande en patient. Dataformatet beskrivs nedan och de viktigaste begreppen är CONTENT och ITEM.

|  |
| --- |
|  |
| *\* Översiktsbeskrivning av EN13606 datastrukturen.* |

Domänmodell för Nationel listningstjänst realiseras i EN13606-1, alltså nedan domänmodell anpassas till ovan dataformat.

|  |
| --- |
| Informationsmodell.jpg |
| *\* Domänmodell för Nationell Listningstjänst.* |

## Nationella anvisningar

Det finns nationella anvisningar för hur nationella tjänster bör implementeras och en av dem är RIV (se ref[1]). RIVs syfte är bl.a. att beskriva hur man realiserar utbyte av information mellan två parter. Följande avsnitt beskriver vad en Producent kan behöva veta för att realisera en anslutningspunkt.

* + 1. **WSDL och RIV**

Det finns många regler för hur WSDL filerna ska utformas i enlighet med RIV Basic Profile 2.0. För att få en förståelse varför WSDL filerna ser ut som de gör, se referens [1]. Här följer en kortfattad summering av reglerna.

1. Regel: SOAP 1.1 ska användas.
2. Regel: SOAP meddelanden skall använda document/literal style.
3. Anpassning utökningsbarhet: EN13606 gör så att datatyperna blir utökningsbara. Detta eftersom ett CONTENT endast specifieras att innehålla ITEM och att det är i *meaningcode* som det definierar vad det är för data som ITEM innehåller. Alltså behövs inte <xsd:any> användas för att uppnå utökningsbarhet i EN13606.
   * 1. **“Contract-first Development”**

WSDL filen utgör kontraktet mellan Konsument och Producent och RIV ger riktlinjen att använda det designmönstret. Detta designmönster anses även som best-practice eftersom det skyddar Konsumenter ifrån kodförändringar i tjänsteimplementationen och skillnader som finns mellan verktygen som genererar WSDL filer ifrån kod.

* + 1. **Riktlinjer ifrån VIT Boken**

Nationella Listningstjänsten definieras som en *fråga-svar* tjänsteinteraktion enligt VIT-boken. Ett icke-funktionellt krav på producenter är att loggning av anropen till tjänsten ska göras, detta främst för att kunna mäta svarstider på tjänsten.

# Utvecklingsstöd för Anslutningspunkter - Java

Här följer några tips vid utveckling av en Anslutningspunkt.

* Generera server artifakter med ett verktyg så att de blir JSR 181 konformt (se ref[12]). wsgen ingår i JDK 1.6 men andra alternativ är t.ex. Apache CFX (ref [7]) eller Metro (ref [8]).
* Använd anvisningar i Meddelandestruktur (ref [5]) för att ta reda på vad som skall returneras till Konsumenter.

# Utvecklingsstöd för Konsumenter - Java

## 4.1 Generera proxy för tjänsten

Det rekommenderade sättet att använda en webservice av denna karaktär är att generera proxys för tjänsten och paketera proxy’na i en .jar fil. Använd ditt favorit verktyg för proxy generering (se ref [7] och [8]),

För att virtualiseringstjänsten ska veta vem som ska utföra en fråga så måste HSAID anges vid varje tjänsteinteraktion.

|  |
| --- |
| AttributedURIType logicalAddress = new AttributedURIType();  logicalAddress.setValue("SE239482390-23SAD"); |

## 4.2 Hämta tjänsteval

Nedan följer Java kod för användningsfallet “Hämta tjänsteval”.För att se XML datat som finns i SOAP Body, se ref [9].

|  |
| --- |
| **package** com.mawell.vval.consumer;  **import** java.net.MalformedURLException;  **import** java.net.URL;  **import** java.util.Iterator;  **import** java.util.List;  **import** javax.xml.namespace.QName;  **import** javax.xml.ws.Service;  **import** org.w3.\_2005.\_08.addressing.AttributedURIType;  **import** riv.carelisting.\_13606.v1.AD;  **import** riv.carelisting.\_13606.v1.ADXP;  **import** riv.carelisting.\_13606.v1.COMPOSITION;  **import** riv.carelisting.\_13606.v1.EHREXTRACT;  **import** riv.carelisting.\_13606.v1.ENXP;  **import** riv.carelisting.\_13606.v1.FUNCTIONALROLE;  **import** riv.carelisting.\_13606.v1.IDENTIFIEDENTITY;  **import** riv.carelisting.\_13606.v1.IDENTIFIEDHEALTHCAREPROFESSIONAL;  **import** riv.carelisting.\_13606.v1.ORGANISATION;  **import** riv.carelisting.\_13606.v1.TEL;  **import** riv.carelisting.\_13606.v1.TELEMAIL;  **import** riv.carelisting.\_13606.v1.TELPHONE;  **import** riv.crm.carelisting.getlisting.\_1.rivtabp20.GetListingResponderInterface;  **import** riv.crm.carelisting.getlisting.\_1.rivtabp20.PersonNotFound;  **import** riv.crm.carelisting.getlisting.\_1.rivtabp20.TechnicalException;  **import** riv.crm.carelisting.getlistingresponder.\_1.GetListingRequestType;  **import** riv.crm.carelisting.getlistingresponder.\_1.GetListingResponseType;  /\*\*  \* Exempelkod för användningsfallet "Hämta tjänsteval".  \*  \* **@author** Robert Siwerz, www.mawell.com.  \*/  **public** **class** UseCaseHamtaTjansteval  {  **private** **void** useCaseHamtaVardval() **throws** MalformedURLException  {  // Hämtar referens till SEI (Service Endpoint Interface).  QName serviceName = **new** QName("urn:riv:crm:carelisting:GetListing:1:rivtabp20", "GetListingResponderService");  Service service = Service.*create*(**new** URL("http://127.0.0.1:8088/mockGetListingResponderBinding?WSDL"), serviceName);  GetListingResponderInterface listingSEI = service.getPort(GetListingResponderInterface.**class**);  // Hämtar listningsinformation för angiven person.  AttributedURIType logicalAddress = **new** AttributedURIType();  logicalAddress.setValue("SE239482390-23SAD"); // HSAID till huvudmannen som skall svara på frågan.  GetListingRequestType request = **new** GetListingRequestType();  request.setSocialSecurityNumber("195005055005");  GetListingResponseType response = **null**;    **try**  {  response = listingSEI.getListing(logicalAddress, request);  } **catch** (PersonNotFound e)  {  // Använd affärsregel för att hantera detta.  } **catch** (TechnicalException e)  {  // Gör ett nytt försök...  }  // Skriver ut debug information  EHREXTRACT patientData = response.getEhrExtract();  // 1. Personnummer  System.*out*.println("1. Personens personnummer: " + patientData.getSubjectOfCare().getExtension());  // 2. Tjänsteutövarens (t.ex. Vårdenhetens) HSAID  String hsaID = "";  Iterator<COMPOSITION> allCompositions = patientData.getAllCompositions().iterator();  hsaID:  {  **while** (allCompositions.hasNext())  {  COMPOSITION composition = allCompositions.next();  // Hämtar Informationsmängden  // "Informationsmängd Vård- och omsorgstjänst".  **if** (composition.getMeaning().getCode().equals("vot"))  {  List<FUNCTIONALROLE> functionalRoles = composition.getOtherParticipations();  Iterator<FUNCTIONALROLE> functionalRoleIterator = functionalRoles.iterator();  // Hämtar roller med code='the', Tillhandahållande enhet.  **while** (functionalRoleIterator.hasNext())  {  FUNCTIONALROLE role = functionalRoleIterator.next();  **if** ("the".equals(role.getFunction().getCode()))  {  hsaID = role.getPerformer().getExtension();  **break** hsaID;  }  }  }  }  }  System.*out*.println("2. Tjänsteutövaren (HSAID): " + hsaID);  // 3. Hämtar mer detaljer om tjänsteutövaren.  List<IDENTIFIEDENTITY> identities = patientData.getDemographicExtract();  Iterator<IDENTIFIEDENTITY> identityIterator = identities.iterator();  **while** (identityIterator.hasNext())  {  IDENTIFIEDENTITY identity = identityIterator.next();  String currentHsaID = identity.getExtractId().getExtension();  // Ser efter om det är rätt HSAID på identifieraren, alltså rätt  // tjänsteutövare.  **if** (hsaID.equals(currentHsaID))  {  // Alternativ om tjänsteutövaren är en person eller en vårdenhet.  **if** (identity **instanceof** ORGANISATION)  {  // tjänsteutövaren är en vårdenhet och innehåller följande data.  // 1. Namn.  // 2. Telefon.  // 3. e-mail.  // 4. Postadress.  // 5. Geografisk plats.  System.*out*.println(" Vårdenhet:");  // \*\*\* Hämtar ut data \*\*\*  // 1. Namn  System.*out*.println(" 1. Namn: " + ((ORGANISATION) identity).getName().getValue());  // 2/3. Telefon och e-mail  Iterator<TEL> telecomIterator = identity.getTelecom().iterator();  **while** (telecomIterator.hasNext())  {  TEL tel = telecomIterator.next();  String value = tel.getValue();  **if** (tel **instanceof** TELPHONE)  {  System.*out*.println(" 2. Telefonnummer: " + value);  } **else** **if** (tel **instanceof** TELEMAIL)  {  System.*out*.println(" 3. E-mail: " + value);  }  }  // 4/5. postadress och greografisk plats.  Iterator<AD> addressesIterator = ((ORGANISATION) identity).getAddr().iterator();  **while** (addressesIterator.hasNext())  {  AD address = addressesIterator.next();  List<ADXP> allParts = address.getPartOrBrOrAddressLine();  ADXP part = allParts.get(0); // Ska bara vara en part.  **if** ("AL".equals(part.getType().value()))  {  System.*out*.println(" 4. Postadress: " + part.getContent());  } **else** **if** ("CEN".equals(part.getType().value()))  {  System.*out*.println(" 5. Geografisk plats: "  + part.getContent());  }  }  } **else** **if** (identity **instanceof** IDENTIFIEDHEALTHCAREPROFESSIONAL)  {  // Tjänsteutövaren är en specifik läkare och innehåller följande  // data.  // 1. Person id.  // 2. Namn.  // 3. Befattning.  System.*out*.println(" Läkare:");  // \*\*\* Hämtar ut data \*\*\*  // 1. Person id.  System.*out*.println(" 1. Personnummer: " + identity.getExtractId().getExtension());  // 2. Namn.  ENXP part = ((IDENTIFIEDHEALTHCAREPROFESSIONAL) identity).getName().getPart().get(0);  System.*out*.println(" 2. Namn: " + part.getValue());  // 3. Befattning.  String displayName = ((IDENTIFIEDHEALTHCAREPROFESSIONAL) identity).getProfession().getDisplayName().getValue();  System.*out*.println(" 3. Befattning: " + displayName);  }  }  }  }  /\*\*  \* Entry point i Java applikationen.  \*  \* **@param** args kommando-prompt argument.  \*/  **public** **static** **void** main(String[] args)  {  **try**  {  UseCaseHamtaTjansteval exempelkod = **new** UseCaseHamtaTjansteval();  exempelkod.useCaseHamtaVardval();  } **catch** (Exception e)  {  e.printStackTrace();  }  }  } |
| *\*Hämtar tjänsteval/vårdval för angiven person.* |

**Följande är en exempelutskrift när ovan program körs när personen är listad på en vårdenhet**

|  |
| --- |
| 1. Personens personnummer: 195005055005  2. Tjänsteutövaren (HSAID): SE2321000164-7381037591079  Vårdenhet:  1. Namn: Barn- och ungdomskliniken USÖ  2. Telefonnummer: tel:+46-46-789012  3. E-mail: mailto:enhet2@landstinget.se  4. Postadress: Universitetssjukhuset Örebro 701 85 ÖREBRO  5. Geografisk plats: Örebro |
| *\* Utskrift ifrån Java Console när programmet exekveras* |

**Följande är en exempelutskrift när ovan program körs när personen är listad på en läkare**

|  |
| --- |
| 1. Personens personnummer: 195005055005  2. Tjänsteutövaren (HSAID): SE2321000164-7381037590003ollvgiv-2  Läkare:  1. Personnummer: SE2321000164-7381037590003ollvgiv-2  2. Namn: Kalle Karlsson  3. Befattning: Vårdgivare |
| *\* Utskrift ifrån Java Console när programmet exekveras* |

## 4.3 Hämta tillgängliga tjänsteutövare

För att se XML datat som finns i SOAP Body, se ref [11].

|  |
| --- |
| **package** com.mawell.vval.consumer;  **import** java.net.MalformedURLException;  **import** java.net.URL;  **import** java.util.Iterator;  **import** java.util.List;  **import** javax.xml.namespace.QName;  **import** javax.xml.ws.Service;  **import** org.w3.\_2005.\_08.addressing.AttributedURIType;  **import** riv.carelisting.\_13606.v1.II;  **import** riv.crm.carelisting.getavailableserviceproviders.\_1.rivtabp20.GetAvailableServiceProvidersResponderInterface;  **import** riv.crm.carelisting.getavailableserviceproviders.\_1.rivtabp20.TechnicalException;  **import** riv.crm.carelisting.getavailableserviceprovidersresponder.\_1.GetAvailableServiceProvidersRequestType;  **import** riv.crm.carelisting.getavailableserviceprovidersresponder.\_1.GetAvailableServiceProvidersResponseType;  /\*\*  \* Exempelkod för användningsfallet "Hämta tillgänglia tjänsteutövare".  \* Tjänsteutövare kan vara en vårdenhet.  \* **@author** Robert Siwerz, www.mawell.com.  \*/  **public** **class** UseCaseHamtaTillgangligaTjansteutovare  {  /\*\*  \* Exempelkod för Use Case "Hämtar tillgängliga tjänsteutövare".  \*  \* **@author** Robert Siwerz,www.mawell.com.  \* **@throws** Fel vid kommunikation med tjänsten.  \*/  **public** **void** useCaseHamtaTillgangligaVardenheter() **throws** MalformedURLException  {  // Hämtar referens till SEI (Service Endpoint Interface).  QName serviceName = **new** QName("urn:riv:crm:carelisting:GetAvailableServiceProviders:1:rivtabp20", "GetAvailableServiceProvidersResponderService");  Service service = Service.*create*(**new** URL("http://127.0.0.1:8088/mockGetAvailableServiceProvidersResponderBinding?WSDL"), serviceName);  GetAvailableServiceProvidersResponderInterface listingSEI = service.getPort(GetAvailableServiceProvidersResponderInterface.**class**);    // Hämta tillgängliga tjänsteutövare.  AttributedURIType logicalAddress = **new** AttributedURIType();  logicalAddress.setValue("SE239482390-23SAD"); // HSAID till huvudmannen som skall svara på frågan.  GetAvailableServiceProvidersRequestType request = **new** GetAvailableServiceProvidersRequestType();  GetAvailableServiceProvidersResponseType response = **null**;    **try**  {  response = listingSEI.getAvailableServiceProviders(logicalAddress, request);  } **catch** (TechnicalException e)  {  // Gör ett nytt försök...  }    // Skriver att tillgängliga tjänsteutövare.  List<II> hsaIDs = response.getHSAId();  System.*out*.println("Tillgängliga tjänsteutövare:");  Iterator<II> hsaIterator = hsaIDs.iterator();  **while** (hsaIterator.hasNext())  {  System.*out*.println("HSAID: " + hsaIterator.next().getExtension());  }  }  /\*\*  \* Entry point i Java applikationen.  \*  \* **@param** args kommando-prompt argument.  \*/  **public** **static** **void** main(String[] args)  {  **try**  {  UseCaseHamtaTillgangligaTjansteutovare exempelkod = **new** UseCaseHamtaTillgangligaTjansteutovare();  exempelkod.useCaseHamtaTillgangligaVardenheter();  } **catch** (Exception e)  {  e.printStackTrace();  }  }  } |
| *\*Hämtar tillgänglia vårdenheter.* |

Exempel på programexekvering

|  |
| --- |
| Tillgängliga vårdgivare:  HSAID: SE2321000016-14J2  HSAID: SE2321000016-14KD  HSAID: SE2321000016-154L  HSAID: SE2321000016-14CD  HSAID: SE2321000016-1003  HSAID: SE2321000016-14FF |
| *\* Utskrift ifrån Java Console när programmet exekveras* |

## 4.4 Göra tjänsteval

För att se XML datat som finns i SOAP Body, se ref [10].

|  |  |
| --- | --- |
| **package** com.mawell.vval.consumer;  **import** java.net.MalformedURLException;  **import** java.net.URL;  **import** javax.xml.namespace.QName;  **import** javax.xml.ws.Service;  **import** org.w3.\_2005.\_08.addressing.AttributedURIType;  **import** riv.carelisting.\_13606.v1.COMPOSITION;  **import** riv.carelisting.\_13606.v1.EHREXTRACT;  **import** riv.carelisting.\_13606.v1.FUNCTIONALROLE;  **import** riv.carelisting.\_13606.v1.II;  **import** riv.crm.carelisting.createlisting.\_1.rivtabp20.CreateListingResponderInterface;  **import** riv.crm.carelisting.createlisting.\_1.rivtabp20.InvalidHSAID;  **import** riv.crm.carelisting.createlisting.\_1.rivtabp20.PersonNotFound;  **import** riv.crm.carelisting.createlisting.\_1.rivtabp20.TechnicalException;  **import** riv.crm.carelisting.createlistingresponder.\_1.CreateListingRequestType;  **import** riv.crm.carelisting.createlistingresponder.\_1.CreateListingResponseType;  /\*\*  \* Exempelkod för användningsfallet "Göra Tjänsteval".  \*  \* **@author** Robert Siwerz,www.mawell.com.  \*/  **public** **class** UseCasePerformAListing  {    **private** **void** useCasePerformASpecificListing() **throws** MalformedURLException  {  // Hämtar referens till SEI (Service Endpoint Interface).  QName serviceName = **new** QName("urn:riv:crm:carelisting:CreateListing:1:rivtabp20", "CreateListingResponderService");  Service service = Service.*create*(**new** URL("http://127.0.0.1:8088/mockCreateListingResponderBinding?WSDL"), serviceName);  CreateListingResponderInterface listingSEI = service.getPort(CreateListingResponderInterface.**class**);    **try**  {  AttributedURIType logicalAddress = **new** AttributedURIType();  logicalAddress.setValue("SE239482390-23SAD"); // HSAID till huvudmannen som skall svara på frågan.    // Skapar ett fråge objekt.  CreateListingRequestType request = **new** CreateListingRequestType();  EHREXTRACT ehrExtract = **new** EHREXTRACT();  // Lägger in personnummer  II personIdentification = **new** II();  personIdentification.setExtension("195005055005");  ehrExtract.setSubjectOfCare(personIdentification);  // Lägger in tjänsteutövaren  COMPOSITION performServiceChoice = **new** COMPOSITION();  FUNCTIONALROLE otherParticipations = **new** FUNCTIONALROLE();  II performer = **new** II();  performer.setExtension("SE345345-ASD323");  otherParticipations.setPerformer(performer);  performServiceChoice.getOtherParticipations().add(otherParticipations);    // Utför tjänstevalet.  CreateListingResponseType response = listingSEI.createListing(logicalAddress, request);    System.*out*.println("Affärsregel som kan förekomma vid listning: " + response.getBusinessRule());  } **catch**(PersonNotFound e)  {  System.*out*.println("Felkod:" + e.getFaultInfo().getCode());  } **catch** (InvalidHSAID e)  {  System.*out*.println("Felkod:" + e.getFaultInfo().getCode());  } **catch** (TechnicalException e)  {  // Gör ett nytt försök...  }  }    /\*\*  \* Entry point i Java applikationen.  \*  \* **@param** args kommando-prompt argument.  \*/  **public** **static** **void** main(String[] args)  {  **try**  {  UseCasePerformAListing exempelkod = **new** UseCasePerformAListing();  exempelkod.useCasePerformASpecificListing();  } **catch** (Exception e)  {  e.printStackTrace();  }  }  } | |
| *\* Göra ett vårdval för en specifik person och vårdenhet.* |

Exempel på programexekvering

|  |
| --- |
| Affärsregel som kan förekomma vid listning: Listningen kommer att gälla omgående. |
| *\* Utskrift ifrån Java Console när programmet exekveras* |

# Testning

För att säkerställa att Anslutningspunkten/Konsument fungerar som den skall så behöver tester göras som simulerar användningsfallen. Testerna kan skrivas för hand eller alternativt kan testverktyg används som simulerar användningsfallen. Ett sådant testverktyg är soapUI (se ref[4]).

# Referenser

[1] - RIV Tekniska anvisningar (Regelverk för Interoperabilitet inom Vård och omsorg): <http://rivta.forge.osor.eu/>

[2] EN13606-1: <http://www.chime.ucl.ac.uk/resources/CEN/EN13606-1/>

[3] – wsimport: <http://java.sun.com/javase/6/docs/technotes/tools/share/wsimport.html>

[4] – soapUI: <http://www.soapui.org/>

[5] VIT Boken: <http://arkitekturledningen.se/undermappar/links/VITstart.htm>

[6] Meddelandestruktur, Meddelandestruktur\_Visa\_vårdval\_Nationell\_Listningstjänst\_EN13606\_version\_0.1.xlsx

[7] Apache CFX: <http://cxf.apache.org/>

[8] Metro: <https://metro.dev.java.net/>

[9] Hämta vårdval: XML-struktur\_Visa\_vårdval\_version\_0.1.xml

[10] Göra vårdval: XML-struktur\_Göra\_vårdval\_version\_0.1.xml

[11] Visa tillgängliga tjänsteutövare: XML-struktur\_Visa\_möjliga\_tjänsteutövare\_0.1.xml

[12] JSR 181 Web Services Metadata: <http://jcp.org/en/jsr/detail?id=181>