Converting HL7v2.6 to FHIR
– One idea on how to perform it –

Mattes Rhein [1], Dr. Stefan Schlichting [2], PD Dr. Josef Ingenerf [3]

[1] Medizinische Informatik, Universität zu Lübeck, rhein@student.uni-luebeck.de
[2] Drägerwerk AG, stefan.schlichting@draeger.de
[3] Institute für Medizinische Informatik, Universität zu Lübeck, ingenerf@imi.uni-luebeck.de

Abstract

Because of the wide variety of medical messaging standards it is sometimes needed to convert one message with a particular standard to another standard. In this paper we describe an idea to write PCD-01(Patient Care Device) Messages on a FHIR Server. FHIR (Fast Healthcare Interoperability Resources) is a draft standard created by the HL7 (Health Level Seven International). To write the information of the PCD-01 message we need to convert the information to FHIR conform resources. We describe the way to convert a PCD-01 message segment to the FHIR resource on the hand of the Patient Identification (PID) segment which represents the patient FHIR resource.

1 Introduction

A central problem in medical communication standards is the diversity of standards, which is caused by variety of healthcare processes. Over the years the cost and complexity of implementing were increasing through the adding of more fields and optionality to the specification.
Fast Healthcare Interoperability Resources, or short FHIR, tries to solve this problems by defining a framework for extending and adapting the existing resources. FHIR tries to combine the best features from HL7's v2, HL7 v3 and CDA, Clinical and Administrative Domains. With use of modern web standards and a thight focus on implementability. To model healthcare data it uses basic building blocks, called resources. With this approach it is easier for healthcare providers to use and share the clinical data. Each resource can carry a human-readable text representation. For complex clinical information, where many systems using a simple text or document based approach, HTML is used as a fallback display option for the clinical safety [1].
The Personal Care Device, short PCD, domain is defined through the Integrating the Healthcare Enterprise initiative, short IHE. PCD deals with use cases in which at least on participant is a controlled patient-centric point-of-care medical device that wants to communicate with, at least, one other participant like a medical device or an information system[2].
In this paper we want to describe a way to convert incoming PCD-01 ORU R01 messages to FHIR conform resources to be written on a FHIR server.

2 Material and Methods

FHIR is a draft standard created by the Health Level Seven International, short HL7, health care organization. The used resources of FHIR can be combined to working systems that solve real world clinical and administrative problems. FHIR is useable in wide variety of context, like mobile phone apps, cloud communication or server communication in large institutional healthcare providers and many more[1]. In comparison over other HL7 standards, like HL7 v2 or HL7 v3, it offers many improvements. For Example:

- Strong focus on implementation
- Multiple implementation libaries
- Support for RESTful architectures
- Strong foundation in Web-Standards like XML, JSON, HTTP
- Human-readable wire format

Administrative concepts like patient, organization and device, as well as clinical concepts like medications and diagnostics are covered by FHIR with its resources.
PCD coordinates with other IHE clinical specialty based domains like medical imaging, to ensure consistency of medical device integration solutions across all IHE technical frameworks[3]. It aims to raise the bar from the expensive integration projects to an easy out of the box interoperability solution.[4].
The HIMSS, the Healthcare Information and Management Systems Society, defines interoperability as:
Interoperability means the ability of health information systems to work together within and across organizational


```
<Patient xmlns="http://hl7.org/fhir">
  <id value="glossy"/>
  <meta>
    <lastUpdated value="2014-11-13T11:41:00+11:00"/>
  </meta>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Henry Levin the 7th</p>
      <p>MRN: 123456. Male, 24-Sept 1932</p>
    </div>
  </text>
  <extension url="http://example.org/StructureDefinition/trials">
    <valueCode value="renal"/>
  </extension>
  <identifier>
    <use value="usual"/>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
      </coding>
    </type>
    <system value="http://www.goodhealth.org/identifiers/mrn"/>
    <value value="123456"/>
  </identifier>
  <active value="true"/>
  <name>
    <family value="Levin"/>
    <given value="Henry"/>
    <suffix value="The 7th"/>
  </name>
  <gender value="male"/>
  <birthDate value="1932-09-24"/>
  <careProvider>
    <reference value="Organization/2"/>
    <display value="Good Health Clinic"/>
  </careProvider>
</Patient>
```

Figure 1: Example Resource of FHIR [1]

boundaries in order to advance the effective delivery of healthcare for individuals and communities. [4].
The problem with this definition is that it is focus on health information exchange and not on medical devices. The AAMI, the Advancing Safety in Healthcare Technology, gives a definition more focused on medical devices:
Medical device interoperability is the ability of medical devices, clinical systems, or their components to communicate in order to safely fulfill an intended purpose.
In some point an interoperability is achieved, but in an expensive way. For example an expensive middleware is needed for integration and you have continuous support issues as device or software systems are updated and interfaces are lost, while the complexity of maintaining each part of the communication chain is increasing[4].

In this paper we are working with PCD-01 transactions of the IHE Patient Care Device Technical Framework. PCD-01 is used to transmit patient care device data between systems. The transaction is used by two Use-Case-Roles first the Device Observation Reporter and second the Device Observation Consumer. The name of these two roles are linked to an abstract function and not to physical devices. The Device Observation Reporter could be implemented in a freestanding system or in the Patient Care Device by itself. [5]
To convert the PCD-01 messages to FHIR resources, a mapping of the content of the message is needed. The content of a PCD-01 message is shown in figure 3.The content of the PCD-01 message is written in HL7v2.6 syntax which is defined in HL7 v2.6 Chapter 7 Observation Reporting, it also contains the coding requirements related to observation reporting used for PCD data communication.

As programming language we used Java 1.8 with the FHIR

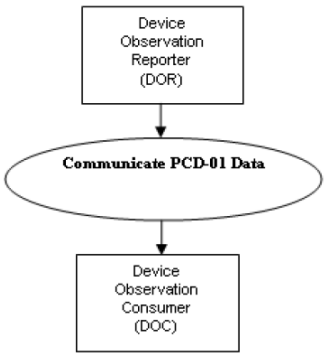

Figure 2: Use-Case-Roles in PCD-01 [5]

Segment	Meaning	Usage	Card	HL7 chapter
MSH	Message Header	R	[1..1]	2
[{SFT}]	Software Segment	X	[0..0]	2
[UAC]	User Authentication Credential	O	[0..1]	
{	--- PATIENT_RESULT begin			
[	--- PATIENT begin			
PID	Patient Identification	R	[1..1]	3
[PD1]	Additional Demographics	X	[0..0]	3
..[{NTE}]	Notes and Comments	X	[0 0]	2
..[{NK1}]	Next of Kin/Associated Parties	O	[0..3]	3
[	--- VISIT begin			
PV1	Patient Visit	R	[1..1]	3
[PV2]	Patient Visit – Additional Info	X	[0..0]	3
]	--- VISIT end			
]	--- PATIENT end			
{	---ORDER_OBSERVATION begin			
[ORC]	Order Common	X	[0..0]	4
OBR	Observation Request	R	[1..1]	7
[{NTE}]	Notes and Comments	O	[0..1]	2
[{	--- TIMING_QTY begin			
TQ1	Timing/Quantity	R	[1..1]	4
[{TQ2}]	Timing/Quantity Order Sequence	X	[0..0]	4
{]	--- TIMING_QTY end			
[CTD]	Contact Data	X	[0..0]	11
[{	--- OBSERVATION begin			
OBX	Observation Result	R	[1..1]	7
[{NTE}]	Notes and comments	O	[0..1]	2
}]	--- OBSERVATION end			
[{FT1}]	Financial Transaction	X	[0..0]	6
[{CTI}]	Clinical Trial Identification	X	[0..0]	7
[{	--- SPECIMEN begin			
SPM	Specimen	X	[0..0]	7
[{OBX}]	Observation related to Specimen	X	[0..0]	7
}]	--- SPECIMEN end			
}	--- ORDER_OBSERVATION end			
}	--- PATIENT_RESULT end			
[DSC]	Continuation Pointer	X	[0..0]	2

Figure 3: Content of PCD-01 [5]

API, called HAPI-FHIR. As development environment we used Eclipse Mars. The server which was used for testing is a Spark Server from Furore and runs on a Microsoft Windows Server 2012 R2 system which is running in a virtual machine. It is written in C# and is build for FHIR DSTU 2 [6].

To map the data of the message to FHIR resources we used the standard mapping which is given by the HL7.
The resources on the FHIR server are saved under URL addresses which also represent the logical identifier of the resource. For example the resource for a test patient could be found under "baseserver\fhir\patient\id". This id is set

PCD Segment	FHIR Resource
MSH	messageheader
PID	patient
PV1	encounter
OBR	Observation, device, devicecomponent, devicemetric (depending on the segmentpart)
OBX	Observation, device, devicecomponent, devicemetric (depending on the segmentpart)

Table 1: Short mapping overview of PCD-01 elements on FHIR Resources

by the server and is unique on the server system.
On the example of the PID segment of the PCD data we are showing how we established the converter. The PID segment stands for patient identification and contains all necessary information about the patient, like name, address, gender, birth-date, patient id and many others, see IHE PCD Technical Framework Volume 2 for the complete list of content [5].
In this paper we will exemplary concentrate on the mapping for the patient. The patient id will show how we got the information out of the message and afterwards we will show how we build the patient resource which will be written on the server. The patient id is converted from the

PCD-01 PID Segment content	datatype
Patient ID	CX(Extended Composite ID with Check Digit)
Patient Name	XPN(Extended Person Name)
Date/Time of Birth	DTM(Date and Time)
Administrative Sex (Gender)	IS(Coded values for user-defined tables)
Patient Address	XAD(Extended Address)

Table 2: Datatypes of the diffrent contents

CX[] datatype to the identifier type of the FHIR resource. This identifier represents the business identifier which can be set by the software system.

The method "getFirstPatientID()" shows how to get the Patient ID out of the received message. The ID is contained in the first part of the PID segment. It is stored in an "Extended Composite ID" which itself is parted in ten parts.

- ID Number
- Identifier Check Digit
- Check Digit Scheme
- Assigning Authority
- Identifier Type Code
- Assigning Facility
- Effective Date
- Expiration Date
- Assigning Jurisdiction
- Assigning Agency or Department

The ID which we want to use is stored in the first part of the CX[] datatype, the ID Number. The ID Number itself is stored in a String Data element, therefore the best way to use it further is to store it in a String.

```
public String getFirstPatientId(ORU_R01 theMessage){
        String patientIdString=null;
                if (theMessage!=null)
                {
                        ORU_R01_PATIENT_RESULT patient_RESULT = theMessage
                                        .getPATIENT_RESULT();

                        CX[] patientIDs = patient_RESULT
                                                .getPATIENT().getPID()
                                                .getPatientIdentifierList();

                        if (patientIDs!=null && patientIDs.length>0)
                        {
                                patientIdString=patientIDs[0].getCx1_IDNumber()
                                                .getValue();
                        }
                }
        return patientIdString;
        }
```

After storing the ID in the String "patientIdString" we need to get the "ID System" for the Identifier. The ID System describes the namespace for the identifier. The ID System itself is build with the "Assigning Authority" and the "ID Type", here is the Example for getting the "Assigning Authority".

```
public String getAssigningAuth (ORU_R01 theMessage) {
        String AssignAuth = "";
        if (theMessage != null) {
                ORU_R01_PATIENT_RESULT patient_RESULT = theMessage
                                .getPATIENT_RESULT();
                CX[] patientAssignAuth = patient_RESULT.getPATIENT()
                                .getPID().getPatientIdentifierList();

                if (patientAssignAuth!=null
                                && patientAssignAuth.length>0)
                        {
                        AssignAuth=patientAssignAuth[0]
                        .getAssigningAuthority().getHd1_NamespaceID()
                        .getValueOrEmpty();

                        }
                }
        return AssignAuth;
        }
```

After getting every information of the patient out of the message we are building the patient resource for the fhir server.

```
public Patient setup_patient () throws Exception {

        FileImportTest message = new FileImportTest();
        ORU_R01 oru_message = message.testReadFromFile();
        System.out.println("Setting up Patient");
        String patient_given_name = new ORUHandler()
                        .getPatientGivenName(oru_message);
        String patient_last_name = new ORUHandler()
                        .getPatientLastName(oru_message);
        Date patient_birthdate = new ORUHandler()
                        .getPatientBirthdate(oru_message).getValueAsDate();
        String patient_id = new ORUHandler()
                        .getFirstPatientId(oru_message);
        String patient_address_street = new ORUHandler()
                        .getPatientStreet(oru_message);
        String patient_address_city = new ORUHandler()
                        .getPatientCity(oru_message);
        String patient_address_state = new ORUHandler()
                        .getPatientState(oru_message);
        String patient_address_zipcode = new ORUHandler()
                        .getPatientZipCode(oru_message);
        String patient_address_country = new ORUHandler()
                        .getPatientCountry(oru_message);
        String patient_gender = new ORUHandler()
                        .getPatientGender(oru_message);
        String patient_system = new ORUHandler()
                        .getAssigningAuth(oru_message);
        String patient_id_type = new ORUHandler()
                        .getPatientIdType(oru_message);

        Patient patient = new Patient();
        IdentifierDt patId=new IdentifierDt(createPatientSystemURI
                        (patient_system,patient_id_type), patient_id);
```

```
        patient.addIdentifier(patId);

        patient.addAddress().setCity(patient_address_city)
                            .setCountry(patient_address_country)
                            .setState(patient_address_state)
                            .setPostalCode(patient_address_zipcode)
                            .addLine(patient_address_street);

        patient.addName().addFamily(patient_last_name)
                            .addGiven(patient_given_name);

        patient.setBirthDateWithDayPrecision(patient_birthdate);

        BoundCodeDt<AdministrativeGenderEnum> genderElement
                        = patient.getGenderElement();

        genderElement.setValueAsEnum
                        (getGenderEnumValueFromString(patient_gender));

        return patient;
        }
```

After setting every needed information of the resource we are checking if a resource with the same id is already been written on the server, if so the resource is going to be updated with the new values, if not the patient resource is written on the server. The update work like the create shown below, the difference is that the update is getting the logical id for the resource which has the patient id as business identifier to work with.

```
public MethodOutcome createPatient () throws Exception{

        // Create a DSTU2 context, which will use DSTU2 semantics
        FhirContext ctx = FhirContext.forDstu2();

        // This client supports DSTU2
        String ServerBase = "http://localhost:49734/fhir";
        client = ctx.newRestfulGenericClient(ServerBase);

        Patient patient = new CreatePatientFromMessage()
                        .setup_patient();

        MethodOutcome outcome = null;

        List<IdentifierDt> matchingID = patient.getIdentifier();
        List<BaseIdentifierDt> matchingIdsList=
                        new ArrayList<BaseIdentifierDt>(matchingID.size());
                        matchingIdsList.addAll(matchingID);

        TestExistence existance = new TestExistence();
        boolean existancebool = existance.testExistens(matchingIdsList);

        if(existancebool == true) {
                try {
                        MethodOutcome updatePatient = new UpdatePatient()
                                        .updatePatient(patient);

                        return updatePatient;
                } catch (InvalidRequestException invalidRequest){
                        System.err.println
                                ("400_Bad_Request_for_Update");

                        return outcome;
                }

        } else if(existancebool==false){
                System.out.println
                        ("Entry_not_on_Server_creating_entry_"+patient);

                outcome = client.create()
                                .resource(patient)
                                .execute();
                return outcome;
        }
        return outcome;
}
```

The code fragment above shows how a resource is created on the a server which is running on the localhost. If there are problems with the id, which is needed for the update request, the server will give back the HTTP "400 Bad Request" status code. The creation of other resource is in principle the nearly the same. The differences for the other resources are the content in these resources. For example can a device be build with device components and device metrics, which represents measurements or changes in the status of component. The components describe parts of a medical device, like sensors or pumps. Metrics and components are resources in FHIR which contain specific data and references to other metrics, components and devices.

3 Conclusion

Thanks to the implementation guides and documentation of FHIR, PCD and HL7v2.6 the mapping of the information given in the PCD messages to the FHIR resources, are not that complicated. The bigger problem is that FHIR is still in an early development phase therefore not everything went as we expected it to work. At some points we needed to build a work-around for problems which could not be solved the way we thought it would. During the development we found a bug in the specifications of FHIR. Hopefully this bug will be fixed in the next version of FHIR.

Acknowledgement

The work has been carried out at Drägerwerk AG, Moislinger Allee 53, 23558 Lübeck. Under the supervision of Dr. Stefan Schlichting.

4 References

[1] HL7, "Introducing fhir." http://hl7.org/fhir/summary.html, Janurary 2015.

[2] IHE, *IHE Patient Care Device(PCD) Technical Framework Volume 1 IHE PCD TF-1 Profiles*, 2014.

[3] IHE, "Ihe webinar on patient care device," 2011.

[4] P. Paul Schluter, "Understanding interoperability with the ihe profiles," 2012.

[5] IHE, *IHE Patient Care Device(PCD) Technical Framework Volume 2 IHE PCD TF-2 Transactions*, 2014.

[6] Furore, "Spark fhir server." http://spark.furore.com/, Janurary 2016.