OpenHealth

A Framework for the Delivery of CCR-based Services

October 24, 2006

TABLE OF CONTENTS

1. Introduction

The AAFP’s Center for Health Information Technology site states that, “The ASTM Continuity of Care Record standard (CCR[1]) was developed in response to the need to organize and make transportable a set of basic information about a patient's health care that is accessible to clinicians and patients.”[2]

The purpose of this design is to leverage a variety of widely accepted protocols, techniques and design principles, and couple them with the use of CCR as a *lingua franca* to represent patient medical information, and create a practical, easy to use service infrastructure.

[1] http://www.astm.org/cgi-bin/SoftCart.exe/DATABASE.CART/WORKITEMS/WK4363.htm?L+mystore+oaob9148
[2] http://www.centerforhit.org/x201.xml

This design elevates CCR to becoming the lingua franca used in the delivery of patient-centric services.

Provider-facing solutions and infrastructures (practice management systems, electronic health record systems, and RHIOs) can benefit from a standardized approach to service interoperability. By standardizing the mechanisms that support the exchange of clinically-based content, this framework creates a unique opportunity for government and medical associations to disseminate evidence-based, quality information to providers, in a way that seamlessly and directly integrates to familiar tools. This level of standardization greatly increases the benefits of federally funded efforts to affect quality of health care, requiring little to no effort interpreting, implementing and integrating recommendations into proprietary systems.

It is expected that the ubiquitous use of CCR and the availability of patient-centric CCR-based services might act as an unprecedented agent of consumer empowerment, incenting patients to take on a more active role in their overall health care management.

Finally, from a technical perspective, this design builds upon SOA[3]-enabling technologies, protocols and best practices. Service Oriented Architectures encourage agile software development by supporting the creation of composite applications based on the use of loosely coupled and interoperable services. The discovery and use of SOA services is typically the work of software architects, designers and engineers. The OpenHealth Services (OHS) framework builds on SOA principles and enhances its applicability by supporting the discovery and use of services by non-technical individuals. OHS in effect commoditizes SOA, allowing health care providers and patients direct access to web-enabled services.

2. Goals and Strategies

The overarching design principles for this service delivery framework are intended to encourage and support early adoption, while enabling services to evolve into more intricate and elaborate constructs, as the market dictates the need.

A careful balance of these requirements has lead to a design that reduces barrier to entry (for instance, by minimizing complexity and cost of development, and by fostering the reuse of existing solutions), while supporting flexibility and extensibility.

The design takes into account these considerations on behalf of service providers, client solutions and human end-users.

2.1 Major Goals

Major goals focus on lowering barrier to adoption while supporting flexibility.

[3] http://en.wikipedia.org/wiki/Service-oriented_architecture

- Make it simple for users (providers and patients-alike) to use;
- Make it simple and inexpensive to create services;
- Make it simple and inexpensive to create service-aware clients;
- Create a system that has enough designed-in flexibility to support and even promote experimentation, without being so flexible as to make interoperability a challenge;
- Support a maximal number of user-agent configurations, transparently, including support for web-hosted and web-enabled applications as well as potentially disconnected PDA, cell phone and desktops;
- A user must be able to 'try' a new OpenHealth Service without *ANY* technical computer knowledge beyond what can be expected of a typical consumer.

In terms of architectural quality attributes[4], this framework has been designed with the intent of supporting and fostering:

- Limited *cost* of implementation,
- Low *complexity*,
- *Flexibility*,
- *Interoperability*, and
- *Usability*.

2.2 Minor Goals

Minor goals of the framework design are intended to encourage the development of business models that support the framework's adoption by vendors and the use of compliant tools by patients, providers, etc.

- Services should be *combinable* – so that an intelligent user-agent – for example – can combine them to gain effects far greater than what one or two services could provide individually (composite application[5]),
- Services should enable *metering* – to provide a profit motive for the provision of services,
- Services should be capable of supporting *anonymous* use (if desired).

2.3 Design Strategies

- Leverage existing standards, in particular SOA-related standards and best practices, such as:
    - WS-I Basic Profile[6]
    - WSDL[7]
    - UDDI[8]

[4] http://en.wikipedia.org/wiki/ISO_9126
[5] http://looselycoupled.com/glossary/composite%20application
[6] http://www.ws-i.org/
[7] http://www.w3.org/TR/wsdl

    - SOAP[9]
    - WS-Security[10]
    - Orchestration (BPEL)[11]
    - etc
- The KISS[12] principle,
- Define basic **core** functionality and decide which advanced features could be layered on, as extensions[13].

3. Design Overview

The OpenHealth Framework formalizes mechanisms that support the delivery of services to CCR-enabled clients.

3.1 Working Concepts and Terminology

The OpenHealth Framework is described in terms of the following concepts (described below):

- CCR-enabled Client
- OpenHealth Service
- OpenHealth Service Definition
- Registering OpenHealth Services
- Local OpenHealth Service
- Remote OpenHealth Service
- Hybrid OpenHealth Service

3.1.1 CCR-enabled Client

A ***CCR-enabled client*** will likely be a personal health record (PHR) software product or an electronic medical record (EMR) that is capable of interpreting (importing) and generating (exporting) a patient's clinical history summary, using the ASTM CCR standard.

Examples of CCR-enabled clients include desktop applications, web-hosted applications, cell-phone applications, PDA applications, etc.

CCR-enabled clients may operate in a state that has access to the internet (aka *connected*) or in a state that is not connected to the internet (aka *disconnected*).

The OpenHealth Services Framework is designed to work with all types of CCR-enabled clients and it supports implementation of both *connected* and *disconnected* services.

[8] http://www.uddi.org/
[9] http://www.w3.org/TR/soap/
[10] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
[11] http://en.wikipedia.org/wiki/Business_Process_Execution_Language
[12] http://en.wikipedia.org/wiki/KISS_Principle
[13] See Appendix I

CCR-enabled clients that are additionally capable of using OpenHealth Services are known as *OpenHealth Service Clients*.

3.1.2 OpenHealth Service

An ***OpenHealth Service*** is, broadly speaking, a computerized system, designed to deliver specific health-related functionality. Prototypically, service requests will take the form of an input CCR, and outputs will likely be an HTML or PDF document.

Examples of potential OpenHealth Services include, but are not limited to:

- Patient health risk assessment,
- Getting quotes for HSA/HDHP (Health Savings Accounts/High-Deductible Health Plan) products,
- Access to applicable clinical trials,
- Locating Medicare suppliers (contact information, maps and directions),
- Anonymous search for addiction counseling and support groups,
- Integrating wellness management with healthcare – exercising journals[14],
- Merging and reconciling CCR documents obtained from different sources,
- Normalizing CCR coding,
- Translation into/from other file formats, e.g. X12-837, CCD and CDA,
- Locating specialists,
- “Generics” medication substitution, etc.

An OpenHealth Service can be ***Local***, ***Remote*** or ***Hybrid*** (more on this later). Service locality is transparent to the CCR-enabled client – though latency will be apparent to remote system users.

Services can be logically associated with categories, creating a standardized taxonomy. Examples of categories including perhaps ‘common forms’, ‘risk assessment and decision support’, ‘anonymization services’, etc.

3.1.3 OpenHealth Service Definition (OHSD)

An ***OpenHealth Service Definition*** is a very simple XML-based format that (together with this specification) defines **all** the details needed to access an OpenHealth service (be it local or remote).

The structure and content of an OpenHealth Service Definition (OHSD) are detailed later[15] in this document. At a high level, an OHSD contains the name and description of the service, authorship and copyright information, as well as the technical references for invoking the service (URL, script(s), parameter definitions, etc).

[14] http://bimactive.com/ba/ui/land_main.php
[15] See section 3.2.3 Service Definition

A primary purpose of the OHSD is to support the **commoditization** of CCR-enabled services. OHSD act as proxies for the services themselves, in a sense that non-technical individuals can identify with. Users interact with OHSD as commodities that can be downloaded, emailed, purchased and that can plug into familiar applications.

Intelligent user-agents will be capable of supporting the discovery, concatenation and use of OpenHealth services by non-technical individuals, unfamiliar with SOA technologies.

Additionally OHSD supports the registration and use of OpenHealth Service clients in *connected* and *disconnected* states, **transparently**. The OHSD has the capability of supporting a service in such a way that some capability is enabled in disconnected state, supporting graceful degradation of service.

It should be noted that OHSD files, while useful in the engagement of non-technical users, are not required for programmatic use of the web services capabilities of the framework. SOA developers can use OpenHealth web services utilizing conventional techniques and technologies, such as UDDI.

All of the subsequent sections will refer to the use of OHSD in the registration and use of OpenHealth services. If desired, the use of OHSD can be replaced with the URL[16] of the web service. Note, however, that this technique works for *remote* services but would not work for *local* services.

3.1.4 Registering OpenHealth Services ("Plug-In")

The OpenHealth Service framework enables the creation of a market where multiple service providers can publish one or multiple OpenHealth Services. Published services can be either implemented as local or remote and provided for free or in return for payment.

When a user decides to use a service published by a particular provider, he/she will download the desired OpenHealth Service Definition (OHSD) file. This file will be used to register the service with the OpenHealth Service client (plug-in).

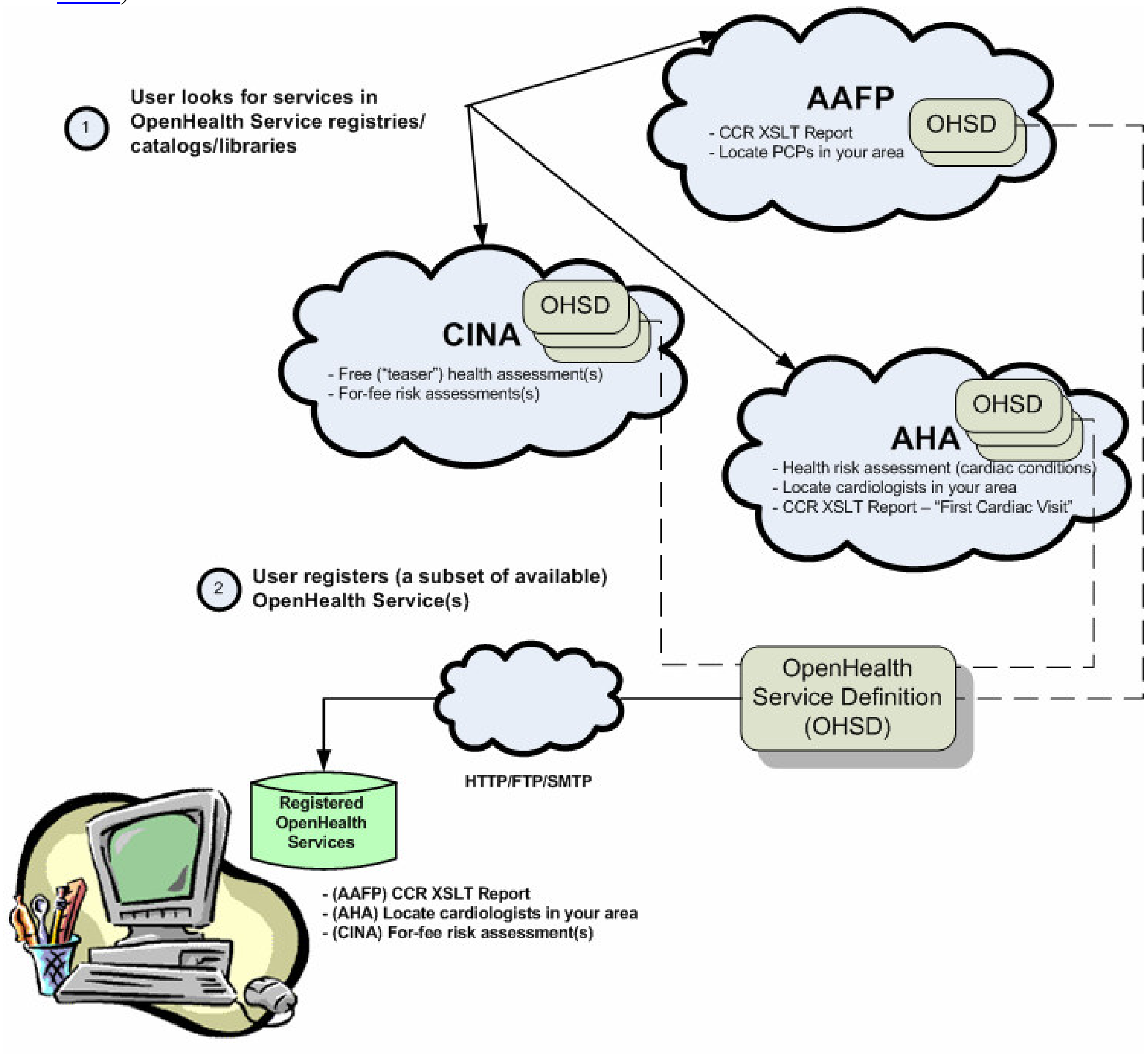
The specific manner in which service registration takes place is client-specific. In desktop clients, double-clicking, dragging-and-dropping might be supported gestures of the client application. Web-based clients (and also desktop) may support an interactive presentation that allows the user to select the service (with or without direct user access to OHSD file).

Additionally, it is also possible for OpenHealth Services to be automatically discovered and registered on behalf of the user. In such cases, the client application can communicate with service registries directly and apply appropriate rules to decide on specific services to plug in. If the client application is provided by specific groups (such

[16] http://en.wikipedia.org/wiki/URL

as employers, insurers, providers, etc) it is likely that automated service registration will keep users 'up-to-date' with service offerings derived as a benefit of their associations.

Besides the manner in which a user interacts with the service registration, storage of OHSD registration information is also client-specific. Clients (desktop, web-hosted, cell phone and PDA applications) are likely to implement a database of registered OHSD which can later be accessed in support of service execution (as shown in sections 3.1.5 and 3.1.6).


1
User looks for services in OpenHealth Service registries/ catalogs/libraries
AAFP
- CCR XSLT Report
- Locate PCPs in your area
OHSD
CINA
OHSD
- Free ("teaser") health assessment(s)
- For-fee risk assessments(s)
AHA
OHSD
- Health risk assessment (cardiac conditions)
- Locate cardiologists in your area
- CCR XSLT Report – "First Cardiac Visit"
2
User registers (a subset of available) OpenHealth Service(s)
OpenHealth Service Definition (OHSD)
HTTP/FTP/SMTP
Registered OpenHealth Services
- (AAFP) CCR XSLT Report
- (AHA) Locate cardiologists in your area
- (CINA) For-fee risk assessment(s)

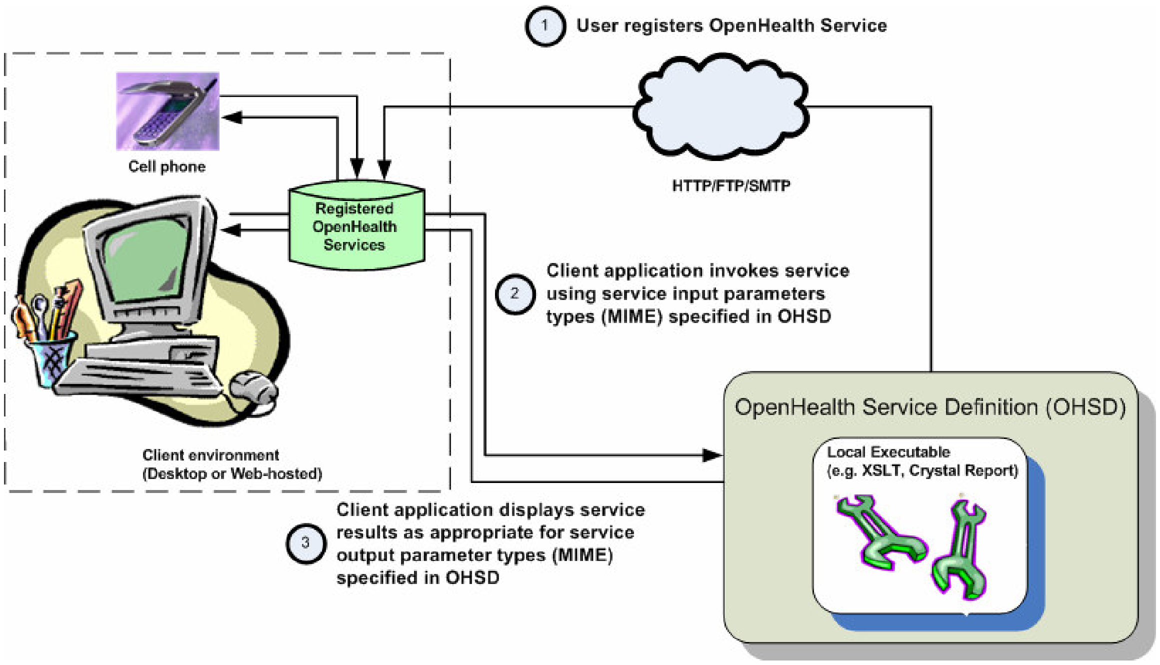
In the picture above, the user has registered three services from three different service providers. One of the registered services is local (AAFP's CCR XSLT Report) and the other two are remote.

When a service is registered with the client application, it is not only available for use (in a manner that is integrated with the user's environment) but the service is automatically grouped with services of similar category(ies).

3.1.5 Local OpenHealth Services

A ***Local OpenHealth Service*** is one that runs in the client's computer or device.

An example of this might be an XSLT script that accepts as input a CCR and generates humanly-readable HTML. This could be used for instance in support of local generation of pre-filled (with the individual's data), standardized forms – e.g. Boy Scouts summer camp health forms[17]. Another example might be a small diabetes management script loaded on the user's cell phone or PDA, that alerts the user that the keyed-in glucose measurements are high and may require the patient to contact his/her doctor.


1 User registers OpenHealth Service
Cell phone
HTTP/FTP/SMTP
Registered OpenHealth Services
2 Client application invokes service using service input parameters types (MIME) specified in OHSD
OpenHealth Service Definition (OHSD)
Local Executable (e.g. XSLT, Crystal Report)
Client environment (Desktop or Web-hosted)
3 Client application displays service results as appropriate for service output parameter types (MIME) specified in OHSD

The picture above illustrates a user who registered OpenHealth services, as described in section 3.1.4. The service, being local, includes enough executable information to run locally on the client machine (desktop, cell phone, PDA, etc). When the user selects to execute the service, the client application locates the service definition in its database of registered services and reads the local execution information (script, XSLT transform, etc). The local executable is run and the results are presented to the user.

3.1.6 Remote OpenHealth Services (aka Web/H Services)

A ***Remote OpenHealth Service*** is one that is accessed via a web connection, and runs on a remote computer.

Remote OpenHealth Services require internet connectivity and rely on web service standards[18], tools and technologies.

[17] http://www.scouting.org/forms/34414.pdf
[18] http://www.w3.org/2002/ws/

Remote OpenHealth Services support an extensible interface, pseudo-coded as:

```
// Input parameters will typically include CCR-based content.
// Return results will typically be HTML or PDF.
// Additional parameters can be used for the exchange of
// account information.
public abstract DataElement[] Process(DataElement[] input);
public class DataElement
{
      public String name;
      public String type;                  // MIME type
      public String stringValue;
      public byte[] base64Value;
}
```

For instance, a service invocation that provides a CCR, in return for PDF content (e.g. patient's health risk assessment), would look something like:

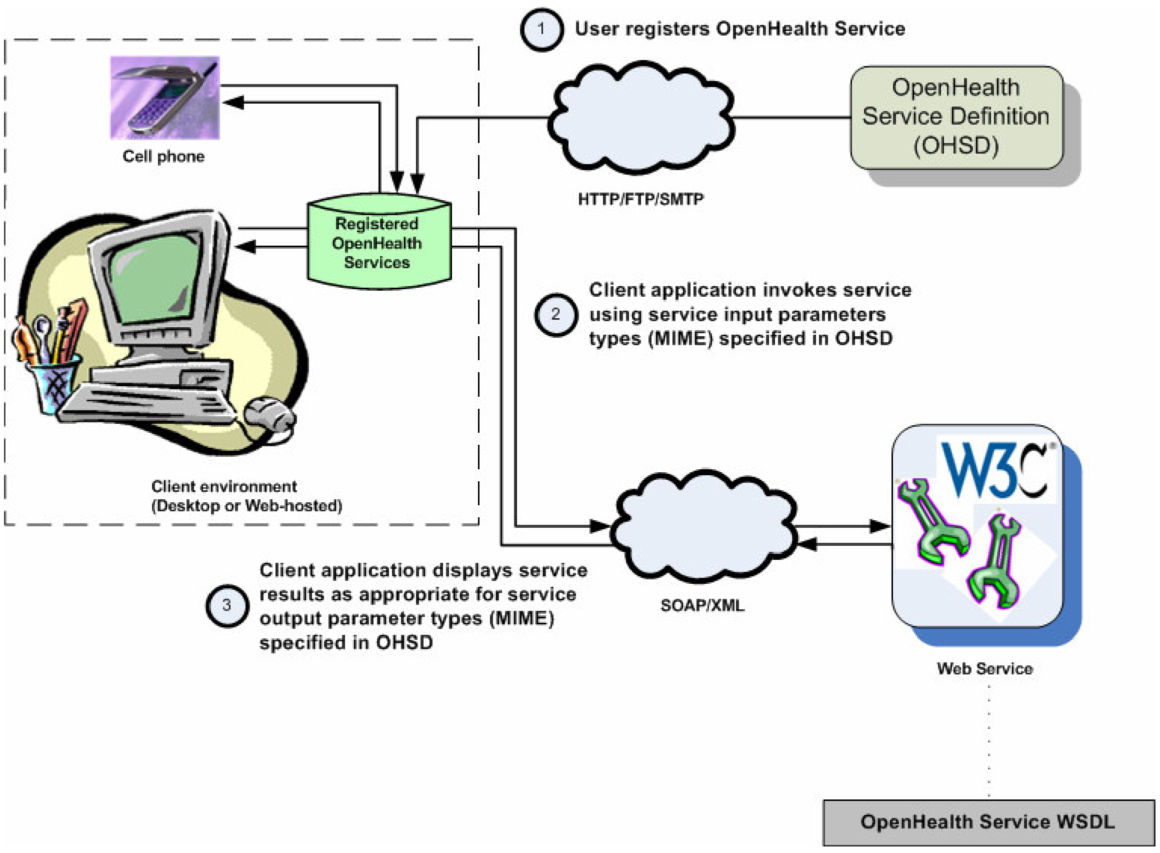
```
// Service accepts patient's CCR and generates (returns) PDF
// file with patient's health risk assessment

DataElement inputCCR = new DataElement();
inputCCR.name = "PatientCCR";
inputCCR.type = "application/x-ccr";
inputCCR.stringValue = "<?xml version='1.0' encoding='UTF-8' ?>
      <ContinuityOfCareRecord xmlns='urn:astm-org:CCR'> ...";

DataElement inputParams[] = new DataElement[1];
inputParams[0] = inputCCR;

DataElement outputParams[] = service.Process (inputParams);

// The return will look something like...
// outputParams[0].type = "application/pdf";
// outputParams[0].base64Value = "PDF file stream";
```


1 User registers OpenHealth Service
Cell phone
OpenHealth Service Definition (OHSD)
HTTP/FTP/SMTP
Registered OpenHealth Services
2 Client application invokes service using service input parameters types (MIME) specified in OHSD
Client environment (Desktop or Web-hosted)
3 Client application displays service results as appropriate for service output parameter types (MIME) specified in OHSD
SOAP/XML
W3C
Web Service
OpenHealth Service WSDL

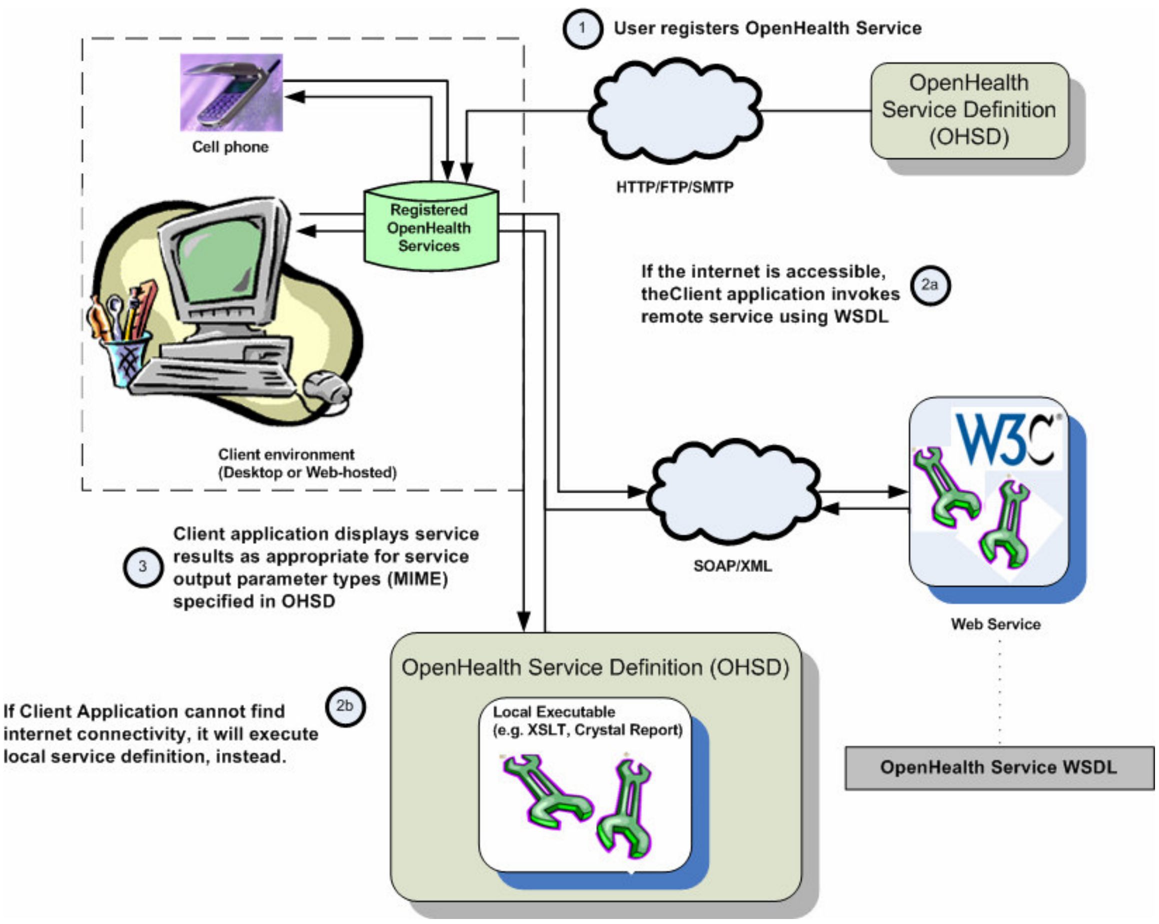
The picture above illustrates a user who registered OpenHealth services, as described in section 3.1.4. The service, being remote, includes WSDL information required to invoke the web service. When the user selects to execute the service, the client application locates the service definition in its database of registered services and reads WSDL URL. The web service is invoked and the results are presented to the user.

3.1.7 Hybrid OpenHealth Services

A ***Hybrid OpenHealth Service*** is one that can be accessed both locally and via a web connection, transparently.

Registration of a Hybrid OpenHealth service is done identically to the way it is for local and remote services (hence transparently).

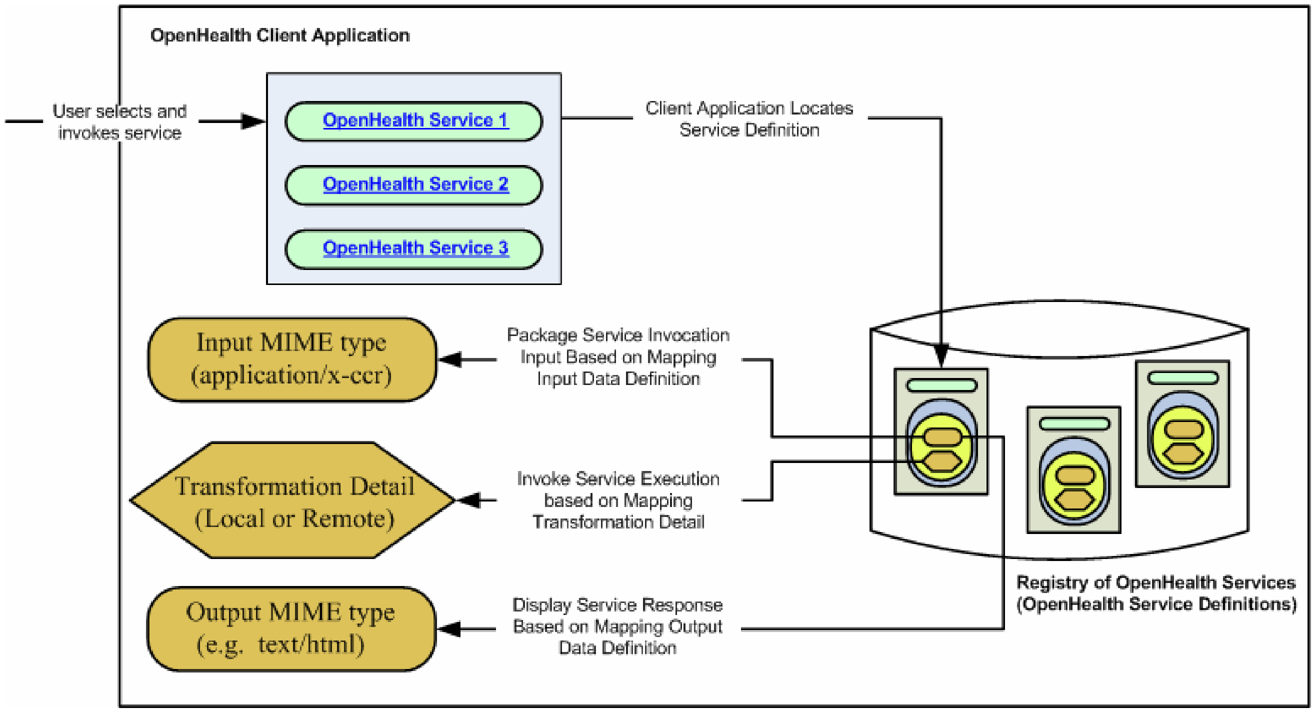
Once the service is installed, the client application can invoke the web service (if the internet is available) or revert to local service execution. This supports graceful service degradation in cases where the hosting device (e.g. cell phone, laptop, etc) may be out of internet service coverage. Graceful degradation support includes the ability to support data capture as well as minimizing functionality (e.g. reports, but no graphs, etc).


1
User registers OpenHealth Service
OpenHealth Service Definition (OHSD)
HTTP/FTP/SMTP
Cell phone
Registered OpenHealth Services
If the internet is accessible, theClient application invokes remote service using WSDL
2a
Client environment (Desktop or Web-hosted)
SOAP/XML
Web Service
3
Client application displays service results as appropriate for service output parameter types (MIME) specified in OHSD
OpenHealth Service Definition (OHSD)
Local Executable (e.g. XSLT, Crystal Report)
2b
If Client Application cannot find internet connectivity, it will execute local service definition, instead.
OpenHealth Service WSDL

3.2 OpenHealth Service Interface Design

The OpenHealth Service interface design relies on a number of different standards and techniques to create a framework that is initially flexible and overtime can evolve to more rigid interface definitions, as the market requires:

- Weak typing,
- The *generic* software pattern, and
- Service introspection (OpenHealth Service Definition and MIME content types).


OpenHealth Client Application
User selects and invokes service
OpenHealth Service 1
OpenHealth Service 2
OpenHealth Service 3
Client Application Locates Service Definition
Input MIME type (application/x-ccr)
Package Service Invocation Input Based on Mapping Input Data Definition
Transformation Detail (Local or Remote)
Invoke Service Execution based on Mapping Transformation Detail
Output MIME type (e.g. text/html)
Display Service Response Based on Mapping Output Data Definition
Registry of OpenHealth Services (OpenHealth Service Definitions)

3.2.1 Weak and Strong typing

Strong typing is a very useful technique for encoding communications protocols where the nature of the messages and protocols is well understood and largely immutable. Weak typing is more flexible and promotes greater experimentation.

As interoperability among CCR-enabled systems mature, there will be a tendency towards using strongly typed interfaces. Initially, a weak typed interface will support easy adoption and implementation, via implementation of a parameterized solution.

The OpenHealth framework encourages initial use of weak typing, while still providing a mechanism [19]by which we can evolve towards strong typing as pieces of the protocol become better understood and agreed upon.

3.2.2 MIME Content Types[20]

MIME (Multipurpose Internet Mail Extensions) is an internet standard originally designed to enable information other than text to be sent via email.

MIME use has extended well beyond its original intent (email) and in particular is the technology that is used by web browsers to figure out the type of information they are about to receive[21].

[19] See section 3.2.4 Categories
[20] http://www.w3schools.com/media/media_mimeref.asp

The OpenHealth framework leverages MIME content types to specify the information types that are exchanged by *specific* OpenHealth Services (local or remote). Services are invoked using the polymorphic *Process* method. The types for input and output parameters to the *Process* method are defined using MIME content types.

When a service is defined, the service's OpenHealth Service Definition includes the specific input and output MIME content types. Clients and servers can then rely on traditional, standard MIME-enabled mechanisms to handle incoming and outgoing content.

One can think of the OpenHealth Service interface as a *generic* (in the C++ template or java-sense[22]) that can be specialized with specific parameterized (MIME) types. This software pattern is recognized as being both type-safe and extensible.

The use of MIME content types in support of OpenHealth Services supports extending service definitions both by using different, but standard application types and also by defining additional, custom MIME types.

3.2.3 OpenHealth Service Definition

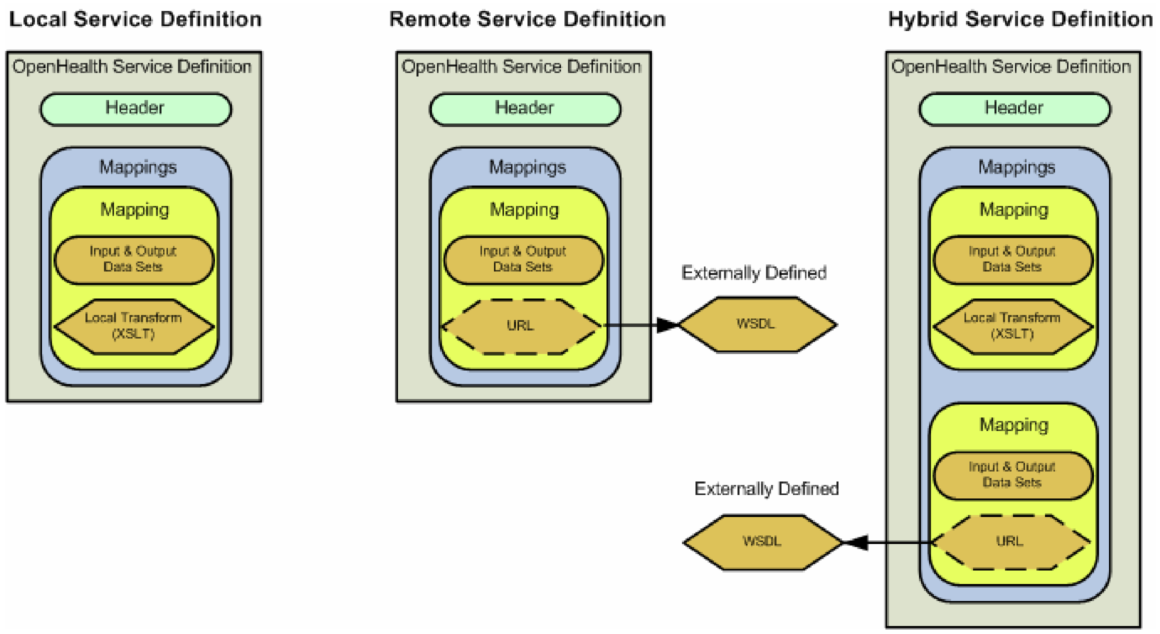
The ***OpenHealth Service Definition File*** (OHSD) will be used to define information specific to a given OpenHealth Service.

The OHSD file contains a single header that describes the service and multiple (one or more) mappings, which contain details on how to execute the service:

Header:
- Name
- Description
- Version Info (upgradeability)
- Author Info
- Categories [23](zero or more)
- Digital Signature

Mappings [24](one or more):
- Input Data Set
- Output Data Set
- Transformation Detail

[21] To see how Mozilla-based browsers handle MIME types: http://developer.mozilla.org/en/docs/How_Mozilla_determines_MIME_Types; Internet Explorer: http://msdn.microsoft.com/library/default.asp?url=/workshop/networking/moniker/overview/appendix_a.asp
[22] http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf and http://java.sun.com/j2se/1.5.0/docs/guide/language/generics.html
[23] See detail in section 3.2.4 Categories
[24] See detail in section 3.2.5 Mappings



Specific details of header categories and mappings are covered in subsequent sections.

OpenHealth Service Definitions hide the complexity of service definition and locality of service execution from non-technical users.

3.2.4 Categories

An OHSD can have zero or more categories, which define the kind of service provided.

Service categories, when explicitly identified in the OpenHealth Service Definition file, can be used in support of:

- Domain-specific usage, e.g. pre-natal care, AIDS, diabetes,
- Automated service classification, or
- As an extension mechanism that can be used to redirect usage of type-specific WSDL.

Domain-specific usage allows for the easy (automated) organization of service instances – by categories.

As an extension mechanism, categories allow subsets of services to evolve into more specific informational exchanges (in other words, more specialized knowledge of required content, etc – as a formal API). In this respect, OpenHealth categories would be handled similarly to MIME Content types. Client applications implement MIME support, on a type-specific manner. Initially, all OpenHealth categories will be implemented similarly (i.e. invoking the *Process* method, as polymorphed in accordance with the service definition). It is possible for category types to evolve, to support category-specific WSDL definitions, with methods other than the *Process* method. Once this evolution occurs, applications will know to handle the differentiated categories, appropriately. This is a key example of how the OpenHealth Framework is designed to support protocol evolution and extensibility.

Categories are an open enumeration, whereby new entries can be added, in support of the framework's extensibility.

As of this version of the specification, proposed categories include:

Enumeration Name	Category	Description
Expenses	Expenses	
TranslationService	Translation Service	Translation services will modify the content and structure of the input parameters to generate a new document. Translation services are useful for instance when converting from CCR into CCD or CDA or generating PDF/H from CCR.
DeIdentificationService	De-Identification Service	It is often appropriate for clinical information to be exchanged without corresponding identification information (e.g. name, address, social security number). In these cases, one can use a de-identification service which is aware of the input document format (e.g. CCR) and how/which information should be de-identified. These services are typically compounded with other services (e.g. directory services for counseling agencies/groups).
QueuingService	Queuing Service	Queues can be used in support of asynchronous workflows. A queuing service can return queuing transaction ids that can be later used in completion of the long-term transaction.
Summary	Summary	A service that provides summary reporting, e.g. list of providers, immunization record, medical history summary, etc.
Graph	Graph	A service that produces graphical reports.
CommonForm	Common Form	A number of hospitals, insurers, associations, etc have standard forms that are routinely filled in the delivery of care. Common form services will produce pre-filled forms (in HTML, PDF or other format), using input clinical information (CCR). Examples of

		forms include hospital admission forms, Boy Scouts of America summer camp forms, etc.
ClinicalDirectivesAndRecommendations	Medical Decision Support	Decision support services are a unique category of information reporting, which is likely to evolve into specialized category processing.
ExplicitSubmitRequired	ExplicitSubmitRequired	Some services can require an explicit SUBMIT button be present. The purpose of this category is to support services whose execution is “expensive” either in time or some other resource. These services only get invoked at the user’s explicit request and not on-the-fly, automatically by default.

Each category should be further documented in implementation guides that illustrate unique usage, pertaining to each category.

3.2.5 Mappings

A mapping structure contains a list of input arguments to the service, and a list of output result values. The service interface defines these arguments as weakly typed - that is to say they are passed as untyped blobs of data, along with a MIME Content-Type to specify how to interpret the blob.

In addition to the standard MIME Content-Types, this specification recognizes an additional ‘*well known*’ MIME Content-Type:

MIME Content Type	Description
application/x-ccr	CCR document

Important other types that, it is expected, will typically be used as return values include:

- text/html,
- application/pdf[25],
- application/x-url[26], and
- image/png.

Please note that you can return ANY MIME content type from a service, and if a reader for it is installed on the users’ computer, it should display properly. It is important to note that this does not necessarily require any additional coding on the part of the

[25] It is unclear at the point this is being written whether PDF-H will use a different MIME content type. It is unlikely, however.

[26] The application/x-url MIME type is used in support of scenarios that require additional content and/or interaction with the user.

OpenHealth Service client application. It is common for most operating systems to support registry services that map MIME content types to default applications. This is most commonly seen when using a browser: when you target specific types, the registry is used to locate registered viewers, which are then invoked by the browser. Similarly, OpenHealth Service client applications can leverage local operating system support.

Compliant applications need not support ANY of these formats themselves.

Pragmatically – one can expect that most OpenHealth Services will accept 'application/x-ccr' as input and produce 'application/pdf' or 'text/html' as output.

Mapping Transformation Detail

The mapping transformation detail includes information that is required to invoke the execution of the OpenHealth service (local or remote).

The transformation detail can be either a **LocalTransform** or a **RemoteTransform**. (Note that hybrid services contain both a local and remote transform.)

LocalTransform

A LocalTransform consists of a typed BLOB which is the actual executable code/program that needs to be run to map inputs to outputs.

A LocalTransform contains an 'Encoding' attribute to indicate a standard MIME encoding (such as base64), and an actual 'Type'.

For now – the supported MIME Content Type include:

- application/x-xslt, for locally executable XSLT transformation scripts, and
- application/java-archive[27], for locally executable java applications.

It is expected that in the future, the framework will support additional types such as Crystal Reports, etc.

Example

```
<OpenHealthServiceDefinition …>
    <Header>
        …
    </Header>
    <Mappings>
        <Mapping>
            <Input>
                <DataDefinition>
                    <Format>
                        <Type>application/x-ccr</Type>
                    </Format>
                </DataDefinition>
            </Input>
```

[27] In order to package a locally executable java application, package the required .class files into a jar file, containing a manifest file that identifies the "main" class (http://www.cs.princeton.edu/introcs/85application/jar/jar.html).

```
            <Output>
                <DataDefinition>
                    <Format>
                        <Type>text/html</Type>
                    </Format>
                </DataDefinition>
            </Output>
            <TransformDetail>
                <LocalTransform Type="application/xslt" Encoding="base64">
                    PD9…
                </LocalTransform>
            </TransformDetail>
        </Mapping>
    </Mappings>
</OpenHealthServiceDefinition>
```

The contents of the <LocalTransform> would in this case be an XSLT script which maps a document in CCR format to HTML. The XSLT can be created in any XSLT tool, and then base64 encoded.

Note that some OHSD editing tools – like HealthFrame [28]– have built-in support for encoding and packaging; alternatively, one can use uuencode.

RemoteTransform

A RemoteTransform consists of nothing more than a URL for the web service definition.

Note that you can access the WSDL used by that service by just using the url for *specified_URL*?WSDL

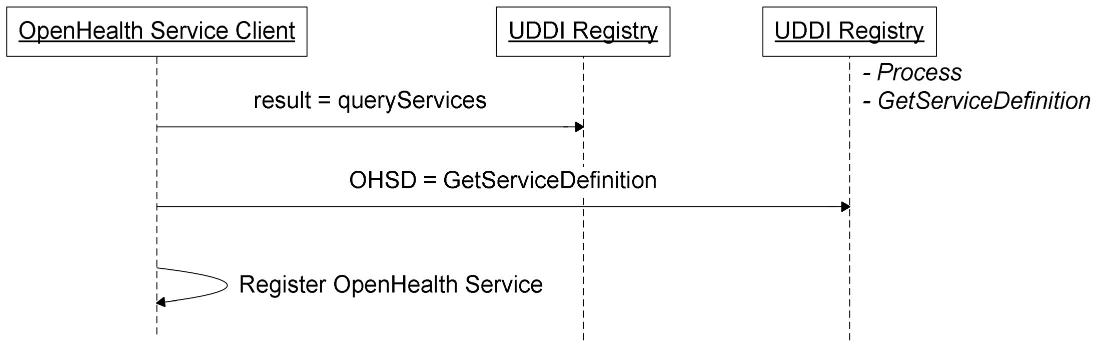
For example, as of this writing one can use http://www.recordsforliving.com/OpenHealthServices/Services/OpenHealthServicesSimpleSampleServer/WebService.asmx?WSDL

3.2.6 OpenHealth Service Discovery

Remote (and hybrid) OpenHealth Services (web services) can be discovered using UDDI.

Using UDDI, two operations can be discovered: *Process* and *GetServiceDefinition*.

[28] HealthFrame is a personal health record software tool that supports the OpenHealth Service framework, including being an OpenHealth Service Definition editor.


OpenHealth Service Client
UDDI Registry
UDDI Registry
- Process
- GetServiceDefinition
result = queryServices
OHSD = GetServiceDefinition
Register OpenHealth Service

The *Process* operation is the polymorphic method that invokes the execution of the OpenHealth Service. The invocation of the *Process* operation has been illustrated in section 3.1.6.

The *GetServiceDefinition* gives access to additional information about the service, as specified in the OpenHealth Service Definition. Once the system has access to the OHSD, additional information not typically captured in UDDI can be accessed:

- Specific MIME types for polymorphic *Process* operation (input/output parameters)
- Authorship
- Categorization (including support for OpenHealth Service evolution)
- Mappings (including possibly local transformations, for hybrid services).

3.3 Specific Usages

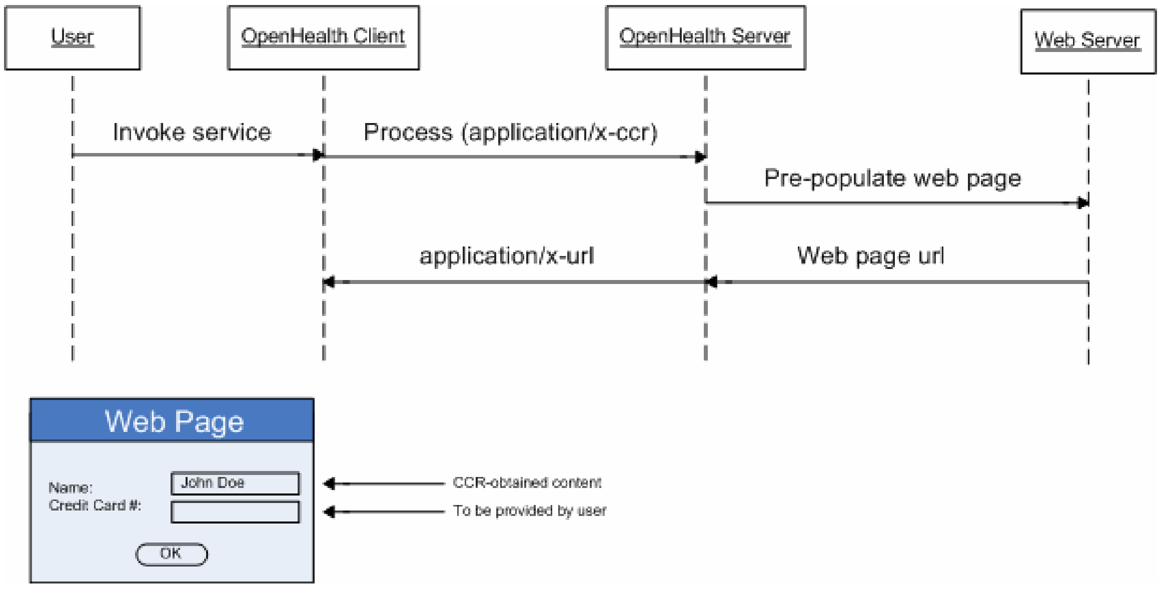
3.3.1 Additional Content or Interactive Scenarios

At times it is necessary for OpenHealth Services to prompt the user for additional information not contained in the CCR. In some cases, this information is part of a workflow that requires the user to provide preferences, account information/registration or simply to interact with the service in a more direct way. In these cases, the web service may need to return rich content, as for instance a web page with scripting.

In these cases, the service can return the application/x-url MIME content-type, containing the URL of the redirected web page. Assuming, for instance, that the service is defined to accept application/x-ccr as the input, the net effect is that the user's CCR is sent to the service, which in turn pre-populates the web page as appropriate, and returns the URL of the pre-filled web page to be displayed by the client application.

This particular use is of great value to users who are often asked to enter the same health information multiple times. Besides being annoying, this manual re-entry of data can lead to data entry mistakes and oftentimes relies on the user's memory. Using OpenHealth Services, clinical information contained in the CCR will be automatically used to pre-fill the web form, saving the user's time and avoiding common mistakes.

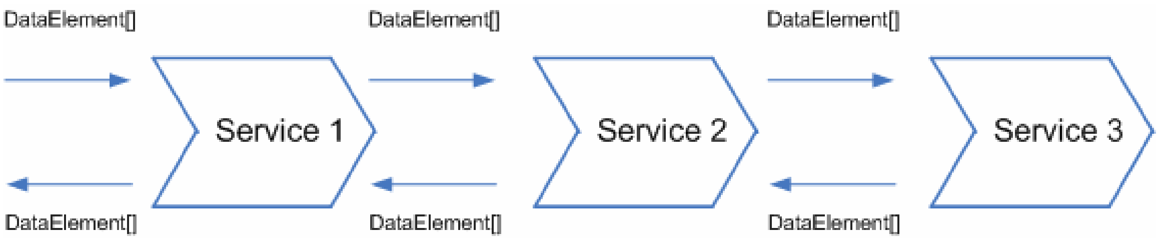
```
<?xml version="1.0" encoding="UTF-8"?>
<OpenHealthServiceDefinition
        xmlns="http://www.RecordsForLiving.com/Schemas/
                2006-08/OpenHealthServiceDefinition/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Header>
                <ShortName>Simple (Remote) URL Sample</ShortName>
                <LongName>OpenHealth Services Simple (Remote) Sample of returning a
                                URL
                </LongName>
                <VersionInfo>
                        <FamilyGUID>ID-1B2595A5-CB80-4527-8CE3-
                                7828C548F202</FamilyGUID>
                        <VersionNumber>20060905</VersionNumber>
                        <VersionName>2006-09-05</VersionName>
                </VersionInfo>
                <AuthorInformation>
                        <Copyright>Copyright 2006 Records For Living</Copyright>
                        <FullName>Lewis G. Pringle of Records For Living</FullName>
                        <Icon64x64 mimeType="image/gif">
                        ...
                        </Icon64x64>
                </AuthorInformation>
                <Categories>
                        <Category>Report</Category>
                </Categories>
        </Header>
        <Mappings>
                <Mapping>
                        <Input>
                                <DataDefinition>
                                        <Format>
                                                <Type>application/x-ccr</Type>
                                        </Format>
                                </DataDefinition>
                        </Input>
                        <Output>
                                <DataDefinition>
                                        <Format>
                                                <Type>application/x-url</Type>
                                        </Format>
                                </DataDefinition>
                        </Output>
                        <TransformDetail>
                                <RemoteTransform>
                                        <URL>http://www.recordsforliving.com/
                                            OpenHealthServices/Services/
                                            OpenHealthServicesSimpleURLSampleServer/
                                            WebService.asmx
                                        </URL>
                                </RemoteTransform>
                        </TransformDetail>
                </Mapping>
        </Mappings>
</OpenHealthServiceDefinition>
```



3.3.2 Service Chaining

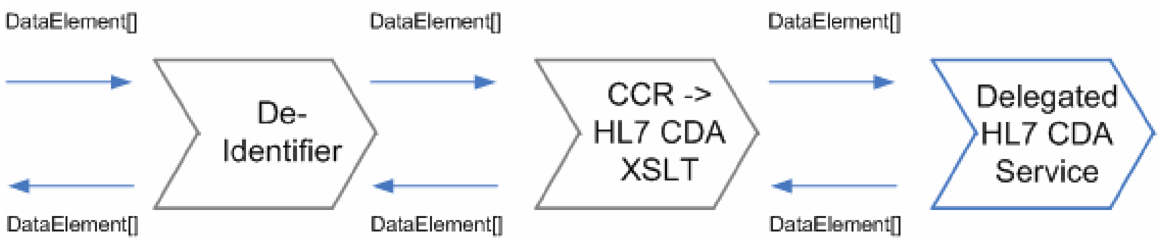
Service chaining enables the output of one service to be used as the input of another. This is a pattern commonly used in UNIX systems (pipes) and allows for fairly complex systems to be built using smaller, simpler components. In particular, service chaining allows for the creation of reusable utility services such as user anonymity or dynamic translation to/from CCR and HL7 CDA.

OpenHealth Services supports service chaining by defining WSDL input and output parameters that are of the same type (i.e. DataElement[]).



The user experience is the net combined effect of applying each service in turn, as if it were a single service. This transparent service composition allows for localized customizations by the user (e.g. “turn on/off” of identity content).

Additionally, service functional composition allows for the quick deployment of different flavors of services by merely combining existing utility services with new service functionality. For instance, the service combination below re-uses the de-identification utility service (which removes identifying information about the patient) and the CCR to HL7 CDA mapping (XSLT). When these services are applied in combination with a hypothetical HL7 CDA delegation service, it allows the user anonymous access to services that were natively deployed for HL7 CDA:



Finally, it should be noted that service location independence allows for a mix and match of local and remote services in a way that provide the best quality of service (for instance, translation can be done locally, to minimize the exchange of multiple, potentially large clinical records.

OpenHealth Service composition is not a replacement for WS-* orchestration and collaboration mechanisms, which can be used to build composite applications. OpenHealth Service composition provides a mechanism that can be user accessible – as opposed to programmatic in nature.

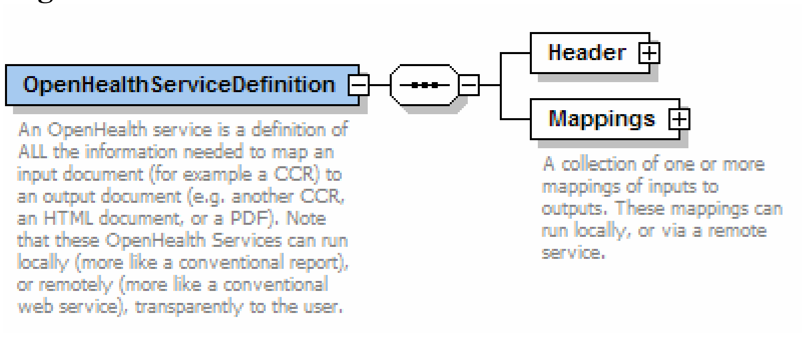
5. Acknowledgements

This document was written by Records for Living (http://www.RecordsForLiving.com), in collaboration with members of the AAFP's Center for Health Information Technology (http://www.centerforhit.org/).

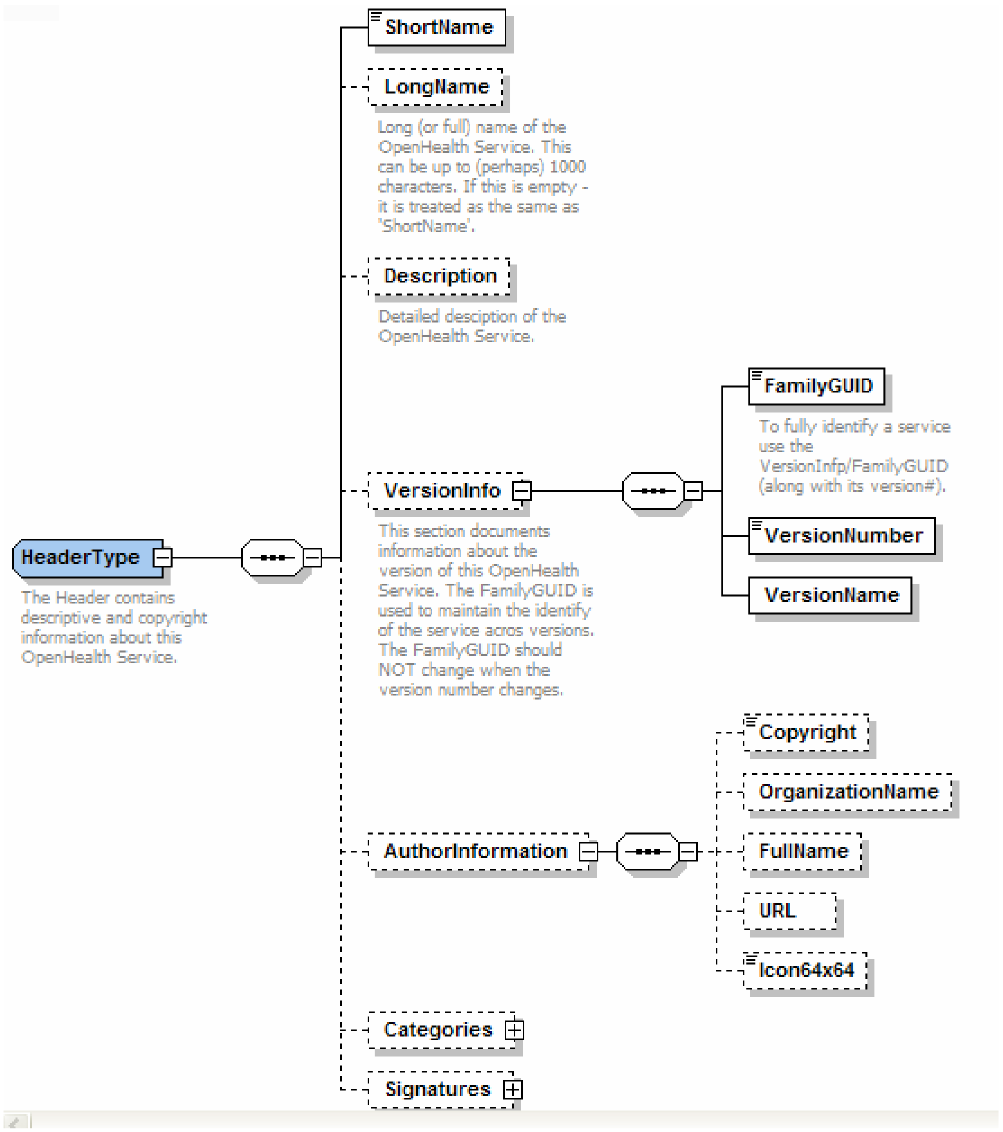
We would like to acknowledge significant design and review contributions from CINA (http://www.cina-us.com/), and MedCommons (http://www.medcommons.net).

APPENDIX I – OpenHealth Service Definition – Schema

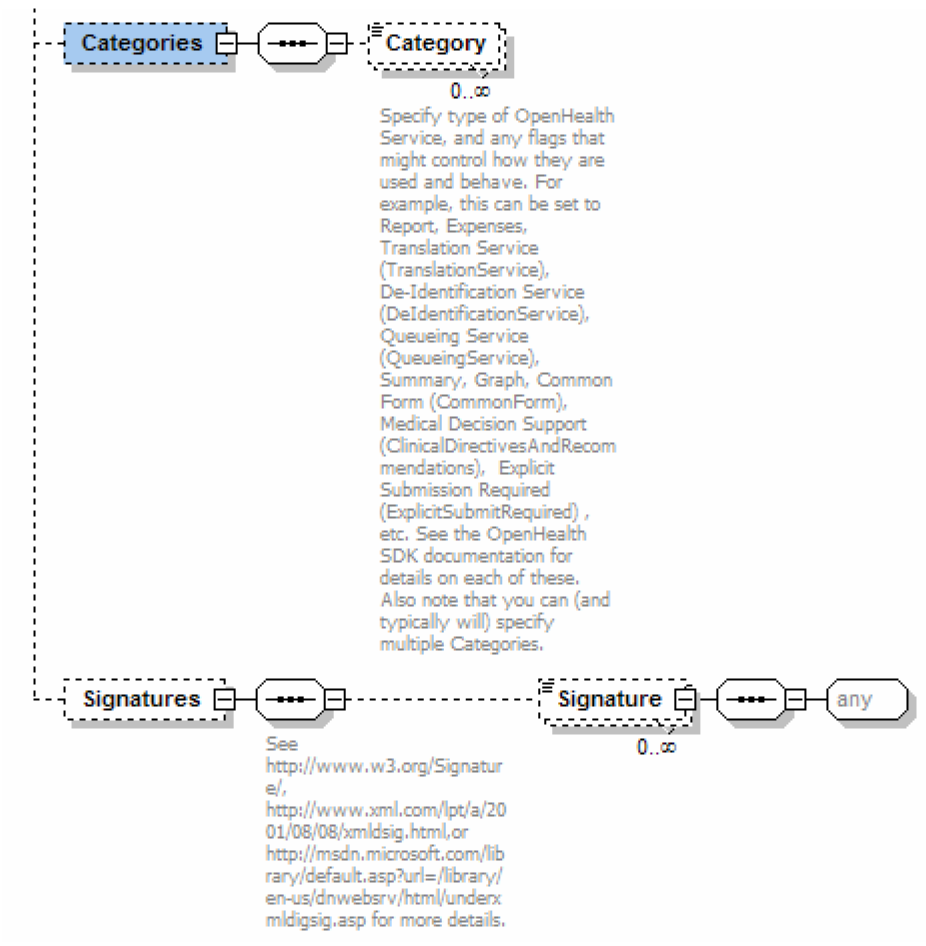
High-level schema:


OpenHealthServiceDefinition
An OpenHealth service is a definition of ALL the information needed to map an input document (for example a CCR) to an output document (e.g. another CCR, an HTML document, or a PDF). Note that these OpenHealth Services can run locally (more like a conventional report), or remotely (more like a conventional web service), transparently to the user.
Header
Mappings
A collection of one or more mappings of inputs to outputs. These mappings can run locally, or via a remote service.

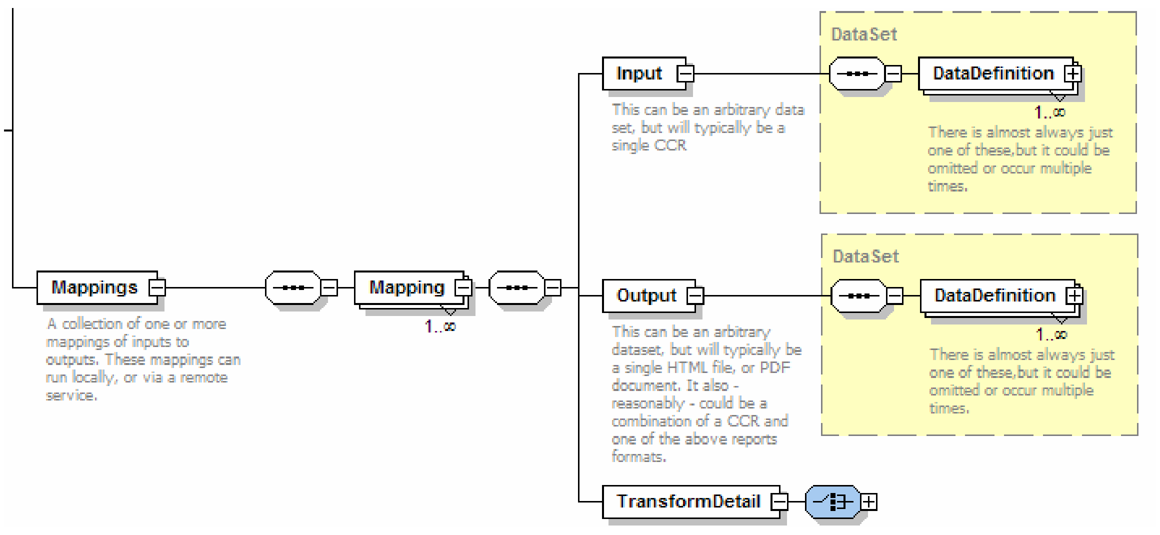
Header information:


ShortName
LongName
Long (or full) name of the OpenHealth Service. This can be up to (perhaps) 1000 characters. If this is empty - it is treated as the same as 'ShortName'.
Description
Detailed desciption of the OpenHealth Service.
FamilyGUID
To fully identify a service use the VersionInfp/FamilyGUID (along with its version#).
VersionInfo
This section documents information about the version of this OpenHealth Service. The FamilyGUID is used to maintain the identify of the service acros versions. The FamilyGUID should NOT change when the version number changes.
VersionNumber
VersionName
HeaderType
The Header contains descriptive and copyright information about this OpenHealth Service.
Copyright
OrganizationName
AuthorInformation
FullName
URL
Icon64x64
Categories
Signatures

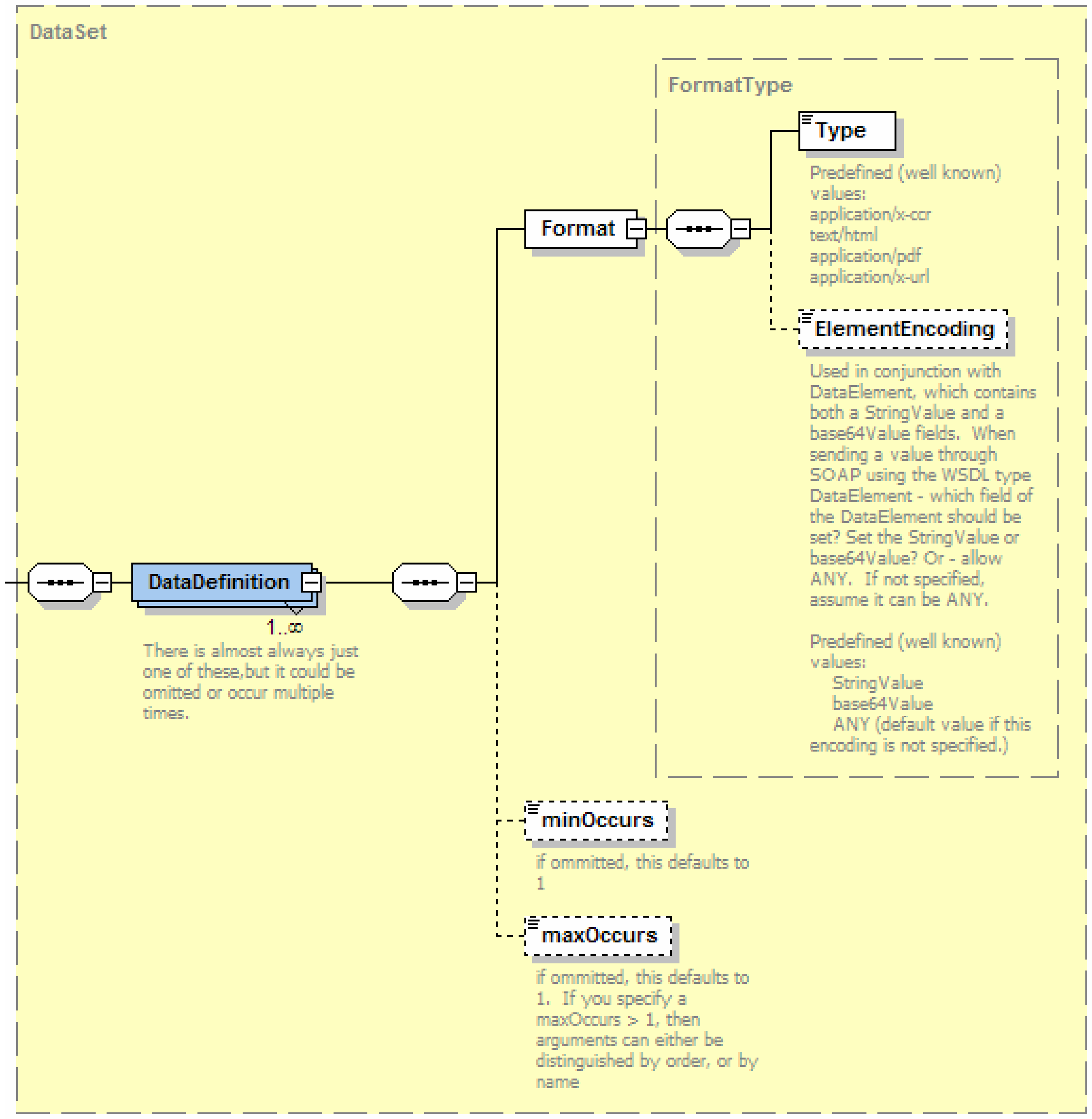
Header/Categories and Header/Signatures information:


Categories
Category
0..∞
Specify type of OpenHealth Service, and any flags that might control how they are used and behave. For example, this can be set to Report, Expenses, Translation Service (TranslationService), De-Identification Service (DeIdentificationService), Queueing Service (QueueingService), Summary, Graph, Common Form (CommonForm), Medical Decision Support (ClinicalDirectivesAndRecommendations), Explicit Submission Required (ExplicitSubmitRequired) , etc. See the OpenHealth SDK documentation for details on each of these. Also note that you can (and typically will) specify multiple Categories.
Signatures
Signature
any
0..∞
See http://www.w3.org/Signature/, http://www.xml.com/lpt/a/2001/08/08/xmldsig.html,or http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/underxmldigsig.asp for more details.

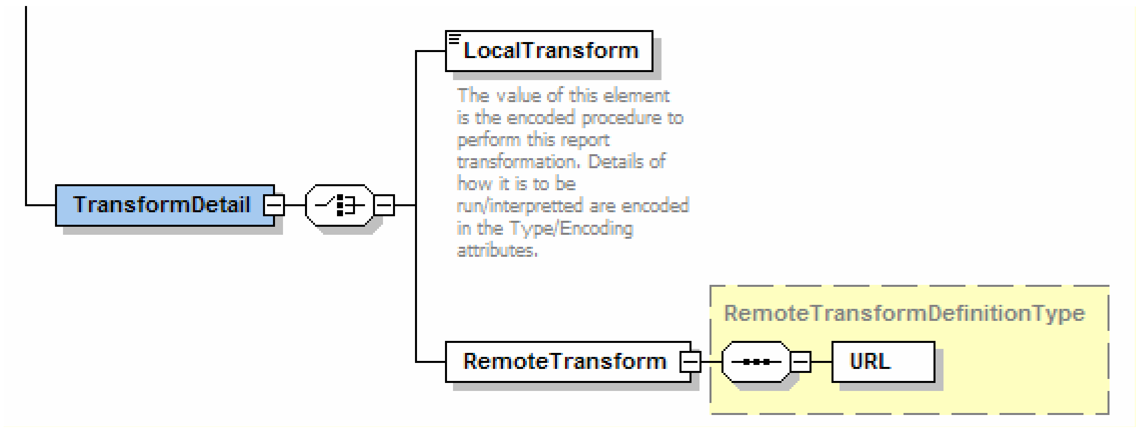
OpenHealthServiceDefinition/Mappings information:


Mappings
A collection of one or more mappings of inputs to outputs. These mappings can run locally, or via a remote service.
Mapping
1..∞
Input
This can be an arbitrary data set, but will typically be a single CCR
DataSet
DataDefinition
1..∞
There is almost always just one of these,but it could be omitted or occur multiple times.
Output
This can be an arbitrary dataset, but will typically be a single HTML file, or PDF document. It also - reasonably - could be a combination of a CCR and one of the above reports formats.
DataSet
DataDefinition
1..∞
There is almost always just one of these,but it could be omitted or occur multiple times.
TransformDetail

DataDefinition:


DataSet
FormatType
Type
Predefined (well known) values:
application/x-ccr
text/html
application/pdf
application/x-url
Format
ElementEncoding
Used in conjunction with DataElement, which contains both a StringValue and a base64Value fields. When sending a value through SOAP using the WSDL type DataElement - which field of the DataElement should be set? Set the StringValue or base64Value? Or - allow ANY. If not specified, assume it can be ANY.
Predefined (well known) values:
StringValue
base64Value
ANY (default value if this encoding is not specified.)
DataDefinition
1..∞
There is almost always just one of these,but it could be omitted or occur multiple times.
minOccurs
if ommitted, this defaults to 1
maxOccurs
if ommitted, this defaults to 1. If you specify a maxOccurs > 1, then arguments can either be distinguished by order, or by name

Transform Detail:


TransformDetail
LocalTransform
The value of this element is the encoded procedure to perform this report transformation. Details of how it is to be run/interpretted are encoded in the Type/Encoding attributes.
RemoteTransform
RemoteTransformDefinitionType
URL

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns="http://www.RecordsForLiving.com/Schemas/
                2006-08/OpenHealthServiceDefinition/"
        targetNamespace="http://www.RecordsForLiving.com/Schemas/
                2006-08/OpenHealthServiceDefinition/"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <xs:annotation>
    <xs:documentation>An OpenHealth Service Definition (OHSD) file contains
    information pertaining to a specific instance of an OpenHealth Service.
    Information contained in OHSD files include summary information about
    the service such as the service name, description, version, author info,
    and service categories.  OHSD files can be digitally signed.  OHSD files
    also contain type information for input and output service parameters,
    as well as optional transformation scripts.

    OpenHealth Service Definition files are part of the OpenHealth Service
    Framework.

    This schema specifies the format for all OHSD files.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType name="ImageType">
    <xs:annotation>
      <xs:documentation/>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute name="mimeType" type="xs:string"
          use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="FormatType">
    <xs:annotation>
      <xs:documentation>The format types are specified with MIME types
      (see http://www.iana.org/assignments/media-types/).  Many types
      needed to support the exchange of health-related content, however,
      are not included in the standard set of MIME types.  Some well-
      known types are listed below (in the Type field). Other future
      types might include ACCOUNT#, BILLING_INFORMATION, and other basic
      types that may be needed to augment CCR data in common transactions.  Note that
      applications recieving data in a type not listed below may look up the argument type
      and how to display it in their computer's registry.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Type" type="xs:string">
        <xs:annotation>
          <xs:documentation>Predefined (well known) values:
            application/x-ccr
            text/html
            application/pdf
            application/x-url
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ElementEncoding" type="xs:string" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Used in conjunction with DataElement, which contains both a
            StringValue and a base64Value fields.  When sending a value through SOAP using
            the WSDL type DataElement - which field of the DataElement should be set? Set
            the StringValue or base64Value? Or - allow ANY.  If not specified, assume it
            can be ANY.

            Predefined (well known) values:
                  StringValue
                  base64Value
                  ANY (default value)
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="FileSuffix" type="xs:string" minOccurs="0">
        <xs:annotation>
          <xs:documentation>
            This file suffix field (like .x12-837) is used to register in
            the operating system an association between the
            given MIME Content-Type and this file suffix. Its optional,
            but helpful,  if you wish to have a client application automatically
            recognize files of the given type (suffix)
            - for example - for a source for Translation Service - like X12-837.
          </xs:documentation>
        </xs:annotation>
      </xs:element>  </xs:sequence>
  </xs:complexType>
  <xs:complexType name="DataSet">
    <xs:annotation>
      <xs:documentation>Most OpenHealth services take a single input and
        produce a single output, but this abstraction allows for multiple inputs and
        multiple outputs. The point of this is - for example - to allow a service which
        merged two documents, or a service which produced both a printed report and a CCR
        as output.
       </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="DataDefinition" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>There is almost always just one of these,but it could be
            omitted or occur multiple times.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Format" type="FormatType"/>
            <xs:element name="minOccurs" type="xs:int" default="1" minOccurs="0">
              <xs:annotation>
                <xs:documentation>if ommitted, this defaults to 1</xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="maxOccurs" type="xs:int" default="1" minOccurs="0">
              <xs:annotation>
                <xs:documentation>if ommitted, this defaults to 1.  If you specify a
                  maxOccurs > 1, then arguments can either be distinguished by order, or by
                  name</xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="HeaderType">
    <xs:annotation>
      <xs:documentation>The Header contains descriptive and copyright information about
        this OpenHealth Service.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ShortName">
        <xs:simpleType>
          <xs:annotation>
            <xs:documentation>Short name of the OpenHealth Service (roughly 100 characters
              or less)
            </xs:documentation>
          </xs:annotation>
          <xs:restriction base="xs:string">
            <xs:maxLength value="100"/>
          </xs:restriction>
        </xs:simpleType>
```

```
</xs:element>
<xs:element name="LongName" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Long (or full) name of the OpenHealth Service. This can be up
      to (perhaps) 1000 characters. If this is empty - it is treated as the same as
      'ShortName'.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="Description" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Detailed desciption of the OpenHealth Service.
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="VersionInfo" minOccurs="0">
  <xs:annotation>
    <xs:documentation>This section documents information about the version of this
      OpenHealth Service. The FamilyGUID is used to maintain the identify of the
      service acros versions. The FamilyGUID should NOT change when the version
      number changes.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="FamilyGUID" type="xs:ID">
        <xs:annotation>
          <xs:documentation>To fully identify a service use the
            VersionInfp/FamilyGUID (along with its version#).</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="VersionNumber" type="xs:nonNegativeInteger"/>
      <xs:element name="VersionName"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="AuthorInformation" minOccurs="0">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Copyright" type="xs:string" minOccurs="0"/>
      <xs:element name="OrganizationName" minOccurs="0"/>
      <xs:element name="FullName" minOccurs="0"/>
      <xs:element name="URL" minOccurs="0"/>
      <xs:element name="Icon64x64" type="ImageType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Categories" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Category" type="xs:string" minOccurs="0"
          maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>Specify type of OpenHealth Service, and any flags that
            might control how they are used and behave. For example, this can be set
            to Report, Expenses, Translation Service, De-Identification Service,
            Queueing Service, Summary, Graph, Common Form, Medical Decision Support,
            Explicit Submission Required, etc. See the OpenHealth SDK documentation
            for details on each of these. Also note that you can (and typically will)
            specify multiple Categories.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Signatures" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:annotation>
```

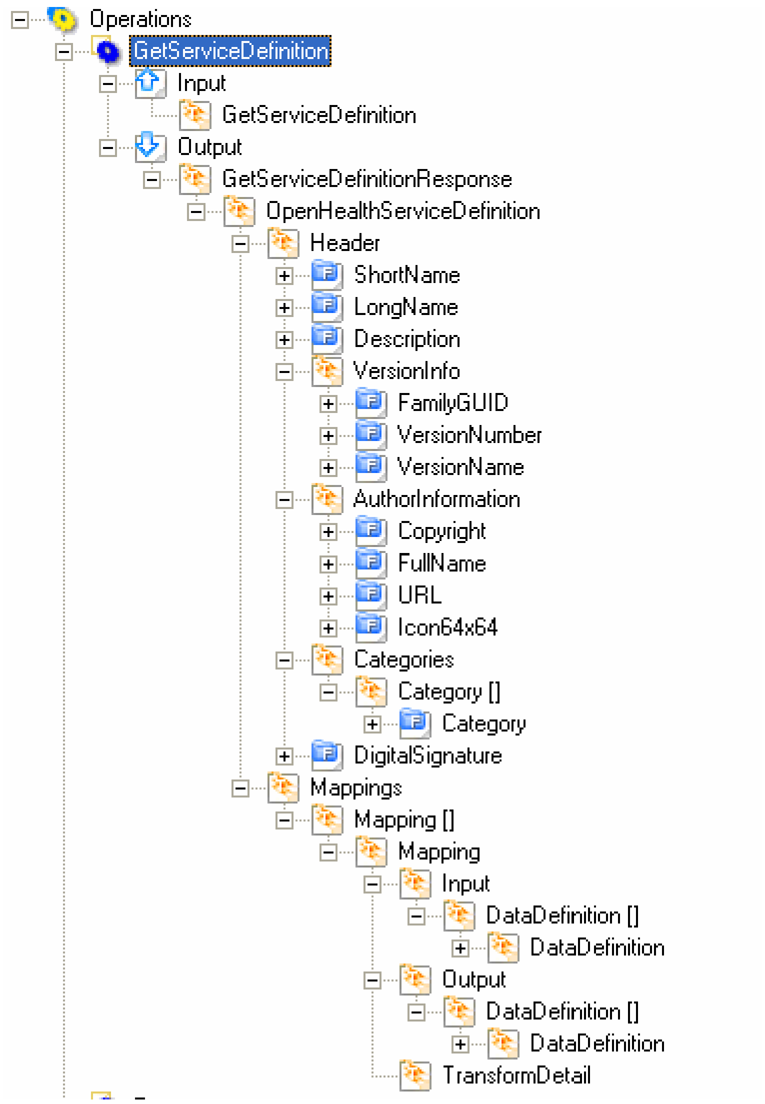
```
          <xs:documentation>See http://www.w3.org/Signature/,
            http://www.xml.com/lpt/a/2001/08/08/xmldsig.html,or
            http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/
              html/underxmldigsig.asp for more details.
          </xs:documentation>
        </xs:annotation>

        <xs:element name="Signature" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType mixed="true">
            <xs:sequence>
              <xs:any processContents="skip"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="LocalTransformDefinitionType">
  <xs:annotation>
    <xs:documentation>This is a definition of how to transform data locally - on this
      machine (direct report rather than service based).
      Examples of valid Type fields include 'http://www.w3.org/1999/XSL/Transform', for
      XSLT scripts. Examples of valid Encoding fields include 'base64' for base-64
      encoded strings.
    </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Type" type="xs:anyURI" use="required"/>
      <xs:attribute name="Encoding" type="xs:string" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name="RemoteTransformDefinitionType">
  <xs:annotation>
    <xs:documentation>This type of transform is handled via a SOAP call (specify SOAP
      WSDL). From the user-point of view - there is no obvious difference between local
      and remote transforms (except possibly the requirement for particular software to
      be installed on the computer for local transforms and the requirement for network
      access for remote transforms). Note - RemoteTransformDefinitionType are what are
      usually referred to as 'services'.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="URL"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="OpenHealthServiceDefinition">
  <xs:annotation>
    <xs:documentation>An OpenHealth service is a definition of ALL the information
      needed to map an input document (for example a CCR) to an output document (e.g.
      another CCR, an HTML document, or a PDF). Note that these OpenHealth Services can
      run locally (more like a conventional report), or remotely (more like a
      conventional web service), transparently to the user.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Header" type="HeaderType"/>
      <xs:element name="Mappings">
        <xs:annotation>
          <xs:documentation>A collection of one or more mappings of inputs to outputs.
              These mappings can run locally, or via a remote service.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Mapping" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
```

```
                    <xs:element name="Input" type="DataSet"> </xs:element>
                    <xs:element name="Output" type="DataSet"> </xs:element>
                    <xs:element name="TransformDetail">
                      <xs:complexType>
                        <xs:choice>
                          <xs:element name="LocalTransform"
                            type="LocalTransformDefinitionType"/>
                          <xs:element name="RemoteTransform"
                            type="RemoteTransformDefinitionType"/>
                        </xs:choice>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

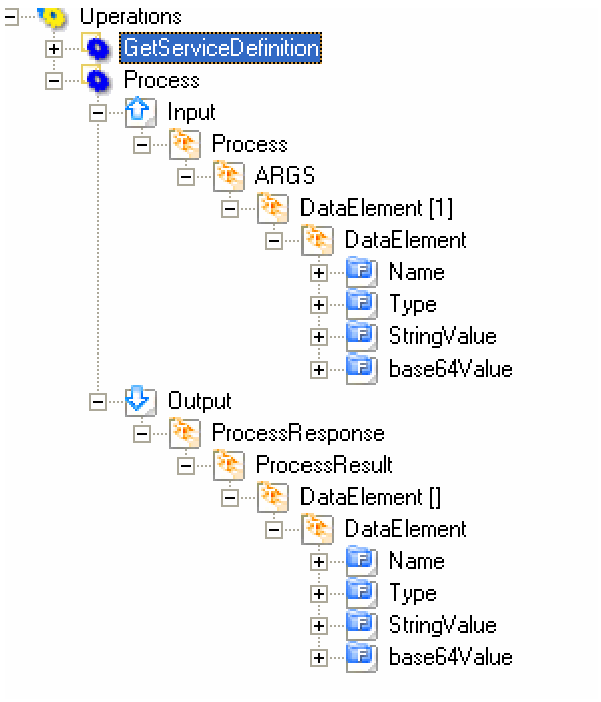
APPENDIX II – Remote OpenHealth Service – WSDL

GetServiceDefinition:



```
Operations
  GetServiceDefinition
    Input
      GetServiceDefinition
    Output
      GetServiceDefinitionResponse
        OpenHealthServiceDefinition
          Header
            ShortName
            LongName
            Description
            VersionInfo
              FamilyGUID
              VersionNumber
              VersionName
            AuthorInformation
              Copyright
              FullName
              URL
              Icon64x64
            Categories
              Category []
                Category
            DigitalSignature
          Mappings
            Mapping []
              Mapping
                Input
                  DataDefinition []
                    DataDefinition
                Output
                  DataDefinition []
                    DataDefinition
                TransformDetail
```

Process:



```
Operations
  GetServiceDefinition
  Process
    Input
      Process
        ARGS
          DataElement [1]
            DataElement
              Name
              Type
              StringValue
              base64Value
    Output
      ProcessResponse
        ProcessResult
          DataElement []
            DataElement
              Name
              Type
              StringValue
              base64Value
```

```
<?xml version="1.0" encoding="utf-8"?>

<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:tns="http://www.RecordsForLiving.com/Schemas/2006-08/OpenHealthServicesWSDL/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:ohsd="http://www.RecordsForLiving.com/Schemas/
        2006-08/OpenHealthServiceDefinition/"
    targetNamespace="http://www.RecordsForLiving.com/Schemas/
        2006-08/OpenHealthServicesWSDL/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <wsdl:types>
        <s:schema elementFormDefault="qualified"
            targetNamespace="http://www.RecordsForLiving.com/Schemas/
                2006-08/OpenHealthServicesWSDL/">
            <s:complexType name="DataElement">
                <s:annotation>
                    <s:documentation>This basic type is used as the building
                    block for passing arguments to a service, and getting back
                    results.  This is a polymorphic type (dynamic type specified
                    as a MIME Content-Type in the Type field). It is optional
                    because it can  often be infererd from the OHSD
                    specification of this service.
                    </s:documentation>
                </s:annotation>
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="1" name="Name"
                        type="s:string"/>
                    <s:element minOccurs="0" maxOccurs="1" name="Type"
                        type="s:string"/>
                    <s:element minOccurs="0" maxOccurs="1" name="StringValue"
                        type="s:string"/>
                    <s:element minOccurs="0" maxOccurs="1" name="base64Value"
                        type="s:base64Binary"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="ArrayOfDataElement">
                <s:annotation/>
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="unbounded"
                        name="DataElement" type="tns:DataElement"/>
                </s:sequence>
            </s:complexType>
            <s:element name="Process">
                <s:annotation>
                    <s:documentation>Polymorphic array of elements for the
                    process message request.</s:documentation>
                </s:annotation>
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="1" name="ARGS"
                            type="tns:ArrayOfDataElement"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="ProcessResponse">
                <s:annotation>
                    <s:documentation>Polymorphic array of elements for the
                        process message response.</s:documentation>
                </s:annotation>
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="1"
                        name="ProcessResult" type="tns:ArrayOfDataElement"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="GetServiceDefinition">
                <s:annotation/>
                <s:complexType/>
            </s:element>
            <s:element name="GetServiceDefinitionResponse">
                <s:annotation>
                    <s:documentation>Can return no result, but should be a
                        result of type application/x-ohsd. </s:documentation>
                </s:annotation>
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="1"
                            ref="ohsd:OpenHealthServiceDefinition"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
        </s:schema>
    </wsdl:types>
    <wsdl:message name="GetServiceDefinitionSoapIn">
        <wsdl:part name="parameters" element="tns:GetServiceDefinition"/>
    </wsdl:message>
    <wsdl:message name="GetServiceDefinitionSoapOut">
        <wsdl:part name="parameters" element="tns:GetServiceDefinitionResponse"/>
    </wsdl:message>
    <wsdl:message name="ProcessSoapIn">
        <s:annotation>
            <s:documentation>This is the default style of SOAP message for
                remote service OpenHealth requests..</s:documentation>
        </s:annotation>
        <wsdl:part name="parameters" element="tns:Process"/>
    </wsdl:message>
    <wsdl:message name="ProcessSoapOut">
        <wsdl:part name="parameters" element="tns:ProcessResponse"/>
    </wsdl:message>
    <wsdl:portType name="OpenHealthServices_Basic_Soap">
        <wsdl:operation name="GetServiceDefinition">
            <wsdl:input message="tns:GetServiceDefinitionSoapIn"/>
            <wsdl:output message="tns:GetServiceDefinitionSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="Process">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
                Returns an array of Clients.
            </wsdl:documentation>
            <wsdl:input message="tns:ProcessSoapIn"/>
            <wsdl:output message="tns:ProcessSoapOut"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="OpenHealthServices_Basic_Soap"
        type="tns:OpenHealthServices_Basic_Soap">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="GetServiceDefinition">
            <soap:operation
                soapAction="http://www.RecordsForLiving.com/Schemas/
                    2006-08/OpenHealthServicesWSDL/GetServiceDefinition"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Process">
            <soap:operation
                soapAction="http://www.RecordsForLiving.com/Schemas/
                    2006-08/OpenHealthServicesWSDL/Process"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="OpenHealthServices_Basic_Soap12"
        type="tns:OpenHealthServices_Basic_Soap">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="GetServiceDefinition">
            <soap12:operation
                soapAction="http://www.RecordsForLiving.com/Schemas/
                    2006-08/OpenHealthServicesWSDL/GetServiceDefinition"
                style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Process">
            <soap12:operation
                soapAction="http://www.RecordsForLiving.com/Schemas/
                    2006-08/OpenHealthServicesWSDL/Process"
                style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```