

Meeting the Requirements of  
*Project HealthDesign:*

Comparative Analysis with Respect to  
Existing and Emerging Clinical Data Standards and  
Commercial PHR Data Repositories

Note: Comparative information presented in this document accurately reflects the data available on May 30, 2008. Modifications to standards and repositories subsequent to that date are not reflected in this document.

August, 2008

Prepared by  
Sujansky & Associates, LLC  
For  
*Project HealthDesign*



Robert Wood Johnson Foundation



CALIFORNIA  
HEALTHCARE  
FOUNDATION

Welcome from the *Project HealthDesign* Team  
Patricia Flatley Brennan, RN, PhD, FAAN  
National Program Director

Funded by the Robert Wood Johnson Foundation (RWJF), in collaboration with the California HealthCare Foundation, *Project HealthDesign* is a \$5 million national program of PHR systems. Administered by a national program office at the University of Wisconsin-Madison, *Project HealthDesign's* goal is to design and test a variety of PHR tools and applications that work together to help people achieve their various and specific health goals in an integrated fashion. The program is supported by the Foundation's Pioneer Portfolio, which funds innovative projects that can lead to breakthrough improvements in the future of health and health care.

Reaching this goal requires innovation in two key arenas – the applications themselves and the infrastructure needed to support the applications, including data standards and repositories. Fundamental to that innovation, we believe, is a design philosophy that separates applications from the infrastructure that supports them. Thus, the nine interdisciplinary grantee teams supported by *Project HealthDesign* created a wide variety of personal health records applications, tools that help people monitor their health status and make everyday health decisions. To support these applications, we derived a set of requirements and analyzed the ability of available resources (PHR platforms, health data standards, data structures) to meet those requirements. The requirements and their functional specifications can be found at [http://www.projecthealthdesign.org/media/file/Common\\_Platform\\_Requirements.pdf](http://www.projecthealthdesign.org/media/file/Common_Platform_Requirements.pdf)

We now present another in a series of documents designed to inform the PHR community about innovation in PHRs – the results of a gap analysis conducted to determine the ability of existing clinical data standards and commercial PHR repositories to meet the requirements identified by the developers of the *Project HealthDesign*-funded applications. Although existing data standards and PHR repositories meet many of the requirements of these applications, the analysis highlights specific features identified by *Project HealthDesign* that are not yet fully supported by these existing resources.

We invite the community of PHR developers and users, and health information technology vendors and researchers, to examine our findings. Through dialogue and discussion of this analysis, we hope to stimulate additional innovation in both personal health records and the infrastructure components needed to create a vibrant suite of information technology applications that support individuals as they seek to manage their health and health care more actively.

Please provide us your feedback at [http://www.projecthealthdesign.org/form\\_page2](http://www.projecthealthdesign.org/form_page2). Your comments will provide *Project HealthDesign* with valuable insight as we consider future program directions.

## Contents

1.	Introduction .....	4
2.	Industry Standards Relevant to PHR Interoperability.....	4
2.1.	HL7 PHR Functional Model.....	4
2.2.	HL7 RIM and Related Data Models.....	5
2.3.	ASTM Continuity of Care Record (CCR).....	6
2.4.	HL7 Continuity of Care Document (CCD).....	7
2.5.	HITSP Consumer Empowerment Interoperability Specifications .....	7
2.6.	AHIP/BCBSA Interoperability Specifications .....	8
2.7.	Calendaring Standards.....	9
2.8.	Terminology Standards.....	11
2.9.	Security Standards .....	12
2.10.	Device Interface Standards.....	14
3.	Software Initiatives Relevant to Common Platform Components .....	15
3.1.	Google Health.....	15
3.1.1.	Google Calendar .....	17
3.2.	Microsoft HealthVault.....	18
3.3.	Indivo / Dossia.....	21

## 1. Introduction

*Project HealthDesign* is a research initiative sponsored by the Robert Wood Johnson Foundation and the California HealthCare Foundation to develop innovative personal health applications and related technologies (see [www.projecthealthdesign.org](http://www.projecthealthdesign.org)). In addition to providing grant funding for the development of nine personal health applications, *Project HealthDesign* is exploring the role of *common platform components* as auxiliaries to these applications. Common platform components (CPCs) are shared software components that provide common data-storage and data-retrieval services, as well as means to readily and securely share personal health data.

As an initial step in the study of CPCs, *Project HealthDesign* elicited functional requirements from its nine grantees in four areas of platform functionality considered most useful: Medication management, observation recording, calendaring, and access control (see [www.projecthealthdesign.org/media/file/Common\\_Platform\\_Requirements.pdf](http://www.projecthealthdesign.org/media/file/Common_Platform_Requirements.pdf)). Based on these requirements, the project analyzed existing clinical data repositories, clinical data standards, and other relevant industry standards to determine their applicability as common platform components or parts thereof. The results of this “gap analysis” are reported here. If significant gaps exist between the requirements of the grantees and the capabilities of existing data repositories and standards, these gaps will need to be addressed before *Project HealthDesign* applications can use these resources as common platform components.

The functional requirements on which this gap analysis is based may not be common to all PHAs. However, the nine funded applications are sufficiently diverse that *Project HealthDesign* believes the results of the analysis will be of general interest to developers of personal health applications, clinical data repositories, and clinical data standards. To the extent that other PHAs share the requirements of the *Project HealthDesign* applications, an awareness of the gaps identified may help PHAs to select appropriate platform resources, as well as help guide the ongoing development of clinical data repositories and industry standards for PHAs.

It is important to note that this analysis focuses primarily on the *gaps* between the resources and the requirements of *Project HealthDesign*, and therefore omits mention of many features and capabilities that met the grantees’ requirements. Readers are advised to seek other sources or conduct their own assessments to get a comprehensive view of the analyzed data standards and data repositories.

The evaluation first considers a variety of clinical data standards that have been proposed for or may be applicable to achieving interoperability among PHAs. These standards include data models, functional models, and programmatic interfaces. The evaluation next analyses several health information technology initiatives that aim to serve as clinical data repositories for PHAs. These initiatives include Google Health, Microsoft HealthVault, and Indivo/Dossia.

## 2. Industry Standards Relevant to PHR Interoperability

A variety of industry consensus standards and proprietary *de facto* standards exist that are relevant to the structuring, coding, exchanging, and securing of health data within CPCs. This section surveys the most prominent among these standards and discusses their applicability to the development of shared common platform components for *Project HealthDesign* (PHD).

### 2.1. HL7 PHR Functional Model

The HL7 Personal Health Record Functional Model is a “reference” model of PHR functionality that was developed and officially published by a work group within HL7 in April 2008. The stated purposes of the functional model include “to promote a common understanding of PHR functions upon which developers, vendors, users and other interested parties can plan and evaluate PHR functions” and “to inform those concerned with secondary use of PHR data and national infrastructure what functions can be expected in a PHR System.”

Specifically, the HL7 PHR model specifies functions in three areas:

1. Personal Health functionality: Functions that manage information and features related to self-care and provider based care over time.
2. Supportive functionality: Functions that assist with the administrative and financial requirements associated with the delivery of health care
3. Information Infrastructure functionality: Functions that ensure that the PHR provides information privacy and security, promote interoperability between PHRs and potentially EHRs, and helps make PHR features accessible and easy to use

The HL7 PHR functional model is different than the PHD functional requirements because the former addresses the requirements of entire Personal Health Record Systems, including end-user functionality such as

- “the system shall provide the ability to print a current medication list”
- “the system shall employ a method to terminate sessions after a certain period of inactivity” and
- “the system shall display a privacy policy to users”

The PHD functional requirements, in contrast, are confined to “platform” functionalities related to data representation, data retrieval, system-to-system interoperability, and security. Although the HL7 model also lists requirements with respect to data retrieval, interoperability, and security in its “Information Infrastructure” section, it does so in a much more general way than the PHD platform requirements. For example, the HL7 requirements specify only that “recognized” terminologies and interoperability standards should be supported by PHR systems, but do not specify any particular terminologies or data models in order to achieve interoperability. The PHD functional requirements, in contrast, are at a level of technical detail that directly supports the design and implementation of functioning interoperable software.

At the same time, the HL7 PHR functional requirements are more comprehensive than the PHD requirements with respect to the full range of functionalities that a PHR may provide (i.e., beyond those expressed by the nine grantees of Project Health Design). For example, the HL7 requirements address the representation of drug allergies and medical procedures, as well as support for workflow. Therefore, the HL7 requirements can help to inform the design of the PHD platform components as those components are “generalized” beyond the needs of the PHD grantees, themselves.

## **2.2. HL7 RIM and Related Data Models**

The HL7 Reference Information Model (RIM) is a formal data model for health information that was defined for HL7 version 3. The RIM specifies at a very high level the data objects that health information systems communicate and store. For example, the RIM includes the data object “Observation,” which is intended to represent any patient-specific observation, such as a lab result, radiology report, diagnosis, documented medication allergy, etc. The RIM, itself, does not define a data structure for any of these specific observations, but the constructs of the RIM may be used to define such data structures.

In fact, HL7 itself is in the process of defining such structures for a variety of clinical domains (lab, pharmacy, etc.). These structures comprise “Domain Message Information Models” (DMIMs) and “Refined Message Information Models” (RMIMs) that augment the generic RIM with domain-specific semantics. However, few such detailed clinical structures are part of the HL7 standard to date. For example, detailed structures exist for medication prescriptions, but not for lab results, diagnoses, vital signs, and many other data types. Therefore, health care organizations and vendors who are using HL7 version 3 have had to define their own RIM-based structures to store and communicate many types of clinical data. This is particularly true in the United States, where no government-sponsored initiatives have yet created detailed data models based on the HL7 RIM (unlike in Canada, for example). The result in the U.S. is a diverse and overlapping set of data models that, while all based on the standard HL7 RIM, do not necessarily support interoperability.

For example, one vendor has developed an open-source PHR platform based on HL7 version 3. Although the vendor's data model is "conformant" with the HL7 RIM, the vendor has had to define its own detailed data structures to store objects such as "Diagnosis," "Allergy," and "Respiration Rate." Although these data structures are defined using the basic constructs of the RIM, they are not a part of the HL7 standard and are likely to differ from the RIM-based data structures developed by other vendors to store the same types of information. Hence, for many types of clinical data, the HL7 RIM does not yet support interoperability among health information systems.

### **2.3. ASTM Continuity of Care Record (CCR)**

The Continuity of Care Record (CCR) is a standard XML document specification for storing a "core" data set of the most relevant administrative, demographic, and clinical information about a patient's healthcare. The primary use case for the CCR, as stated in the specification, is to "provide a snapshot in time containing the pertinent clinical, demographic, and administrative data for a specific patient" to enable "one healthcare practitioner, system, or setting to aggregate all of the pertinent data about a patient and forward it to another practitioner, system, or setting to support the continuity of care." A secondary use case addresses PHRs: "A person may keep copies of his/her CCRs and supplement them, for example, with alternative medicine information and other personal health information. It should be noted, as well, that a person may also generate their own CCR."

The structure of the CCR contains the following clinical data components:

- Problems
- Family History
- Social History
- Allergies/Alerts
- Medications
- Immunizations
- Vital Signs
- Results
- Procedures
- Encounters
- Functional Status
- Plan of Care
- Payers
- Medical Equipment
- Advance Directives
- Health care providers
- Non-professional care givers

The CCR is designed to store the "pertinent" and "core" data about a patient's health at one point in time and at a level of abstraction that can support the transition of care from one health care setting to another. As such, the CCR document standard is explicitly not intended to be a standard data model for a health record platform, per se. Although certain PHRs have used the CCR as the basis for their data models (notably, Google Health – see Section 3.1), the CCR has a number of gaps with respect to the platform requirements elicited from the PHD grantees:

- The CCR standard defines no application programming interface to the contents of a CCR document, such as the operations available to create, retrieve, update, and delete the components of a CCR. As far as the standard is concerned, CCR documents are intended to be created and retrieved as entire documents only. This model could prove unsuitable for a PHR data repository platform because
  - (1) PHAs would have to retrieve entire CCRs when only a portion of the record was required for an operation, a requirement that could be quite inefficient if the CCR becomes large. For example, an application that needed only to display a patient's medication list would also have to retrieve the full set of daily glucose measurements,

blood pressure measurements, accelerometer measurements, and all other data that appeared in the CCR.

- (2) If optimistic concurrency controlled were used, PHAs would have to check whether any part of an entire CCR had been updated when they modified even a small part of the CCR. If pessimistic concurrency control were used, a PHA would have to lock an entire CCR to update any data within it, which would prevent any other PHA from accessing the CCR during that time. Both of these constraints are inefficient relative to a more granular model of health data.
- The CCR includes no data structures for versioning documents or part of documents as updates are made. This gap would complicate concurrency control as well as the maintenance of historical versions of the patient record.
- The CCR includes no mechanism to indicate that a portion of a document previously created has been or should be corrected.
- The CCR does not require that an entire document or any specific entry within a document (e.g., a medication, a condition) carry a globally unique identifier (although it does “recommend” that the identifiers be globally unique). The absence of this requirement may prevent the reliable association of information between CCR documents created by different PHAs (e.g., referencing a previous prescription in the record of a medication refill event), and it may prevent restricting access to individual CCR data entries that a patient considers highly confidential.
- Binary multi-media attachments (images, voice, video) are not allowed as parts of a CCR record (although externally stored multi-media files may be referenced from within a CCR).

On the positive side, the CCR standard does contain a much broader array of clinical data types than that defined for the PHD platform. For example, the following data types are defined in CCR but not the PHD platform components: allergy, social history, family history, immunizations, procedures, medical equipment, and plan of care. To the extent that these data types are ultimately needed in the PHD platform components, their CCR definitions could be used as the basis for modeling them.

#### **2.4. HL7 Continuity of Care Document (CCD)**

The Continuity of Care Document (CCD) is essentially an alternative version of the CCR, modeled using the constructs of the HL7 Clinical Document Architecture (CDA) and the HL7 RIM. Like the CCR, the CCD is intended to standardize a document structure for communicating core patient data during a transition of care settings. The high-level structure of the CCD is virtually identical to that of the CCR, and contains the same “header” information and clinical data components (see Section 2.3).

The CCD, therefore, has many of the same gaps as the CCR does with respect to the PHD platform requirements. An exception is that each entry in a CCD document (medication, problem etc.) must be identified by a globally unique “instance identifier.” This requirement allows CCD entries to be referenced from other CCD documents, something that the CCR does not guarantee. Another distinction is that the CCD allows the designation of certain individual data entries as “confidential,” which may be used by applications to restrict access.

Beyond these differences, the CCD manifests greater complexity than the CCR, because the modeling constructs of the HL7 RIM on which it is based are, themselves, relatively complex. Although this complexity is not at variance with the PHD functional requirements, *per se*, it would increase the level of training and the level of tooling required to integrate a PHA with a data repository based on the CCD standard.

#### **2.5. HITSP Consumer Empowerment Interoperability Specifications**

The Health Information Technology Standards Panel (HITSP) is a consortium of public and private organizations convened in 2003 by the federal government for the purpose of selecting and creating

standards to promote interoperability among health information systems in the United States (see [www.hitsp.org](http://www.hitsp.org)). Since its inception, HITSP has released a set of “interoperability specifications” for seven use cases in health data interoperability. One of these specifications is the “HITSP Consumer Empowerment Interoperability Specification (IS-03)” which defines specific implementations of established standards for the use case of exchanging a consumer’s personal health record information. The specific tasks that drive the requirements of this specification include:

- Ability for the consumer to retrieve, store, graph and share laboratory test results
- Ability for consumers to retrieve and store lists of current and previous health conditions, lists of current medications, and lists of diagnosis codes
- Ability for PHR systems to accept “batch” data from other organizations and match to the appropriate consumers
- Ability for a consumer to identify those providers who are permitted to access information in the consumers’ PHR, and which of those data they are permitted to access
- Ability for a consumer to request, consolidate, and access audit log information

To support these tasks, HITSP’s IS-03 amalgamates numerous existing standards and adds additional constraints and extensions where needed to support the consumer-empowerment use case. These standards include:

- HL7 CCD
- IHE XD\*-Lab Profile, based on HL7 CDA
- IHE Patient ID Cross-Referencing (PIX) transaction standard, based on HL7 v2.5
- The SNOMED-CT, LOINC, and UCUM terminology standards
- Various security standards, including WS-Trust, WS-Federation, SAML, and XACML

All of these standards, many of which were developed independently, are cited in the IS-03 specification as part of the solution for PHR interoperability. However, the specific manner in which the standards should be combined to create functioning interoperable systems is not adequately described in the specification. For example, the use of the security standards in conjunction with the clinical document standards is not described and, to our knowledge, has not been defined in any detail. In this regard, the designation of various existing standards will not result in interoperability among PHRs even if the entire industry implements these standards, because the standards themselves and the way in which they may be combined still allow for considerable latitude and heterogeneity.

Additionally, the HITSP Consumer Empowerment specification is intended to address the *exchange* of data relevant to PHRs, not necessarily the representation of such data in PHR “platform” components, nor the functionality that such platform components should provide to personal health applications that may use them. For example, IS-03’s reliance on the document-oriented models of the CCD and CDA present many of the same issues for a platform component as cited in the discussion of the CCR model above. Most importantly, a document-oriented standard does not provide a programmatic interface for accessing just portions of a patient’s health data stored within one or more documents, or for updating entries within a patient’s health record efficiently.

## **2.6. AHIP/BCBSA Interoperability Specifications**

America’s Health Insurance Plans (AHIP) is a national trade association for health insurers. In 2007, AHIP partnered with the Blue Cross Blue Shield Association (BCBSA) to publish an interchange standard for transferring PHR data between health plans (see <http://www.wpc-edi.com/products/publications/X271>). The goal of the standard is to enable health plan members to easily transfer the contents of their insurer-hosted PHRs when they switch health plans, i.e., without losing historical data or needing to re-enter data.

Note that the use case for the AHIP/BCBSA interoperability specifications is confined to the *transfer* of PHR data *between health plans*. Specifically, it is not intended for data transfer between insurer-hosted PHRs and other types of clinical information systems, such as EHRs, lab systems, pharmacies, or independent commercial PHRs. Also, the AHIP/BCBSA specification is not intended to standardize the structure or the functionality of PHR data repositories (platforms), nor the authentication, access-control, and other identity-management capabilities of such repositories.

Given its intended audience and use case, the current version of the AHIP/BCBSA specification is defined using constructs of the ASC X12 275 (claims attachment) standard, which are similar to neither the HL7 nor CCR formatting rules. Also, the current version of the specification addresses the transfer of only those data generally available to health insurers. These data are:

- Patient demographic information
- Encounters
- Prescription medications
- Providers or Facilities
- Health plan information
- Subscriber information

However, AHIP and BCBSA, in cooperation with HL7, are in the process of augmenting this data set with additional *clinical* data types based specifically on the HL7 CCD model (see Section 2.4). While consistent with the CCD, these data types will be further refined with respect to field requirements, vocabularies and value sets. An HL7 project team is currently pursuing this work, which will be balloted in HL7 later in 2008. At the time that these clinical enhancements to the AHIP/BCBSA specification are completed, the resulting specification will warrant a second look by *Project HealthDesign* as a resource for sharing data among personal health applications. Nevertheless, certain of the caveats regarding the use of the CCD for PHR platform components (as noted in Section 2.4) may still apply to this derivative work of the CCD.

## 2.7. Calendaring Standards

The PHD platform requirements include specifications for the representation of calendar objects (events, tasks, and reminders) as well as for a set of operations to create, read, update, and delete such objects. These specifications were developed to fulfill the specific requirements of the PHD grantees and were not necessarily based on any existing industry standards. At the same time, non-proprietary industry standards do exist for calendar data (such as *iCalendar*) and calendar APIs (such as *WCAP* and *CalDAV*). This section discusses the overlap and gaps in these industry standards with respect to the PHD grantees' requirements.

### **iCalendar (iCal)**

iCal is a data-representation standard for calendar data, specifically calendar events (“VEVENT” in iCal), calendar to-do items (“VTODO”), and calendar reminders (“VALARM”). iCal was developed by the Internet Engineering Task Force (IETF) over 10 years ago, and is widely used by calendaring applications today to exchange individual calendar objects (e.g., an appointment) and/or entire calendars (e.g., a user's entire set of events, to-do items, and reminders). The iCal syntax consists of text delimited by colon (“:”) and end-of-line characters. An XML version of the standard exists (“xCal”), which represents exactly the same content using XML syntax.

iCal is a domain-independent standard that contains no features specific to health care or personal health records. As such, the built-in structure lacks many of the data elements defined for calendar events, calendar tasks, and calendar reminders in the PHD platform calendar component. Examples of these missing data elements appear below:

- Calendar Events (VEVENT)
  - PatientRecordID
  - EventType
  - Reference links to non-calendar objects (e.g., observations and medications)
  - Reference links to data-entry forms
  - Version number (to store past versions of updated events)
- Calendar Tasks (VTODO)
  - PatientRecordID
  - TaskType
  - Reference link to observations
  - Reference link to data-entry forms
  - Version number (to store past versions of updated tasks)
- Calendar Reminders (VALARM)
  - Reminder UniqueID (reminders must be *part* of VEVENT and VTODO objects)
  - PatientRecordID
  - ReminderType
  - ReminderMessageSource
  - Reminder “Status” values (“set”, “executed”, “acknowledged”)
  - Version number (to store past versions of updated reminders)

(A full comparison of the differences between the iCal standard and the calendaring requirements of the PHD project appear in the appendix to this document.)

Nevertheless, the iCal/xCal standard allows one to define additional properties (“experimental properties”) for each of these object types within a specific implementation of the standard. This extensibility enables one to augment iCal/xCal with the required but missing data elements, so as to create an iCal-compliant “PHR calendar data standard” that meets most of the data-representation needs of the PHD grantees. At a minimum, calendar objects stored in the platform-component calendar defined by *Project HealthDesign* could be converted to iCal data structures with no loss of information (provided that the required experimental properties were part of the iCal conversion).

### **WCAP and CalDAV**

An appropriate data representation meets only part of the calendaring requirements of the PHD platform. The PHD personal health applications also require an application programming interface (API) for reading and writing calendar data that are stored in a platform repository. At least two standards exist for such an API: *WCAP* and *CalDAV*.

*The Web Calendar Access Protocol (WCAP)* was developed by Sun Microsystems for its Sun Java Calendar Server. Although WCAP is proprietary to Sun, it has been used in non-Sun open-source calendar projects, so its licensing status is somewhat unclear. WCAP uses simple HTTP GET commands for writing, reading, and querying iCalendar data objects. WCAP responses are either the traditional text form of iCal objects or an “xml-ized” form (xCal). Several WCAP plug-ins exist for mail client applications, including for Mozilla Thunderbird, Novell Evolution and Microsoft Outlook. The API calls in WCAP offer most of the features specified for the PHD calendar component. The following functions are, however, not supported by WCAP:

- Retrieving calendar events, tasks, and reminders by their *type*
- Retrieving calendar events, tasks, and reminders by their *status*
- Retrieving calendar reminders by their unique IDs
- Retrieving calendar reminders by Patient IDs

*The Calendaring Extensions to the Web Distributed Authoring and Versioning (CalDAV)* protocol was developed by the Internet Engineering Task Force (IETF) in 2007 as a non-proprietary standard for

accessing, managing, and sharing calendaring and scheduling information based on the iCal format. Like WCAP, CalDAV defines an HTTP-based protocol for writing, reading, and querying calendar objects; the results of these operations are iCal-formatted events, tasks, and reminders. Like WCAP, CalDAV provides many of the standard calendar functions required of the PHD platform, but seems to lack certain query capabilities, such as retrieving calendar objects by their type, status, or Patient ID.

Although neither WCAP nor CalDav meet all of the functional requirements of the PHD platform calendar component, work-arounds and customizations may exist such that a calendar service based on these standards would be adequate for the PHD project grantees. A more detailed analysis is required to determine whether either or both WCAP and CalDAV are adequate as the basis of such a calendar service.

## 2.8. Terminology Standards

Coding systems and terminologies are important components of PHR systems that intend to provide decision-support and data-analysis capabilities. Without a predefined, controlled coding system for medical concepts, such as medications, vital signs, and lab tests, PHRs can provide little more than a storage system for health records and a text-based search capability. Functions such as drug-drug interaction checking, alerts for high glucose or blood-pressure readings, and customized diet and exercise advice for diabetics require the constrained data representation that controlled terminologies provide. Additionally, the sharing of patient data among multiple PHR systems that provide such functions requires not only that each system uses a controlled terminology, but that all of the systems agree to use the same *standardized* terminology.

The PHD platform requirements specify (optional) terminology standardization in a number of areas, including medications, signs and symptoms, lab tests, vital signs, food ingredients, and physical activities. The good news is that these terminology requirements are largely fulfilled by two non-proprietary standard terminologies: RxNorm and SNOMED-CT. However, certain gaps exist in these terminologies that will require either extending the terminologies through their respective standards organizations, using alternative terminologies for certain medical concepts, or developing customized “Project Health Design” codes.

The tables below show how RxNorm and SNOMED-CT accommodate most of the coding needs of the PHD Medications and Observations components, as well as where gaps exist.

Table 1. Standard Terminologies for Medications Component

Num	Coded Object	Terminology/Version	Type or Root Concept	Example Code(s)	Example Text
1	Drug Name + Str + Form (Gen)	RxNorm	Type = SCD	RXCUI = 315025 RXCUI = 197774	Cetirizine 5MG Oral Tablet Acetaminophen 500MG / Hydrocodone 5 MG Oral Tablet
2	Drug Name+ Str + Form (Brand)	RxNorm	Type = SBD	RXCUI = 315025 RXCUI = 210776	Cetirizine 5MG Oral Tablet [Zyrtec] Acetaminophen 500MG / Hydrocodone 5 MG Oral Tablet [Vicodin]
3	Drug Name + Form (Gen)	RxNorm	Type = SCDF	RXCUI = 371364 RXCUI = 370641	Cetirizine Oral Tablet Acetaminophen / Hydrocodone Oral Tablet
4	Drug Name + Form (Brand)	RxNorm	Type = SBDF	RXCUI = 367925	Cetirizine Oral Tablet [Zyrtec]
5	Dosage Form	RxNorm	Type = DF	RXCUI = 317541	Oral Tablet
6	UnitsOfMeasure	SNOMED-CT (Intl 0707)	Root = "Unit" (258666001)	258684004 258798001	mg mg/mL
7	Dosing Frequency	SNOMED-CT (Intl 0707)	Roots = "Regular Frequency" (307430002) = "Irregular Frequency" (307485003)	229799001 225761000	twice a day PRN
8	Route of Administration	SNOMED-CT (Intl 0707)	Root = "Route of Administration Values" (284009009)	26643006 78421000	Oral route Intramuscular
9	NDC Codes	RxNorm	Attributes of "SCD" and "SBD" concepts in RxNorm file "RXNSAT.RRF" (normalized, RXNORM asserted)		

Table 2. Standard Terminologies for Observations Component

Num	Coded Object	Terminology/Version	Type or Root Concept	Example Code(s)	Example Text
1	Medication IDs	See # 1 - 4 in Medications			
2	UnitsOfMeasure	See # 6 in Medications			
3	Med Route of Administration	See # 8 in Medications			
4	Physical Activity	SNOMED-CT	Roots = "sport" (415577004) = "walking" (418060005) = "therapeutic exercise" (51998003) = "recreational therapy" (42364006)	20461001 129006008 404928000 229224000	Swimming Walking Pilates Yoga (Missing: rowing, weight lifting,...?)
5	Time Units	SNOMED-CT	Roots = "unit of time" (258695005) = "unit of length" (258667005)	258701004 258678002	minute mile
6	Exercise intensity value / units	PHD or local terminology	n/a	150 "medium" 20	{Max HR achieved} { "low", "medium", "high"} {minutes before breaking sweat}
7	Sign/Symptom	SNOMED-CT	Root = "clinical finding" (404684003)	267036007 49218002 79890006	Shortness of breath Hip pain Lack of appetite
8	Pain Severity	PHD pain scale terminology	n/a	3 8	{1-10 integer scale}
9	Observable Parameter	SNOMED-CT	Roots = "observable entity" (363787002) = "laboratory test" (15220000)	271649006 33747003 27113001	Systolic blood pressure Blood glucose measurement Body weight
10	Parameter Value Data Type	Subset of HL7 value types	See HL7 version 2.5, Chapter 7, field OBX-2	NM SN ST CE	Numeric Structured Numeric String Coded Entity
11	Parameter Value Units	See # 6 in Medications			
12	Parameter Recording Context	PHD or Local Terminology	n/a		

## Gaps

*RxNorm* lacks two important features:

(1) a comprehensive mapping from NDC codes to *RxNorm* codes. This deficiency complicates decision support and reporting functions when medication data (especially pharmacy dispensing data) is provided with NDC coding alone (the usual situation for pharmacy dispense data and pharmacy claims data). Specifically, many decision-support and reporting functions require that NDC product codes be abstracted to the clinically oriented *RxNorm* codes, which requires a reliable and complete mapping. Ideally, the National Library of Medicine (which maintains *RxNorm*) will expand its efforts to build and maintain an accurate and comprehensive mapping between NDC and *RxNorm* (although this is not a trivial task).

(2) a therapeutic classification hierarchy, which also complicates decision support and reporting functions that require the abstraction of specific medications to classes such as “hypoglycemic agents” or “oral corticosteroids.” Deriving therapeutic classifications for *RxNorm* codes currently requires mapping *RxNorm* to other drug terminologies in the Unified Medical Language System (UMLS), but a number of such terminologies exist in the UMLS, each with different classification hierarchies, so no “standard” therapeutic classification hierarchy currently exists as part of *RxNorm*.

*SNOMED-CT* also lacks a number of features that are required for the PHD platform components:

(1) certain physical activities that are relevant to PHRs (e.g., “rowing”, “weightlifting”).

(2) certain foods and food ingredients that are relevant to PHRs

(3) a standardized grading scores for exercise intensity and pain severity

(4) a comprehensive mapping to LOINC lab test codes. As with NDC codes for pharmacy prescriptions, standard LOINC codes are increasingly used by labs to report test results. Again, many decision-support and reporting functions require that the very detailed LOINC codes be abstracted to more clinically oriented SNOMED-CT codes, but no standard, non-proprietary mapping between LOINC and SNOMED-CT yet exists.

*RxNorm* and *SNOMED-CT* are potentially very useful resources for PHR platform components if the relatively few gaps identified here are addressed.

## 2.9. Security Standards

The security aspects of the PHD platform components address verifying that users are who they say they are (authentication), allowing personal health applications to securely access multiple platform components with minimal end-user inconvenience (single sign-on), and enabling patients to control who can access their personal health data (access control). In the realms of authentication and access control for web services, a host of industry standards have been developed over the past 10 years by organizations such as the World Wide Web Consortium (W3C), the Organization for the Advancement of Structured Information Standards (OASIS), and the Internet Engineering Task Force (IETF). Most of the standards that these organizations have developed are relatively mature and widely used in a variety of domains that employ web services, including health care.

The specific standards that are relevant to the security requirements of the PHD platform are:

*WS-Security*: Defines how to extend SOAP messages to enable secure Web services. Specifically, WS-Security defines how to encrypt and digitally sign parts of the message. Encryption and digital signatures can be applied to a header entry, to the body, or to part of the body. Also, WS-Security defines how to add timestamps and security tokens to a message.

*WS-Trust*: Builds on WS-Security to add the ability to establish, assess the presence of, and broker trust relationships. It also defines methods for issuing, renewing, and validating security tokens.

*WS-SecureConversation*: Builds on WS-Trust and WS-Security to add the ability to establish security contexts between Web services (such as PHR platform components) and their clients (such as personal health applications). A security context can span a series of message exchanges, allowing the creation of an authenticated session and avoiding the need for repeated authentication during a session.

*Security Assertion Markup Language (SAML)*: SAML provides an extensible set of XML data formats to communicate identity and authentication information in a variety of environments, including Web services. Specifically, SAML defines how security tokens are formatted and how to exchange such tokens using the SOAP protocol.

*eXtensible Access Control Markup Language (XACML)*: Expresses authorization policies (i.e., access-control rules) in XML against objects that are themselves identified in XML.

Collectively, these standards support secure authentication, single sign-on, and complex access control policies, and they have been demonstrated to work together. The existing implementation of the PHD platform, however, does not employ these standards because their use would have significantly increased the effort to implement the platform components and to integrate the platform components with the grantees' prototypes. Nevertheless, these are proven standards that are supported by a variety of proprietary and open-source computing platforms (e.g., Apache web server, Microsoft IIS, etc.) and have been used to implement robust security solutions in the past. With only a few exceptions (see below), they are likely to support the authentication and authorization needs of the PHD platform, and should be employed in future versions.

### Gaps

Existing access-control standards (such as XACML) do not adequately support the requirement that users have different roles with respect to different patients. Typically, users may be assigned to roles (usually called "groups"), but group membership spans all of the data in a data repository (rather than applying only with respect to a specific patient's data). This prevents a patient from specifying that only someone who is *her* physician may have access to her medication list, rather than any physician. In the absence of the ability to specify group membership in the context of a specific patient, patient's must either specify access-control rules redundantly for each of their physicians (in the preceding example), or allow all members of the physician group to access their data (whether their physician or not).

## 2.10. Device Interface Standards

A number of the PHD project grantees need to capture personal health data from electronic medical devices, such as blood glucose monitors and accelerometers. Although the PHD platform specifications include API calls that enable client applications to store data captured by a medical device, this process typically requires the data to first be transferred to a PC or cell phone, because most medical devices lack the computing capability to communicate directly over the internet using SOAP messaging protocols. Ideally, this connectivity between medical devices and PCs or cell phones would also be standardized across vendors, so that users could choose any such device to capture and upload their physiological data. To this end, two organizations have defined device-interface specifications for personal health applications: *Microsoft HealthVault* and the *Continua Health Alliance*.

### Microsoft HealthVault Connection Center

Microsoft has defined and implemented a desktop software application called “HealthVault Connection Center” (HVCC) [See [www.healthvault.com/WhatIsConnectionCenter.htm](http://www.healthvault.com/WhatIsConnectionCenter.htm)]. HVCC provides the ability for users of HealthVault to transfer data from a compliant medical device (such as a blood glucose monitor, electronic BP cuff, or exercise monitor) onto their desktop computer and then to easily upload the data to their HealthVault medical record. “Compliant” medical devices are those whose manufacturers have partnered with Microsoft to create device drivers that allow their device to communicate with the HVCC application when connected to a PC (these drivers are downloaded and installed by HealthVault users at the time they install HVCC or purchase the device).

Although HVCC has an “open” interface that enables manufacturers to develop device drivers that are compliant with HVCC, the interface itself and all related technologies and specifications were created solely by and are proprietary to Microsoft. Hence, HVCC represents a *de facto* standard (to the extent that it is adopted by medical device manufacturers), rather than an industry consensus standard. Nevertheless, for those PHD projects that may be interested in using Microsoft HealthVault as a platform data repository, the availability of HVCC and HVCC-compliant medical devices will greatly simplify the uploading of device data to HealthVault.

Note that Google Health does not yet offer a similar software capability. Any uploading of data from medical devices to Google Health accounts must be handled by the 3<sup>rd</sup>-party PHR applications linked with Google Health in an application-specific way.

### Continua Health Alliance

The Continua Health Alliance (CHA) is a consortium of 150 health care and technology companies whose stated mission is to “establish an eco-system of interoperable personal health systems that empower people & organizations to better manage their health and wellness.” [See [www.continuaalliance.org](http://www.continuaalliance.org).] CHA, first formed in 2006, has approached this mission through the specification of key interoperability standards and a testing/certification program to validate products’ compliance with these standards.

The initial set of standards that CHA will publish include a set of “device connectivity” standards<sup>1</sup>. These standards are intended to enable a variety of consumer medical devices to transfer collected data to patients’ personal computing devices (e.g., PCs, cell phones), where they may be incorporated into standalone personal health applications, uploaded to web-based PHRs, or transmitted to EHR systems. The standards encompass both the semantic structure of the data to be transferred (using the IEEE 11073 Point-of-Care Medical Device Communication Standards) and the technical means by which the data are transferred between devices (using the Bluetooth wireless communication standards and the USB wired communication standards).

Note that, unlike the HealthVault Connection Center, the CHA standards do not also address the uploading of health data from a patient’s personal computer or cell phone to a PHR repository such as HealthVault or Google. In this sense, the Connection Center standards currently provide more end-to-end

---

<sup>1</sup> The projected publication date of CHA’s first set of standards is Fall 2008.

functionality than those of CHA. At the same time, the CHA standards are non-proprietary, were developed through an industry consensus process, and are agnostic with respect to the device that may collect the data or the PHR repositories that may store the data.

### **3. Software Initiatives Relevant to Common Platform Components**

Within the past two years, several commercial software systems have emerged that provide clinical data storage and related services for personal health applications. Ostensibly, one or more of these commercial systems could serve as a common platform for the *Project HealthDesign* PHAs if the systems meet the grantees' functional requirements. The sections below compare the capabilities of various of these systems with the functional requirements for common platform components that the PHD grantees expressed.

#### **3.1. Google Health**

Google Health is a personal health record platform that consists of a data repository and an end-user interface<sup>2</sup>. The data repository consists of two distinct sections, one that stores the patient's *Profile* and one that stores *Notices* related to the profile.

The Profile is the patient's health summary, which represents the problems, medications, test results, immunizations, and other clinical data pertaining to the patient. Each patient has a single profile, which is represented as an XML document. Notices are other XML documents that a user or a 3<sup>rd</sup>-party application may upload to Google Health and associate with a patient profile. Notices may contain the initial "seed" data for a new patient profile or various updates to an existing patient profile, such as new lab results, prescriptions, immunization records, hospital discharge summaries, etc. Each patient's record may have many Notices and new Notices may be continually added. The data within notices are automatically imported into the patient's Profile and an "auto-reconciliation" process helps to prevent the addition of duplicate or inconsistent data from Notices. Users may also populate and update a patient Profile directly through the graphical user interface.

#### **Data Model**

The data within both Profiles and Notices are based on the ASTM Continuity of Care Record (CCR) (see Section 2.3). Each Notice must be a CCR-compatible document if it contains data destined for a patient's profile (although Notices not intended for the profile, such as notes from a patient's physician, need not be CCR documents). Each profile is a single document structured using a *subset* of the CCR, which we will refer to as "CCR-s" for purposes of this document. CCR-s, which was defined by Google specifically for Google Health, is a constrained version of the CCR that explicitly excludes many of the optional fields that are part of the CCR and imposes certain additional data-coding requirements that are not part of the CCR. For example, CCR-s does not include the optional "Advance Directives," "Encounters," and "Healthcare Providers" sections that are part of the CCR, although both CCR and CCR-s include Problems, Medications, Results, Immunizations, Allergies, Family History, and Social History. With respect to coding, CCR-s requires medications to be coded using RxNorm or NDC and vital signs to be coded using SNOMED-CT (other data coding requirements also exist).

CCR-s supports many of the data elements related to medications and observations that were specified by the *Project HealthDesign* grantees, although certain important data elements are absent. For example, CCR-s has no data elements to represent Medication Administration events, Physical Activity events, Meals, Journal Entries, or multimedia attachments such as images or voice recordings. For a detailed comparison of the *Project HealthDesign* clinical data requirements and the Google Health data model, see the appendix.

---

<sup>2</sup> This analysis was based on the version of Google Health available as of June 1, 2008.

## APIs for 3<sup>rd</sup>-party applications

Google Health provides external applications read-only access to the patient's Profile via the "gData" API, which is based on a standard XML-document retrieval interface called Atom. Google has created Java and C# wrappers for the gData interface, which greatly simplify application programming.

The available APIs allow an application to retrieve the entire Profile as a single document or retrieve specific categories of data from the Profile document (such as all medications or all medications called "Ibuprofen"). The Google API does not currently support all of the query parameters specified in the PHD functional requirements, such as date of entry, effective date or status (e.g., a medication's "active" vs. "discontinued" status). Also, only the gAtom API provides the ability to query profile documents and retrieve only portions of them – the query parameters to do this are not currently exposed in the Java, .NET, etc. APIs.

The APIs do not allow applications to write data directly to the Profile. 3<sup>rd</sup>-party applications write Notices, parts of which are then automatically imported into the profile, per the auto-reconciliation process. The reliability and thoroughness of this import process depends on the capabilities of the auto-reconciliation algorithms, something that we did not assess. The Notices themselves retain all of the data that an application has uploaded (in case certain information in a Notice is not imported into the Profile by the auto-reconciliation process), but there is no ability for external applications to read Notices, only to write them. Hence, the Profile is the sole source of patients' clinical data for 3<sup>rd</sup>-party applications.

For a detailed comparison of the *Project HealthDesign* API requirements and the Google Health API, see the appendix.

## Authentication and Access-Control

Each user of Google Health must have a general Google account (i.e., one that she may also use to access other Google services, such as Google Calendar, gMail, and iGoogle). With such an account, a user may create a Google Health account, which then allows her to create and manage one or more Google Health Profiles (and related Notices). Access to these Profiles subsequently requires password authentication by the Google account holder who created them.

Authentication is handled differently when a user logs directly into the Google Health portal and when a user accesses the Google Health repository via a 3<sup>rd</sup> party application. For portal access, the user logs into Google Health using her Google account name and password. This log-in must be performed each time the user wishes to access her Google Health account, but once logged in, the user may access other Google accounts that she holds (Calendar, gMail, iGoogle, etc.) with no further log in. In this sense, access to Google Health via the Google portal confers "single sign-on" across all of Google's other services. Note that the converse is not true – if logged into gMail, for example, a user must still submit her password to access Google Health, a prudent security feature. However, the ability to use one account across multiple Google services could have unintended consequences if a user happens to share or disclose her gMail or Calendar password, as the same password grants access to her entire Google Health record (users should probably be counseled to set up a different Google account for their health record).

When accessing Google Health via a 3<sup>rd</sup>-party application, authentication is handled differently. When a user first signs up with the 3<sup>rd</sup>-party application provider, the user is prompted to "link" her Google Health account with her account on the 3<sup>rd</sup>-party application. To authorize this linking, the user is redirected to a Google web page in which she enters her Google account name and password. This redirection to Google ensures that the account information is never disclosed to the 3<sup>rd</sup>-party application, a nice security feature that the PHD platform authentication process does not provide. Once a link is authorized by the Google account holder, the 3<sup>rd</sup>-party application receives a security "token" that henceforth enables it to *continuously* access the user's Google Health data, i.e. with no further requirement for the user to log into Google Health or even the 3<sup>rd</sup>-party application. This security token is similar to the "secure context token" issued by the PHD authentication service, except that the Google token never expires (the PHD token expires in 24 hours, by default, and can specify different lifetimes). Unless the user explicitly revokes the Google token, the 3<sup>rd</sup>-party application has access to the user's Google Health data continuously and in perpetuity. An implication of this policy is that, if a hacker or

malicious process were to gain access to the 3<sup>rd</sup>-party application, it could use the tokens stored therein to retrieve data from all of the Google Health accounts that are linked to the application<sup>3</sup>. Although the same risk exists for applications that use the PHD platform, the finite lifetime of the PHD secure context tokens limits the number of users to whose accounts those applications have access at any given time.

Another difference in the authentication process when accessing health data via the Google portal versus a 3<sup>rd</sup>-party application is that single sign-on to all of a user's Google accounts (Health, Calendar, etc.) is not possible via a 3<sup>rd</sup>-party application. For an application to link to multiple Google services on behalf of a user, the user must separately authorize each link via a separate log in. However, because the links are thenceforth active until revoked, the 3<sup>rd</sup>-party application itself can subsequently provide single-sign-on access to all of a user's Google services. This advantage with respect to convenience is the flipside of the disadvantage with respect to security that is entailed by the continuous and perpetual data access enabled by links to Google Health accounts.

Access controls within the Google Health repository are limited. The only levels of access control available currently for a 3<sup>rd</sup> party application are (1) write Notices only, and (2) write Notices and read the Profile. No other granularity of access control is available, such as granting access to only certain types of data for only certain types of users through certain specific applications. This is a notable difference from the fine-grained control available through the PHD platform's access-control rules. For example, a user of Google Health cannot limit an application to access only problem list and lab results data nor prevent an application from accessing specific sensitive data items in a Profile (e.g., particular medications or conditions).

Also, there is no way currently for a user to share their Google Health data with other Google users through the Google Health portal. The only way to share data in a Google Health account with others is to allow a 3<sup>rd</sup>-party application to access the data, and then use that application's access-control capabilities to grant or deny access to other users of the application. Hence, Google effectively delegates all fine-grained access control to 3<sup>rd</sup>-party applications. As with the delegated access-control model of HealthVault (see below), this approach favors smooth and predictable operations of 3<sup>rd</sup>-party personal health applications over centralized privacy control for patients.

## **Calendaring**

Google Health does not currently include a healthcare-specific calendaring function. Google Calendar may be used as a general calendaring service (like Google Health and most of Google's other applications, Google Calendar includes a programmatic API). However, unless a user is accessing Google Health and Google Calendar directly via the Google Portal UI itself, single sign-on is not supported for 3<sup>rd</sup>-party applications (i.e., via the API). Also, explicit links between objects in the two applications are not supported, such as links between a medication prescription and the calendar events indicating the times at which the medication should be taken.

The following section evaluates more generally the concordance between the features of Google Calendar and the requirements for a calendar platform service as expressed by the PHD grantees.

### **3.1.1. Google Calendar**

Like Google Health, Google Calendar provides a repository for calendar events and an end-user interface to this repository. The repository may be accessed by 3<sup>rd</sup>-party applications that wish to store and retrieve calendar data on behalf of Google users. The data model of Google Calendar is a proprietary model that is not based on industry calendaring standards (such as iCal or xCal – see Section 2.7). The model allows the specification of calendar events (including repeating events) and calendar reminders. Calendar tasks (i.e., “to-do” items) are not yet supported.

Unlike the PHD calendar data model, Google calendar has no predefined types for calendar events, such as “medication-administration event,” “physical exercise event,” etc. A 3<sup>rd</sup>-party application could

---

<sup>3</sup> Note: If the 3<sup>rd</sup>-party application were operating in “enhanced security” mode, the malicious process would also have to access the application's digital certificate in order to connect to Google Health.

achieve the same capability, however, by defining multiple distinct calendars for a single user, with each calendar storing one type of calendar event. Alternatively, custom properties may be defined for calendar events, one of which could store the specific event type. A 3<sup>rd</sup>-party application could also use custom properties to store other domain-specific information defined in the PHD calendar model but not explicitly supported by Google Calendar (e.g., text annotations, links to related calendar, medication, and observation entries, and links to data-entry forms). Because the values of custom properties may not exceed 1024 characters, however, custom properties could not be used to store arbitrary structured data, something that the PHD model supports.

Like the PHD calendar data model, Google Calendar supports the specification of one or more reminders for a calendar event, and allows one to specify the schedule for issuing the reminder and the means for delivering it (i.e., email, SMS, or pop-up message). However, the scheduling of a Google Calendar reminder may be specified only *prior to the start-time* of the associated calendar event. The PHD calendar model allows reminders to be scheduled before or after the start time or end time of an event. This additional flexibility allows reminders that are only issued if the scheduled time window for an event has started or has ended without the user marking the event as completed (which may be less intrusive than a preemptory reminder in all cases).

The API is based on the same gData model as all other Google services, with object wrappers available for C#, Java, and other languages. However, the set of available API calls, itself, is limited relative to the PHD calendar API. Applications may selectively retrieve Google Calendar events only by their date/times and any contained text, whereas the PHD calendar model supports event retrieval by date/time, type of event, sub-type of event, event status (“scheduled”, “cancelled”, or “fulfilled”), and text label. An additional distinction is that all deletions and updates of Google Calendar events are destructive (such that earlier versions of events are no longer available via the API), whereas changes in the PHD Calendar model are non-destructive.

### **3.2. Microsoft HealthVault**

Like Google Health, Microsoft HealthVault is a personal health record platform that consists of a data repository and an end-user interface<sup>4</sup>. 3<sup>rd</sup>-party applications may read and write personal health records in the repository via programmatic interfaces (APIs) when granted appropriate access privileges. In addition, users may directly view summaries of their personal health records or delete specific data items via HealthVault’s “account manager” portal, again based on access privileges specified by the owners of the data. Beyond a general similarity, there are a number of significant differences between Google Health and Microsoft HealthVault.

#### **Data Model**

The clinical data in a HealthVault patient record consist of various pre-defined clinical data objects (“Item Types”) that are represented as XML structures. Currently, approximately 60 such types have been defined. Examples include many of the general types of data found in the Continuity of Care record (such as “Problem”, “Medication”, “Allergy”, and “Test Result”), as well as a number of more specific clinical objects (such as “Blood Pressure Measurement”, “HgbA1C Measurement”, “Insulin Injection”, and “Diabetic Profile”).

As distinct from the Google Health approach, data in HealthVault are not represented within “documents” that applications must access and manage as entire units, but rather as granular and discrete instances of the various ItemTypes. These data instances may be created, read, and updated independently by client applications. Hence, applications that use HealthVault can easily retrieve just selective pieces of a medical record, such as a medication list or set of glucose measurements. Also, there is no segregation of data into a “Profile” and “Notices” section; all data loaded by 3<sup>rd</sup>-party applications immediately become part of the comprehensive patient record, with no need for the import of data from Notices into the patient’s Profile.

---

<sup>4</sup> This analysis was based on the beta version of Microsoft HealthVault as of May 29, 2008.

The set of pre-defined ItemTypes support many of the data elements for medications and observations that were specified by the *Project HealthDesign* grantees, but certain gaps do exist. For example, only a single “Medication” object currently exists, with no distinction between prescriptions, dispense records, and ad hoc medication records. Microsoft indicated that a forthcoming release of HealthVault will include a richer data model for medications, which will bring it much closer to the Medications data model defined in the PHD requirements. With respect to observations, there are no ItemTypes to generally represent “SignOrSymptom”, “Meal/Snack”, “JournalEntry”, “Pain”, and “PhysicalActivity”. Also, there exists no general observation type “ObservableParameter.” Instead, HealthVault observations are represented by various specific Item Types, such as blood pressure, height, weight, vital signs, blood glucose, spirometer measurement, and lab test result. Aside from these predefined structures, ItemTypes for other observable parameters have not yet been defined and would require extension/customization of the HealthVault data model. Microsoft has indicated that HealthVault is working with its development partners on an ongoing basis to extend and refine the data model per its partners’ needs, while at the same time engineering forward and backward compatibility among different versions of the data model.

Like the PHD platform, HealthVault does allow patient data to be annotated by linking an Annotation item to a medication, condition, or other health data entry using the “RelatedItems” field, which is part of every data item. The RelatedItems field may also be used more generally to link instances of ItemTypes to each other for purposes of cross-reference or association, for example, to link a medication prescription with the record(s) of when/how the medication was administered. This field is quite similar to the “ReferenceLink” fields of the PHD platform’s Medication and Observation types.

Also, like the PHD platform, the HealthVault data model retains previous versions of data when updates or deletions are made (“non-destructive updates”), so a full audit trail of patient data state is available.

### **APIs for 3<sup>rd</sup>-party applications**

For programmatic access, HealthVault provides several options. A native “low-level” API exists, that consists of XML messages transmitted over HTTP (generally similar to Google’s gData approach). However, the syntax and semantics of the HealthVault API are not based on any industry standard, and therefore are unlikely to be supported by 3<sup>rd</sup>-party tools that facilitate programming (such as are available for standard WSDL or Atom interfaces). Hence, application programmers who wish to use the direct API must write their own code to generate and read the XML messages that communicate with HealthVault. However, Microsoft has created a higher-level API for the .NET platform, which “wraps” the XML over HTTP calls within .NET object classes. Similar wrapper classes for the Java and Ruby languages are now available from Microsoft’s “CodePlex” shared-development environment (e.g. see <http://www.codeplex.com/HealthVaultJavaLib> ).

All of these APIs allow 3<sup>rd</sup>-party applications to create, read, update, and delete the ItemTypes defined in HealthVault. In addition, the APIs provide a relatively rich query capability that allows applications to retrieve only certain types of data, to specify certain date ranges, and to apply a variety of other filters based on the XPath XML-query language.

Like the PHD platform, the API and data model allow multiple applications to access a patient’s record simultaneously, controlling conflicting data updates through optimistic concurrency control (Google Health uses the same technique, although, as mentioned, the granularity of data versioning in the Google model is an entire document rather than individual data items within the document).

In summary, the functionality of the HealthVault APIs is generally consistent with that of the Project Health platform.

### **Authentication and Access-Control**

Each user of HealthVault must establish a HealthVault *account*. An account entitles a user to create and manage one or more HealthVault patient records, as well as to gain access to other patient records if explicitly authorized. Users may access a patient’s HealthVault record either directly through the HealthVault account-management portal or through a 3<sup>rd</sup>-party application that uses HealthVault as its

data repository. Authentication is handled similarly for both modes of access, although access control is handled differently.

When a user logs into the HealthVault portal or into a 3rd-party application integrated with HealthVault, the user is authenticated using the Microsoft LiveID service (Note: since our original evaluation of HealthVault in May 2008, Microsoft now supports the use of several other identity-provider services, based on the OpenID standard). If a user is authenticating into a 3rd party application integrated with HealthVault, and has authorized that application to see some data in their record, then upon successful authentication, the HealthVault service creates a token which is passed through the user's browser to the application. The application then presents this token as part of subsequent requests it makes on behalf of the authenticated user. In addition, there is an application/server authentication step where the 3rd party application proves its identity to HealthVault by demonstrating knowledge of its private key. At this step, the application also provides a symmetric key, and is handed back a token. This token is passed back to the platform on subsequent calls made by the application, and the symmetric key is used to sign subsequent messages to ensure they originated from the authenticated app. This model is similar to that employed by the PHD platform components, except that the HealthVault User token only allows the application to call a single service – the HealthVault repository – rather than a set of services participating in a “secure context.” The latter approach is more flexible, because it provides a “single sign-on” capability for a personal health application, allowing it to access multiple platform services and data repositories without requiring separate authentication.

Access control is managed through the specification of “Permissions”, which are access control rules that define who can perform what operations on what datatypes from which application. These rules are conceptually similar to the rules in the PHD platform's Access Control Service. When a patient's health-record data is viewed through the HealthVault portal, access to the data is controlled entirely by the permissions that the manager of the patient record has defined. Specifically, based on these permissions, HealthVault will control whether a particular user has any access at all to a record, and if so, which actions on which types of data are allowed. This model is similar to that of the PHD access-control rules.

When a patient's health-record data is viewed through a 3<sup>rd</sup>-party application, the access-control model is different and is handled to a greater extent by the 3<sup>rd</sup>-party application than by HealthVault. Specifically, HealthVault and the 3<sup>rd</sup>-party application mutually agree only upon which operations and which data, *in general*, will be available to the application. These accessible data are divided into two sets (as determined by the 3<sup>rd</sup>-party application): *optional* and *mandatory*. A patient may selectively control an application's access to his optional data items by changing the permissions for those types of items using the account manager tool. This allows the patient to turn on or off access to an ItemType, although it does not allow the patient to select which users or groups of users of an application may see the item (the PHD platform's access-control functions, in contrast, do allow access control with respect to specific users and roles).

For *mandatory* data items, the degree of user control is even less. An application's permission to access mandatory data items applies across all patient records, and cannot be selectively modified (within HealthVault) by an individual patient for his or her record. For example, a patient cannot specify that he grants an application access to his lab results but not his medication list, if both are among the mandatory data items available to the application. The patient can only control whether the application will or will not have access to *all* of his mandatory data by authorizing such access when prompted by the application (this authorization step also requires LiveID authentication). Thereafter, all of the access that HealthVault has made available to the application will be available for that patient's record. The only way for a patient to reduce the application's access to his mandatory data is to revoke authorization altogether. Also, the only way for a patient to limit a specific user's access to his record is through whatever additional access-control mechanisms are implemented and enforced by the 3<sup>rd</sup>-party application itself. Hence, for *mandatory* data items, there is no way for a patient to specify permissions within HealthVault for the following types of access-control restrictions:

- No 3<sup>rd</sup>-party applications may automatically read or write data to my record, except for my physician's EHR, and that application may only write medication prescriptions and records of my office visits. (Note: this level of access control *would* be available for optional data items)
- Regardless of what 3<sup>rd</sup>-party application is used, only myself, my family members and physicians may read the contents of my record. Only my daughter and myself may write contents to my record. (this level of access control *would not* be available for optional data items either)

In contrast, the PHD access-control model centralizes within the access-control service itself all of the access-control rules pertaining to all data items within a patient record (regardless of what 3<sup>rd</sup>-party application is used to access the record). The HealthVault model distributes these rules among the HealthVault permissions database and the access-control facilities of the various 3<sup>rd</sup>-party applications that may access HealthVault. Upon allowing a 3<sup>rd</sup>-party application to access the mandatory items in his HealthVault record when the patient is not logged in (i.e., in *offline* mode), the patient must trust that application to show these data to only those users that the patient has specified and perform only that subset of the available operations that the patient has authorized – HealthVault will not enforce these constraints beyond limiting the application to the set of operations that were agreed upon when the application was first integrated with HealthVault (again, HealthVault will enforce constraints that the patient has specified with respect to *optional* data items, but the granularity of such permissions is limited). There exist trade-offs between these two models of access control in terms of centralized privacy control for patients versus smooth and predictable operations of 3<sup>rd</sup>-party personal health applications.

Additional differences: The HealthVault permissions do not support specifying individual data items for access control, for example to prevent family members from viewing a specific medication, while allowing physicians to view the entire medication list. Although HealthVault does allow the designation of individual data items as “Personal,” this creates a blanket embargo that withholds the items from all 3<sup>rd</sup>-party applications and all users. Also, HealthVault does not support role-based access control in the current version, i.e., no assignment of users to roles or groups is supported. Hence, access control rules must be specified separately for each user with whom the patient would like to share his record, rather than at the level of classes of users (“physician”, “family member”, etc.).

### **Calendaring**

The HealthVault data model and API currently include no facilities to support calendaring as specified in the PHD functional requirements. Microsoft expressed that it is looking to application partners to define and implement the calendaring functionality needed in PHR applications, which HealthVault may later support through the definition of appropriate “base types” of data (presumably, Events, Tasks, and Reminders).

### **3.3. Indivo / Dossia**

Indivo Health is a platform developed at Harvard Children's Hospital for storing and serving personal health data<sup>5</sup>. Indivo began as a research project, but was recently selected by the Dossia consortium as the software platform for its production PHR project. In many ways, the Indivo system and Microsoft HealthVault are similar from a technical perspective (although Indivo preceded HealthVault).

#### **Data Model**

An Indivo patient record consists of a collection of strongly typed XML documents, each of which is similar to an ItemType instance in the HealthVault model. For example, Indivo document types include “Encounter,” “Immunization,” “Medication,” “Problem,” and “Vital Sign.” This granularity of data is consistent with the various object types defined by the PHD platform components (“Medication”, “HealthcareEncounter,” “ObservableParameter,” etc.). Notable gaps in the Indivo data model as compared to the PHD platform include an absence of document types for signs and symptoms, pain, meals/snacks, physical activity, and medication administration. A document type does exist for

<sup>5</sup> This analysis was based on version 3.1 of Indivo Health as of May 27, 2008.

“Annotations,” although multi-media attachments are not supported for most document types (the sole exception we found was “Radiology Report,” which allowed image data to be included). The Indivo data model supports versioning of each document instance, which is presumably used as the basis for an optimistic concurrency-control model and for non-destructive modifications.

### **APIs for 3<sup>rd</sup>-party applications**

Programmatic access in the current version of Indivo Health (version 3) is provided via an XML-over-HTTP interface with a custom (non-SOAP) envelope that’s used for authentication. To facilitate programming, version 3 includes a Java API library. The next version of Indivo will implement a very different web-services API consisting of a simpler XML-over-HTTP protocol consistent with the “REST” model (the details of REST are not relevant here). The available API calls allow 3<sup>rd</sup>-party applications to create, retrieve, update, and delete document instances. When retrieving documents, the API provides a very basic querying capability that can selectively retrieve documents by (1) unique document ID, (2) semantic classification, (3) current, historical, or current+historical status, and (4) the values of certain fields. Note that the “classification” of a document is a semantic label that is different than its XML type (although a one-to-one correspondence exists in many cases). In comparison to the PHD platform’s query capabilities, the Indivo systems specifically lacks the ability to query by a date range, by a medication’s identity, by a medication’s dosing status (“Active” vs. “Discontinued”), and by an observation’s identity.

### **Authentication and Access-Control**

The Indivo authentication process is similar to that of the PHD platform. To authenticate, a 3<sup>rd</sup>-party application submits a valid username/password combination to the Indivo server, and receives an authentication “ticket” (token) in response. The ticket is then included with all subsequent interactions between application and Indivo on that user’s behalf. The application may request a ticket that is valid for a single server request only, that is valid until a specified expiration time, or that is valid indefinitely. Unlike the “secure context” model of the PHD platform, no single sign-on capability is supported across multiple services, as the Indivo platform comprises a single service only. The next version of Indivo will use a more sophisticated authentication methodology based on the “OAuth” standard, which will allow various modes of authentication, including biometrics and hardware tokens, and will support single sign-on.

The current version of Indivo provides a very rich model for specifying access control, based on the XACML standard for representing access-control rules. XACML is a very expressive and complex language for specifying access-control rules, but an Indivo researcher indicated that it may be too complex for users to understand and work with. The next version of Indivo will, in fact, no longer use XACML and will provide much less granularity of access control.

### **Calendaring**

Indivo includes no calendaring functionality at this time.