

Common Platform Components
For Personal Health Applications:
Accelerating Development,
Enhancing Interoperability, and
Improving Security

Project HealthDesign

Robert Wood Johnson Foundation

September 17, 2008

Prepared by
Sujansky & Associates, LLC

Contents

1.	Common Platform Components for Personal Health Applications	3
2.	Why a New Initiative?	4
3.	Architecture and Features of the Existing Software Components	6
3.1.	Registry Service.....	6
3.2.	Authentication Service	7
3.3.	Access Control Service.....	7
3.4.	Medications Service	8
3.5.	Observations Service	8
3.6.	Common Features Across Services	9
3.6.1.	Application Programming Interface	9
3.6.2.	Version Control and Audit Logging	10
3.6.3.	Concurrency Control.....	10
3.6.4.	Encryption	10
4.	Development Roadmap	11
5.	Additional Documentation.....	12

1. Common Platform Components for Personal Health Applications

Project HealthDesign, an initiative of the Robert Wood Johnson Foundation (RWJF) with additional support from the California HealthCare Foundation, has developed a set of software components that provide *common platform services* for personal health applications (PHAs). Common platform services are software functions that are common to many different types of PHAs. These services include user authentication, access control, and the storage of structured and coded clinical data. In the absence of pre-existing components that provide such services, each PHA project must design and implement its own sub-systems for authentication, access control, structured data storage, etc. This process results in unneeded additional work, diverse implementations that impede interoperability, and a complex overlapping set of security mechanisms for users. To overcome these inherent barriers to and limitations of PHAs, Project HealthDesign has developed an architecture for *common platform components* (CPCs) and has implemented an initial set of such components. The implementation is based on open, platform-independent web-services technologies, including Web Services Definition Language (WSDL), SOAP, and WS-* security standards. Although other initiatives have developed platforms for PHAs, the requirements of Project HealthDesign necessitated a distinct design and implementation (see Section 2).

The five components (or *services*) that have been implemented are listed in Table 1. Figure 1 of Section 4 shows how the components interoperate.

Table 1. Common Platform Components Implemented by Project HealthDesign

Component	Description
1. Registry Service	Stores demographic and password information for the users and applications that may access the CPCs, as well as for any patients whose data are managed by the components.
2. Authentication Service	Authenticates the identities of users and applications that wish to access the CPCs, and provides single sign-on across all of the components.
3. Access Control Service	Stores the rules that specify which resources and operations PHAs have access to. Enforces these rules when users request read and/or write access to specific patient data.
4. Medications Service	Stores and makes available the list of medications that patients take regularly. Relies on the Authentication and Access Control services to control access.
5. Observations Service	Stores and makes available health-related observations that are captured by or on behalf of patients outside of health care encounters. Relies on the Authentication and Access Control services to control access.

Eight PHA projects, supported by Project HealthDesign, are currently using these components for data-storage, interoperability, and/or security services. These projects address a variety of conditions, including diabetes, heart disease, chronic musculoskeletal pain, cystic fibrosis, and obesity. Feedback to date has indicated that the platform components operate reliably, integrate effectively, and meet the functional requirements of these varied projects. Feedback has also indicated that use of the CPCs by multiple PHAs enables the sharing of patient data among them.

Given the apparent utility of this approach, Project HealthDesign is considering publication of the common platform components under an open source license. In addition, Project HealthDesign is evaluating whether it may be constructive for an open-source development community to coalesce around these components and work to refine and extend them in a sustainable fashion. The goal of such a

community would be to research and publicize “best practices” related to platform components for PHAs, as well as to provide specific platform components under an open-source license.

2. Why a New Initiative?

A number of platforms and standards for PHRs have recently emerged. Two commercial systems now exist that offer data storage and authentication/access-control services for PHAs (Google Health and Microsoft HealthVault). A third system offers similar capabilities and is available as open source software (Indivo Health). A variety of industry standards have been proposed for representing and sharing personal health data (CCR, CCD, HL7 PHR Functional Model, HL7 RIM, AHIP Interoperability Specification, etc.). Why then did Project HealthDesign design and develop new platform software to support personal health applications?

The answer is that the existing resources and standards do not yet fully address the platform requirements of all PHAs. Specifically, nine PHAs under development within Project HealthDesign expressed certain requirements that were unmet by currently available systems and standards (see www.projecthealthdesign.org/media/file/Common_Platform_Requirements.pdf and www.projecthealthdesign.org/media/file/Meeting_the_PHD_Req_Comp_Analysis_8-15.pdf). It is unknown whether these requirements are universal, but they were sufficiently important to the Project HealthDesign applications to warrant addressing. The most important of these unmet requirements are listed below:

Modularity and Flexibility: The existing platform systems are monolithic and support a single integration architecture only. PHAs may be quite diverse with respect to their native capabilities, performance requirements, and security preferences. This diversity necessitates a platform architecture that is modular and flexible. For example, certain PHAs may choose to rely on platform components for registry, authentication, access-control, and data-storage services, whereas other PHAs may already have registry and authentication capabilities and require only access-control and data-storage components. Also, certain PHAs may have performance requirements that necessitate the tight integration of platform components as linked libraries, whereas other PHAs may prefer loose integration with remotely hosted web services. Lastly, certain PHAs may require that patient data reside on servers hosted within their own security domains, whereas other PHAs may accept the storage and sharing of patient data on servers hosted by 3rd parties. The modular architecture of the platform components implemented through Project HealthDesign is designed to accommodate a variety of configurations, integration strategies, and security preferences.

Centralized Fine-grained Access Control: Although the existing platforms provide some control over which users and applications may access a patient’s health data, the degree of control is relatively coarse. For example, the two commercial platforms allow users to specify only whether a 3rd-party application may access their health record at all; fine-grained control over which specific users of the 3rd-party application may perform which specific operations on which specific data is not supported. Without such fine-grained control, a patient is unable to specify constraints identified as important for Project HealthDesign applications, such as “any family member or medical professional may view my medication list, but only my physicians and I may edit the medication list, and only my sister and I may view or edit my health journal entries.” Although such fine-grained control may be provided by 3rd-party applications that use existing platforms (i.e., rather than enforced by the platform software itself), the delegation of such controls to various external applications can be more complex and less reliable than a centralized point of control for all health data access. The Project HealthDesign platform components, therefore, provide a centralized and fine-grained access-control model for patient health data.

Open Platform-Independent APIs: Although the existing platform systems offer programming interfaces for external applications, the available application programming interfaces (APIs) are either limited to a handful of programming environments (.NET, Java, and PHP) or require the programming of low-level XML-over-HTTP messages (a relatively complex process). Today, however, industry standards exist for

specifying APIs in platform-independent and language-independent ways that can automatically map to a wide variety of computing platforms and programming languages. This model for specifying APIs greatly facilitates interoperability between web-hosted platform components and a wide variety of applications and devices. The APIs of the Project HealthDesign components are specified using the best-known of these standards, the Web-Services Definition Language (WSDL).

Integrated Calendaring Functions: A number of use cases for the Project HealthDesign PHAs require integrated calendaring functions, including:

- Representation of detailed dosing schedules for medications, to help patients taking multiple medications and/or medications with complex dosing regimens to keep track of when and how to administer their doses.
- Representation of complex treatment schedules for specific illnesses, such as breast cancer, which consist of numerous tests and treatments over many months. A calendar-based representation helps patients to understand and follow long treatment regimens.
- Tracking of health-related goals and “to-do” tasks that have associated deadlines, such as the goal to exercise three times each week, the goal to receive a flu vaccination by a particular date, or the goal to record entries in a pain diary at least once per day (including reminders when needed).

None of the existing platform systems incorporate support for healthcare calendaring functions, such as a data model that explicitly represents calendar events and tasks or the relationships between calendar events and prescribed medications, recorded observations, and other calendar events. The Project HealthDesign functional requirements and API specifications explicitly support such calendaring functions.

Comprehensive Data Models: The data models of the existing platform systems are based either on industry standards designed for other purposes (e.g., the Continuity of Care Record) or on the requirements of a limited number of client applications. These approaches have resulted in models that lack certain important data elements required by the Project HealthDesign applications, and perhaps others. For example, the existing systems lack data types for physical activities, the composition of meals and snacks, and free-text journal entries. Two of the three existing platform systems also lack support for multi-media attachments, such as images and voice recordings. This proposal considers whether a more open and collaborative process for defining new data elements (with appropriate oversight to maintain consistency and order within the data model) could result in a more complete data model for personal health information.

Non-proprietary Interoperability Standards: None of the existing platform systems fully adhere to industry standards for data representation, APIs, or security mechanisms. With respect to data modeling, this variance is understandable, because no existing data standards adequately support the needs of personal health applications at this time. Specifically, the HL7 RIM, the HL7 PHR Functional Model, and the AHIP Interoperability Specification are too general to represent the clinical data that typically appear in personal health records. Other standards, such as the HITSP Consumer Empowerment Interoperability Specification, the ASTM Continuity of Care Record, and the HL7 Continuity of Care Document, are document-based and may not provide the granular data-representation and data-access model needed by advanced personal health applications. With regards to APIs and security mechanisms, the existing platform systems currently use models of their own design. In any case, no existing systems have implemented industry standards in a way that supports the flexible mixing and matching of personal health applications and common platform components, nor the sharing of patient data among common platform systems. The approach proposed in this document would seek to create a forum for the identification and widespread implementation of such standards, specific to personal health applications.

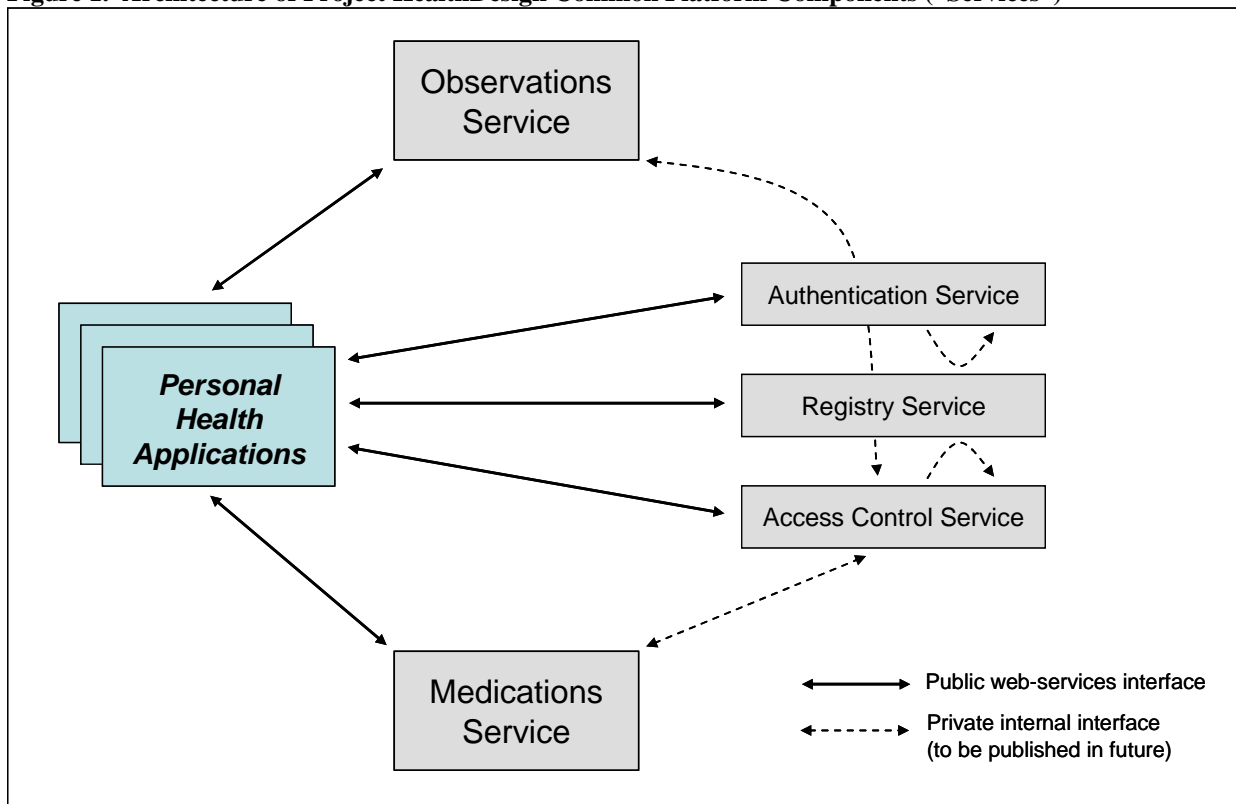
3. Architecture and Features of the Existing Software Components

The common platform components were designed and implemented as a suite of modular software components that interoperate through formal APIs. Each component interfaces both with PHAs and with other platform components, as illustrated in Figure 1. The public APIs (solid lines in Figure 1) allow PHAs to access the components over a wide area network as *web services*. See Section 3.6.1 for details of these APIs. The interfaces among the platform components (dashed lines in Figure 1) are not yet public, but will be made available as web-service APIs in a future release. The common platform architecture is an instance of a *Service Oriented Architecture (SOA)*

Each component is implemented as a Java software package that uses a MySQL relational database and runs on a Linux server (all technologies used in the implementation are available as open source software). If performance or security considerations preclude a web-services architecture, the source code would be available and could be compiled directly into Java-based applications. Alternatively, the components could be hosted on any other (non-Linux) server platform that supports the Java runtime engine (e.g., Windows).

The following sections describe each of the platform components. The subsequent section describes the web-service APIs and other features common to the components.

Figure 1. Architecture of Project HealthDesign Common Platform Components (“Services”)



3.1. Registry Service

The Registry Service maintains a master list of the applications and users that may access the platform components, as well as the patients whose data are managed by the platform components. The application and user records include authentication credentials, which are accessed by the Authentication service when a specific user attempts to log in via a specific PHA. The Access Control Service also consults the registry at the time new access-control rules are created, to validate that the users,

applications, and patients referenced in the rules are known to the common platform. Users and PHAs with appropriate access privileges may edit the data in the Registry Service, for example to create new users and new patient records.

A centralized registry (i.e., one that is independent of any specific PHA) allows users to access personal health data through various PHAs using a single account name and password and to share their data with other registered users, regardless of the applications they use.

3.2. Authentication Service

The authentication service validates the identities of the users and applications that are seeking access to the data and resources managed by the platform components. Successful authentication provides single sign-on across all of the components. Specifically, upon presentation of appropriate credentials by a requesting user and application, the service generates a security token that is sent to the requesting application and to each of the platform components. All subsequent requests from the application to the components reference this token in an encrypted manner known only to the authenticated party, thereby authenticating each request without requiring a separate log-in per platform component or per request. The security tokens automatically expire after a limited time (typically 24 hours).

The mechanisms for authentication and single sign-on are based on WS-Security and related standards, although the current implementation is not yet fully compliant with these standards.

3.3. Access Control Service

The Access Control Service stores and enforces rules regarding which users may access a patient's health data, which specific data these users may access, and what operations the users may perform on the data. The access-control rules support role-based, fine-grained access control. Several features of the Access Control Service are customized to the requirements of PHAs:

- Roles are assigned to users for specific patients only. For example, a user may be a Family Member with respect to patient record #1, a Physician with respect to patient record #2, and the Custodian of patient record #3. In conventional access-control systems, roles (or "group memberships") apply across all patient records, such that a user assigned to the Physician role may access the data of *any* patient that have been made available to physicians. This policy does not correctly reflect that users wish to share their data only with their own physicians. To address this need, the Access Control service in the Project HealthDesign platform requires that role relationships be assigned for specific patient records.
- The types of operations for which access may be granted are specific to the PHA domain. For example, the access-control system distinguishes between granting the privilege to update a patient's medication list, which the patient may wish to reserve for himself, and granting the privilege to *annotate* entries in the medication list, which patients may wish to confer to caregivers or family members. Also, the access-control system distinguishes between read access to the *current* contents of a patient's record and read access to the *historical* contents (which the platform maintains, for audit purposes).
- The access-control service reflects the clinical data model of the platform components. For example, access-control rules may specify that a user be granted read access to all data of the type "Observations," except for the sub-types "Journal Entry" and "Meal/Snack". Another set of rules may grant a user access to the prescriptions on a medication list, but not the records of dispensed medications nor the over-the-counter medications in use.

The Access-Control Service helps other platform components enforce the access rules at the time that PHAs request read/write access. Specifically, the Medications and Observations components consult the Access-Control Service to authorize each specific request, based on the user and application making the

request and the data and operations requested. The centralization of this service in a single component allows patients to specify and maintain a single set of access-control preferences that govern the sharing of their health data with multiple users across multiple PHAs.

3.4. Medications Service

The Medications Service is a data repository for managing the medications that a patient has been prescribed, has been dispensed, or is taking over-the-counter. The service consists of a data model for medication-list entries and an API for creating, editing, and retrieving the entries. The data model includes those discrete data elements relevant to personal health applications and supports the (optional) coding of medications and dosing instructions. The model is not based on any existing standard data structures or information models because none were deemed appropriate for a personal health medication list.

Noteworthy aspects of the Medications Service include:

- Support for linking related objects in a patient's personal health record, such as the medication dispense records that correspond to a medication prescription record, the calendar events that indicate the dates/times at which a medication should be taken, or the observation records that document when a medication was actually taken.
- Text descriptions of medication entries and their components are required, so that PHAs lacking a code-lookup capability may always display the information. The controlled coding of medications is also supported (albeit optional), so that PHAs that need coding for decision support or data analysis may be accommodated.
- Support for extending the medication entries with application-specific data elements, while maintaining a core set of data elements that all applications can process.
- Support for multi-media attachments, such as pill images.

3.5. Observations Service

The Observations Service is a data repository for managing health-related observations that a patient has recorded in the course of her daily life (i.e., outside of her interactions with the health care system). The service consists of a data model for various types of observations and an API for creating, editing, and retrieving these entries. The data model currently includes the following types of observations:

- **Medication Administration** (when and how a medication was taken)
- **Sign or Symptom** (e.g., fatigue, insomnia, rash)
- **Pain** (a sub-type of Sign or Symptom, with data elements specific to describing pain)
- **Observable Parameter** (e.g., blood glucose level, blood pressure, weight)
- **Healthcare Encounter** (e.g., hospitalization, office visit, chemotherapy session)
- **Physical Activity** (for logging various exercise sessions, their durations, etc.)
- **Meal/Snack** (for logging the composition and size of individual meals and snacks)
- **Journal Entry** (for text or multi-media journaling related to personal health and wellness)
- **General Observation** (a catch-all type for other observations, with extensible data elements)

The observations service includes many of the features noted above for the Medications service, i.e., support for linking objects, optional support for controlled coding, support for extending the data types, and support for multi-media attachments.

Note that other specific data types that are commonly part of personal health records are not yet available in the platform components, including diagnoses, medication allergies, surgical procedures, and family

histories. These types were not expressed requirements of the Project HealthDesign PHA projects, but may be added as the platform's development continues. In the meantime, the "General Observation" type may be used to record these types of data.

3.6. Common Features Across Services

Certain features and functionalities are common across all of the platform components, as described below.

3.6.1. Application Programming Interface

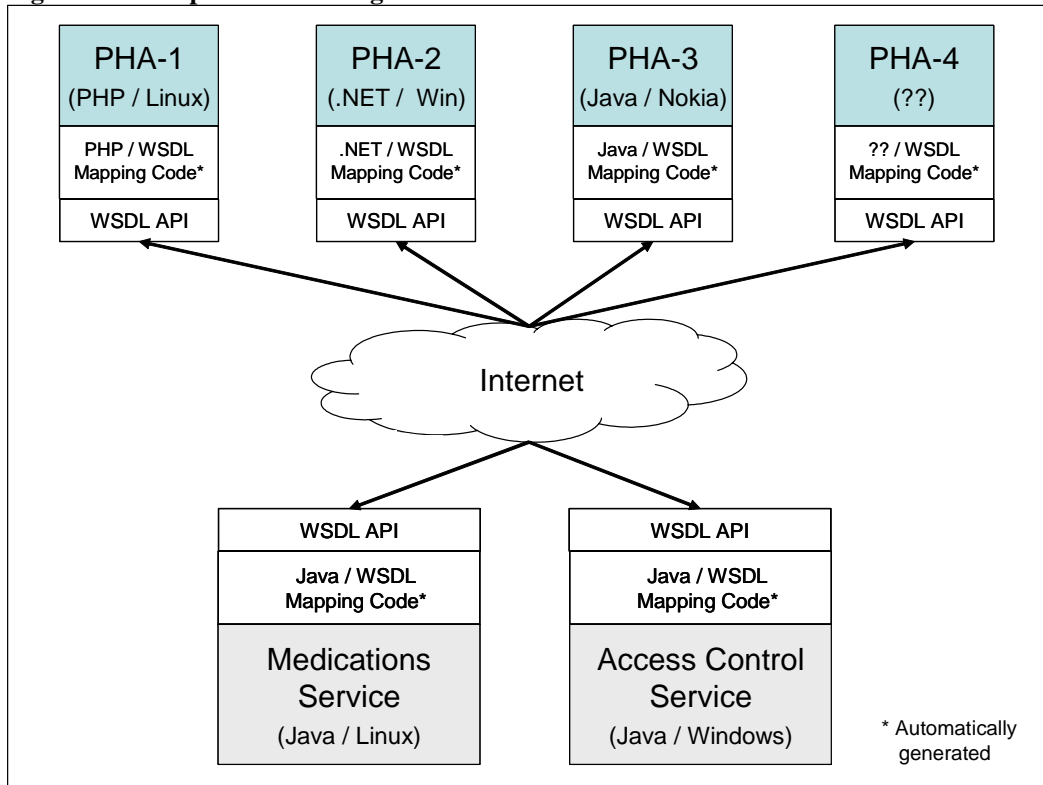
For web-based access, the platform components provide an application programming interface (API) defined using the *Web-Services Definition Language* (WSDL). WSDL is a non-proprietary, platform-independent language based on XML. WSDL specifications define the operations that each platform component provides (e.g., storing an observation, retrieving a medication, updating an access-control rule) and the data structures that these operations expect (e.g., the types of observations that may be stored and the required and optional data elements for each type).

PHAs communicate with the platform components by sending and receiving XML messages that contain requests for services and responses to these requests, formatted per the WSDL specifications. The WSDL of the common platform specifies the Simple Object Access Protocol (SOAP) as the messaging protocol. PHAs that wish to use the platform components may be implemented in any programming language and run on any computing platform, provided that they can generate, transmit, and receive the WSDL-specified SOAP messages.

To facilitate mapping between the SOAP messaging model and the native programming environments of PHAs, software utilities exist that *automatically translate* WSDL specifications into source code libraries for various programming languages and environments (including Java, .NET, and PHP). Programmers then use the generated code in these libraries to transparently convert Java or PHP method calls into SOAP messages and to transmit the messages over the internet. Because WSDL and SOAP are industry standards, these code-generation utilities exist for many programming environments.

Figure 2 illustrates the means by which PHAs that run on various computing environments can interface to the common platform components via the WSDL-defined APIs. Additional documentation for the WSDL APIs may be accessed from Section 5.

Figure 2. Example of interfacing PHAs and Platform Services via WSDL APIs



3.6.2. Version Control and Audit Logging

When PHAs update (modify or delete) patient health data stored by the platform components, the components maintain all of the previous versions of the data and make them available for auditing purposes or historical review. The current and historical versions are appropriately designated, and the temporal sequence of all versions is maintained. Further, for auditing purposes, the platform components record the user and application that requested each update, as well as the date it occurred.

3.6.3. Concurrency Control

The platform components allow multiple PHAs to access the same patient data concurrently. To prevent applications from inadvertently overwriting each other's updates, however, the components implement an optimistic concurrency control policy. This policy allows applications to update a data item only if the version that they originally retrieved is still the current version. If another application has modified that version in the interim, the updating application is required to retrieve the new (current) version before it can submit its own update..

This policy has the advantages of (a) ensuring that a user is aware of the current value of data prior to changing it, and (b) ensuring that, in updating part of a data item (e.g., a medication's dosing frequency), a user does not inadvertently undo an earlier update to a different part of the data item (e.g., a medication's refill allowance).

3.6.4. Encryption

Strong encryption of messages between PHAs and the platform components is currently provided at the transport level via Secure Socket Layer (SSL) and server certificates. As the platform components evolve and the WS-* security model is fully implemented, message-level encryption may be implemented as a replacement or alternative to SSL.

4. Development Roadmap

The existing implementation of the platform components provides (1) a reliable prototype system for securely storing and sharing personal health data, (2) a robust test bed for exploring the optimal design of platform services, and (3) a solid foundation for the ongoing development of platform software for personal health applications. However, further work remains before the platform can fully meet the requirements of a broad range of PHAs. The areas for further applied research, technical design, and software development include (but are not limited to):

- Completion of the Calendar Service, which has been specified but not yet implemented.
- Design and implementation of additional platform components and data types. Examples may include data types for representing and managing problem lists, medication allergies, past surgical procedures, and family histories, as well as platform components for performing drug-drug interaction checking or providing patient-education materials.
- Enhancement of the authentication service to fully implement WS-Security and related standards. A fully compliant implementation will better support integration with additional or alternative platform components provided by other initiatives.
- Extension of the authentication model to support credentials beyond simple passwords (such as smart cards, hardware tokens, biometrics, etc.)
- Implementation of audit logging for all data retrieval (currently, the platform components log data updates only, i.e., the creation, modification, or deletion of records).
- Specification and implementation of public WSDL APIs *among* the platform components (to enable architectures in which certain of the platform components are substituted with a PHA's own capabilities or with software modules from third parties). For example, a WSDL API specification for the Access Control Service would allow a PHA's own medication-list component to consult the centrally stored access control rules.
- Specification of public Java interfaces for all of the platform components (the Java interfaces are currently internal only). Publication of the Java interfaces will enable compilation of some or all of the platform components directly into Java applications. This capability would enable use of the platform component modules even when the performance overhead of a web-services architecture is prohibitive.
- Specification of mandatory coding systems for certain data elements as a part of the platform data model. Stricter coding requirements would enhance interoperability among PHAs that provide decision-support or analytic capabilities. For example, mandatory coding standards may include use of RxNorm for coding medications or use of SNOMED-CT for coding Signs and Symptoms.
- Development of user-friendly utilities for creating and reviewing access-control rules. Such utilities are needed to help users understand the powerful but potentially complex sets of rules that the Access Control Service allows.
- Development of utilities to import/export data via other standard formats, such as CCR documents, iCal calendar entries, NCPDP SCRIPT prescriptions, and HL7 lab results.

5. Additional Documentation

- Common Platform Components – Functional Requirements
(http://www.sujansky.com/docs/CommonPlatform_FunctionalRequirements.pdf)
- Common Platform Components – Technical Specifications Framework
(http://www.sujansky.com/docs/CommonPlatform_TechnicalSpecificationsFramework.pdf)
- Common Platform Components – Technical Specifications Overview
(http://www.sujansky.com/docs/CommonPlatform_TechnicalSpecificationsOverview.pdf)