

Project HealthDesign
Common Platform Components

Technical Specifications
Overview

Version 1.1
March 7, 2008

Sujansky & Associates, LLC

Contents

1.	Introduction	3
2.	Architecture and Functional Overview	3
2.1.	Authentication	4
2.2.	Registration of Users, Applications, and Patients	5
2.3.	Creation/Editing of Access-Control Rules	8
2.4.	Medication Management	10
2.5.	Observation Management	11
3.	Security Model	12
3.1.	Authentication	12
3.2.	Encryption	13
3.3.	Authorization	13
4.	Concurrency Control	13
5.	Terminology Recommendations	13
6.	Appendix A: WSDL Specifications for the Common Platform Components	16

Revision History

Version	Author/Editor	Date	Comment
0.9	Walter Sujansky / Sam Faus	1/15/2008	Pre-baselined version, pending final review
1.0	Walter Sujansky / Sam Faus	2/20/2008	Baselined version
1.1	Walter Sujansky	3/7/2008	Updates to reflect current implementation and other documentation

1. Introduction

This document provides an overview of the technical specifications for the common platform components (CPCs) of Project HealthDesign, particularly the programmatic interfaces to these services. The CPCs are intended to provide certain common, shared services to a wide variety of personal health applications (PHAs), with the goal of reducing implementation time and increasing interoperability. PHAs may use the defined interfaces to receive services from one or more common platform components, as needed. Initially, the CPCs will be implemented as web services that may be jointly accessed by multiple PHAs via internet protocols.

This documentation is intended to supplement the web services definition language (WSDL) specifications that formally define the programmatic interfaces to the CPCs. The WSDL specifications are provided in a separate set of files (see Appendix A for further information about the WSDL specifications). Further technical information about the specifications and the process of integrating PHAs with these web services appears in the document *CommonPlatform_WebServicesClientGuide.pdf* (available at the ProjectHealthDesign webex work space).

2. Architecture and Functional Overview

For the initial implementation of the Project HealthDesign common platform, five components (or *services*) have been defined and will be implemented, as listed in Table 1.

Table 1. Common Platform Components provided in initial implementation of Project HealthDesign

Component	Description
1. Registry Service	Stores demographic and password information of the users and applications that may access the common platform components, as well as of any patients whose data are managed by the components.
2. Authentication Service	Authenticates the identities of users and applications upon log in, and provides single sign-on across all of the CPCs.
3. Access Control Service	Stores the rules that specify which resources and operations users have access to. Also, 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.
5. Observations Service	Stores and makes available health-related observations that are captured by or on behalf of patients in the course of daily living.

Figure 1 illustrates how these components interact with each other and with a PHA to provide the defined services.

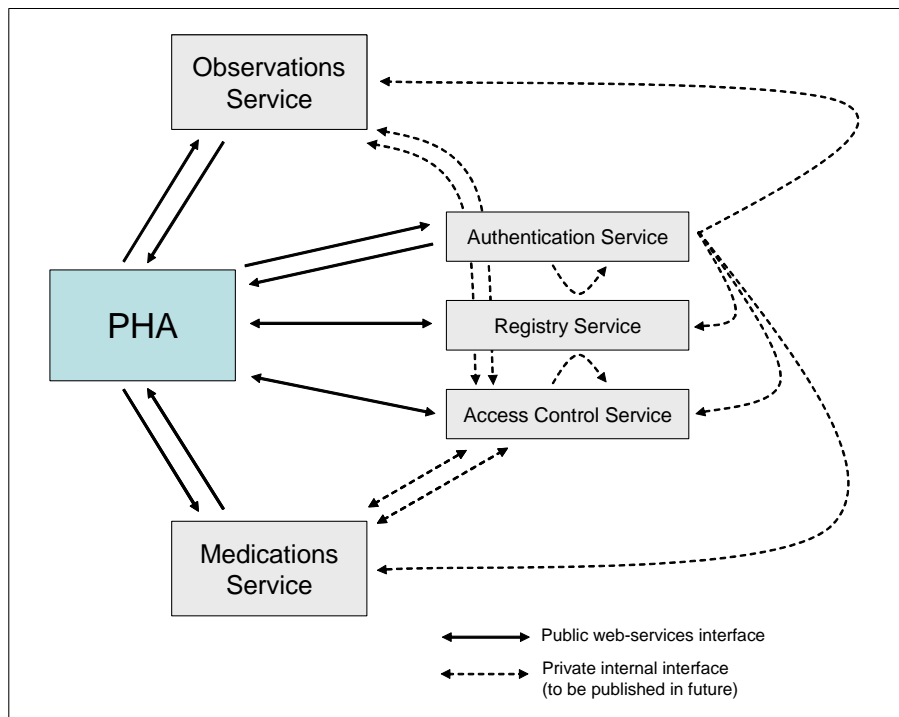


Figure 1. Component architecture of the common platform and points of interaction among the components

Note that only those interfaces depicted by solid arrows in Figure 1 will be publicly available in the pilot implementation and are described in the WSDL specifications. These are the interfaces that allow PHAs to directly request services from each of the five CPCs. To perform certain of these services, however, the CPCs must also interact with each other via additional internal interfaces (depicted by dashed arrows in Figure 1). These internal interfaces will not be publicly available during the pilot implementation of the CPCs because of time constraints and prioritization. However, these interfaces will be published at a later time to support alternative architectures in which PHAs can use Project HealthDesign CPCs for certain functions, but substitute their own or third-party components for other functions. For example, a PHA might wish to leverage the Project HealthDesign authentication, access-control, and observations services, while using its own registry and medications services. This architecture would require that the PHAs medications service be able to access the common Access Control Service via a public interface.

The following sections explain in detail how the components interact to provide various services (e.g., authentication, registration, medication management, etc.).

2.1. Authentication

Authentication must precede all other operations involving the CPCs (see Section 3 for a description of the security model for the common platform). The Authentication Service and the Registry Service combine to perform authentication. At the conclusion of authentication, the PHA and each CPC receive a “secure context token” that is used to authenticate each subsequent message exchange among the components. Figure 2 illustrates the component interactions involved in authentication, with each numbered step described in Table 2.

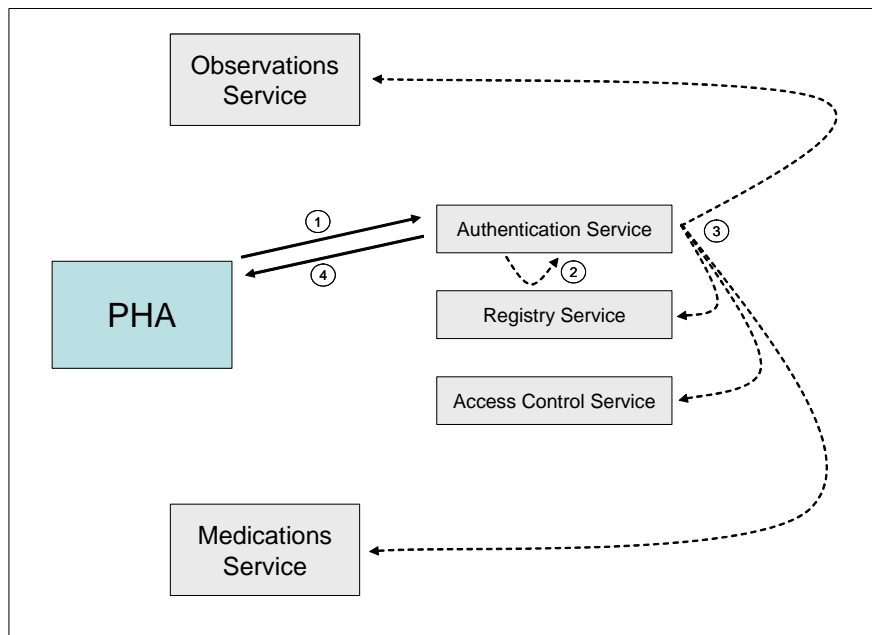


Figure 2. Component interactions involved in the authentication of a user and PHA upon login.

Table 2. Steps involved in authenticating users and PHAs

Step	Relevant <u>Service</u> and <u>Operation</u> in WSDL
1. Login request (with submission of user and application credentials)	<u>Authentication Service</u> : <i>requestSecurityToken (request)</i>
2. Password lookup	<internal interface>
3. Transmission of secure context token to all platform components in the web-services domain	<internal interface>
4. Transmission of secure context token to requesting PHA	<u>Authentication Service</u> : <i>requestSecurityToken (response)</i>

Note: Login and password information for each PHA must be pre-loaded into the registry database (see Section 2.2) via a super-user account, so that PHAs can initially authenticate themselves to allow the creation of additional user accounts.

2.2. Registration of Users, Applications, and Patients

The registry service maintains a database of all of the users and applications that may access the CPCs. This database is referenced by the authentication service when users attempt to log in. The registration data includes descriptive (“demographic”) information about the users and applications, as well as their login passwords. The registry service also maintains a database of all patients whose data are managed by one or more of the CPCs. This patient data is referenced by the access-control service when access-control rules are created or modified. The patient data maintained in the registry include demographic information only (i.e., name, DOB, address, etc.).

Figure 3 illustrates the component interactions involved in the registration of users, applications, and patients, with each step described in the table that follows.

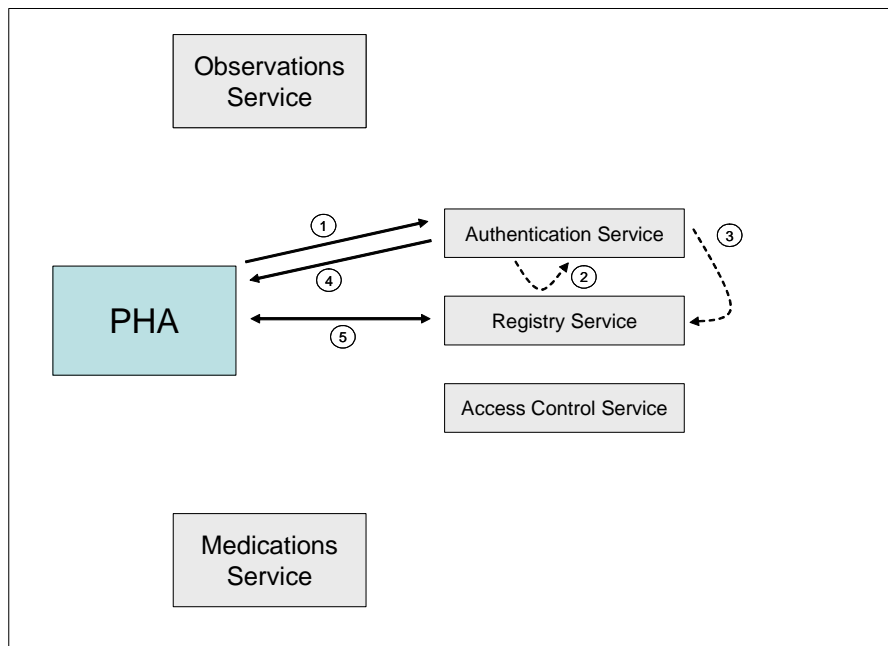


Figure 3. Component interactions involved in registration of users, applications, and patients.

Table 3. Steps involved in registering users, applications, and patients

Step	Relevant <u>Service</u> and <i>Operation</i> in WSDL
1 – 4. Authentication (as described in Sec.2.1)	<u>Authentication Service</u> : <i>requestSecurityToken (request / response)</i>
5. Registration of Users, Applications, and Patients -Create user account -Retrieve user account demographics -Edit user-account demographics -Edit user-account password -Delete user account -Create application account -Retrieve application account -Delete application account -Create patient record -Retrieve patient-record demographics -Edit patient-record demographics -Delete patient record	<u>Registry Service</u> : <i>registerUser (request / response / fault)</i> <i>getUserRecordsSimple (request / response / fault)</i> <i>updateUserDemographics (request / response / fault)</i> <i>updateUserPassword (request / response / fault)</i> <i>deleteUser (request / response / fault)</i> <i>registerApplication (request / response / fault)</i> <i>getApplicationRecordsSimple (req. / resp. / fault)</i> <i>deleteApplication (request / response / fault)</i> <i>registerPatient (request / response / fault)</i> <i>getPatientRecordsSimple (request / response / fault)</i> <i>updatePatientDemographics (req. / resp. / fault)</i> <i>deletePatient (request / response / fault)</i>

As described in Section 2.3, access to patient-specific data stored within the CPCs is defined within and controlled by the Access Control Service. However, user accounts and application accounts within the Registry Service are not associated with any specific patient record, so access to these data cannot be managed by the Access Control Service. Instead, access is controlled directly by the Registry Service using a set of pre-defined and hard-coded access-control rules. These rules are described in Table 4. Note that Table 4 also describes the access-control policies for patient records in the Registry Service, which are controlled by the configurable rules in the Access Control Service (as with any other patient-specific data).

Table 4. Access-control policies for management of user, application, and patient records.

<i>Operation</i>	<i>Authentication Requirement / Access-Control Restriction</i>
1. Create user account	Valid user & application login / any user may create another user account; each application itself has a “user” login, so the application may create accounts without a human user first logging in*
2. Retrieve user account	Valid user & application login / any user may view any other user’s account record (to allow granting of access privileges – see Sec.2.3)
3. Edit user-account demographics	Valid user & application login / only users may change their own demographic info
4. Edit user-account password	Valid user & application login / only users may change their own password
5. Delete user account	Valid user & application login / only users may delete their own account
6. Create application account	Valid user & application login / only SUPERUSER** may create an application account
7. Retrieve application account	Valid user & application login / only SUPERUSER** may view application accounts
8. Edit application account	Valid user & application login / only SUPERUSER** may edit application accounts
9. Delete application account	Valid user & application login; only SUPERUSER** may delete application accounts
10. Create patient record	Valid user & application login / any registered user may create a new patient record for herself or others
11. Retrieve patient-record demographics	Valid user & application login / access per access-control rules specified for the patient record
12. Edit patient-record demographics	Valid user & application login / access per access-control rules specified for the patient record
13. Delete patient record	Valid user & application login / access per access-control rules specified for the patient record

*Each application may be registered as both a *user* and as an *application*. This allows an application to automatically log in (i.e., without requiring a human user login) to perform certain operations (when allowed by the access-control rules). For example, applications may log in without a human user to create new user accounts or to automatically import data from an external system.

**SUPERUSER will have full access privileges to all data, including Registry and clinical data

2.3. Creation/Editing of Access-Control Rules

Authorization of users to access and update patient data is specified and controlled by two types of records in the access-control service, *Role relationships* and *access-control rules*.

- Role relationships represent the role(s) that a specific user has with respect to a specific patient record. Each relationship is represented as a triplet of user ID, Role ID, and patient-record ID. For example, a role relationship may indicate that “user 1234” has the role of “Parent” with respect to “patient record 9876”. This assertion means that user 1234 has all of the privileges that have been defined for the role “Parent” in the context of patient record 9876. These privileges, in turn, are represented as access-control rules.
- Access-control rules represent the allowed combinations of patient records, read/write operations, data resources, user roles and application contexts (see Section 7.4 of the document *Project HealthDesign: Functional Requirements, Version 1.0* for a detailed description of the format and meanings access-control rules). For example, an access-control rule might specify that for “patient record 1234”, a “VIEW RECORD” (read) operation is allowed on any “MEDICATION” data for a user with the role “Parent” in the context of any personal health application. Many such rules may exist for a single patient record to provide fine-grained access control, and these rules are created and edited via interactions between the PHA and the access-control service.

Figure 4 illustrates the component interactions involved in the creation and editing of access-control information, with each step described in the table that follows.

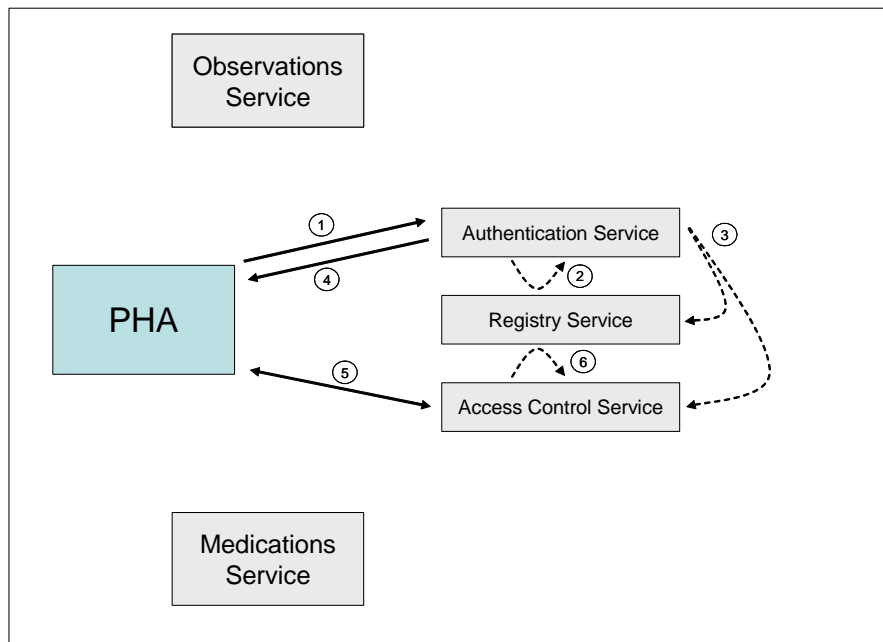


Figure 4. Steps involved in creation/editing of access control rules

Table 5. Steps involved in creating and editing access-control rules

Step	Relevant <u>Service</u> and <u>Operation</u> in WSDL
1 – 4. Authentication (as described in Sec.2.1)	<u>Authentication Service</u> : <i>requestSecurityToken (request / response / fault)</i>
5. Creation/Editing of Access Control rules -Create role relationship -List role relationships for a patient record -Update role relationship -Delete role relationship -Create access-control rule -List access-control rules for a patient record -Update access-control rule -Delete access-control rule	<u>Access Control Service</u> : <i>createRelationship (request / response / fault)</i> <i>getPatientRelationships (request / response / fault)</i> <i>updateRelationship (request / response / fault)</i> <i>deleteRelationship (request / response / fault)</i> <i>createAccessRule (request / response / fault)</i> <i>getPatientAccessRules (request / response / fault)</i> <i>updateAccessRule (request / response / fault)</i> <i>deleteAccessRule (request / response / fault)</i>
6. Lookup of user ID or patient ID when creating a new role relationship or access-control rule	<internal interface>

NOTE: When a user creates a new patient record (via the Registration Service), the following role relationship and access control rules must be automatically created by the Access Control Service for that patient record:

Role Relationship:

User ID	Role	Patient Rec ID
<UserID>	“RecordCustodian”	<PatientRecID>

Access Control Rules:

Patient Rec ID	Operation	Resource	Role	Context	Action
<PatientRecID>	“RecordModification”	“AllHealthData”	“RecordCustodian”	“AllApplications”	“Grant”
<PatientRecID>	“RecordViewing”	“AllHealthData”	“RecordCustodian”	“AllApplications”	“Grant”

These relationships and rules have the effect of granting to the creator of the patient record the ability to create, edit, and view any data for that record, including additional role relationships and access-control rules. This initial set of rules must be automatically created, because the user does not have the ability to create or view anything for the record (including the record’s access control rules themselves) until these minimum rules exist.

2.4. Medication Management

Once the required users, applications, and patients are registered and the desired access-control rules are defined, a PHA may begin using the Medications Service and the Observations Service to manage patient-specific medication and observation data. Figure 5 illustrates the component interactions involved in the use of the Medications Service, with each step described in the table that follows. Analogous interactions are involved in use of the Observations Service.

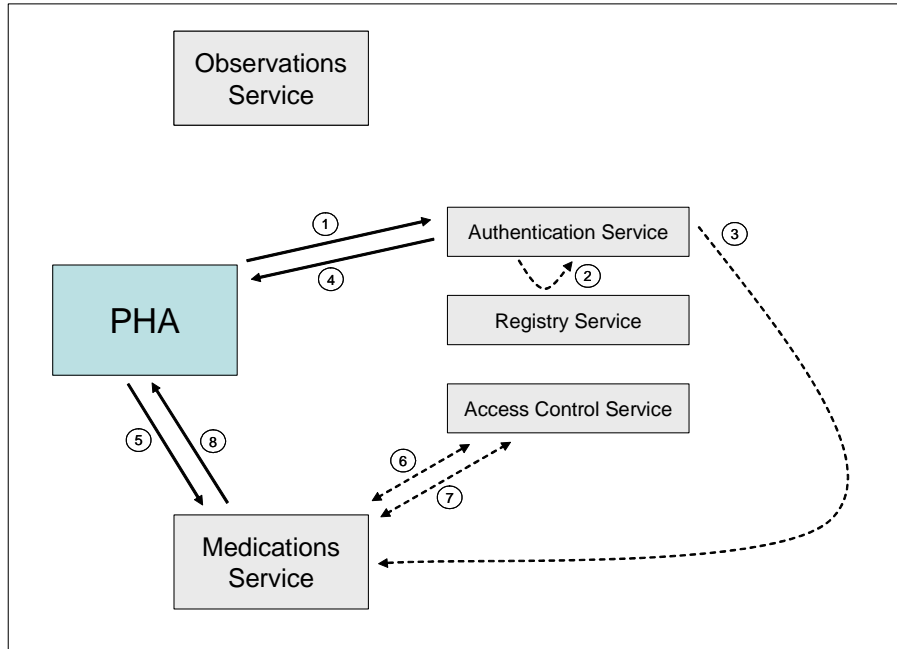


Figure 5. Interactions involved in the management of medication data.

Table 6. Steps involved in creating and editing medication data

Step	Relevant <u>Service</u> and <u>Operation</u> in WSDL
1 – 4. Authentication (as described in Sec.2.1)	<u>Authentication Service</u> : <i>requestSecurityToken (request / response)</i>
5. Request for medication management -Create a medication record -Update a medication record -*Retrieve medication records by various search criteria -*Retrieve medication records by unique ID -Delete a medication record -Create an annotation to a medication record -Update an annotation to a medication record -Retrieve all annotations for a medication record -Retrieve annotations by unique ID -Delete an annotation -Create an attachment to a medication record -Retrieve attachment by unique ID -Delete attachment	<u>Medications Service</u> : <i>createMedRecord (request)</i> <i>updateMedRecord (request)</i> <i>getMedRecordsSimple (request)</i> <i>getMedRecordsByID (request)</i> <i>deleteMedRecord (request)</i> <i>createAnnotation (request)</i> <i>updateAnnotation (request)</i> <i>getMedicationAnnotations (request)</i> <i>getAnnotationByID (request)</i> <i>deleteAnnotation (request)</i> <i>createAttachment (request)</i> <i>getAttachment (request)</i> <i>deleteAttachment (request)</i> <i>(cont'd)</i>

<p>6. Authorization of request</p> <p>A. Compare nature of request to the access control rules for the relevant patient record</p> <p>B. If request is authorized, continue with operation</p> <p>C. If request is not authorized, report access-control error to PHA</p>	<p><internal interface></p>
<p>7. Filter retrieved records based on access control</p> <p>A. Compare the type of each retrieved record to the access control rules for the relevant patient</p> <p>B. Indicate to medication service which records are authorized for release and which should be suppressed.</p> <p>[Note: This operation only applies to those services in step 5 that are preceded by “*”]</p>	<p><internal interface></p>
<p>8. Return results of the service request to the PHA</p>	<p><u>Medications Service</u> :</p> <p><response or fault corresponding to the request operations in step 5; see WSDL></p>

Note:

- Medication and observation records contain only *references* to annotations and attachments. PHAs must retrieve the actual annotation and attachment data through a separate call to the medication or observation service (this approach supports distinct access-control rules for medication data and for annotations, as well as avoids potentially long retrieval times when a medication record contains large attachments).
- PHAs may not retrieve all of the attachments associated with a medication or observation record in a single operation, because the size of these attachments could make such an operation quite slow. Rather, the PHA must request each attachment individually. In contrast, PHAs may retrieve all of the annotations for a medication or observation record in a single operation, because text annotations are likely to be much smaller than multi-media attachments.
- When an attachment or an annotation is created for a medication or observation, the medication or observation record is automatically updated (along with the record version information) to include a reference to the new attachment.
- When an attachment or an annotation in a medication or observation is deleted, the medication or observation record is automatically updated (along with the record version information) to remove the reference to the deleted attachment.
- When an annotation in a medication or observation is updated, the medication or observation record is automatically updated (along with the record version information) to indicate that a component of the record has been changed.

2.5. Observation Management

See Section 2.4 (all of the steps involved in Medications Management have corresponding steps for Observations Management).

3. Security Model

There are three aspects of security that the common platform must address: Authentication, Encryption, and Authorization. This section describes the way that the prototype implementation of the platform will address these areas.

3.1. Authentication

For the common platform, authentication is based on the concept of a web-services “secure conversation”*. This concept allows a PHA to establish a “secure context” among several web-service components, which then allows the PHA to exchange web-service messages with any of the components within that context for some period of time without separately authenticating itself to each component. The advantages of this approach are that it allows the PHA to access multiple CPCs with a single sign-on, while ensuring that only authenticated users and PHAs access those resources.

The common platform establishes a secure context through the following steps:

1. A user that wishes to access one or more CPCs from a specific PHA will request a “Secure Context Token” (SCT) from the Authentication Server. This request requires that the user and the PHA authenticate themselves by providing appropriate passwords.
2. Upon successful authentication, the Authentication Server creates the SCT, which contains a unique identifier for the token, the identities of the user and the PHA that were authenticated, the date/time that the token was created, and an encryption key (“proof key”) that the PHA will later use to substantiate possession of the SCT.
3. The Authentication Service securely sends copies of the SCT to all of the CPCs in the common platform domain (see Figure 2 in Section 2.1), as well as to the PHA that originated the request. A secure context has now been established among these components that will remain valid for a period of time defined by the participating components or (optionally) by the Authentication Service.
4. When the PHA requests any services from any of the CPCs, it includes a *reference* to the Secure Context Token within its request. This reference includes a data element that has been encrypted using the proof token contained within the SCT. If the CPC is able to successfully decrypt the datum using its copy of the proof token, then this substantiates that the requestor of the service is, indeed, in possession of the SCT and must be the user and the application that were previously authenticated. The web service may then use the identity of the user and the PHA as specified within the SCT to determine what access privileges that user and PHA have (i.e., in conjunction with the Access Control Service).
5. When the SCT expires, a PHA may no longer use it to request services from the common platform, and a new SCT must be requested (i.e., the user and the PHA must re-authenticate). SCTs are typically valid for a period of several hours to a day. The reason that they expire is to limit the time window available to a malevolent process to guess the encryption key via a brute force attack.

* This concept has been standardized by the Organization for the Advancement of Structured Information Standards (OASIS) in the form of the WS-SecureConversation standard. This standard is the model for the design of the authentication system for the platform components, although the actual implementation will not conform precisely to the standard and will not provide all of its functionality. This simplification of WS-SecureConversation has been chosen both to accelerate implementation of the common platform and to simplify the PHAs’ interaction with the common platform. The actual WS-SecureConversation standard may be specified and implemented for the common platform at a later time, and we believe it will be compatible with the security infrastructure specified at this time.

3.2. Encryption

As described, the establishment of a secure context requires the secure (encrypted) exchange of authentication information and Secure Context Tokens. Additionally, the exchange of patient health information among the participants in a secure conversation itself requires encryption of web-service messages (i.e., the messages that contain medications, observations, and patient-demographic information, which all pass over the public internet).

The web-services security standard WS-Security provides encryption of individual SOAP messages, by embedding in the header information of SOAP messages the relevant encryption/decryption information and by encrypting/decrypting messages within the SOAP-processing layer of software. For the pilot implementation of the common platform components, however, encryption will not be provided at the message level via WS-Security, but rather at the transport level via Secure Socket Layer (SSL) connections. Although the encryption of individual messages may have some advantages in a web-services architecture, SSL has been selected because it is already available in most commercial web servers, readily activated, and familiar to most software developers today. This approach will therefore streamline implementation of the Project HealthDesign security infrastructure during the pilot phase. In addition, to minimize implementation complexity and risks, the initial implementation and testing of the common platform may not even include SSL encryption (with the plan to add this capability as integration testing proceeds).

3.3. Authorization

Authorization for accessing any patient-specific data will be handled by the Access Control Service, as described in Section 2.3. Authorization for accessing user account and application account records will be handled by built-in access rules in the Registry Service, as described in Section 2.2.

4. Concurrency Control

One of the objectives for the common platform components is to allow multiple PHAs to share the same patient data related to medications and observations, patient demographics, and access-control rules. When there exists the potential for multiple applications to access the same data concurrently, however, unintended consequences may occur when updates to the data are made. For example, one application may unintentionally overwrite the changes made by another. To prevent these error, a system may either prohibit simultaneous access to data (through locking) or may allow simultaneous access, but detect and prevent conflicting changes. The common platform has specified the latter approach through the careful tracking of versioning information, as described in Appendix B.

5. Terminology Recommendations

Terminology standards are an important component of semantic interoperability among personal health applications. Such standards dictate which data elements must be coded and which coding systems must be used.

In the course of the requirements-gathering process for the CPCs, however, the participants in ProjectHealthDesign determined that it was premature to establish and enforce such terminology standards within the common platform. Rather, it was preferred that the CPCs *support* the use of coding for relevant data elements and *recommend* specific coding systems in these cases. Conformance with these recommendations will be voluntary, and the CPCs will not enforce the coding of any medical data nor the use of any specific coding systems.

The specific data types that support coding are defined in the WSDL specifications (specifically, see the files `Medications.xsd`, `Observations.xsd`, and `CommonTypes.xsd`). Table

7 and Table 8 list the terminology recommendations for various data elements in the Medications and Observations Services, respectively.

Table 7. Terminology recommendations for Medications Service.

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 8. Terminology Recommendations for Observations Service.

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		

6. Appendix A: WSDL Specifications for the Common Platform Components

The common platform components will be implemented as web services that can be accessed over the internet using standard internet protocols (specifically, SOAP). The interface specifications for the common platform components are defined using the Web Services Definition Language (WSDL). A previous document described WSDL specifications and how they may be used to develop client PHAs that access web services via the defined interfaces (see *TechnicalSpecifications_ProposedFramework_2007-12-10.doc*).

The WSDL specifications for the CPCs consist of a set of WSDL and XML Schema files that are organized within a directory structure as shown in Figure 6.

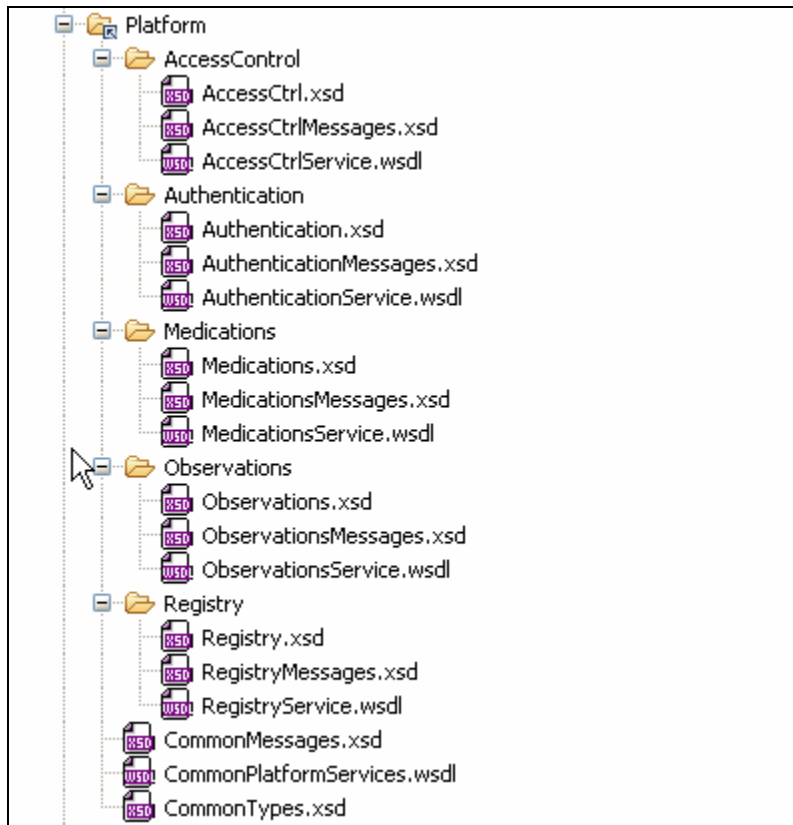


Figure 6. Files that define the WSDL specifications for the CPCs.

Note that the specification for each component <component> is defined by three files:

- <component>Service.wsdl** Lists the operations that the component provides
- <component>Messages.xsd** Describes the XML structure of the request and response messages used in the operations
- <component>.xsd** Describes the XML data types that appear in the messages

Additionally, there are a set of operations, messages, and data types that are used by multiple of the components, and these are defined within the files:

Common Messages.xsd

CommonTypes.xsd

To assist in the navigation and review of the WSDL specifications, a set of HTML documents has been created that list and describe the services, messages, and data types for each of the web services. These documents are available within a directory structure as shown in Figure 7:



Figure 7. Directories containing HTML documentation of WSDL interface specifications

Each of the directories shown in Figure 7 contains an “index.html” file. This file contains the entry point for a hypertext documentation of the WSDL specification for each web service, which can be viewed and navigated in a web browser (IE works best for these documents). For example, the entry-point file for the MedicationsService is shown in Figure 8.

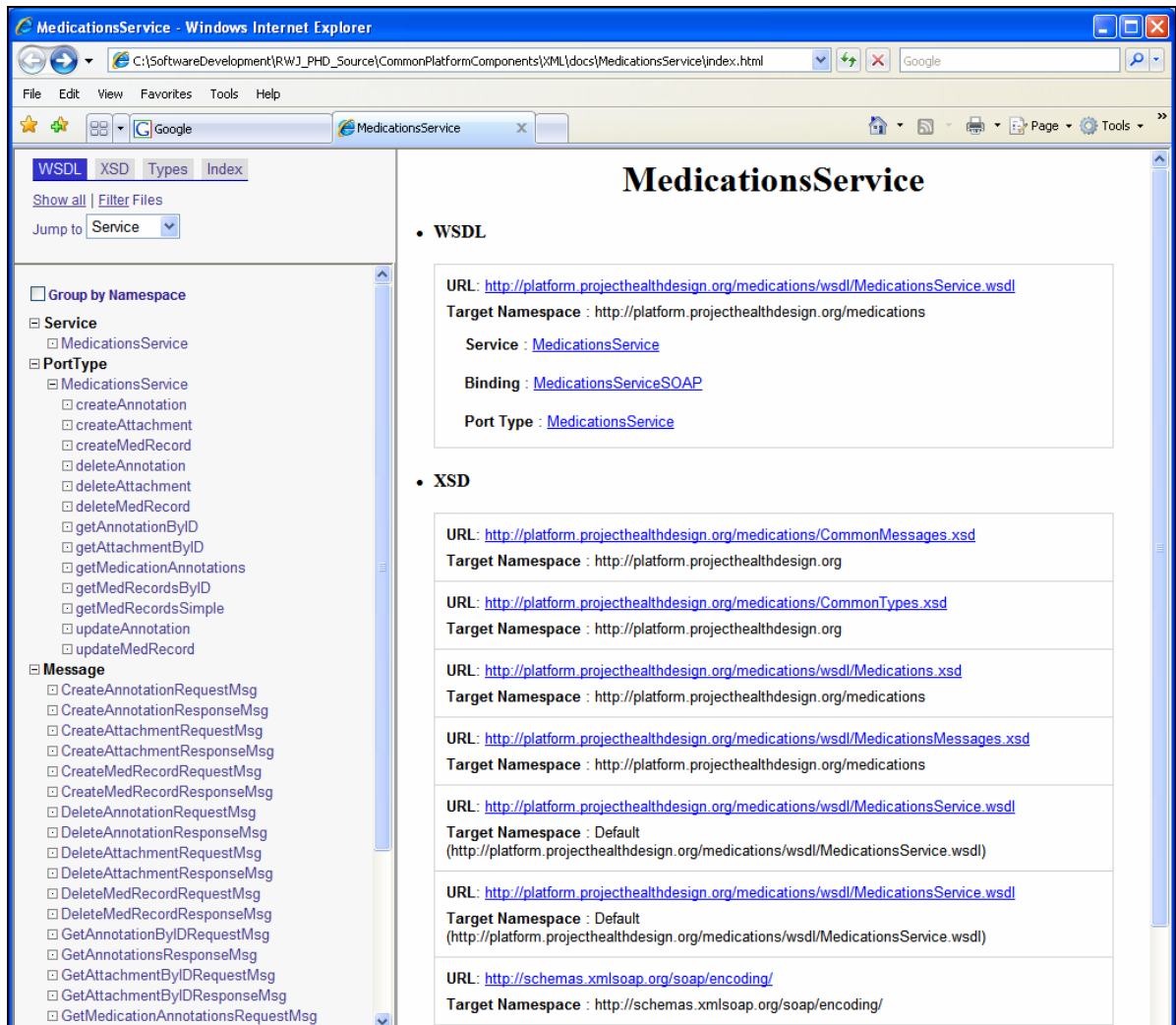


Figure 8. HTML documentation of the WSDL interface specifications for the Medications component.

