

Common Platform Components: Web Services Client Guide

**Version 0.5
March 7, 2008**

**Prepared by
Sujansky & Associates, LLC**



This work is licensed under a [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/).
© 2009 Board of Regents of the University of Wisconsin System

Revision History

Version	Author/Editor	Date	Comment
0.1	Sam Faus	1/31/08	Created document.
0.2	Walter Sujansky and Sam Faus	2/4/08	Added clarification to query logic description for observations. Also corrected RecordState value list.
0.3	Sam Faus	2/5/08	Added instructions for how to populate the Authentication Element in most request messages.
0.4	Sam Faus	2/15/08	Updated document based on changes made to WSDLs and Schema documents (now reflects v1.1 of the WSDL). Added RecordStateData and RecordState descriptions. Added Appendix B: Access Control Ontological Hierarchies
0.5	Sam Faus	3/7/08	Added a description of the Record versioning model implemented by the common platform services

Contents

General Web Services Information	1
Unique IDs	1
Providing Authentication Information in Requests (In Progress)	2
RecordStateData	4
RecordState	5
Record Versioning Model	6
Annotation Operations	8
createAnnotation	8
updateAnnotation	9
deleteAnnotation	10
getAnnotationByID	10
Attachment Operations	11
createAttachment	11
deleteAttachment	12
getAttachmentByID	13
Fault Error Codes (In Progress)	13
Registry Service	14
User Registration Operations	14
registerUser	14
getUserRecordsSimple	15
updateUserPassword	15
updateUserDemographics	15
deleteUser	16
Patient Registration Operations	16
registerPatient	16
getPatientRecordsSimple	16
updatePatientDemographics	17
deletePatient	17
Application Registration Operations	17
registerApplication	17
getApplicationRecordsSimple	18
deleteApplication	18
Authentication Service	18
Security Token Operations	18
requestSecurityToken	18
Medications Service	20
Medication List Operations	20
createMedRecord	20
getMedRecordsSimple	24
getMedRecordsByID	25
updateMedRecord	26
deleteMedRecord	26

getMedicationAnnotations.....	26
Observations Service.....	27
Observation Operations	27
createObsRecord	27
getObsRecordSimple.....	33
getObsRecordsByID	35
updateObsRecord	36
deleteObsRecord	36
getObservationAnnotations	36
Access Control Service.....	37
Access Control Rule Operations	37
createAccessRule	37
updateAccessRule	40
deleteAccessRule.....	40
getPatientAccessRules	40
Patient Record Relationship Operations	41
createRelationship	41
updateRelationship	42
deleteRelationship.....	42
getPatientRelationships.....	42
Appendix A: Summary of Coded Values (Forthcoming).....	43
Appendix B: Access Control Hierarchies.....	44
Operation Hierarchy	44
Context Hierarchy	44
Resource Hierarchy.....	45
User Hierarchy	46

Index of Tables

Table 1: Common Platform Component Assigned OIDs	1
Table 2: Authentication Element	2
Table 3: RecordStateData Element	4
Table 4: RecordState Element	6
Table 5: CreateAnnotationRequest Element.....	9
Table 6: GetAnnotationByIDRequest Element.....	10
Table 7: CreateAnnotationRequest Element.....	11
Table 8: GetAttachmentRequest Element.....	13
Table 9: GetUserRecordsSimpleRequest Element	15
Table 10: GetPatientRecordsSimpleRequest Element	16
Table 11: RequestSecurityToken Element.....	19
Table 12: SecureContextToken Element	20
Table 13: Medication Element	21
Table 14: GetMedRecordsSimpleRequest Element.....	24
Table 15: GetMedRecordsByIDRequest Element.....	25
Table 16: GetAnnotationsRequest Element (Medications)	26
Table 17: GeneralObservation Element.....	27
Table 18: MedicationAdministration Element.....	29
Table 19: PhysicalActivity Element	30
Table 20: MealOrSnack Element	31
Table 21: SignOrSymptom Element.....	31
Table 22: Pain Element.....	31
Table 23: ObservableParameter Element.....	32
Table 24: GetObsRecordSimpleRequest Element.....	33
Table 25: DateTimeQry Element:.....	34
Table 26: GetRecordsByIDRequest Element.....	35
Table 27: GetAnnotationsRequest Element (Observations)	36
Table 28: Rule Element	38
Table 29: ListPatientAccessRulesRequest Element	41
Table 30: CreateRelationshipRequest Element	41
Table 31: ListPatientRelationshipsRequest Element	42

General Web Services Information

Information in this section applies across all of the common platform web services.

Unique IDs

Every record created by a common platform component is assigned a globally unique identifier upon creation. This process applies to medication records, observation records, text annotations, and multi-media attachments, as well as patients, users, and applications. All unique ids used by common platform components (e.g., RecordUniqueID, PatientUniqueID, ParentRecordUniqueID, etc.) are represented within the WSDL interface as XML elements of type `phd:UniqueIDType`, a simple string data type. However, the strings generated by the common platform components are composed of the following parts:

`<local identifier>@<component assigned OID>`

The local identifier component will be a simple incrementing number. The local identifier is followed by the “@” (ASCII character #64) symbol and then a unique object identifier (OID). Each Common Platform component is assigned a globally unique OID specific to the Project HealthDesign Common Platform. This OID was obtained from an ISO Name Registration Authority, to guarantee global uniqueness. A RecordUniqueID for a Medication List item may look like the following:

123@1.3.6.1.4.1.30291.0.4

Globally unique identifiers allow shared platform components to reference data objects that are created and managed by multiple independent data repositories without risk of duplicating identifiers. For example, the access-control component may reference medication records managed by one repository and observations managed by another, or the access-control component may reference user records managed by a separate registry service. Also, the globally unique nature of the identifier will allow a PHA processing data that originated from a particular Common Platform component to determine the source of the record identifier and the data.

Table 1 below lists the OIDs assigned to each of the Common Platform components.

Table 1: Common Platform Component Assigned OIDs

Component	Component Assigned OID
Registry Component	1.3.6.1.4.1.30291.0.1
Access Control Component	1.3.6.1.4.1.30291.0.2
Observations Component	1.3.6.1.4.1.30291.0.3
Medication List Component	1.3.6.1.4.1.30291.0.4
Calendar Component	1.3.6.1.4.1.30291.0.5
Authentication Component	1.3.6.1.4.1.30291.0.6

Providing Authentication Information in Requests (In Progress)

This section describes the process for populating the Authentication element within requests made to any of the Project HealthDesign Common Platform Component (CPC) Service. Note that this element must be populated in all requests to ensure that only valid authenticated users and applications can read or write data in the components.

Authentication Population Instructions:

1. Upon successful authentication via the Authentication Service *getSecurityToken* operation, a *SecureContextToken* is returned to the PHA, which contains a *SecurityTokenUniqueID* and a *ProofTokenEncryptionKey*.
2. The client PHA gets the *ProofTokenEncryptionKey* from the *SecureContextToken*. The *ProofTokenEncryptionKey* is a String consisting of 40 characters.
e.g., "fbada4676477322fb3e2ae6353e8bd32b6d0b49c"
3. The client PHA chooses a *Nonce* value that is likely to be unique and never re-used (such as the date/time of the request). This can be any text value but should be something easy to produce, like the user's *UserUniqueID* or the current date.
e.g., "20080214152345"
4. The client PHA concatenates the *Nonce* value with the *ProofTokenEncryptionKey* value as follows: *Nonce* + *ProofTokenEncryptionKey*.
e.g., "20080214152345fbada4676477322fb3e2ae6353e8bd32b6d0b49c"
5. Using the commonly available SHA-1 cryptographic digest algorithm, the client PHA generates a hexadecimal encoded hash digest of the concatenated *Nonce* and *ProofTokenEncryptionKey*. This value is then used as the *ProofToken* in subsequent requests to CPCs.
e.g., "87655ac2cbde95c853dd86faae433235dbe958a7"
6. When making requests to Common Platform components, the client PHA populates the Authentication element of the request with the following information:

Table 2: Authentication Element

Element	Optional	Unbounded	Types {Values}	Comments
SecurityTokenUniqueID	No	No	phd:UniqueIDType	The ID of the security token stored by the Authentication Service CPC and distributed to all connected CPC services. Use the exact value of the <i>SecurityTokenUniqueID</i> that was returned from the <i>getSecurityToken</i> operation.
Nonce	No	No	xs:string	Populate this element only with

				the client PHA generated Nonce value (per step 3 above). Do NOT include any part of the ProofTokenEncryptionKey in this element.
ProofToken	No	No	xs:string	The hexadecimal encoded hash digest value generated from the Nonce + ProofTokenEncryptionKey (per step 5 above)

Note: To facilitate testing, a special nonce value has been defined that allows client applications to circumvent the authentication process described above. When this value is placed in the <Nonce> field of a request message, the web services will authenticate the request, regardless of the value in the <ProofToken> field. However, the <SecurityTokenUniqueID> must still contain the ID of a valid security token that has not yet expired.

Authentication Example:

The Client PHA receives the following information in a RequestSecurityTokenResponse message:

```

/S:Envelope/S:Body/RequestSecurityTokenResponse/RequestedSecurityToken/
<SecureContextToken>
  <ns2:SecurityTokenUniqueID>5259099512514897665</ns2:SecurityTokenUnique
  ID>
  <ns2:UserUniqueID>super</ns2:UserUniqueID>
  <ns2:AppUniqueID>access</ns2:AppUniqueID>
  <ProofTokenEncryptionKey>2fd4e1c67a2d28fced849ee1bb76e7391b93eb12</Proo
  fTokenEncryptionKey>
  <CreatedDateTime>2008-02-06T19:39:02.765Z</CreatedDateTime>
  <ExpiresDateTime>2008-02-06T19:39:02.765Z</ExpiresDateTime>
</SecureContextToken>

```

The Client PHA generates a ProofToken using the SHA-1 hash function (pseudo-code):

```

Nonce = 123456789@10.123.4567.89
ProofTokenEncryptionKey = 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
SHA-1(Nonce + ProofTokenEncryptionKey) =
87655ac2cbde95c853dd86faae433235dbe958a7

```

The client PHA uses the stored and generated information to populate the Authentication element of a CreateMedRecordRequest message:

```

/SOAP-ENV:Envelope/SOAP-ENV:Body/m:CreateMedRecordRequest/
<m0:Authentication>
  <m0:SecurityTokenUniqueID>5259099512514897665</m0:SecurityTokenUniqueID
  >
  <m0:Nonce>20080214152345</m0:Nonce>
  <m0:ProofToken>87655ac2cbde95c853dd86faae433235dbe958a7</m0:ProofToken>
</m0:Authentication>

```

RecordStateData

The RecordStateData element contains meta-data about the *version* of a record at a given time. Note that the common platform components retain previous versions of data when records have been updated (i.e., all updates are “non-destructive”). The meta-data in the RecordStateData element is used specifically to indicate which version of a record is the current (most recent) version, as well as to document which user created and/or updated any previous versions.

Information in the RecordStateData element includes the record version number, the date range during which the version was current, the operation performed to arrive at that version, the user who performed the operation, and the application that was used to perform the operation. All Common Platform components will include RecordStateData information for each patient record item managed by the component. Furthermore, all Common Platform components shall require a PHA to provide current RecordStateData information for a record when a user attempts to update or delete that record. This policy ensures that the record instance a PHA is attempting to modify is the most recent version of the record. If it is not, the PHA is required to first retrieve the current version before updating or deleting the record. The purpose of this policy is to prevent PHAs from inadvertently overwriting each others changes, given that multiple PHAs may have concurrent access to the same data records.

Table 3: RecordStateData Element

Element	Optional	Unbounded	Type {Values}	Comments
StartStateDate	No	No	xs:dateTime	The date/time at which the record entered the described state. If there are previous states for the record, then the StartStateDate of the record shall be equal to the EndStateDate of the previous record.
EndStateDate	Yes	No	xs:dateTime	The date/time at which the record left the described state. This element will be empty for the current version of the record only.
RecordState	No	No	xs:string {Current, Historical, Deletion}	Indicates whether the record instance represents the current record (Current), an historical view of the record (Historical), or the documentation of the record’s deletion (Deletion).
StateAction	No	No	xs:string {Created, Updated, Annotated, Deleted}	Describes the type of user action that brought about the described state.
UserUniqueID	No	No	phd:UniqueIDType	The ID of the user that brought about the described state. Note that this is not the LoginName of the user.
AppUniqueID	No	No	phd:UniqueIDType	The ID of the application used to bring about the described state.
Version	No	No	xs:integer	An integer that is automatically incremented for each new version of the record, beginning with version 1

and proceeding to 2, 3, etc. (Version > 0).

Submitting RecordStateData in Requests for Record Updates and Deletions

PHAs are required to provide the RecordStateData element for the current version of a record when attempting to execute an Update or Delete operation. This requirement ensures that the contents of the record being updated or deleted by the PHA are the same as that of the most recent version of the record stored in the Common Platform component. The component compares the RecordStateData element, provided by the PHA in the update/delete command to the RecordStateData for the current version of the record in the data repository. If the RecordStateData elements match, then the update or delete may proceed (assuming the user has the adequate access). If the RecordStateData elements do NOT match, then the update/delete will fail, the Common Platform component will return a Fault message, and the PHA will be required to retrieve the latest version of the record prior to re-executing the update/delete. This approach helps to avoid the inadvertent overwriting of changes to a record made concurrently by another user or another PHA.

Note: Special RecordStateData rules exist for updating and deleting [Annotation](#) and [Attachment](#) records. Please refer to the sections detailing annotation and attachment operation for details.

RecordState

The Common Platform components provide a mechanism that allows PHAs to selectively retrieve only the current version of a record, only the historical version(s), only the documentation of a record deletion (i.e., “deletion records”), or any combination of these types. When executing a “get” operation (e.g., getMedRecordsByID or getObsRecordsSimple), the element RecordState is used to allow the client PHA to filter the results by the record’s current, historical, and/or deletion states. This element is always available on all Common Platform component patient record item request elements.

Table 4: RecordState Element

Element	Optional	Unbounded	Types {Values}	Comments
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value

The valid values for RecordState are: “Current”, “Historical”, and “Deletion”. The default value is “Current”. Note that the RecordState element in a “get” operation may be repeating. If users provide multiple RecordState elements, the CPC shall return all records that match any of the values provided (i.e., equivalent to a logical “OR” query).

When the value of RecordState in a “get” request is set to “Current” (or defaults to current when no value is provided), then only the most recent version of the file will be returned. Note that there shall be at most one version of an instance record with a value of “Current” (although there may be none if the record has been deleted).

If the value of RecordState is set to “Historical,” the CPC will return only the historical records that match the search terms. There may be multiple historical records for a given RecordUniqueID, depending on how many times the record may have been updated. When returning multiple historical versions, the CPC shall return the records in reverse chronological order beginning with the most recent record. The component may return zero historical records for a given RecordUniqueID if only the current record exists.

Finally, if the value of RecordState is set to “Deletion”, then the component shall return that version of the record that documents its deletion (if one exists). There shall only ever be one Deletion version of an instance record. The component may return zero deletion records for a given RecordUniqueID if the identified record has not been deleted.

Note: A user’s ability to query for current, historical, or deletion records is subject to the user’s defined read privileges for a given patient record.

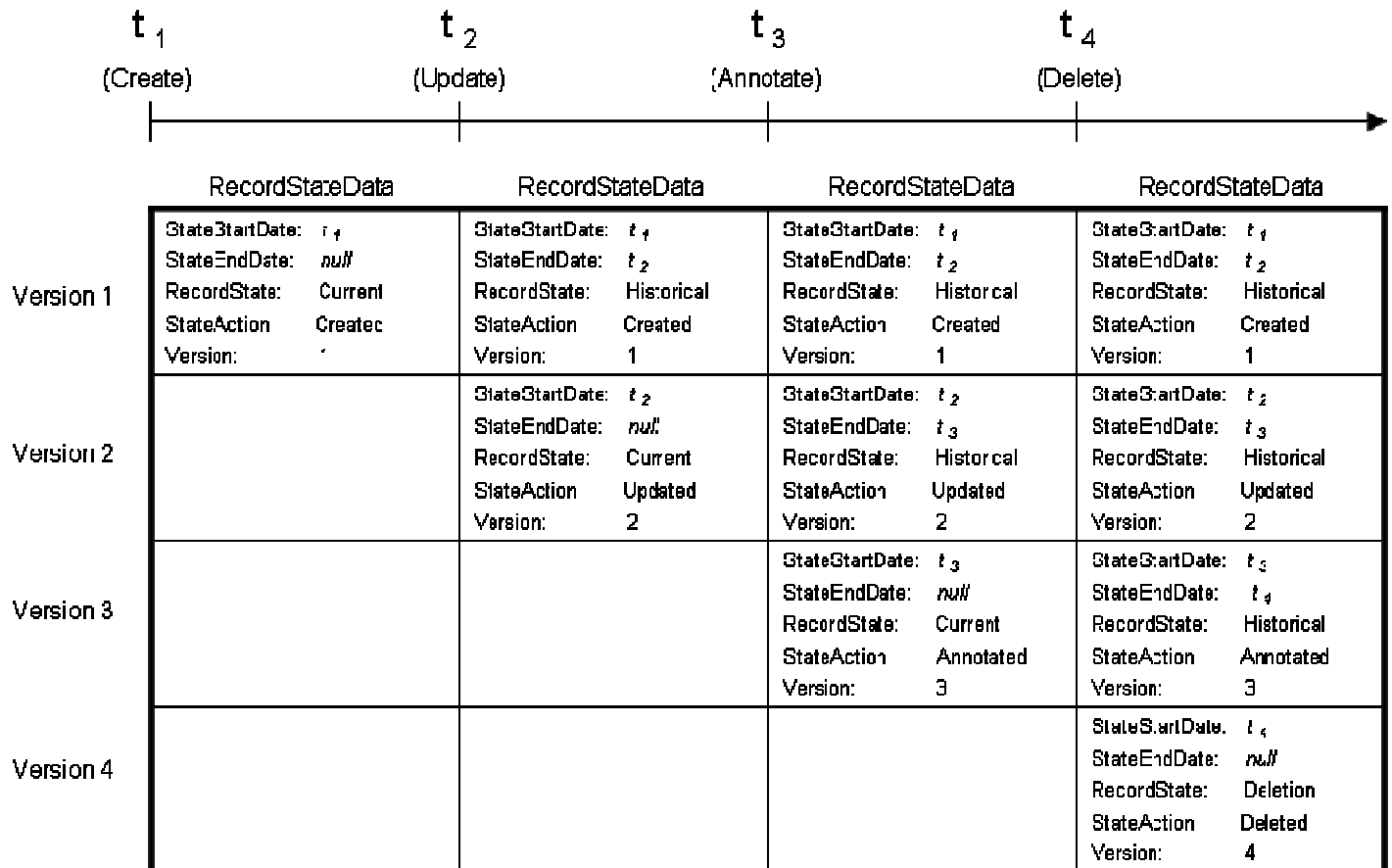
Record Versioning Model

As a record progresses through its lifecycle of creation, update, annotation, and possible deletion, the Common Platform Component services keep track of these transitions as part of its record versioning model. This section describes the changes that the Common Platform Component makes to the RecordStateData elements of a record as it is created, updated, and deleted by client PHA systems.

Every time a record is created or changed as a result of PHA action, the Common Platform Component creates a new version (i.e., copy) of the record. The RecordStateData of this new record version indicates who changed the record, how it was changed, and via which application the change was enacted. In addition, the component updates certain of the elements of RecordStateData in the previous version

of the record (if a previous version existed). Figure 1 below shows how RecordStateData of a record is populated and modified as the record is created, updated, and deleted by a PHA.

Figure 1: Versioning of Records as they are Created, Updated, and Deleted



Please note the following important aspects of the Common Platform Record Versioning Model:

1. Once a record has been deleted there will no longer be any versions of the record with the RecordState value of “Current”.
2. The StateStartDate of a given record version never changes
3. The StateEndDate of a given record version will always be *null* while the RecordState is “Current”
4. The StateAction of a given record version never changes, since this information describes the action that created the record version.
5. If a record has an Annotation added to it, or if an Annotation for a record is updated or deleted, a new version of the parent record will be created and the StateAction of that version will be “Annotated”. Note that the Annotation record will also have associated RecordStateData that is distinct from that of the parent

record to which it belongs (the RecordStateData of Annotation records is not shown in Figure 1) .

6. If a record has an Attachment added to it, or if an Attachment for a record is deleted, a new version of the parent record will be created and the StateAction of the parent record will be “Updated”. Note that the Attachment record will also have associated RecordStateData that is distinct from that of the parent record to which it belongs (the RecordStateData of Attachment records is not shown in Figure 1).
7. RecordState shall never change from “Updated”/”Annotated” to “Created”
8. RecordState shall never change from “Deletion” to any other value.

Annotation Operations

Annotations may be added to observation and medication records. Annotations are distinct “child” records that have their own record state, and may be associated with a single observation or medication “parent” record. Each respective service provides operations that allow PHAs to create, read, update, and delete the annotations for a given observation or medication record. Annotations may be queried by ID (getAnnotationByID), returning only a single annotation record. Annotations may also be queried by medication or observation id (getMedicationAnnotations and getObservationAnnotations), returning all annotations for the specified parent record id. Those operations are discussed in the relevant component sections (i.e., Medication List Operations and Observation Operations) in this document.

Note: When a Medication or Observation parent record is retrieved from the data repository, only the identifiers of any annotations associated with the record are returned (specifically, in the “AnnotationReferences” element). If a user wishes to see the actual text of these annotations, the PHA must separately retrieve (either by specific annotation id or by the medication/observation record id) the associated annotation records.

createAnnotation

Request: CreateAnnotationRequest

Once an observation or medication list has been created, users with the appropriate access control privilege may add annotations to that record. The operation for adding annotations to records is the same across all components that support annotations (currently only medication lists and observations).

Annotations may be up to 64 kilobytes in size.

Note: Special attention must be given to the handling of XML reserved characters in annotation text (i.e., the characters ‘<’, ‘>’, ‘&’, and ‘”’ [double quotation mark]). If these characters appear within the text of an annotation, they must be encoded as “<”, “>”, “&”, and “"” (respectively) within SOAP message sent to a platform components. Depending on the SOAP-message-generation software that the PHA

uses, the PHA may need to explicitly convert XML reserved characters to their encoded representations (“>”, “&”, and “"”) before submitting annotations to the platform components. Note that auto-generated client web service stub code may not necessarily perform this character conversion for the PHA.

Table 5: CreateAnnotationRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
PatientUniqueID	No	No	phd:UniqueIDType	The ID of the patient
ParentRecordUniqueID	No	No	phd:UniqueIDType	The ID of the record being annotated
ParentRecordStateData	No	NO	phd:RecordStateDataType	The current state information for the record being annotated. This must be provided to ensure that the client PHA has the most recent version of the parent record prior to adding an annotation.
AnnotationText	No	No	xs:string	The text content of the annotation. May be up to 64kb in size.

Note that the request must reference the parent record being annotated and the unique ID of the patient to whom that record belongs (via PatientUniqueID), in addition to including the AnnotationText itself. The request must also include the RecordStateData information for the parent record to ensure that the client PHA has the latest version of the parent record (the addition of an annotation to a record is considered an “update” of that record, and only the current version may be updated).

Response: CreateAnnotationResponse

The CPC service will return the RecordUniqueID and RecordStateData of the newly created annotation, as well as the ParentRecordUniqueID and the updated RecordStateData of the parent record.

updateAnnotation

Request: UpdateAnnotationRequest

When updating an annotation, the client PHA must provide the record state data information for the parent record, as well as for the Annotation record being updated. This ensures that the PHA has the latest version of the annotation record and the parent record prior to updating the annotation (updating an annotation to a record is considered an update of that record, and only the current version may be updated).

Response: UpdateAnnotationResponse

The CPC service will return the RecordUniqueID and updated RecordStateData of the updated annotation, as well as the ParentRecordUniqueID and the updated ParentRecordStateData of the parent record.

deleteAnnotation

Request: DeleteAnnotationRequest

When deleting an annotation the client PHA must provide the record state data information for the parent record as well as for the Annotation record being deleted. This ensures that the PHA has the latest version of the annotation record and the parent record prior to deleting the annotation (the deletion of an annotation to a record is considered an “update” of that record, and only the current version may be updated).

Response: DeleteAnnotationResponse

The CPC service will return the RecordUniqueID and updated RecordStateData of the deleted annotation as well as the ParentRecordUniqueID and updated ParentRecordStateData of the parent record.

getAnnotationByID

This operation returns the annotation record that corresponds to the annotation UniqueRecordID provided in the request. The option of requesting the current and/or historical version of the annotation record is available. Hence, multiple versions of the record may be returned.

Request: GetAnnotationByIDRequest

GetAnnotationByIDRequest requires the RecordUniqueID of the Annotation record being requested. The RecordState element is also required, but if no RecordState is specified, the current record will be requested by default.

Table 6: GetAnnotationByIDRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
RecordUniqueID	No	No	phd:UniqueIDType	ID of the annotation being requested
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value

Response: GetAnnotationsResponse

The component will return zero or more AnnotationRecord elements. The text of the annotation is contained in the AnnotationText element of these records.

Note: Special attention must be given to the handling of XML reserved characters in annotation text retrieved from the platform components (i.e., the characters '<', '>', '&', and '"' [double quotation mark]). If these characters appear within the text of an

annotation, the web-service components will always return these encoded as “<”, “>”, “&”, and “"” (respectively) within SOAP responses. Depending on the SOAP-processing software and the text-display software that the PHA uses, the PHA may need to explicitly convert these encodings back to their original representations (‘<’, ‘>’, ‘&’, and “”) before displaying the annotation text to users. Note that auto-generated client web service stub code may not necessarily perform this character conversion for the PHA.

Attachment Operations

createAttachment

Request: CreateAttachmentRequest

Once an observation or medication list has been created, users may add multi-media attachments to that record. The operation for adding attachments to records is the same across all components that support attachments (currently only medication lists and observations).

Attachments may be up to 16 megabytes in size when encoded as base64 strings.

Table 7: CreateAnnotationRequest Elements

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
PatientUniqueID	No	No	phd:UniqueIDType	The ID of the patient
ParentRecordUniqueID	No	No	phd:UniqueIDType	The ID of the record to which the attachment is being added.
ParentRecordStateData	No	No	phd:RecordStateDataType	The current state information for the record getting the attachment. This must be provided to ensure that the client PHA has the most recent version of the parent record prior to adding an attachment.
Attachment.FileDescriptor	Yes	No	xs:string	A short description of the attachment contents (optional). The purpose is to label links and thumbnails for the attachment.
Attachment.MimeType	No	No	xs:string {application/octet-stream, audio/mpeg, audio/x-ms-wma, audio/vnd.rn-realaudio,	Any allowed mime type may be used. It is the responsibility of the PHA to correctly assign the mime type

			audio/x-wav, image/gif, image/jpeg, image/png, image/tiff, text/css, text/html, text/plain, text/xml, video/mpeg, video/mp4, video/quicktime, video/x-ms-wmv,}	when creating an attachment record and to correctly render the mime type when retrieving an attachment.
Attachment.Value	No	No	xs:base64Binary	Base 64 encoded string. May be up to 16 MB in size.

You must reference the parent record that the data is being attached to in addition to providing the data for the attachment and the mime type when sending a `CreateAttachmentRequest`. The PHA may optionally provide a brief file descriptor for the attachment. This value may be a file name or some other human understandable information to help describe to contents of the attachment. The data of the attachment must be a base-64 encoded string. It is the responsibility of the PHA to base64-encode the binary content of the attachment, although most web service client code generators (e.g., Apache Axis2 and Microsoft wsdl.exe) provide facilities to convert binary data to base64 encoded strings. You must also provide the `RecordStateData` information for the parent record when creating an annotation. This ensures that the client PHA has the latest version of the parent record (the addition of an attachment to a record is considered an “update” of that record, and only the current version may be updated).

Response: `CreateAttachmentResponse`

The CPC service will return the `RecordUniqueId` and `RecordStateData` of the newly created attachment record as well as the `ParentRecordUniqueId` and updated `ParentRecordStateData` of the parent record that received the attachment.

deleteAttachment

Request: `DeleteAttachmentRequest`

When deleting an attachment, the client PHA must provide the record state data information for the parent record as well as for the attachment record being deleted. This ensures that the PHA has the latest version of the parent record prior to deleting the attachment (the deletion of an attachment from a record is considered an “update” of that record, and only the current version may be updated).

Response: `DeleteAttachmentResponse`

The CPC service will return the `RecordUniqueId` and updated `RecordStateData` of the deleted attachment as well as the `ParentRecordUniqueId` and the updated `ParentRecordStateData` of the parent record.

getAttachmentByID

Request: GetAttachmentRequest

GetAttachmentRequest requires the RecordUniqueID of the Attachment record being requested. Note that Attachment records cannot be updated, but only deleted and re-created.

Table 8: GetAttachmentRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
RecordUniqueID	No	No	phd:UniqueIDType	ID of the annotation being requested
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value.

Response: GetAttachmentResponse

The component will return zero or more AttachmentRecord elements. The attachment data will be returned as base64 encoded text. Most web service client code generators (i.e., Apache Axis2 and Microsoft wsdl.exe) provide facilities to convert base64 encoded strings back to binary data.

Fault Error Codes (In Progress)

Code	Description
100	Application's account name and/or password failed
101	Application could not be found
102	Application could not be created
103	Application's could not be deleted
200	User's account name and/or password failed
210	Neither a Last Name nor a Login Name was provided
300	The account name is already in use
400	Failed XML validation
420	Access denied
450	Invalid security token
451	Security token has expired
452	Invalid proof token
453	Missing security token
454	Security token not found
500	Relationship could not be created
510	Specified relationship cannot be found
520	Specified relationship is invalid
521	Relationship could not be updated
522	Relationship could not be deleted
550	Specified record is invalid
610	Specified rule cannot be found
611	Specified Rule could not be updated
612	Rule could not be created
613	Specified Rule could not be deleted

700	Annotation could not be created
701	Annotation could not be updated
702	Annotation not found
703	Annotation could not be deleted
800	Update/concurrency collision: another user has updated the record
850	A system error occurred
900	Not yet implemented
1000	Attachment could not be created
1001	Attachment could not be deleted
1002	Attachment not found
1100	Medication could not be created
1101	Medication could not be deleted
1102	Medication could not be updated
1103	Medication not found
1200	Observation could not be created
1201	Observation could not be deleted
1202	Observation could not be updated
1203	Observation not found
1300	Patient could not be created
1301	Patient could not be deleted
1302	Patient could not be updated
1303	Patient not found
1400	User could not be created
1401	User could not be deleted
1402	User could not be updated
1403	User not found
1404	Password not updated
1500	Missing required input data

Registry Service

User Registration Operations

registerUser

Request: RegisterUserRequest

See *WSDL documentation*. Note that login names must be unique within the registry and that login names are NOT case sensitive. Hence, a new login name may not match an existing login name, even if the case is different.

Response: RegisterUserResponse

See *WSDL documentation*.

getUserRecordsSimple

Request: GetUserRecordsSimpleRequest

The prototype Registry Service shall provide the ability to search for a user in the system based on login name or last name. This function allows users to retrieve other users' records for purposes of granting those users access to their PHR data. All users will be visible to other users and there will be no supported mechanism to hide a user account. It is expected that more complex user privacy policies will be implemented in a production level version of the Registry Service CPC.

When requesting user records, the PHA may search by either login name or last name. The Registry Service CPC will perform an exact-match lookup of the login name or last name, i.e., no wildcard characters or partial-name search is supported. However, the search is NOT case sensitive.

Table 9: GetUserRecordsSimpleRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
LoginName	Yes	No	xs:string	The login name of the account being queried for.
LastName	Yes	No	xs:string	The registered last name of the user being queried.

Response: GetUserRecordsResponse

The getUserRecordsSimple Operation will return 0 or more User records. The expected number of records returned for users based on a provided LoginName value is one because the LoginName must be unique within the registry. It is possible that the Registry Service CPC will return multiple records when the PHA queries by LastName. It is left to each PHA to determine how to best handle when multiple UserRecords are returned by the Registry Service CPC.

updateUserPassword

Request: UpdateUserPasswordRequest

See WSDL documentation. Note that a user's password may only be updated by the user himself/herself.

Response: UpdateUserPasswordResponse

See WSDL documentation.

updateUserDemographics

Request: UpdateUserDemographicsRequest

See WSDL documentation. Note that a user's demographic information may only be updated by the user himself/herself.

Response: UpdateUserDemographics

See WSDL documentation.

deleteUser**Request: DeleteUserRequest**

See WSDL documentation. Note that a user’s account may only be deleted by the user himself/herself.

Response: DeleteUserResponse

See WSDL documentation.

Patient Registration Operations**registerPatient****Request: RegisterPatientRequest**

See WSDL documentation. Any authenticated user may create a new patient record.

Response: RegisterPatientResponse

See WSDL documentation.

getPatientRecordsSimple**Request: GetPatientRecordsSimpleRequest**

The getPatientRecordsSimple operation allows PHAs to search for patient records by the last name of the patient. For this version of the service, only exact patient last name matches will be supported (i.e., no wildcards or partial name searches will be supported). However, the search is NOT case sensitive.

Table 10: GetPatientRecordsSimpleRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
LastName	No	No	xs:string	The registered last name of the patient being queried. Must match exactly to the registered patient last name. The search is not case sensitive.

Note: Any user may request patient records in this way, but the Registry service will only return those records to which the user has access-control privileges. Hence, a null result does not necessarily mean that no matching records exist in the database.

Response: GetPatientRecordsResponse

The getPatientRecordsSimple operation will return zero or more PatientRecord elements depending on (1) the number of matching last names in the Registry Service and (2) the patient(s) to whom the user has access privileges. Note that only those patient records to which the user has read access shall be returned by the Registry Service component, regardless of whether other patient records also match on last name.

updatePatientDemographics

Request: UpdatePatientDemographicsRequest

See WSDL documentation.

Response: UpdatePatientDemographicsResponse

See WSDL documentation.

deletePatient

Request: DeletePatientRequest

See WSDL documentation.

Response: DeletePatientResponse

See WSDL documentation.

Application Registration Operations

registerApplication

The registerApplication operation will not be exposed to PHA applications. All application registration will take place by a super user account available only to Project HealthDesign. If you need assistance registering your application please contact Sam Faus at Sujansky & Associates (sfaus@sujansky.com).

Request: RegisterAppRequest

See WSDL documentation. Only the super user account may register applications.

Response: RegisterAppResponse

See WSDL documentation.

getApplicationRecordsSimple**Request: GetAppRecordsSimpleRequest**

See WSDL documentation. Only the super user account may retrieve application records. If you need assistance reviewing your application or other applications, please contact Sam Faus at Sujansky & Associates (sfaus@sujansky.com).

Note: Applications cannot not be queried by ID.

Response: GetAppRecordsResponse

The getApplicationRecordsSimple returns a list of AppRecords, one for each registered application. The complete list of registered applications is always returned by the getApplicationRecordsSimple operation.

deleteApplication

The deleteApplication operation will not be exposed to PHA applications. All application administration will take place by a super user account available only to Project HealthDesign. If you need assistance administering your application please contact Sam Faus at Sujansky & Associates (sfaus@sujansky.com)..

Request: DeleteAppRequest

See WSDL documentation. Only the super user account may retrieve application records

Response: DeleteAppResponse

See WSDL documentation.

Authentication Service***Security Token Operations*****requestSecurityToken****Request: RequestSecurityToken**

The requestSecurityToken operation is used to authenticate a user (using a specific PHA) with the Authentication Service CPC. Authentication must precede any request to create, retrieve, update, or delete data in the system.

A security token request contains a number of parameters in addition to the login names and passwords of the user and application. These parameters specify the type of security token to create, the type of authentication request being made, and the mode of authentication. For the prototype, only the default values of these parameters are supported (see below).

Note that the user name, user password, application ID, and application password are passed within the SubmittedCredentials element.

Table 11: RequestSecurityToken Element

Element	Optional	Unbounded	Types {Values}	Comments
TokenType	No	No	xs:string {http://authentication.platform.projecthealthdesign.org/simpleSecureContextToken}	The type of security token to be generated. There is only one valid (default) value for the prototype.
RequestType	No	No	xs:string {http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue}	The type of authorization request. There is only one valid (default) value for the prototype.
AuthenticationType	No	No	xs:string {http://authentication.platform.projecthealthdesign.org/passwordsOnly}	The mode of authentication (e.g. password, biometric, etc.). There is only one valid (default) value supported in the prototype.
SubmittedCredentials.Username	No	No	xs:string	The username of user attempting to log in.
SubmittedCredentials.UserPwd	No	No	xs:string	The password of the user attempting to log in.
SubmittedCredentials.AppUniqueID	No	No	xs:string	The unique ID of the application the user is using to log in.
SubmittedCredentials.AppPwd	No	No	xs:string	The password of the application the user is using to log in.

Response: RequestSecurityTokenResponse

The requestSecurityToken operation will return a RequestSecurityTokenResponse message that contains a SecureContextToken, as well as the TokenType and RequestType values originally sent in the request. The SecureContextToken element contains the information necessary to authenticate subsequent requests to the common platform components (see Table 12). For detailed information on the use of the SecureContextToken to authenticate such requests, see the section in this guide on “Providing Authentication Information in Requests (In Progress).”

Table 12: SecureContextToken Element

Element	Optional	Unbounded	Types {Values}	Comments
SecurityTokenUniqueID	No	No	xs:string	This is the unique ID that references the SecurityToken created by the Authentication Service CPC and sent out to all connected CPCs. This ID must be sent out in all subsequent requests to HealthDesign CPCs.
UserUniqueID	No	No	phd:UniqueIDType	The User ID of the user who has been authenticated
AppUniqueID	No	No	phd:UniqueIDType	The App ID of the application that the user used to request the authentication (and the only application for which the SecureContextToken is valid).
ProofTokenEncryptionKey	No	No	xs:string	This is the key that must be used to encrypt the unique nonce value that the client PHA must generate, encrypt, and send with any subsequent requests to HealthDesign CPCs.
CreatedDateTime	No	No	xs:dateTime	The date/time that the SecureContextToken was created (can be used to calculate a standard expiration date/time)
ExpiresDateTime	Yes	No	xs:dateTime	The explicit date/time that the SecureContextToken expires. This element is optional and may not be sent by the Authentication CPC, in which case the datetime of expiration is at the discretion of each common platform component.

Medications Service

Medication List Operations

createMedRecord

Request: CreateMedRecordRequest

The createMedRecord operation allows PHAs to create medication records for a patient. When creating a medication, the client PHA must supply the id of the patient record and an Authentication element, along with the information for the medication record in a Medication element.

Table 13: Medication Element

Element	Optional	Unbounded	Types {Values}	Comments
MedicationType	No	No	Xs:string {Prescription, DispenseRecord, AdHoc	Describes the type of Medication record.
MedicationSubType	Yes	No	xs:string	A PHA controlled value that further defines the type of medication record. Can be used to specify the type of content included in the AdditionalStructuredData element.
MedicationIdentity	Yes	No	phd:OptionalCoded-Entity	Identity of the medication as prescribed, dispensed, or described by user (e.g. Lotensin). Allows for optional coding system use.
GenericIngredients. Ingredient	Yes	Yes	phd:OptionalCoded-Entity	The identity of the generic ingredient(s) of the medication (e.g., benazepril). Allows for optional coding system use.
UnitStrength	Yes	No	phd:QuantityOptional- CodedEntity	A description of the medication unit strength (e.g., “20 mg”). Allows for optional coding system use.
DosageForm	Yes	No	phd:OptionalCoded-Entity	A description of the form the medication takes (e.g., “tablet”, “capsule”). Allows for optional coding system use.
Route	Yes	No	phd:OptionalCoded-Entity	A description of the intended route of administration (e.g., “by mouth”, “intramuscular”). Allows for optional coding system use.
DispensedQuantity	Yes	No	phd:QuantityOptional- CodedEntity	A description of the intended or actual quantity dispensed (e.g., “100 tablets”). Allows for optional coding system use.
RefillsAuthorized.	Yes	No	n/a	Indicates if refills have been authorized for the medication. May be populated with either a NumRefills element that allows a numeric count of available refills or with a PRN element that contains an info element allowing the PHA to specify any addition information about the PRN

				refill status (e.g., the date the refill expires, etc.).
DosingFrequency	Yes	No	phd:OptionalCoded-Entity	A description of the medication dosing frequency (e.g., "Once per day"). Allows for optional coding system use.
DosingQuantity	Yes	No	phd:QuantityOptional-CodedEntity	A description of the intended dosing quantity (e.g., "2 tablets"). Allows for optional coding system use.
DosingInstructions	Yes	No	xs:string	A text-only description of the dosing instructions.
FullTextDescription	No	Yes	xs:string	A complete description of the Medication record. Any information about the medication, if reported in any element of the Medication record should be also reported here to support applications that cannot handle coded data representation of the medication components and to allow the import of entirely unstructured medication data from external sources.
OutOfPocketCostUS	Yes	No	xs:decimal	Out-of-pocket cost for the prescribed/dispensed quantity of the medication, in U.S. Dollars (e.g., 99.00)
AnnotationReferences. RecordUniqueID	Yes	Yes	phd:UniqueIDType	A list of RecordUniqueIDs referencing the annotations that have been added to the Medication Record. NOTE: This list is READ-ONLY. Any modifications made to this list via an update request will be ignored. To create/update/delete annotations for an observation use the Annotation Operations provided by the Medication Service CPC.
AttachmentReferences. AttachmentReference	Yes	Yes	phd:Attachment-Reference	A list of AttachmentReference elements that reference the attachments that have been added to the Medication Record. Includes the RecordUniqueID, mime type, file size, and a description of the

				attachment record. This list is READ-ONLY. Any modifications made to this list via an update request will be ignored. To create/update/ delete attachments for an observation use the Attachment Operations provided by the Observation Service CPC.
MedicationReferenceLinks.ReferenceLink	Yes	Yes	phd:ReferenceLink	Refers to another related Medication Record (using the RecordUniqueID of the medication record and an optional RelationshipType). The client PHA is responsible for correctly creating and managing these reference links and types.
CalendarReferenceLinks.ReferenceLink	Yes	Yes	phd:ReferenceLink	Refers to a CalendarRecord (using the RecordUniqueID of the calendar record and an optional RelationshipType). The client PHA is responsible for correctly creating and managing these reference links and types.
StructuredData	Yes	No	phd:AnyStructured-Data	Allows a client PHA to add any structured xml data to a MedicationRecord. The schema and content is completely at the discretion of the PHA. It is recommended that MedicationSubType be used to indicate the contents of the StructuredData.
DosingStatus	No	No	xs:string {Active, OnHold, Discontinued, Unspecified}	Indicates whether the patient is actively taking the medication or not.
EffectiveStartDate	Yes	No	xs:dateTime	Indicates the date on which the prescription was written, the medication was dispensed to the patient, or the patient indicates that she began taking the medication
EffectiveEndDate	Yes	No	xs:dateTime	Indicates the date on which the medication dosing status of a medication

				record was changed to "discontinued"
Provenance	No	No	xs:string {MedicalSource, PersonalSource, UnknownSource}	This coded flag denotes whether the observation originated from an authorized and trusted medical source (such as directly from an EHR or laboratory) or whether it originated from a patient or other lay caregiver or custodian.

Response: CreateMedRecordResponse

See WSDL documentation.

getMedRecordsSimple

Request: GetMedRecordsSimpleRequest

PHAs retrieving medication records may use the getMedRecordsSimple operation to query for a set of medication records for a particular patient. This section describes the server-side logic behind the getMedRecordsSimple operation call and what data a client PHA can expect to receive in a response to this call.

Table 14: GetMedRecordsSimpleRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
PatientUniqueID	No	No	phd:UniqueIDType	The ID of the Patient
MedicationType	Yes	Yes	xs:string {Prescription, DispenseRecord, AdHoc}	The type of medication record to return. If left blank, medication records of all types will be returned.
MedicationStatus	Yes	Yes	xs:string {Active, OnHold, Discontinued, Unspecified}	The status of the medication record to return. If left blank, medication records of all statuses will be returned.
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value.

Medication Record Query Examples

The values supplied for each of the elements are combined as a conditional "AND" join by the Common Platform Component server. For example, supplying the following values in a GetMedRecordsSimpleRequest:

```

.....
PatientUniqueID = "1234567890"
MedicationType = "Prescription"
RecordState = "Current"
.....

```

Results in the following query as interpreted by the component:

```
Get medication records where
the PatientUniqueID is "1234567890"
AND the MedicationType is "Prescription"
AND the RecordState is "Current"
```

In addition, when multiple values are provided for a single element (e.g., MedicationType and Medication Status) they are combined as a conditional “OR” join by the component. For example, supplying the following values in a **GetMedRecordsSimpleRequest**:

```
PatientUniqueID = "1234567890"
MedicationType = "Prescription"
MedicationType = "DispenseRecord"
MedicationStatus = "Active"
MedicationStatus = "OnHold"
RecordState = "Current"
```

Results in the following query as interpreted by the component:

```
Get medication records where
the PatientUniqueID is "1234567890"
AND (the MedicationType is "Prescription" OR
     "DispenseRecord")
AND (the MedicationStatus is "Active" OR "OnHold")
AND the RecordState is "Current"
```

Response: **GetMedRecordsResponse**

The `getMedRecordsSimple` Operation returns zero or more `MedRecord` elements that match the provided query parameters and user access privileges.

getMedRecordsByID

Request: GetMedRecordsByIDRequest

The `getMedRecordsByID` operation allows a PHA to request `MedicationRecords` based on the `RecordUniqueID` for that medication. The operation supports queries for the current and historical versions of the associated `MedicationRecord`. The PHA may provide any number of `RecordUniqueID` values in the request and is not limited to requesting `MedicationRecords` for a single patient record.

Table 15: GetMedRecordsByIDRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
RecordUniqueID	No	Yes	phd:UniqueIDType	PHAs may provide 1 or more medication <code>RecordUniqueID</code> values when requesting med records by ID.
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value

Response: GetMedRecordsResponse

The getMedRecordsByID operation returns 0 or more MedicationRecord elements depending on the number of RecordUniqueIDs requested and the user's access privileges with respect to those records.

updateMedRecord

Request: UpdateMedRecordRequest

See WSDL documentation.

Response: UpdateMedRecordResponse

See WSDL documentation.

deleteMedRecord

Request: DeleteMedRecordRequest

See WSDL documentation.

Response: DeleteMedRecordResponse

See WSDL documentation.

getMedicationAnnotations

Request: GetAnnotationsRequest

The Medications Service operation getMedicationAnnotations allows the client PHA to retrieve a list of all annotations for a given Medication record. The PHA may control whether current, historical, or deletion records related to the supplied medication RecordUniqueID are returned (or any combination thereof).

AnnotationRecords may be queried independently by using the operation getAnnotationsByID operation, also available from the Medications Service.

Table 16: GetAnnotationsRequest Element (Medications)

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
ParentRecordUniqueID	No	No	phd:UniqueIDType	The ID of the medication record
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value.

Response: GetAnnotationsResponse

The getMedicationAnnotations operation returns 0 or more annotations that are related to the MedicationRecord referenced by the ParentRecordUniqueID value.

Observations Service

Observation Operations

createObsRecord

Request: StoreObsRecordRequest

The createObsRecord operation allows PHAs to create observations of any supported type. When creating an observation the client PHA must supply the id of the patient record along with the information for the observation record in an Observation element (or a sub-type of an observation element).

The required elements of an Observation are dependent on the specific sub-type of observation being created. The attributes (required and optional) of a general observation are listed below. In addition, for each of the supported sub-types of observations (those derived from a general observation), the list of additional attributes that pertain only to that specific kind of observation are listed in separate tables.

Note that the value provided for the element ObservationType must be consistent with the type of Observation record being created. As such, the ObservationType element, along with the appropriate value for the ObservationType in question, is repeated for each type of observation listed below.

Every sub-type of observation supports the following elements

Table 17: GeneralObservation Element

Element	Opt- ional	Un- Bound- ed	Types {Values}	Comments
ObservationType	No	No	xs:string {GeneralObservation}	The kind of observation being created.
ObservationSubType	Yes	No	xs:string	PHAs may arbitrarily assign sub-types to observation records to assist in the processing of observations.
ObservationEntryDateTime	No	No	xs:dateTime	The date/time at which the observation was recorded by the Observation service CPC.
ObservationEffective- StartDateTime	No	No	xs:dateTime	The beginning of the time interval at which the recorded action, symptom, medical appointment, etc.

Element	Optional	Un-Bounded	Types {Values}	Comments
				actually took place. For a “point-in-time” observation the date/time should be recorded here as well as in ObservationEffectiveEndDateTime.
ObservationEffectiveEndDateTime	No	No	xs:dateTime	The end of the time interval at which the recorded action, symptom, medical appointment, etc. actually took place. For observations occurring at a single point in time, this value may be equal to ObservationEffectiveStartDateTime. For observations that occurred over a longer duration this element should be populated when the observation logically ended.
TextComment	Yes	No	xs:string	Stores an arbitrary text comment typically made by the recorder of the observation. This comment field is distinct from an “annotation”
AnnotationReferences. RecordUniqueID	Yes	Yes	phd:UniqueIDType	A list of RecordUniqueIDs referencing the annotations that have been added to the Observation Record. This list is READ-ONLY. Any modifications made to this list via an update request will be ignored. To create/update/ delete annotations for an observation use the Annotation Operations provided by the Observation Service CPC.
AttachmentReferences. AttachmentReference	Yes	Yes	phd:AttachmentReference	A list of AttachmentReference elements that reference the attachments that have been added to the Observation Record. Includes the RecordUniqueID, mime type, file size, and description of the attachment record. This list is READ-ONLY. Any modifications made to this

Element	Optional	Un-Bounded	Types {Values}	Comments
				list via an update request will be ignored. To create/update/ delete attachments for an observation use the Attachment Operations provided by the Observation Service CPC.
CalendarReferenceLinks. ReferenceLink	Yes	Yes	phd:ReferenceLink	Refers to a CalendarRecord (using the RecordUniqueID of the calendar record and an optional RelationshipType). The client PHA is responsible for correctly creating and managing these reference links.
ObservationReferenceLinks	Yes	No	phd:ReferenceLink	Refers to an Observation-Record (using the RecordUniqueID of the observation record and an optional RelationshipType). The client PHA is responsible for correctly creating and managing these reference links.
StructuredData	Yes	No	phd:AnyStructuredData	Allows a client PHA to add any structured xml data to an ObservationRecord. The schema and content is completely at the discretion of the PHA. It is recommended that ObservationSubType be used to indicate the contents of the StructuredData.
Provenance	No	No	xs:string {MedicalSource, PersonalSource, UnknownSource}	Indicates where the record originated and can be used in some cases to specify or determine the “source of truth” for an observation.

The following tables contain elements that are relevant only to specific sub-types of Observation records. All of the sub-type definitions listed below also include the elements listed under the GeneralObservation definition (i.e., the sub-types “inherit” elements from GeneralObservation).

Table 18: MedicationAdministration Element

Element	Optional	Un-Bounded	Types {Values}	Comments
---------	----------	------------	----------------	----------

Element	Optional	Un-Bounded	Types {Values}	Comments
ObservationType	No	No	xs:string {MedicationAdministration}	The kind of observation being created.
MedicationIdentity	No	No	phd:OptionalCodedEntity	Used to describe the medication administered. Can be text only or text with a coded component.
MedicationReferenceLink	Yes	No	phd:ReferenceLink	A link to the medication list record to which this observation refers
DoseAmount-Administered	Yes	No	phd:QuantityOptional-CodedEntity	The amount (number, volume, quantity) administered to the patient
PhysicalQuantity-Administered	Yes	No	phd:QuantityOptional-CodedEntity	Description of the administered quantity (e.g., 1, 2.5). Allows for optional coding system use.
RouteOfAdministration	Yes	No	phd:OptionalCodedEntity	Description of administration route (e.g., oral, handheld injection, pump injection, transdermal). Allows for optional coding system use
SiteOfAdministration	Yes	No	phd:OptionalCodedEntity	Description of administration site (e.g., thigh, shoulder). Allows for optional coding system use.
ReasonForAdministration	Yes	No	phd:OptionalCodedEntity	Description of the reason for administration (e.g., scheduled administration, PRN), Allows for optional coding system use.

Table 19: PhysicalActivity Element

Element	Optional	Un-Bounded	Types {Values}	Comments
ObservationType	No	No	xs:string {PhysicalActivity}	The kind of observation being created.
ActivityIdentity	No	No	phd:OptionalCodedEntity	Used to describe the kind of activity recorded in the observation. Can be text only or may contain optional coded identification of the activity.
ActivityDuration	Yes	No	phd:QuantityOptional-CodedEntity	A description of the duration of the physical activity (e.g., 30 minutes, 1 hour). Allows for optional coding system use.
ActivityIntensity	Yes	No	phd:QuantityOptional-CodedEntity	A description of the intensity of the physical activity (e.g., “Moderate”, “This one goes to 11”).

Element	Optional	Un-Bounded	Types {Values}	Comments
				Allows for optional coding system use.

Table 20: MealOrSnack Element

Element	Optional	Un-Bounded	Types {Values}	Comments
ObservationType	No	No	xs:string {MealOrSnack}	The kind of observation being created.
FullTextDescription	No	No	xs:string	A text description of the meal or snack
DiscreteFoodItems. FoodItem	Yes	Yes	obs:FoodItem	Repeatable discrete list of food items that are part of the meal or snack (see definition of FoodItem in the WSDL for the sub-structure of this element).

Table 21: SignOrSymptom Element

Element	Optional	Un-Bounded	Types {Values}	Comments
ObservationType	No	No	xs:string {SignOrSymptom}	The kind of observation being created.
SignOrSymptomIdentity	No	No	phd:OptionalCodedEntity	A Description of the sign or symptom as free text with optional coded components.
SignOrSymptomStatus	No	No	obs:PresenceStatus {Present, Absent, Unknown}	Present is the default value.

Table 22: Pain Element

Element	Optional	Un-Bounded	Types {Values}	Comments
ObservationType	No	No	xs:string {Pain}	The kind of observation being created.
PainAnatomicLocation	No	No	xs:string	Text description of the location of the pain
PainSeverity	No	No	phd:OptionalCodedEntity	Text description of severity with optional coded component (e.g., 1-10 scale)
PainRadiation	No	No	xs:boolean	“True” indicates the pain has a Radiating quality. “False” indicates that the pain does not radiate.
PainQuality	No	No	xs:string	e.g., “Throbbing”, “Sharp”, “Aching”, etc.

Element	Optional	Un-Bounded	Types {Values}	Comments
PainOnsetText	No	No	xs:string	A text representation of when the pain began (e.g., "Last week")
PainOnsetStructured	No	No	phd:TimestampInterval	A structured interval to indicate the start and end date/Times of when the pain occurred.
PainTemporalPattern	No	No	xs:string	e.g., "Constant", "Intermittent", "worsening during day"
PainAttemptedTreatments	No	No	xs:string	Text description of alleviation efforts
PainPerceivedPrecipitants	No	No	xs:string	Text description of the perceived cause of pain.

Table 23: ObservableParameter Element

Element	Optional	Un-Bounded	Types {Values}	Comments
ObservationType	No	No	xs:string {ObservableParameter}	The kind of observation being created.
ParameterIdentity	No	No	phd:OptionalCodedEntity	A description or name of the observable parameter. Allows for text with optional coding system.
ParameterValue	No	No	phd:ParameterValueType	The value of the observation. The ParameterValueType element allows for a choice between StringValue, NumericValue, StructuredNumericValue, and CodedEntityValue element types when populating this element.
ParameterValueUnits	Yes	No	phd:CodedEntity	Optional units used to qualify the ParameterValue element.
ParameterRecordingContextID	Yes	No	xs:string	The means by which the observed parameter was collected.

Response: StoreObsRecordResponse

See WSDL documentation.

getObsRecordSimple

Request: GetObsRecordSimpleRequest

The getObsRecordSimple provides client PHAs a mechanism for querying ObservationRecords. PHAs retrieving observation records may use the getObsRecordsSimple operation to query for a set of observation records for a particular patient. This section describes the server-side logic behind the getObsRecordsSimple operation call and what data a client PHA can expect to receive in response to this call.

Table 24: GetObsRecordSimpleRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
PatientUniqueID	No	No	phd:UniqueIDType	The ID of the patient
ObservationType	Yes	Yes	xs:string {GeneralObservation, HealthcareEncounter, JournalEntry, MealOrSnack, MedicationAdministration, ObservableParameter, Pain, PhysicalActivity, SignOrSymptom}	The type(s) of Observations to be returned. If left blank, Observations of all types will be returned
ObservationSubType	Yes	Yes	xs:string	The PHA-specified sub-type(s) of observations to be returned. Values may be any string.
ObservationEntry-DateTimeQry	Yes	No	phd:DateTimeQry	See below for a description of the phd:DateTimeQry element.
ObservationEffective-StartDateTimeQry	Yes	No	phd:DateTimeQry	See below for a description of the phd:DateTimeQry element.
ObservationEffective-EndDateTimeQry	Yes	No	phd:DateTimeQry	See below for a description of the phd:DateTimeQry element.
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value.

Querying Observation Records

The values supplied for each of the elements are combined as a conditional “AND” join by the Common Platform Component server. For example, supplying the following values in a GetObsRecordsSimpleRequest:

```
PatientUniqueID = "1234567890"  
ObservationType = "PhysicalActivity"  
RecordState     = "Current"
```

Results in the following query as interpreted by the component:

```
Get observation records where  
the PatientUniqueID is "1234567890"  
AND the ObservationType is "PhysicalActivity"
```

AND the RecordState is "Current"

In addition, when multiple values are provided for a single element (e.g., ObservationType) they are combined as a conditional "OR" join by the component. For example, supplying the following values in a GetMedRecordsSimpleRequest:

```
PatientUniqueID = "1234567890"  
ObservationType = "PhysicalActivity"  
ObservationType = "Pain"  
RecordState     = "Current"
```

Results in the following query as interpreted by the component:

```
Get medication records where  
the PatientUniqueID is "1234567890"  
AND (the ObservationType is "PhysicalActivity" OR  
      "Pain")  
AND the RecordState is "Current"
```

If no Observation type is provided as part of the request message then the Observation Service CPC will return observations of any type.

Querying Observations by Date/Time: DateTimeQry

Observations may also be queried based on the date/time that the observation was recorded (ObservationEntryDateTimeQry), the start time of the observation (ObservationEffectiveStartDateTimeQry), or the end time of the observation (ObservationEffectiveEndDateTimeQry) using the DateTimeQry element. The DateTimeQry element consists of an optional "After" value and an optional "Before" value.

Table 25: DateTimeQry Element:

Element	Optional	Unbounded	Types {Values}	Comments
After	Yes	No	xs:dateTime	The earliest date/time to begin matching records against (i.e., "find only those records with timestamp values equal to or later than the provided date/time value").
Before	Yes	No	xs:dateTime	The last date/time to match records against (i.e., find only those records with timestamp values equal to or before the provided date/time value").

The DateTimeQry element allows for retrieval of events after a given date/time, before a given date/time, or within a given range of date/times. For example, if a dateTime value of "2008-01-01T12:00:00-08:00" is provided for the "After" element of the ObservationEntryDateTimeQry, the component will return all observations that were recorded *on or after* noon, PST, January 1, 2008. (Note that the comparison is *inclusive* of the specified date/time). Similarly, when populating the Before field only, the component will return observations that were recorded *on or before* that provided date/time. To specify a range of dates between which observation records should be retrieved (e.g., all observations recorded between midnight on January 1, 2008 and January 15, 2008, inclusive) populate both "After" and "Before" elements of the

DateTimeQry. When the DateTimeQry is excluded from the query parameters observations entered at any date/time are returned by the Observation Service CPC.

The following query that includes a DateTimeQry element:

```
PatientUniqueID = "1234567890"
ObservationType = "MealOrSnack"
RecordState = "Current"
ObservationEntryDateTimeQry
  After = "2008-01-01T00:00:00-08:00"
  Before = "2008-01-15T00:00:00-08:00"
```

Results in the following query as interpreted by the component:

```
Get medication records where
the PatientUniqueID is "1234567890"
AND the ObservationType is "MealOrSnack"
AND (the ObservationEntryDateTime is after "2008-01-01T00:00:00-08:00"
AND before "2008-01-15T00:00:00-08:00")
```

Response: GetObsRecordsResponse

See WSDL documentation.

getObsRecordsByID

Request: GetObsRecordsByIDRequest

The getObsRecordsByID operation allows a PHA to request ObservationRecords based on the RecordUniqueID for that observation. The operation supports queries for the current and historical versions of the associated ObservationRecord. The PHA may provide any number of RecordUniqueID values in the request and is not limited to requesting ObservationRecords for a single patient record.

Table 26: GetRecordsByIDRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
RecordUniqueID	No	Yes	phd:UniqueIDType	PHAs may provide 1 or more medication RecordUniqueID values when requesting med records by ID.
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value

Response: GetObsRecordsResponse

The getObsRecordsByID operation returns 0 or more ObservationRecord elements depending on the number of RecordUniqueIDs requested and the user's access privileges with respect to those records.

updateObsRecord

Request: UpdateObsRecordRequest

See WSDL documentation.

Response: UpdateObsRecordResponse

See WSDL documentation.

deleteObsRecord

Request: DeleteObsRecordRequest

See WSDL documentation.

Response: DeleteObsRecordResponse

See WSDL documentation.

getObservationAnnotations

Request: GetAnnotationsRequest

The Observations Service operation `getObservationAnnotations` allows the client PHA to retrieve a list of all annotations for a given Observation record. The PHA may control whether current, historical, or all annotation records related to the supplied observation `RecordUniqueID` are returned.

AnnotationRecords may be queried independently by using the operation `getAnnotationsByID` operation, also available from the Observations Service.

Table 27: GetAnnotationsRequest Element (Observations)

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
ParentRecordUniqueID	No	No	phd:UniqueIDType	The ID of the observation record
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value.

Response: GetAnnotationsResponse

The `getObservationAnnotations` operation returns 0 or more annotations that are related to the `ObservationRecord` referenced by the `ParentRecordUniqueID` value.

Access Control Service

The Access Control service manages two types of information related to patient-record access: user-patient relationships and access-control rules. User-patient relationships describe the role that a user of a PHA has in relation to a specific patient record (e.g., “Physician”, “Parent”, etc.). These relationships are created and maintained by the owner of the patient record. Access-control rules specify the degree of access that a specific role has to various data in a specific patient record. Like user-patient relationships, access control rules are created and maintained by the owner of the patient record.

When PHAs make requests to platform components to read or write data, the components consult the access-control service to determine whether the requested operations are allowed, based on the source and nature of the request and the specified user relationships and access-control rules that have been defined. Note that the access-control rules are applied in a hierarchical fashion with respect to user roles, data types, and component operations. Specifically, user roles, data types, and platform operations are organized in hierarchies (see Appendix B), and access-control rules specified at more general levels or these hierarchies govern requests for services made at more specific levels. For example, an access control rule that allows a family member to modify any observation data will allow a parent to create a new Physical-Activity observation.

Access Control Rule Operations

createAccessRule

Request: CreateAccessRuleRequest

Access control rules specify allowed operations with respect to six dimensions:

<u>Patient Record</u>	(which patient does this rule relate to?)
<u>Operations</u>	(what operations does this rule allow or forbid?)
<u>ResourceID or ResourceType</u>	(to what instance or type of data does this rule pertain?)
<u>Role</u>	(for which user role does this rule allow or forbid access?)
<u>Context</u>	(to which application context does this rule pertain?)
<u>Action</u>	(does this rule <i>grant</i> [allow] or <i>deny</i> [forbid] the specified operation?)

Note that access-control rules can specify access with respect to individual record instances (“ResourceID”) or types of records (“ResourceType”). For example, an access control rule may grant record read access to a specific medication record or to all medication records of type “Prescription” .

Given the ability to grant or deny access, it is possible that multiple access-control rules may conflict. In these cases, the following rules apply:

- 1. If the operation is not allowed [granted] by at least one access-control rule, then it is prohibited.*

2. If the operation is allowed [granted] by at least one access-control rule and prohibited [denied] by no access-control rules, then it is allowed.

3. If the operation is prohibited [denied] by at least one access-control rule, then it is prohibited regardless of whether it is allowed [granted] by other rules.

In other words, access to all resources for all users is denied by default and must be explicitly granted, and the explicit denial of access to a resource takes precedence over the explicit granting of access.

To create an access control rule the PHA must specify the PatientUniqueID of the patient to whose record the rule applies. In addition, the PHA must populate the fields indicated in Table 28 below. For most access-control dimensions (with the exception of Operation and Action), the PHA may provide either an **Instance Record ID** or one of the specified **Record Categories**. A value for each access-control dimension must be provided when creating an access control rule. See Appendix B: Access Control Hierarchies for a detailed view of the supported values and how the values are hierarchically organized.

Table 28: Rule Element

Dimension	Instance Record	-or-	Categories: Types {Values}	Comments
Operation	N/A		Operation: xs:string {Insert, Update, Delete, Annotate, ReadRecord, ReadHistory, ReadDeletion, ReadAnnotation, AccessAdmin}	The kind of operational access being granted
Resource	ResourceInstanceID - A RecordUniqueid of the data. Used to grant access to a single record at a time)		ResourceType: xs:string {AllHealthData, AllMedicationListData, Prescription, DispenseRecord, AdHoc, AllObservationData, GeneralObservation, HealthCareEncounter, JournalEntry, MealOrSnack, MedicationAdministration, ObservableParameter, Pain, PhysicalActivity, SignOrSymptom}	The record or kind of record being accessed.
User	N/A (not currently supported)		Role: xs:string {RecordSubject, RecordCustodian, FamilyMember, Parent, Child, Sibling, HealthCareProvider, Physician, Nurse, Physician Assistant, Nutritionist, PersonalTrainer, PhysicalTherapist, GuestRecord1 – GuestRecord10}	The user or user role being granted access.
Context	ContextInstanceID (The application id of the application that may be used to access data).		ContextType: xs:string {AllApplications}	The application(s) the user may use to access data.
Action	N/A		xs:string {Grant, Deny}	Indication of whether the rule grants access to a resource or denies access to a resource.

Granting access to Individual Users: GuestRecords

In addition to the predefined relationships, the AccessControl CPC service also allows record owners to assign one of 10 pre-defined “guest record” roles to a user (GuestRecord1 – GuestRecord10). This feature allows the specification of access-control rules with respect to a single user, rather than a category of users (i.e., a role). Each patient record has a maximum of 10 guest record slots. A PHA may create a user-specific access control rule for a specific patient’s record via the following process:

1. Identify one of the predefined guest record roles that does not currently appear in any user relationship or any access-control rule for the specific patient record (e.g., “GuestRecord1”).
2. Create a user-relationship record that assigns the role “GuestRecord1” to the intended user for intended the patient record
3. Create the desired access-control rules for the role “GuestRecord1” and the intended patient record.

The net effect of these operations is that the user will be uniquely subject to the specified access control rules. *Caveats:* The user may also be subject to other access control rules if the user has been assigned other roles with respect to that patient record or if any global rules have been defined for the role “AllUsers”. Note: In the future, a more convenient and safe mechanism will be provided to automatically create user-specific access control rules.

Access Control Rule Creation Examples:

1. A rule that allows users with the role “FamilyMember” to read all Medication List records of type “Prescription” from any PHA.

```
...
<phd:PatientUniqueID>123456789@1.3.6.1.4.1.30291.0.1</phd:PatientUniqueID>
<Rule>
  <Operation>ReadRecord</Operation>
  <ResourceType>Prescription</ResourceType>
  <phd:Role>FamilyMember</phd:Role>
  <ContextType>AllApplications</ContextType>
  <Action>Grant</Action>
</Rule>
```

2. A rule that allows all users assigned by the patient record owner to the role “Nutritionist” to Annotate Observation Record 111@1.3.6.1.4.1.30291.0.3 (an instance record of an Observation of type MealOrSnack), but only from application 555@1.3.6.1.4.1.30291.0.1 (the application ID of a Diabetes management PHA).

```
...
<phd:PatientUniqueID>123@1.3.6.1.4.1.30291.0.1</phd:PatientUniqueID>
<Rule>
  <Operation>Annotate</Operation>
```

```

<ResourceInstanceID>111@1.3.6.1.4.1.30291.0.3</ResourceInstanceID>
<Role>Nutritionist</Role>
<ContextInstanceID>555@1.3.6.1.4.1.30291.0.1</ContextInstanceID>
<Action>Grant</Action>
</Rule>

```

3. A rule that denies all users all access to the medication record 222@1.3.6.1.4.1.30291.0.4 (an instance record of a medication that the record owner does not wish to share with anyone) from any PHA application.

```

...
<phd:PatientUniqueID>123@1.3.6.1.4.1.30291.0.1</phd:PatientUniqueID>
<Rule>
  <Operation>AllOperations</Operation>
  <ResourceInstanceID>222@1.3.6.1.4.1.30291.0.4</ResourceInstanceID>
  <Role>AllUsers</Role>
  <ContextInstanceID>AllApplications</ContextInstanceID>
  <Action>Deny</Action>
</Rule>

```

Response: CreateAccessRuleResponse

See WSDL documentation.

updateAccessRule

Request: UpdateAccessRuleRequest

See WSDL documentation.

Response: UpdateAccessRuleResponse

See WSDL documentation.

deleteAccessRule

Request: DeleteAccessRuleRequest

See WSDL documentation.

Response: DeleteAccessRuleResponse

See WSDL documentation.

getPatientAccessRules

Request: ListPatientAccessRulesRequest

The getPatientAccessRules operation allows PHAs to retrieve all of the access control rules for a given patient. It does not allow access control rules to be returned across a patient population (i.e. the component cannot provide a list of all patients that have

given a certain kind of access or have given access to a particular user). To retrieve the list of access control rules for a patient record, the PHA must provide the PatientUniqueID of the patient. Current, Historical, and Deletion records may be requested.

Table 29: ListPatientAccessRulesRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
PatientUniqueID	No	No	phd:UniqueIDType	ID of the user for who access rules are being requested
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value.

Response: ListPatientAccessRulesResponse

The getPatientAccessRulesRequest operation returns 0 or more access control RuleRecord elements.

Patient Record Relationship Operations

createRelationship

Request: CreateRelationshipRequest

Three pieces of information are required to create a user-patient relationship: The ID of the user record, the ID of the patient record, and the type of relationship being created.

Table 30: CreateRelationshipRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
PatientUniqueID	No	No	phd:UniqueIDType	The ID of the Patient record to which the relationship pertains
UserUniqueID	No	No	phd:UniqueIDType	The ID of the User record to which the relationship pertains
Role	No	No	xs:string {RecordCustodian, FamilyMember, Parent, Child, Sibling, HealthCareProvider, Physician, Nurse, Physician Assistant, Nutritionist, PersonalTrainer, PhysicalTherapist}	The set of relationships (roles) supported by the Access Control Service CPC. The Role "RecordSubject" is a relationship created automatically by the CPC to connect a user account as the primary subject of a patient record. This role type is not available to PHAs for manual relationship creation.

Response: CreateRelationshipResponse

See WSDL documentation.

updateRelationship

Request: UpdateRelationshipRequest

See WSDL documentation.

Response: UpdateRelationshipResponse

See WSDL documentation.

deleteRelationship

Request: DeleteRelationshipRequest

See WSDL documentation.

Response: DeleteRelationshipResponse

See WSDL documentation.

getPatientRelationships

Request: ListPatientRelationshipsRequest

The getPatientRelationships operation allows PHAs to request all of the registered user-patient relations for a given patient ID. The Access Control Service CPC supports requesting Current and Historical Relationships.

Table 31: ListPatientRelationshipsRequest Element

Element	Optional	Unbounded	Types {Values}	Comments
Authentication	No	No	phd:Authentication	Authentication credentials
PatientUniqueID	No	No	phd:UniqueIDType	The ID of the patient for whom the user-patient relationships are being requested.
RecordState	No	Yes	xs:string {Current, Historical, Deletion}	Current is the default value.

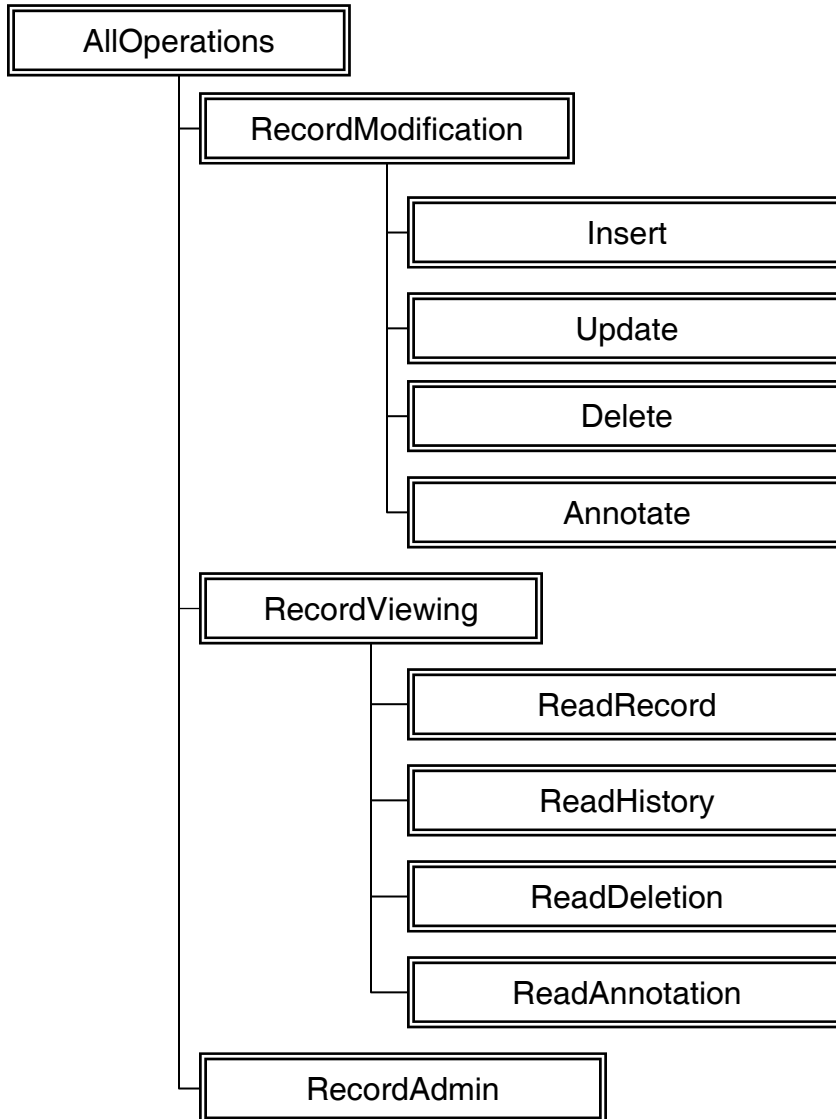
Response: ListPatientRelationshipsResponse

The getPatientRelationships operation returns one or more user-patient relationships. There will always be a minimum of one RelationshipRecord returned by the getPatientRelationships operation that ties a user account to a patient record with the relationship (or role) of "RecordSubject". Any other user-patient relationships that have been created for the patient record will also be returned.

Appendix A: Summary of Coded Values (Forthcoming)

Appendix B: Access Control Hierarchies

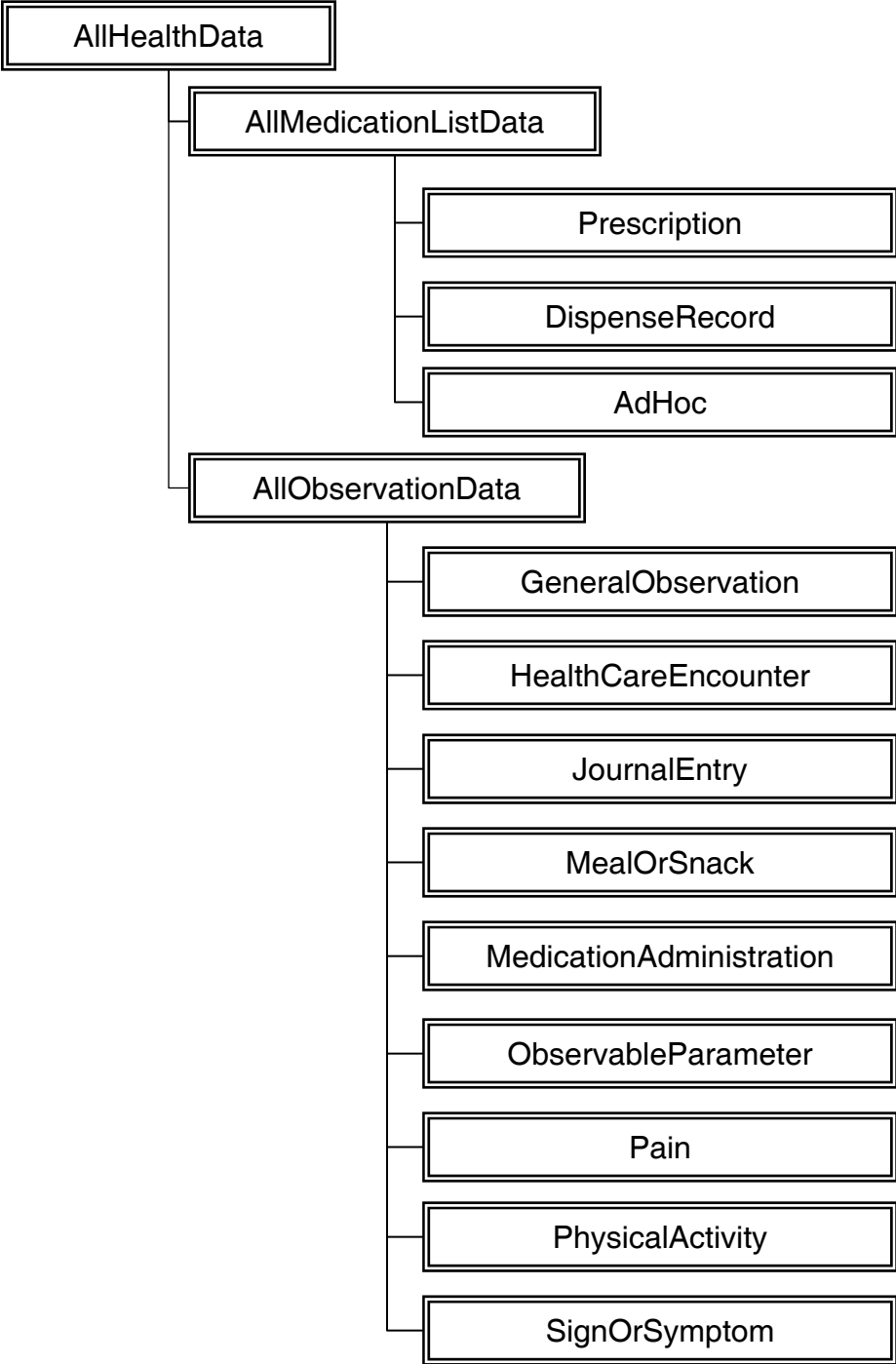
Operation Hierarchy



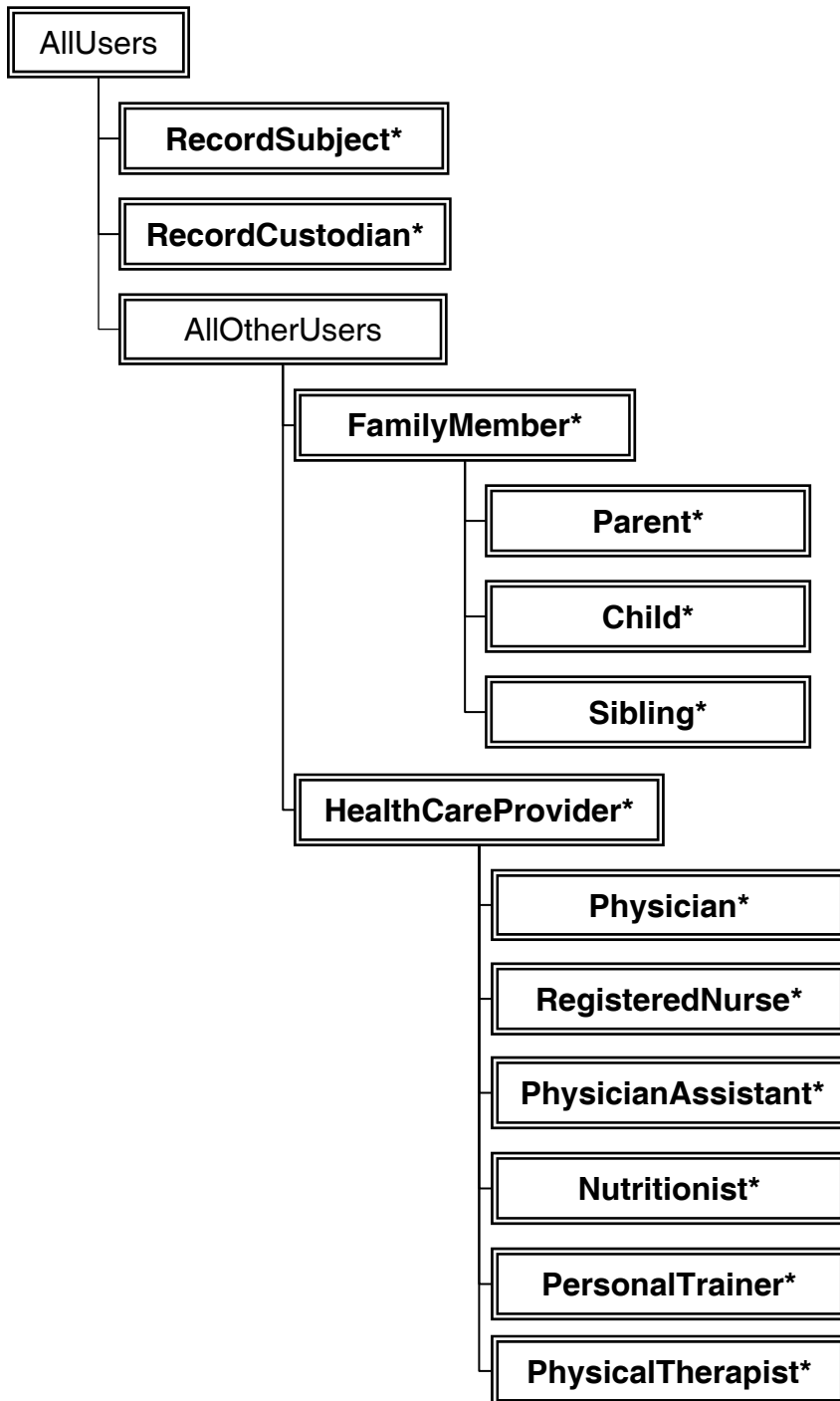
Context Hierarchy



Resource Hierarchy



User Hierarchy



* - Role may be used when creating a user-patient Relationship record