Specification for the Core Data Service

Contents

Overview

The Core Data Service is a RESTful service that provides methods for storing, retrieving and modifying records containing patients' medical information. The need for this service arose in a project to explore early draft versions of HL7's FHIR standard for exchanging clinical data among computer systems. This specification describes the interface service consumers use. The operations, data and behavior were implemented by software programs that realize the service.

In order to restrict limiting the content and organization of clinical data, the records are structured as JSON documents containing two information sets.

medical record = metadata + FHIR resource (in JSON format)

The core service supports multiple collections of such record. The metadata include a *classifier* to categorize clinical content and some *key properties* that identify individual records. The service API includes an operation to *create a record collection* (which may be empty or which may be initialized using data provided by the consumer). This operation returns a collection identifier that consumers must supply in order to store, read or update a record in that collection. There are four additional operations which can *list all the patient id's and descriptions* for records in a given collection, or *retrieve a summary record for a specified patient in a given collection*, or *replace a record* (update) in a given collection, or *store a new record* in a given collection. The core service does not examine a FHIR resource's content but allows a collection's creator to supply a method for doing so.

A clinical data service that works with the content of FHIR resources can be layered over this core service. See https://www.hl7.org/fhir/overview.html for current information about the FHIR standard.

Terminology

The specification uses certain terms and phrases with a technical meaning. See the following pages.

Terminology	Page	Notes
classifier	2	describes medical record
clinical data collection	3	aka CDC
medical record	2	encapsulates a FHIR resource
well-formed	3	well-formed Clinical Data Collection
well-formed medical record	2	
malformed	2	applied to medical record
malformed	3	applied to Clinical Data Collection

Record Descriptions

The term *medical record* is used to refer to a record that includes a FHIR resource. A metadata element called a *classifier* value is associated with a FHIR resource in order to categorize the kind of information in the resource. The service recognizes these classifier values: “patient”, “encounter”, “condition”, “medication”, “summary”. A medical record whose classifier value is "patient" is called a *patient description record*. (Future versions of the service specification may work with other classifier values.) When a record envelops a revisable FHIR resource, the service SHALL assign the revision id metadata element in order to distinguish one stored revision from another. Note that a medical record whose FHIR resource is non-revisable MAY include a revision id.

A *well-formed medical record* other than a patient’s summary record is a medical record that associates the following non-empty metadata character strings with the record’s FHIR resource:

- exactly one subject identifier
- exactly one classifier
- exactly one revision id
- exactly one timeStamp

A medical record's timeStamp is a date/time that indicates the moment it was added to the collection or was most recently changed. The key for the FHIR resource MUST be "doc". *Note that the doc value cannot be null or undefined (these aren't FHIR resources)*. See Core Service Operations below for more details.

A summary record is well-formed provided it has its own classifier element and a timeStamp indicating when the summary was created. A summary record doesn’t contain subject identifiers and revision id’s aside from the ones in its constituent medical records. See the description of the operation ***Retrieve Patient Summary Record*** below.

A medical record is *malformed* if it is not *well-formed*, i.e. doesn't contain the required metadata or whose doc value isn't a FHIR resource. An implementation which includes malformed records within a collection is a *non-conforming implementation*. (A non-conforming implementation might provide, inadvertently or by design, some method for adding records to a collection other than by using the operations defined in this specification.)

This specification of records adopts an "open world" perspective that permits a service implementation to include additional metadata, which the service's consumers MAY ignore but MUST tolerate.

Record Collections

The term *clinical data collection* refers to a collection of records that contains at least one medical record if it is not empty. It *MAY* contain other records as well. The core service works with multiple clinical data collections. Each clinical data collection has a *clinical data collection identifier* (cdcId) which differs from any other collection's cdcId. A clinical data collection is *well-formed* if and only if the following conditions are true.

1. The collection has one and only one cdcId identifier.
2. Every medical record in the collection is well-formed.
3. Two medical records in the collection with the same classifier have different subject identifiers.
4. Each encounter, medication or condition record in the collection has the same subject id as a patient description record in that collection.

NOTE. The term "collection" in this specification simply denotes a set of records. The empty set is also considered a well-formed clinical data collection. A collection's relationship to particular storage structures or to constructs provided by any particular database management system is an implementation detail which may differ from one implementation to the next. In particular, a collection is unlike a file or a container such as an array (for the actual contents of two different containers cannot overlap) but is like a set as understood in mathematics.

NOTE. A record with the same FHIR resource and metadata may be present in two different well-formed clinical data collections. A collection satisfying condition 1 and 2 but having no records also satisfies conditions 3 – 4 and, therefore, is well-formed. A well-formed collection containing a medical record must contain a patient description record according to condition 4. A collection MAY contain well-formed medical records with several different subject identifiers, but a collection without any patient description record or with not enough patient description records isn't well-formed.

A clinical data collection is *malformed* if it is not well-formed. An implementation which includes a malformed collection is a *non-conforming implementation*. Version 1 of the Core Data Service provides no operation that's guaranteed to detect a malformed collection.

A clinical data collection MAY have an associated *patient identity function* which can be applied to a patient description record's FHIR resource in order to obtain the patient's medical records number (aka "MRN") as well as the patient's full name and gender. See the descriptions of the Create Clinical Data Collection and List Patients operations.

Core Service Operations

The domain & port number portion of a URL is abbreviated by <…> in the following descriptions. All service URL's begin with the string <…>/fire/.

The term "token string" in this specification designates a string containing only the lower- or upper-case alphabetic characters "a"…"z" and the digits "0"…"9".

This specification assumes the HTTP 1.1 protocol is being used.

The *ver* identifier in a record takes the form *a.b* and identifies the version of the service whose specification describes that record. Don't confuse this with the revision identifiers associated with individual medical records. *The "semantic versioning" convention applies to the service version, but revision metadata format and content are established by service implementations subject to the restrictions in this specification of the operations for creating and updating a medical record*. For additional information see https://semver.org.

The *timeStamp* element present in each well-formed record is a string in ISO8601 standard format like 2014-11-07T01:45:02.887Z (yyyy-mm-ddThh:mm:ss.sssZ) or 2014-11-07T01:45:02.887-09:00 (yyyy-mm-ddThh:mm:ss.sss±hh:mm) where the UTC offset appears after the "+" or "-" character, "Z" being equivalent to "+00:00" (or "-00:00"). See https://en.wikipedia.org/wiki/ISO_8601. UTC offsets for some locations are given at https://en.wikipedia.org/wiki/List_of_UTC_time_offsets.

NOTE. An implementation of the Core Data Service *MAY* use the UTC zone for all its timeStamp values. Service consumers, therefore, may not be able to infer geographic location from a medical record's metadata.

NOTE. The term "parameters" in operation descriptions refers to data/information content that the operation requires clients to provide (designated by [in]) or that the operation generates and provides to clients (designated by [out]). The information may be part of the URL, part of request or of response messages. Required information in the URL or request message may be echoed in the operation's response message. Presence in the response message is not sufficient for an element to be designated an [out] parameter. (The notion of parameter and the [in] and [out] designations are being used similarly to but not exactly the same as programming languages use them). Variable elements in URL and request/response message specifications are enclosed in angle brackets < >. Parameters will be visually highlighted.

CAUTION. *Because service version is a required element in every request and response message, the operation's Parameters section will not explicitly describe it, although examples include it.*

Future revisions of the API specification as well as API client programs and API implementations MAY use the version element to support different message specifications.

Create Clinical Data Collection

Post

URL: <…>/fire/cdc.json
Request/Response format: JSON
Request body:
`{"ver":"1.0", "cdcId":"<prefix>", "load":"<name>"}`
Response status code: 200, 400, 500
Response:
body is `{"ver":"1.0", "cdcId":"<id>", "timeStamp":"<isoDate>"}` collection created with status=200;
body is `{"ver":"1.0", "code":"01", "text":"valid cdcId prefix or ver missing"}` no collection, status=400
body is `{"ver":"1.0", "code":"11", "text":"load parameter invalid"}` no collection, status=400
body is `{"ver":"1.0", "code":"12", "text":"invalid load record <sub>"}` no collection, status=400
body is `{"ver":"1.0", "code":"02", "text":"resources unavailable"}` no collection, status=500
Parameters:
[in] ***prefix*** *must be a token string whose length is at least 3 and no more than 8 characters;*
[in] (optional) ***name*** *is a string which refers to a CommonJS module whether stored locally or at a remote web site*
[out] ***id*** string identifying a new collection, ***isoDate*** is the time when the collection was created, ***sub*** indicates entries of the <***name***>.records array that contain an invalid medical record

Note: The response body itself is not a medical record – it doesn't contain a FHIR resource.

Example: `POST <…>/fire/cdc.json` with request body `{"ver":"1.0","cdcId":"amr"}` and response `{"ver":"1.0","cdcId":"amr-249","timeStamp":"2014-11-07T01:45:02.887Z"}` with status 200.

Effects

The optional request element *load* allows the service consumer to make available a set of well-formed medical records as the initial content of the collection. Note that these records SHOULD comprise a well-formed collection, but the operation does not verify the collection is actually well-formed. To specify initial content, set *load* to the name of a CommonJS module that exports an array named *records*. Each entry of the *records* array SHOULD be a medical record which is well-formed except for the possible omission of a timeStamp. If and only if the data for a medical record from the *load* module include no timestamp, the operation uses the current time and date to supply one before adding the record to the new CDC. The key for the FHIR resource MUST be "doc".

The *load* module MUST also export a function named *patientIdentity* which returns an object that provides identifying information about the subject of each patient description record in the collection:

`patientIdentity( doc) returns {mrn:"…", fullName:"…", gender:"…"}`

where `doc` is the FHIR resource from a patient record, `mrn` is a medical record number for the patient, `fullName` is the patient's name, `gender` is the patient's (HL7 administrative) gender. One or more of these elements may be "?" if the information cannot be determined from the FHIR resource. Different collections MAY have different patient identity functions.

The operation returns status 200 if and only if a new collection is created which contains only well-formed medical records if it isn't empty. If the *load* property was missing from the request or was the empty string, the new collection is empty.

The response body accompanying a status code 200 gives an identifier for the new, initialized Clinical Data Collection. This identifier will be two token strings separated by "-" (format is `<tk>-<tk2>`). The first token string is the prefix value given in the request body. The second token is supplied by the service. It is a string of no more than 4 characters.

A response status code = 400 with codes 01 or 11 indicates the request message is ill-formed, or has the wrong version, or the prefix value is not an acceptable token string, or the load parameter is malformed or its value doesn't name a CommonJS module that exports a *records* array and a patientIdentity function. Retrying with the same input will produce the same result.

A response status code = 400 with code 12 indicates an entry of the exported *records* array contains a malformed medical record or is null. The *sub* value indicates the subscript (0,1,…) of the invalid record. Retrying with the same input will produce the same result.

A response status code = 500 with code 02 indicates some system resources necessary to create another collection aren't available. The condition is assumed to be transient, i.e. a resolution of the problem(s) blocking creation of a new collection may be likely within a relatively short period of time. Retrying at a future time with the same input is likely to be successful. Resolution of the problem may require manual intervention.

List Patients

Get

---

*URL: <…>/fire/<**cdcId**>/ patient/list.json*
Response format: JSON
Request body: NONE
Response status code: 200, 400, 500
Response:
body is {"ver":"1.0", "cdcId":"<**cdcId**>", "list":[{"subject": "<**id**>", desc:{…}}...]}
status 200 when list is returned;
body is {"ver":"1.0", "code":"03", "text":"unknown collection or ver missing"} status 400, no list returned;
body is {"ver":"1.0", "code":"04", "text":"resources unavailable"} with status 500 and no list returned.
Parameters: [in] ***cdcId***, [out] ***id's***

Example: `GET /fire/amr-341/patient/list.json with response`
`{"ver":"1.0","cdcId":"amr-341","list":[{"subject":"34x9a-9!",desc:null}, {"subject":"332",desc:null}]}`
`with status = 200`

Effects

The operation returns status 200 if and only if the referenced collection is found and the operation returns a subject list for it. *A collection with no patient description records returns status 200 with an empty array [ ] for the list property*. The response message accompanying a 200 code lists both the subject id from each patient medical record in the referenced collection and a *desc* object which is returned by the patientIdentity function applied to the record's FHIR resource if there is a patient identity function associated with the collection. The desc will be null if the collection has no patient identity function. *Each id value appears only once in the list array.* (Note that a malformed clinical data collection could have several patient description records with the same subject id!)

If there's no collection with the given cdcId value, the operation returns a 400 response. Retrying with the same input and the same collection data will produce the same result.

A status of 500 indicates that a technical problem with the service's implementation has prevented creation of the list. The condition is assumed to be transient, i.e. a resolution of the problem(s) blocking creation of the list may be likely within a relatively short period of time. Manual intervention may be required. Retrying with the same input at a future time is likely to be successful.

NOTE. Version 1.0 of the service provides no operation to discard a Clinical Data Collection or to retrieve all the cdcId's of existing collections. This version of the core service treats collection management as an implementation detail.

Retrieve Patient Summary Record

Get

---

*URL: <…>/fire/<**cdcId**>/ patient/summary.json?id=<**id**>*
Response format: JSON
Request body: NONE
Response codes: 200, 400, 500
Response:
body is {"ver":"1.0", "cdcId":"<**cdcId**>", "classifier":"summary", "timeStamp":"<**isoDate**>",
"summary": <**{ … }**>} with status = 200;
body is {"ver":"1.0", "code": "05", "text":"invalid request: unknown collection/subject id or ver missing"} status = 400;
body is {"ver":"1.0", "code": "06", "text":"necessary resources unavailable"} status = 500;
Parameters:
[in] ***cdcId, id***
[out] ***isoDate*** (date/time summary record was created), **{ … }** (the summary object)

Example: GET <…>/fire/**amr-341**/patient/summary.json?id=**654321** with response
{"ver":"1.0",“cdcId”:"**amr-341**",“classifier”:"summary","timeStamp":"**2014-11-07T01:45:02.887Z**"
"summary":**{“patient”:{…},”encounters”:{…},…}**}

Effects

The operation returns status 200 if and only if it returns a patient summary record in the response. The summary property’s value in a response has the following structure. Each of the four properties must be objects with a value other than null.

```
{
  "patient": {“classifier”:”patient”,
              “subject”:"id",
              "revision":"rev-id",
              "doc":{patient description resource}
              },
  "encounters": {“classifier”:”encounter”,
                 “subject”:"id",
                 "revision":"rev-id",
                 "doc":{encounter resource}
                },
  "conditions": {“classifier”:”condition”,
                 “subject”:"id",
                 "revision":"rev-id",
                 "doc":{condition resource}
                },
  "medications": {“classifier”:”medication”,
                 “subject”:"id",
                 "revision":"rev-id",
                 "doc":{medication resource}
                 }
}
```

A summary returned by the service must include one and only one patient property whose value must not be { }. If the clinical data collection contains two patient description records with the same subject id, the record returned is an implementation choice. (Such a collection is malformed.) The value of an encounters, conditions or medications property is an empty object { } if and only if no record of that type is associated with the subject id. *The subject id's in the summary are identical to the id in the URL*. The summary itself does not have a single revision indicator for the record as a whole. The entire set of revision indicators of its constituents distinguishes one version of a patient's summary from another.

NOTE. Whether a summary is retained in a collection is an implementation decision not a requirement of the specification.

If the collection contains no patient description record for the subject id, the operation will return a status 400 with a code 5 error.

Retrying with the same input and the same patient records in the collection will produce the same status 400 result.

A status of 500 indicates that a technical problem with the service's implementation has prevented creation of the summary. The condition is assumed to be transient, i.e. a resolution of the problem(s) blocking a normal return is presumed to be likely within a relatively short period of time. Retrying at a future time is likely to be successful. Manual intervention may be required.

Update Medical Record

Put

---

*URL: <…>/fire/<**cdcId**>/ patient/<**classifier**>.json*
Request/Response format: JSON
*Request body: {"ver":"1.0", "subject": "<**id**>", "revision":"<**rev-id**>", "doc":<**FHIR resource**>}*
Response codes: 200, 400, 500
Response:

```
body is {"ver":"1.0","cdcId":"<cdcId>","classifier":"<classifier>", "subject":"<id>",
         "revision":"<new-rev-id>", "timeStamp":"<isoDate>"}, status = 200;
body is {"ver":"1.0", "code":"07", "text":"invalid request: unknown collection/subject/revision or
           ver missing "}, status = 400;
body is {"ver":"1.0", "code":"08", "text":"necessary resources unavailable"}, status = 500;
```

Parameters:
[in] ***cdcId, id, rev-id, classifier, FHIR resource***
[out] ***new-rev-id, isoDate***
cdc value: token string returned by the Create Collection operation; isoDate: date/time the record was updated
Classifier value: one of "patient", "encounter", "condition", "medication"

Example: `PUT /fire/amr-341/patient/condition.json`

```
with request body {"ver":"1.0","subject":"00123456789a","revision":"0","doc":{...}}
with response body {"ver":"1.0","cdcId":"amr-341","subject":"00123456789a",
                    "classifier":"condition","revision":"1",
                    "timeStamp":"2014-11-07T01:45:02.887Z"}
```

Effects

The operation doesn't require the collection or the record being updated to be well-formed. For example, it may update a record which has no FHIR resource or patient description record.

The operation returns status = 200 if and only if the following two conditions hold.

1. The request rev-id is not null and is equal to the rev-id of an existing medical record in the collection identified by <cdcId> and whose classifier matches <classifier> from the URL and whose subject id matches the id in the request body.
2. The doc value is neither null nor the object { } (these are not FHIR resources).

When the status = 200 the record of the <cdcId> collection whose metadata matched the values in the request has been replaced by a newly stored record whose FHIR resource is given by the request's doc element and whose classifier and subject id remain the same. The newly stored record's revision metadata is returned as the new-rev-id in the response. The new-rev-id will differ from the rev-id given in the request. The revision id's returned from any series of update requests for a record with a given pair (classifier, subject id) are guaranteed to be distinct. However, the same revision id may appear in two records whose subject id's are different or in two records with the same subject id but different classifiers. *Clients using the service SHOULD NOT assume that there is any meaningful ordering relationship between the request and response revision id's*. The composition of revision identifiers is determined by the particular implementation subject to restrictions given here, and clients may safely assume only the behavior and characteristics of revision id's that are described here.

When the status = 400 the request was missing a valid collection id, subject id, revision id or doc satisfying the conditions given above. Retrying with the same input will produce the same result.

When the status = 500 some service resources were inadequate to carry out the update. The condition is assumed to be transient, i.e. a resolution of the problem(s) blocking update of the record is presumed to be likely within a relatively short period of time. Retrying at a future time is likely to be successful.

NOTE: This operation does not examine doc contents. The service allows replacing an entire FHIR resource by a new version, and leaves to client programs the responsibility of controlling the changes they allow to individual items in the resource.

Store Medical Record

Post

---

*URL: <…>/fire/<**cdcId**>/patient/<**classifier**>.json*
Request/Response format: JSON
*Request body: {"ver":"1.0", "subject": "<**id**>", "doc":<**FHIR resource**>}*
Response codes: 200, 400, 500
Response:

```
body is {"ver":"1.0","cdcId":"<cdcId>","classifier":"<classifier>", "subject":"<id>",
         "revision":"<rev-id>", "timeStamp":"<isoDate>"}, status = 200;
body is {"ver":"1.0", "code":"09", "text":"invalid request", "reason": 1|2|3|4|5},
        status = 400;
body is {"ver":"1.0", "code":"10", "text":"necessary resources unavailable"}, status = 500;
```

Parameters:
[in] ***cdcId, id, classifier, FHIR resource***
[out] ***rev-id, isoDate***
cdcId value: token string returned by the Create Collection operation; isoDate: date/time the record was added to collection
Classifier value: one of "patient", "encounter", "condition", "medication

---

Example: POST <…>/fire/**amr-341**/patient/**encounter**.json

```
with request body {"ver":"1.0","subject":"00123456789a","doc":<FHIR resource>}
with response body {"ver":"1.0","cdcId":"amr-341","subject":"00123456789a",
```

```
                                    “classifier”:”encounter”,"revision":"0".
                                    "timeStamp":"2014-11-07T01:45:02.887Z"} with status = 200
```

This operation Stores the patient’s first encounter record (assuming the patient’s description record already exists). If the operation is rerun, the returned status = 400 and the response body is

```
{"ver":"1.0"code":"07", "text”: “invalid request”, “reason”:4}
```

Because the patient already has an encounter record, condition 4 below is violated.

Effects

The operation returns status 200 if and only if a new record is added to the collection identified by **cdcId**. The added record’s metadata are subject id = **id** from request’s *subject* parameter, classifier = **classifier** from the URL, revision = **rev-id** which the operation returns in the response message. The operation assumes that the FHIR resource for the new record is given by the request message’s *doc* element. The following conditions must be true for the service to return status 200.

1. The collection identified by **cdcId** exists.
2. The request’s *doc* element in neither null nor { }.
3. If the **classifier** is “patient”, the collection contains no patient record whose subject id = **id** in the request.
4. If the **classifier** is not “patient”, the collection contains a patient description record whose subject id = **id** in the request but does not contain a record whose classifier is **classifier** and whose subject id = **id**.
5. Request message contains a valid "ver" element.

A response status code = 400 indicates the request message is invalid for the reason indicated by 1, 2, 3, 4 or 5. Retrying with the same input (and patient records in the collection) will produce the same result.

A response status code = 500 indicates some system resources necessary to Store the record aren't available. The condition is assumed to be transient, i.e. a resolution of the problem(s) blocking creation of a new record is presumed to be likely within a relatively short period of time. Manual intervention may be required. Retrying at a future time is likely to be successful.