FRED API Specification Document

FRED API Version: 1.0.0  
Publication Date: [Publish Date]

# Document Information

## Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Version** | **Date** | **Rationale** |
| Mohawk College | 0.1 | 19-NOV-2012 | Initial Version |
|  |  |  |  |
|  |  |  |  |

## Related Documents

This document relies on or references the following documents

|  |  |  |
| --- | --- | --- |
| **Name** | **Url** | **Relation** |
| Collaborative Health Platform | <http://tinyurl.com/c4zhyru> | High level system description / role description. |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

Will introduce the project at a high level.

## Overview of the FRED API

High level overview of the FRED API

### Versioning of the FRED API Specification

All FRED API specifications published by the FRED team are assigned a unique version number un the format ***MAJOR.MINOR.REVISION***. These version numbers follow a semantic versioning pattern whereby:

1. ***REVISION*** is incremented for revisions to a ***MINOR*** version. These changes represent nonfunctional changes to the API.
2. ***MINOR*** version numbers are incremented when new functionality is introduced which is backwards compatible with existing functionality in the ***MAJOR*** version. ***MINOR*** versions numbers are semantically compatible with previous ***MINOR*** versions.
3. ***MAJOR*** version numbers are incremented when new functionality is introduced which is semantically incompatible with previous versions.

For example, a service implementing FRED API spec 1.1.0 can be consumed by clients implemented against the 1.0.0 version of the specification. A client implementing version 1.2.0 of the FRED API specification may consume services from registries implementing the 1.1.0 or 1.0.0 version of the API, however would not be capable of consuming 2.0.0 services.

## Definitions

1. “CHP” is used to describe the actors, transactions and roles described in the Collaborative Health Platform document released on the HUB in December 2011.
2. “Client” describes a consumer of health care services and it most often interchangeable with “patient”
3. “Facility” describes a logical place or point of care where health services are provided to clients.
4. “System” describes the overall health infrastructure, its components, interactions and actors. This term is often used to describe the overall health infrastructure in which the facility registry will operate
5. “Actor” is used to describe a series of responsibilities that a consumer or provider application must provide in order to participate in a clinical act. This term is interchangeable with “role” used in the CHP document.
6. “API” is used to describe an application programming interface which allows FRED Consumers to consume the services offered by the FRED Provider. It is the concrete data models and operations executed, at runtime, against application acting in the FRED Provider role.
7. “HIX” is a term used in the CHP document to describe a centralized health information exchange, and is used in this document in an informative manner.

## Purpose

Describe the purpose or goal of the document

## Scope

Identifies the scope of the document

## Standards & Real-world Architectures

Will relate the data collected in this document to our audience and the architectures currently deployed in Rwanda, etc.

## Collaborative Health Platform

This document is specifies both abstract data elements and concrete API definitions required for applications to maintain facility registry data. This functionality is a very close map to the abstract Facility Registry Service Supplier (ROL05) and Facility Registry Service Consumer (ROL06) roles defined in the CHP document.

This specification does not assume a full CHP infrastructure has been put in place, and has been specified to operate as a standalone service, or as part of a larger HIX. The specification will make reference to the CHP roles and interactions found between l. 535 (p. 24) and l. 635 (p. 27) of the CHP document.

# Reading this Document

This document will use several types of diagrams to illustrate how the actors within the system interact with one another. Where possible, this verbiage is aligned with the CHP roles/transactions, and many of the diagrams are common with the CHP framework document.

## Communications Diagrams

Communications diagrams are used to illustrate (at a high level) how consumer and provider roles interact with one another. Figure 1 illustrates a sample communication diagram whereby the FRED Provider (identified using CHP ROL05) interacts with the FRED consumer (identified using CHP ROL06). The figure illustrates that a consumer must send a query message (CHP FR03) to the service provider and must be capable of interpreting query results (CHP FR04).



Figure - Sample communications diagram

It is important to note that these communications diagrams have no implied order; they are simply used to show what needs to be sent/received by the actors.

## Model Diagrams

Although the first FRED RESTful APIs leverage JSON, the data model diagrams are illustrated as XML schema visualizations. This pattern was chosen because there is no formal, standardized way to represent JSON data contracts.

Simple data elements (strings, numbers, etc…) are represented as attributes and complex data elements (dates with precision, etc…) are represented as sequences. For example, the JSON object represented in Figure 2, would be visualized as represented in Figure 3.

"identity" : {

"agency" : "foo",

"context" : "foo",

"id" : "foo",

"nonStandardElement" : false

}

Figure - Sample JSON structure

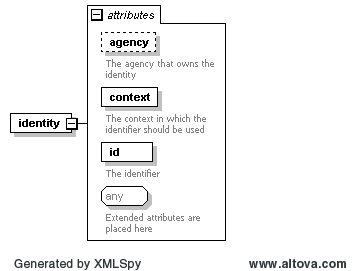


Figure - Sample model structure diagram

# FRED Transactions

All FRED transactions are RESTful operations against a collection of facility resources. All operations are performed against the facility resource which is further defined in Appendix A. Any operation that deviates from or restricts the facility resource will declare these modifications in the message semantics section.

## Communicate via HTTP

All actors which communicate with the facility registry must implement the HTTP transport (FRED transaction 1) as described in this section. Many standard stacks implement the HTTP transport itself, this section’s aim is not to specify HTTP, rather to constrain the meaning of HTTP aspects to facilitate interoperability between FRED actors.

### Scope

This transaction is global in scope. All actors must implement the “communicate via HTTP” transaction. Figure 3 illustrates the actors involved in this transaction.



Figure 4 - Actors communicating via HTTP

### Open Data Formats / Standards Referenced

This transaction makes use of the following standards:

* IETF RFC 2616 (HTTP 1.1)

### Interactions

FRED implementers are expected to support the HTTP interactions described in Figure 3.



Figure 5 - HTTP interactions between FRED actors

All communications between actors MUST be performed over HTTP 1.1 and MUST use the appropriate HTTP headers as specified in RFC 2616.

### Triggering Events

All communications MUST be solicited from a data source or consumer to the facility registry. The facility registry MUST NOT assume that a consumer/data source is capable of processing HTTP request messages.

#### POST

In the scope of the FRED registry API, the HTTP POST verb is reserved for any operation whereby a data source wishes to create a discrete record on the facility registry. In all cases a POST will never update information on the facility registry.

A facility registry MUST respond to HTTP POST messages using one of the codes listed in Table 1.

#### PUT

The HTTP PUT verb in the scope of the FRED registry API is reserved for any operation whereby a data source wishes to update an already existing record in the facility registry. In all cases PUT operations will fail whenever the requested resource does not exist.

A facility registry MUST respond to HTTP PUT messages using one of the codes listed in Table 1.

#### DELETE

In the FRED API, the HTTP DELETE verb is reserved for operations whereby data is removed or obsoleted from the registry. In all cases DELETE operations will fail whenever the requested resource does not point to a discrete record in the facility registry.

A facility registry MUST respond to HTTP DELETE messages using one of the codes listed in Table 1.

#### GET

HTTP GET is reserved from any operation that retrieves data from the facility registry. The GET verb MUST NOT modify any data in the facility registry and is considered a read only operation. GET operations against a collection MUST result in a query, returning zero or more matching results, whereas GET operations performed against a single resource MUST return zero or one results.

### Response Messages

All operations executed against the facility registry will result in an HTTP response containing one of the codes listed in Table 1. Implementers MUST be capable of generating/interpreting all error codes marked R for the specified trigger event.

Table 1 – HTTP response codes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Name** | **Opt** | **Scope** | **Trigger** |
| 200 | OK | R | All | Indicates that the specified action was successfully completed. |
| 401 | Unauthorized | R | All | Raised when the client attempts to perform an operation against a resource which requires authorization. This error code indicates a challenge for client credentials. |
| 403 | Forbidden | R | All | Indicates that the client does not have the necessary permission to perform the specified operation against the requested resource. |
| 404 | Not Found | R | GET | Indicates that a resource was not found or is not available. |
| 405 | Not Allowed | R | All | Indicates that the requested operation is not allowed on the current resource (for example: DELETE on a collection) |
| 409 | Conflict | R | POST | Indicates that the facility registry has detected a conflict in the operation and has refused to perform the operation. |
| 410 | Gone | O | GET | Indicates that a resource did exist but has been permanently removed. |
| 415 | Unsupported Media Type | O | POST, PUT | Indicates that the content supplied in the request is not supported by the facility registry. This code MAY be used when a client submits XML data to a facility registry which does not support XML formatting for facilities. |
| 422 | Invalid | R | POST, PUT | Indicates that the request is not well-formed, is missing data, or is semantically invalid. |
| 500 | Internal Server Error | R | All | Indicates that the server encountered an error while attempting to execute the desired action. |

Facility registries MUST provide detailed, structured error messages in JSON format to any client which has executed an operation triggering an HTTP 422 error. This structure is illustrated in Figure 3.

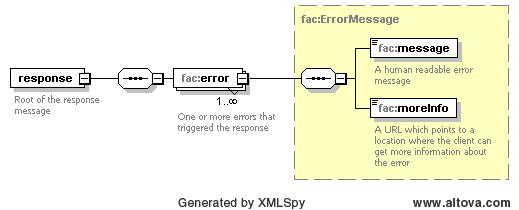


Figure 6 - Error message structure

Facility registries MAY leverage GZIP compression for response messages. When GZIP encoding is implemented, the registry MUST only apply GZIP compression when requesting systems indicate their support for decompression via the “Accept-Encoding” HTTP header.

Facility registry implementers MAY choose to leverage the HTTP headers “ETag” and/or “Cache-Control” if it is deemed necessary in the operating environment.

#### Examples

Figure 4 illustrates a sample response whereby the facility registry has determined the request to register a facility is invalid because of missing “name” and “lat” properties.

HTTP/1.1 422 INVALID

Content-Type: application/json

Date: Thu, 29 Nov 2012 15:36:56 GMT

Content-Length: XXX

Cache-Control: no-cache

{

"errors" : [

{

"message" : "Facility missing name",

"moreInfo" : "http://api.facilityregistry.org/errors/12345"

},

{

"message" : "Facility missing latitude",

"moreInfo" : "http://api.facilityregistry.org/errors/32143"

}

]

}

Figure 7 - Sample error response

### URLs

Facility registries MUST expose all FRED functionality for any one version of the FRED API specification at one base url. This specification does not prescribe the base URL for FRED resources, however all facility registry base URLs:

* MUST identify which FRED specification (version) is implemented this MAY be directly expressed in the URL (for example “v1.0”) or may be a semantic reference to the FRED API (for example /201301 points to v1.0 of the FRED spec)
* MUST NOT contain “facilities” as the last path portion on the URL
* Base URLs MUST be treated as permalinks
* One base URL MAY expose resources in any format which is semantically compatible with reported version (for example /v1 can be used to expose the most recent version of the 1.x FRED spec but cannot be used to expose 2.x FRED resources). For more information about FRED API versioning see “2.1.1 Versioning of the FRED API Specification” on page 3.

Examples of valid base URLs are:

* http://facilities.net/api/fred/1
* http://facilities.gov.ab/api/v1
* https://secure.facilities.org/api/fred/20130101

Examples of invalid base URLs are:

* http://facilities.net/fred (Does not expose the FRED version)
* http://facilities.net/api/fred/v1/v2 (Conflicting version numbers)
* http://facilities.net/api/fred/v1andv2 (Two semantically incompatible versions on the same base URL)

## Authenticate

The authenticate transaction .. Describe authentication here.

## Record and Maintain Facility Data

The record and maintain facility data transaction (FRED transaction 3) describes the processes under which a facility data source notifies the facility repository when new facilities are registered, or updated. This transaction fulfills FR03 (Register Facility) and FR05 (Update Facility) interactions identified in the CHP framework.

### Scope

The actors that are involved in this transaction are illustrated in Figure 3.



Figure – Record and maintain facility data actors

|  |  |
| --- | --- |
| **Actor** | **Purpose** |
| Facility Registry | Services facility data for an organization or jurisdiction. Responsible for the maintenance of that data over a long period of time and making that data available to other applications. |
| Facility Data Source | An application which is capable of generating or updating facility data. This application may provide a user interface for editing of facility details or may simply be an export function of a legacy system. |

### Use Case(s)

Place any use cases that support this transaction here, or merely reference them and provide them in an index.

### Open Data Formats / Standards Referenced

This transaction makes use of the following standards:

* IETF RFC 2616 (HTTP 1.1)
* IETF RFC 4627 (JSON Media Types)
* ISO 8601 (Date/Time Format)

### Interactions

Figure 4 illustrates the sequence of messaging between the Facility Data Source (FRED\_SRC) actor and Facility Registry (FRED\_REG).



Figure - Record and Maintain facility interactions

### Triggering Events

The facility data source will execute one of register facility or update facility events against the facility registry.

#### Register Facility

When a new facility has been added to the facility data source or when the facility data source deems it appropriate to share a facility it has not previously shared with the facility registry, it will notify the facility registry of this addition using the record facility transaction.

Changes to an existing record on the facility data source will trigger a revise facility action.

##### Message Semantics

**HTTP Method:** POST  
**Resource:** {base}/facilities  
**Content Type:** application/json  
Facility data sources MUST a JSON encoded facility resource as the payload of the HTTP message. All data sources MUST send the content-type header “application/json” which describes the type of data conveyed in the payload.

###### “url” / “id” Element Restrictions

The “url” and “id” elements of the facility resource MUST NOT carry a value on the register facility request as this value is to be populated by the facility registry.

###### “createdAt” / “updatedAt” Element Restrictions

The “createdAt” and “updateAt” elements of the facility resource MUST NOT carry a value on the registry facility request as these values are to be populated by the facility registry.

##### Examples

Figure 5 illustrates a sample request to create a facility named “Good Health Hospital” which was created sometime in November 2012.

POST [http://example.com/api/fred/1.1/facilities HTTP/1.1](http://example.com/api/fred/1.1/facilities%20HTTP/1.1)

Content-Type: application/json

Host: [example.com](http://www.example.com)

Content-Length: 567

{

"name" : "Good Health Hospital",

"id" : "urn:uuid:57A69100-26C4-4db4-897B-63F37866F0F5",

"url" : "http://example.com/api/fred/1/facilities/1304954",

"identifiers" : [

{

"agency" : "MOH",

"context" : "HR",

"id" : "20294"

}

],

"active" : true,

"created\_at" : "2012-11",

"lat" : "1.69172",

"long" : "29.52505",

"links" : [

{

"name" : "providers",

"url" : "http://providers.moh.gov.za/providers?fac=20294"

}

]

}

Figure - Sample register facility operation

##### Expected Behavior

When the facility registry receives a request to register a facility, the facility registry will first check its current datastore to determine if an existing facility already exists. The matching algorithm used by the facility registry is not specified in this document and should be whatever algorithm is deemed appropriate for the deployment environment of the facility registry.

Depending on the outcome of the match one of two actions are to be taken by the facility registry:

1. If the facility registry determines that the facility has already been registered, it will return a 409 error signaling to the data source that the record already exists. The method by which the facility registry determines duplicate registrations is not specified in this document.
2. If the facility registry finds no matching facilities on file, it will create a new facility entry.

After the facility registry has completed its write operation, it MUST make the facility data available to consumers and MUST respond with an HTTP 200 code with the url of the newly created facility in the format “{base}/facilities/{id}”. The facility registry MUST NOT make facility records available prior to ensuring all facility data has been committed to its datastore (partial data MUST NOT be disclosed or available for update).

#### Revise Facility

When a facility registration record changes in the facility data source’s datastore, it will notify the facility registry of this change using the revise facility operation.

##### Message Semantics

**HTTP Method:** PUT  
**Resource:** {base}/facilities/{id}  
**Content Type:** application/json  
Facility data sources MUST submit a JSON encoded facility resource as the payload of the HTTP message. All data sources MUST send the content-type header “application/json” which describes the type of data conveyed in the payload.

Revision operations MUST be executed against the fully qualified url for the resource which is being updated, and the body of the update request MUST contain both the url and id elements.

##### Examples

Figure 6 illustrates a sample update to the Good Health Hospital facility (identified as resource #10252152).

PUT http://example.com/api/fred/11/facilities/1304954 HTTP/1.1

Content-Type: application/json

Host: example.com

Content-Length: 734

{

"name" : "Good Health Hospital",

"id" : "urn:uuid:57A69100-26C4-4db4-897B-63F37866F0F5",

"url" : "http://example.com/api/fred/1/facilities/1304954",

"identifiers" : [

{

"agency" : "MOH",

"context" : "HR",

"id" : "20294"

},

{

"agency" : "UNICEF",

"context" : "DHIS",

"id" : "58845858"

}

],

"active" : true,

"createdAt" : "2012-11",

"updatedAt" : "2012-12-09T14:55:23Z",

"closedAt" : "2013-01",

"lat" : "1.69172",

"long" : "29.52505",

"links" : [

{

"name" : "providers",

"url" : "http://providers.moh.gov.za/providers?fac=20294"

}

]

}

Figure - Sample update facility message

##### Expected Behavior

When the facility registry receives a request to revise a facility, the facility registry MUST validate that the requested facility exists. If the requested target of revision (the facility to be updated) does not exist the facility registry MUST reply with an HTTP 404 error.

If the facility resource exists, the facility registry MUST update its datastore with the new information and MUST respond with an HTTP 200 response code. The facility registry MUST only update fields that were provided in the update payload. Any fields missing MUST be considered unchanged. The facility registry MUST return a complete copy of the facility record containing the applied revisions.

#### Delete Facility

When a facility record is no longer relevant, or was created in error, the facility data source will notify the facility registry of this change using the obsolete facility.

##### Message Semantics

**HTTP Method:** DELETE  
**Resource:** {base}/facilities/{id}

The request to delete a facility MUST NOT contain any data.

##### Response Message Semantics

Todo

##### Examples

Todo

##### Expected Behavior

When the facility registry receives a request to delete a facility, the facility registry MUST validate that the facility exists. If the requested target of deletion does not exist, the facility registry MUST respond with an HTTP 404 error.

If the facility resource exists, the facility registry MUST delete the facility resource such that the record is no longer discoverable to consumers. The process by which the facility registry marks the facility as deleted is not specified in this document, and is left to implementers to determine the most appropriate method. If a deleted resource is requested by a consumer, the server MAY return an HTTP 410 error but MUST at minimum return an HTTP 404 error.

Once the record is deleted, the facility registry MUST return an HTTP 200 response with the URL of the deleted facility.

### Extended Properties

The facility registry MUST be capable of receiving facility resources which have extended properties and MUST be capable of storing the data contained within these extended elements. Any extended properties MUST be returned back to consumers when the resource is fetched.

There is no requirement that the facility registry be able to meaningfully process additional elements outside the scope of the core facility resource defined in Appendix A.

## Query Facility Data

The query facility data transaction (FRED transaction 4) describes the process whereby a facility data consumer queries and consumes facility data from a facility registry.

### Scope

The actors that are involved in this transaction are illustrated in Figure 6.



Figure – Record and maintain facility data actors

|  |  |
| --- | --- |
| **Actor** | **Purpose** |
| Facility Registry | Services facility data for an organization or jurisdiction. Responsible for the executing of queries, and fetching of facility detail data. |
| Facility Data Consumer | An application which is capable of constructing and consuming queries against the facility registry. |

### Use Case(s)

Place any use cases that support this transaction here, or merely reference them and provide them in an index.

### Open Data Formats / Standards Referenced

This transaction makes use of the following standards:

* HTTP 1.1
* IETF RFC4627

### Interactions

Figure 7 illustrates the sequence of messaging between the Facility Data Consumer (FRED\_CONSUMER) actor and Facility Registry (FRED\_REG).



Figure - Record and Maintain facility interactions

### Triggering Events

The facility registry consumer will execute one of query facility or get facility details to list and/or get facility details respectively.

#### Query Facilities

Todo

##### Message Semantics

**HTTP Method:** GET  
**Resource:** {base}/facilities[.json|.xml]

Facility data consumers MUST execute a GET against the facilities collection to initiate a query. Consumers MUST instruct the facility registry to return data in JSON format by appending an extension of “.json” to the request URL.

Registry implementers may choose to support XML as an alternate representation of facility resources. Consumers may request this by appending “.xml” as an extension to the GET request. If the registry does not support XML as a return format it MUST respond with an HTTP 404 error.

Consumers MUST pass query parameters to the facility registry via query parameters in the format *propertyName=filterValue*.

Query parameters MUST be passed as one value per parameter. Query parameters are identified in Table 1 and MUST map to core properties with the same name. Query parameter values MUST be URL encoded when sent to the facility registry service.

Table - Query facility property filter parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Opt** | **Formatⱡ** | **Description** | **Example** |
| active | R | Boolean | Performs a filter on the active field of the facility. | ?active=true |

ⱡ - See appendix A for formatting of these data-types

Implementers MAY choose to extend the available query parameters made available to consumers. When extended query parameters are implemented, they MUST be implemented such that:

* The name of the query parameter exactly matches the name of a property the parameter filters, and
* Repetitions of the same named parameter MUST be considered an OR operation. For example, if a facility registry supports filtering on creation date then a filter for all facilities created in January or February of 2012 would be represented as: “?created\_at=2012-01&created\_at=2012-02”, and
* Implementers must declare which extended query parameters they support.

Query operations also expose a series of query control parameters. These parameters do not map directly to core facility properties are provided to control the result. Table 2 lists the query control parameters defined in the FRED service.

Table - Query facility query control parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Opt** | **Formatⱡ** | **Description** | **Example** |
| allProperties | R | Boolean | When specified, instructs the facility registry to return all properties stored for a facility resource. | ?allProperties=true |
| updatedSince | R | Date | Instructs the facility registry to limit the results to only those updated since the specified date. | ?updatedSince=2011-01-01 |
| fields | R | String[] | A list of comma-separated strings which identify which properties should be returned in the result. | ?fields=name,id |

ⱡ - See appendix A for formatting of these data-types

Consumers MUST NOT pass more than one repetition of any one query control parameter.

##### Response Message Semantics

The response for the query facilities message

###### “url” Element Restrictions

The “url” and “id” elements of the facility resource MUST NOT carry a value on the register facility request as this value is to be populated by the facility registry.

The facility registry MUST generate a globally unique identifier for all facilities which it registers, and MUST make this identifier available via the “id” element. The type of identifier generated is not specified here however it MAY be representable using a URI syntax and MUST be globally unique. Some recommended identifier formats are:

* UUIDs in the format : urn:uuid:E8A1650D-7FF9-4d3e-B390-D6FDD4CFB2E0
* URLs which point to the resource: http://example.com/api/fred/1.1/ facilities/10293
* ISO OIDs in the format: urn:oid:1.3.6.1.5.6.7.8343

###### “x” Element Restrictions

Todo

##### Examples

Examples

HTTP/1.1 200 OK

Content-Type: application/json

Date: Thu, 29 Nov 2012 15:36:56 GMT

Content-Length: 1194

{

"facilities" : [

"facility" : {

"name" : "Good Health Hospital",

"id" : "urn:uuid:57A69100-26C4-4db4-897B-63F37866F0F5",

"url" : "http://example.com/api/fred/1/facilities/1304954",

"identifiers" : [

{

"agency" : "MOH",

"context" : "HR",

"id" : "20294"

},

{

"agency" : "UNICEF",

"context" : "DHIS",

"id" : "58845858"

}

],

"active" : true,

"created\_at" : "2012-11",

"updated\_at" : "2012-12-09T14:55:23Z",

"closed\_at" : "2013-01",

"lat" : "1.69172",

"long" : "29.52505",

"links" : [

{

"name" : "providers",

"url" : "http://providers.moh.gov.za/providers?fac=20294"

}

]

},

"facility" : {

...

}

],

"queryAck" : {

"current" : 2,

"offset" : 0,

"total" : 20

}

}

Figure - Sample register query facilities response

##### Expected Behavior

Todo

# Facility resource implementation details

This appendix describes additional details related to the

Include a schema diagrams here.

# XML representation of facility resources

All FRED resources may be represented in XML. This section describes the transformation operations that are required to display/render FRED facility resources in XML.

# Facility Registry Behaviors

## Reporting of Errors

If the facility registry encountered an internal error (datastore is not available, full, etc…) the facility registry MUST respond with an HTTP 500 error, signaling that an internal registry problem occurred. Implementers may choose to use extended HTTP 500 error codes to convey more detailed error messages. These extended 500 error codes are outside the scope of this specification however must adhere to HTTP status code conventions.