FHIR Bulk Data Proposal

Overview

Providers and organizations accountable for managing the health of populations often need to efficiently access large volumes of information on a group of individuals. For example, a health system may want to periodically retrieve updated clinical data from an EHR to a research database, a provider may want to send clinical data on a roster of patients to their ACO to calculate quality measures, or an EHR may want to access claims data to close gaps in care. Currently, this is often done by exporting a specific subset of fields from the source system into a file format like CSV, transmitting these files to a server, and then importing the files into the target system. The effort involved in manually mapping the data fields to read and write these extracts raises the cost of integration projects and can act as a counter incentive to data liquidity.

Existing FHIR APIs work well for accessing small amounts of data, but large exports can require hundreds of thousands of requests. This document outlines a standardized, FHIR based approach to bulk data export.

Request Flow

Authorization

Bulk data servers should implement the OAuth based SMART backend services authorization process. On the requests outlined below, clients should include an `Authorization` header containing the bearer token received from the authorization flow. If the server responds to a request with a `401 Unauthorized` header, the client should follow the authorization flow to obtain a new token.

Bulk Data Kick-off Request

This FHIR Operation initiates the asynchronous generation of data files for all patients or a group of patients contained in a FHIR server.

Note: Only data the client application has authorization to access and that the relevant business agreements allow should be returned.

Endpoint - All Patients

`GET [fhir base]/Patient/$export`

Endpoint - Group of Patients

`GET [fhir base]/Group/[id]/$export`

FHIR Operation to obtain data on all patients listed in a single FHIR Group Resource. Note: How these groups are defined will be implementation specific for each clinical system. For example, a payer may send a healthcare institution a roster file that can be imported into their EHR to create or update a FHIR group. FHIR based roster management is out of scope for the bulk data project, but would be a valuable project.

Headers

- `Accept` (required)

  Specifies the format of the optional OperationOutcome response to the kick-off request. Currently, only application/fhir+json is supported.

- `Prefer` (required)

  Specifies whether the response is immediate or asynchronous. Currently must be set to `respond-async`.

Query String Parameters

- `_outputFormat` (string, optional, defaults to `application/fhir+ndjson`)

  The format for the generated bulk data files. Currently, only ndjson is supported. Servers should support the full content type of `application/fhir+ndjson` as well as abbreviated representations including `application/ndjson` and `ndjson`.

- `_since` (FHIR instant type, optional)

  Resources updated after this period will be included in the response

Note: This parameter was named `start` in an earlier version of this proposal

- `_type` (string of comma-delimited FHIR resource types, optional)

  Only resources of the specified resource types(s) will be included in the response. If this parameter is omitted, the server should return all supported resources that the client has authorization to access and that the relevant business agreements allow. The Patient Compartment should act as a point of reference for recommended resources to be returned as well as other resources outside of the patient compartment that are helpful in interpreting the patient data such as Organization and Practitioner.

  Note: Some implementations may limit the resources returned to specific subsets of FHIR like those defined in the Argonaut Implementation Guide

Response - Success

- HTTP Status Code of `202 Accepted`
- `Content-Location` header with a url for subsequent status requests
- Optionally a FHIR OperationOutcome in the body

Response - Error (eg. unsupported search parameter)

- HTTP Status Code of `4XX` or `5XX`

- Optionally a FHIR OperationOutcome in the body

Bulk Data Delete Request:

After a bulk data request has been kicked-off, clients can send a delete request to the url provided in the `Content-Location` header to cancel the request.

Endpoint

`DELETE [polling content location]`

Response - Success

- HTTP Status Code of `202 Accepted`
- Optionally a FHIR OperationOutcome in the body

Response - Error Status

- HTTP status code of `4XX` or `5XX`
- Optionally a FHIR OperationOutcome in the body

Bulk Data Status Request:

After a bulk data request has been kicked-off, clients can poll the url provided in the `Content-Location` header to obtain the status of the request.

Note: Clients should follow the an exponential backoff approach when polling for status. Servers may supply a Retry-After header with a http date or a delay time in seconds. When provided, clients should use this information to inform the timing of future polling requests.

Endpoint

`GET [polling content location]`

Response - In-Progress Status

- HTTP Status Code of `202 Accepted`
- Optionally an `X-Progress` header with a text description of the status of the request that's less than 100 characters. The format of this description is at the server's discretion and may be a percentage complete value or a more general status such as "in progress". Clients can try to parse this value, display it to the user, or log it.
- Optionally a FHIR OperationOutcome in the body

Response - Error Status

- HTTP status code of `5XX`
- Optionally a FHIR OperationOutcome in the body

Response - Complete Status

- HTTP status of `200 OK`

- `Content-Type header` of `application/json`

- Optionally an `Expires` header indicating when the files listed will no longer be available.

- A body containing a json object providing metadata and links to the generated bulk data files.

  Required Fields:

  - `transactionTime` - a FHIR instant type that indicates the server's time when the query is run. No resources that have a modified data after this instant should be in the response.
  - `request` - the full url of the original bulk data kick-off request
  - `secure` - boolean value indicating whether downloading the generated files will require an authentication token. Note: This may be false in the case of signed S3 urls or an internal file server within an organization's firewall.
  - `output` - array of bulk data file items with one entry for each generated file. Note: If no data is returned from the kick-off request, the server should return an empty array.

  Each file item should contain the following fields:

  - `type` - the FHIR resource type that is contained in the file. Note: Each file may only contain resources of one type, but a server may create more than one file for each resources type returned. The number of resources contained in a file is may vary between servers. If no data is found for a resource, the server should not return an output item for it in the response.

  - `url` - the path to the file. The format of the file should reflect that requested in the `_outputFormat` parameter of the initial kick-off request.

Example response body:

```
{
  "transactionTime": "[instant]",
  "request" : "[base]/Patient/$everything?_type=Patient
,Observation",
  "secure" : true,
  "output" : [{
    "type" : "Patient",
    "url" : "http://serverpath2/patient_file_1.ndjson"
  },{
    "type" : "Patient",
    "url" : "http://serverpath2/patient_file_2.ndjson"
  },{
    "type" : "Observation",
    "url" : "http://serverpath2/observation_file_1.ndjs
on"
  }]
}
```

File Requests:

Using the urls supplied in the completed status request body, clients can download the generated bulk data files (one or more per resource type). Note: These files may be served by a file server rather than a FHIR specific server. Also, if the `secure` field in the status body is set to `true` the request must include a valid access token in the `Authorization` header in these requests.

Endpoint

`GET [url from status request output field]`

Headers

- `Accept` (optional, defaults to `application/fhir+ndjson`)

Specifies the format of the file being returned. Optional, but currently only application/fhir+ndjson is supported.

Response - Success

- HTTP status of `200 OK`
- `Content-Type` header of `application/fhir+ndjson`
- Body of FHIR resources in newline delimited json - ndjson format

Response - Error

- HTTP Status Code of `4XX` or `5XX`

Out of scope in v1 of this specification

- Legal framework for sharing data between partners - BAAs, SLAs, DUAs should continue to be negotiated out-of-band
- Real-time data (although data loaded through bulk data can be supplemented at with synchronous FHIR REST API calls)
- Data transformation and transmission - different step of the ETL process
- Patient matching (although, it’s possible to include identifiers like subscriber number in FHIR resources)
- Management of FHIR groups within the clinical system - the bulk data operation will require a valid group id, but does not specify how FHIR Groups resources are created and maintained within a system

Server Implementations

- SMART (supports v0.1)

  https://github.com/smart-on-fhir/bulk-data-server (code)

  https://bulk-data.smarthealthit.org (online)

- HL7 (supports v0.1)

https://test.fhir.org/r3

- Cerner (supports v0.1)

  https://fhir-open.stagingcerner.com/stu3/a758f80e-aa74-4118-80aa-98cc75846c76/Patient/$everything

- ONC (supports v0.1)

  http://52.70.192.201/open-fhir/fhir/ (open)

  http://52.70.192.201/secure-fhir/view/newuser.html (registration)

  http://52.70.192.201/secure-fhir/fhir/ (secure)

Client Implementations

- SMART NodeJs (supports v0.1)

  https://github.com/smart-on-fhir/sample-apps-stu3/tree/master/fhir-downloader

- Python client application to stream data from a Bulk FHIR API into BigQuery, determining schema on the fly (supports v0.1)

  https://github.com/jmandel/synthea-to-bigquery/tree/bulk-data-scratch/bulk-data-loader

- Python client with auth support that converts a bulk data server into a generator for lightweight iteration through bulk resources returned (supports v0.2)

  https://github.com/plangthorne/python-fhir (code)

  https://github.com/plangthorne/python-fhir/blob/master/demo/BulkDataDemo.ipynb (demo notebook)

- Go client that fetches data and stores to local FS, google cloud storage and/or export to bigquery (supports v0.1)

  https://github.com/toby-hu/test/tree/master/client

Participate!

- Join the “Bulk Data” stream on https://chat.fhir.org
- Open an issue or pull request at https://github.com/smart-on-fhir/fhir-bulk-data-docs/

Change Log:

1/27/2018 (Draft v0.1)

- Moved output links from `Link` header to body and defined an output JSON format
- Added `output-format` parameter to differentiate between the format of the response to the request and the format of the linked

output files

1/28/2018 (Draft v0.2)

- Renamed `output-format` parameter to `_outputFormat` since it can apply to any async request
- Added `DELETE` method to status endpoint
- Changed the operation name to `$export` (under discussion)
- Changed `start` parameter to `_since` to align with existing FHIR semantics

2/1/2018 (Draft v0.2.1)

- Added recommendation around use of `Retry-After` header