hData Packaging and Network Transport Specification v0.3

Gerald Beuchelt, Robert Dingwell, Andrew Gregorowicz, Harry Sleeper

The MITRE Corporation
202 Burlington Rd.
Bedford, MA 01730
U.S.A.

1 Introduction

The hData Record Format (HRF) [1] describes the XML representation of the continuity of care information in an electronic health record (EHR). This specification creates a compact serialization format for the entire HRF , and a network transport API for accessing components of an HRF.

1.1 Namespaces

This document uses the following namespaces. This specification uses a number of namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Namespace Prefix	Namespace URI	Description
hrf	http://projecthdata.org/hdata/schemas/2009/06/core	Namespace for elements in this document

1.2 Glossary (Non-Normative)

hData Record Format (HRF) - The part of the hData specification that defines the abstract hierarchy, meta-data schema, and document organization of the hData record.

hData Record (HDR) - an single instantiation of the HRF.

hData Restful API (HRA) - the part of the hData specification that defines the basic HTTP-based API for accessing or modifying an HDR.

hData Specification - a normative specification that defines the HRF, the HRA, and a file-based serialization format.

hData Content Profile (HCP) - a profile of the medical content of an HDR. An HCP is specified separately from the HRF. The hData Project defines an initial HCP (iHCP) that covers the 35 data elements for EHRs/EMRs defined by the National Quality Foundation.

Electronic Medical Record (EMR) - the medical record or records of a single patient in the IT system of an actor (health provider, government entity, payer, etc.). In this definition, an HDR is a type of EMR.

Electronic Health Record (EHR) - the collection of all EMRs of a single patient, across organizational and national boundaries.

EHR System - An IT system that creates, stores, and manages EMRs.

Clinical Document Architecture (CDA) - an XML specification by Health Layer 7 (HL7) that is intended to be used for EMRs.

Continuity of Care Record (CCR) - a specification by ASTM that is intended to be used for summary/continuity of care documentation. A CCM is a type of EMR.

Continuity of Care Document (CCD) - a profile of the CDA that accommodates the medical information of the CCR.

HITSP/C32 (C32) - a constrained profile of the CCD that is intended to simplify implementation and improve interoperability. There is no normative schema for C32. Note that HITSP has recently split up C32 into HITSP/C80 and HITSP/C83.

MITRE/L32 (L32) - a significantly constrained profile of the C32 specification. L32 comes with a normative schema and can be mapped onto the HRF.

1.3 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

When describing concrete XML schemas, this specification uses the following notation: each member of an element's [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard. The use of @{any} indicates the presence of an attribute wildcard.

Note also that only the W3C XML schemas linked in Appendix A at the end of this document are normative – any schema fragment or other schema description is informational only.

2 File System Serialization

An HDR can be serialized for transporting the entirety of all data contained in the HDR. Serialization is a simple process that uses the ZIP archive format [2] in the same way the Open Document Format does. Sections are represented as file system folders and the root document and section documents are

stored in as serialized XML files, using UTF-8 encoding. The following process can be applied to UNIX and Windows file systems.

2.1 Section Mapping

The first step in the file system serialization process is to map existing sections to a file folder structure on disk. For this, the *base folder* is selected, which MUST be completely empty. For each section directly below the `<sections>` element in the root document, a new file folder with name equal to the `path` attribute of the corresponding `<section>` element is created in the *base folder*.

The same process is then iteratively applied to each child node of the first level `<section>` nodes and their children. The result is a folder hierarchy which represents the section structure of the HDR.

2.2 Document Serialization

The root document is serialized into the *base folder* created in section 2.1. For each `<section>` under the `<sections>` node in the root document, all documents belonging into this section are enumerated, serialized into XML standalone documents with UTF-8 encoding and stored in with file name "*document name*.xml" in the file folder that represents the section. If two documents have an identical document name, any UTF-8 encoded character-based discriminator may be appended to the document name, but prior to the ".xml" extension.

This process is performed on all `<section>` nodes and all of their children.

2.3 Packaging

Once all sections, the root document, and all section documents are represented in the file folder hierarchy, a ZIP archive [2] is created of the *base folder* and all its descendant files and folders.

3 Network Transport and RESTful API

Any HDR can be represented as HTTP resources in a canonical way. The entire HDR is referenced by a base URL which depends on the implementation. This base URL will be denoted as *baseURL* in the following. This specification does not dictate the format of the base URL, but it is NOT RECOMMENDED to use matrix parameterization. All content within an HDR MUST be expressible as a HTTP resource. In the following, the minimum version for HTTP is 1.1, unless explicitly specified otherwise.

This specification does not define any access controls to the web resources. It is RECOMMENDED that a comprehensive access control management system is always deployed with any hData installation.

Any GET, PUT, POST, or DELETE operations on a given resource that are either (i) unspecified or (ii) not implemented MUST return an HTTP status code of 405. All operations may return HTTP status codes in the 5xx range if there is a server problem.

3.1 Operations on the Base URL

3.1.1 GET

If there is no HRF at the base URL, the server SHOULD return a 404 - Not found status code.

Status Code: 404

3.1.1.1 Default

There is no defined default behavior for an HTTP GET to the *baseURL*. It is RECOMMENDED that a GET returns a human-friendly user interface that represents the content of the HDR.

Status Code: 200

3.1.1.2 TE Header: Deflate

If the HTTP TE header contains "deflate", the server MAY return a ZIP archive containing the serialized version of the current content of the entire HDR. Implementations MAY also define other mechanisms to obtain a serialized version of the HDR, but these MUST NOT collide with any provisions of this specification.

Status Code: 200

3.1.1.3 Parameter: type

The parameter "type" MUST have value "sections". The server MUST return an Atom 1.0 compliant feed of all child sections, as identified in the corresponding sections node in the root document. Each entry MUST contain a link to the resource for each child section.

If type has any other value, the server MUST return a 404 status code.

Status Code: 200, 404

3.1.2 POST

3.1.2.1 Parameters: type, typeId, requirement

For this operation, the value of type MUST equal "extension". The typeId MUST be a URI string that represents a type of section document. The requirement parameter MUST be either "optional" or "mandatory". If any parameters are incorrect or not existent, the server MUST return a status code of 400.

If the system supports the extension identified by the typeId URI string, this operation will modify the extensions node in the root document and add this extension with the requirement level identified by the requirement parameter. The server MUST return a 201 status code.

If the system does not support the extension, it MUST not accept the extension if the requirement parameter is "mandatory" and return a status code of 409. If the requirement is "optional" the server MAY accept the operation, update the root document and send a status code of 201.

Status Code: 201, 400, 409

3.1.2.2 Parameters: type, typeId, path, name

The type parameter MUST equal "section". The typeId MUST be an URI that is registered in the extensions node of the root document of the HDR identified by *baseURL*. The path MUST be a string that can be used as a URL path segment. If any parameters are incorrect or not existent, the server MUST return a status code of 400.

The system MUST confirm that there is no other section registered as a child node that uses the same path name. If there is a collision, the server MUST return a status code of 409.

If the typeId is note registered as a valid extension, the server MUST return a status code of 406. It MAY provide additional entity information.

When creating the section resource, the server MUST update the root document: in the node of the parent section a new child node must be inserted. The server MUST return a 201 status code and SHOULD include the location of the new section.

Status Code: 201, 400, 406, 409

3.1.3 PUT

This operation MUST NOT be implemented.

Status Code: 405

3.1.4 DELETE

This operation MAY be implemented, but special precaution should be taken: if a DELETE is sent to the *baseURL*, the **entire** HDR represented by the *baseURL* is completely deleted. Future requests to the *baseURL* SHOULD return a status code of 410, unless the record is restored. Any DELETE operation MUST by logged by the underlying system.

Status Code: 204, 410

3.2 *baseURL*/root.xml

3.2.1 GET

3.2.1.1 Default

This operation return an XML representation of the current root document.

Status Code: 200

3.2.2 POST, PUT, DELETE

This operations MUST NOT be implemented.

Status Code: 405

3.3 *baseURL/sectionpath*

3.3.1 GET

3.3.1.1 Default

This operation MUST return an Atom 1.0 compliant feed of all section documents contained in this section. Each entry MUST contain a link to a resource that uniquely identifies the section document. If the section document type defines a creation type, is is RECOMMENDED to set the Created node to that datetime. The Content node MAY contain the XML representation of each section document.

Status Code: 200

3.3.1.2 Parameter: type

The parameter "type" has two allowed values: "documents" and "sections". If "type" is of value "documents", the server MUST return the same content as for a Default GET to the section resource.

If "type" is "sections", the server MUST return an Atom 1.0 compliant feed of all child sections, as identified in the corresponding sections node in the root document. Each entry MUST contain a link to the resource for each child section.

If type has any other value, the server MUST return a 404 status code.

Status Code: 200, 404

3.3.2 POST

Both POST operations MUST provide a type parameter which MUST be either of value "section" or "document". For "section", three additional parameters are required, and the POST will create a new child section within this section. For "document" an XML document MUST be sent that conforms to the schema identified by the typeId attribute of this section.

3.3.2.1 Parameters: type, typeId, path, name

The type parameter MUST equal "section". The typeId MUST be an URI that is registered in the extensions node of the root document of the HDR identified by *baseURL*. The path MUST be a string that can be used as a URL path segment. If any parameters are incorrect, the server MUST return a status code of 400.

The system MUST confirm that there is no other section registered as a child node that uses the same path name. If there is a collision, the server MUST return a status code of 409.

When creating the section resource, the server MUST update the root document: in the node of the parent section a new child node must be inserted. The server MUST return a 201 status code.

Status Code: 201, 400, 409

3.3.2.2 Parameters: type, Content Type: application/xml

When sending a section document, type MUST be of value "document", and the content MUST conform to the XML schema that corresponds to the typeId of this section. If the parameter is incorrect or the

content cannot be validated against the schema identified by the typeId of this section, the server MUST return a status code of 400.

If the request is successful, the new section document MUST show up in the document feed for the section. The server returns a 201.

Status Code: 201, 400.

3.3.3 PUT

This operation MUST NOT be implemented.

Status Code: 405

3.3.4 DELETE

This operation SHOULD be implemented, but special precaution should be taken: if a DELETE is sent to the section URL, the **entire** section, its documents, and subsections are completely deleted. Future requests to the section URL MAY return a status code of 410, unless the record is restored. Any DELETE operation MUST by logged by the underlying system.

Status Code: 204, 410

3.4 *baseURL/sectionpath/documentname*

3.4.1 GET

This operation returns a UTF-8 encoded representation of the document that is identified by *documentname* within the section identified by *sectionpath*. The *documentname* is typically assigned by the underlying system and is not guaranteed to be stable across two different systems. Implementations MAY use identifiers contained within the infoset of the document as *documentnames*.

If no document of name *documentname* exists, the implementation MUST return a HTTP status code 404.

Status Codes: 200, 404

3.4.2 PUT

This operation is used to update a document. The content MUST conform to the XML schema that corresponds to the typeId of this section and the media type MUST be application/xml. If the parameter is incorrect or the content cannot be validated against the schema identified by the typeId of this section, the server MUST return a status code of 400.

If the request is successful, the new section document MUST show up in the document feed for the section. The server returns a 200.

Status Code: 200, 400.

3.4.3 POST

This operation MUST NOT be implemented.

Status Code: 405

3.4.4 DELETE

This operation SHOULD be implemented, but special precaution should be taken: if a DELETE is sent to the document URL, the document is completely deleted. Future requests to the section URL MAY return a status code of 410, unless the record is restored. Any DELETE operation MUST by logged by the underlying system.

Status Code: 204, 410

4 Bibliography

[1] G. Beuchelt, R. Dingwell, A. Gregorowicz, and H. Sleeper, "hData Record Format," The MITRE Corporation, 2009.

[2] PKWare, Inc. .ZIP Application Note. [Online]. http://www.pkware.com/support/zip-application-note