Beacon
DICOMweb Conformance Statement

Laurel Bridge Software, Inc.
302-453-0222
www.laurelbridge.com

Document Version: 1.5.0
Last Saved: 11/30/2022 2:39:00 PM

1 Overview

Beacon supports a subset of the RESTful web services specified in the DICOM PS3.18 - Web Services standard (commonly referred to as DICOMweb). Specifically, it supports the Store Transaction of the Studies Service and Resources (commonly referred to as STOW-RS, which stands for Store Over the Web – Restful Service) as an origin server. It does not support the Retrieve Transaction or the Search Transaction of the Studies Service.

Beacon does not support the URI Service, Worklist Service, Non-Patient Instance Service, or any of the Capabilities Transactions.

2 Store Transaction (STOW-RS)

The Store Transaction is a RESTful web service for storing DICOM imaging data.

The store transaction accepts either DICOM Part 10 binary files sent directly, or it accepts the splitting of the DICOM information into metadata (represented in JSON or XML) and bulk data sent separately.

2.1 DICOM Part 10 File HTTP Requests

The following Content-Types are supported:

- application/dicom
- multipart/related; type=application/dicom

Note that the Store Transaction accepts both a single part HTTP request (containing a single DICOM instance with content type application/dicom) or a multipart HTTP request (containing one or more DICOM instances, each with content type multipart/related; type=application/dicom).

Any valid SOP class will be accepted, and support for the following transfer syntaxes can be configured:

- 1.2.840.10008.1.2 (Implicit VR Little Endian)
- 1.2.840.10008.1.2.1 (Explicit VR Little Endian)
- 1.2.840.10008.1.2.2 (Explicit VR Big Endian – Retired)
- 1.2.840.10008.1.2.4.50 (JPEG Baseline, Process 1)
- 1.2.840.10008.1.2.4.51 (JPEG Extended, Processes 2 & 4)
- 1.2.840.10008.1.2.4.57 (JPEG Lossless, Non-Hierarchical, Process 14)
- 1.2.840.10008.1.2.4.70 (JPEG Lossless, Non-Hierarchical, First-Order Prediction, Process 14, Selection Value 1)
- 1.2.840.10008.1.2.4.80 (JPEG-LS Lossless)
- 1.2.840.10008.1.2.4.81 (JPEG-LS Lossy)
- 1.2.840.10008.1.2.4.90 (JPEG 2000 Lossless)

- 1.2.840.10008.1.2.4.91 (JPEG 2000)
- 1.2.840.10008.1.2.4.92 (JPEG 2000 Part 2, Multi-Component, Lossless Only) *
- 1.2.840.10008.1.2.4.93 (JPEG 2000 Part 2, Multi-Component) *
- 1.2.840.10008.1.2.4.100 (MPEG2 Main Profile, Main Level) *
- 1.2.840.10008.1.2.4.101 (MPEG2 Main Profile, High Level) *
- 1.2.840.10008.1.2.5 (RLE Lossless)

***** Indicates limited support with no transcoding to a different transfer syntax within Beacon.

No coercion or replacing of attributes is done by the server.

Only minimal validation of the DICOM file is done to extract some key DICOM elements, including the following:

- TransferSyntaxUID
- StudyInstanceUID
- SeriesInstanceUID
- SOPClassUID
- SOPInstanceUID

2.2 JSON Metadata with Bulk Data HTTP Requests

When storing instances using JSON metadata and bulk data, a multipart HTTP request must be used. The first multipart section (the DICOM metadata) must contain a JSON array of DICOM instances. The following Content-Type is supported for the DICOM metadata section:

- `multipart/related; type=application/dicom+json`

The DICOM metadata section can also have an optional transfer syntax encoded in the Content-Type: `multipart/related; type=application/dicom+json; transfer-syntax=`***`TransferSyntaxUID`***, where ***`TransferSyntaxUID`*** is the transfer syntax which is used to encode binary data in InlineBinary objects. If no InlineBinary objects are present in the metadata, the `transfer-syntax` parameter can be omitted.

No coercion or replacing of attributes is done by the server.

The following DICOM elements are required to be present in every instance in the JSON metadata:

- TransferSyntaxUID
- StudyInstanceUID
- SeriesInstanceUID
- SOPClassUID
- SOPInstanceUID
- The mandatory (type 1) Group 0028 (Image Pixel Module) elements

Note that the server does not derive Image Pixel Description Macro attributes, so they must be present in the metadata.

The JSON metadata must be encoded in Unicode UTF-8. The TransferSyntaxUID must be set to the following transfer syntax, if present:

- 1.2.840.10008.1.2.1 (Explicit VR Little Endian)

Within the JSON metadata, the user must specify BulkDataURIs (one per instance) for the Pixel Data elements. These BulkDataURIs indicate that the binary data for the Pixel Data element will be sent in a following part which has the Content-Location header set to the value of the BulkDataURI (which can be stored within the DICOM element as either a Value property or as a BulkDataURI property).

Note that each instance in the JSON metadata can have at most one BulkDataURI, and there must not be any duplicated BulkDataURIs in the JSON metadata. Compressed image bulk data is not currently supported.

The following Content-Type is supported for bulk data parts:

- multipart/related; type=application/octet-stream

Binary data other than the Pixel Data can be encoded as an InlineBinary base64-encoded string. When InlineBinary objects are used in the JSON metadata, they must be encoded in the Explicit VR Little Endian transfer syntax.

2.3 XML Metadata with Bulk Data HTTP Requests

When storing instances using XML metadata and bulk data, a multipart HTTP request must be used. The first multipart section(s) (the DICOM metadata) must contain one or more XML objects in Native DICOM Model format. The following Content-Type is supported for the DICOM metadata section(s):

- multipart/related; type=application/dicom+xml

The DICOM metadata sections can also have an optional transfer syntax encoded in the Content-Type: multipart/related; type=application/dicom+xml; transfer-syntax=***TransferSyntaxUID***, where ***TransferSyntaxUID*** is the transfer syntax which is used to encode binary data in InlineBinary objects. If no InlineBinary objects are present in the metadata, the transfer-syntax parameter can be omitted.

No coercion or replacing of attributes is done by the server.

The following DICOM elements are required to be present in every instance in the JSON metadata:

- TransferSyntaxUID
- StudyInstanceUID

- SeriesInstanceUID
- SOPClassUID
- SOPInstanceUID
- The mandatory (type 1) Group 0028 (Image Pixel Module) elements

Note that the server does not derive Image Pixel Description Macro attributes, so they must be present in the metadata.

The XML metadata must be encoded in Unicode UTF-8. The TransferSyntaxUID must be set to the following transfer syntax, if present:

- 1.2.840.10008.1.2.1 (Explicit VR Little Endian)

Within the XML metadata, the user must specify BulkDataURIs (one per instance) for the Pixel Data elements. These BulkDataURIs indicate that the binary data for the Pixel Data element will be sent in a following part which has the Content-Location header set to the value of the BulkDataURI (which can be stored within the DICOM element as either a Value element or as a URI attribute within a BulkData element).

Note that each instance in the XML metadata can have at most one BulkDataURI, and there must not be any duplicated BulkDataURIs in the XML metadata. Compressed image bulk data is not currently supported.

The following Content-Type is supported for bulk data parts:

- multipart/related; type=application/octet-stream

Binary data other than the Pixel Data can be encoded as an InlineBinary base64-encoded string. When InlineBinary objects are used in the XML metadata, they must be encoded in the Explicit VR Little Endian transfer syntax.

2.4 HTTP Responses

On success (all instances stored), Beacon will return an HTTP status code of 200 (= OK). The HTTP response body will contain a Store Instances Response Module data set (encoded in either JSON or XML), containing a list of the instances successfully stored (ReferencedSOPSequence).

On partial success (one or more instances failed to store, but not all), Beacon will return an HTTP status code of 202 (= Accepted). The HTTP response body will contain a Store Instances Response Module data set (encoded in either JSON or XML), containing a list of the instances successfully stored (ReferencedSOPSequence) and a list of the instances that failed to store (FailedSOPSequence).

On error (no instances were stored successfully), one of the following HTTP status codes will be returned:

- 400 = Bad Request (exception while parsing request)

- 404 = Not Found (if STOW-RS not enabled)
- 409 = Conflict (duplicate SOP instance(s))
- 415 = Unsupported Media Type
- 500 = Internal Server Error
- 503 = Service Unavailable (temporary problem)

In addition, the HTTP response body will contain a Store Instances Response Module data set (encoded in either JSON or XML), containing a list of DICOM failure codes (OtherFailuresSequence). The supported DICOM failure codes include:

- 0110_{hex} = Processing Failure
- 0111_{hex} = Duplicate SOP Instance
- $A700_{hex}$ = Refused – Out of Resources (temporary problem)
- $C122_{hex}$ = Referenced Transfer Syntax Not Supported

As mentioned above, the response data set can be encoded in either JSON or XML format, which can be controlled via the following Accept header values:

- `application/json` or `application/dicom+json` (JSON is used if the header is not present)
- `application/xml` or `application/dicom+xml`

If the Accept header is not present, the JSON format is used.

