# Contents

## 1 Overview
1.1 Processes ................................................................. 1
1.2 Activities .................................................................. 1

## 2 API
2.1 Introduction .................................................................. 3
2.2 Document Intake .......................................................... 3
   2.2.1 Document Intake: Fax Notify ................................. 4
   2.2.2 Document Intake: Generic ..................................... 7
2.3 Document Return .......................................................... 8
   2.3.1 Document Return: Delivery by HTTP PUSH .............. 8
   2.3.2 Document Return: Retrieval by HTTP PULL ............ 10

## 3 Getting Started
3.1 .NET Core - C# ............................................................ 19
3.2 Java ........................................................................ 19
3.3 JavaScript ................................................................. 20
3.4 Python ..................................................................... 20
The NEXTSTEP API empowers developers to leverage the power of the platform’s document processing capabilities from any application or system.

This is achieved by providing a simple API that enables documents to be submitted to the platform, processed, and then returned along with expanded, actionable metadata.

Let’s begin by establishing a basic understanding of NEXTSTEP processing and activities.

### 1.1 Processes

A NEXTSTEP Process defines a series of activities and configuration designed to prepare and transform an incoming document into the central artifact in a business process.

Processes are associated with documents during document intake, with selection of the appropriate process determined by inspecting the metadata received with the incoming document.

Once document intake is complete, process execution is managed by queueing the document for processing in each scheduled activity in sequence. Once the final activity is executed process execution is completed.

### 1.2 Activities

NEXTSTEP activities each provide a distinct function designed to inspect, modify or augment a document in some way. The set of available activities will continue to evolve with each release.
<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcode Recognition</td>
<td>Recognize and extract barcode values and metadata</td>
</tr>
<tr>
<td>Document Cleanup</td>
<td>Perform common document imaging actions to improve image quality</td>
</tr>
<tr>
<td>Document Conversion</td>
<td>Create alternate format representations of a document</td>
</tr>
<tr>
<td>Document Delivery</td>
<td>Delivers image and metadata files to a configured external location</td>
</tr>
<tr>
<td>Document Separation</td>
<td>Splits a document into multiple new documents</td>
</tr>
<tr>
<td>Page Rotation</td>
<td>Applies page rotation to pages that are oriented incorrectly</td>
</tr>
<tr>
<td>Metadata Processing</td>
<td>Maps metadata to custom fields in the Workflow application</td>
</tr>
<tr>
<td>Metadata Transform</td>
<td>Transforms metadata into a text file format such as XML or JSON</td>
</tr>
<tr>
<td>OCR</td>
<td>Recognizes text and related metadata in a document image</td>
</tr>
<tr>
<td>Signature Detection</td>
<td>Inspects a location on a page to determine if a signature is present</td>
</tr>
<tr>
<td>Value Extraction</td>
<td>Leverages AI to extract values from an unstructured document</td>
</tr>
</tbody>
</table>
2.1 Introduction

Integrating with the NEXTSTEP platform requires only a few simple API calls.

- **Document Intake**: Submitting new documents by making a request to the platform
- **Document Return**: Receiving or retrieving completed documents and metadata

The APIs that you leverage will depend on the nature and requirements of your integration.

All public APIs exposed by the NEXTSTEP use Representational State Transfer (REST) architecture. API documentation presented here is platform and language agnostic and described simply as its HTTP request or response components.

For tutorials demonstrating how to integrate with NEXTSTEP using a particular platform or programming language refer to *Getting Started*.

2.2 Document Intake

Document Intake is the process by which new documents are submitted to the platform. Intake supports submission of documents in the following formats:

- Tagged Image File Format (TIFF)
- Joint Photographic Experts Group Format (JPEG)
- Portable Network Graphics Format (PNG)
- Portable Document Format (PDF)

When submitting documents to the platform, callers will select an appropriate API endpoint based on the type of system that produced the incoming document, or that accepts the set of incoming metadata that best describes the document.
<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fax Notify</td>
<td>Accepts new documents with fax-centric metadata</td>
</tr>
<tr>
<td>Generic</td>
<td>Accepts new documents with generic key-value pair metadata</td>
</tr>
</tbody>
</table>

2.2.1 Document Intake: Fax Notify

Introduction

The fax notify intake endpoint accepts documents from fax systems that are capable of delivering to a REST API endpoint.

The endpoint accepts a series of named metadata values follows by a single file containing the incoming document.

Request Details

To submit a document to the platform via the fax notify endpoint, construct an HTTP POST request with the multipart form data content type, and the content sections described below:
### Table 1: Headers & Form Content Sections

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Sample Value &amp; Description</th>
</tr>
</thead>
</table>
| faxMessageId    | Content Section  | bbe58db7-b3f3-461a-9c47-5b803586cc53
A source message identifier that is generated by the originating system. (Optional) |
| messageType     | Content Section  | Inbound
Indicates the type of fax document being submitted. Must be one of: Inbound or Outbound |
| attachmentName  | Content Section  | bbe58db7-b3f3-461a-9c47-5b803586cc53.tif
Logical file name for the document being submitted. (Optional) |
| faxSenderCSID    | Content Section  | Test User (123-555-1212)
Fax CSID value for the sender (Optional) |
| callerNumber    | Content Section  | 1-123-555-1212
The fax number for the sender |
| calledNumber    | Content Section  | 1-123-555-6789
The fax number for the recipient |
| faxReceivedDateTime | Content Section | 10/07/2018 10:45:21 AM or 2018-10-07T10:45:21-06:00
The date/time when the fax was received by the source fax system. |
| faxRecipientTimeZone | Content Section | (GMT-06:00) Central Time (US & Canada)
The timezone corresponding to the fax received date/time value, in Windows format (Optional, when fax date/time is in ISO8601 format) |
| faxPages        | Content Section  | 1
The number of pages contained in the fax file |
| faxResolution   | Content Section  | 1
A number indicating the resolution of the fax file (Optional, source system specific) |
| faxSpeed        | Content Section  | 2
A number indicating the transmission speed of the fax (Optional, source system specific) |
| attachmentCount | Content Section  | 1
The number of attachment files processed by the fax system (Optional, source system specific) |
| account         | Content Section  | 12345
The NEXTSTEP user account number used for authentication
This will be provided by Concord Customer Success team |
| authUser        | Content Section  | 12345-6789-1212-12345-0001
The API key for the provided NEXTSTEP user account
This will be provided by Concord Customer Success team |
| file            | Content Section  | <binary file data>
The bytes representing the raw file content for the fax file |

### Example

An example HTTP POST request for the intake API (in raw text format) is provided below:

```
POST /api/intake/fax-notify HTTP/1.1
Host: https://nextstep.concord.net
Content-Type: multipart/form-data; boundary=-----WebKitFormBoundary7MA4YWxkTrZu0gW

```

(continues on next page)
Content-Disposition: form-data; name="FaxMessageId"
bbe58db7-b3f3-461a-9c47-5b803586cc53

Content-Disposition: form-data; name="MessageType"
Inbound

Content-Disposition: form-data; name="AttachmentName"
bbe58db7-b3f3-461a-9c47-5b803586cc53.tif

Content-Disposition: form-data; name="FaxSenderCSID"
Test User (123-555-1212)

Content-Disposition: form-data; name="CallerNumber"
1-123-555-1212

Content-Disposition: form-data; name="CalledNumber"
1-123-555-6789

Content-Disposition: form-data; name="FaxReceivedDateTime"
10/07/2018 10:45:21 AM

Content-Disposition: form-data; name="FaxRecipientTimeZone"
(GMT-06:00) Central Time (US & Canada)

Content-Disposition: form-data; name="FaxPages"
1

Content-Disposition: form-data; name="FaxResolution"
1

Content-Disposition: form-data; name="FaxSpeed"
2

Content-Disposition: form-data; name="AttachmentCount"
1

Content-Disposition: form-data; name="Account"
12345

Content-Disposition: form-data; name="AuthUser"
12345-6789-1212-12345-0001
2.2.2 Document Intake: Generic

Introduction

The generic intake endpoint accepts documents from any system that is capable of delivering to a REST API endpoint. The endpoint accepts metadata values as a set of simple key value pairs followed by a single file containing the incoming document.

Request Details

To submit a document to the platform via the generic endpoint, construct an HTTP POST request with the multipart form data content type, and the headers and content sections described below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Sample Value &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Auth-User</td>
<td>Header</td>
<td>12345&lt;br&gt;The NEXTSTEP user account number used for authentication&lt;br&gt;This will be provided by Concord Customer Success team</td>
</tr>
<tr>
<td>X-Auth-ApiKey</td>
<td>Header</td>
<td>12345-6789-1212-12345-0001&lt;br&gt;The API key for the provided NEXTSTEP user account&lt;br&gt;This will be provided by Concord Customer Success team</td>
</tr>
<tr>
<td>metadata</td>
<td>Content Section</td>
<td>See Example&lt;br&gt;The document metadata as a JSON object.</td>
</tr>
<tr>
<td>file</td>
<td>Content Section</td>
<td>&lt;binary file data&gt;&lt;br&gt;The bytes representing the raw file content for the fax file</td>
</tr>
</tbody>
</table>

Table 3: Metadata Definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Sample Value &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceId</td>
<td>bbe58db7-b3f3-461a-9c47-5b803586cc53&lt;br&gt;A source message identifier that is generated by the originating system. (Optional)</td>
</tr>
<tr>
<td>process</td>
<td>See Example&lt;br&gt;Identifies the process that should be associated with the new document. (Optional, however one of either the process or queue attributes must be provided)&lt;br&gt;These values will be provided by Concord Customer Success team</td>
</tr>
<tr>
<td>queue</td>
<td>See Example&lt;br&gt;Identifies the queue that the new document should be moved to following creation. (Optional, however one of either the process or queue attributes must be provided)&lt;br&gt;These values will be provided by Concord Customer Success team</td>
</tr>
<tr>
<td>properties</td>
<td>See Example&lt;br&gt;A set of name-value pairs that define additional metadata that should be stored with the document.</td>
</tr>
</tbody>
</table>
Example

An example HTTP POST request for the intake API (in raw text format) is provided below:

```plaintext
POST /api/intake HTTP/1.1
Host: https://nextstep.concord.net
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
X-Auth-User: 12345
X-Auth-ApiKey: 12345-6789-1212-12345-0001

Content-Disposition: form-data; name="metadata"
{
  "sourceId": "bbe58db7-b3f3-461a-9c47-5b803586cc53",
  "process": {
    "accountId": "1234567890",
    "processId": "1234567891"
  },
  "queue": {
    "accountId": "1234567890",
    "queueId": "1234567892"
  },
  "properties": {
    "name1": "value1",
    "name2": "value2",
    "name3": "value3"
  }
}

Content-Disposition: form-data; name="file"; filename="undefined"
Content-Type: file

<binary file data>

----WebKitFormBoundary7MA4YWxkTrZu0gW--
```

2.3 Document Return

Following completion of document processing the image and metadata files for the document are ready to be returned. Files are returned to the original calling application or system in one of the following ways:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery by HTTP PUSH</td>
<td>Receiving files by accepting a request from the platform</td>
</tr>
<tr>
<td>Retrieval by HTTP PULL</td>
<td>Retrieving files by making a request to the platform</td>
</tr>
</tbody>
</table>

2.3.1 Document Return: Delivery by HTTP PUSH

Introduction

The Document Delivery API enables integration with applications that are capable of hosting a Web API endpoint that can receive requests from the NEXTSTEP platform.

Each document will be delivered to the provided endpoint via an HTTP PUSH request that will contain the document image and metadata files.
The specific files that will be included in the delivery request are defined by the process configuration, which should be discovered, documented and implemented by the process owner or administrator prior to starting the API integration.

**Request Details**

Requests from the platform for document delivery will always be an HTTP POST with a multipart form data content type, containing the headers and content sections described below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Sample Value &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery URL</td>
<td>Configuration</td>
<td><code>https://api.your-domain.com/api/nextstep/delivery</code>&lt;br&gt;During discovery and configuration of your NEXTSTEP process, you will need to provide the host name where delivery requests should be sent.&lt;br&gt;The values above are for example purposes only</td>
</tr>
<tr>
<td>Authorization</td>
<td>Header</td>
<td><code>Bearer mF_9.B5f-4.IJqM</code>&lt;br&gt;The authorization header will be populated with the static bearer token provided to the platform during configuration.&lt;br&gt;(Optional)&lt;br&gt;Currently, only bearer token authentication is supported</td>
</tr>
<tr>
<td>&lt;File Name&gt;</td>
<td>Content Sections</td>
<td><code>&lt;binary file data&gt;</code>&lt;br&gt;Each configured delivery file will be added to the request in a separate content section. Each section will include the customer specified name to represent the file, as well as a logical file name, with file type extension, and content type header value.&lt;br&gt;The bytes representing the file content will be sent as the section body.&lt;br&gt;See <strong>Example</strong> below for a sample request that posts both an image file and metadata file to the delivery endpoint.</td>
</tr>
</tbody>
</table>

**Example**

This HTTP POST request for document delivery (in raw text format) uses a sample URL and path, and simulates delivery of a TIFF image and metadata file in JSON format. Other image and metadata formats are also supported. See **API** for additional image format details. Metadata files can be generated in either XML, or JSON format.

Sample Request:

```plaintext
POST /api/nextstep/delivery
Host: https://api.your-domain.com/
Content-Type: multipart/form-data; boundary=-----WebKitFormBoundaryWfPNVh4wuWBlyEyQ

-----WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="file"; filename="123456789.tif"
Content-Type: image/tiff

<binary file data>

-----WebKitFormBoundaryWfPNVh4wuWBlyEyQ
```

(continues on next page)
2.3.2 Document Return: Retrieval by HTTP PULL

Introduction

The Document Retrieval API enables applications to query the NEXTSTEP platform for document status. As the monitored documents are completed and published for retrieval the files associated with each document (image, metadata, searchable PDF, etc.) are then downloaded by as second call to the API. After downloading files for a particular document, the caller also has the option to disposition the document within the Workflow Application by transitioning that document to an archived or deleted state, or purging the document completely.

The specific files that will be included in the delivery request are defined by the process configuration, which should be discovered, documented and implemented by the process owner or administrator prior to starting the API integration.

Request Details

The lifecycle of requests for an application consuming the Document Retrieval APIs consists of two calls: List Documents and Retrieve Document.
List Document

The list document request is made each time the application needs to query for available documents. The set of returned documents will indicate the document processing or publishing status, as well as the queue and folder where the document is currently stored.

The set of documents returned will be dependent on the authorization permissions applied to the authenticated user calling the API.
<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Sample Value &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Request</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The HTTP method/verb to use when calling this API.</td>
</tr>
<tr>
<td>Request URL</td>
<td>Request</td>
<td><a href="https://nextstep.concord.net/api/retrieval">https://nextstep.concord.net/api/retrieval</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The retrieval endpoint used to list available documents.</td>
</tr>
<tr>
<td>X-AuthUser</td>
<td>Header</td>
<td>123456</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The User ID representing the user executing the list API call.</td>
</tr>
<tr>
<td>X-AuthApiKey</td>
<td>Header</td>
<td>ab123456-1234-1234-12ab-1ab23c4de567</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The API key corresponding to the X-AuthUser. Taken together the User ID and API Key are used to authenticate the user to the platform.</td>
</tr>
<tr>
<td>queueId</td>
<td>Query (Optional)</td>
<td>123456/abc12defg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Queue Identifier filter option enables filtering of the returned documents to those items that are assigned to the specified workflow queue. When omitted documents in any queue will be returned. <strong>The value here is a sample and should be replaced with a real value</strong></td>
</tr>
<tr>
<td>folderId</td>
<td>Query (Optional)</td>
<td>ab123456-1234-1234-12ab-1ab23c4de567</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Folder Identifier filter option enables filtering of the returned documents to those items that are assigned to the specified folder within a queue. When omitted documents in any folder will be returned. <strong>The value here is a sample and should be replaced with a real value</strong></td>
</tr>
<tr>
<td>retrievalStatus</td>
<td>Query (Optional)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Retrieval Status filter option enables filtering of the returned documents to those items that have the specified status assigned. When omitted documents with any retrieval status will be returned. <strong>Possible values include:</strong></td>
</tr>
</tbody>
</table>
|                  |                 | • Pending or 0: Return documents with pending status  
|                  |                 | • Completed or 1: Return documents with completed status |
| sort             | Query (Optional) | 0                          |
|                  |                 | The Sort option allows the caller to specify the sort direction of the returned documents. Documents are sorted by their Creation Date (**TODO: Needs confirmation). When omitted documents will be sorted in descending order by default. **Possible values include:** |
|                  |                 | • Descending, DESC, or 0: Sort in descending order  
|                  |                 | • Ascending, ASC, or 1: Sort in ascending order |
| take             | Query (Optional) | 50                         |
|                  |                 | The Take query option enables applications to implement paging of retrieved results using a skip/take model. The take option indicates the number of items to be returned, and the return data structure will indicate the total count of items available. |
| skip             | Query (Optional) | 100                        |
|                  |                 | The Skip query option enables applications to implement paging of retrieved results using a skip/take model. The skip option indicates the number of items to skip or bypass before selecting the item set to be returned, and the return data structure will indicate the total count of items available. |
Example

Below you will find an HTTP GET request and corresponding response for a sample document retrieval list operation (in raw text format). The request is querying for pending documents in a specific queue only, limited to 100 items returned, and not skipping any previously processed pages of items.

Sample Request:

```
GET /api/retrieval?queueId=5c76a7035b53411a149de3d2&retrievalStatus=Pending&take=100&skip=0 HTTP/1.1
Host: https://nextstep.concord.net/
X-AuthUser: 123456
X-AuthApiKey: ab123456-1234-1234-12ab-1ab23c4de567
```

Sample Response:

```
Content-Type: application/json

{
    "totalCount": 2,
    "documents": [
        {
            "id": "5c61181191acc818204667bb-7:5cecdee0a6cb577238656804",
            "queueId": "5c76a7035b53411a149de3d2",
            "folderId": "00000000-0000-0000-0000-000000000000",
            "files": [
                {
                    "id": "1bb0f308-7296-46a6-a8e2-f309856c8d47",
                    "format": "TIF",
                    "name": "document",
                    "fileName": "document.tif",
                    "version": 1,
                    "status": "Pending"
                },
                {
                    "id": "c9cfa81c-aid8-40ee-aa2e-2cbf1f43d5b1",
                    "format": "JSON",
                    "name": "metadata",
                    "fileName": "metadata.json",
                    "version": 0,
                    "status": "Pending"
                }
            ]
        },
        {
            "id": "5c61181191acc818204667bb-7:5cecdee0a6cb577238656804",
            "queueId": "5c76a7035b53411a149de3d2",
            "folderId": "00000000-0000-0000-0000-000000000000",
            "files": [
                {
                    "id": "1bb0f308-7296-46a6-a8e2-f309856c8d47",
                    "format": "TIF",
                    "name": "document",
                    "fileName": "document.tif",
                    "version": 1,
                    "status": "Pending"
                }
            ]
        }
    ]
}
```

(continues on next page)
### Table 6: **Response Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalCount</td>
<td>Response</td>
<td>The total number of documents matching the filter options.</td>
</tr>
<tr>
<td>documents</td>
<td>Response</td>
<td>A JSON array containing the documents satisfying the query.</td>
</tr>
<tr>
<td>id</td>
<td>Document Object</td>
<td>The unique identifier for the document. This value is used to retrieve the document and its associated files when calling the document retrieval endpoint. (See below)</td>
</tr>
<tr>
<td>queueId</td>
<td>Document Object</td>
<td>The unique identifier for the queue where the document is currently stored.</td>
</tr>
<tr>
<td>folderId</td>
<td>Document Object</td>
<td>The unique identifier for the folder within the queue where the document is currently stored. This value may be an empty identifier (all zeroes) when the document is stored in the queue root folder.</td>
</tr>
<tr>
<td>files</td>
<td>Document Object</td>
<td>A JSON array containing the files that have been published for retrieval for the current document.</td>
</tr>
<tr>
<td>id</td>
<td>File Object</td>
<td>The unique identifier for the document file.</td>
</tr>
<tr>
<td>format</td>
<td>File Object</td>
<td>The file format describing the file contents. Possible values include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TIF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PNG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JPG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• XML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JSON</td>
</tr>
<tr>
<td>name</td>
<td>File Object</td>
<td>The logical file name (without extension) defined for this file by the Workflow Process Owner. API developers should work with the process owner to define logical names for files that will aid in locating and consuming the appropriate content from the API response. Note: For file types that produce multiple files, such as JPEG output of a multi-page document image, the logical file name will indicate the root portion of the file’s complete name, with a page number and extension appended to form the complete file name. For example, on output of a 3 page TIFF document image, where the logical filename selected is document the files returned would be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• document_1.jpg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• document_2.jpg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• document_3.jpg</td>
</tr>
<tr>
<td>fileName</td>
<td>File Object</td>
<td>The complete file name (with any page number data and extension) defined for this file. This value is always an expansion of the logical name defined above.</td>
</tr>
<tr>
<td>version</td>
<td>File Object</td>
<td>Indicates the document image file version this file represents. Possible values include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: The original version of the document image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: The current version of the document image</td>
</tr>
<tr>
<td>status</td>
<td>File Object</td>
<td>The file retrieval status, which indicates whether the current file has been downloaded or not. Possible values include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Pending: The file has not been downloaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Complete: The file has been downloaded successfully</td>
</tr>
</tbody>
</table>

2.3. **Document Return**


Get Document

The get document request should be made for each document to retrieve the files that have been published for that document.

The set of files returned for each document will be dependent on the parameters passed to the API call.

Example

Below you will find an HTTP GET request and corresponding response for a sample document retrieval get operation (in raw text format). The request is querying for the files available for download for a specific document, specifying that the document disposition should be *Retain*, and opting not to request the output in a zip (archived) format.

The name and file name parameters are not specified, so all available files for the requested document will be returned.

Sample Request:

```
GET /api/retrieval/5c61181191acc818204667bb-7:5cecdee0a6cb577238656804?disposition=0&
    zip=false HTTP/1.1
Host: https://nextstep.concord.net/
X-AuthUser: 123456
X-AuthApiKey: ab123456-1234-1234-12ab-1ab23c4de567
```

Sample Response:

```
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryWfPNVh4wuWBlyEyQ

------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="document"; filename="document.tif"
Content-Type: image/tiff
<binary file data>

------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="metadata"; filename="metadata.json"
Content-Type: application/json

{
    "sourceId": "123456789",
    "fields": [
        {
            "name": "patient_name",
            "value": "John Smith"
        },
        {
            "name": "patient_dob",
            "value": "02/20/1964"
        },
        {
            "name": "date_of_encounter",
            "value": "01/30/2018"
        },
        {
            "name": "mrn",
            "value": "142637A"
    ]
}
```

(continues on next page)
The content of the metadata file (if configured) in the multi-part form will depend on what has been published by the process applied to the document.
All public APIs exposed by the NEXTSTEP use Representational State Transfer (REST) architecture. API documentation presented here is platform and language agnostic and described simply as its HTTP request or response components.

In this section, you will find tutorials demonstrating how to integrate with NEXTSTEP using various platforms and programming languages.

### 3.1 .NET Core - C#

![UNDER CONSTRUCTION]

### 3.2 Java

![UNDER CONSTRUCTION]
3.3 JavaScript

![UNDER CONSTRUCTION]

3.4 Python

![UNDER CONSTRUCTION]