KonaSearch Data Upload API Specification

Spec version: 1
Last modified: Oct. 8, 2019

Summary

The sections below outline the KonaSearch Data Upload API and focuses on the format and structure of the RESTful requests and responses.

The data format of the upload data files is documented in the next section.

Making RESTful Calls to Upload Data into the Kona System

The RESTful interface of the API is described in the sections below.

You must have two pieces of data in order to make these RESTful calls, in terms of authorization:

- The Salesforce org ID.
- The Kona API key obtained from your Kona account (business) contact.

Note that all data, including both the upload request parameters and the upload data itself, is encrypted via SSL/HTTPS.

The RESTful Interface for Uploads

The request to upload data using the Data Upload API is structured as follows.

The Data Upload Client Utility (see later in document) takes care of most of the complexity of putting the requests together and executing them.

Request type	POST
Request URL example	`https://gcpdata.konadsc.com/konauploader/dataupload?api.key=<...>&org.id=<...>`
Request parameters	• **api.key** - mandatory; must hold a valid Kona Data Upload API key. This is the key the caller is required to receive from their Kona account (business) contact. • **org.id** - mandatory; must hold a valid Kona org ID The API key must be valid, non-expired, non-disabled key issued for the org whose ID is passed in.

Request body	• The request body is expected to be:
	  ○ either upload JSON as defined in the Kona Data Upload JSON schema
	  ○ or upload XML as defined in the Kona Data Upload XML schema
	• The data for upload requests is compressed using the GZIP algorithm. Set the HTTP "**Accept-Encoding**" header to the value "gzip".

Response on success	• Upon a successful upload, the HTTP code 200 is returned with a short message acknowledging that the data has been received.
	`<response>`
	`  <status>OK</status>`
	`  <message>Received uploaded file. Org ID: 00DE0000000MdP7FBL, obj. type: uploads, saved to bucket: uploads, location: /kona/2019/07/11/13/00DE0000000MdP7FBL_uploads_4674cbb6-12d7-4a71-b0f1-bf4ed0a6fee1.dat.</message>`
	`</response>`

Response on error	The HTTP code of 400 is returned if a request error has been detected. An error message is included in the response.
	
	`<response>`
	`  <status>FAIL</status>`
	`  <message>Missing request parameter 'org.id'</message>`
	`</response>`
	
	The HTTP code of 500 is return if an internal server error has occurred. An error message is included in the response.
	
	`<response>`
	`  <status>FAIL</status>`
	`  <message>Error while persisting an upload workqueue entry. Filename: 00DE0000000MdP7FBL_uploads_4674cbb6-12d7-4a71-b0f1-bf4ed0a6fee1.dat</message>`
	`</response>`

Using the Kona Data Upload Client Utility

The Kona Data Upload Client utility is a command-line tool which allows you to point it at an input directory and have your JSON or XML data upload files uploaded into the Kona system. See later in document for more information on this utility. The distro for the utility also contains the source code which can be used as a reference if you choose to develop your own uploader utility.

KonaSearch Data Upload Format Specification

Spec version: 1
Last modified: Oct. 8, 2019

Summary

This page summarizes the format of the data uploaded into Kona using the Kona Data Upload API.

The Data Upload Format

The uploaded data must be specified **either as XML or as JSON**. The sections below describe the structure of the upload files, with examples. These sections also outline how object types, ID's, and content and metadata fields are specified.

General Data Structure

- The general data structure of the upload JSON is that an upload JSON document consists of an "objects" element of type JSON array, which contains one or more objects.
- Each object must have an **ID** property which must be a string of ASCII characters, no longer than 300 characters in length. See the section 'Specifying the Object ID's', for details.
- Each object must contain the **required metadata** properties, as specified in the section 'The Required Metadata Properties of an Object'.
- Each object may have zero or more of the non-required '**core**' metadata properties. See the section 'Handling of the Core Metadata Properties of an Object'.
- Each object may have zero or more of the '**custom**' metadata properties. See the section 'Handling of the Custom Metadata Properties of an Object'.
- An object may or may not have a '**content**', set as a value of the content field. See the section 'Handling of the Content Property of an Object'.

The Relevant Schemas (JSON and XML)

- The upload JSON must adhere to the Kona Data Upload JSON Schema Specification (later in document, including examples)
- The upload XMLmust adhere to the Kona Data Upload XML Schema Specification (later in document, including examples)

The Required Metadata Properties of an Object

The following properties are **required** on every upload object:

- **content_source**
- **id_id**
- **object_type**

See the section 'Handling the Core Metadata Properties of an Object' for more details on these.

Specifying the Original Source of the Content ("content_source")

- Each object type must have its original content source specified via the "content_source" field.
- The type must be a string of UTF-8 characters as follows:
  - It is recommended that the string be no longer than 128 characters.
  - The value is case-sensitive.
- Care must be taken to make sure the value of this property is unique per ***instance*** of a given source type.
  - For example, extracting and uploading data from 2 different SharePoint instances warrants two content_source values. For instance, "acme_sharepoint_nam" and "acme_sharepoint_apac" (for Acme's SharePoint installations in North America vs. Asia Pacific).
  - This is not a concern for cases where the instance guaranteed to be unique.

Specifying the Object ID's ("id_id")

- Each object within a Kona upload JSON must have the "**id_id**" field.
  - This value must be UTF-8 characters.
  - The value is case-sensitive.
  - This value may be no longer than 32K (Solr's limit) however, using much shorter values is recommended.

- This ID is expected to uniquely identify the given object within the data source it came from.
- This value may not change from one data upload to another.
- **Upon uploading, the Data Upload API adjusts the value of the "id_id" property by prepending the content_source value to it. This is done so as to ensure global uniqueness of the id_id value once it's indexed into the Kona search engine.**
    - **The "id_id" value is used in the Kona engine internally and must be made unique when indexed.**

Specifying the Object Type ("object_type")

- Each upload object must have a type specified.
- The type is specified using the mandatory "**object_type**" property on the object.
- The type must be a string of UTF-8 characters as follows:
    - It is recommended that the string be no longer than 128 characters.
- In the Kona search engine, the type is indexed into the "**table**" field.

The Generation of the Unique Document ID's for Upload Objects

The Upload API automatically generates a universally unique ID for each uploaded object for proper handling at ingest time and query time in the Kona search index.

This ID is generated using the following format:

```
<content source value>_<object type value>_<the digest of the id_id value>
```

where:

1. **Content source value**: the (sanitized) value of the "content_source" property.
2. **Object type value**: the (sanitized) value of the "object_type" property.
3. **id_id value digest**: the MD5 digest of the id_id value (32 characters long).

When the values are sanitized, it means that any non-word characters and underscores are removed. Word characters are defined as the regular expression a-zA-Z_0-9. In addition, the value is converted to all lowercase characters.

For example:

content_source	object_type	id_id	Generated unique ID	Notes
"Acme_Email_Server"	"emailmessage"	"48040dc7-0a4f-44ca-8057-5ca29cd2b9e7"	"acmeemailserver_emailmessage_97a8ce2e5521cf2d49a32a8c90951b8a"	
"abc_def \"':.!/"	"acme_report \"':.!/"	"https://www.acmeinc.com/reports/q3_report.pdf"	"abcdef_acmereport_a4f0ee93fe494e7ac75e75750134b5c2"	This demonstrates the handling of special characters in properties such as object_type and content_source. Generally, the recommendation is to use ASCII only for these values.

Handling of the Core Metadata Properties of an Object

The table below outlines the 'Core' metadata properties that may be defined on objects uploaded into Kona.

Note that this list includes the 'Required' metadata properties.

Note also that some of these property names may clash with internal Kona field names, names of metadata fields extracted by Tika during the Text Extraction pipeline stage.

In such a case the strategy for handling of a naming conflict of these properties is as follows:

1. Append any values extracted by Tika to the field, if the field is multi-valued.
2. If the field is single-valued, take the value provided in the upload over the value extracted by Tika (i.e. drop the value(s) extracted by Tika for the field).

Property Name	JSON Type	Required?	Muli-valued?	Description	Grouping	Example	Dev Notes
author	array of string	No	Yes	The author of the given object. Despite the singular property name, this property may be multi-valued.		`"jsmith1234@acmeinc.com"`	
bcc	array of string	No	Yes	The list of the "BCC" recipients, typically on an email.	email related properties	`"abarnes2343@gmail.com", "samantha.obrien4456456@yahoo.com"`	
bytes	array of string	No	Yes	The Base 64-encoded raw content bytes.		`"0M8R4KGxGuEAA....=="`	
cbpid_reference	string	No	No	The ID ("id_id" value) of a Salesforce object with which the given upload object is associated.			This value determines the visibility of the object in the Solr index.
cc	array of string	No	Yes	The list of the "CC" recipients, typically on an email.	email related properties	`"mbennett0404@gmail.com", "stucker34534@gmail.com"`	

child_objects	object array	No	Yes	Nested / child objects such as attachments are placed under the "child_objects" object array element.			In the Solr index, child documents will be associated with their parents via the "parentid_reference" field value.
comments	array of string	No	Yes	The list of comments, if any.		`"Note: work in progress, review needed"`	
company	array of string	No	Yes	The company name, if any.		`"Acme Financial Services Inc."`	
content_source	string	**YES**	No	The name of the original data source that the object came from.		`"Acme_Email_Server"`	
createddate_datetime	string	No	No	The creation date of the object.		`"2019-06-17T17:44:55Z"`	This must be a full timestamp (ISO8601).
description	string	No	No	The description of the object, if any.		`"This Excel spreadsheet contains our projections for the 3rd quarter."`	

filename	string	No	No	The filename, if any.	file system related	`"PotentialInvestors.docx"`	
from	array of string	No	Yes	The list of senders, typically on an email.	email related properties	`"jsmith1234@acmeinc.com"`	
id_id	string	**YES**	No	The ID of the object in its original data source.		`"075de4fdf884a905fb28a3ea21faa844"`	Must be unique within the original data source. Made universally unique in Kona search indexes (augmented with other field values).
is_attachment_boolean	boolean	No	No	If true, indicates that the object is an attachment.		`true`	
is_deleted_boolean	boolean	No	No	If true, indicates that the object has been deleted in its original data source.		`true`	
keywords	array of string	No	Yes	The list of any keywords set on the object.		`"analytics", "projections", "Q3"`	

lastmodifieddate_datetime	string	No	No	The last modified date of the object.		`"2019-06-17T17:44:55Z"`	This is a date, preferably a full timestamp (ISO8601). TODO: consider adding lastmodifieddate_datetime to solr schema
lastmodifier	string	No	No	The last modifier of the object.		`"jsmith1234@acmeinc.com"`	
latitude	number (double)	No	No	The latitude part of geo search fields (latitude / longitude)	geo search properties	`15.4041`	The valid range of latitude in degrees is **-90 and +90** for the Southern and Northern hemisphere respectively.
links	array of string	No	Yes	The list of any links.		`"https://www.acmefinserv.com/guidelines.html", "https://www.acmefinserv.com/governance.html"`	
location	array of string	No	Yes	The location, if any.		`"699 Eagles Nest Drive, Chico CA, 95926"`	
longitude	number (double)	No	No	The longitude part of geo search fields (latitude / longitude)	geo search properties	`30.1675`	Longitude values are in the range of -180 **and** +180 specifying coordinates West **and** East of the Prime Meridian, respectively.

object_type	string	**YES**	No	The type of the object.		`"emailmessage"`	
parentid_reference	string	**YES** for child objects	No	Internally set property which associates a child object such as an attachment with its respective parent object.			This property is set internally by the Data Upload API. The value is set to the "id_id" value of the parent. Note that the "id_id" at that point is prepended with the "content_source" value which ensures uniqueness of the "id_id" in the index (with content from possibly multiple sources).
size	number (long)	No	No	The size of the object, if available.	typically file system related	`150603`	
source_id	string	No	No	The unique ID of the object in its original data source.		`"075de4fdf884a905fb28a3ea21faa844"`	
sourcepath	string	No	No	The path to the object in the original data source; typically a file path, for file system types of data sources.	typically file system related	`"/mnt/acme/files/investors/2019/PotentialInvestors.docx"`	

subject	array of string	No	Yes	The subject value(s), typically on an email.	email related properties	"Potential investors"	If a given upload object has no "title" set but has "subject" set then the value of "subject" is copied into "title".
tags	array of string	No	Yes	The list of tags on the object, if any.		"reviewed", "projections"	
teaser	string	No	False	The teaser for the object.		"Hi there. Looking to raise about $30mm from the attached..."	
text	array of string	No	Yes	The textual content of the object.		"Hi there. Looking to raise about $30mm from the attached list of names. Let me know if you can use any of these. Thanks, John"	
title	array of string	No	Yes	The title(s) of the object.		"Financial forecast for Q3"	
to	array of string	No	Yes	The list of the "TO" recipients, typically on an email.	email related properties	"er92950@acmeinc.com", "rk67890@acmeinc.com"	
uri	string	No	No	The URI of the object; typically this is a universal		"file:/mnt/acme/files/investors/2019/PotentialInvestors.docx"	

				resource identifier pointing back at the location where the object resides in its original data source.			

Handling of the Custom Metadata Properties of an Object

Any object properties that are not a part of the 'standard' set must adhere to the below specifications.

Custom Property Name Validation

All custom property names must adhere to the below specification:

- Custom property names should consist of alphanumeric or underscore characters only and not start with a digit.
- The length of a custom field name may not exceed 128 characters.

This is based on the Solr doc (https://lucene.apache.org/solr/guide/6_6/defining-fields.html):

Field names should consist of alphanumeric or underscore characters only and not start with a digit.

This is not currently strictly enforced, but other field names will not have first class support from all components and back compatibility is not guaranteed.

Names with both leading and trailing underscores (e.g., _version_) are reserved.

Custom Property Names as Related to Search Index Data Types

Any custom metadata properties may be specified as follows.

Note that in the Kona search index schema, these fields will be treated as dynamic fields. See the respective type definitions in the column 'Solr type definition".

Data Type	Custom Property Name	JSON Type	Example of Handling in JSON	Notes	Solr Type Definition
string	must end in postfix "_string"	string	"order_notes_string": "This order has been completed."	Currently no support for multi-valued (single valued only). See "multipicklist" for how multiple values can be handled.	`<dynamicField name="*_string" type="text_general" indexed="true" stored="true" omitNorms="true" multiValued="true" />`
ID	must end in postfix "_id"	string	"custid_id": "1234567"		`<dynamicField name="*_id" type="string" indexed="true" stored="true" omitNorms="true" docValues="true" />`
email address	must end in postfix "_email"	string	"messagesender_email": "jsmith@acmefin.com"		`<dynamicField name="*_email" type="string" indexed="true" stored="true" omitNorms="true" docValues="true" />`
phone number	must end in postfix "_phone"	string	"customer_phone": "(617) 245-9566"		`<dynamicField name="*_phone" type="string" indexed="true" stored="true" omitNorms="true" docValues="true" />`

reference	must end in postfix "_reference"	string	"createdbyid_reference": "0050000000740QDAAY"		<dynamicField name="*_reference" type="string" indexed="true" stored="true" omitNorms="true" docValues="true" />
URL	must end in postfix "_url"	string	"website_url": "www.acmeinc.com/contact_us.html"		<dynamicField name="*_url" type="string" indexed="true" stored="true" omitNorms="true" docValues="true" />
boolean	must end in postfix "_boolean"	boolean	"is_reviewed_boolean": true		<dynamicField name="*_boolean" type="boolean" indexed="true" stored="true" docValues="true" />
integer	must end in postfix "_integer"	integer	"total_amt_integer": 1000		<dynamicField name="*_integer" type="int" indexed="true" stored="true" docValues="true" />
double	must end in postfix "_double"	number	"total_reading_double": 234234.33		<dynamicField name="*_double" type="double" indexed="true"

					`stored="true" docValues="true" />`
float	must end in postfix "_float"	number	"some_value_float": 3423.3453		`<dynamicField name="*_f" type="float" indexed="true" stored="true" docValues="true" />`
currency	must end in postfix "_currency"	number	"order_amount_currency": 22.99		`<dynamicField name="*_currency" type="double" indexed="true" stored="true" docValues="true" />`
date	must end in postfix "_date"	string	"review_date": "2019-06-14Z"	Value must be ISO 8601, **UTC**, in the format: yyyy-MM-ddZ, where • yyyy is the year; • MM is the month; • dd is the day of the month; • Z is a literal 'Z' character indicating that this string representation of the date is in UTC. Value may also be yyyy-MM-dd'T'HH:mm:ssZ (however, value will be treated as Solr 'date').	`<dynamicField name="*_date" type="date" indexed="true" stored="true" docValues="true" />`

timestamp	must end in postfix "_datetime"	string	"published_datetime": "2019-06-17T17:44:50Z"	Value must be ISO 8691, **UTC**, in the format: yyyy-MM-dd'T'HH:mm:ssZ, where • yyyy is the year; • MM is the month; • dd is the day of the month; • HH is the hour of the day as on a 24-hour clock; • mm is minutes; • ss is seconds; • Z is a literal 'Z' character indicating that this string representation of the date is in UTC.	`<dynamicField name="*_datetime" type="date" indexed="true" stored="true" docValues="true" />`
binary	must end in postfix "_bytes"	Base64-encoded string or array of such strings			`<dynamicField name="*_bytes" type="binary" indexed="false" stored="true" multiValued="true"/>`
long	must end in postfix "_long"	number	"item_fingerprint_long": 1242342345		`<dynamicField name="*_l" type="long" indexed="true" stored="true" docValues="true" />`

multipicklist	must end in postfix "_multipicklist"	string	"abc_multipicklist": "aaa; bbb; ccc" "def_multipicklist": "ddd"	The joined, concatenated value is parsed into multiple individual values. The semicolon character is used as the delimiter. In the example of "aaa; bbb; ccc" the field is indexed with 3 distinct values: "aaa", "bbb", and "ccc".	`<dynamicField name="*_multipicklist" type="text_general" indexed="true" stored="true" omitNorms="true" multiValued="true" />`
anything else		any supported type	• "customer_contactinfo": "761 Garrett St., Grand Rapids MI 49503" • "ordernumber": 234	Handled as a **string** in Kona search engine.	`<dynamicField name="*" type="string" indexed="false" stored="true" docValues="true" />`

Custom Property Value Validation

- **String values:** String values in the upload data JSON may not be longer than 5 MB.
- **Binary values:** may not be longer than 50 MB prior to being Base 64-encoded.

Handling of the Content Property of an Object

Content may be specified via the "**bytes**" field which may contain the Base64-encoded content of the object, and may be multi-valued.

Note that content on a given object may be specified as textual and/or 'binary':

1. An object may have just **textual** content in which case the textual content should be placed into the "**text**" property.
2. An object may have just '**binary**' content in which case the binary content must be Base-64 encoded and placed into the "**bytes**" property. In this case:

1. The contents of the "bytes" field will be processed via the Text Extraction stage by Kona. The results will be set as the value of the "text" property. The raw bytes are discarded prior to indexing into the search engine.

3. An object may have **both textual and 'binary'** content. In such a case:
    1. The textual content should be placed into the "text" property.
    2. The binary content must be Base-64 encoded and placed into the "bytes" property.
    3. The contents of the "bytes" field will be processed via the Text Extraction stage by Kona. The results will be added as a separate value to the "text" property. The raw bytes are discarded prior to indexing into the search engine.

Handling of Attachments

- Attachment objects are expected to accompany their respective parents, as nested documents within the upload JSON i.e. as elements of the "**child_objects**" property.
- Each such object must have its "**is_attachment**" Boolean property set to true.
- See section 'Handling of the Content Property of an Object' for details of how the raw content bytes are handled.

Handling of Large Content and Attachments

Version 1 of the specification dictates that the largest size (prior to encoding as Base 64) of an upload object's content or attachment's content may not exceed 50 megabytes.

This is currently not strictly enforced, but adherence to this limit is strongly recommended.

It is expected that each upload data file (JSON or XML) will have a reasonable size (e.g. no more than a few larger attachments per file, given that a pre-encoded attachment can only be up to 50 MB in size).

Handling of Nested Objects

- Objects may be nested under parent objects.
- Once indexed, each such object is linked to its parent via the "**parentreference_id**" field in the search index.
- Multiple levels of nestedness are allowed, i.e. child objects may have their own child objects, those also may have children, and so on.

Handling of Deleted Objects

- Deletions are propagated into the Kona system by performing a data import, with respective objects annotated as deleted, using the "**is_deleted_boolean**" field.
- Its value must be set to Boolean true in order to mark an object as deleted.
- If the object marked for deletion is not found, its data in the imported JSON is ignored.

Upload Data Encryption

- Data uploaded into the Kona system is encrypted on the client side. A Kona encryption key for the given Kona organization is required in order for uploads to work.

Upload Data Compression

- At data upload time, the data may optionally be compressed (using GZip).
- The Phase 1 implementation of the Kona Upload API client will have the Gzip compression of uploaded content turned ON by default.

Authorization (Content Visibility)

- The ingestion of access control information from 3rd party sources is out of scope in version 1 of this specification.
- By default, any documents indexed into the Kona search engine, are considered inaccessible, unless explicitly made accessible.
- For Phase 1, access is granted if the following property values are set on the indexed upload objects:
    - Set the cbpid_reference property on the object (or child object / attachment) to cause it to be associated with a particular Salesforce object such as an Account.
        - Users with access to the Account will automatically gain access to the uploaded objects associated with the Account (via the cbpid_reference).

- At upload time, all uploaded objects are automatically "stamped" with the value of the user ID for the User associated with the API key of the given org.
  - This User ID is selected by the org as the ID for access to the uploaded data. The API key handed to the org is created based on the org ID and the User ID.
  - The Upload API automatically stamps all upload objects with a field called **ownerid_reference** whose value is set to the User ID (of the User associated with the API key).
  - This ensures that at query time, search results are properly filtered.
  - With this model, any custom portal or application making Kona Search API requests needs to provide this User ID at query time regardless of what users are accessing the data. In other words, all of their requests go through this dedicated User ID.

The KonaSearch Data Upload Client Utility

Summary

The KonaSearch Data Upload Client Utility is a command-line utility which allows users to upload their data into the Kona system via the RESTful Kona Data Upload API.

The input upload data is expected to be either XML or JSON, adhering to the KonaSearch Upload API specification described in the first section of this document. More information about making RESTful calls into the Kona Data Upload API is available in this section. This section outlines how to use the out-of-the-box Kona upload command line utility.

Installation

To install the utility, simply uncompress the provided **kona-upload-client tar.gz** archive file to the target directory of your choosing.

The provided Upload Client distro includes a source jar with Java classes which serve as a reference implementation of a client-side upload library.

Uploading Your Data

On Linux/UNIX:

User can kick off a data upload by running the **kona_upload_json.sh** or the **kona_upload_xml.sh** shell script, with the required arguments filled in.

On Windows:

User can kick off a data upload by running the **kona_upload_json.cmd** or the **kona_upload_xml.cmd** shell script, with the required arguments filled in.

The available command-line arguments are as follows:

Option	Description
-?, -h, -help	Prints the help summary.
-apikey <apiKey>	[Required] The Kona API key. (Contact your Kona representative to receive this key).
-orgid <orgId>	[Required] The Kona organization ID.
-rootdir <inputRootDirectory>	[Required] The root directory for locating the input data files.
-serveruri <uploadServerUri>	[Required] The Kona upload server URI, similar to: https://gcpdata.konadsc.com:443.

The additional options available are as follows:

Option	Description
-dryrun	If present, the dry run option causes the utility to iterate over the files and count them without doing any uploading. Additionally, if the validate option is present, any errors from validating the files against the schema are reported.
-validate	If the validate option is present, the input data files are validated againt the appropriate JSON or XML schema.
-failfast	If the validate option is present and failfast is present, then upon the first validation error the uploading is halted. If failfast is not present, the 'offending' file is skipped over and processing continues. This option applies to both the dry run scenario and regular upload scenario.

Notes

- For reference, the conf/ subdirectory of the installation contains both the XML and the JSON Kona upload file format schemas.

Kona Data Upload JSON Schema

Spec version: 1
Last modified: Oct. 8, 2019

Summary

Below is the summary of the JSON schema for the Kona data upload JSON files.

The general JSON schema documentation from json-schema.org can be found here.

The Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "The KonaDSC data upload JSON schema, version 1.",
  "definitions": {
    "custom_property": {
      "id": "#custom_property",
      "type": "object",
      "title": "A custom metadata property",
      "required": [ "n", "v" ],
      "properties": {
        "n": {
          "id": "#custom_property/items/properties/n",
          "type": "string",
          "title": "The name of a custom property, e.g. order_id.",
          "pattern": "^(.*)$"
        },
        "t": {
          "id": "#custom_property/items/properties/t",
          "type": "string",
```

```
        "enum": [
          "boolean",
          "binary",
          "date",
          "double",
          "float",
          "int",
          "long",
          "string"
        ],
        "title": "The type of a custom property. If omitted, then string is assumed.",
        "pattern": "^(.*)$"
      },
      "v": {
        "id": "#custom_property/items/properties/v",
        "type": "string",
        "title": "The value of a custom property, e.g. \"200.2343\", \"34534543536\".",
        "pattern": "^(.*)$"
      }
    },
    "additionalProperties": false
  },
  "obj": {
    "id": "#obj",
    "type": "object",
    "title": "The object schema",
    "required": [
      "content_source",
      "id_id",
      "object_type"
    ],
    "metadata": {
      "id": "#obj/items/properties/metadata",
      "type": "array",
      "title": "The custom metadata properties of the given object.",
      "items": {
        "$ref": "#/definitions/custom_property"
```

```
        }
      },
      "properties": {
        "author": {
          "id": "#obj/items/properties/author",
          "type": "array",
          "title": "  The author(s) of the given object.",
          "items": {
            "id": "#obj/items/properties/author/items",
            "type": "string",
            "title": "The author items schema. Sample values: \"jsmith1234@acmeinc.com\", \"John Smith\".",
            "pattern": "^(.*)$"
          }
        },
        "bcc": {
          "id": "#obj/items/properties/bcc",
          "type": "array",
          "title": "The list of the BCC recipients, typically on an email.",
          "items": {
            "id": "#obj/items/properties/bcc/items",
            "type": "string",
            "title": "The bcc items schema. Sample values: \"abarnes2343@gmail.com\",
\"samantha.obrien4456456@yahoo.com\".",
            "pattern": "^(.*)$"
          }
        },
        "bytes": {
          "id": "#obj/items/properties/bytes",
          "type": "array",
          "title": "The list of content byte arrays, with each value Base 64 encoded.",
          "items": {
            "id": "#obj/items/properties/bytes/items",
            "type": "string",
            "title": "The bytes items schema. Sample value: \"0M8R4KGxGuEAA....==\".",
            "pattern": "^(.*)$"
          }
        },
```

```
        "cc": {
          "id": "#obj/items/properties/cc",
          "type": "array",
          "title": "The list of the BCC recipients, typically on an email.",
          "items": {
            "id": "#obj/items/properties/cc/items",
            "type": "string",
            "title": "The cc items schema. Sample values: \"mbennett0404@gmail.com\",
\"stucker34534@gmail.com\".",
            "pattern": "^(.*)$"
          }
        },
        "child_objects": {
          "id": "#obj/items/properties/child_objects",
          "type": "array",
          "title": "The list of child objects, if any",
          "items": {
            "$ref": "#/definitions/obj"
          }
        },
        "comments": {
          "id": "#obj/items/properties/comments",
          "type": "array",
          "title": "The list of comments on the object, if any.",
          "items": {
            "id": "#obj/items/properties/comments/items",
            "type": "string",
            "title": "The comments items schema. Sample value: \"Note: work in progress, review needed\".",
            "pattern": "^(.*)$"
          }
        },
        "company": {
          "id": "#obj/items/properties/company",
          "type": "array",
          "title": "The company name(s) on the object, if any.",
          "items": {
            "id": "#obj/items/properties/company/items",
```

```
            "type": "string",
            "title": "The company items schema. Sample value: \"Acme Financial Services Inc.\".",
            "pattern": "^(.*)$"
          }
        },
        "content_source": {
          "id": "#obj/items/properties/content_source",
          "type": "string",
          "title": "The name of the original data source that the object came from. Sample value:
\"AcmeEmailServer\".",
          "pattern": "^(.*)$"
        },
        "createddate_datetime": {
          "id": "#obj/items/properties/createddate_datetime",
          "type": "string",
          "format": "date-time",
          "title": "The creation date of the object. Sample value: \"2019-06-17T17:44:55Z\".",
          "pattern": "^(.*)$"
        },
        "description": {
          "id": "#obj/items/properties/description",
          "type": "string",
          "title": "The description of the object, if any. Sample value: \"This Excel spreadsheet contains our
projections for the 3rd quarter.\".",
          "pattern": "^(.*)$"
        },
        "filename": {
          "id": "#obj/items/properties/filename",
          "type": "string",
          "title": "The filename, if any. Sample value: \"PotentialInvestors.docx\".",
          "pattern": "^(.*)$"
        },
        "from": {
          "id": "#obj/items/properties/from",
          "type": "array",
          "title": "The sender(s), typically on an email.",
          "items": {
```

```
            "id": "#obj/items/properties/from/items",
            "type": "string",
            "title": "The from items schema. Sample value: \"jsmith1234@acmeinc.com\".",
            "pattern": "^(.*)$"
          }
        },
        "id_id": {
          "id": "#obj/items/properties/id_id",
          "type": "string",
          "title": "The ID of the object. This is the unique ID of the object in its original data source. This
value is expected to be the same as that of the source_id property. Upon uploading, the id_id value, however, is
prepended with the content_source value so as to make it globally unique across documents from potentially
multiple original sources. Sample value: \"075de4fdf884a905fb28a3ea21faa844\".",
          "pattern": "^(.*)$"
        },
        "is_attachment_boolean": {
          "id": "#obj/items/properties/is_attachment_boolean",
          "type": "boolean",
          "title": "If true, indicates that the object is an attachment. Sample value: \"true\"."
        },
        "is_deleted_boolean": {
          "id": "#obj/items/properties/is_deleted_boolean",
          "type": "boolean",
          "title": "If true, indicates that the object has been deleted in its original data source. Sample
value: \"true\"."
        },
        "job_id": {
          "id": "#obj/items/properties/job_id",
          "type": "string",
          "title": "The job ID of the data upload. Sample value: \"job-123\".",
          "pattern": "^(.*)$"
        },
        "keywords": {
          "id": "#obj/items/properties/keywords",
          "type": "array",
          "title": "The list of any keywords set on the object, if any.",
          "items": {
```

```
            "id": "#obj/items/properties/keywords/items",
            "type": "string",
            "title": "The keywords items schema. Sample values: \"analytics\", \"projections\", \"Q3\".",
            "pattern": "^(.*)$"
          }
        },
        "lastmodifieddate_datetime": {
          "id": "#obj/items/properties/lastmodifieddate_datetime",
          "type": "string",
          "format": "date-time",
          "title": "The last modified date of the object. Sample value: \"2019-06-17T17:44:55Z\".",
          "pattern": "^(.*)$"
        },
        "lastmodifier": {
          "id": "#obj/items/properties/lastmodifier",
          "type": "string",
          "title": "The last modifier of the object. Sample value: \"jsmith1234@acmeinc.com\".",
          "pattern": "^(.*)$"
        },
        "latitude": {
          "id": "#obj/items/properties/latitude",
          "type": "number",
          "minimum": -90,
          "maximum": 90,
          "title": "The latitude part of geo search fields (latitude / longitude). Sample value: 15.4041"
        },
        "links": {
          "id": "#obj/items/properties/links",
          "type": "array",
          "title": "The list of any links.",
          "items": {
            "id": "#obj/items/properties/links/items",
            "type": "string",
            "title": "The links items schema. Sample values: \"https://www.acmefinserv.com/guidelines.html\",
\"https://www.acmefinserv.com/governance.html\".",
            "pattern": "^(.*)$"
          }
```

```
        },
        "location": {
          "id": "#obj/items/properties/location",
          "type": "array",
          "title": "The location, if any.",
          "items": {
            "id": "#obj/items/properties/location/items",
            "type": "string",
            "title": "The location items schema. Sample value: \"699 Eagles Nest Drive, Chico CA, 95926\".",
            "pattern": "^(.*)$"
          }
        },
        "longitude": {
          "id": "#obj/items/properties/longitude",
          "type": "number",
          "minimum": -180,
          "maximum": 180,
          "title": "The longitude part of geo search fields (latitude / longitude). Sample value: 30.1675"
        },
        "object_type": {
          "id": "#obj/items/properties/object_type",
          "type": "string",
          "title": "The type of the object. Sample value: \"emailmessage\".",
          "pattern": "^(.*)$"
        },
        "size": {
          "id": "#obj/items/properties/size",
          "type": "number",
          "title": "The size of the object, if available. Sample value: 150603"
        },
        "source_id": {
          "id": "#obj/items/properties/source_id",
          "type": "string",
          "title": "The ID of the object. This is the unique ID of the object in its original data source. This
value is expected to be the same as that of the id_id property. Upon uploading, the id_id value, however, is
prepended by the content_source value so as to make it globally unique across documents from potentially multiple
original sources. Sample value: \"075de4fdf884a905fb28a3ea21faa844\".",
```

```
          "pattern": "^(.*)$"
        },
        "sourcepath": {
          "id": "#obj/items/properties/sourcepath",
          "type": "string",
          "title": "The path to the object in the original data source; typically a file path, for file system
types of data sources. Sample value: \"/mnt/acme/files/investors/2019/PotentialInvestors.docx\".",
          "pattern": "^(.*)$"
        },
        "subject": {
          "id": "#obj/items/properties/subject",
          "type": "array",
          "title": "The subject value(s), typically on an email.",
          "items": {
            "id": "#obj/items/properties/subject/items",
            "type": "string",
            "title": "The subject items schema. Sample value: \"Potential investors\".",
            "pattern": "^(.*)$"
          }
        },
        "tags": {
          "id": "#obj/items/properties/tags",
          "type": "array",
          "title": "The list of tags on the object, if any.",
          "items": {
            "id": "#obj/items/properties/tags/items",
            "type": "string",
            "title": "The tags items schema. Sample values: \"reviewed\", \"projections\".",
            "pattern": "^(.*)$"
          }
        },
        "teaser": {
          "id": "#obj/items/properties/teaser",
          "type": "string",
          "title": "The teaser for the object. Sample value: \"Hi there. Looking to raise about $30mm from the
attached...\".",
          "pattern": "^(.*)$"
```

```
        },
        "text": {
          "id": "#obj/items/properties/text",
          "type": "array",
          "title": "The textual content of the object.",
          "items": {
            "id": "#obj/items/properties/text/items",
            "type": "string",
            "title": "The text items schema. Sample value: \"Hi there. Looking to raise about $30mm from the
attached list of names. Let me know if you can use any of these. Thanks, John\".",
            "pattern": "^(.*)$"
          }
        },
        "title": {
          "id": "#obj/items/properties/title",
          "type": "array",
          "title": "The title(s) of the object.",
          "items": {
            "id": "#obj/items/properties/title/items",
            "type": "string",
            "title": "The title items schema. Sample value: \"Financial forecast for Q3\".",
            "pattern": "^(.*)$"
          }
        },
        "to": {
          "id": "#obj/items/properties/to",
          "type": "array",
          "title": "The list of the 'to' recipients, typically on an email.",
          "items": {
            "id": "#obj/items/properties/to/items",
            "type": "string",
            "title": "The to items schema. Sample value: \"er92950@acmeinc.com\", \"rk67890@acmeinc.com\".",
            "pattern": "^(.*)$"
          }
        },
        "uri": {
          "id": "#obj/items/properties/uri",
```

```
          "type": "string",
          "format": "uri",
          "title": "The URI of the object; typically this is a universal resource identifier pointing back at the
location where the object resides in its original data source. Sample value:
\"file:/mnt/acme/files/investors/2019/PotentialInvestors.docx\".",
          "pattern": "^(.*)$"
        }
      }
    }
  },
  "type": "object",
  "required": [
    "objects"
  ],
  "properties": {
    "schema_info": {
      "id": "#/definitions/schema_info",
      "type": "string",
      "title": "The version string summary of the upload JSON schema. Sample value: \"http://konadsc.com/data-
upload-v1.json\".",
      "pattern": "^(.*)$"
    },
    "created_by": {
      "id": "#/definitions/created_by",
      "type": "string",
      "title": "The short description of the generator of the JSON data. Sample value: \"Kona data uploader\".",
      "pattern": "^(.*)$"
    },
    "generated_datetime": {
      "id": "#/definitions/generated_datetime",
      "type": "string",
      "format": "date-time",
      "title": "The timestamp of when the JSON data was generated. Sample value: \"2019-06-17T17:44:55Z\".",
      "pattern": "^(.*)$"
    },
    "objects": {
      "type": "array",
```

```
      "items": { "$ref": "#/definitions/obj" }
    }
  }
}
```

Sample Files

Below are some examples of Kona data upload JSON files:

- Example 1 - email messages
- Example 2 - files from a file system
- Example 3 - extracted database records
- Example 4 - a comprehensive sample illustrating various aspects of the upload JSON

Sample JSON File - 1 - Email Messages

Below is an example of how a set of email messages may be represented as Kona upload data JSON.

- The first object represents an email message with two attachments, an Excel spreadsheet and a Word document.
- The second object represents an email message with an Excel spreadsheet attached.

```
{
  "schema_info": "http://konadsc.com/kona-data-upload-json-schema-v1.json",
  "created_by": "Kona data uploader",
  "generated_datetime": "2019-06-25T17:44:55Z",
  "objects": [
    {
      "author": [
        "jsmith1234@acmeinc.com"
      ],
      "bcc": [
        "abarnes2343@gmail.com",
        "samantha.obrien4456456@yahoo.com"
      ],
      "cc": [
```

```
        "mbennett0404@gmail.com",
        "stucker34534@gmail.com"
      ],
      "content_source": "Acme_Email_Server",
      "createddate_datetime": "2019-06-17T17:44:55Z",
      "from": [
        "jsmith1234@acmeinc.com"
      ],
      "id_id": "075de4fdf884a905fb28a3ea21faa844",
      "job_id": "job-123",
      "lastmodifieddate_datetime": "2019-06-17T17:44:55Z",
      "lastmodifier": "jsmith1234@acmeinc.com",
      "object_type": "emailmessage",
      "source_id": "075de4fdf884a905fb28a3ea21faa844",
      "subject": [
        "Potential investors"
      ],
      "text": [
        "Hi there. Looking to raise about $30mm from the attached list of names. Let me know if you can use any
of these. Thanks, John"
      ],
      "to": [
        "er92950@acmeinc.com",
        "rk67890@acmeinc.com"
      ],
      "cbpid_reference": "0012v00002KLb5cAAD",
      "child_objects": [
        {
          "bytes": [
            "24534534.....=="
          ],
          "content_source": "Acme_Email_Server",
          "createddate_datetime": "2019-06-17T17:44:55Z",
          "filename": "MyFinReport2018.xls",
          "id_id": "075de4fdf884a905fb28a3ea21faa844_MyFinReport2018.xls",
          "is_attachment_boolean": true,
          "object_type": "file",
```

```
        "size": 3538683,
        "cbpid_reference": "0012v00002KLb5cAAD",
        "source_id": "075de4fdf884a905fb28a3ea21faa844_MyFinReport2018.xls",
        "job_id": "job-123"
      },
      {
        "bytes": [
          "453453463.....=="
        ],
        "content_source": "Acme_Email_Server",
        "createddate_datetime": "2019-06-17T17:44:55Z",
        "filename": "PotentialInvestors.docx",
        "id_id": "075de4fdf884a905fb28a3ea21faa844_PotentialInvestors.docx",
        "is_attachment_boolean": true,
        "object_type": "file",
        "size": 150603,
        "cbpid_reference": "0012v00002KLb5cAAD",
        "source_id": "075de4fdf884a905fb28a3ea21faa844_PotentialInvestors.docx",
        "job_id": "job-123"
      }
    ]
  },
  {
    "author": [
      "wschumaker@acmeinc.com"
    ],
    "content_source": "Acme_Email_Server",
    "createddate_datetime": "2019-06-17T18:13:50Z",
    "from": [
      "wschumaker@acmeinc.com"
    ],
    "id_id": "786abfcfd7859a05ba30a3ae12ade855",
    "job_id": "job-123",
    "lastmodifieddate_datetime": "2019-06-17T18:13:50Z",
    "lastmodifier": "wschumaker@acmeinc.com",
    "object_type": "emailmessage",
    "source_id": "786abfcfd7859a05ba30a3ae12ade855",
```

```
      "subject": [
        "Q3 Projections"
      ],
      "text": [
        "Hi Jay, attached are my projections for the next quarter. Regards, Bill."
      ],
      "to": [
        "jaybrown2@acmeinc.com"
      ],
      "cbpid_reference": "0012v00002KLb5cAAD",
      "child_objects": [
        {
          "bytes": [
            "4364574745.....=="
          ],
          "content_source": "Acme_Email_Server",
          "createddate_datetime": "2019-06-17T17:44:55Z",
          "filename": "q3_projections.xlsx",
          "id_id": "786abfcfd7859a05ba30a3ae12ade855_q3_projections.xlsx",
          "is_attachment_boolean": true,
          "object_type": "file",
          "size": 27082,
          "cbpid_reference" : "0012v00002KLb5cAAD",
          "job_id": "job-123"
        }
      ]
    }
  ]
}
```

Sample JSON File - 2 - Files from File System

Below is an example of how file crawled from a file system can be represented as Kona upload JSON.

- The first object represents a Word document, with its content included as Base64-encoded bytes.
- The second object represents an Excel spreadsheet, with its content also included as Base64-encoded bytes.

```
{
```

```
"schema_info": "http://konadsc.com/kona-data-upload-json-schema-v1.json",
"created_by": "Kona data uploader",
"generated_datetime": "2019-06-25T17:44:55Z",
"objects": [
  {
    "bytes": [
      "0M8R4KGxGuEAA....=="
    ],
    "content_source": "Acme File Server",
    "createddate_datetime": "2019-06-17T17:44:50Z",
    "filename": "PotentialInvestors.docx",
    "id_id": "2948b248-4668-4cab-97ab-8ca9e0043cfa",
    "object_type": "file",
    "source_id": "2948b248-4668-4cab-97ab-8ca9e0043cfa",
    "sourcepath": "/mnt/acme/files/investors/2019/PotentialInvestors.docx",
    "size": 150603,
    "uri": "file:/mnt/acme/files/investors/2019/PotentialInvestors.docx",
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "job_id": "job-123"
  },
  {
    "bytes": [
      "UEsDBBQABgAIA....=="
    ],
    "content_source": "Acme File Server",
    "createddate_datetime": "2019-06-17T17:44:55Z",
    "filename": "q3_projections.xlsx",
    "id_id": "42858215-496b-43e2-a479-ba0b79fe63b1",
    "object_type": "file",
    "size": 27082,
    "source_id": "42858215-496b-43e2-a479-ba0b79fe63b1",
    "sourcepath": "/mnt/acme/files/analytics/2019/q3_projections.xlsx",
    "uri": "file:/mnt/acme/files/analytics/2019/q3_projections.xlsx",
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "job_id": "job-123"
  }
]
```

```
}
```

Sample JSON File - 3 - Database Records

Below is an example of how extracted database records can be represented as Kona upload JSON.

- This sample represents a fictitious report made of Acme Inc. orders database.
- The report is a list of customer records extracted from CUSTOMERS table and also a join into the ORDERS table, to extract the order ID's per customer.
- Thus, in this JSON representation, each customer record has customer metadata as well as the list of one or more order numbers for orders associated with the respective customer.
- This sample provides an example of how to create JSON with object that contain 'custom' properties.
    - For example, CUST_CODE is a field in the CUSTOMERS table which is represented here as 'cust_code_id', to indicate that it is an ID.
    - The 'cust_name_string' field is actually the CUST_NAME field in the CUSTOMERS table and is represented here with the suffix '_string' to indicate that this is a string property.
    - Similarly, 'grade_integer' is GRADE (type = INT) in the CUSTOMERS table and is represented here with the suffix '_integer' to indicate that this is an integer property.
    - The currency suffix '_integer' is used for table columns OPENING_AMT, RECEIVE_AMT, PAYMENT_AMT, and OUTSTANDING_AMT to indicate their respective JSON properties are of type 'currency' (i.e. double).
    - The '_datetime' suffix on 'processed_datetime' is used for the table column of type TIMESTAMP, to indicate that this property holds date values.
    - The actual data type is included in the "t" JSON field for each of the custom properties.

```
{
  "schema_info": "http://konadsc.com/kona-data-upload-json-schema-v1.json",
  "created_by": "Kona data uploader",
  "generated_datetime": "2019-06-25T17:44:55Z",
  "objects": [
    {
      "id_id": "C00013-20190617230000",
      "content_source": "Acme Orders DB",
      "createddate_datetime": "2019-06-17T17:44:55Z",
      "object_type": "dbrecord",
      "cbpid_reference" : "0012v00002KLb5cAAD",
```

```
"source_id": "C00013-20190617230000",
"job_id": "job-123",
"metadata": [
  {
    "n": "cust_code_id",
    "t": "string",
    "v": "C00013"
  },
  {
    "n": "cust_name_string",
    "t": "string",
    "v": "Holmes Ltd."
  },
  {
    "n": "cust_city_string",
    "t": "string",
    "v": "London"
  },
  {
    "n": "working_area_string",
    "t": "string",
    "v": "London"
  },
  {
    "n": "cust_country_string",
    "t": "string",
    "v": "UK"
  },
  {
    "n": "grade_integer",
    "t": "int",
    "v": "2"
  },
  {
    "n": "opening_amt_currency",
    "t": "double",
    "v": "6000.00"
```

```
      },
      {
        "n": "receive_amt_currency",
        "t": "double",
        "v": "5000.00"
      },
      {
        "n": "payment_amt_currency",
        "t": "double",
        "v": "7000.00"
      },
      {
        "n": "outstanding_amt_currency",
        "t": "double",
        "v": "4000.00"
      },
      {
        "n": "order_numbers_id",
        "t": "string",
        "v": "200100 | 200101"
      },
      {
        "n": "processed_datetime",
        "t": "date",
        "v": "2019-06-17T23:00:00Z"
      }
    ]
  },
  {
    "id_id": "C00001-20190617230000",
    "content_source": "Acme Orders DB",
    "createddate_datetime": "2019-06-17T17:44:55Z",
    "object_type": "dbrecord",
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "source_id": "C00001-20190617230000",
    "job_id": "job-123",
    "metadata": [
```

```
{
  "n": "cust_code_id",
  "t": "string",
  "v": "C00001"
},
{
  "n": "cust_name_string",
  "t": "string",
  "v": "SPQ Industries"
},
{
  "n": "cust_city_string",
  "t": "string",
  "v": "New York"
},
{
  "n": "cust_city_string",
  "t": "string",
  "v": "New York"
},
{
  "n": "working_area_string",
  "t": "string",
  "v": "New York"
},
{
  "n": "cust_country_string",
  "t": "string",
  "v": "USA"
},
{
  "n": "grade_integer",
  "t": "int",
  "v": "2"
},
{
  "n": "opening_amt_currency",
```

```
        "t": "double",
        "v": "3000.00"
      },
      {
        "n": "receive_amt_currency",
        "t": "double",
        "v": "5000.00"
      },
      {
        "n": "payment_amt_currency",
        "t": "double",
        "v": "2000.00"
      },
      {
        "n": "outstanding_amt_currency",
        "t": "double",
        "v": "6000.00"
      },
      {
        "n": "order_numbers_id",
        "t": "string",
        "v": "200102 | 200103 | 200104"
      },
      {
        "n": "processed_datetime",
        "t": "date",
        "v": "2019-06-17T23:00:00Z"
      }
    ]
  },
  {
    "id_id": "C00020-20190617230000",
    "content_source": "Acme Orders DB",
    "createddate_datetime": "2019-06-17T17:44:55Z",
    "object_type": "dbrecord",
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "source_id": "C00020-20190617230000",
```

```
"job_id": "job-123",
"metadata": [
  {
    "n": "cust_code_id",
    "t": "string",
    "v": "C00020"
  },
  {
    "n": "cust_name_string",
    "t": "string",
    "v": "Albert Johnson Inc."
  },
  {
    "n": "cust_city_string",
    "t": "string",
    "v": "New York"
  },
  {
    "n": "working_area_string",
    "t": "string",
    "v": "New York"
  },
  {
    "n": "cust_country_string",
    "t": "string",
    "v": "USA"
  },
  {
    "n": "grade_integer",
    "t": "int",
    "v": "3"
  },
  {
    "n": "opening_amt_currency",
    "t": "double",
    "v": "4000.00"
  },
```

```
        {
          "n": "receive_amt_currency",
          "t": "double",
          "v": "7000.00"
        },
        {
          "n": "payment_amt_currency",
          "t": "double",
          "v": "6000.00"
        },
        {
          "n": "outstanding_amt_currency",
          "t": "double",
          "v": "6000.00"
        },
        {
          "n": "order_numbers_id",
          "t": "string",
          "v": "200105 | 200106 | 200107"
        },
        {
          "n": "processed_datetime",
          "t": "date",
          "v": "2019-06-17T23:00:00Z"
        }
      ]
    }
  ]
}
```


Sample JSON File - 4 - Comprehensive Sample

Below is a comprehensive data upload JSON sample which demonstrates the following:

1. The overall structure.
2. All the required properties, with examples (e.g. "id_id", "content_source", "object_type").
3. All the 'Core' metadata properties are included with example values.

4. Usage of nested objects is illustrated.
5. Specifying the Base64 encoded data for raw content bytes is included (see "bytes").
6. Handling of deleted objects is illustrated (see "is_deleted_boolean").
7. Handling of custom, non-core properties is illustrated. See e.g. "customer_name_string", "is_reviewed_boolean", "published_datetime".


```
{
  "schema_info": "http://konadsc.com/kona-data-upload-json-schema-v1.json",
  "created_by": "Kona data uploader",
  "generated_datetime": "2019-06-17T17:44:55Z",
  "objects": [
    {
      "id_id": "48040dc7-0a4f-44ca-8057-5ca29cd2b9e7",
      "content_source": "Acme_Test_Data",
      "createddate_datetime": "2019-06-17T17:44:55Z",
      "object_type": "acme_test_1",
      "author": [
        "jsmith1234@acmeinc.com"
      ],
      "bcc": [
        "abarnes2343@gmail.com",
        "samantha.obrien4456456@yahoo.com"
      ],
      "bytes": [
        "0M8R4KGxGuEAA....=="
      ],
      "cbpid_reference" : "0012v00002KLb5cAAD",
      "cc": [
        "mbennett0404@gmail.com",
        "stucker34534@gmail.com"
      ],
      "comments": [
        "This is the first sample object.",
        "This object has some comments on it."
      ],
      "company": [
        "Acme Financial Services Inc."
      ],
```

```
      "description": "The first object",
      "filename": "file1.eml",
      "from": [
        "jsmith1234@acmeinc.com"
      ],
      "job_id": "job-123",
      "keywords": [
        "analytics",
        "projections",
        "Q3"
      ],
      "lastmodifieddate_datetime": "2019-06-17T17:44:55Z",
      "lastmodifier": "jsmith1234@acmeinc.com",
      "latitude": 15.4041,
      "links": [
        "https://www.acmefinserv.com/guidelines.html",
        "https://www.acmefinserv.com/governance.html"
      ],
      "location": [
        "699 Eagles Nest Drive, Chico CA, 95926"
      ],
      "longitude": 30.1675,
      "source_id": "48040dc7-0a4f-44ca-8057-5ca29cd2b9e7",
      "subject": [
        "Potential investors"
      ],
      "tags": [
        "draft3",
        "reviewed",
        "approved"
      ],
      "teaser": "Hi there. Looking to raise about $30mm from the attached...",
      "text": [
        "Hi there. Looking to raise about $30mm from the attached list of names. Let me know if you can use any
of these. Thanks, John",
        "Addendum: Will follow up with more information."
      ],
```

```
"title": [
  "Financial forecast for Q3"
],
"to": [
  "er92950@acmeinc.com",
  "rk67890@acmeinc.com"
],
"child_objects": [
  {
    "bytes": [
      "24534534.....=="
    ],
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "content_source": "Acme_Email_Server",
    "createddate_datetime": "2019-06-17T17:44:55Z",
    "filename": "MyFinReport2018.xls",
    "id_id": "075de4fdf884a905fb28a3ea21faa844_MyFinReport2018.xls",
    "is_attachment_boolean": true,
    "job_id": "job-123",
    "object_type": "file",
    "size": 3538683,
    "source_id": "075de4fdf884a905fb28a3ea21faa844_MyFinReport2018.xls",
    "sourcepath": "/mnt/acme/files/analytics/2019/MyFinReport2018.xls",
    "uri": "file:/mnt/acme/files/analytics/2019/MyFinReport2018.xls"
  },
  {
    "bytes": [
      "453453463.....=="
    ],
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "content_source": "Acme_Email_Server",
    "createddate_datetime": "2019-06-17T17:44:55Z",
    "filename": "PotentialInvestors.docx",
    "id_id": "075de4fdf884a905fb28a3ea21faa844_PotentialInvestors.docx",
    "is_attachment_boolean": true,
    "job_id": "job-123",
    "object_type": "file",
```

```
        "size": 150603,
        "source_id": "075de4fdf884a905fb28a3ea21faa844_PotentialInvestors.docx",
        "sourcepath": "/mnt/acme/files/investors/2019/PotentialInvestors.docx",
        "uri": "file:/mnt/acme/files/investors/2019/PotentialInvestors.docx"
      }
    ]
  },
  {
    "id_id": "e3eb5a57-6963-4e1a-8d54-23ce360ae5fb",
    "content_source": "Acme_Test_Data",
    "createddate_datetime": "2019-06-17T17:44:57Z",
    "object_type": "acme_test_1",
    "author": [
      "kbrown@acmeinc.com"
    ],
    "bcc": [
      "mbennett0404@gmail.com",
      "stucker34534@gmail.com"
    ],
    "bytes": [
      "UEsDBBQABgAIA....=="
    ],
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "cc": [
      "abarnes2343@gmail.com",
      "samantha.obrien4456456@yahoo.com"
    ],
    "comments": [
      "This is the second sample object.",
      "This object has some comments on it."
    ],
    "company": [
      "Acme Financial Services Inc."
    ],
    "description": "The second object",
    "filename": "file2.eml",
    "from": [
```

```
    "wschumaker@acmeinc.com"
  ],
  "job_id": "job-123",
  "lastmodifieddate_datetime": "2019-06-17T17:44:57Z",
  "lastmodifier": "gphentges@acmeinc.com",
  "source_id": "e3eb5a57-6963-4e1a-8d54-23ce360ae5fb",
  "subject": [
    "Q3 Projections"
  ],
  "tags": [
    "draft1",
    "in-review"
  ],
  "teaser": "Hi Jay, attached are my projections...",
  "text": [
    "Hi Jay, attached are my projections for the next quarter. Regards, Bill."
  ],
  "to": [
    "jaybrown2@acmeinc.com"
  ],
  "child_objects": [
    {
      "bytes": [
        "4364574745.....=="
      ],
      "cbpid_reference" : "0012v00002KLb5cAAD",
      "content_source": "Acme_Email_Server",
      "createddate_datetime": "2019-06-17T17:44:55Z",
      "filename": "q3_projections.xlsx",
      "id_id": "786abfcfd7859a05ba30a3ae12ade855_q3_projections.xlsx",
      "is_attachment_boolean": true,
      "job_id": "job-123",
      "object_type": "file",
      "size": 27082,
      "source_id": "786abfcfd7859a05ba30a3ae12ade855_q3_projections.xlsx",
      "sourcepath": "/mnt/acme/files/analytics/2019/q3_projections.xlsx",
      "uri": "file:/mnt/acme/files/analytics/2019/q3_projections.xlsx"
```

```
      }
    ]
  },
  {
    "id_id": "fc4817b7-641e-42c6-a13a-9ee2ca31a806",
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "content_source": "Acme_Test_Data",
    "createddate_datetime": "2019-06-17T17:44:57Z",
    "job_id": "job-123",
    "object_type": "acme_test_2",
    "source_id": "fc4817b7-641e-42c6-a13a-9ee2ca31a806",
    "is_deleted_boolean": true
  },
  {
    "id_id": "af16d9d3-178e-422c-91a0-a821b316515d",
    "cbpid_reference" : "0012v00002KLb5cAAD",
    "content_source": "Acme_Test_Data",
    "createddate_datetime": "2019-06-17T14:20:30Z",
    "job_id": "job-123",
    "object_type": "acme_test_3",
    "source_id": "af16d9d3-178e-422c-91a0-a821b316515d",
    "metadata": [
      {
        "n": "customer_name_string",
        "t": "string",
        "v": "Acme Inc."
      },
      {
        "n": "order_notes_string",
        "t": "string",
        "v": "This order has been completed. | Need to follow up with customer on other recommended items."
      },
      {
        "n": "custid_id",
        "t": "string",
        "v": "1234567"
      },
```

```
{
  "n": "messagesender_email",
  "t": "string",
  "v": "jsmith@acmefin.com"
},
{
  "n": "customer_phone",
  "t": "string",
  "v": "(617) 245-9566"
},
{
  "n": "website_url",
  "t": "string",
  "v": "www.acmeinc.com/contact_us.html"
},
{
  "n": "createdbyid_reference",
  "t": "string",
  "v": "00500000000740QDAAY"
},
{
  "n": "is_reviewed_boolean",
  "t": "boolean",
  "v": "true"
},
{
  "n": "amount_integer",
  "t": "int",
  "v": "200"
},
{
  "n": "total_amt_integer",
  "t": "int",
  "v": "1000"
},
{
  "n": "item_fingerprint_long",
```

```
    "t": "long",
    "v": "1242342345"
  },
  {
    "n": "meter_reading_double",
    "t": "double",
    "v": "120.9"
  },
  {
    "n": "total_reading_double",
    "t": "double",
    "v": "234234.33"
  },
  {
    "n": "some_value_float",
    "t": "float",
    "v": "3423.3453"
  },
  {
    "n": "order_amount_currency",
    "t": "double",
    "v": "22.99"
  },
  {
    "n": "review_date",
    "t": "date",
    "v": "2019-06-14Z"
  },
  {
    "n": "published_datetime",
    "t": "date",
    "v": "2019-06-17T17:44:50Z"
  },
  {
    "n": "customer_contactinfo",
    "t": "string",
    "v": "761 Garrett St., Grand Rapids MI 49503"
```

```
        },
        {
          "n": "ordernumber",
          "t": "int",
          "v": "234"
        },
        {
           "n": "abc_multipicklist",
           "t": "string",
           "v": "aaa; bbb; ccc"
        }
      ]
    }
  ]
}
```

Kona Data Upload XML Schema

Spec version: 1
Last modified: Oct. 8, 2019

Summary

Below is the summary of the XML schema for the Kona data upload XML files.

The general XML schema documentation can be found here.

The kona-data-upload-v1.xsd Schema

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- The upload object data type -->
  <xs:complexType name="uploadObject">
    <xs:all>
      <!-- Start mandatory core properties -->
      <xs:element type="xs:string" name="id_id" maxOccurs="1" minOccurs="1" />
      <xs:element type="xs:string" name="content_source" maxOccurs="1" minOccurs="1" />
      <xs:element type="xs:string" name="object_type" maxOccurs="1" minOccurs="1" />
      <!-- End mandatory core properties -->
      <!-- Start optional core properties -->
      <xs:element name="author" maxOccurs="1" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="bcc" maxOccurs="1" minOccurs="0">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="bytes" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element type="xs:string" name="cbpid_reference" maxOccurs="1" minOccurs="0" />
  <xs:element name="cc" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!-- Child objects such as attachments are also upload objects. -->
  <xs:element name="child_objects" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="object" maxOccurs="unbounded" minOccurs="1" type="uploadObject" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="comments" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="company" maxOccurs="1" minOccurs="0">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element type="xs:dateTime" name="createddate_datetime" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:string" name="description" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:string" name="filename" maxOccurs="1" minOccurs="0" />
  <xs:element name="from" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element type="xs:boolean" name="is_attachment_boolean" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:boolean" name="is_deleted_boolean" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:string" name="job_id" maxOccurs="1" minOccurs="0" />
  <xs:element name="keywords" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element type="xs:dateTime" name="lastmodifieddate_datetime" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:string" name="lastmodifier" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:double" name="latitude" maxOccurs="1" minOccurs="0" />
  <xs:element name="links" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="location" maxOccurs="1" minOccurs="0">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element type="xs:double" name="longitude" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:unsignedLong" name="size" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:string" name="source_id" maxOccurs="1" minOccurs="0" />
  <xs:element type="xs:string" name="sourcepath" maxOccurs="1" minOccurs="0" />
  <xs:element name="subject" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="tags" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element type="xs:string" name="teaser" maxOccurs="1" minOccurs="0" />
  <xs:element name="text" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="title" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
```

```
    </xs:complexType>
  </xs:element>
  <xs:element name="to" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="v" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element type="xs:string" name="uri" maxOccurs="1" minOccurs="0" />
  <!-- End optional core properties -->
  <!-- Any custom metadata properties other than the core properties -->
  <xs:element name="metadata" maxOccurs="1" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="prop" maxOccurs="unbounded" minOccurs="1">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <!-- The property name -->
                <xs:attribute type="xs:string" name="n" use="required" />
                <!-- The property type. If omitted, string type is assumed. -->
                <xs:attribute name="t" use="optional">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <!-- Base64 encoded binary data -->
                      <xs:enumeration value="binary" />
                      <!-- Boolean: true or false -->
                      <xs:enumeration value="boolean" />
                      <!-- Date, as in xsd:dateTime -->
                      <xs:enumeration value="date" />
                      <!-- Double -->
                      <xs:enumeration value="double" />
                      <!-- Float -->
                      <xs:enumeration value="float" />
                      <!-- Integer -->
                      <xs:enumeration value="int" />
```

```
                        <!-- Long -->
                        <xs:enumeration value="long" />
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:attribute>
                  <!-- The property value -->
                  <xs:attribute type="xs:string" name="v" use="required" />
                </xs:extension>
              </xs:simpleContent>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>
<!-- Top-level element -->
<xs:element name="upload_data">
  <xs:complexType>
    <xs:all>
      <!-- Header info items -->
      <xs:element type="xs:string" name="created_by" maxOccurs="1" minOccurs="1" />
      <xs:element type="xs:dateTime" name="generated_datetime" maxOccurs="1" minOccurs="1" />
      <!-- The upload objects -->
      <xs:element name="objects" maxOccurs="1" minOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="object" maxOccurs="unbounded" minOccurs="0" type="uploadObject" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:all>
  </xs:complexType>
</xs:element>
</xs:schema>
```