CentricSoftware™

Centric 8 **REST API DEVELOPER GUIDE**

Version 2, Rev 1
February 2017

Centric Software, Inc.
655 Campbell Technology Parkway,
Suite 200, Campbell
CA 95008
Ph. No.: 1.408.574.7802

Technical Support:
866. 796-6218 8:00 AM-8:00 PM EST
email: support@centricsoftware.com

Documentation Feedback:
documentation@centricsoftware.com

TABLE OF CONTENTS

ABOUT THIS GUIDE

Using this Guide

This Guide provides details about the Centric 8 REST API.

Technical Support

Customer Support is available to all registered users of Centric 8.

866-796-6218 8:00 AM-8:00 PM EST

Email: support@centricsoftware.com

1. CENTRIC 8 REST API

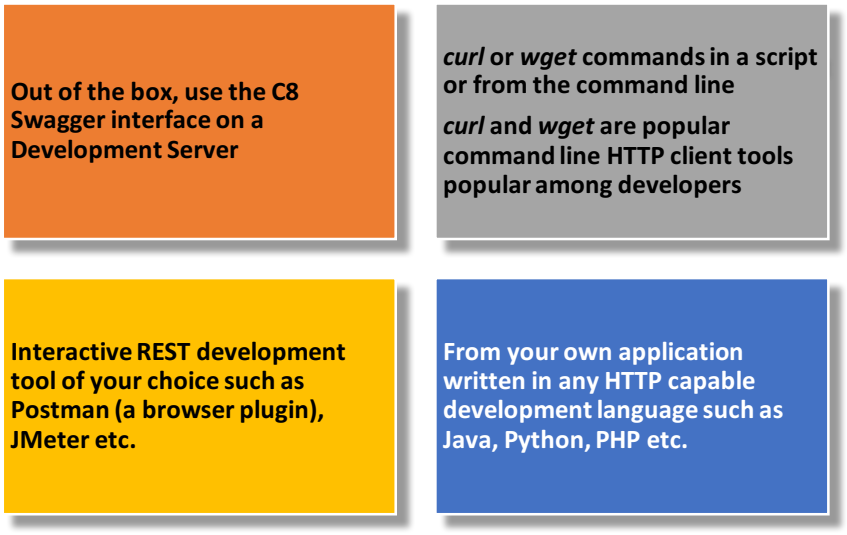
What is the Centric 8 REST API?

The Centric 8 REST API provides access to Centric 8 data (domain objects) using HTTP URI resources. You can use the API to integrate Centric 8 with other enterprise, web and mobile applications. The REST API provides programmatic access to core C8 resources which will benefit from automated integration with external applications.

How can I access it?

The Centric 8 REST API is based on the HTTP open standard. You communicate with this API by making a standard HTTP request and getting the result back in an HTTP response. The API will allow you to perform CRUD (Create Read Update Delete) operations by invoking the API using HTTP **POST**, **GET**, **PUT** and **DELETE** request methods. The API will restrict the logged-in API user to CRUD operations authorized by their C8 role..

Here are multiple ways you can interact with the C8 REST API:



What is Centric 8 REST URI (endpoint)?

The Centric 8 REST API is organized into related groups of information (such as "Season", "Style" etc.) known as "resources". Each resource endpoint has one or

more HTTP Uniform Resource Identifiers (URI) to request specific operations. You invoke the API's CRUD and support operations by making HTTP requests to the appropriate endpoint URI.

1.1.1 URI SYNTAX

The Centric 8 REST URIs have the following syntax:

```
http[s]://{c8server}/csi-requesthandler/api/{version}/{resource_path}
```

Where:

- *http[s]*: Is the URI scheme and indicates a clear text or encrypted connection
- **`{c8server}`**: The name of the Centric 8 server hosting the REST service (this part of the URI might include a port number).
- **`csi-requesthandler/api`**: This should be entered verbatim as it is part of Centric 8 REST URI.
- **`{version}`**: The version of the Centric 8 REST APIs, such as v1, v2 etc. The current version is *V2*.
- **`{resource_path}`**: A `{resource_path}` identifies a particular resource and provides qualifying parameters for the specific resource.
- **{}** Braces ("curly braces") are used to identify a mandatory parameter.
- **[]** Brackets ("square brackets") are used to identify optional parameters.
- **<>** Angle Brackets will be used to denote a term with an extended definition.

1.1.2 VERSION NAMING CONVENTION

We follow the widely adopted REST API versioning approach specifying the version number in the REST URI as illustrated above.

The benefit of this versioning convention is:
The previous version API will still remain accessible and usable untill it is retired under **Centric API deprecation policy**

2. C8 REST ENDPOINT EXAMPLE

Endpoint	GET	PUT	POST	DELETE
`http://c8server/csi-requesthandler/v2/seasons`	Retrieves a collection of Season resources which match the optional search and control parameters		Creates a new season using details provided in the POST request body	
`http://c8server/csi-requesthandler/v2/seasons/C2234`	Retrieves a specific C8 Season resource details, identified by the resource_id	Updates a specific C8 season details, identified by the resource_id		Deletes a specific C8 Season, identified by the resource_id

This is a simple example. In subsequent sections, you will note a variation that some resources are created in a hierarchal structure. For example, a Category1 resource is created using a POST request to the Season endpoint.

3. HOW TO AUTHENTICATE IN CENTRIC 8 REST API?

You must be authenticated as a C8 user to access the REST API using any of the HTTP requests created using the previously described methods. You authenticate by using the C8 REST API Session endpoint and store the authentication token returned by the C8 server such that the correct cookies will be returned in subsequent HTTP requests. From that point, use that authentication token every time you use the REST API. Note: the authentication token remains valid for the session duration configured for your C8 server. Once the token has expired, you need to replace the authentication process and replace the authentication token.

If you want to access the C8 REST API in the development server through Swagger interface, then you must login to the Centric 8 Server using the Centric 8 Web UI or using Swagger to invoke the Session endpoint to create a session before using the Swagger interface.

The following examples will show you how to use the Centric 8 HTTP REST API Session endpoint to login and logout:

Endpoint	POST	DELETE

`http://{c8server}/csi-requesthandler/api/{version}/session`	Creates a new C8 session. Authentication tokens are returned in the HTTP response as cookies and in the response body. The request body may be the JSON formatted object:
	`{"username": "<your-username>", "password": "<your-password>" }`
	
	Or as an XML formatted object:
	
	`<session>`
	`<username>{your-username}</username>`
	`<password>{your-password}</password>`
	`</session>`
		Logs out by deleting the existing session at the server thus invalidating the previous authentication tokens.

Login logout example using cURL

cURL is a popular developer tool to send HTTP requests using the command line from a terminal window. In this example, we illustrate how you can use cURL to invoke the Centric 8 REST API to create a session. This is mostly useful for developer while developing a REST client application.

Login:

```
curl -c "C*Token.txt" -I -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "username": "{user_name}",
  "password": "{password}"
}' 'http://{c8_server}/csi-requesthandler/api/v2/session'
```

Logout:

```
curl -b "C8Token.txt" -I -X DELETE "http://{c8_server}/csi-requesthandler/api/v2/session"
```

Replace {c8_server}, {user_name}, {password} with appropriate values

-c saves cookies (including authentication token) in the text file cookies.txt. You will need this file for subsequent REST API calls.

-I returns only the headers, not the response bodies. Use -I if you only want to check for success (HTTP 200) and utilize cookies to retain the authentication tokens.

-X is always necessary

On successful *curl* POST command, you should get a response like this recorded in the C8Token.txt file created in your file system

```
HTTP/1.1 200 OK
Cache-Control: No-cache
Content-Length: 4283
Content-Type: text/javascript;charset=UTF-8
Expires: -1
```

```
Server: Microsoft-IIS/8.5

PIRequestId: 11074

PIRequestTime: 1

X-Powered-By: Undertow/1

Set-Cookie:        JSESSIONID=c27Gm4PVP-x3K4axbQD8cMLg.origin2;
path=/csi-requesthandler

Set-Cookie:
SecurityTokenURL=centric://_CS_SecurityToken/392776baac14022851
79761a198c60d5; path=/; domain=.csi.local

Server: WildFly/8

PIRequestDBTime: 0

PIRequestTXTime: 0

PIRequestEXTime: 0

X-Powered-By: ASP.NET

Date: Sat, 30 Jul 2016 00:15:02 GMT
```

4. HOW TO PASS PARAMETERS TO THE REST API

The industry REST API best practices are very flexible in receiving input parameters from the client application. It supports many different ways to pass parameters, using:

- Query parameter
- Path parameter separated by slashes
- Path segment separated by semicolon
- Header parameter
- Cookie parameter
- Form parameter

The Centric 8 REST API utilizes many of these parameter passing techniques. This REST API documentation will specify how parameters are be passed to the different endpoints. Swagger will illustrate acceptable usage.

5. RESOURCE (ENDPOINT) DETAILS

The C8 REST URI structure is described earlier.

For example: http://myc8server.com/csi-requesthandler/api/v2/

This URI string will prefix all actual request URIs composed from the following documentation. The documentation uses the `./` prefix to call it out as a partial URI. This prefix is not required (but will be ignored by most web servers).

We assume that developers will understand the Centric 8 user interface and the data resource operations performed via the UI. The C8 REST API provides a mechanism for manipulating the same data resources. To understand those potential manipulations, it is critical that the developer understand the Centric 8 application.

Optional Request resource path parameters in any GET Multiple URI

The following documentation refers to the set of resource parameters as `<filter_parameters>`.

Resource path parameters are also referred to as query parameters. They consist of name / value pairs separated by an equal (=) sign. Names and values must be URL encoded (sometimes called percent encoded). The first parameter is separated from the earlier portion of the resource_path with a question-mark (?) character. Additional parameter pairs are separated using an ampersand (&) character.

Parameter	Description
skip	Use this parameter to control pagination for a large list of objects. It controls the starting point of the array by skipping the specific number of objects from the top of the complete result set. For example: if there are total 100 objects, then a skip value of 10 will return a object list starting from 11th object. Skip=0 will return the result from the very first object.
limit	Use this parameter to control pagination for a large list of objects. It controls the number of objects returned in the result array. The default is 10 but may be any positive integer greater than or equal to 1.
modified_after	This parameter's value is a date in yyyy/dd/mmThh:mm:ss.fffZ format. It limits the GET result to resources modified on or after the specified time. The .fff portion is optional.
<field_id>	Any GET response resource field can be used as a search filter parameter except for ***id*** and ***modified_at.*** Search filter parameter values must be correct for the value type of the

	identified resource field. The values serve to limit the response resource collection to resources whose specified field has a value matching the parameter value.

Extended Search for selected EndPoints

To facilitate retrieval of resources associated with a logical container, a new URI pattern has been introduced to support searching for containers and then subsetting the content of the containers using contained resource attribute filters. The initial implementation of this concept is in support of *ImageContainers* **Style**, **Material** and **Colorway**, each of which can reference contained **Image** objects.

The URI path */styles/images?style.parent_season=C3435&tags=front* makes a request of the **Style** EndPoint to locate all **Style**s in the specified **Season** and then return **Image**s owned by the selected **Style**s whose *tags* attribute contains the value: *front*.

HTTP Request and Response Bodies in the C8 REST API

Bodies of HTTP requests and responses may be in JSON or XML format. This is determined by the request content-type (request format) and accept (response format) HTTP request headers. C8 object attribute names are represented in snake case (lower case with underscore characters inserted where the attribute name transitions between cases).

POST requests must provide required values. Depending on the C8 resource, some fields may only be set in a POST request. Read/only fields can not be set in either GET or PUT. Do not include empty valued fields in POST or PUT unless you intend them to be empty in C8.

The response to POST and PUT is always the resulting GET resource.

It is suggested that you use Swagger to obtain a rough approximation of what the JSON or XML would look like in requests and responses.

The optional HTTP Link header can be used to request that the GET response resource include the _links field which is a map from reference field values to actual URLs required to retrieve the referenced object.

Category1

5.1.1 Creating Category1.

POST

URI	Comments
./seasons/{season_id}/hierarchy	Creates a new Category1 object as a child of Season.

5.1.2 Reading Category1

GET

URI	Comments
./seasons/{season_id}/hierarchy[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Category1 objects owned by Season which match the <filter_parameters> included in the resource_path.
./category1s/{category1_id}	Retrieves the specified Category1 object.
./category1s[?[skip=m][[&]limit=n][[& <filter_parameters>]..]	Retrieves the Category1 objects which match the <filter_parameters> included in the resource_path.

5.1.3 Updating Category1

PUT

URI	Comments
./category1s/{category1_id}	Updates the specified Category1 object.

5.1.4 Deleting Category1

DELETE

URI	Comments
./category1s/{category1_id}	Deletes the specified Category1 object.

5.1.5 Attribute list of Category1

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
cnt_colorway	Integer	integer	R/O	
cnt_style	Integer	integer	R/O	
hierarchy	List	reflist	R/O	
issues	List	reflist	R/O	
parent_season	String	ref	R/O	
code	String	string	R/W	
description	String	string	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URI Lookup map.

Category2

5.1.6 Creating Category2.

POST

URI	Comments
./category1s/{category1_id}/hierarchy	Creates a new Category2 object as a child of Category1.

5.1.7 Reading Category2

GET

URI	Comments
./category1s/{category1_id}/hierarchy[?[skip=m][[&]limit=n][[& <filter_parameters>]..]	Retrieves the Category2 objects owned by Category1 which match <filter_parameters> included in the resource_path.
./category2s/{category2_id}	Retrieves the specified Category2 object.
./category2s[?[skip=m][[&]limit=n][[& <filter_parameters>]..]	Retrieves the Category2 objects which match the <filter_parameters> included in the resource_path.

5.1.8 Updating Category2

PUT

URI	Comments
./category2s/{category2_id}	Updates the specified Category2 object.

5.1.9 Deleting Category2

DELETE

URI	Comments
./category2s/{category2_id}	Deletes the specified Category2 object.

5.1.10 Attribute list of Category2

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
cnt_colorway	Integer	integer	R/O	
cnt_style	Integer	integer	R/O	
category_1	String	ref	R/O	
hierarchy	List	reflist	R/O	
issues	List	reflist	R/O	
parent_season	String	ref	R/O	
code	String	string	R/W	
description	String	string	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URI Lookup map.

Collection

5.1.11 Creating Collection

POST

URI	Comments
./category2s/{category2_id}/hierarchy	Creates a new Collection object as a child of Category2.

5.1.12 Reading Collection

GET

URI	Comments
./category2s/{category2_id}/hierarchy[?[skip=m][[&]limit=n][[& <filter_parameters>]..]	Retrieves the Collection objects owned by Category2 which match the <filter_parameters> included in the resource_path.
./collections/{collection_id}	Retrieves the specified Collection object.
./collections[?[skip=m][[&]limit=n][[& <filter_parameters>]..]	Retrieves the Collection objects which match the <filter_parameters> included in the resource_path.

5.1.13 Updating Collection

PUT

URI	Comments
./collections/{collection_id}	Updates the specified Collection object.

5.1.14 Deleting Collection

DELETE

URI	Comments
./collections/{collection_id}	Deletes the specified Collection object.

5.1.15 Attribute list of Collection

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
cnt_colorway	Integer	integer	R/O	
cnt_style	Integer	integer	R/O	
category_1	String	ref	R/O	
category_2	String	ref	R/O	
hierarchy	List	reflist	R/O	
issues	List	reflist	R/O	
parent_season	String	ref	R/O	
code	String	string	R/W	
description	String	string	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

ColorSpecification

5.1.16 Creating ColorSpecification

POST

URI	Comments
./color_specifications	Creates a new ColorSpecification object.

5.1.17 Reading ColorSpecification

GET

URI	Comments
./color_specifications/{color_specification_id}	Retrieves the specified ColorSpecification object.
./color_specifications[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the ColorSpecification objects which match the <filter_parameters> included in the resource_path.

5.1.18 Updating ColorSpecification

PUT

URI	Comments
./color_specifications/{color_specification_id}	Updates the specified ColorSpecification object.

5.1.19 Deleting ColorSpecification

DELETE

URI	Comments
./color_specifications/{color_specification_id}	Deletes the specified ColorSpecification object.

5.1.20 Attribute list of ColorSpecification

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
sys_id	String	string	R/O	
active	Boolean	boolean	R/W	
cmyk	String	string	R/W	
code	String	string	R/W	
color_model	String	string	R/W	
color_type	String	string	R/W	
description	String	string	R/W	
ok_for_material	Boolean	boolean	R/W	
ok_for_style	Boolean	boolean	R/W	
pantone	String	string	R/W	
pantone_tc	String	string	R/W	
rgb	Integer	integer	R/W	(Not: CMYK, RGBHex, RGBTriple)
rgb_hex	String	string	R/W	(Not: CMYK, RGB, RGBTriple)
rgb_triple	String	string	R/W	(Not: CMYK, RGB, RGBHex)
tags	List	stringvector	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

ColorMaterial

5.1.21 Creating ColorMaterial

POST

URI	Comments
./materials/{material_id}/product_colors	Creates a new ColorMaterial object as a child of Material.

5.1.22 Reading ColorMaterial

GET

URI	Comments
./materials/{material_id}/product_colors[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the ColorMaterial objects owned by Material which match the <filter_parameters> included in the resource_path.
./color_materials/{color_material_id}	Retrieves the specified ColorMaterial object.
./color_materials[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the ColorMaterial objects which match the <filter_parameters> included in the resource_path.

5.1.23 Updating ColorMaterial

PUT

URI	Comments
./color_materials/{color_material_id}	Updates the specified ColorMaterial object.

5.1.24 Deleting ColorMaterial

DELETE

URI	Comments
./color_materials/{color_material_id}	Deletes the specified ColorMaterial object.

5.1.25 Attribute list of ColorMaterial

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
cnt_order	Integer	integer	R/O	
cnt_purchased_order	Integer	integer	R/O	
earliest_target_date	String	time	R/O	
status_time	String	time	R/O	
total_order_volume	Double	double	R/O	
total_sample_volume	Double	double	R/O	
color_material_sk_us	List	reflist	R/O	
samples	List	reflist	R/O	
active	Boolean	boolean	R/W	
description	String	string	R/W	
status	String	enum	R/W	
color_material_samples	Map	refmap	R/W	
color_specification	String	ref	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

Colorway

5.1.26 Creating Colorway

POST

URI	Comments
./styles/{style_id}/product_colors	Creates a new Colorway object as a child of Style.

5.1.27 Reading Colorway

GET

URI	Comments
./styles/{style_id}/product_colors[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Colorway objects owned by Style which match the <filter_parameters> included in the resource_path.
./colorways/{colorway_id}	Retrieves the specified Colorway object.
./colorways[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Colorway objects which match the <filter_parameters> included in the resource_path.

5.1.28 Updating Colorway

PUT

URI	Comments
./colorways/{colorway_id}	Updates the specified Colorway object.

5.1.29 Deleting Colorway

DELETE

URI	Comments
./colorways/{colorway_id}	Deletes the specified Colorway object.

5.1.30 Attribute list of Colorway

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
carry_over	Boolean	boolean	R/O	
cnt_customer_purchase_order	Integer	integer	R/O	Customer Purchase Order Count
cnt_order	Integer	integer	R/O	
cnt_purchased_order	Integer	integer	R/O	Purchased Order Count
customer_earliest_target_date	String	time	R/O	
customer_total_order_volume	Double	double	R/O	
earliest_target_date	String	time	R/O	
inline	Boolean	boolean	R/O	
sales_volume	Integer	integer	R/O	
sys_id	String	string	R/O	Identifier to correlate revised entities.
total_order_volume	Double	double	R/O	
total_sample_volume	Double	double	R/O	
carried_over_from_colorways	List	reflist	R/O	
category_1	String	ref	R/O	Containing Category1
category_2	String	ref	R/O	Containing Category2
collection	String	ref	R/O	Containing Collection
colorway_skus	List	reflist	R/O	
copied_from	String	ref	R/O	
images	Map	refmap	R/O	[6.0] Images associated with this Colorway
parent_season	String	ref	R/O	
samples	List	reflist	R/O	
style	String	ref	R/O	
active	Boolean	boolean	R/W	
code	String	string	R/W	Control code.

description	String	string	R/W	
development_type	String	enum	R/W	
intro_date	String	time	R/W	
production_min	Integer	integer	R/W	Production Minimum
color_specification	String	ref	R/W	Referenced by Color Specification
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

Issue

5.1.31 Creating Issue

POST

URI	Comments
./seasons/{season_id}/issues	Creates a new Issue object as a child of Season.
./category1s/{category1_id}/issues	Creates a new Issue object as a child of Category1.
./category2s/{category2_id}/issues	Creates a new Issue object as a child of Category2.
./collections/{collection_id}/issues	Creates a new Issue object as a child of Collection.
./styles/{style_id}/issues	Creates a new Issue object as a child of Style.
./materials/{material_id}/issues	Creates a new Issue object as a child of Material.

5.1.32 Reading Issue

GET

URI	Comments
./seasons/{season_id}/issues[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Issue objects owned by Season which match the <filter_parameters> included in the resource_path.
./category1s/{category1_id}/issues[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Issue objects owned by Category1 which match the <filter_parameters> included in the resource_path.
./cagetory2s/{category2_id}/issues[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Issue objects owned by Category2 which match the <filter_parameters> included in the resource_path.
./collections/{collection_id}/issues[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Issue objects owned by Collection which match the <filter_parameters> included in the resource_path.
./styles/{style_id}/issues[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the Issue objects owned by Style which match the <filter_parameters> included in the resource_path.

`./materials/{material_id}/issues[?[skip=m][[&]limit=n][& <filter_parameters>]..]`	Retrieves the Issue objects owned by Material which match the <filter_parameters> included in the resource_path.
`./issues/{issue_id}`	Retrieves the specified Issue object.
`./issues[?[skip=m][[&]limit=n][& <filter_parameters>]..]`	Retrieves the Issue objects which match the <filter_parameters> included in the resource_path.

5.1.33 Updating Issue

PUT

URI	Comments
`./issues/{issue_id}`	Updates the specified Issue object.

5.1.34 Deleting Issue

DELETE

URI	Comments
`./issues/{issue_id}`	Deletes the specified Issue object.

5.1.35 Attribute list of Issue

Attribute	Java Type	C8 Type	Read Only	Description
`id`	String	CNL	R/O	C8 System ID
`modified_at`	String	time	R/O	Last time this object was modified.
`node_name`	String	string	R/W	Display name for the object.
`status_update_date`	String	time	R/O	
`comment`	String	string	R/W	
`description`	String	string	R/W	
`due_date`	String	time	R/W	
`status`	String	enum	R/W	
`issue_product_sizes`	List	refvector	R/W	
`owner`	String	ref	R/W	
`links (_links)`	Map	refmap	R/O	C8 Ref to URL Lookup map.

Material

5.1.36 Creating Material

POST

URI	Comments
./materials	Creates a new Material object.

5.1.37 Reading Material

GET

URI	Comments
./materials/{material_id}	Retrieves the specified Material object.
./materials[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the Material objects which match the <filter_parameters> included in the resource_path.

5.1.38 Updating Material

PUT

URI	Comments
./materials/{material_id}	Updates the specified Material object.

5.1.39 Deleting Material

DELETE

URI	Comments
./materials/{material_id}	Deletes the specified Material object.

5.1.40 Attribute list of Material

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time object modified.
node_name	String	string	R/W	Display name for the object.
bom_main_material_count	Integer	integer	R/O	
cnt_order	Integer	integer	R/O	
cnt_purchased_order	Integer	integer	R/O	
earliest_target_date	String	time	R/O	
managed	Boolean	boolean	R/O	
status	String	enum	R/O	
status_time	String	time	R/O	
sys_id	String	string	R/O	Identifier to correlate revised entities.
total_order_volume	Double	double	R/O	
total_shipment_qty	Double	double	R/O	
bom_main_materials	List	refset	R/O	
bom_materials	List	refset	R/O	
product_sizes	List	refvector	R/O	
active	Boolean	boolean	R/W	
code	String	string	R/W	Material Code
composition	String	string	R/O	
construction	String	string	R/W	
default_color_size_availability	Boolean	boolean	R/W	
description	String	string	R/W	
diameter	String	string	R/W	
dimension	String	string	R/W	
fob_calc	Double	double	R/W	
fob_negotiated	Double	double	R/W	
finish	String	string	R/W	
has_season_availability	Boolean	boolean	R/W	
is_size_owner	Boolean	boolean	R/O	
is_stock_managed	Boolean	boolean	R/W	
length	String	string	R/W	
main_materials	String	string	R/W	
ok_color_specs	Boolean	boolean	R/W	

ok_for_material_bom	Boolean	boolean	R/W	
ok_for_style_bom	Boolean	boolean	R/W	
qty_per_container	Double	double	R/W	
sample_cost	Double	double	R/W	
sourcing_model	String	enum	R/W	
tags	List	stringvector	R/W	
texture_emboss_ref	String	string	R/W	
thickness	String	string	R/W	
usage	String	string	R/W	
weight	String	string	R/W	
width	String	string	R/W	
actual_size_range	String	ref	R/W	
copied_from	String	ref	R/O	
images	Map	refmap	R/O	[6.0] Images associated with this Material
default_color	String	ref	R/W	
default_size	String	ref	R/W	
issues	List	reflist	R/O	
material_compatibility	List	reflist	R/W	
material_original_category_1	String	ref	R/W	
material_original_season	String	ref	R/W	
product_colors	List	reflist	R/O	
product_type	String	ref	R/W	
realized_products	List	reflist	R/O	
tooling_last	List	reflist	R/W	
tooling_size_range	String	ref	R/W	
tooling_sizes	List	reflist	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

MaterialType (Read Only)

5.1.41 Reading MaterialType

GET

URI	Comments
./material_types/{material_type_id}	Retrieves the specified MaterialType object.
./material_types[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the MaterialType objects which match the <filter_parameters> included in the resource_path.

5.1.42 Creating, Updating, Deleting unsupported

POST

PUT

DELETE

MaterialType is a read-only endpoint. It does not support POST, PUT or DELETE methods.

5.1.43 Attribute list of MaterialType

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
published	Boolean	boolean	R/O	
available	Boolean	boolean	R/W	
has_color	Boolean	boolean	R/W	
has_size	Boolean	boolean	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

ProductSize

5.1.44 Creating ProductSize

POST

URI	Comments
./materials/{material_id}/owned_product_sizes	Creates a new ProductSize object as a child owned by Material.
./product_sizes	Creates a new ProductSize object.

5.1.45 Reading ProductSize

GET

URI	Comments
./materials/{material_id}/owned_product_sizes[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the ProductSize objects owned by Material which match the <filter_parameters> included in the resource_path.
./product_sizes/{product_size_id}	Retrieves the specified ProductSize object.
./product_sizes[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the ProductSize objects which match the <filter_parameters> included in the resource_path.

5.1.46 Updating ProductSize

PUT

URI	Comments
./product_sizes/{product_size_id}	Updates the specified ProductSize object.

5.1.47 Deleting ProductSize

DELETE

URI	Comments
./product_sizes/{product_size_id}	Deletes the specified ProductSize object.

5.1.48 Attribute list of ProductSize

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
dimension_type	String	enum	R/W	Only in C8 V6.0 rekease. The C8 V5.6.6 requests will assume *Size* as the value when executed on a C8 V6.0 system.
is_two_dim	Boolean	boolean	R/W	Only in C8 V6.0. If true, dimension_1_size and dimension_2_size are required otherwise, they are not allowed.
localized_name	Map	stringmap	R/W	
size_code	Integer	integer	R/W	
us_label	String	string	R/W	
dimension_1_size	String	ref	R/W	Only in C8 V6.0
dimension_2_size	String	ref	R/W	Only in C8 V6.0
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

ProductSource

5.1.49 Creating ProductSource

POST

URI	Comments
./shapes/{shape_id}/product_sources	Create a new ProductSource object as a child of Shape.
./styles/{style_id}/product_sources	Create a new ProductSource object as a child of Style.
./materials/{material_id}/product_sources	Create a new ProductSource object as a child of Material.

5.1.50 Reading ProductSource

GET

URI	Comments
./shapes/{shape_id}/product_sources[?[skip=m][[&]limit=n][&...]]	Retrieve the ProductSource objects referenced by Shape which match filter parameters specified as query string values.
./styles/{style_id}/product_sources[?[skip=m][[&]limit=n][&...]]	Retrieve the ProductSource objects referenced by Style which match filter parameters specified as query string values.
./materials/{material_id}/product_sources[?[skip=m][[&]limit=n][&...]]	Retrieve the ProductSource objects referenced by Material which match filter parameters specified as query string values.
./product_sources/{product_source_id}	Retrieve the specified ProductSource object.
./product_sources[?[skip=m][[&]limit=n][&...]]	Retrieve the ProductSource objects which match filter parameters specified as query string values.

5.1.51 Updating ProductSource

PUT

URI	Comments
./product_sources/{product_source_id}	Update the specified ProductSource object.

5.1.52 Deleting ProductSource

DELETE

URI	Comments
./product_sources/{product_source_id}	Deletes the specified ProductSource object.

5.1.53 Attribute list of ProductSource

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
availability_product	String	ref	R/O	
supplier_code	String	string	R/W	
samples	List	reflist	R/W	
supplier	String	ref	R/W	
supplier_items	List	refset	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

Season

5.1.54 Creating Season

POST

URI	Comments
./seasons	Creates a new Season object.

5.1.55 Reading Seasons

GET

URI	Comments
./seasons/{season_id}	Retrieves the specified Season object.
./seasons[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the Season objects which match the <filter_parameters> included in the resource_path.

5.1.56 Updating Season

PUT

URI	Comments
./seasons/{season_id}	Updates the specified Season object.

5.1.57 Deleting Season

DELETE

URI	Comments
./seasons/{season_id}	Deletes the specified Season object.

5.1.58 Attribute list of Season

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
cnt_colorway	Integer	integer	R/O	Number of Colorways under the Season.
cnt_style	Integer	integer	R/O	Number of Styles under the Season.
hierarchy	List	reflist	R/O	The list of Category 1s' ids that belong to the Season.
issues	List	reflist	R/O	
code	String	string	R/W	The code of the Season.
description	String	string	R/W	The description of the Season.
status	String	enum	R/W	The Season's Status.
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

Session

5.1.59 Creating a new C8 API Session (login)

POST

URI	Comments

./session	Creates a new C8 REST API Session by logging in by using a valid C8 username and password passed. This data is passed as a JSON/XML payload in the HTTP Request object. The data model in JSON format is (see section 3 for the XML example):
	
	{
	"username": "string",
	"password": "string"
	}
	
	On success the API will return a valid session and token in the response object. The JSON data model of the response is:
	
	{
	"session": "string",
	"token": "string"
	}
	
	This session and token should be saved as cookies by your HTTP development framework so that all subsequent REST API calls will pass the required cookies to the C8 server to autnenticate the REST API call.
	
	On failure, the REST API will return error information.

5.1.60 GET, PUT unsupported

GET

PUT

5.1.61 Deleting (invalidating) a user session (logout)

DELETE

URI	Comments
`./session`	Cancels the current user session as identified by the cookie provided authentication tokens. This is used for logging out from REST API session. The REST API will respond with HTTP status 401 to indicate that the user is no longer authorized.

SizeRange

5.1.62 Creating SizeRange

POST

URI	Comments
./size_ranges	Creates a new SizeRange object.

5.1.63 Reading SizeRange

GET

URI	Comments
./size_ranges/{size_range_id}	Retrieves the specified SizeRange object.
./size_ranges[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the SizeRange objects which match the <filter_parameters> included in the resource_path.

5.1.64 Updating SizeRange

PUT

URI	Comments
./size_ranges/{size_range_id}	Updates the specified SizeRange object.

5.1.65 Deleting SizeRange

DELETE

URI	Comments
./size_ranges/{size_range_id}	Deletes the specified SizeRange object.

5.1.66 Attribute list of SizeRange

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
type	String	enum	R/O	
description	String	string	R/W	
dimension_1_type	String	enum	R/W	
dimension_2_type	String	enum	R/W	
is_two_dim	Boolean	boolean	R/W	
ok_for_material	Boolean	boolean	R/W	
ok_for_size_chart	Boolean	boolean	R/W	
ok_for_style	Boolean	boolean	R/W	
active_sizes	List	reflist	R/W	
base_size	String	ref	R/W	
sizes	List	refvector	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

SpecDataSheetSubtype (Read Only)

5.1.67 Reading SpecDataSheetSubtype

GET

URI	Comments
./spec_data_sheet_subtypes/{spec_data_sheet_subtype_id}	Retrieves the specified SpecLibraryItem object.
./spec_library_items[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the SpecLibraryItem objects which match the <filter_parameters> included in the resource_path.

5.1.68 POST, PUT, DELETE unsupported

POST

PUT

DELETE

SpecDataSheetSubtype is a read-only endpoint. It does not support POST, PUT or DELETE methods.

5.1.69 Attribute list of SpecDataSheetSubtype

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/O	Display name for the object.
applied_on	String	time	R/O	
modified	Boolean	boolean	R/O	
modified_at	String	time	R/O	
published	Boolean	boolean	R/O	
setup_applied_on	String	time	R/O	
setup_modified	Boolean	boolean	R/O	
active	Boolean	boolean	R/O	
setup_active	Boolean	boolean	R/O	
setup_validation_rule_group_name	String	string	R/O	

setup_workflow_name	String	string	R/W	
validation_rule_group_name	String	string	R/W	
workflow_name	String	string	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

SpecLibraryItem (Read Only)

5.1.70 Reading SpecLibraryItem

GET

URI	Comments
./spec_library_items/{spec_library_item_id}	Retrieves the specified SpecLibraryItem object.
./spec_library_items[?[skip=m][[&]limit=n][& <filter_parameters>]..]	Retrieves the SpecLibraryItem objects which match the <filter_parameters> included in the resource_path.

5.1.71 POST, PUT, DELETE unsupported

POST

PUT

DELETE

SpecLibraryItem is a read-only endpoint. It does not support POST, PUT or DELETE methods.

5.1.72 Attribute list of SpecLibraryItem

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
sys_id	String	string	R/O	System ID
active	Boolean	boolean	R/W	Active Item
code	String	string	R/W	
description	String	string	R/W	
tags	List	stringvector	R/W	
library_item_sds_subtype	String	ref	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

Style

5.1.73 Creating Style

POST

URI	Comments
./collections/{collection_id}/hierarchy	Creates a new Style object as a child of the specified Collection.

5.1.74 Reading Style

GET

URI	Comments
./collections/{collection_id}/hierarchy[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the Style objects owned by Collection which match the <filter_parameters> included in the resource_path.
./styles/{style_id}	Retrieves the specified Style object.
./styles[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the Style objects which match the <filter_parameters> included in the resource_path.

5.1.75 Updating Style

PUT

URI	Comments
./styles/{style_id}	Updates the specified Style object.

5.1.76 Deleting Style

DELETE

URI	Comments
./styles/{style_id}	Deletes the specified Style object.

5.1.77 Attribute list of Style

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
bom_main_material_count	Integer	integer	R/O	
carry_over	Boolean	boolean	R/O	
cnt_colorway	Integer	integer	R/O	
cnt_customer_purchase_order	Integer	integer	R/O	
cnt_order	Integer	integer	R/O	
cnt_purchased_order	Integer	integer	R/O	
earliest_target_date	String	time	R/O	
inline	Boolean	boolean	R/O	
is_size_owner	Boolean	boolean	R/O	
original_season	String	string	R/O	
total_order_volume	Double	double	R/O	
total_shipment_qty	Double	double	R/O	
bom_main_materials	List	refset	R/O	
bom_materials	List	refset	R/O	
carried_over_from_styles	List	reflist	R/O	
category_1	String	ref	R/O	
category_2	String	ref	R/O	
collection	String	ref	R/O	
issues	List	reflist	R/O	
parent_season	String	ref	R/O	
product_colors	List	reflist	R/O	
product_sizes	List	refvector	R/O	
production_colorways	List	refset	R/O	
realized_products	List	reflist	R/O	
sample_colorways	List	refset	R/O	
active	Boolean	boolean	R/W	
code	String	string	R/W	
description	String	string	R/W	
development_type	String	enum	R/W	
fob_calc	Double	double	R/W	
fob_negotiated	Double	double	R/W	

main_materials	String	string	R/W	
sample_cost	Double	double	R/W	
actual_size_range	String	ref	R/W	
assigned_agents	List	reflist	R/W	
copied_from	String	ref	R/O	
images	Map	refmap	R/O	[6.0] Images associated with this Style
default_color	String	ref	R/W	
default_size	String	ref	R/W	
marketing_main_material	String	ref	R/W	
product_type	String	ref	R/W	
size_set_sample_sizes	List	reflist	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

StyleType (Read Only)

5.1.78 Reading StyleType

GET

URI	Comments
`./style_types/{style_type_id}`	Retrieves the specified StyleType object.
`./style_types[?[skip=m][[&]limit=n][&<filter_parameters>]..]`	Retrieves the StyleType objects which match the <filter_parameters> included in the resource_path.

5.1.79 POST, PUT, DELETE unsupported

POST

PUT

DELETE

StyleType is a read-only endpoint. It does not support POST, PUT or DELETE methods.

5.1.80 Attribute list of StyleType

Attribute	Java Type	C8 Type	Read Only	Description
`id`	String	CNL	R/O	C8 System ID
`modified_at`	String	time	R/O	Last time this object was modified.
`node_name`	String	string	R/O	Display name for the object.
`published`	Boolean	boolean	R/O	Indicates whether the StyleType has been published for production use.
`available`	Boolean	boolean	R/O	Indicates whether the StyleType is active.
`has_color`	Boolean	boolean	R/O	Indicates whether the StyleType allows colors.
`has_size`	Boolean	boolean	R/O	Indicates whether the StyleType allows size.
`links (_links)`	Map	refmap	R/O	C8 Ref to URL Lookup map.

Supplier

5.1.81 Creating Supplier

POST

URI	Comments
./suppliers	Creates a new Supplier object.

5.1.82 Reading Supplier

GET

URI	Comments
./suppliers/{supplier_id}	Retrieves the specified Supplier object.
./suppliers[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the Supplier objects which match the <filter_parameters> included in the resource_path.

5.1.83 Updating Supplier

PUT

URI	Comments
./suppliers/{supplier_id}	Updates the specified Supplier object.

5.1.84 Deleting Supplier

DELETE

URI	Comments
./suppliers/{supplier_id}	Deletes the specified Supplier object.

5.1.85 Attribute list of Supplier

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
actual_lead_time	Integer	integer	R/O	
compliant	Boolean	boolean	R/O	
offline	Boolean	boolean	R/O	
on_time	Double	double	R/O	
po_count	Integer	integer	R/O	
ship_date_variance	Integer	integer	R/O	
agent	String	ref	R/O	
address	String	string	R/W	
address_1	String	string	R/W	
address_2	String	string	R/W	
city	String	string	R/W	
classification	String	enum	R/W	
commission_pct	Double	double	R/W	
company	String	string	R/W	
currency	String	string	R/W	
email	String	string	R/W	
fax	String	string	R/W	
is_agent	Boolean	boolean	R/W	
is_supplier	Boolean	boolean	R/W	
is_warehouse	Boolean	boolean	R/W	
job_title	String	string	R/W	
language	String	string	R/W	
lead_time	String	string	R/W	
localized_address	String	string	R/W	
lock	String	string	R/W	
moq_initial	Integer	integer	R/W	
moq_reorder	Integer	integer	R/W	
mobile	String	string	R/W	
payment_term	String	string	R/W	
phone_number	String	string	R/W	
photocopy_number	String	string	R/W	
postal_code	String	string	R/W	

purpose	String	string	R/W	
state	String	enum	R/W	
supplier_number	String	string	R/W	
supplier_payer_number	String	string	R/W	
telex	String	string	R/W	
trade_term	String	enum	R/W	
type_code	String	string	R/W	
type_desc	String	string	R/W	
website	String	string	R/W	
suppliers	List	reflist	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

SupplierRequest

5.1.86 Creating SupplierRequest

POST

URI	Comments
./supplier_requests	Create a new SupplierRequest object.

5.1.87 Reading SupplierRequest

GET

URI	Comments
./supplier_requests/{supplier_request_id}	Retrieve the specified SupplierRequest object.
./supplier_requests[?[skip=m][[&]limit=n][&...]]	Retrieve the SupplierRequest objects which match filter parameters specified as query string values.

5.1.88 Updating SupplierRequest

PUT

URI	Comments
./supplier_requests/{supplier_request_id}	./supplier_requests/{supplier_request_id}

5.1.89 Deleting SupplierRequest

DELETE

URI	Comments
./supplier_requests/{supplier_request_id}	Delete the specified SupplierRequest object.

5.1.90 Attribute list of SupplierRequest

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/W	Display name for the object.
modified_at	String	time	R/O	
pdf	String	url	R/O	
state	String	enum	R/O	
state_change_time	String	time	R/O	
modified_by	String	ref	R/O	
state_change_user	String	ref	R/O	
due_date	String	time	R/W	
request_type	String	enum	R/W	
submit_quotes_to_wf	Boolean	boolean	R/W	
initiated_by_agent	String	ref	R/W	
requester	String	ref	R/W	
sr_sources	List	reflist	R/W	
sr_suppliers	List	reflist	R/W	
sr_template	String	ref	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.
_url_base_template	String	string	R/O	File (image, pdf, document) retrieval URL template.

SupplierRequestTemplate (Read Only)

5.1.91 Reading SupplierRequestTemplate

GET

URI	Comments
./supplier_request_templates/{supplier_request_template_id}	Retrieve the specified SupplierRequestTemplate object.
./supplier_request_templates[?[skip=m][[&]limit=n][&...]]	Retrieve the SupplierRequestTemplate objects which match filter parameters specified as query string values.

5.1.92 POST, PUT, DELETE unsupported

POST

PUT

DELETE

SupplierRequestTemplate is a read-only endpoint. It does not support POST, PUT or DELETE methods.

5.1.93 Attribute list of SupplierRequestTemplate

Attribute	Java Type	C8 Type	Read Only	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/O	Display name for the object.
modified_at	String	time	R/O	
modified_by	String	ref	R/O	
request_type	String	enum	R/W	
sample_type	String	enum	R/W	
submit_quotes_to_wf	Boolean	boolean	R/W	
srt_quotes	List	reflist	R/W	
srt_suppliers	List	reflist	R/W	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

User (Read Only)

5.1.94 Reading User

GET

URI	Comments
./users/{user_id}	Retrieves the specified User object.
./users[?[skip=m][[&]limit=n][&<filter_parameters>]..]	Retrieves the User objects which match the <filter_parameters> included in the resource_path.

5.1.95 Creating, Updating, Deleting unsupported

POST

PUT

DELETE

User is a read only endpoint. It does not support POST, PUT or DELETE methods.

5.1.96 Attribute list of User

Attribute	Java Type	C8 Type	Read Only*	Description
id	String	CNL	R/O	C8 System ID
modified_at	String	time	R/O	Last time this object was modified.
node_name	String	string	R/O	Display name for the object.
foreign_id	String	string	R/O	
active	Boolean	boolean	R/O	Is User Active
business_phone	String	string	R/O	
city	String	string	R/O	
country	String	string	R/O	
email	String	string	R/O	
fax	String	string	R/O	
first_name	String	string	R/O	
home_phone	String	string	R/O	
last_name	String	string	R/O	
locale	String	string	R/O	
middle_name	String	string	R/O	

mobile_phone	String	string	R/O	
postal_code	String	string	R/O	
province	String	string	R/O	
street_address	String	string	R/O	
user_id	String	string	R/O	
links (_links)	Map	refmap	R/O	C8 Ref to URL Lookup map.

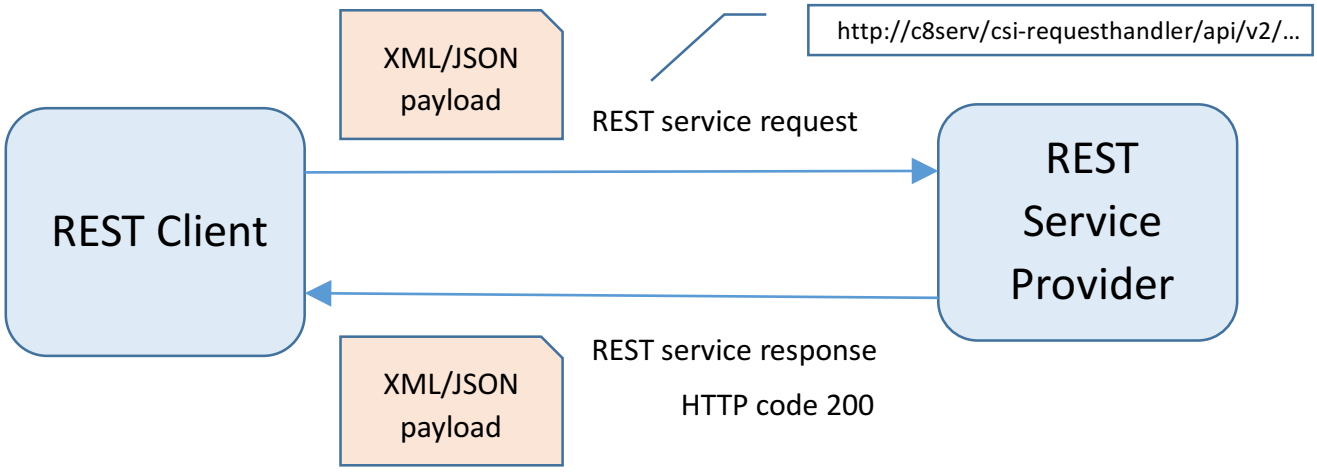
6. LIST OF HTTP STATUS CODES AND MESSAGES

The HTTP standard provides a list of status codes to convey the success or error status of each request. The C8 REST API utilizes the subset of these codes relevant to C8 operation, while preserving full error details in the JSON or XML error response body. The HTTP status codes utilized by the C8 REST API are as follows:

HTTP Status code	Caused by
200	Successful REST API call
201	Successful create
204	Success with no additional content
400	Bad request
401	Not authorized
403	javax.ws.rs.ForbiddenException
404	Not found
405	javax.ws.rs.NotAllowedException
406	javax.ws.rs.NotAcceptableException
415	javax.ws.rs.NotSupportedException
500	Internal server error
503	javax.ws.rs.ServiceUnavailableException

7. Using Centric REST API in Java

We have built a small REST client application to illustrate the important API concepts. The diagram below shows how a REST Client application interacts with a REST Service provider using the HTTP protocol.



The REST Client invokes the REST endpoint (HTTP URI) using appropriate HTTP methods POST, GET, PUT, DELETE and sends the attributes/parameters by loading them in XML or JSON payload or in HTTP header. The REST service provider performs the necessary business transaction for the request and returns the result XML/JSON payload with an appropriate HTTP error code such as 200 (successful HTTP request).

An HTTP client interacts with the REST service provider using HTTP, so the client can be written by using any HTTP capable Internet programming language. In this example, we will use Java to write the client application, however you are free to use any suitable language and framework of your choice such as C++, Python etc.

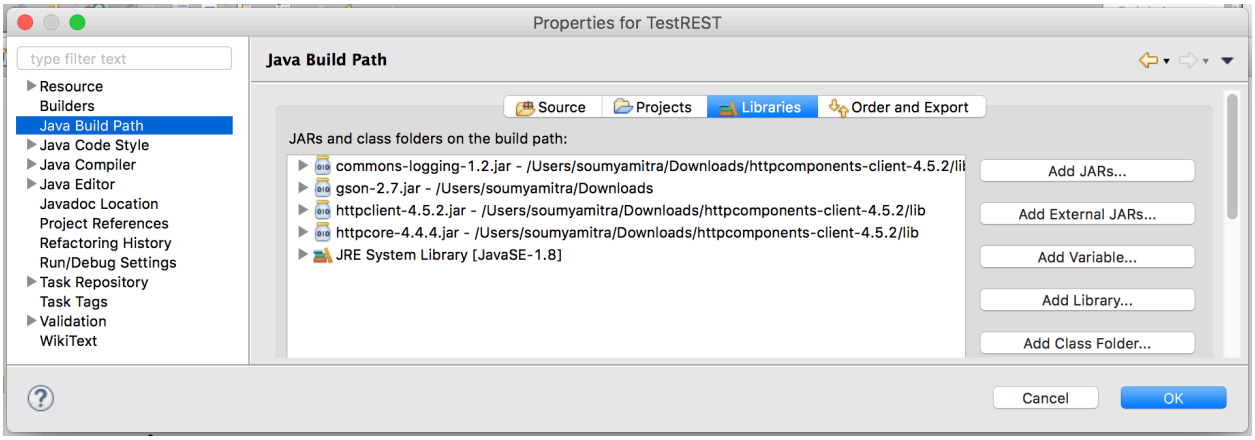
Open source libraries you will need

We have used two open source libraries from Apache Foundation and Google to write this example. It is possible to write an REST client using only out of the box JDK libraries, however using these two open source libraries greatly simplify the communication and cookie management programming so the example code becomes more readable. To try out the example code, you should first download the following open source libraries from Apache and Google Maven repositories and add them to your java IDE library path:

HTTP client and core library from Apache foundation
https://mvnrepository.com/artifact/org.apache.httpcomponents

Gson from Google
https://mvnrepository.com/artifact/com.google.code.gson/gson

The following screenshot shows how I have set the required open source jar files in Eclipse IDE library path



In the following section I will explain the important communication code snippets, without going into line by line explanation of the example code. Please look into the complete source code to see how these code snippets have been used in the actual example.

8. Calling Centric 8 REST API

Use this pattern to call any Centric REST endpoints. In this example, we are invoking the Session endpoint to log on to the Centric 8 server. Without a valid logon, all REST API call will return unauthorized access error. Let us assume your Centric 8 server URI is http://c8server

Step 1

Instantiate a HttpPost/HttpGet/HttpPut/HttpDelete object as appropriate for the REST endpoint.

```
httpPost = new HttpPost(“http://c8server/csi-requesthandler/api/v2/session”);
```

The HttpPost object will call the Session endpoint using Http POST method, HttpGet will call the endpoint with HTTP GET method and so on.

Step 2

Create the JSON payload by initializing a StringEntity with identity information

```
String json = "{\"username\": \"" + username + "\", \"password\": \""
+ password + "\" }";

stringEntity = new StringEntity(json);

stringEntity.setContentType("application/json");
```

Step 3

Load the StringEntity to the HttpPost you have instantiated in step 1

```
httpPost.setEntity(stringEntity);
```

Step 4

Execute the POST call to Session endpoint and retrieve the HTTP response from the Centric 8 server. This response will contain the session token in the JSON payload as well as the HTTP error code

```
CloseableHttpResponse httpResponse;
httpResponse = closableHttpClient.execute(httpPost);
```

Step 5

Parse the HTTPResponse

```
// get the error code
httpResponse.getStatusLine().getStatusCode();
```

```
// get the error message
httpResponse.getStatusLine().getReasonPhrase();

// get the http response headers
httpResponse.getAllHeaders();

// get the entity which contains the JSON or XML return payload
httpResponse.getEntity());
```

STEP 6

Do not forget to close the HTTP response.

```
httpResponse.close();
```

The actual working example code is provided in the following pages.

9. Java Example Source Code

TestJava source code 1/4

```
/*
 * An example java program to illustrate how to use C8 REST API.
 * This is the entry point of the program. It uses a sample C8 REST
API access framework CallREST.java
 *
 * Author:  Soumya Mitra
 * Date:    September 28, 2016
 * Version: 1.0
 *
 * DISCLAIMER: This code is provided as is as an example and
unsupported by
 *             Centric 8 technical support.
 *
 * License:    Creative Commons Attribution-ShareAlike
 */


public class TestREST {

      HttpResultBean resultBean;

      TestREST() {
            // Creating an instance or worker class
            CallREST callrest = new CallREST();
            resultBean = new HttpResultBean();


            try {
                  // ==========   Testing Session Login ================
                        resultBean = callrest.login("<username>",
"<password>" );
                        System.out.println("Headers received from server
for Session POST =====");
                        resultBean.printHeaders();
                        System.out.println("JSON: " +
resultBean.toPrettyFormat() + "\n----------------------\n");
```

```
                // ==========   Testing Season  ================
                    resultBean = callrest.seasons(0, 10);

                    System.out.println("Headers received from server
for Seasons GET =====");
                    resultBean.printHeaders();
                    System.out.println("JSON: " +
resultBean.toPrettyFormatArray() + "\n----------------------\n");

                // ==========   Testing MaterialType  ================
                    resultBean = callrest.materialTypes(0, 10);
                    System.out.println("Headers received from server
for MaterialTypes GET =====");
                    resultBean.printHeaders();
                    System.out.println("JSON: " +
resultBean.toPrettyFormatArray() + "\n----------------------\n");


          // ==========   Testing Session Logout ================
                    resultBean = callrest.logout();
                    System.out.println("Headers received from server
for Session DELETE =====");
                    resultBean.printHeaders();


                // ===========  close the client session  ===========
                    callrest.closableHttpClient.close();

          } catch(Exception e) {   // Catch any exception while
                                   // closing the connection
                e.printStackTrace();
          }
     }

     public static void main(String argv[]){
         new TestREST();
     }


}
```

CallRest source code 2/4

```
/*
 * An example java program to illustrate how to use C8 REST API
 * Uses opensource HTTP client and core library from Apache foundation
https://mvnrepository.com/artifact/org.apache.httpcomponents
 * and Gson from Google
https://mvnrepository.com/artifact/com.google.code.gson/gson
 *
 *
 * Author:  Soumya Mitra
 * Date:    September 28, 2016
 * Version: 1.0
 *
 * DISCLAIMER: This code is provided as is as an example and
unsupported by
 *             Centric 8 technical support.
 *
 * License:    Creative Commons Attribution-ShareAlike
 */



import org.apache.http.impl.client.*;
import org.apache.http.impl.cookie.BasicClientCookie;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.HttpHeaders;
import org.apache.http.client.methods.*;
import org.apache.http.client.protocol.HttpClientContext;
import org.apache.http.entity.*;
import java.io.IOException;
import java.net.CookieHandler;
import java.net.CookieManager;
import java.util.StringTokenizer;


public class CallREST {
      CloseableHttpClient closableHttpClient;
      CloseableHttpResponse httpResponse;
      HttpGet httpGet;
      HttpPost httpPost;
      HttpDelete httpDelete;
      org.apache.http.Header[] httpResponseHeader;
      StringEntity stringEntity;
```

```
	HttpResultBean hrb = new HttpResultBean();
	String[] httpHeaderValues;
	StringTokenizer st;

	BasicHttpContext basicContext;
	BasicCookieStore basicCookieStore;
	BasicClientCookie basicCookie;



	public CallREST() {

		CookieHandler.setDefault(new CookieManager());
		basicContext = new BasicHttpContext();
		basicCookieStore = new BasicCookieStore();
		basicContext.setAttribute(HttpClientContext.COOKIE_STORE,
basicCookieStore);
		closableHttpClient  =
HttpClientBuilder.create().setDefaultCookieStore(basicCookieStore).bui
ld();
	}


/*
 *    ================================================
 *              REST calls to Session endpoint
 *    ==================== START ====================
 */
	// Login
	// Returns HTTP error code and message in HttpResultBean
	public HttpResultBean login(String username, String password)
throws Exception{
		try {
		// Initializing the HttpPost for Session endpoints
			httpPost = new HttpPost(C8Endpoints.SESSION);

		// Initializing a StringEntity with identity JSON object
			String json = "{\"username\": \"" + username + "\",
\"password\": \"" + password + "\" }";
			stringEntity = new StringEntity(json);
			stringEntity.setContentType("application/json");

		// Loading the StringEntiry to the HttpPost request
			httpPost.setEntity(stringEntity);
```

```
            // Executing the POST to Session endpoint for login and
setting session cookie
                  httpResponse = closableHttpClient.execute(httpPost);

            // Loading HttpResultBean with Http Error code and messages
to communicate to caller
                  hrb.setResultBean(httpResponse);
            } catch(Exception e){
                  e.printStackTrace();
            }
            finally{
                  // Closing open httpResponse
                  if(httpResponse != null)
                        httpResponse.close();
            }

            return hrb;
      }


      // Logout
      // Returns HTTP error code and message in HttpResultBean
      public HttpResultBean logout() throws IOException{
            try{
                  // Initializing the HttpDelete for Session endpoints
                  httpDelete = new HttpDelete(C8Endpoints.SESSION);

                  // Executing the DELETE request to the Session
endpoint to logout
                  httpResponse = closableHttpClient.execute(httpDelete);

                  // Loading HttpResultBean with Http Error code and
messages to communicate to caller
                  hrb.setResultBean(httpResponse);
            } catch(Exception e){
                  e.printStackTrace();
            }
            finally{
                  // Closing open httpResponse
                  if(httpResponse != null)
                        httpResponse.close();
                        this.invalidateCookie();
            }
```

```
            // returning the HttpResultBean to the caller
            return hrb;
      }

/*
 *    ================================================
 *              REST calls to Session endpoint
 *    ====================== END =====================
 */



/*
 *    ================================================
 *              REST calls to Season endpoint
 *    ==================== START ====================
 */
      // Get all seasons from Centric 8 Server
      // Returns season array, HTTP error code and message in
HttpResultBean
      public HttpResultBean seasons(int skip, int limit ) throws
Exception{
            try {
                  // Initializing the HttpPost for Season endpoints
                  String uriString = C8Endpoints.SEASON +
"/?skip="+skip+"&limit=" + limit;
                  httpGet = new HttpGet(uriString);

                  // ensure the response will be in JSON format
                  httpGet.addHeader(HttpHeaders.ACCEPT,
"application/json");

                  // Executing the POST to Season endpoint
                  httpResponse = closableHttpClient.execute(httpGet);

                  // Loading HttpResultBean with Http Error code and
messages to communicate to caller
                  hrb.setResultBean(httpResponse);
            } catch(Exception e){
                  e.printStackTrace();
            }
            finally{
                  // Closing open httpResponse
```

```
                if(httpResponse != null)
                      httpResponse.close();
           }
           return hrb;
     }


     /*
      *    ================================================
      *              REST calls to MaterialTypes endpoint
      *    ==================== START ====================
      */
           // Get all MaterialTypes from Centric 8 Server
           // Returns MaterialType array, http error code and message
in HttpResultBean
           public HttpResultBean materialTypes(int skip, int limit )
throws Exception{
                try {
                      // Initializing the HttpGet for MaterialTypes
endpoint
                      String uriString = C8Endpoints.MATERIAL_TYPE +
"/?skip="+skip+"&limit=" + limit;
                      httpGet = new HttpGet(uriString);

                      // ensure the response will be in JSON format
                      httpGet.addHeader(HttpHeaders.ACCEPT,
"application/json");

                      // Executing the GET to MaterialTypes endpoint
                      httpResponse =
closableHttpClient.execute(httpGet);

                      // Loading HttpResultBean with Http Error code
and messages to communicate to caller
                      hrb.setResultBean(httpResponse);
                } catch(Exception e){
                      e.printStackTrace();
                }
                finally{
                      // Closing open httpResponse
                      if(httpResponse != null)
                            httpResponse.close();
                }
                return hrb;
```

```
            }


      public void setCookie(String id, String value) {
            System.out.println("Seeting Cookie: " + id + "   " + value);
            basicCookieStore.addCookie(new
BasicClientCookie(id,value));
      }


      public void invalidateCookie() {
            basicCookieStore.clear();
      }


}
```

HttpResultBean source code 3/4

```
/*
 * An example java program to illustrate how to use C8 REST API
 * Uses opensource HTTP client and core library from Apache foundation
https://mvnrepository.com/artifact/org.apache.httpcomponents
 * and Gson from Google
https://mvnrepository.com/artifact/com.google.code.gson/gson
 *
 * Author:  Soumya Mitra
 * Date:    September 28, 2016
 * Version: 1.0
 *
 * DISCLAIMER: This code is provided as is as an example and not
supported by
 *             Centric 8 technical support.
 *
 * License:    Creative Commons Attribution-ShareAlike
 */


import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.util.EntityUtils;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class HttpResultBean {

   private int httpCode = 0;
   private String httpErrorMessage = "";
   private String[] headerValues = null;
   private Header[] headers = null;
   private HttpEntity entity = null;
   private String stringBody = null;
   private String resultJSON = null;    // Will contain REST API Http
Response JSON Payload

   private void setJSONString(String json) {
```

```
		this.resultJSON = json;
	}

	public void setHttpCode( int i) {
		this.httpCode = i;
	}

	public void setHttpErrorMessage(String s) {
		this.httpErrorMessage = s;
	}

	public void setHeaders(Header[] hdrs){
		if(hdrs != null) {
			headers = hdrs;
			headerValues = new String[hdrs.length];
			for ( int ctr=0; ctr<hdrs.length; ctr++ ) {
				headerValues[ctr] = hdrs[ctr].getValue();
			}
		}
	}

	public void setEntity(HttpEntity httpEntity){
			entity = httpEntity;
			if (entity != null) {
				// A Simple JSON Response Read
				 try {

	   this.setJSONString(EntityUtils.toString(httpEntity));
					} catch (Exception e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					}
			}
	}

	public void setBody(String stringbody) {
		this.stringBody = stringbody;
	}

	public void setResultBean(CloseableHttpResponse
closableHttpResponse) {
		// First initialize all setters to make sure no old values get
cached
		 this.setHeaders(null);
```

```
            this.setHttpCode(0);
            this.setHttpErrorMessage("");
            this.entity = null;
            this.resultJSON = "";

            // Load new values into the bean from HttpResponse

      setHttpCode(closableHttpResponse.getStatusLine().getStatusCode())
;  // getting the HTTP error code

      setHttpErrorMessage(closableHttpResponse.getStatusLine().getReaso
nPhrase());  // getting the error message
            setHeaders(closableHttpResponse.getAllHeaders()); //
getting all http headers
            setEntity( closableHttpResponse.getEntity());
   }

   public String getJSONString() {
         return this.resultJSON;
   }

   public int getHttpCode() {
         return this.httpCode;
   }

   public String getHttpErrorMessage() {
         return this.httpErrorMessage;
   }

   public String[] getHeaderValues(){
         return headerValues;
   }

   public Header[] getHeaders() {
         return headers;
   }

   public HttpEntity getEntity(){
         return entity;
   }

   public String getBody() {
         return this.stringBody;
   }
```

```
	void printHeaders() {
		// Analyzing the returned error code and message
		System.out.println("Http error code: " + this.getHttpCode()
+ " " + this.getHttpErrorMessage());
		System.out.println("Header vales: ");
		if(this.headerValues != null) {
			for(int ctr=0; ctr < this.headerValues.length; ctr++)
{
				System.out.println(this.headerValues[ctr]);
			}
		}
	}


	public String toPrettyFormatArray()   // Utility to print pretty
JSON Array
    {
		JsonParser jsonParser = new JsonParser();
		JsonArray jsonArray = (JsonArray)
jsonParser.parse(this.resultJSON);
		Gson prettyGson = new
GsonBuilder().setPrettyPrinting().create();
		String prettyJson = prettyGson.toJson(jsonArray);
		return prettyJson;
    }

	public String toPrettyFormat()   // Utility to print pretty JSON
object
    {
		JsonParser jsonParser = new JsonParser();
		JsonObject json =
jsonParser.parse(this.resultJSON).getAsJsonObject();
		Gson prettyGson = new
GsonBuilder().setPrettyPrinting().create();
		String prettyJson = prettyGson.toJson(json);
		return prettyJson;
    }

}
```

C8Endpoints source code 4/4

```
/*
 * This class contains static variables defining URI of C8 endpoints.
Modify it to point to your Centric 8 server and endpoints
 *
 * Author:  Soumya Mitra
 * Date:    September 28, 2016
 * Version: 1.0
 *
 * DISCLAIMER: This code is provided as is as an example and
unsupported by
 *             Centric 8 technical support.
 *
 * License:    Creative Commons Attribution-ShareAlike
 */


public class C8Endpoints {
      public static String C8URI = "http://<c8server-and-port>/csi-
requesthandler/api/v2/";  // replace with your C8 server URI
      public static String SESSION = C8URI + "session";
      public static String SEASON = C8URI + "seasons";
      public static String MATERIAL_TYPE = C8URI + "material_types";
      public static String COLORWAY = C8URI + "colorway";
}
```

10. Known Issues

REST API references are not type safe:
Many Centric 8 REST endpoints will return references to other Centric objects, however, Centric 8 server does not check for node/attribute type safety. So it is possible to reference a node or attribute with a wrong type. This happens, the REST API throws a bad request exception.

CentricSoftware™

Centric Software, Inc.

Centric Software, Inc.
655 Campbell Technology Parkway,
Suite 200, Campbell
CA 95008
Phone: 1.408.574.7802

Technical Support

866. 796-6218 8:00 AM-8:00 PM EST
Email: support@centricsoftware.com

Documentation Feedback

Email: documentation@centricsoftware.com