Working with REST-based APIs

Version 9.8

April 2015

This document applies to CentraSite Version 9.8 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: CS-REST-98-20150415

Table of Contents

About this Guide

This guide describes how to create REST-based API and how to manage the REST APIs in the CentraSite Business UI.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

1 About CentraSite RESTful API Model

Overview

CentraSite's REST framework enables you to model APIs conforming to the (Resource Oriented Architecture) ROA design. For example, you might model an API that serves to expose the web service data and functionality as a collection of resources. Each resource will be accessible with unique Uniform Resource Identifiers (URLs). In your API, you expose a set of HTTP operations (methods) to perform on a specific resource, and capture the request and response messages, and status codes that will be unique to the HTTP method and linked within the specific resource of the API.

The basic elements of a REST API in the CentraSite registry are as follows:

- The API itself (For example, *phonestore*)
- Its resource (*phones*), available on the unique base URL (*/phones*)
- The defined HTTP method (*GET*) for accessing the resource (*phones*)
- Parameters for request representations (*412456*)
- A request generated for this method (*Request 123*)
- A response with the status code received for this request (*Response ABCD*)

Instructions throughout the this guide use the term *API* when referring to the REST APIs.

REST APIs

In CentraSite, you document a REST API as an asset instance of the type REST Service or Virtual REST Service.

These APIs are typically collections of resources.

For example, consider an API that is defined to support an online phone store application. Assume, this sample Phone Store API currently has a database that defines the various brands of phones, features in the individual phones and the inventory of each phone.

The Phone Store API is used as a sample to illustrate how to model URL patterns for resources, resource methods, HTTP headers and response codes, content types, and parameters for request representations to resources.

Base URL

The base URL of an API is constructed by the domain, port and context mappings of the API. For example, if the server name is www.phonestore.com, port is 8080, and the API context is api. The full Base URL is:

```
http://www.phonestore.com:8080/api
```

API Parameters

Parameters defined at the higher API level are inherited by all Resources, and by all Methods included in the individual Resources.

CentraSite permits different types of parameters at the API level. For more information about these parameters, see "REST Parameters" on page 14.

REST Resources

Resources are the basic components of an API. Examples of resources from an online Phone Store API include a phone, an order from a store, and a collection of customers.

After you identify a service to expose as an API, you define the resources for the API. CentraSite's flexible metadata store captures the relationships of APIs with resources, and ensures that the APIs are available in the right way.

For example, consider the case of an online Phone Store API. In this example, there are a number of ways to represent the data in the phone store database as an API. The verbs in the HTTP request maps to the operations that the database supports, such as select, create, update, delete.

Each resource needs to be addressable by a unique URI. Along with the URI you're going to expose for each resource, you also need to decide what can be done to each resource. The HTTP methods passed as part of an HTTP request header direct the API what needs to be done with the addressed resource.

Resource URLs

An URL identifies the location of a specific resource.

For example, consider the case of our sample Phone Store API designed to support an online phone store application. The resources will have the following URLs:

URL	Description
`http://www.phonestore.com/api/phones`	Specifies the collection of phones contained in the online store.
`http://www.phonestore.com/api/phones/412456`	Accesses a phone referenced by the product code `412456`.
`http://www.phonestore.com/api/phones/412456/reviews`	Specifies a set of reviews posted for a phone of code `412456`.

URL	Description
`http://www.phonestore.com/api/phones/412456/reviews/78`	Accesses a specific review referenced by the unique ID `78` contained in the reviews of the phone of code `412456`.

CentraSite supports the following patterns of resource URL: a collection of resources or a particular resource.

For example, consider the above example of an online Phone Store API.

Collection URL

`http://phonestore.com/api/phones`

Unique URL

`http://phonestore.com/api/phones/412456`

Collection URL

`http://phonestore.com/api/phones/412456/features`

to retrieve a collection resource describing the key features of phone whose product code is `412456`

Resource Parameters

Parameters defined at the higher Resource level are inherited by all Methods in the particular resource; it does not affect the API.

CentraSite permits different types of parameters at the Resource level. For more information about these parameters, see "REST Parameters" on page 14.

Resource Methods

Individual resources can define their capabilities using supported HTTP methods. To invoke an API, the client would call an HTTP operation on the URL associated with the API's resource. For example, to retrieve the key feature information for phone whose product code is `412456`, the client would make a service call HTTP GET on the following URL:

`http://www.phonestore.com/phones/412456/features`

Supported HTTP Methods

CentraSite supports the standard HTTP methods for modeling APIs: GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH, TRACE, and CONNECT.

Important: During virtualization of an API, CentraSite does not support the following HTTP methods: HEAD, OPTIONS, PATCH, TRACE, and CONNECT. This is because, when a virtual API is published to Mediator, at run-time, CentraSite only supports GET, POST, PUT, DELETE.

The following table describes the semantics of HTTP methods for our sample Phone Store API.

Resource URI	HTTP Method	Description
`/phones/orders`	GET	Asks for a representation of all of the orders.
`/phones/orders`	POST	Attempts to create a new order, returning the location (in the Location HTTP Header) of the newly created resource.
`/phones/orders/{order-id}`	GET	Asks for a representation of a specific Order resource.
`/phones/orders/{order-id}`	DELETE	Requests the deletion of a specified Order resource.
`/phones/orders/{order-id}/status`	GET	Asks for a representation of a specific Order's current status.
`/phones/orders/{order-id}/paymentdetails`	GET	Asks for a representation of a specific Order's payment details.
`/phones/orders/{order-id}/paymentdetails`	PUT	Updates a specific Order's payment details.

For information on the HTTP methods that CentraSite ships, in CentraSite Control, go to **Administration > Taxonomies**. On the Taxonomies page, enable the **Show all Taxonomies** option. Navigate to **HTTP Methods** in the list of taxonomies.

Method Parameters

Parameters defined at the lower Method level apply only to that particular method; it does not affect either the API or the Resource.

CentraSite permits different types of parameters at the Method level. For more information about these parameters, see "REST Parameters" on page 14.

REST Parameters

Parameters specify additional information to a request. You use parameters as part of the URL or in the headers or as components of a message body.

Parameter Levels

A parameter can be set at different levels of an API. When you document a REST API in CentraSite, you define parameters at the API level or Resource level or Method level to address the following scenarios:

- If you have the parameter applicable to all resources in the API, then you define this parameter at the API level. This indirectly implies that the parameter is propagated to all resources and methods under the particular API.
- If you have the parameter applicable to all methods in the API, then you define this parameter at the Resource level. This indirectly implies that the parameter is propagated to all methods under the particular resource.
- If you have the parameter applicable only to a method in the API, then you define this parameter at the Method level.

API-Level Parameters

Setting parameters at the API level enables the automatic assignment of the parameters to all resources and methods included in the API. Any parameter value you specify at the higher API level overrides the parameter value you set at the lower Resource level or the lower Method level if the parameter names are the same.

For example, say you have a header parameter called API Key that is used for consuming an API.

```
x-CentraSite-APIKey:a4b5d569-2450-11e3-b3fc-b5a70ab4288a
```

This parameter is specific to the entire API and to the individual components - resources and methods directly below the API. Such a parameter can be defined as a parameter at the API level.

At an API level, CentraSite allows you to define the following types of parameters:

- Query-String parameter
- Header parameter
- Form parameter

Resource-Level Parameters

Setting parameters at the Resource level enables the automatic assignment of the parameters to all methods within the resource. Any parameter value you specify at the higher Resource level overrides the parameter value you set at the lower Method level if the parameter names are the same. In contrast, the lower Resource level parameters will not affect the higher API level parameters.

Consider our sample Phone Store API maintains a database of reviews about different phones. Here is a request to display information about a particular user review, 78 of the phone whose product code is `412456`.

```
GET /phones/412456/user_reviews/78
```

In the example, `/user_reviews/78` parameter narrows the focus of a GET request to review `/78` within a particular resource `/412456`.

This parameter is specific to the particular resource phone whose product code is `412456` and to any individual methods that are directly below the particular resource. Such a parameter can be defined as a parameter at the Resource level.

At a Resource level, CentraSite allows you to define the following types of parameters:

- Path parameter
- Query-String parameter
- Header parameter
- Form parameter

Method-Level Parameters

If you do not set parameters at the API level or Resource level, you can set them at a Method level. Parameters you set at the Method level are used for the HTTP method execution. They are useful to restrict the response data returned for a HTTP request. Any parameter value you specify at the lower Method level is overridden by the value set at higher API level parameter or the higher Resource level parameter if the names are the same. In contrast, the lower Method level parameters will not affect the higher API level or Resource level parameters.

For example, the Phone Store API described might have a request to display information contributed by user `Allen` in `2013` about a phone whose product code is `412456`.

```
GET /phones/412456/user_reviews/78?year=2013&name=Allen
```

In this example, `year=2013` and `name=Allen` narrow the focus of the GET request to entries that user `Allen` added to user review `78` in `2013`.

At a Method level, CentraSite allows you to define the following types of parameters:

- Query-String parameter
- Header parameter
- Form parameter

Parameter Types

CentraSite supports the following types of parameters:

Query-String Parameters

Query-String parameters are appended to the URI after a `?` with name-value pairs. The name-value pairs sequence is separated by either a semicolon or an ampersand.

For instance, if the URL is `http://phonestore.com/api/phones?itemID=itemIDValue`, the query parameter name is `itemID` and value is the `itemIDValue`. Query parameters are often used when filtering or paging through HTTP GET requests.

Now, consider the online Phone Store API. A customer, when trying to fetch a collection of phones, may wish to add options, such as `android v4.3 OS` and `8MP` camera. The URI for this resource could look like this:

```
/phones?features=androidosv4.3&cameraresolution=8MP
```

Path Parameters

Path parameters are defined as part of the resource URI. For example, the URI can include `phones/item`, where `/item` is a path parameter that identifies the item in the collection of resource `/phones`. Because path parameters are part of the URI, they are essential in identifying the request.

Now, consider the above online Phone Store API example. A customer may wish to fetch details about a phone `{phone-id}` whose product code is "412456". The URI for this resource could look like this:

```
/phones/412456
```

Important: As a best practice, we recommend that you adopt the following conventions when specifying a path parameter in the resource URI:

- Append a path parameter variable within curly `{}` brackets.
- Specify a path parameter variable such that it exactly matches the path parameter defined at the Resource level.

Header Parameters

Header parameters are HTTP headers. Headers often contain metadata information for the client, or server.

```
x-CentraSite-APIKey:a4b5d569-2450-11e3-b3fc-b5a70ab4288a
```

You can create custom headers, as needed. As a best practice, we recommend that you prefix the header name with `X-`.

HTTP/1.1 defines the headers that can appear in a HTTP response in three sections of RFC 2616: 4.5, 6.2, and 7.1. Examine these codes to determine which are appropriate for the API.

Form Parameters

Form parameters and values are encoded in the request message body, in the format specified by the content type (application/x-www-form-urlencoded).

```
features=androidosv4.3&cameraresolution=8MP
```

Important: Although CentraSite allows you to define parameters of the type "Form" at the API level, these parameters are not supported at run-time.

Parameter Data Types

When you add a parameter to the API, you specify the parameter's data type. The data type determines what kind of information the parameter can hold.

CentraSite supports the following data types for parameters.

Data Type	Description
String	Specifies a string of text.
URL	Holds a URL/URI. This type of parameter only accepts values in the form: `protocol://host:port/path` Where: ■ *protocol* is any protocol that java.net.URL supports. ■ *host* is the name or IP address of a host machine. ■ *port* is the port on which the host machine is listening. ■ *path* (optional) is the path to the requested resource on the specified host.
Boolean	Specifies a `true` or `false` value.
Email	Specifies an email address. This data type only accepts values in the format: `anyString@anyString`
Number	Specifies a numeric value.

Data Type	Description
Duration	Specifies a value that represents a period of time as expressed in years, months, days, hours, minutes and seconds. This duration is specified using an `xs:duration` format. It specifies a duration in terms of years (either 0 or 1), months, days, hours, minutes and seconds.
Date/Time	Specifies a timestamp that represents a specific date and/or time. The date/time input parameters allow year, month and day input as well as hour and minute. Hour and minute default to 0. This data type only accepts date values in the format `yyyy-mm-dd`; and time values in the format `hh:mm:ss`.
IP Address	Specifies a numeric IP address in the v4 or v6 format.

Supported Content Types

Clients can optionally specify the content-type format they want the response to use.

CentraSite includes a set of predefined content types that are classified in the following taxonomy categories:

Predefined Taxonomy Category	Description
Applications	An **application** content-type value to transmit API data or binary data, and hence, among other uses, to implement an electronic mail file transfer service. Example - application/xml - application/json
Audio Files	An **audio** content-type value for transmitting audio or voice data. Example - audio/basic - audio/mp4

Predefined Taxonomy Category	Description
Image Files	An **image** content-type value, for transmitting still image (picture) data. Example - image/gif - image/png
Text Files	A **text** content-type value to represent textual information in a number of character sets and formatted text description languages in a standardized manner. Example - text/html - text/plain
Video Files	A **video** content-type value, for transmitting video or moving image data, possibly with audio as part of the composite video data format. Example - video/mpeg

For information on the content types that CentraSite ships, in CentraSite Control, go to **Administration > Taxonomies**. On the Taxonomies page, enable the **Show all Taxonomies** option. Navigate to **Content Types** in the list of taxonomies.

If you would like to use content types that are not provided by CentraSite, you can define your custom content types. For more information about creating custom content types, see the *CentraSite Administrator's Guide*. In addition, if you are using webMethods Mediator, and would like to extend the content types, you can define custom content types as described in *Administering webMethods Mediator*.

Supported HTTP Status Codes

An API response returns a HTTP status code that indicates success or failure of the requested operation.

CentraSite allows you specify HTTP codes for each method, to help clients understand the response. While responses can contain an error code in XML or other format, clients can quickly and more easily understand an HTTP response status code. The HTTP specification defines several status codes that are typically understood by clients.

CentraSite includes a set of predefined content types that are classified in the following taxonomy categories:

Predefined Taxonomy Category	Description
1xx	Informational.
2xx	Success.
3xx	Redirection. Need further action.
4xx	Client error. Correct the request data and retry.
5xx	Server error.

For information on the status codes that CentraSite supports out-of-the-box, in CentraSite Control, go to **Administration > Taxonomies**. On the Taxonomies page, enable the **Show all Taxonomies** option. Navigate to **HTTP Status Codes** in the list of taxonomies.

HTTP/1.1 defines all the legal status codes. Examine these codes to determine which are appropriate for your API.

Consider the case of online Phone Store API. The following table describes the HTTP status codes that each of the URIs and HTTP methods combinations will respond.

Resource URI	Supported HTTP Methods	Supported HTTP Status Codes
`/phones/orders`	GET	200 (OK, Success)
`/phones/orders`	POST	201 (Created) if the Order resource is successfully created, in addition to a Location header that contains the link to the newly created Order resource; 406 (Not Acceptable) if the format of the incoming data for the new resource is not valid
`/phones/orders/{order-id}`	GET	200 (OK); 404 (Not Found) if Order Resource not found
`/phones/orders/{order-id}`	DELETE	204 (OK); 404 (Not Found) if Order Resource not found

Resource URI	Supported HTTP Methods	Supported HTTP Status Codes
`/phones/orders/{order-id}/status`	GET	200 (OK); 404 (Not Found) if Order Resource not found
`/phones/orders/{order-id}/paymentdetails`	GET	200 (OK); 404 (Not Found) if Order Resource not found
`/phones/orders/{order-id}/paymentdetails`	PUT	201 (Created); 406 (Not Acceptable) if there is a problem with the format of the incoming data on the new payment details; 404 (Not Found) if Order Resource not found

Sample Requests and Responses

To illustrate the usage of an API, you provide sample request and response messages. Consider the sample Phone Store API that maintains a database of phones in different brands. The Phone Store API might provide the following examples to illustrate its usage.

Sample 1 - Retrieve a list of phones

Client Request

```
GET /phones HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.api.phonestore.com
Accept-Language: en-us
Accept-Encoding: text/xml
Connection: Keep-Alive
```

Server Response

```
HTTP/1.1 200 OK
Date: Mon, 14 July 11:53:27 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 18 June 2014 09:18:16 GMT
Content-Length: 356
Content-Type: text/xml
<phones>
   <phone>
       <name>Asha</name>
       <brand>Nokia</brand>
       <price currency="irs">11499</price>
       <features>
           <camera>
               <back>3</back>
```

```
            </camera>
            <memory>
                <storage scale="gb">8</storage>
                <ram scale="gb">1</ram>
            </memory>
            <network>
                <gsm>850/900/1800/1900 MHz</gsm>
            </network>
        </features>
    </phone>
    <phone>
        <name>Nexus7</name>
        <brand>Google</brand>
        <price currency="irs">16499</price>
        <features>
            <camera>
                <front>1.3</front>
                <back>5</back>
            </camera>
            <memory>
                <storage scale="gb">16</storage>
                <ram scale="gb">2</ram>
            </memory>
            <network>
                <gsm>850/900/1800/1900 MHz</gsm>
                <HSPA>850/900/1900 MHz</HSPA>
            </network>
        </features>
    </phone>
</phones>
```

Sample 2 - Find a phone that doesn't exist

Client Request

```
GET /phones/phone-4156 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.api.phonestore.com
Accept-Language: en-us
Accept-Encoding: text/xml
Connection: Keep-Alive
```

Server Response

```
HTTP/1.1 404 Not Found
Date: Mon, 14 July 11:53:27 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 18 June 2014 09:18:16 GMT
Content-Type: text/xml
```

Sample 3 - Create a phone

Client Request

```
POST /phones/phone HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.api.phonestore.com
Accept-Language: en-us
Accept-Encoding: text/xml
Content-Length: 156
Connection: Keep-Alive
<phones>
```

```
   <phone>
       <name>iPhone5</name>
       <brand>Apple</brand>
       <price currency="irs">24500</price>
       <features>
           <camera>
               <front>1.2</front>
               <back>8</back>
           </camera>
           <memory>
               <storage scale="gb">32</storage>
               <ram scale="gb">2</ram>
           </memory>
           <network>
               <gsm>850/900/1800/1900 MHz</gsm>
               <HSPA>850/900/1900 MHz</HSPA>
           </network>
       </features>
   <phone>
</phones>
```

Server Response

```
HTTP/1.1 200 OK
Date: Mon, 14 July 11:53:27 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 18 June 2014 09:18:16 GMT
Content-Type: text/xml
Content-Length: 15
<id>2122</id>
```

2 Importing a RESTful API to CentraSite

Overview

An importer in CentraSite is a utility that takes in the specification file of a particular format as the input, and then creates an asset of the particular metadata format in the registry. For example, the CentraSite importer for RAML reads a RAML specification and from it, creates a REST Service asset that best describes the RESTful API with RAML specification. The importer also uploads the input file to the CentraSite repository and links the file to the REST API. When you import a REST API using a RAML file, for example, the importer copies the RAML file into the repository and then links the file to the REST API.

CentraSite supports the RAML and Swagger REST metadata formats. The table also identifies the types of files they require as input.

To import this format of REST API...	You must supply this type of file...
RAML	RESTful API Modeling Language (RAML) file. - OR - Yet Another Markup Language (YAML) file.
Swagger	JavaScript Object Notation (JSON) file. - OR - Yet Another Markup Language (YAML) file.

Importing RAML Files

The CentraSite importer for RAML copies a RAML file to the repository, and creates a REST Service asset that best describes the RESTful API representing the RAML specification.

The registry objects are as follows:

- A REST Service asset (REST API) with RAML specification. This RAML specification is stored as a file attribute in the **Documents** field of the **Specification** profile.
- A REST endpoint URI object that refers to the REST API.
- A REST Resource object that refers to the REST API.
- A REST Method object that refers to the REST API.
- A REST Parameter object that is referred to by the one of the above objects.

- A REST Request Content-Type (REST Payload) object that is referred to by the REST API or REST Method object.
- A REST Response Content-Type (REST Payload) object that is referred to by the REST API or REST Method object.
- A REST Status Code object that is referred to by the Response Content-Type object.

RAML to CentraSite REST API Mappings

This section describes how the RAML objects and properties (as described in the RAML specification) are mapped in the CentraSite REST data model. It shows the relationships between the RAML root section, parameters, body fields, and their REST data model. It also lists the relationships between RAML resources, methods, reference documentation, and security schemes, and how these values are mapped in REST API.

Note: CentraSite supports RAML version 0.8.

The following tables list the correspondences (mappings) between the fields of a RESTful API and RAML document.

RAML Root Section Fields

The root section of the RAML definition describes the basic information of a RESTful API, such as its title, version, base URI, default media types and common schema references.

RAML Root Section Field	REST API Field	Notes
`RAML Document`	REST Service	The `RAML Document` property is represented by an asset of type REST Service in the CentraSite registry. The `RAML Document` is stored as a file attribute in the **Documents** field of the **Specification** profile in the REST API's details page. One or more RAML documents can be attached to a REST API. The attached documents are stored in the asset-specific WebDAV folder

RAML Root Section Field	REST API Field	Notes
		like the schema and WSDL documents.
`API Title`	Name	The `API Title` property is represented by the **Name** field in the REST API's details page.
`API Version`	Version	`API Version` is represented by the **Version** field in the REST API's details page.
`Base URI`	Base URL	The `Base URI` property is represented by the **Base URL** field in the **Technical Details** profile.
`baseUri Parameters`		Currently, CentraSite does not support mapping `URI Parameter` property at the Base URL level.

`Protocols`	Base URLs	In CentraSite, the `Protocols` property is represented by the multiple **Base URL** fields of the **Technical Details** profile.
		Example:
		A RAML Specification:
		`baseUri:`
		`http://products.api.apievangelist.com protocols:[HTTP, HTTPS]`
		Reads in CentraSite REST Model:

RAML Root Section Field	REST API Field	Notes
		`http://products.api.apievangelist.com` `https://products.api.apievangelist.com`
`Default Media Type`	Request Content Type Response Content Type	In CentraSite, the `Default Media Type` property is represented by the **Request Content-Type** and **Response Content-Type** fields of the **Technical Details** in the REST API's details page.
`Schemas`	Schema	The `Schemas` property is represented by the Request **Schema** or Response **Schema** field of a REST Method.
`URI Parameters (for baseUri)`		Currently, CentraSite does not support the `URI Parameter` property at the Base URL level.
`User Documentation`	Description	The `User Documentation` property is represented by the **Description** field in the REST API's details page.

RAML Parameter Fields

The following tables show the mappings between various RAML parameter types, data types, and attributes and how they are used by REST API. The mappings are grouped into tables for RAML Parameter Types, RAML Parameter Data Types, and RAML Parameter Attributes.

RAML Parameter Types

RAML Parameter Field	REST API Field	Notes
`URI Parameter`	Parameter of type "Path"	The `URI Parameter` property is represented by the "Path" option in the **Parameter Type** field of a REST Parameter, and is specifically defined at the Resource level. Any `URI Parameter` defined at the API level or Method level is not mapped in CentraSite.
`Query Parameter`	Parameter of type "Query-String"	The `QueryParameter` property is represented by the "Query-String" option in the **Parameter Type** field of a REST Parameter.
`Header`	Parameter of type "Header"	The `Header` property is represented by the "Header" option in the **Parameter Type** field of a REST Parameter.
`Form Parameter`	Parameter of type "Form"	The `Form Parameter` property is represented by the "Form" option in the **Parameter Type** field of a REST Parameter.

RAML Parameter Data Types

RAML Data Type Field	REST API Field	Notes
`string`	String	The `string` property is represented by the String option in the **Data Type** field of a REST Parameter.

RAML Data Type Field	REST API Field	Notes
`number`	Number	The `number` property is represented by the Number option in the **Data Type** field of a REST Parameter.
`integer`	Number	The `integer` property is represented by the Number option in the **Data Type** field of a REST Parameter.
`date`	Date/Time	The `date` property is represented by the Date/Time option in the **Data Type** field of a REST Parameter.
`boolean`	Boolean	The `boolean` property is represented by the Boolean option in the **Data Type** field of a REST Parameter.
`file`	URL/URI	Currently, CentraSite does not support parameter of the data type `file`.

RAML Parameter Attributes

RAML Attribute Field	REST API Field	Notes
`enum`	Possible Values	The `enum` property is represented by the **Possible Values** field of a REST Parameter.
`required`	Required	The `required` property is represented by the

RAML Attribute Field	REST API Field	Notes
		Required field of a REST Parameter.
`default`	Default Value	The `default` property is represented by the **Default Value** field of a REST Parameter.
`pattern`		Currently, CentraSite does not support the `pattern` attribute.
`minLength`		Currently, CentraSite does not support the `minLength` attribute.
`maxLength`		Currently, CentraSite does not support the `maxLength` attribute.
`minimum`		Currently, CentraSite does not support the `minimum` attribute.
`maximum`		Currently, CentraSite does not support the `maximum` attribute.
`example`		Currently, CentraSite does not support the `example` attribute.
`repeat`		The `repeat` property is represented by the **Array** field of a REST Parameter.

RAML Resource Fields

RAML resource properties (data) are mapped to the REST API resource objects. The following table shows the resource mappings between a RAML definition and a REST API:

RAML Resource Field	REST API Field	Notes
`Resource`	REST Resource	The `Resource` property is represented by an asset of type REST Resource in the CentraSite registry.
`Resource Relative Path`	Resource Path	The `Resource Relative Path` property is represented by the **Resource Path** field of a REST Resource.
`Display Name`	Name	The `Display Name` property is represented by the **Name** field of a REST Resource.
`Description`	Description	The `Description` property is represented by the **Description** field of a REST Resource.
`Template URIs`	Resource Path with Path parameter	The `Template URI` property is represented by the **Resource Path** field which has the "Path" parameter appended to it, and is defined as a REST Resource object.
`URI Parameters`	Parameter of type "Path"	The `URI Parameter` property is defined as a REST Parameter object of the type "Path" at the REST Resource level.
`Base URI Parameters`		Currently, CentraSite does not support mapping `URI Parameter` property at the Base URL level.

RAML Resource Field	REST API Field	Notes
`Resource Types and Traits`		**Limitation:** `Resource Types and Traits` are not explicitly mapped to the REST API; instead, propagated to the referencing REST Resources and REST Methods.

RAML Method Fields

The following tables show the mappings between various RAML method objects, bodies and responses, and how they are used by REST API. The mappings are grouped into tables for RAML Method Headers, RAML Method Body, and RAML Method Responses.

RAML Method Headers

RAML Method Header Field	REST API Field	Notes
`Method`	REST Method	The `Method` property is represented by an asset of type REST Method in the CentraSite registry.
`Description`	Description	`Description` property is represented by the **Description** field of a REST Method.
`Header`	Parameter of type "Header"	The `Header` property is defined as a REST Parameter object of the type "Header" at the REST Resource level.
`Protocols`		Currently, CentraSite does not support mapping `Protocol` property at the REST Method level.

RAML Method Header Field	REST API Field	Notes
`Query Parameters`	Parameter of type "Query-String"	The `QueryParameters` property is defined as a REST Parameter object of the type "Query-String" at the REST Resource level.

RAML Method Body

RAML Method Body Field	REST API Field	Notes
`Body`	Request	The `Body` property is defined as the **Request Payload** object. CentraSite automatically populates the `Media type` as **Name** of a REST Payload object.
`Media type`	Request Content-Type Response Content-Type	The `Media type` property is represented by the **Request Content-Type** or **Response Content-Type** field of a REST Payload object.
`Schema`	Schema	The `Schema` property is represented by the Request or Response **Schema** field of a REST Payload object.
`Example`	Example	The `Example` property is represented by the Request or Response **Example** field of a REST Payload object.

RAML Method Responses

RAML Method Response Field	REST API Field	Notes
`HTTP status code`	REST Payload	The `HTTP status code` property is defined as the **REST Payload** object of a REST Method. CentraSite automatically populates the `HTTP status code` as the **Name** of a REST Payload object.
`Example`	Response Example	The `Example` property is represented by the Response **Example** field of a REST Payload object.
`Description`	Description	The `Description` property is represented by the Response **Description** field of a REST Payload object.
`Body`	Response Content-Type	The `Body` property is represented by the **Response Content-Type** field of a REST Payload object.
`Body media type`	Response Content-Type	The `Body Media type` property is represented by the **Response Content-Type** field of the REST Payload.

RAML Resource Type and Trait Fields

All RAML resource types and traits properties are not explicitly mapped to the REST API. Instead, the RAML resource types are propagated to the referencing document structure named REST Resource and RAML traits are propagated to the document structure named REST Method.

RAML Security Scheme Fields

RAML security schemes are not explicitly mapped to the REST API. Instead, the RAML security schemes are propagated to the taxonomy structure named **Security Types**, and represented by the **Supported Access Token Types** property defined in the **API-Portal Information** profile.

Security Type	Description
API Key	The API's authentication requires using an API key.
Basic Authentication	The API's authentication requires using Basic Access Authentication as described in RFC2617.
Digest Authentication	The API's authentication requires using Digest Access Authentication as described in RFC2617.
OAuth 1.0	The API's authentication requires using OAuth 1.0 as described in RFC5849.
OAuth 2.0	The API's authentication requires using OAuth 2.0 as described in RFC6749.
x-{other}	The API's authentication requires using another authentication method.

The supported security schemes along with their scheme reference name are mapped in the CentraSite registry. For more information about the security scheme, see the RAML specification.

Importing a REST API Using RAML File

Before you begin, you must have the RAML file that you want to import.

To import a new REST API with RAML definition

1. In the activity bar, click **Create Asset**. This opens the **Create Asset** wizard.
2. In the **Basic Information** profile, enter the following information in the fields provided:

In this field...	Do the following...
Name	*Optional.* Enter a name for the new REST API. This name must be NCName-conformant, meaning that:

In this field...	Do the following...
	■ The name must begin with a letter or the underscore character (_). ■ The remainder of the name can contain any combination of letters, digits, or the following characters: . - _ (i.e., period, dash, or underscore). It can also contain combining characters and extender characters (e.g., diacriticals). ■ The name cannot contain any spaces. ■ Furthermore, if the name contains any non-conformant character, then that character is simply replaced with the underscore character (_). For more information about the NCName type, see http://www.w3.org/TR/xmlschema-2/#NCName **Note:** The API name does not need to be unique. However, to reduce ambiguity, we recommend that you adopt appropriate naming conventions to ensure that API is distinctly named within an organization. If you have not specified a **Name** for the REST API, CentraSite automatically maps this field with the `API Title` property, which is defined in the RAML specification.
Type	Choose **REST Service** asset type.
Organization	Select the organization in which the new API will be created. (The drop-down list displays the list of organizations in which you are permitted to create APIs.)
Version	*Optional.* Specify a version identifier for the new API. If you have not specified a **Version** for the REST API, CentraSite automatically maps this field with the `API Version` property, which is defined in the RAML specification.
Description	*Optional.* Enter a description for the new API. If you have not specified a **Description** for the REST API, CentraSite automatically maps this field with the `Description` property, which is defined in the RAML specification. **Note:** This is the description information that users will see when they view instances of this type in the

In this field...	Do the following...
	user interface, therefore, the description should be meaningful.
Import From Specification File	Choose **RAML-0.8** as the specification file type.
URL or **File**	Specify whether the input file will be read from your local file system (the **File** option) or from a URL-addressable location on the network (the **URL** option). If the file you are importing resides on the network, specify its URL. If the file resides in your local file system, specify the file name. You can use the **Browse** button to navigate to the required folder.

3. Expand the **Advanced Settings** node. Enter the following information:

In this field...	Do the following...
Credentials	If you have specified a URL, and the site you want to access via the URL requires user authentication, enter a username and password for authentication at the URL site.
Resolution	Specifies how an already existing imported/included file is handled. **Note:** Currently, CentraSite does not support this option for a REST API with RAML specification.

4. Click **Next**.

   You will not be allowed to move to the next screen unless all of its required parameters have been set.

5. After you specify the value for all of the required attributes, click the **Save** icon to save the REST API with RAML definition.
6. Review the import log that CentraSite generates for the import process. If errors occur while reading and processing the file, they will be reported in this log.
7. Configure the API's extended attributes as described in "Changing a REST API" on page 84.

Tip: If you had previously imported a RAML file that has an associated schema file, and you now re-import just the schema file with modifications, your

browser might not display the updated contents of the schema file. This can happen if the browser cache is not being updated automatically. To rectify the problem, you can change your browser settings so that pages are always updated on every visit.

Importing Swagger Files

The CentraSite importer for Swagger copies a Swagger file to the repository, and creates a REST Service asset that best describes the RESTful API representing the Swagger specification.

The registry objects are as follows:

- A REST Service asset (REST API) with Swagger specification. This Swagger specification is stored as a file attribute in the **Documents** field of the **Specification** profile.
- A REST endpoint URI object that refers to the REST API.
- A REST Resource object that refers to the REST API.
- A REST Method object that refers to the REST API.
- A REST Parameter object that is referred to by the one of the above objects.
- A REST Request Content-Type object that is referred to by the REST API or REST Method object.
- A REST Response Content-Type object that is referred to by the REST API or REST Method object.
- A REST Status Code object that is referred to by the Response Content-Type object.

Swagger to CentraSite REST API Mappings

This section describes how the Swagger objects and properties (as described in the Swagger specification) are mapped in the CentraSite REST data model. It shows the relationships between the Swagger root section, parameters, body fields, and their REST data model. It also lists the relationships between Swagger resources, methods, reference documentation, and security schemes, and how these values are mapped in REST API.

Note: CentraSite supports Swagger version 2.0.

The following tables list the correspondences (mappings) between the fields of a RESTful API and Swagger document.

Swagger Root Section Fields

The Root section of the Swagger definition describes the basic information of a RESTful API, such as its title, external documents, security schemes and references.

Swagger Root Section Field	REST API Field	Notes
`swagger`		Currently, CentraSite does not support mapping the `swagger` version in the REST API's details page
`title`	Name	The `title` property is represented by the **Name** field in the REST API's details page.
`description`	Description	The `description` property is represented by the **Description** field in the REST API's details page.
`termsOfServiceUrl`		Currently, CentraSite does not support mapping the `termsOfServiceUrl`at the API level.
`contact`		Currently, CentraSite does not support mapping the `contact` property at the API level.
`license`		Currently, CentraSite does not support mapping the `license` property at the API level.
`licenseUrl`		Currently, CentraSite does not support mapping the `licenseUrl` at the API level.
`externalDocs`		The `externalDocs` are not explicitly mapped to the REST API; instead, the external documentation is attached as a file attribute to the **Specification** profile in the REST API details page.

Swagger Root Section Field	REST API Field	Notes
`host`		Currently, CentraSite does not support mapping the `host` property at the API level.
`basePath`		Currently, CentraSite does not support mapping the `basePath` property at the API level.
`schemes`		The `schemes` are not explicitly mapped to the REST API; instead, the schemes (HTTP, HTTPS) are represented by the multiple base URLs in the **Technical Details** profile. Note that CentraSite does not support mapping the WS and WSS schemes.
`produces`		Currently, CentraSite does not support mapping the `produces` property at the API level. However, it is represented by the **Response Content-Type** field of a REST Method.
`consumes`		Currently, CentraSite does not support mapping the `produces` property at the consumes level. However, it is represented by the **Request Content-Type** field of a REST Method.
`paths`	REST Resource	The `paths` property is represented by a collection of resources, wherein each of the resources have its path

Swagger Root Section Field	REST API Field	Notes
		value mapped by the **Name** field of a REST Resource.
`definitions`		The `definitions` property is represented by the Request **Schema** or Response **Schema** field of a REST Method.
`security`		Currently, CentraSite does not support mapping the `security` at the API level. However, it is represented by the **Supported Access Token Types** field in the **API-Portal Information** profile.

Swagger Parameter Fields

The following tables show the mappings between various Swagger parameter types and attributes, and how they are used by REST API. The mappings are grouped into tables for Swagger Parameter Types, and Swagger Parameter Attributes.

Swagger Parameter Types

Swagger Parameter Field	REST API Field	Notes
`URI Parameter`	Parameter of type "Path"	The `URI Parameter` property is represented by the "Path" option in the **Parameter Type** field of a REST Parameter.
`Query Parameter`	Parameter of type "Query-String"	The `QueryParameter` property is represented by the "Query-String" option in the **Parameter Type** field of a REST Parameter.
`Header`	Parameter of type "Header"	The `Header` property is represented by the "Header" option in the

Swagger Parameter Field	REST API Field	Notes
		Parameter Type field of a REST Parameter.
`Form Parameter`	Parameter of type "Form"	The `Form Parameter` property is represented by the "Form" option in the **Parameter Type** field of a REST Parameter.
`Body Parameter`		Currently, CentraSite does not support mapping parameters of the type "Body". However, a `body parameter` is represented by the **Sample Request** field of a REST Method.

Swagger Parameter Attributes

Swagger Attribute Field	REST API Field	Notes
`name`	Name	The `Display Name` property is represented by the **Name** field of a REST Parameter.
`description`	Description	The `Description` property is represented by the **Description** field of a REST Parameter.
`in`		The `in` property is represented by the **Parameter Type** field of a REST Parameter. Currently, CentraSite does not support mapping parameters of

Swagger Attribute Field	REST API Field	Notes
		the type "formData" and "body".
`required`	Required	The `required` property is represented by the **Required** field of a REST Parameter.
`type` `type=[string, number, boolean, array, file]`	Data Type	The `type` property is represented by the **Data Type** field of a REST Parameter. Currently, CentraSite does not support mapping parameters of the data type "file".
`format` `format=[int32, int64, float, double, byte, date, date-time, email, uuid]`	Data Type	The `format` property is represented by the **Data Type** field of a REST Parameter. Currently, CentraSite does not support mapping parameters of the data type "file".
`items`	Possible Values	The `enum` property is represented by the **Possible Values** field of a REST Parameter. **Limitation:** Currently, CentraSite supports the **Possible Values** field only for a "string" data type.
`collectionFormat`		Currently, CentraSite does not support the `collectionFormat` attribute.
`default`	Default Value	The `default` property is represented by the

Swagger Attribute Field	REST API Field	Notes
		Default Value field of a REST Parameter.
`maximum`		Currently, CentraSite does not support the `maximum` attribute.
`exclusivemaximum`		Currently, CentraSite does not support the `maximum` attribute.
`minimum`		Currently, CentraSite does not support the `minimum` attribute.
`exclusiveminimum`		Currently, CentraSite does not support the `minimum` attribute.
`minLength`		Currently, CentraSite does not support the `minLength` attribute.
`maxLength`		Currently, CentraSite does not support the `maxLength` attribute.
`enum`	Possible Values	The `enum` property is represented by the **Possible Values** field of a REST Parameter.
`pattern`		Currently, CentraSite does not support the `pattern` attribute.
`maxItems`		Currently, CentraSite does not support the `maxItems` attribute.

Swagger Attribute Field	REST API Field	Notes
`minItems`		Currently, CentraSite does not support the `minItems` attribute.
`uniqueItems`		Currently, CentraSite does not support the `uniqueItems` attribute.
`multipleOf`		Currently, CentraSite does not support the `multipleOf` attribute.

Swagger Resource Fields

Swagger resource properties (data) are mapped to the REST API resource fields. The following table shows the resource mappings between a Swagger definition and a REST API:

Swagger Resource Field	REST API Field	Notes
`$ref`		Currently, CentraSite does not support the `$ref` property. However, the `$ref` property is represented by mapping the contained REST Resources and attaching the referenced files to the **Specification** profile in the API details page.
`get, put,post,delete,options,head,patch`	REST Method	The `get` property is represented as the REST Method object.
`parameters`	REST Parameter	The `parameter` property is represented as a REST Parameter object.

Swagger Method Fields

Swagger method properties (data) are mapped to the REST API method fields. The following table shows the resource mappings between a Swagger definition and a REST API:

Swagger Method Field	REST API Field	Notes
`tags`		Currently, CentraSite does not support the `tags` property.
`summary`	Name	The `summary` property is represented by the **Name** field of a REST Method.
`description`	Description	The `description` property is represented by the **Description** field of a REST Method.
`externalDocs`		The `externalDocs` are not explicitly mapped to the REST API; instead, the external documentation is attached as a file attribute to the **Specification** profile.
`operationId`		Currently, CentraSite does not support the `operationId` property.
`security`		Currently, CentraSite does not support mapping the `security` at the API level. However, it is represented by the **Supported Access Token Types** field in the **API-Portal Information** profile.
`parameters`	Parameter	The `parameter` property is represented by the

Swagger Method Field	REST API Field	Notes
		Parameter field of a REST Method.
`responses`	Status Code	The `responses` property is represented by the **Status Code** field of a REST Method.
`produces`	Response Content-Type	The `produces` property is represented by the **Response Content-Type** field of a REST Method.
`consumes`	Request Content-Type	The `consumes` property is represented by the **Request Content-Type** field of a REST Method.
`deprecated`	Deprecated	The `deprecated` property is propagated to the taxonomy structure named **Deprecated**,, and represented by the **API Maturity Status** property defined in the **API-Portal Information** profile.

Swagger Status Code Fields

Swagger status code properties (data) are mapped to the REST API status code fields. The following table shows the resource mappings between a Swagger definition and a REST API:

Swagger Method Field	REST API Field	Notes
`code`	Status Code	The `HTTP status code` property is represented by the **Status Code** field of a REST Method.
`description`	Description	The `description` property is represented

Swagger Method Field	REST API Field	Notes
		by the **Description** field of a REST Method level.
`schema`		Currently, CentraSite does not support the `schema` property.
`headers`		Currently, CentraSite does not support the `headers` property.
`examples`		Currently, CentraSite does not support the `examples` property.

Swagger Security Scheme Fields

Swagger security schemes are not explicitly mapped to the REST API. Instead, the Swagger security schemes are propagated to the taxonomy structure named **Security Types**, and represented by the **Supported Access Token Types** property at the **API-Portal Information** profile. For more information on the supported security schemes, see

Security Type	Description
API Key	The API's authentication requires using an API key.
Basic Authentication	The API's authentication requires using Basic Access Authentication as described in RFC2617.
Digest Authentication	The API's authentication requires using Digest Access Authentication as described in RFC2617.
OAuth 1.0	The API's authentication requires using OAuth 1.0 as described in RFC5849.
OAuth 2.0	The API's authentication requires using OAuth 2.0 as described in RFC6749.
x-{other}	The API's authentication requires using another authentication method.

The supported security schemes along with their scheme reference name are mapped in the CentraSite registry. For more information about the security scheme, see the Swagger specification.

Importing a REST API Using Swagger File

Before you begin, you must have the Swagger file that you want to import.

To import a new REST API with Swagger definition

1. In the activity bar, click **Create Asset**. This opens the **Create Asset** wizard.
2. In the **Basic Information** profile, enter the following information in the fields provided:

In this field...	Do the following...
Name	*Optional.* Enter a name for the new REST API. This name must be NCName-conformant, meaning that: ■ The name must begin with a letter or the underscore character (_). ■ The remainder of the name can contain any combination of letters, digits, or the following characters: . - _ (i.e., period, dash, or underscore). It can also contain combining characters and extender characters (e.g., diacriticals). ■ The name cannot contain any spaces. ■ Furthermore, if the name contains any non-conformant character, then that character is simply replaced with the underscore character (_). For more information about the NCName type, see http://www.w3.org/TR/xmlschema-2/#NCName **Note:** The API name does not need to be unique. However, to reduce ambiguity, we recommend that you adopt appropriate naming conventions to ensure that API is distinctly named within an organization. If you have not specified a **Name** for the REST API, CentraSite automatically maps this field with the `API Title` property, which is defined in the Swagger specification.
Type	Choose **REST Service** asset type.

In this field...	Do the following...
Organization	Select the organization in which the new API will be created. (The drop-down list displays the list of organizations in which you are permitted to create APIs.)
Version	*Optional.* Specify a version identifier for the new API. If you have not specified a **Version** for the REST API, CentraSite automatically maps this field with the `API Version` property, which is defined in the Swagger specification.
Description	*Optional.* Enter a description for the new API. **Note:** This is the description information that users will see when they view instances of this type in the user interface, therefore, the description should be meaningful. If you have not specified a **Description** for the REST API, CentraSite automatically maps this field with the `Description` property, which is defined in the Swagger specification.
Import From Specification File	Choose **Swagger-2.0** as the specification file type.
URL or **File**	Specify whether the input file will be read from your local file system (the **File** option) or from a URL-addressable location on the network (the **URL** option). If the file you are importing resides on the network, specify its URL. If the file resides in your local file system, specify the file name. You can use the **Browse** button to navigate to the required folder.

3. Expand the **Advanced Settings** node. Enter the following information:

In this field...	Do the following...
Credentials	If you have specified a URL, and the site you want to access via the URL requires user authentication, enter a username and password for authentication at the URL site.

In this field...	Do the following...
Resolution	Specifies how an already existing imported/included file is handled. **Note:** Currently, CentraSite does not support this option for a REST API with Swagger specification.

4. Click **Next**.

   You will not be allowed to move to the next screen unless all of its required parameters have been set.

5. After you specify the value for all of the required attributes, click the **Save** icon to save the REST API with Swagger definition.
6. Review the import log that CentraSite generates for the import process. If errors occur while reading and processing the file, they will be reported in this log.
7. Configure the API's extended attributes as described in "Changing a REST API" on page 84.

Tip: If you had previously imported a Swagger file that has an associated schema file, and you now re-import just the schema file with modifications, your browser might not display the updated contents of the schema file. This can happen if the browser cache is not being updated automatically. To rectify the problem, you can change your browser settings so that pages are always updated on every visit.

3 Creating a RESTful API from Scratch

Overview

Now that we are familiar with the RESTful paradigm, let us focus on documenting an API using the enhanced REST data model in CentraSite Business UI.

If you are documenting our sample online phone store application as an API by capturing the metadata collected from online phone store application. The metadata to manage the latest phone details online will include the following:

- A list of resources. For example, `phones`
- A list of HTTP methods the API will support for each individual resource `phones`. For example, GET, PUT, DELETE, and POST.
- A list of parameters that will best describe the resource `phones`. For example, `features=androidosv4.3&cameraresolution=8MP`
- A list of sample HTTP request and response messages

The sequence of documenting a REST API using the CentraSite's Business user interface can be best understood with the following illustration:

- Start by defining basic details of an API, and then define the base URL, schemas and parameters.
- Easily capture resources and methods, and then add the required details as you want.
- Add parameters and other details (content types, status codes) specific to each call.
- Specify samples for requests specific to each call, and then the corresponding samples for expected responses.

Adding a RESTful API

Note: Beginning with Version 9.7, CentraSite supports the ability to define APIs using the Business UI only. Please note that using the CentraSite Control interface to define REST APIs is no longer supported.

Before You Begin

To define a REST API in an organization, you must belong to a role that has the Create Assets or Manage Assets permission for that organization. For more information about roles and permissions, see *Getting Started with CentraSite*.

Adding a REST API to CentraSite

Perform these steps to create the simple REST API and save it to CentraSite.

To add a REST API

1. In the activity bar, click **Create Asset**. This opens the **Create Asset** wizard.
2. In the **Basic Information** profile, enter the following information in the fields provided:

In this field...	Do the following...
Name	*Mandatory.* Enter a name for the new REST API. This name must be NCName-conformant, meaning that: ■ The name must begin with a letter or the underscore character (_). ■ The remainder of the name can contain any combination of letters, digits, or the following characters: . - _ (i.e., period, dash, or underscore). It can also contain combining characters and extender characters (e.g., diacriticals). ■ The name cannot contain any spaces. ■ Furthermore, if the name contains any non-conformant character, then that character is simply replaced with the underscore character (_). For more information about the NCName type, see http://www.w3.org/TR/xmlschema-2/#NCName **Note:** The API name does not need to be unique. However, to reduce ambiguity, we recommend that you adopt appropriate naming conventions to ensure that API is distinctly named within an organization.
Type	The **REST Service** asset type.
Organization	Select the organization in which the new API will be created. (The drop-down list displays the list of organizations in which you are permitted to create APIs.)
Version	*Optional.* Specify a version identifier for the new API.
Description	*Optional.* Enter a description for the new API. **Note:** This is the description information that users will see when they view instances of this type in the

In this field...	Do the following...
	user interface, therefore, the description should be meaningful.
URL or **File**	Specify whether the input file will be read from your local file system (the **File** option) or from a URL-addressable location on the network (the **URL** option). If the file you are importing resides on the network, specify its URL. If the file resides in your local file system, specify the file name. You can use the **Browse** button to navigate to the required folder.

3. Expand the **Advanced Settings** node. Enter the following information:

In this field...	Do the following...
Credentials	If you have specified a URL, and the site you want to access via the URL requires user authentication, enter a username and password for authentication at the URL site.
Resolution	Choose a resolution strategy, which will allow you to specify how an already existing imported/included file is handled. For each of the imported/included files you have one of these options: ■ Overwrite the importing file with new content. ■ Create a new version of the file with the new content (if, for example, you want to modify a XSD file but want to retain its previous version).

4. Click **Next**.

   Be aware that you will not be allowed to move to the next panel unless all of its required parameters have been set.

5. After you specify the value for all of the required fields, click the **Save** icon to save and add the new REST API to CentraSite registry.

   The newly created API's details page is displayed. Here you can enter the values of various attributes of the new API.

6. Configure the API's extended attributes as described in the subsequent sections.

Configuring Global Details

After you define the REST API, you then expose the API's base URLs. A base URL path includes the hostname of the server where the API is actually hosted.

Before You Begin

When you configure the global details for an API, keep the following points in mind:

- A base URL is the core design element that serves as the only way to access the identified API.

  CentraSite allows you to configure multiple base URL paths for the API. For example, you can configure a sandbox URL for testing purposes, and a production URL for accessing real-world data.

  For our sample Phone Store API, here are the sandbox and production base URLs:

  Sandbox URL

  `https://www.sandbox.phonestore.com/api/v2`

  Production URL

  `https://www.phonestore.com/api/v2`

  For example, a base URL for our sample Phone Store API would look like:

  `https://www.phonestore.in/api/v2`

  A complete URL is formed by combining the resource path with the base URL.

  For example, here is the URL you would use in a request to get the list of phones:

  `GET https://www.phonestore.in/api/v2/phones`

  Or, retrieve a phone with product code is `412456`

  `GET https://www.phonestore.in/api/v2/phones/phone-412456`

  Where,

  GET - HTTP request method

  https://www.phonestore.in/api/v2 - URL

  `phones` - resource URI

  `412456` - path parameter

- An API parameter is an expression that represents a value that the client passes to the API specified in the client call. Here, the parameters are specified at the API level. Since these are defined at the API level, the parameters will be available for all child resources and methods below the API in the hierarchy.

Adding Base URL to REST API

Configure the base URLs for the API using which users would traverse to any of the API's resources. In order to execute this task, you must know the URL of the server that is hosting the API you intend to model.

To add a base URL

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Technical Details** profile.
4. Enter the following information in the fields provided:

In this field...	Do the following...
Base URL	*Optional.* Enter the server base URL in the text box. **Note:** If you are specifying multiple base URLs for an API, it must be unique among all URLs in the API. If you want to specify additional base URLs, use the plus button beside the text box to create a new base URL input field, and enter another URL. If at any time you want to remove a base URL, use the minus button.
Sandbox	*Optional.* The sandbox category by which you want to classify base URL for the API. a. Click **Choose**. b. When you click the button, the **Choose Sandbox Categories** dialog appears which allows you to select the required categories for base URL. c. Click the expand node next to Sandbox taxonomy to view the categorization tree. d. Mark the checkbox beside the name of the category to classify the base URL. e. Click **OK**. CentraSite includes a set of predefined categories for the taxonomy node "Sandbox", especially for classifying base URLs of REST APIs. By default, the base URLs can

In this field...	Do the following...
	be classified into these following predefined categories: Development, Production, Test. For information on the Sandbox categories that CentraSite supports out-of-the-box, in CentraSite Control, go to **Administration > Taxonomies**. In the Taxonomies page, navigate to **Sandbox** in the list of taxonomies. If you would like to use sandbox categories that are not supported by CentraSite, you can define your custom categories. **Note:** Although it is possible to define subcategories for the predefined and custom categories within the Sandbox taxonomy, you cannot use these subcategories to classify the base URLs. CentraSite only displays the names of the top-level categories (that is, categories that are defined for the Sandbox taxonomy) for the classification.
Namespace	*Optional.* Specify the target namespace.
Request Content-Type	Select the content format for request message. (The drop-down list displays the list of supported content formats.) **Note:** By default, this field shows an empty value.
Response Content-Type	Select the content format for response message. (The drop-down list displays the list of supported content formats.) **Note:** By default, this field shows an empty value.
Parameters - Add Parameter	*Optional.* Specify one or more request parameters of the following types at the API level: - Query-String - Header - Form Although CentraSite allows you to define parameters of the type "Form" at the API level, these parameters are not supported at run-time. Only parameters of the type Query-String and Header are supported at run-time. **Note:** You cannot add more than one parameter with the same name and the same type within the API level.

In this field...	Do the following...
	a. Click the **Add Parameter** link to open the **Add Parameter** dialog. b. In this dialog, you define input parameters at the API level. To specify multiple parameters, click the **Add Parameter** link to add each new parameter. The new parameter is added to the **Technical Details** profile. For a complete description of how to add the request parameters, see "Adding Parameter to REST API" on page 70.

5. To further update the new parameter, mouse over the parameter, and then click the **Edit** icon. Repeat for each parameter that you want to modify.
6. To specify multiple parameters, click the **Add Parameter** link to add each new parameter.
7. After you specify the value for all of the required fields, click the **Save** icon to save the changes.

Configuring REST Resources

Resources are the basic components of a REST API.

Before You Begin

After you have exposed the base URIs for accessing the API, you must identify and first define the resources for it. By identifying the resources in the API, you can make the API more useful and easier to develop.

Each resource has its own unique URI. Defining URI patterns is important because URIs enable clients to directly access a resource.

For example, consider the case of our sample Phone Store API. Assume this API exposes a database that defines a list of phones and the features and specifications of each phone; wherein the list of phones is represented with the collection URI and the features of each phones is represented as an individual URI.

Collection Resource URI

```
/phones
```

Unique Resource URI

```
/phones/412456
```

We recommend that you follow these simple tips and guidelines for structuring the resource path (URI):

- **Short name URIs** as much as possible - for example, prefer `/phones/412456` than `/phones/phone.php?phone_id=412456`
- **Straightforward and meaningful URIs** to ease use of the resource - `/phones/412456`
- **Consistent and predictable URIs** patterns -`/phones/412456/features`
- **Simple and hierarchical URIs** to represent the relationships - for example,
  - `/phones`
  - `/phones/412456`
  - `/phones/412456/features`
- **Nouns, not verbs** - for example, plural nouns to represent list of things - `/phones`; singular nouns to represent a particular thing - `/phones/412456`
- **Hyphens, avoid spaces or underlines** to improve and enhance aesthetic interaction with the resource - for example, prefer `/phones/412456` than `/phones/412456`
- **Lower case, avoid mixed case and upper case** to improve readability - `/phones/412456` than `/Phones/412456`

Here are some common examples for our sample resource `/phones`:

- `/phones`
- `/phones/412456`
- `/phones/412456?fields=(make,features,bodytype)`
- `/phones/412456/make`
- `/phones/search?q=(make,eq,apple)`
- `/phones/?make=apple&features=3g&price.min=44101`

Adding Resource to REST API

In this task, you identify the resource of the API, and capture the resource URI for the API. Indirectly, a URI address implies the relationship between each of the resources.

To add a resource

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Resource and Methods** profile.
4. Click the **Add Resource** link.

5. In the **Add Resource** dialog box, enter the following information in the fields provided:

In this field...	Do the following...
Name	*Mandatory.* Enter a display name for the new resource. This name must be NCName-conformant, meaning that: ■ The name must begin with a letter or the underscore character (_). ■ The remainder of the name can contain any combination of letters, digits, or the following characters: . - _ (i.e., period, dash, or underscore). It can also contain combining characters and extender characters (e.g., diacriticals). ■ The name cannot contain any spaces. ■ Furthermore, if the name contains any non-conformant character, then that character is simply replaced with the underscore character (_). For more information about the NCName type, see http://www.w3.org/TR/xmlschema-2/#NCName **Note:** If you are specifying multiple resources for an API, the name must be unique among all of the resources in that particular API.
Resource Path	*Mandatory.* Enter the new resource URI. Structure the URI with the simple tips and guidelines described above. **Important:** As a best practice, we recommend that you adopt the following conventions when specifying a path parameter in the resource URI: ■ Append a path parameter variable within curly `{}` brackets. ■ Specify a path parameter variable such that it exactly matches the path parameter defined at the Resource level.
Description	*Optional.* Enter a description for the new resource. **Note:** This is the description information that users will see when they view instances of this type in the

In this field...	Do the following...
	user interface, therefore, the description should be meaningful.
Upload Schema	*Optional.* Specify a XML Schema Definition (XSD) file for the new resource.
Upload Files	*Optional.* Specify one or more input files for API. You can use the **Browse** button to navigate to the required folder. If you want to specify additional input files, use the plus button to create a new input field, and then use the **Browse** button to select another input file.
Parameters - Add Parameter (link)	*Optional.* Specify one or more request parameters of the following types at the Resource level: ■ Path ■ Query-String ■ Header ■ Form Although CentraSite allows you to define parameters of the type "Form" at the Resource level, these parameters are not supported at run-time. Only parameters of the type Path, Query-String and Header are supported at run-time. **Note:** You cannot add more than one parameter with the same name and the same type within the resource. a. Click the **Add Parameter** link to open the **Add Parameter** dialog. b. In this dialog, you define input parameters at the Resource level. To specify multiple parameters, click the **Add Parameter** link to add each new parameter. The new parameter is added to the resource. For a complete description of how to add the request parameters, see "Adding Parameter to REST API" on page 70.

6. Expand the resource whose details you want to view.
7. To further update the new resource, mouse over the resource, and then click the **Edit** icon. Repeat for each resource that you want to modify.

8. To specify multiple resources, click the **Add Resource** link to add each new resource.
9. After you specify the value for all of the required fields, click the **Save** icon to save the changes.

Configuring HTTP Methods

The HTTP methods passed as part of an HTTP request tell the API what operation needs to be done with the addressed resource.

Before You Begin

Understand the predefined HTTP methods and their known attributes. See the HTTP method definitions information to learn more about the common set of methods for HTTP.

Clients use HTTP methods to perform certain operations. Multiple methods exist - GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH, TRACE, and CONNECT.

Important: During virtualization of an API, CentraSite does not support the following HTTP methods: HEAD, OPTIONS, PATCH, TRACE, and CONNECT. This is because, when a virtual API is published to webMethods Mediator, at run-time, the HTTP methods GET, POST, PUT, DELETE are only supported.

Let us consider the following scenarios for our sample Phone Store API:

Resource URI	Supported HTTP Methods	Description
`/phones`	GET	List all phones.
`/phones`	POST	Creates a new phone with product code `412456`.
`/phones/412456`	GET	Retrieves details of a phone whose product code is `412456`.
`/phones/412456`	DELETE	Removes a phone whose product code is `412456`.
`/phones/412456?fields=(make,features,bodytype)`	GET	Retrieves additional details (such as Brand, Features, Body Type) of a phone

Resource URI	Supported HTTP Methods	Description
		whose product code is `412456`.
`/phones/412456/make`	GET	Identifies the brand of a phone whose product code is `412456`.
`/phones/search?q=(make,eq,apple)`	GET	Retrieves a list of all phones whose brand is `Apple`.
`/phones/412456?make=apple&features=3g`	PUT	Updates a phone whose product code is `412456`, brand is `Apple`, and also `3G` compatible.

Resource methods can also use parameters to identify or pass additional information.

You can capture the sample requests and responses to facilitate clients easily interact with the resources of API.

You can set HTTP status codes to help client quickly and more easily understand the HTTP response messages.

Adding HTTP Method to REST API

In this task, you define the valid operations for the resources. In addition, you can define the resource representation formats and the samples to represent the HTTP requests and responses.

To add a HTTP method

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Resource and Methods** profile.
4. Locate the resource you want to add HTTP methods.
5. Click the **Add Method** link.
6. In the **Add Method** dialog box, enter the following information in the fields provided:

In this field...	Do the following...
Name	*Optional.* Enter a name for the method.
Description	*Optional.* Enter descriptive information about the method.
HTTP Method	Select the HTTP operation you want to perform on the resource. (The drop-down list displays the list of supported HTTP methods.)
Request Content-Type	Select the content format for request message. (The drop-down list displays the list of supported content formats.) **Note:** By default, this field shows an empty default value.
Response Content-Type	Select the content format for response message. (The drop-down list displays the list of supported content formats.) **Note:** By default, this field shows an empty default value.
Deprecated	*Optional.* Specify whether or not the REST method is deprecated.
Parameters - **Add Parameter** (link)	*Optional.* Specify one or more request parameters of the following types at the Method level: ■ Query-String ■ Header ■ Form Although CentraSite allows you to define parameters of the type "Form" at the Method level, these parameters are not supported at run-time. Only parameters of the type Query-String and Header are supported at run-time. **Note:** You cannot add more than one parameter with the same name and the same type within the method. a. Click the **Add Parameter** link to open the **Add Parameter** dialog.

In this field...	Do the following...
	b. In this dialog, you define input parameters at the Method level. To specify multiple parameters, click the **Add Parameter** link to add each new parameter. The new parameter is added to the method. For a complete description of how to set up the request parameters, see "Adding Parameter to REST API" on page 70.
Requests - **Add Request** (link)	*Optional.* Specify one or more HTTP requests to the resources of the REST API. For a complete description of how to set up the HTTP requests, see "Adding HTTP Request to REST API" on page 72.
Responses - **Add Response** (link)	*Optional.* Specify one or more HTTP responses to the resources of the REST API. For a complete description of how to set up the HTTP responses, see "Adding HTTP Response to REST API" on page 74.
Sample Requests and Responses - **Add Request and Response** (link)	*Optional.* Write one or more sample requests to the resources of the REST API, and the corresponding sample responses from the API. For a complete description of how to expose the HTTP requests and response messages, see "Adding Request and Response Messages to REST API" on page 77.

7. Click the newly added method to view its details.
8. To further update the new method, mouse over the method, and then click the **Edit** icon. Repeat for each method that you want to modify.
9. To specify multiple methods, click the **Add Method** link to add each new method.
10. After you specify the value for all of the required fields, click the **Save** icon to save the changes.

Configuring REST Parameters

Parameters are used to pass and add additional information to a request. You can use parameters as part of the URL or in the headers or as part of the message body.

Before You Begin

Multiple parameter types exist - the most widely used parameters are path and query-string parameters.

- Path parameters, which are integral part of the request URL, and correspond to the URL path variable names.

  The following example shows the path parameter representations and the result expected:

  `GET /phones/412456` - Returns the details for a specific phone whose product code is `412456`.

  In the above snippet, the URL path variable name `412456` is passed as a parameter to the GET method.

  Note that CentraSite allows you to define a path parameter only at the Resource level.

- Query-String parameters, which are passed as the request URL query parameters.

  The following examples show the different query-string parameter representations and the results expected:

  - `GET /phones?make=apple` - Returns a list of all the phones that match the specified brand `Apple`.
  - `GET /phones/412456?format=JSON` - Returns the details for phone whose product code is `412456` in the JSON format.

- Header parameters, which are passed as custom HTTP headers.

  The following example shows the header parameter representation:

  ```
  GET /phones?412456
  Accept-Encoding: application/json
  x-CentraSite-APIKey:66f4b263-cc6e-11e3-85a7-a2d064a5bd02
  ```

- Form parameters, which are encoded as the POST part of the request message body.

Important: Although CentraSite allows you to define parameters of the type "Form" at the API level, these parameters are not supported at run-time.

Adding Parameter to REST API

In this task, you define input parameters either at the API level, Resource level or Method level. Defining a parameter at the API level (in the **Technical Details** profile) means that it is inherited by all Resources, and by all methods under the individual Resources. Defining a parameter at the Resource level (in the **Add Resource** dialog box) means that it is inherited by all Methods under it. Defining it at the Method level (in the **Add Method** dialog box) only applies the parameters to that particular method; it does not affect either the API level or Resource level.

To add a parameter

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).

2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Technical Details** or **Resource and Methods** profile.
4. Expand the **Parameters** section in the **Technical Details** profile or the **Add Resource** dialog box or the **Add Method** dialog box, as required.
5. Click the **Add Parameter** link.
6. In the **Add Parameter** dialog box, enter the following information in the fields provided:

In this field...	Do the following...
Name	*Mandatory.* Enter a display name for the parameter.
Description	*Optional.* Enter a comment or descriptive information about the parameter.
Parameter Type	Select a parameter type (Path, Header, Query-String, Form). For a list of the supported parameter types, see "Parameter Types" on page 16.
Data Type	Select the parameter's data type.
Required	Specify whether the parameter is mandatory or optional for invoking the REST API.
Array	Specify whether the parameter can hold a just a single value or multiple values (an array of values). Enabling the **Array** option allows you to assign more than one value to the parameter.
Possible Values	*Optional.* Enter a list of possible values for this parameter.
Default Value	*Optional.* If you want to specify a default value, enter a value in this field.

7. To further update the new parameter, mouse over the parameter, and then click the **Edit** icon. Repeat for each parameter that you want to modify.
8. To specify multiple parameters, click the **Add Parameter** link to add each new parameter.
9. If you need to delete a parameter, mouse over the parameter, and then click the **Delete** icon. Repeat for each parameter that you want to delete.
10. After you specify the value for all of the required fields, click the **Save** icon to save the changes.

Configuring HTTP Requests

A HTTP request describes the input to a HTTP method (as a collection of parameters) for the addressed resource.

Adding HTTP Request to REST API

In this task, you define the valid requests for the resources. In addition, you can define the request representation formats, and the schemas and examples to represent the HTTP requests.

To add a HTTP request

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Resource and Methods** profile.
4. In the **Add/Modify Method** dialog box, expand the **Requests** section.
5. Click the **Add Request** link.
6. In the **Add Request** dialog box, enter the following information in the fields provided:

In this field...	Do the following...
Name	This is a label that you assign as a meaningful name for the HTTP request. For example, you may call a HTTP request of the type application/xml, as `XML Request` or `XML Payload`. You may also call a HTTP request based on the data that it holds. For example, you could specify a POST request to create new customer as `Create Customer Request`.
Description	*Optional.* Enter a comment or descriptive information about the request.
Request Content-Type	Select the content format for request message. (The drop-down list displays the list of supported content formats.) **Note:** By default, this field displays an empty default value.
Schema	*Optional.* Specify a HTTP request message (payload) using either XML schema or JSON schema.

In this field...	Do the following...
Inline or **External File**	Specify whether the schema definition will be read from an inline text (the **Inline** option) or from an external file or URL (the **External File** option). If you want to use an inline schema, specify the schema definition manually in the **Inline** text box. If the schema definition you are uploading resides on the network, specify its URL. If the schema definition resides in an external file, specify the file name. You can use the **Browse** button to navigate to the required folder. **Note:** As a best practice, you should use the **Inline** option to include small number of data, and use the **External File** option to include large number of data stored in an external file.
Example	*Optional.* Specify a HTTP request message (payload) using examples to show how a schema is used. An example is simply an XML code or JSON code.
Inline or **External File**	Specify whether the example will be read from an inline example code (the **Inline** option) or from an external file or URL (the **External File** option). If you want to use an inline example, specify the example code manually in the **Inline** text box. If the example definition you are uploading resides on the network, specify its URL. If the example definition resides in an external file, specify the file name. You can use the **Browse** button to navigate to the required folder. **Note:** As a best practice, you should use the **Inline** option to include small number of data, and use the **External File** option to include large number of data stored in an external file.

7. To further update the new request, mouse over the request, and then click the **Edit** icon. Repeat for each request that you want to modify.
8. To specify multiple requests, click the **Add Request** link to add each new request.
9. If you need to delete a request, mouse over the request, and then click the **Delete** icon. Repeat for each request that you want to delete.

10. After you specify the value for all of the required fields, click the **Save** icon to save the changes.

Configuring HTTP Responses

A HTTP response indicates the success or failure of an API invocation.

Before You Begin

HTTP/1.1 defines all the legal status codes. Examine these codes to determine which are appropriate for your API.

HTTP response status codes provide information about the status of a HTTP request. The HTTP specification defines several status codes that are typically understood by clients.

Adding HTTP Response to REST API

In this task, you define the valid responses for the HTTP requests. In addition, you can define the response representation formats, and the schemas and examples to represent the HTTP responses.

To add a HTTP response

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Resource and Methods** profile.
4. In the **Add/Modify Method** dialog box, expand the **Responses** section.
5. Click the **Add Response** link.
6. In the **Add Response** dialog box, enter the following information in the fields provided:

In this field...	Do the following...
Status Code	Select a HTTP response status code number. **Examples** ■ HTTP 200 OK ■ HTTP 400 Bad Request ■ HTTP Error 404 Not Found

In this field...	Do the following...
	■ HTTP Error 500 Internal Server Error
Name	This is a label that you assign as a meaningful name for the HTTP response. For example, you may call a HTTP 400 response as `Validation Error`, instead of the `Bad Request`.
Description	*Optional.* Enter a comment or descriptive information about the response.
Response Content-Type	Select the content format for response message. (The drop-down list displays the list of supported content formats.) **Note:** By default, this field displays an empty default value.
Schema	*Optional.* Specify a HTTP response message (payload) using either XML schema or JSON schema.
Inline or **External File**	Specify whether the schema definition will be read from an inline text (the **Inline** option) or from an external file or URL (the **External File** option). If you want to use an inline schema, specify the schema definition manually in the **Inline** text box. If the schema definition you are uploading resides on the network, specify its URL. If the schema definition resides in an external file, specify the file name. You can use the **Browse** button to navigate to the required folder. **Note:** As a best practice, you should use the **Inline** option to include small number of data, and use the **External File** option to include large number of data stored in an external file.
Example	*Optional.* Specify a HTTP response message (payload) using examples to show how a schema is used. An example is simply an XML code or JSON code.
Inline or **External File**	Specify whether the example will be read from an inline example code (the **Inline** option) or from an external file or URL (the **External File** option).

In this field...	Do the following...
	If you want to use an inline example, specify the example code manually in the **Inline** text box. If the example definition you are uploading resides on the network, specify its URL. If the example definition resides in an external file, specify the file name. You can use the **Browse** button to navigate to the required folder. **Note:** As a best practice, you should use the **Inline** option to include small number of data, and use the **External File** option to include large number of data stored in an external file.
Headers - Add Header (link)	*Optional.* Specify one or more headers that you want to include in the HTTP response. **Note:** A header name must be unique within the response. a. Click the **Add Header** link. b. In the **Add Header** dialog box, specify the required fields.

7. To further update the new response, mouse over the response, and then click the **Edit** icon. Repeat for each response that you want to modify.
8. To specify multiple responses, click the **Add Response** link to add each new response.
9. If you need to delete a response, mouse over the response, and then click the **Delete** icon. Repeat for each response that you want to delete.
10. After you specify the value for all of the required fields, click the **Save** icon to save the changes.

Configuring Sample Request and Response Messages

REST APIs can produce successful response and errors. CentraSite allows you to capture the default error responses when an exception or error occurs.

For a list of the sample client request and server response messages, see "Sample Requests and Responses" on page 21.

Note: Beginning with version 9.8, although CentraSite supports the existing **Sample Requests and Responses** wizard, we enforce you use the **REST Requests and REST Responses** wizards to specify additional details about the REST payload. You may use the **Sample Requests and Responses** wizard if required.

Adding Request and Response Messages to REST API

In this task, you define sample request and response messages for each method.

To add a sample request and response message

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Resource and Methods** profile.
4. In the **Add/Modify Method** dialog box, expand the **Sample Requests and Responses** section.
5. Click the **Add Request and Response** link.
6. In the **Add Sample Request and Response** dialog box, enter the following information in the fields provided:

In this field...	Do the following...
Request	Enter the HTTP request message. **Important:** As a best practice, we recommend that you use sample messages that could be sent from the client to the server.
Response/Error	Enter the HTTP response/error message. If you want to specify additional response message, use the plus button to create a new input field. If you want to remove an existing response message, use the minus button. Repeat for each response message that you want to remove.

7. To further update the new sample request or response message, click the **Edit** icon. Repeat for each message that you want to modify.
8. To specify multiple request and response messages, click the **Add Request and Response** link to add each new request and response messages.
9. If you need to delete a request and response message, click the **Delete** icon. Repeat for each request and response message that you want to delete.
10. After you specify the value for all of the required fields, click the **Save** icon to save the changes.

4 Managing RESTful APIs

Viewing a REST API

You use the details page of the REST API to examine your existing documentation.

Before You Begin

The following general guidelines apply when examining the existing API documentation:

- If you are not the owner of the API, you cannot view the API unless you have View permission on the API (granted though either a role-based permission or instance-level permission). For more information about roles and permissions, see *Getting Started with CentraSite*.
- You will only see profiles for which you have the instance-level View permission.
- If you are viewing the API details page, you can choose to display in **Resources** view and **Methods** view. The **Methods** view is set by default.
- In either of the views, mouse hover the resource or method details and delete that resource or method using the **Delete** icon.

If You Migrate REST APIs from a Pre-9.7 Release

If you have REST APIs that were created prior to version 9.7, those APIs will continue to hold the old version's metadata in the enhanced REST API interface implemented by current version of CentraSite.

Viewing the List of REST APIs

The **Search Results** page displays the list of REST APIs in CentraSite. Note that this list displays APIs for which you have the View permission.

You can sort the list by attribute. To specify the sorting preference, select the attribute from the drop-down list labeled **Sort by**.

To view the list of REST APIs

1. In CentraSite Business UI, click the **Browse** link (in the upper left corner of the page).
   - Alternatively, select `Everything` or `Assets` from the drop-down beside the keyword search text box.
2. Expand the splitter panel using the given arrow.
3. In the **Narrow Your Results** section, locate **Additional Search Criteria**.
4. Select **Asset Types** from the drop-down list.

5. Enter the asset type `REST Service` in the text box. Click the plus button next to the text box or press Enter to add REST Service to the search recipe.
   - Else, click **Choose**. This opens the **Choose Asset Types** dialog.
     i. Navigate to `REST Service` and select the appropriate check box.
     ii. Click **OK**.
6. If you want to further filter the list to see a subset of the available APIs, Type a partial string in the **Keyword** text field. Click the plus button next to the text field or press Enter to add the keyword to the search recipe. For more information about advanced search options, see *Working with the CentraSite Business UI* .
7. The **Search Results** page provides the following information about each API.

> **Note:** By default, only the attributes described below are displayed in this list. Use the **View** menu to display the additional attributes.

Column	Description
Name	The name assigned to the REST API.
Description	Additional comments or descriptive information about the API.
Asset Type	The type of asset, `REST Service`.
Last Updated Date	The date on which the API was last modified.
Owner	The user to which the API belongs.
Organization	The organization to which the API belongs.
Version	The user-assigned version identifier for the API.

Viewing the Details of a REST API

In this task you view the various basic and type-specific attributes associated with the API. You can view the resources, methods and parameters in the **Resources** view and the **Methods** view; in addition, you can delete the existing API parameters, resources and methods.

To view the details of a REST API

1. Display the list of REST APIs (see "Viewing the List of REST APIs" on page 80 for details).

2. In the displayed list, click the link of the API whose details you want to view. This shows the API's basic information (version identifier, last updated date, asset type, owning organization, owning user, a description of the API).

   You can view a tooltip text for some of the attributes in the profiles of the API's details by moving the cursor to the info icon. The tooltip text gives a summary of the attribute's purpose. The tooltip text shown is the content of the attribute's Name and Description fields as defined in the API.

3. Use the **Resources and Methods** profile to view the extended details of the API, namely, resources, methods, parameters, status codes, and request and response messages

   The content in the **Resources and Methods** profile reflects the view selection that you make - resource-centric, or method-centric. The profile displays multiple fields which are dependent on the view you select.

4. To select a view, use the **Resources | Methods** menu displayed on the right hand side. Depending on the view selected, the profile displays a list of resources, or methods.
5. Expand the resource name whose details you want to view.
6. Click the name button of the method to view its details.
7. Drill down to different levels in the API Parameters, Resource Parameters, Method Parameters, HTTP Requests, HTTP Responses, and Sample Requests and Responses to see details of each of them.
8. Click the hyperlinked parameter name to its view details.
9. If you have resources and methods defined for the API, you can delete one or more of these entities by using the **Delete** icon in the **Resources and Methods** profile as follows:

   a. Ensure that the **Resources and Methods** profile of the API is selected in the API's detail view mode.

   b. In the Resource-Centric View, locate the resource that you want to delete. Do the following:

      a. Move the cursor over the resource you want to delete.

      b. Click the **Delete** icon

      c. Repeat for each resource that you want to delete.

      d. You can also drill down to the individual HTTP methods that are defined for the selected resource and repeat the previous steps for each method as required.

   c. In the Method-Centric View, locate the HTTP method that you want to delete. Do the following:

      a. Move the cursor over the method you want to delete.

      b. Click the **Delete** icon.

c. Repeat for each method that you want to delete.

Using the Method-Centric View

The Method-Centric View can be accessed by clicking on **Methods** menu in the right hand side of **Resources and Methods** profile.

This view displays the available HTTP methods for an API. For an API, if there are HTTP methods defined at the Resource level, the method-centric view displays the list of all HTTP methods defined at various Resource levels for that API. This view provides you a consolidated list of supported HTTP methods for a resource path URI. The Method-Centric View is displayed by default.

In short, a Method-Centric View displays:

- ..> HTTP Methods...
- … > Resource Path, Name…
  - … > Method Description, Request Content Type, Response Content-Type ….
  - … > Method Parameters > Name, Type, Description…
  - … > HTTP Requests > Name, Description, Request Content-Type, Schema, Example…
  - … > HTTP Responses > Status Code, Name, Description, Response Content-Type, Schema, Example…
  - … > Sample Requests and Responses > Sample 1 > Request, Response / Error…

Using the Resource-Centric View

The Resource-Centric View can be accessed by clicking on **Resources** menu in the right hand side of **Resources and Methods** profile.

This view displays the available resources for an API. For the selected API, if there are multiple resources, and each resource defined with multiple HTTP methods, the resource-centric view displays all components - resources, and methods that apply to the selected API. This includes parameters, HTTP requests, HTTP responses, and the sample requests and responses defined at various Method and Resource levels in the API.

This view provides you a consolidated list of available resources for the selected API.

In short, a Resource-Centric View displays:

- …> Resources …
  - … > Description, Resource Path, Resource Parameters, Documents, Schema…
  - … > HTTP Methods, Resource Path, Resource Name…
    - ... > Description, Request Content Type, Response Content Type ….

- … > Method Parameters > Name, Type, Description…
- … > HTTP Requests > Name, Description, Request Content-Type, Schema, Example…
- … > HTTP Responses > Status Code, Name, Description, Response Content-Type, Schema, Example…
- … > Sample Requests and Responses > Sample 1 > Request, Response / Error…

Changing a REST API

You use the details page of the REST API to refractor your existing documentation.

Before You Begin

The following general guidelines apply when refractoring the existing API documentation:

- If you are not the owner of the REST API, you cannot transform the API documentation unless you have Modify permission on the API (granted though either a role-based permission or instance-level permission). For more information about roles and permissions, see *Getting Started with CentraSite*.
- If you are viewing the API details page, you can choose to display in **Resources** view and **Methods** view. The **Methods** view is set by default.
- When you are viewing the resources and methods, you can delete one or more of the top level REST details - resources and methods by using the **Delete** icon. For more information about resources and methods and the **Delete** option, see "Viewing the Details of a REST API" on page 81.
- However, if you are editing the resources and methods, then you can delete the remaining resources and methods, namely - request parameters, status codes and sample requests and responses.
- If you want to edit the resources and methods, ensure that the **Resources and Methods** profile of the API is selected in the edit mode by using the **Edit** icon in the action bar for the API.
- In the edit mode, you will only see an editable user interface of the Resource-Centric View. There is no Method-Centric View in the edit mode.
- If you are editing the API details page, you can modify one or more of the existing entities - resources, methods, request parameters, HTTP requests, HTTP responses, and the sample requests and responses; also you can delete these entities by using the **Delete** icon.
- Currently, CentraSite supports only specific properties of RAML and Swagger specifications. For example, if the Swagger specification includes a `swagger` version

property, you cannot define the swagger's version in the API documentation. For a list of the supported RAML or Swagger to CentraSite REST API mappings, see "Importing a RESTful API to CentraSite" on page 25.

REST API Compatibility

Beginning with version 9.7, CentraSite supports the enhanced interface for REST APIs (in contrast, earlier versions of CentraSite supported a standardized interface for REST Service). Documentation of the prior REST service interface is available to CentraSite customers who have a current maintenance contract in Empower Product Support website.

- **If You Migrate REST APIs from a Pre-9.7 Release**: If you have REST APIs that were created prior to CentraSite version 9.7, these REST APIs will continue to hold the version's metadata in the enhanced REST Service interface implemented by current version of CentraSite.
- **If You Migrate REST APIs from a 9.7 Release**: If you have REST APIs that were created in the CentraSite version 9.7 using the CentraSite Business UI, these REST APIs will again continue to hold the version's metadata in the enhanced REST Service interface implemented by current version of CentraSite. However, you will find the following information in the migrated REST API:
  - The sample request and response messages that existed under the REST Method display without changes.
  - The status codes that existed under the REST Method will now display under the REST Response.

Note: Beginning with version 9.8, although CentraSite supports the existing REST Sample Requests and Responses, we enforce you use the REST Requests and REST Responses to specify additional details about the REST payload. You may use the Sample Requests and Responses if required.

Editing the Details of a REST API

In this task you examine and change the various basic and type-specific attributes associated with the API. In addition, you can examine and change the resources and methods, such as the resources, HTTP methods, parameters, HTTP requests and responses, and the sample messages in the **Resources** view and the **Methods** view, as applicable; also, you can delete the existing parameters (either the resource parameters or the method parameters), HTTP requests and responses, and the sample messages.

To edit the details of a REST API

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.

3. To edit the API's basic attributes, place the cursor in the appropriate field and modify the text as required.
4. To modify the extended attributes associated with the API, do the following:
   a. Select the profile that contains the attribute(s) that you want to modify.
   b. Edit the attributes on the profile as necessary.
   c. Repeat steps 5.a and 5.b for each profile that you want to edit.

      If at any time you want to abandon your unsaved edits, click **Close**. CentraSite will ask you if you want to save your edits. Click **Discard** to abandon your edits and return the API's attributes to their previous settings.
5. When you have finished making your edits, click the **Save** icon to save the changes.
6. When you are prompted to confirm the save operation, click **Yes**.

Editing the Details of Resources and Methods

You use the **Resources and Methods** profile of an API to view, modify, and delete its resources and methods. The content in this profile reflects the view selection that you make - resource-centric, or method-centric.

When you edit the details for an API, be aware that you will not be allowed to edit its resources and methods in the method-centric view.

If you have resources and methods defined for the API, you modify one or more of these entities by using the appropriate **Edit** the icon in the **Resources and Methods** profile.

To edit the details of resources and methods

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Edit** icon.
3. Select the **Resource and Methods** profile. Add or modify the REST details at the Resource level and Method level, as required.
   - Resource level details include the basic information for a resource, and its request parameters.
   - Method level details include the basic information for a method, its request parameters, content types, status codes, and HTTP messages that are available for the selected method.
4. To modify the resource details, do the following:
   a. Move the cursor over the resource whose details you want to modify.
   b. Click the **Edit** icon.
   c. In the **Edit Resource** dialog, modify the details, as needed:

Field	Description
Name	*Mandatory.* The name of the resource. Make sure the resource name you specify in this field is a valid value for NCName.
Resource Path	*Mandatory.* The resource URL.
Description	*Optional.* Additional comments or descriptive information about the resource.
Upload Schema	*Optional.* An XML schema for the API. **Note:** If you have an API that uses XML as content, then you can optionally upload an XML schema document.
Upload Files	*Optional.* External files that provide additional information about the resource.
Parameters	*Optional.* One or more request parameters at the Resource level. **Modify an Existing Parameter** a. Move the cursor over the parameter whose details you want to modify. b. Click the **Edit** icon. c. In the **Edit Parameter** dialog, modify the details, as needed. d. Click **OK**. e. Repeat for each parameter that you want to modify. **Delete an Existing Parameter** a. Move the cursor over the parameter you want to delete. b. Click the **Delete** icon. c. Repeat for each parameter that you want to delete. **Add a New Parameter** a. Click the **Add Parameter** link.

Field	Description
	b. In the **Add Parameter** dialog, specify the details, as needed. c. Click **OK**. d. Repeat for each parameter that you want to add. For a complete description of how to set up the request parameters, see "Adding Parameter to REST API" on page 70.

5. To modify the method details, do the following:
   a. Move the cursor over the method whose details you want to modify.
   b. Click the **Edit** icon.
   c. In the **Edit Method** dialog, modify the details as necessary:

Field	Description
Name	*Mandatory.* The name of the method.
Description	*Optional.* Additional comments or descriptive information about the method.
HTTP Method	The HTTP operation to perform on the resource.
Request Content-Type	The content format for request message.
Response Content-Type	The content format for response message.
Parameters	*Optional.* One or more request parameters at the Method level. **Modify an Existing Parameter** a. Move the cursor over the parameter whose details you want to modify. b. Click the **Edit** icon. c. In the **Edit Parameter** dialog, modify the details, as needed. d. Click **OK**.

Field	Description
	e. Repeat for each parameter that you want to modify. **Delete an Existing Parameter** a. Move the cursor over the parameter you want to delete. b. Click the **Delete** icon. c. Repeat for each parameter that you want to delete. **Add a New Parameter** a. Click the **Add Parameter** link. b. In the **Add Parameter** dialog, specify the details, as needed. c. Click **OK**. d. Repeat for each parameter that you want to add. For a complete description of how to set up the request parameters, see "Adding Parameter to REST API" on page 70.
Requests	*Optional.* One or more HTTP requests that indicate the operation to be performed with the addressed resource. **Modify an Existing Request** a. Move the cursor over the request whose details you want to modify. b. Click the **Edit** icon. c. In the **Edit Request** dialog, modify the details, as needed. d. Click **OK**. e. Repeat for each request that you want to modify. **Delete an Existing Request** a. Move the cursor over the request you want to delete. b. Click the **Delete** icon. c. Repeat for each request that you want to delete. **Add a New Request** a. Click the **Add Request** link.

Field	Description
	b. In the **Add Request** dialog, specify the details, as needed. c. Click **OK**. d. Repeat for each request that you want to add. For a complete description of how to set up the request, see "Adding HTTP Request to REST API" on page 72.
Responses	*Optional.* One or more HTTP responses that indicate the success or failure of a request invocation. **Modify an Existing Response** a. Move the cursor over the response whose details you want to modify. b. Click the **Edit** icon. c. In the **Edit Response** dialog, modify the details, as needed. d. Click **OK**. e. Repeat for each response that you want to modify. **Delete an Existing Response** a. Move the cursor over the response you want to delete. b. Click the **Delete** icon. c. Repeat for each response that you want to delete. **Add a New Response** a. Click the **Add Response** link. b. In the **Add Response** dialog, specify the details, as needed. c. Click **OK**. d. Repeat for each response that you want to add. For a complete description of how to set up the response messages, see "Adding HTTP Response to REST API" on page 74.
Sample Requests and Responses	*Optional.* One or more sample requests to the resources of the web application, and the corresponding sample responses from the application.

Field	Description
	Modify an Existing Sample a. Move the cursor over the sample whose details you want to modify. b. Click the **Edit** icon. c. In the **Edit Sample Request and Response** dialog, modify the details, as needed. d. Click **OK**. e. Repeat for each sample that you want to modify. **Delete an Existing Sample** a. Move the cursor over the sample you want to delete. b. Click the **Delete** icon. c. Repeat for each sample that you want to delete. **Add a New Sample** a. Click the **Add Request and Response** link. b. In the **Add Sample Request and Response** dialog, specify the details, as needed. c. Click **OK**. d. Repeat for each sample that you want to add. For a complete description of how to expose sample request and response messages, see "Adding Request and Response Messages to REST API" on page 77.

6. If you edited any of the details on the **Resources and Methods** profile, click the **Save** icon to save the changes.

Deleting a REST API

You delete an API to permanently remove it from the CentraSite registry.

Before You Begin

Before you delete an API, keep in mind the following points:

- If you are not the owner of the API, you cannot delete the API unless you have Full permission on the API (granted though either a role-based permission or instance-

level permission). For more information about roles and permissions, see *Getting Started with CentraSite*.

- You cannot delete an API that is in pending state (e.g., awaiting approval).
- You cannot delete an API if any user in your CentraSite registry is currently modifying the API.

Deleting a Single REST API

In this task, you delete a single API to remove it from CentraSite permanently.

To delete a single API

1. Display the details page of the REST API (see "Viewing the Details of a REST API" on page 81 for details).
2. In the actions bar for the API, click the **Delete** icon.
3. When you are prompted to confirm the delete operation, click **Yes**.

   The API is permanently removed from the CentraSite registry.

Deleting Multiple REST APIs in a Single Operation

You can delete multiple APIs in a single step.

To delete a set of APIs

1. Display the list of REST APIs (see "Viewing the List of REST APIs" on page 80 for details).
2. In the displayed list, mark the checkbox of the APIs that you want to delete.
3. In the actions bar, click the **Delete** icon.

> **Note:** If have you selected a set of APIs, where one or more APIs is in pending state (e.g., awaiting approval), CentraSite ignores the pending list of APIs, and deletes any remaining APIs for which you have the required permission.