REST Fundamentals of Canada Post Web Services

Print

- **Conduct your testing in the sandbox (development) environment. You will be billed for shipments and orders submitted in the production environment. Read more.**

Canada Post services allow e-commerce solution providers and online merchants to integrate Canada Post services, such as shipping, rating and tracking data, into a platform or website. Once integrated, your application will access Canada Post servers over REST style architecture using XML.

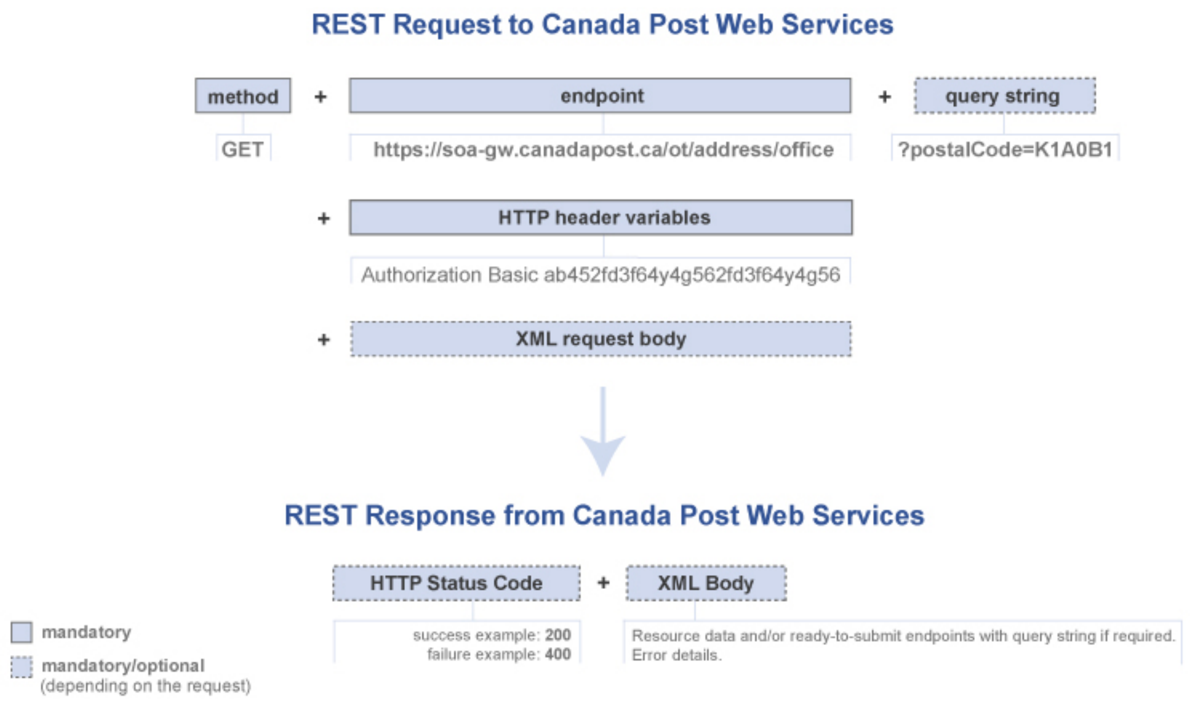
View typical use case scenarios.

Before beginning integration work

Read **Getting Started** to find out how to sign up and get your API keys.
Review the information included here in *Fundamentals of Canada Post Services*. It details essential information common to all our services.
Run our sample application to test your access to our services and observe the data sent to (and returned by) our web services.
Read the detailed functional and technical descriptions for each of our web services in the Service Directory.

Using REST to interact with web services

The general format for any REST request is illustrated below. See the service directory for details by call.


REST Request to Canada Post Web Services
method
+
endpoint
+
query string
GET
https://soa-gw.canadapost.ca/ot/address/office
?postalCode=K1A0B1
+
HTTP header variables
Authorization Basic ab452fd3f64y4g562fd3f64y4g56
+
XML request body
REST Response from Canada Post Web Services
HTTP Status Code
+
XML Body
success example: 200
failure example: 400
Resource data and/or ready-to-submit endpoints with query string if required.
Error details.
mandatory
mandatory/optional (depending on the request)

Endpoints

You will use two types of endpoints to access Canada Post Services:

Endpoints constructed by your application code.
Endpoints provided in XML responses to prior requests.

Each endpoint has been given a handy name for reference purposes. This reference name does not appear directly in the endpoint.

Become familiar with the reference names. When they need to be invoked by application code, consult the documentation in the service directory for details about each endpoint or code sample.

Constructed endpoints

Constructed endpoints consist of a fixed URL part and may also contain variable parts. The endpoint for Get Manifests is shown below.

Endpoint example (as per documentation) for Get Manifests:

XX/rs/{mailed by customer}/{mobo}/manifest
?start=YYYYMMDD&end=YYYYMMDD

Replace {mailed by customer} with your customer number
Replace {mobo} with the mailed on behalf of customer number or repeat your customer number
Replace YYYYMMDD with the start and end dates.

Endpoint example (with variable values populated):

GET *https://ct.soa-gw.canadapost.ca/rs/123456789/123456789/manifest?start=20100324&endDate=20100405*

Constructed endpoint variables

There are two kinds of variable parts:

The "domain" of the request. For example:

the mailed by customer number

the mobo (mailed-on-behalf-of) customer number associated with the request

The query string of the request

Contains one or more query parameters that may be required depending on the service being invoked.

Domain variables: mailed by customer and mobo

The domain might be fixed for all your calls if your application acts on behalf of one customer only. In the detailed service documentation, domain variables are represented inside curly brackets. For example:

{mailed by customer} means that your application must provide your customer number.

{mobo} means that your application must provide the customer number of the entity you are mailing on behalf of—referred to as a mailed-on-behalf-of customer. If you are not mailing on behalf of anyone, repeat your customer number here.

The domain variables are protected in the same way as the XML body under the HTTPS connection.

Query string

If present, the query string will vary based on your current application needs. For example, the query string could pass user input of **Postal Code** to the **Get Nearest Post Office** request. To be interpreted properly, a query string cannot contain blank spaces. Blank spaces, can, however, be replaced by **%20** (e.g., for the space between the two parts of a Postal Code).

The query string is protected in the same way as the XML body under the HTTPS connection.

Provided endpoints

The Canada Post API provides dynamically generated endpoints as a response from a previously called web service. In these cases, you should use the endpoint as received and should not attempt to parse it or construct it.

All of the Canada Post web services invoked by provided endpoints require an HTTP GET method, and do not require an XML body.

Provided endpoints are described in an XML response element named "link". The link element takes the following form.

```
<link rel="{rel-indicator}" href="{endpoint URL and query string}" media-type="{media-type index="{xx}" >
```

Link attributes of provided endpoints

Link attribute	Meaning
rel (relationship to link)	Indicates the resource related to the current response.
href (hypertext reference)	Indicates the endpoint to be used to call the web service. It contains the URL and may also contain a query string.
media-type	A character string that indicates the format and version of the data to expect when the web service is invoked. The value in this attribute should be copied to the HTTP header variable "Accept" when the href link is invoked.
index (optional)	Present on links to formatted output such as labels. The value starts at 0 for the first page and subsequent pages (if extant) count up by 1.

Call patterns

The first interaction with the Canada Post API will always be through a constructed endpoint. Subsequent interactions will typically invoke provided endpoints until all information related to the original call has been retrieved. Two common patterns can be used. See **Ad hoc and batch call patterns**.

If at any time during processing your application fails and stored links are lost, they can be recovered with **synchronization calls**.

HTTP header variables

Mandatory HTTP header variables must be provided for each request. Foremost is the authorization value which is used to authenticate each request.

HTTP Header Variable	Applicable Methods	Mandatory / Optional	Value	Description
Authorization	GET, POST, DELETE	Mandatory	Basic userid:password	• The word "Basic" is a literal value that must be present. • A single space follows the word Basic. • The API key follows the space as a single Base64-encoded string (52 characters). • The encoded string is generated from the API key (a concatenation of the userid, a colon and the password) • For an example of a utility that can generate the Base64-encoded string from your userid and password parts, see **Test Tools**.
Content-Type	POST	Mandatory	Unique per service group. See service documentation for details.	This is the XML version of the body you are sending in the POST. (**Note:** */* in place of the header value will return an error)
Accept	GET	Mandatory	Unique per service group. See service documentation for details or use "media-type" from the provided link.	This is the XML version of the response that you are expecting to receive. (**Note:** */* in place of the header value will return an error)
Accept-Language	GET, POST	Optional	"fr-CA" or "en-CA"	This indicates the preferred language of the human readable error message (if error is generated).
If-None-Match	GET	Optional	As per Etag value from original response.	May be included in request if a prior "GET" was cached. Either "304 – not changed" will be returned or the new version of the resource will be returned.

Structure of the HTTP response

The general form of an HTTP response to Canada Post web services contains the following standard headers:

```
Status Code: 200 OK
Date: Wed 06 Jul 2011 17:21:53 GMT
Content-Type: application/vnd.cpc.shipment+xml
Etag: "6"
{body }
```

HTTP status codes

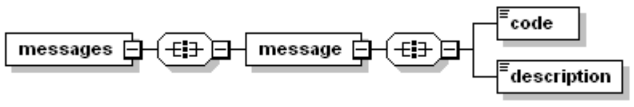
The HTTP status code returned indicates the success or error of the request as described below.

HTTP Status Code	Success / Error	Body Contents	Action
200	Success	As per specific service (see individual service documentation).	Process the details in the message body.
202	Try Later	Error Msg. Structure	The requested resource is not yet available. Please try again later. For example, after a Create Non-Contract Shipment request, a delay of approximately one second is required before you can retrieve the shipping label with a Get Artifact call.
204	Success	None	No action required. e.g., Delete was successful.
304	Success	None	Use the stored version of the data. (The resource has not been modified. Used in combination with ETag and If-None-Match headers).
	Error	Error Msg. Structure	Code = "Server"

400			A request failed schema validation. The description contains a detailed message to help you troubleshoot. These errors should only occur during development and should be resolved before any application goes to production.
	Error	Error Msg. Structure	Code = numeric If the code is numeric, the error could either be corrected by an end-user, given a flexible UI, or could be corrected by the developer in the form of logical selections, filters or validations by the application in advance of sending the request. **View a list of errors you can mitigate in advance**.
401	Error	Error Msg. Structure	Correct the API key in the "Authorization" header.
403	Error	Error Msg. Structure	AA001 - The API key in the "Authorization" header has been deactivated. If you withdrew from the Developer Program, rejoin the program.
	Error	Error Msg. Structure	AA002 - Correct the API key in the "Authorization" header. You used a development API key against production or vice versa.
	Error	Error Msg. Structure	AA003 - The API key in the "Authorization" header does not match the mailed-by customer number in the request.
	Error	Error Msg. Structure	AA004 - You cannot mail on behalf of the requested customer.
404	Error	Error Msg. Structure	The resource path is incorrect or the resource is no longer available. If received for a recently created resource, retry later. e.g., the artifact has not finished rendering.
406	Error	Error Msg. Structure	The interface version expected to be returned by the request (GET) is invalid or not supported. Change the "accept" header variable.
415	Error	Error Msg. Structure	The interface version declared in the request (POST) is invalid or not supported. Change the "content-type" header variable.
500	Error	Error Msg. Structure	Correct the XML schema error as per the detailed message.

Error message structure

The error message structure (if returned) takes the place of the normal success structure. The structure of the error message is illustrated below.



Sample XML response to an error condition

```
HTTP/1.1 403
Content-type:application/vnd.cpc.shipment+xml
  <messages>
    <message>
      <code>7007</code>
      <description> The weight value is invalid. The weight of each piece must be less than or equal to 30 kg.</description>
    </message>
    <message>
      <code>1722</code>
      <description> The options are invalid for the selected Service. Please change the options or select another service.
      </description>
    </message>
  </messages>
```

Web service environments

Two web service environments are available to you:

Sandbox (development) – used for testing your code.
Production – used in your production environment.

Select the web service environment by using:

Your development or production API key in your "Authorization" header, and
A development or production endpoint.

Note: HTTP status code 403 will be returned if you use the wrong API key for the endpoint selected.

Sandbox (development) environment

Use your development key against the sandbox environment. The sandbox environment is a replica of production and includes valid test data and responses, with the exception of the following:

Item	Appears in Response From	Response Values in the Sandbox Environment
tracking-pin element	Create Shipment Get Shipment Details	123456789012
return-tracking-pin element	Create Shipment	123456789012
po-number element	Create Shipment Get Shipment Details Get Manifest Details Get Manifest	P123456789
rel="receipt" link (REST)	Create Shipment	This link/element is not returned in the sandbox environment
Artifacts, such as shipping labels, return labels and manifests	Get Artifact	Labels and manifests are clearly identified as "for testing purposes only" and therefore cannot be used for actual shipping.
Tracking web services	Get Tracking Summary Get Tracking Details Get Signature Image Get Delivery Confirmation Certificate	Regardless of the tracking number you submit in the request to tracking web services, the response "No Pin History" is returned. To test a full set of tracking scenarios, use production tracking PINs in the production environment. (To test "not found" use an invalid PIN.)

Sandbox endpoints

The endpoints for the sandbox environment are in the code samples and in the detailed service documentation. For sandbox endpoints,
XX = https://ct.soa-gw.canadapost.ca.

Test values for contract shipping, rating and returns

If you are not a commercial customer of Canada Post with a contract, but you are developing a third-party shipping solution intended for commercial customers, use the numbers below for testing contract shipping, rating, and returns web services. Please note the following:

These numbers are only valid in the sandbox environment.
You must be a *Solutions for Small Business*™ customer to use these test numbers. **Join now**.
Several Canada Post customers will be using the same test numbers. To avoid confusion, make sure you create your own group-ids when you create your shipments. **Read more about group-ids**.
We have set up a default test credit card under this test customer to allow you to test credit card transactions.
To obtain commercial test rates, in your request to Get Rates, you must include both the customer-number and the contract-id.
Rates returned are for test purposes only.

Element name/key	Text values for sandbox environment
Customer-number	2004381
Contract-id	42708517
API key	6e93d53968881714:0bfa9fcb9853d1f51ee57a

Test value for promo codes

Promotion Code DEVPROTEST can be used to test promo code functionality in the sandbox environment. This promo code is only valid in the sandbox environment and for the following products:

Xpresspost (DOM.XP)
Xpresspost International (INT.XP)

Production environment

After your application has been tested successfully against the sandbox environment, you can change your API key to the production key and change the endpoints to production endpoints.

Test the endpoints that do not incur a charge first, such as those in Rating, Tracking and Find a Post Office. If you create shipments as a test, to avoid incurring a charge, call Void Shipment for these test shipments before calling Transmit Shipments.

Production endpoints

The endpoints for the production environment begin with https://soa-gw.canadapost.ca. Further details can be found in the documentation for each service.

ZPL II thermal labels – truncation requirement

If you are printing thermal shipping labels in ZPL II format, your printer must have the ability to truncate text. Without this ability, you risk printing over top of important information on Canada Post shipping labels. Check your printer to make sure it can truncate text. Citizen printers, for example, as well as Zebra printers with older firmware, do not support truncation. You can test your printer using our sample code below.

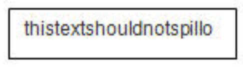
Sample code to stream to your thermal printer

Copy and paste the code below into the utility provided by your printer manufacturer and print this to a thermal label.

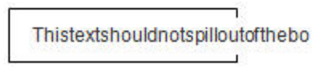
```
^XA
^CI13
^MMT
^PW812
^LL1218
^LS0
^FO63,63^A0N,25,20^FDThe text below should not spill out of the box^FS
^FO127,127^A0N,25,20^TBN,190,25^FDthistextshouldnotspilloutofthebox^FS
^FO121,121^GB197,32,1^FS
^PQ1,0,1,Y
^XZ
```

Printer results

If your printer supports truncation, the printout will be a box with the truncated text appearing within the box, as shown below:



If your printer does not support truncation, the text will spill outside the box as shown below:



If your printer does not support truncation, we recommend you use PDF format for labels instead. See the encoding element for more details: **REST** | **SOAP**.

Versioning

Please refer to our Versioning Strategy page.

Using XSD files

All web service requests and responses are validated against a corresponding schema. The XML Schema Definition (XSD) files are useful to generate code or to validate XML requests on the client side in advance of making a call. The XSD files were used to generate the data classes in the code samples. **Download XSD files**.

Questions? Find answers in Support.