

Usage

Postman

- Install [Postman](#)
- Open Postman and import the vendor-test-extract collection JSON(vendor-tests.postman_collection.json) and environment JSON (vendor-env.postman_environment.json) from the zip file.
- Apply test data file included in the zip file (shop_inputs.csv for shopping endpoints and book_inputs.csv for booking related endpoints).
- Ensure "baseUrl" environment variable is set in the postman environment to appropriate values for your deployment.
- Run requests as you see fit. Change request parameters by changing the variables from the provided environment file.

Newman (Optional)

- Install [newman](#), the cli tool for running Postman collections

```
brew install newman
```

or

```
npm i -g newman
```

or

```
npm install -g newman
```
- For Linux/MacOS, Execute the shell script from the zip folder, `./run-vendor-tests.sh` which will execute scenarios for both shop and book endpoints.
- For Windows, Execute the bat file from command prompt using command `.\run-vendor-tests.bat` .

Environment variables

The following variables are being set in the global environment. For testing with real credit cards, please update corresponding payment related variables. NOTE: Fields which contain ** are the required fields in the schema to make the requests.

Env Parameters	Description	Example value
cardType **	PaymentCardType.	VISA.
cardNumber **	Payment card number corresponding to the card type	4242-4242-4242-4242

Env Parameters	Description	Example value
cvv	Card Verification Value - three or four-digit number on credit card for security	369
cardHolderName	Name of the card holder	John Smith
addressLines	Address of the card holder	910 Mainland Street
city **	City name	Vancouver
state	Two-character state code	BC
countryCode **	Two-character ISO code (ISO ALPHA-2) for country.	CA
postalCode	Postcal code	V5K 0A1
programCode **	programCode	EM.
accountld **	account ID	209875030.
baseUrl **	Vendor endpoint	
posRequestorId **	An identifier of the entity making the request (e.g. ATA/IATA/ID number)	HTL:91369
username	user name for authentication.	
password	password for authentication.	

Data files variables

The following data is from the csv file which is being used for this postman collection.

NOTE: Fileds which contain ** are the required fields in the schema to make the requests.

Dynamic Parameters	Description	Example value
travelerUuid **	UUID that identifies the traveler within Concur	123e4567-e89b-12d3-a456-426614174000
loginId	Login ID of traveler within Concur. Only sent when available.	abc@concur.com
bookingForSelf	Is logged in person booking for self or on behalf of someone else	true
numGuests	No. of guests for accomodation as entered by traveler	1

Dynamic Parameters	Description	Example value
guestCountryCode	Two-character ISO code (ISO ALPHA-2) for country	CA
locationType **	Type of location associated with this search	Hotel
name **	Name of the location corresponding to it's locationtype	Sheraton DFW Airport Hotel
latitude **	Geo location latitude coordinates for search	49.246292
longitude **	Geo location longitude coordinates for search	-123.116226
otaCode **	Code based on OTA Rate Plan Type (RPT) list. (https://www.opentraveldevelopersnetwork.com/code-list)	1
value	Optional value for the given rate plan type code	AAA
searchSessionToken	Session token to be generated and provided by server on initial "search" call that can be referenced back for future api calls on the same session	3fa85f64-5717-4562-b3fc-2c963f66afa6
paymentModelIndicator	Type of the card from [PERSONAL_CARD, CORPORATE_CARD, CONCUR_VIRTUAL_CARD, VENDOR_VIRTUAL_CARD].	PERSONAL_CARD
customFieldName **	Vendor specific field name if setup for vendor integration	OrgUnit
customeFieldValue **	Value of the custom field	Travel Agents

OpenAPI validation in Postman








The tests in this collection validate requests and responses against the HS4 OpenAPI spec. It does this by first reading the `schema` environment variable which stores the HS4 spec, and registering each object in the schema. Postman ships with the [Tiny Validator for v4 JSON Schema](#) included. In each of the tests, the `tv4` library is used to validate the request and response models against the schema.

Test Evidence to be shared with Concur Team

To help with quicker onboard process and early triage of issues it is recommended the test execution evidence is shared with concur team upon completion of test execution.

1. Irrespective of what tool is used for doing the testing the suppliers must document and share at-least one example for each request and response (in JSON Format) of all REST endpoints on the suppliers host

system that's implementing HSv4 spec.

Name	Date Modified	Size	Kind
 Cancel_req_res.txt	Yesterday at 10:49 AM	574 bytes	Plain Text
 HotelDetails_req_res.txt	Yesterday at 10:49 AM	61 KB	Plain Text
 RateDetails_req_res.txt	Yesterday at 10:49 AM	3 KB	Plain Text
 Rates_req_res.txt	Yesterday at 10:49 AM	153 KB	Plain Text
 Read_req_res.txt	Yesterday at 10:49 AM	4 KB	Plain Text
 Reservation_req_res.txt	Yesterday at 10:49 AM	6 KB	Plain Text
 Search_req_res.txt	Yesterday at 10:49 AM	382 KB	Plain Text

2. The data files utilized in tests (shop_inputs.csv & book_inputs.csv) and the host URL information.
3. The vendor-env.postman_environment.json file used to conduct the tests.

On top of these two documents the following additional evidence documents should be shared with concur team depending on the tool of choice used for testing.

Additional Test Execution Evidence when a Tester utilized Newman to run the tests.

The contents of the console when `./run-vendor-tests.sh` (in case of testers host machine operating system being OS X) or `.\run-vendor-tests.bat` (in case of testers host machine operating system being Windows) needs to be extracted and shared with concur Team. Please find below for an example console Log

(only partial log is shown in screen print for brevity of this guide) in case of a successful test execution:

- ✓ Request Successful
- ✓ Request: schema validated successfully
- ✓ Response: schema validated successfully
- ✓ Request Validation: valid checkin, checkout
- ✓ All hotels have valid hotelName
- ✓ Hotel Addresses: All hotels have addressLines, city, countryCode
- ✓ Hotel LeadRate: All hotels have valid lead rates
- ✓ All hotels have amenities
- ✓ Search response has 139 hotel properties 0 duplicate hotel properties

	executed	failed
iterations	1	0
requests	141	0
test-scripts	4	0
prerequest-scripts	4	0
assertions	23	0
total run duration: 14.3s		
total data received: 21.86MB (approx)		
average response time: 6.9s [min: 153ms, max: 13.5s, s.d.: 3.6s]		

Process finished with exit code 0

Additional Test Execution Evidence when a Tester utilized postman to run the tests.

Capture the screen prints of each request execution in such a way the response HTTP status and test execution results were visible. Please find below for an example postman test evidence screenshot for search transaction.

POST search

vendor-env

vendor-tests / shop / search

POST

{{baseUri}}/hotels/search

Response Status

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

BodyCookiesHeaders (4)Test Results (14/14)

AllPassedSkippedFailed

PASS

Request schema is valid

PASS

Request: schema validated successfully

PASS

Response: schema validated successfully

PASS

Search request contains location

PASS

Response size is less than 5000000b

PASS

Request Validation: valid checkin, checkout

PASS

Hotel Addresses: All hotels have addressLines, city, countryCode

PASS

Hotel LeadRate: All hotels have valid lead rates

PASS

All hotels have amenities

PASS

Search response has 139 hotel properties 0 duplicate hotel properties

PASS

Search response has less than 200 properties. Total number of returned properties was 139

PASS

All hotels have valid hotel names

PASS

All leadImageURLs have https protocol

PASS

All image extensions are jpg or png

Status: 200 OK

Time: 155 ms

Size: 48.29 KB

Save Response

Use this option to save response to a file

2 Errors278 WarningsAll LogsClear