Livn API Documentation

Revision 2018-02-05

1. Introduction

1.1. About Livn and its API

Livn is an aggregated network of tour, accommodation and transport operators, who are integrated via an Application Programming Interface (API) allowing GDSs, OTAs or essentially any interested travel retailer to retrieve and distribute land based travel products with real-time availability & rates, curated and enriched content and to make instant confirmed reservations through a single connection.

For more information about Livn please visit www.livn.world

1.2. Getting started

To start using the API you must first obtain authentication by contacting support at Livn.
This can easiest be done by emailing support@livngds.com.

While developing and testing your client implementation, please work on the development server and use the authentication credentials, all of which you will be given to you by our support staff.

When you have finished developing and testing your client implementation, please contact support again to begin the certification process. Once certified you will be issued with another server address and credentials to access the live production server.

If you receive inventory from the Livn Holidays Wholesale program, the production endpoint can be found at: https://gds1.livngds.com/livngds/api/application.wadl . Customers of Livn Direct, operating their own instance of the application and API, will be assigned a different unique subdomain under livngds.com.

Please note: While the interface on all development and production systems follow the same architecture defined in this document, the actual data output by any one server is not transferable to any other system. To speed up development and testing, a set of generic tour operator mappings will have been automatically set up with the development server. If you are seeing variations in results between live and training data, this is because we are using generic test mappings and not specified mappings to your account or actual supplier base.

1.3. API Overview

Livn uses a RESTful API to provide a simplified method of access to its centralised database and our system in turn connects to multiple tour, accommodation and transport operators' reservation systems. The functionality listed here allows a client application to retrieve operator and product information, as well as assemble the necessary data to commit a reservation into an operator's booking system.

1.3.1. Message body format

Livn API's HTTP requests and responses consist of well defined key-value pairs or collections thereof in the JSON format. Alternatively you can request message exchange using XML, however, due to JSON's quite significantly lower overhead when compared to XML (i.e. less wasted bandwidth) and it's overall better performance, we recommend JSON as our preferred message format and XML support is currently in a deprecated status.

Following the REST standard please specify the format of your requests and the desired format of response messages returned by our system, by including `Content-Type` and `Accept` HTTP requests headers with the respective value (usually `application/xml` or `application/json`).

The entire scope of the REST API's available resources, consumable input and available output content types, as well as the structural blueprints for all involved data types, can be seen at a glance in the application's WADL document (`https://[API base URL]/application.wadl`) and the XML Schema XSD documents referenced therein.

1.3.2. Mandatory and Optional Fields

Livn API response messages commonly only include fields that contain any actual information in our database (at the time of the request).

As the application aggregates several operator reservation systems, and data available from each source when we make the initial connection to bring a new product into our system may vary, we cannot guarantee the same complete depth and scope of information will be returned for every operator, product, pickup point and so on.

To assist with the implementation of your interface any fields that will always be returned with a successful request's response have been specifically marked in the following tables.

1.4. API Security

All communication with the Livn API is confined to HTTPS, thus using SSL back-to-front encryption. The system further uses Basic HTTP authentication and authorisation with all resources, except those few found in the `/public` path and the WADL documentation. This uses a HTTP request header of the form:

```
Authorization: Basic <base64 concatenated loginName and password>
```

As of February 2017 there is an alternative to HTTP Basic Authentication, using JSON Web Tokens (JWT) instead. The process is to initially retrieve a token by POST-ing loginCredentials, containing loginName and password, to /public/login. The returned token is valid for 24h and can be used for all subsequent secured API requests, by adding a different header of the form:

```
Authorization: Bearer <token>
```

Unauthenticated requests to any protected resource will result in a HTTP response with status 401 (Unauthorized), containing a `WWW-Authenticate` header in accordance with IETF RFC 1945 (http://tools.ietf.org/html/rfc1945#section-11.1), prompting clients to authenticate by including an `Authorization` header in all subsequent requests.

As preceding every successful access to the API with failed unauthenticated request and response messages, results in unnecessary system load and overhead, please include the correct `Authorization` header, containing your hashed user credentials or JWT token, with every single request.

1.5. Request rate limits

As some of the API's services require communication with our supplier APIs and are subject to rules limiting the number of requests we can make in a specific time frame, all the more so as these quotas have to be shared by all of our clients, these calls may be put under similar restrictions. If the limit is exceeded, services will return a HTTP error code 429 Too Many Requests (in accordance with RFC 6585). Please see the resource reference below for concrete limitations that may apply.

1.6. Publish/Subscribe Messaging for Push(-like) Notifications

Seeing as the data at the base of nearly every aspect of Livn API, just like in any other travel reservation system, is far from static, clients relying exclusively on the one-way communication provided by a RESTful API, while perfectly viable, can benefit significantly from the introduction of a reverse channel, making for an altogether smoother implementation.

A common approach to establishing a form of two way communication, is through HTTP callbacks, in more recent years popularly referred to as Webhooks, which essentially requires the consumers of an API, to implement and provide their own API endpoints and register these with the opposite party, so these services can be called whenever a specific event is triggered. This solution suffers from several intrinsic flaws, most importantly (in our opinion) scalability, the requirement to provide a publicly accessible endpoint and the long list of complications surrounding guaranteed delivery and reconciliation in the event of missed events (e.g. what happens if the registered callback resource cannot be reached? How often does system A try to contact system B, before it discards the message?)

After looking at several potential solutions and weighing off their respective advantages and disadvantages, we have decided to implement the reverse channels for Livn API, using publish and subscribe style messaging via Amazon Web Services, more specifically the Simple Notification Service (SNS) and Simple Queue Service (SQS).

If you are unfamiliar with AWS in general, and SNS/SQS in particular, there are many excellent online articles, tutorials and blog posts, regarding publish & subscribe messaging using SNS + SQS, such as: http://budiirawan.com/how-to-connect-amazon-sns-to-amazon-sqs-queues/ or of course on AWS's own pages: https://aws.amazon.com/sns/

Harnessing AWS as the message broker to convey inventory, and potentially at a later stage, availability and rates related changes, to our clients, comes with a certain peace of mind regarding uptime and scalability.

Please see chapter *5. Inventory event notifications using AWS Pub/Sub Messaging* for a step-by-step guide to implementing Livn API's notification services.

1.7. Adult = Rack Rate

Across the API, for any resource that includes pricing information, in the absence of any concrete child and/or infant pricing, a product's adult rate will be applicable for all participants regardless of age, and essentially represents the single rack rate for the product.

Unless explicitly limited by a minimum age rule (product property *minAge*), this single price, as well as any associated commissions, net rates etc, is to be used for all pax.

2. Functional Resource Reference

All resources are relative to the respective API's base URL (e.g. the API server you are most likely to begin developing against can be found at `https://dev1.livngds.com/livngds/api`)
Unless stated otherwise, all resource service endpoints can consume and produce both `application/json` and `application/xml` formatted data. Please note: XML support is currently deprecated and might be dropped in future revisions of the API.

2.1. Operators & Operator specific Products

getOperators: Returns a list of all operators available in Livn
Request: `GET /operators`
Request parameters:

- boolean `includeFullDetails` (optional, default: true)
- timestamp `modifiedSince` (optional, include this to limit results to only those operators, that have been added or modified since a specific time instant)
- boolean `includeManualBooking` (optional, default: false, include operators, that only have products which need to be booked manually by the wholesaler, and that cannot be booked over this API)

Request body: empty
Response: List of all active and available operators in the system.

getOperator: Returns a specific operator
Request: `GET /operators/{operator.id}`
Request body: empty
Response: The requested operator

customiseOperator: Allows the calling API user to update the four reserved customStrX fields of the operator. Only the customStr[1..4] fields and modified timestamp will be updated.
Request: `PUT /operators`
Request body: operator
Request required fields:

- operator.id

Response: The updated operator

getCustomerDataFormat: Returns the pax details required by this operator for a reservation
Request: `GET /operators/{operator.id}/customerDataFormat`
Request body: empty
Response: The requested operator's customerDataFormat
Note: As of 31 August 2016 this resource is deprecated. Please use the product level resource instead as this resource will be removed eventually.

getProducts: Returns the list of all products available from a specific operator
Request: `GET /operators/{operator.id}/products`
Request parameters:

- boolean `includeFullDetails` (optional, default: true)
- boolean `includeDisabled` (optional, default: false)
- boolean `includeOptions` (optional, default true, set to false to exclude optional add-ons, that cannot be purchased on their own, from the search results)
- boolean `treeView` (optional, default: false, return response formatted as a tree, with possible variations of the same tour/activity product (so called tour flavours) and optional add-ons nested underneath their parent product. See change notes from 07 November 2016 for more details.
- timestamp `modifiedSince` (optional, include this to limit results to only those products, that have been added or modified since a specific time instant, including where any change was made on the parent tour.
  Note: Unlike with the resource *searchProducts*, this filter does not take into account changes made to the operator itself, as this would obviously affect all products in this resource.)
- boolean `includeManualBooking` (optional, default: false, include products that need to be booked manually by the wholesaler, and that cannot be booked over this API)

Request body: empty
Response: List of all available products for the specified operator.

getProduct: Returns a specific product
Request: `GET /operators/{operator.id}/products/{product.id}`
Request parameters:

- boolean `includeRetailCommission` (optional, default: false, set to true to explicitly include the product- and requesting API user-specific retail commission percentage)

Request body: empty
Response: Only the specified product for the specified operator.
Note: This method has been deprecated and is only kept in place for backwards compatibility. We recommend using the getProduct method defined for the direct `/products` endpoint instead (See below, section 2.3).

customiseProduct: Allows the calling API user to update the four reserved customStrX fields of the product. Only the customStr[1..4] fields and modified timestamp will be updated.
Request: `PUT /operators/{operator.id}/products`
Request body: product
Request required fields:

- product.id

Response: The updated product
Note: This method has been deprecated and is only kept in place for backwards compatibility. We recommend using the customiseProduct method defined for the direct `/products` endpoint instead (See below, section 2.3).

getTnc: Returns the operator's terms and conditions (if exist)
Request: `GET /operators/{operator.id}/tnc`
Request body: empty
Response: The requested operator's terms and conditions

getOperatorIdCidTuples: Returns the ids and cids of all operators
Request: `GET /operators/idCidTuples`
Request body: empty
Response: A List of idCidTuple elements, each representing a single operator

2.2. Outlets

getOutlets: Returns a list of all outlets (i.e. physical shops) associated with the same outlet group as your API account.
Request: `GET /outlets`
Request parameters:

- boolean `includeFullDetails` (optional, default: true)

Request body: empty
Response: List of all outlets in the same group as the API user

getOutlets: Returns a list of all outlets (i.e. physical shops) associated with the same outlet group as your API account.
Request: `GET /outlets/{outlet.id}`
Request body: empty
Response: Outlet with the specified id

customiseOutlet: Allows the calling API user to update the four reserved customStrX fields of any outlet that is part of the same outlet group as the user itself. Only the customStr[1..4] fields and modified timestamp will be updated.
Request: PUT /outlets
Request body: outlet
Request required fields:

- outlet.id

Response: The updated outlet

2.3. Products

getProduct: Returns a specific product
Request: `GET /products/{product.id}`
Request parameters:

- boolean `includePickups` (optional, default: true, if false the response records will not include pickups, resulting in potentially much faster response times)
- boolean `includeRetailCommission` (optional, default: false, set to true to explicitly include the product- and requesting API user-specific retail commission percentage)

Request body: empty
Response: Only the specified product.

customiseProduct: Allows the calling API user to update the four reserved customStrX fields of the product. Only the customStr[1..4] fields and modified timestamp will be updated.
Request: `PUT /products`
Request body: product
Request required fields:

- product.id

Response: The updated product

searchProducts: Used to search all products with matching search criteria
Request: `GET /products/search`
Request body: empty
Request parameters:

- string `tags` (optional, matches products by their tags, separate multiple tags by comma).
  Deprecated as of 03/11/2016, use fts instead.
- string `name` (optional, matches products by their name, separate multiple search terms by comma).
  Deprecated as of 03/11/2016, use fts instead.
- string `fts` (optional, full text search query. The search value must be in single quotes. The available search fields are `tags`, `name`, `desc` and `highlights`. Query syntax is `{searchField}='{searchValue}'`.
  To chain multiple criteria you can use + for AND and `|` for OR)
  e.g `desc='Harbour Bridge'+tags='Adventure'`
  (encoded as `fts=desc%3D%27Harbour+Bridge%27%2Btags%3D%27Adventure%27`)
- string `location` (optional, limits results by distance from reference coordinates; format is `{latitude -90.0 to 90.0},{longitude -180.0 to 180.0},{search radius in kilometers}` e.g. `location=-33.875,151.222,2.5`)
- string `airport` (optional, limits results by distance from given airport; format is `{3-character IATA airport code},{search radius in kilometers}` e.g. `airport=SYD,50.5`)
- localdate `startDate` (not in the past) and `endDate` (optional, not more than 1 year in future, defaults to value of `startDate`) to filter search results to products, that have open, bookable departures in the specified date range. Results can be further limited by additional parameter integer `requiredUnits` (optional, default: 1), to exclude depatures with insufficient availability.
- timestamp `modifiedSince` (optional, include this to limit results to only those products, that have been added or modified since a specific time instant, including where any change was made on the parent tour or operator.)
- boolean `includeFullDetails` (optional, default: true)
- boolean `matchAll` (optional, default: false, if true search will only return results that match ALL of the specified `tags` and `name` criteria. Please note: `location` or `airport` filters are always applied (linked with AND), i.e. including `location` or `airport` will only return results in the specified area, regardless of `matchAll`'s value)
  Deprecated as of 03/11/2016, as only relevant for deprecated parameters `tags` and `name`.
- boolean `matchPartial` (optional, default: true, e.g. searching for tag 'diving' partially matches 'sky diving')
- boolean `includeDisabled` (optional, default: false)
- boolean `includeOptions` (optional, default true, set to false to exclude optional add-ons, that cannot be purchased on their own, from the search results)
- boolean `includePickups` (optional, default false, set to true to include the pickups of each product.
  Note: Including the full list of pickups can significantly slow down the search, so please only use where it necessary)
- string `order` (optional, default: id, allowed values id, random; specifies sort order of results)
- integer `limit` (optional, limits the number of returned search results)
- boolean `treeView` (optional, default: false, return response formatted as a tree, with possible variations of the same tour/activity product (so called tour flavours) and optional add-ons nested underneath their parent product. See change notes from 07 November 2016 for more details.
- boolean `includeManualBooking` (optional, default: false, include products that need to be booked manually by the wholesaler, and that cannot be booked over this API)

Response: Unless limited by parameter `limit`, returns all products, that match the given search criteria. Unless `treeView` is set to false, the response will only include directly bookable products, and depending on the `includeOptions` parameter, optional add-ons, in a flat list.
Note: You must define at least one search criterion (fts/name/tags/location/airport), but only one of the geographic parameters `airport` or `location` can be used at a time.

getMultipleProducts: Used to retrieve multiple products by their id or cid
Request: `GET /products/multi`
Request body: empty
Request parameters:

- string `id` (optional, matches products by their positive, big integer id, separate multiple ids by comma)
- string `cid` (optional, matches products by their positive, big integer cid, separate multiple cids by comma)
- boolean `includeFullDetails` (optional, default: true)
- boolean `includePickups` (optional, default: false, if true the response records will include pickups)

Response: All products that match the given search criteria.
Note: You must define at least one search criterion id or cid. The maximum number of records that can be queried this way is 25.

checkAvailability: Used to check real-time product availability for a specific date or date range
Request: `GET /products/{product.id}/checkAv`
Request body: empty
Request parameters:

- localdate `startDate` (required, not in the past)
- localdate `endDate` (optional, defaults to `startDate`, max date range is 7 days)
- integer `requiredUnits` (optional, number of required units, defaults to 1, value must be 1-10)

Response: An availabilityCheck element with an availability item for each requested date
Request limit: As this request is proxied to our supplier APIs and are subject to limitations, clients are by default restricted to 100 requests per minute (sliding window).

checkRates: Used to check real-time product pricing information for a specific date or date range
Request: `GET /products/{product.id}/checkRates`
Request body: empty
Request parameters:

- localdate `startDate` (optional, not in the past; if omitted rate for open dated booking is returned unless product does not allow open dated reservations)
- localdate `endDate` (optional, defaults to `startDate`, max date range is 7 days)
- string `paxAges` (optional, comma separated list of pax ages to check pricing for, e.g. '24,22' or '42,38,6'; defaults to '35' to return standard adult/rack rate)

Response: A ratesCheck element with a ratesDate item for each requested date, each containing rates items for requested pax ages
Request limit: As this request is proxied to our supplier APIs and are subject to limitations, clients are by default restricted to 100 requests per minute (sliding window).

getTnc: Returns the product's specific terms and conditions (if exist).
Request: `GET /products/{product.id}/tnc`
Request body: empty
Response: The requested product's terms and conditions
Note: If defined, these terms apply on top of those defined at the operator level.

getCategories: Returns the list of product categories
Request: `GET /products/categories`
Request body: empty
Response: List of all our product categories.

getDepartures: Used to retrieve valid product departure dates and availability, plus optional rates from Livn's cache
Request: `GET /products/{product.id}/departures`
Request body: empty
Request parameters:

- localdate `startDate` (optional, not in the past, see below for more details on behaviour if omitted)
- localdate `endDate` (optional, defaults to `startDate`, max date range is 100 days)
- boolean `includeRates` (optional, defaults to false, specifies whether the response is to include rates)
- integer `limit` (optional, limits the number of returned departures)
- integer `minUnits` (optional, excludes departures that don't have at least this number of available units)

Response: A departureCheck element with a departure item for each valid departure date in the requested date range, which in turn contains a departureRates element with pricing information, if requested.
If no `startDate` and `endDate` are specified, the system returns the first, as in soonest, open departure with any availability (i.e. `units>=1`). You can use the parameters `minUnits` and `limit` to further influence this behaviour, e.g. to return the first 3 departures with 2 or more available units use `limit=3` and `minUnits=2`.

getDeparturesMulti: Used to retrieve valid product departure dates and availability, plus optional rates from Livn's cache
Request: `GET /products/{tour.id}/departures-multi`
Request body: empty
Request parameters: same as *getDepartures*
Response: A list of departureCheck elements for every bookable product under the specified top level tour, i.e. the tour itself or all of its flavours, plus where applicable all of the tour's options.

getPaxDetails: Returns the pax details required to book this product
Request: `GET /products/{product.id}/paxDetails`
Request body: empty
Response: The requested product's paxDetails

getProductAirports: Retrieves all airports, in whose vicinity (radius 100km) there are bookable products
Request: `GET /products/airports`
Request body: empty
Response: List of airport elements, sorted by IATA code

getProductCities: Retrieves a list of the nearest cities (population > 50,000) for all bookable products
Request: `GET /products/cities`
Request body: empty
Response: List of productCity elements, sorted by name

getProductIdCidTuples: Returns the ids and cids of all products
Request: `GET /products/idCidTuples`
Request body: empty
Response: A List of idCidTuple elements, each representing a single product

2.4. Carts

getDummyCart: Returns an example or template of a new cart as it can be posted to the API
Request: `GET /carts`
Request body: empty
Response: a bare cart with 4 paxes, each selecting one product.

postCart: Allows the API user to post a selection of products making up a new reservation request. This automatically initiates the availability and rates checks, which by request can be handled asynchronously, requiring subsequent polling of the cart and monitoring the field dateAvAndRatesChecked.
Request: `POST /carts`
Request body: cart element containing all required pax details and their selection of bookable products
Required fields:

- paxes (1-10 pax elements), each pax element requires at least the pax details defined by the customer's selected product(s) (see product.paxDetails)
- cartItems (1+ cartItem elements). Each cartItem element requires:
  - paxIndex
  - productId
  - productDate (or null/omitted for open dated booking, where allowed)
  - pickupId (if product requires a pickup selection but pickupId is omitted, the system tries to select the best suitable pickup automatically)

Request headers:

- `Synchronous` (optional, default true; Requests that include this header, with a value of *false, off, no, f* or *n* (case insensitive), are processed asynchronously, i.e. you will receive a response almost immediately, before the potentially longer running exchange with the upstream reservation system API(s) has been completed. This approach requires you to subsequently poll your cart using the id from the initial response.

Response: Your cart, including its newly assigned cart.id, filled in with all data from our cache and, unless explicitly not requested, the results of a live availability and rates check with the respective supplier reservation system(s).

getCart: Returns the cart with the given id.
Request: `GET /carts/{cart.id}`
Request body: empty
Request parameters:

- boolean `includeFullDetails` (optional, default: true)

Response: the complete cart including all fields or just the minimal fields required to poll and confirm status of the cart.
Note: once the availability and rates checks are complete (timestamp field `dateAvAndRatesChecked`), all relevant details (applicable rates etcetera) and a link leading to the next checkout step below are included. The checkout remains valid for 15 minutes, after which the offered rates and availability expire and need to be checked again by re-posting the cart.

getAllTickets: Retrieves a collated PDF with all tickets for the specified cart.
Request: `GET /carts/{cart.id}/tickets`
Request body: empty
Response: PDF document as binary data with Content-Type: application/pdf

checkoutCart: Request to process the cart through checkout and make reservations with the individual operators at the offered conditions.
Request: `GET /carts/{cart.id}/checkout`
Request body: empty
Request parameters:

- string `paymentAuthorisation` (optional)

Request headers:

- `Synchronous` (optional, default true; Requests that include this header, with a value of *false, off, no, f* or *n* (case insensitive), are processed asynchronously, i.e. you will receive a response almost immediately, before the potentially longer running exchange with the upstream reservation system API(s) has been completed. This approach requires you to subsequently poll your cart using the id from the initial response.

Response: If parameter `paymentAuthorisation` is included, the value is used in lieu of a separate *authorisePayment* call. The reservation process (same as the rates and availability checks following *postCart*) involves communication with our supplier APIs and may take a while to complete. By default the checkout is handled synchronously, i.e. it will not return your response until that reservation process is completed, unless the *Synchronous* header is used to explicitly opt for asynchronous processing, in which case you need to poll the cart again, by calling *getCart* (ideally including the parameter `includeFullDetails = false`), until its status changes to:

- PENDING_AUTHORISATION Proceed to *authorisePayment*
- PENDING_CREDIT_CARD_PAYMENT Proceed to *makeCreditCardPayment*
- COMPLETED Cart will include the individual reservations along with their respective reservation items (usually one per pax and product)
- FAILED_CHECKOUT Check the *problems* element specifying the underlying cause(s)

authorisePayment: Used to re-confirm and/or authorise payment for a successfully processed cart when prompted (status = CartStatus.PENDING_AUTHORISATION). Only applies where payment gateway funds transfer is required between the API user and distributor, or manual re-confirmation is otherwise required by the API user's outlet group. Setting the payment authorisation prevents the cart from timing out and being rolled back. This stop is not necessary if a paymentAuthorisation code has already been set, during postCart or checkoutCart.
Request: GET `/carts/{cart.id}/authPayment`
Request body: empty
Request parameters:

- string `paymentAuthorisation` (required)

Response: The complete cart

customiseCart: Allows you to set the four customisable fields customStr[1..4]
Request: `PUT /carts`
Request body: cart element
Required fields:

- id, customStr1, customStr2, customStr3, customStr4 (omitted customStrX fields are set to null)

Response: The complete updated cart including the custom fields

searchCarts: Used to search all carts that were POSTed with matching *cart.customStrX* field values
Request: `GET /carts/search`
Request body: empty
Request required fields: none
Request parameters:

- string `customStr1`, `customStr2`, `customStr3`, `customStr4` (optional)

Response: All carts that match the given customStr1-4 criteria.
Note: The returned carts won't include the final reservation details, to get those call getCart directly.
getReservationsByCart: Retrieves all reservations created from the specific cart.
Request: `GET /carts/{cart.id}/reservations`
Request body: empty
Response: All reservations created from the specified cart.

getRetailTotals: Retrieves a stand-alone listing of the grand total retail commission, gross and net amounts.
Request: `GET /carts/{cart.id}/retailTotals`
Request body: empty
Response: A retailTotals element.

makeCreditCardPayment: Used to make a payment from a credit card for a successfully processed cart when prompted (status = CartStatus.PENDING_CREDIT_CARD_PAYMENT). Only applies where credit card payment is required between the retailer/API user and distributor. Making a payment prevents the cart from timing out and being rolled back.
Request: `POST /carts/{cart.id}/ccPayment`
Request body: creditCardPayment element containing all required payment details
Response: The complete cart after the payment has been processed and recorded
Note: Livn does not store the submitted credit card details in any way, shape or form, not in our database, log files or anywhere else. The card details are seamlessly relayed, via a secure connection, to Stripe for processing of your payment, and we only store the unique transaction number (com.stripe.model.Charge.id) returned by their API, which cannot be used to retrieve your credit card details. All Livn Group applications that process credit card details in this form, operate in full compliance with PCI Data Security Standards.

2.5. Reservations

getReservation: Returns the complete reservation with the given id.
Request: `GET /reservations/{reservation.id}`
Request body: empty
Response: the complete reservation including all fields and reservationItems

searchReservationsByPaxNameOrEmail: Retrieves all reservations belonging to the requesting API user's outlet group, by pax name/email, ignoring case and including partial matches.
Request: `GET /reservations/search`
Request parameters:

- string `q` (mandatory, pax last name or email)

Request body: empty
Response: list of all matching complete reservations including all fields and reservationItems

getReservationFlatView: Returns the reservation with the given id and reduced details.
Request: `GET /reservations/{reservation.id}/flatView`
Request body: empty
Response: the specified reservation reduced to the following details: id, cartCustomStr1, cartCustomStr2, cartCustomStr3, cartCustomStr4, reservationItems [id, paxId, paxIndex, productName, paxName, externalResId, grossRate, productCustomStr1, productCustomStr2, productCustomStr3, productCustomStr4, paxCustomStr1, paxCustomStr2, paxCustomStr3, paxCustomStr4]

getCancellationQuote: Requests a quote for the cancellation of a previously complete reservation
Request: `GET /reservations/{reservation.id}/cancellationQuote`
Request body: empty
Response: a cancellation element outlining the cost of cancelling the reservation at the current stage. Includes a cancellation quoteId and generic cancellationReason (which should be overwritten before posting back). Quote is to be posted back to commence actual cancellation. After 5 minutes quote expires and needs to be re-requested.

getAllTickets: Retrieves a collated PDF with all tickets for the specified reservation.
Request: `GET /reservations/{reservation.id}/tickets`
Request body: empty
Response: PDF document as binary data with Content-Type: application/pdf

2.6. Cancellations

postCancellation: Allows the API user to post back a cancellation quote, initiates actual asynchronous cancellation process.
Request: `POST /cancellations`
Request body: Original cancellation quote. You may override the field cancellationReason, as well as individual values for cancellationItem.retailRateOverride (i.e. the pax rate you wish to use on tickets).

> Note: If your API user has been granted the required permission by your wholesaler/distributor (`ApiUserRole OVERRIDE_CANCELLATION`), you may further override the individual quoted cancellationItem.operatorRate values with a non-negative value. In this case the operator claimable wholesaleNetRate is changed to the supplied amount and wholesaleCommissionAmount is set to 0.00. Please note: The operator currency cannot be changed, i.e. the override amount has to be supplied in the operator currency form the cancellation quote. For you convenience you may also override the retailRate in a similar fashion, doing so will reduce the recorded retailCommissionAmount to 0.00.

Request headers:

- `Synchronous` (optional, default true; Requests that include this header, with a value of *false*, *off*, *no*, *f* or *n* (case insensitive), are processed asynchronously, i.e. you will receive a response almost immediately, before the potentially longer running exchange with the upstream reservation system API(s) has been completed. This approach requires you to subsequently poll your cancellation using the id from the initial response.

Response: Your cancellation including its newly assigned cancellation.id, and unless explicitly not requested, the full details of the cancellation.

getCancellation: Used to poll the cancellation with the given id.
Request: `GET /cancellations/{cancellations.id}`
Request body: empty
Response: the complete cancellation including all fields and individual cancellationItems

2.7. Paxes

customisePax: Allows you to set the four customisable fields customStr[1..4]
Request: `PUT /paxes`
Request body: pax element
Required fields:

- id, customStr1, customStr2, customStr3, customStr4 (omitted customStrX fields are set to null)

Response: The complete updated pax including the custom fields

getPax: Used to retrieve the individual pax record with the given id.
Request: `GET /paxes/{pax.id}`
Request body: empty
Response: the complete pax including all details

2.8. Reports

getSalesReport: Retrieves a full sales report for the specified period of time
Request: GET `/reports`
Request body: empty
Request parameters:

- timestamp `startDate`
- timestamp `endDate`
- boolean `includeDemoSales` (optional, default: false, include/omit sales made with demo operators)
- string `customStr1`, `customStr2`, `customStr3`, `customStr4` (optional, if specified, returns only such sales, that were created from a *cart*, that was POSTed with the specified *customStrX* value. Note: It is currently not possible to explicitly search for null values, and a default/null value for either of these parameters, is interpreted as the filter being omitted altogether.)

Response: The salesReport with all individual sales completed in the specified period of time

getSalesReport: Retrieves a full sales report for the specified period of time as a comma delimited flat file
Request: GET `/reports`
Produces: text/csv
Request body: empty
Request parameters:

- timestamp `startDate`
- timestamp `endDate`
- boolean `includeDemoSales` (optional, default: false, include/omit sales made with demo operators)
- boolean `includeColumnHeaders` (optional, default: true, include/omit a header row declaring column names)
- string `customStr1`, `customStr2`, `customStr3`, `customStr4` (optional, if specified, returns only such sales, that were created from a *cart*, that was POSTed with the specified *customStrX* value. Note: It is currently not possible to explicitly search for null values, and a default/null value for either of these parameters, is interpreted as the filter being omitted altogether.)

Response: The salesReport with all individual sales completed in the specified period of time as a comma delimited CSV flat file. Each row represents one sale.
Note: This method has been hidden from our Swagger API documentation, as it was hiding the equally named method *getSalesReport* above, that only differs by its response MIME type (`application/json` or `application/xml`).
For this reason the method getSalesReportCsv has been added, which returns the same response as this method, and could be included in the Swagger API specification document.

getSalesReportCsv: Retrieves a full sales report for the specified period of time as a comma delimited flat file
Request: GET `/reports/csv`
Produces: text/csv
Request body: empty
Request parameters:

- timestamp `startDate`
- timestamp `endDate`
- boolean `includeDemoSales` (optional, default: false, include/omit sales made with demo operators)
- boolean `includeColumnHeaders` (optional, default: true, include/omit a header row declaring column names)
- string `customStr1`, `customStr2`, `customStr3`, `customStr4` (optional, if specified, returns only such sales, that were created from a *cart*, that was POSTed with the specified *customStrX* value. Note: It is currently not possible to explicitly search for null values, and a default/null value for either of these parameters, is interpreted as the filter being omitted altogether.)

Response: The salesReport with all individual sales completed in the specified period of time as a comma delimited CSV flat file. Each row represents one sale.

getVoucherClaimsReportByInvoiceNumber: Retrieves a full report of all items claimed on a specific invoice
Request: GET `/reports/voucherclaims/{invoiceNumber}`
Request body: empty
Response: The voucherClaimsReport with all individual voucherClaims
Note: This resource is only available if API user is part of the wholesaler/distributor company.

getVoucherClaimsReportByInvoiceNumber: Retrieves a full report of all items claimed on a specific invoice as a comma delimited flat file
Request: GET `/reports/voucherclaims/{invoiceNumber}`

Produces: text/csv
Request body: empty
Request parameters:

- boolean `includeColumnHeaders` (optional, default: true, include/omit a column header row in the CSV file)

Response: The voucherClaimsReport with all individual voucherClaims as a comma delimited CSV flat file.
Each row represents one claimed voucher.
Note: This resource is only available if API user is part of the wholesaler/distributor company.

getVoucherClaimsReportsByDateRange: Retrieves a list of full reports of all items claimed in a specific period of time
Request: GET `/reports/voucherclaims/{invoiceNumber}`
Request body: empty
Request parameters:

- timestamp `startDate`
- timestamp `endDate`

Response: A list of voucherClaimsReports with all individual voucherClaims, for all claims that were invoiced in the given time.
Note: This resource is only available if API user is part of the wholesaler/distributor company.

2.9. Tickets resources

getTicket: Retrieves a specific reservation item's ticket.
Request: `GET /tickets/{reservationItem.id}`
Request body: empty
Response: PDF document as binary data with Content-Type: application/pdf

getTicketStyle: Retrieves information about the style of the ticket PDFs we generate: the SVG vector format company logo incl. print and screen dimensions, and the ticket accent colour.
Request: `GET /tickets/style`
Request parameters: None
Request body: empty
Response: Element of type *ticketStyle*.

updateTicketStyle: Customise the style of the ticket PDFs we generate, by providing an SVG vector format company logo, metric print dimensions (millimetres), and the ticket accent colour. Any image not already hosted by Livn, is sideloaded to our CDN and the finished resource details are returned similar to the above call *getTicketStyle*.
Request: `PUT /tickets/style`
Request parameters: None
Request body: Element of type *ticketStyle* containing a freely accessible *logoUrl* from where to sideload the SVG, print dimensions *logoWidth* and *logoHeight* (in mm) and *accentColor* (the RGB hex colour code of ticket accent colour)
Send the existing logo URL if you only wish to update the print size and/or accent colour, without changing the logo itself.
Response: Element of type *ticketStyle*.

2.10. AWS publish & subscribe notification resources

getInventorySNSTopicARN: Retrieves the Amazon Resource Name (ARN) of the AWS Simple Notification Service (SNS) Topic, used by the calling user's distributor, to publish notifications about inventory related events, e.g. Operator deleted, Tour updated, etc.
Request: `GET /aws/inventory`
Request body: empty
Response: Element of type *arn*, identifying the SNS topic required to establish a reverse notification channel.

grantPermissionToSubscribeToInventorySNSTopic: Grants the AWS Simple Queue Service (SQS) queue with the specified Amazon Resource Name (ARN), the permission to self-subscribe to the AWS Simple Notification Service (SNS) Topic returned by *getInventorySNSTopicARN*.
Request: POST `/aws/inventory`
Request body: Element of the type *arn*, identifying the SQS queue owned by the caller, forwhich to grant the right to subscribe to the calling API user's distributor's SNS topic.
Response: Element of type *arn*, identifying that SNS topic (Same as *getInventorySNSTopicARN*).

2.11. Distributor/Wholesaler resources

getDistributorWholesaler: Returns details about the calling API user's distributor/wholesaler, e.g. Livn Holidays.
Request: `GET /distributor`
Request body: empty
Response: Element of the type *distributor.*

getDistributorTncHtml: Returns only the distributor's terms and conditions, if these are defined, contained in a complete HTML document.
Request: `GET /distributor/tnc`
Request header: None or `Accept:text/html`
Request body: empty
Response: The wholesaler terms and conditions as complete, self-contained HTML document. (The terms found in the response of *getDistributorWholesaler*, in the field *distributor.tnc.content* are also formatted in HTML, but come without any HTML header, styles etc, as pure content)

2.12. User resources

getUser: Retrieves detailed information about the API *user*, it's parent *outlet* (travel agency/shop), *outletGroup* (agency consortium) and *distributor* (wholesaler).
Request: `GET /user`
Request parameters: None
Request body: empty
Response: Element of type *user*, including its parent *outlet* (where applicable), *outletGroup* and *distributor*.

2.13. Public resources (no authentication required, except for *login*)

getAddressFormats: Used to retrieve the list of supported operator address format values. Does not require authentication.
Request: `GET /public/addressFormats`
Request body: empty
Response: the list of addressFormat data type values

getCountries: Used to retrieve the list of valid countries (ISO codes and display names).
Request: `GET /public/countries`
Request body: empty
Response: the list of country data type values

getDocumentationPdf: Used to retrieve the up to date API manual PDF (this document).
Request: `GET /public/api-docs-pdf`
Request body: empty
Response: the documentation PDF

getLanguages: Used to retrieve the list of valid languages (ISO codes and display names).
Request: `GET /public/languages`
Request body: empty
Response: the list of language data type values

getServerTime: Used to retrieve the current system time on the API server. Can be used as a ping endpoint to test general accessibility to the server. Does not require authentication.
Request: `GET /public/time`
Produces: text/plain
Request body: empty
Response: the current server time as plain text

getTags: Used to retrieve the list of tag strings.
Request: `GET /public/tags`
Request body: empty
Response: the list of recommended search tags

login: Returns a JSON Web Token for subsequent use in HTTP `Authorization` header
Request: POST `/public/login`
Request body: loginCredentials (type comprised of loginName and password)
Response: Element of type loginToken, includes field token with encrypted JWT string.
Note: See 1.4 on how to further handle this token.

3. Data types

The following is a list of all data types in alphabetical order.

Please note: The Livn API is undergoing constant development to be able to connect users with a growing number of supplier reservation system. While care is taken to minimise structural changes between revisions, from time to time it may be necessary to introduce new fields to these data types or rename existing fields to clear up any prevalent confusion.

Further, other fields included until now, may have proven to be unnecessary to our API users and are to be deprecated, i.e. their use is being discouraged, before dropping these elements altogether in future revisions.
Currently deprecated fields are marked in red and should be expected to be removed in the next major update.
Please let us know if you require a deprecated field for your ongoing operations.

The added character * symbolises fields that are always included in documents returned by the API, unless manually suppressed in the request.
Fields marked with ^{S} are only included once the data record has been saved in our system.
Wholesaler information fields marked ^{W} are only included if API user is part of the wholesaler/distributor company.
Other, more specific notations are explained directly in the type description table.

Fields not marked with an asterisk* are generally nullable, with null values generally being omitted from the response in order to preserve bandwidth.

Enumerated string type: addressFormat	
Value	**Description**
NONE	No address details are required
COUNTRY	At least the country of residence is required
FULL	At least address1, city and country are required

Complex type: airport		
Field Name	**Description**	**Data Type**
iata*	3 letter IATA airport code	string
name*	Airport name	string
country*	2 letter ISO 3166 country code	string
city*	Airport's associated city/municipality	string
state	State/county the airport is located in	string
latitude*	The latitude of the airport	double
longitude*	The longitude of the airport	double

Complex type: arn		
Field Name	**Description**	**Data Type**
value*	Amazon Resource Name, short ARN, a type of identifier used by Amazon Web Services for arbitrary resources. See AWS Documentation on ARNs	string

Complex type: availability			
Field Name	**Description**	**Data Type**	**Char. Limit**
available*	Denotes whether the required number of units is available to be booked through the API	boolean	

availableUnits	Where available to us, the actual number of available units	integer	
checkFailed*	Signalises when the availability check with the supplier system failed	boolean	
requestOnly*	If true the departure cannot be booked through the API. It may be possible to make a reservation through direct communication with the operator. If value is true, the field available will always be false.	boolean	
created*	Time and date the specific availability check was completed	timestamp	
date*	The specific date checked	localdate	

Complex type: availabilityCheck			
Field Name	**Description**	**Data Type**	**Char. Limit**
availabilities*	List of all availabilities in the specified date range	availability sequence	
created*	Time and date the availability check as a whole was initiated	timestamp	
endDate*	End date of checked period	localdate	
productId*	ID of the requested product	long	
productCid*	Central CID of the requested product	long	
requiredUnits*	Number of required units	integer	
startDate*	Start date of checked period	localdate	

Complex type: awsNotification		
Field Name	**Description**	**Data Type**
operatorCid	Central CID of the affected operator	long
operatorId	ID of the affected operator	long
parentCid	Central CID of the affected product's parent tour	long
parentId	ID of the affected product's parent tour	long
productCid	Central CID of the affected product	long
productId	ID of the affected product	long
type*	Enumerated type of the event, e.g. *OPERATOR_DELETE*, or *PRODUCT_UPDATE*	awsNotificationType

Enumerated string type: awsNotificationType	
Value	**Description**
OPERATOR_UPDATE	An operator has been modified in a way affecting API clients selling its products. Users who cache inventory are advised to update their cache of the specified operator, and ideally all of the operator's products they use.
OPERATOR_DELETE	An operator has been deleted. Users are advised to delete the specified operator from their inventory, along with all of its products. Note: You will also receive separate *PRODUCT_DELETE* notifications for each of the operator's products.

PRODUCT_UPDATE	A product has been modified in a way affecting API clients selling it. Users are advised to update their cached record of the specified product, and where applicable, any related child products (flavours/options) they use.
PRODUCT_DELETE	A product has been deleted. Users are advised to delete the specified product, and where applicable, any related child products (flavours/options) they use. Note: You will also receive separate *PRODUCT_DELETE* notifications for any such related child products.
PRODUCT_INSERT_TOUR	A new tour has been added to the distributor's inventory under the specified operator. Users who cache Livn API's inventory are invited to import the new tour into their inventory/cache.
PRODUCT_INSERT_FLAVOUR	A new flavour (=variant) of the specified parent tour has been added to the distributor's inventory. Users who cache inventory are invited to import the new tour flavour into their records.
PRODUCT_INSERT_OPTION	A new option (=add-on service or product) has been added to the specified parent tour. Users who cache Livn API's inventory are invited to import the new optional add-on into their records.

Complex type: cancellation

Field Name	**Description**	**Data Type**	**Char. Limit**
cancellationItems	The individual cancellation items, each representing the cancellation of an original reservationItem	cancellationItem sequence	
cancellationReason*	A short description of the reason for the cancellation	string	255
confirmed	Date and time the cancellation was confirmed	timestamp	
created*	Date and time of record creation	timestamp	
errors	Reason why the cancellation failed	string	
globalRef	Globally unique reference number	string	255 (realistically < 20)
grandTotalOperator*	The grand total of the operator supplied product rates	decimal	
grandTotalRetail*	The grand total of the recommended retail prices	decimal	
grandTotalRetailOverride*	The grand total of the retail prices as overridden by the retailer	decimal	
handlingSurchargePercentage[W]	A potential handling surcharge applied to the operator supplied rate. Primarily intended to cover currency exchange losses on retailer to wholesale payments.	decimal	
id[S]	ID of this cancellation	long	
idExternal	ID of this cancellation in the supplier reservation system	string	255
notes	Operator notes	string	1000
numberOfPax*	The number of pax on the cancellation	integer	
operatorCurrency*	Operator's currency	currency	
operatorId*	ID of this cancellation's operator	long	
operatorCid*	Central CID of this cancellation's operator	long	
productDate	Travel date of original reservation	localdate	
quoteId[Q]	The unique identifier for a newly generated cancellation quote. Needs	string	36 (Version 4 UUID)

	to be included when posting back the cancellation to proceed with the actual cancellation process		
reservation*	A hyperlink back to the original reservation	URI string	depends on API base URL, generally under 255
reservationId*	ID of the original reservation	long	
retailCommissionTotal*	The sum of retail commission paid on the cancellation	decimal	
retailCurrency*	The retailer's (i.e. API user) currency	currency	
status*	The progress/status of this cancellation	cancellationStatus	
wholesaleCommissionTotal^{W}	The sum of wholesaler commissions paid on the cancellation.	decimal	
Q: Only included in a newly requested cancellation quote			

Complex type: cancellationItem			
Field Name	**Description**	**Data Type**	**Char. Limit**
created*	Date and time of record creation	timestamp	
date	Travel date of the original reservation item	localdate	
id^{S}	ID of this cancellation	long	
idExternal	ID of the cancellation item in the supplier reservation system	string	255
operatorNote	Operator notes, i.e. a message to be conveyed to the operator	string	255
operatorRate*	The cancellation cost quoted by the operator	decimal	
paxId*	ID of the pax record	long	
paxIndex*	Index of the pax based on the initial *cart.paxes*, 0 based	integer	
paxName	Full name of the pax	string	255
paxNote	Pax notes, i.e. a message to the pax, that should be included with the ticket	string	255
paxNumber*	Number of the pax as part of the cancellation (e.g. pax 2 of 3)	integer	
pickupId	ID of the original reservation item's selected pickup	long	
productCode*	Code of the cancelled product	string	255
productCustomStr1	Customisable string (*customStr1*) of the cancelled product at the time of cancellation	string	255
productCustomStr2	See above	string	255
productCustomStr3	See above	string	255
productCustomStr4	See above	string	255
productId*	ID of the cancelled product	long	
productCid*	Central CID of the cancelled product	long	
productName*	The name of the cancelled product, prefixed with *CANCELLED-*	string	255
retailCommissionAmount*	The retail commission paid on the	decimal	

	cancellation ticket		
retailRate*	The recommended retail price of the cancellation item	decimal	
retailRateOverride*	The retail price as overridden by the retailer	decimal	
ticketInfo	Additional information to include on the ticket. Normally this will be a standardised text for all cancellation tickets.	string	2000
voucherId*	Ticket/voucher number	string	19
wholesaleCommissionAmount^{W}	The wholesaler commission paid on the cancellation ticket.	decimal	
wholesaleNetRate^{W}	The resulting wholesale net rate = operatorRate-wholesaleCommission	decimal	

Complex type: cancellationPolicyCondition		
Field Name	**Description**	**Data Type**
feePerc*	Percentage of the original reservation values charged/withheld in the event of a cancellation requested within the specified number of hours till departure	decimal
hoursTillDeparture*	Hours till departure	integer

Enumerated string type: cancellationStatus	
Value	**Description**
CONFIRMED	Cancellation confirmed by the supplier
FAILED	Cancellation failed. Check *cancellation.errors* for details
PENDING	Cancellation yet to be confirmed with & by the supplier

Complex type: cart			
Field Name	**Description**	**Data Type**	**Char. Limit**
cartItems^{*F}	A list of individual cart items (product selections)	cartItem sequence	
checkout^{C}	A hyperlink leading to the checkout step when applicable	URI string	
consultantName	Full name of the selling consultant where used in a face to face sale scenario.	string	255
created*	Date and time of record creation	timestamp	
currency^{*F}	The retailer's (i.e. API user) currency. Deprecated, use cart.retailCurrency.	currency	
customStr1	Customisable string. Can be set at the time the cart is posted or using *customiseCart* method	string	255
customStr2	See above	string	255
customStr3	See above	string	255
customStr4	See above	string	255
dateAvAndRatesChecked^{C}	Date and time when live availability was checked and the proposed rates were retrieved from the supplier. Conditions are valid for 15 minutes after which cart expires.	timestamp	
datePendingAuth^{C}	Signalises the cart requires payment authorisation	timestamp	

	(=confirmation) by the API user to complete the sale. By this time the cart was successfully checked out with the supplier(s) and will time out 40 minutes later unless confirmed.		
dateSold[C]	Date and time when the sale was completed (i.e. checked out and, where applicable, confirmed by user)	timestamp	
dateTimedOut[C]	Date and time when the cart timed out. At this time all previously checked out reservations are rolled back	timestamp	
expired[*]	Signalises that the cart has expired before being taken to checkout at the proposed conditions	boolean	
hasClaimedReservationItems[WR]	Signalises whether any reservationItem created from this cart has been claimed by the operator.	boolean	
id[S]	ID of this cancellation	long	
modified	Date and time the database record was last modified	timestamp	
paxes[*F]	An ordered list of all customers involved in the sales session	pax sequence	
paymentAuthorisation	Payment authorisation / confirmation code. Depending on the API user's outlet group's payment mode setting this can be a Payment Gate code, shop pseudo code or arbitrary string. Please confirm with LIVN	string	255
problems	Gives a detailed description of any problems encountered during cart creation, availability/rates check or checkout	problem sequence	
reservations[R]	A list of all reservations that were created from this cart.	reservation sequence	
retailCommissionPerc[F]	The applicable retail commission percentage for the sale	decimal	
retailCurrency[*F]	The retailer's (i.e. API user) currency	currency	
retailRef	A freely typed field the API calling retailer can use to record their reference number. Please note: This reference is applied to the entire cart, which may result in multiple reservations with one or more suppliers. This reference is passed on to the suppliers, alongside a unique reference Livn generates for each booking. See *reservation.globalRef*	string	**50**
retailTotals[CF]	The grand total retail commission, net and gross amounts. Only included when status is not NEW, EXPIRED, IN_ASYNC_LIVE_CHECKS or FAILED_LIVE_CHECKS	retailTotals	
status[*]	The retailer's (i.e. API user) currency	cartStatus	

F: These fields/sequences can be omitted by specifying the request parameter `includeFullDetails=false` where available, in order to reduce load and response time, when data is not actually required.
C: Conditional fields, only included when actually applicable, e.g. the cart is ready to be checked out, or is awaiting payment authorisation.
R: Only displayed once the sale it completely finalised (checked out and where applicable, payment authorised). Omitted in results of method searchCarts (response could potentially include a huge number of carts)

Complex type: cartItem			
Field Name	**Description**	**Data Type**	**Char. Limit**
cancellationPolicy	ordered list of conditions, each specifying the cancellation fee percentage, that applies for cancellations made within a certain range of	cancellationPolicyCondition sequence	

	hours till departure. (The list is sorted from shortest to longest time to departure, so the first matching condition would apply at the time of cancellation).		
currency*	The retailer's (i.e. API user) currency. Deprecated, use cart.retailCurrency.	currency	
duration	Duration of the selected product in milliseconds	long	
handlingSurchargePercentage[W]	A potential handling surcharge applied to the operator supplied rate. Primarily intended to cover currency exchange losses on retailer to wholesale payments.	decimal	
id[S]	ID of the cart item record	long	
levies	A list of levies incurred on top of the retail rate. These are paid directly to the operators or their subcontractors	levy sequence	
leviesTotal	The total sum of applicable local levies.	decimal	
operatorCurrency*	The operator's currency	currency	
operatorCustomStr1	Copy of the selected product's operator's *customStr1* at the time of cart creation	string	255
operatorCustomStr2	See above	string	255
operatorCustomStr3	See above	string	255
operatorCustomStr4	See above	string	255
operatorId*	ID of the selected product's operator	long	
operatorCid*	Central CID of the selected product's operator	long	
operatorNote	Any comments to be passed onto the operator	string	255
operatorRate*	The rack rate as supplied by the operator	decimal	
paxCount*	If >1 specifies the cartItem's product, while sold and linked to a single specific pax, is actually to be used/occupied by multiple guests. This is only viable for products that have a *product.maxPaxCount* > 1. Usually these are products that include accommodation for more than one pax but are sold and priced "by the room".	integer	
paxIndex*	The 0-based index of the cartItem's pax in *cart.paxes*	integer	
paxNote	Any comments intended for the pax	string	255
pickupId[P]	ID of the selected pickup, i.e. pickup.id (not pickupPoint.id)	long	
pickupPointAddress	Address of the selected pickup point	string	255
pickupPointName[P]	Name of the selected pickup point	string	255
pickupTime[P]	Local time of the selected pickup	localtime	
problems	Gives a detailed description of any problems regarding the specific item, encountered during posting the cart, availability/rates check or checkout	problem sequence	
productCustomStr1	Copy of the selected product's *customStr1* at the time of cart creation	string	255
productCustomStr2	See above	string	255
productCustomStr3	See above	string	255
productCustomStr4	See above	string	255
productDate	Travel date of the selected product, null for open dated sale	localdate	

productId*	ID of the selected product	long	
productCid*	Central CID of the selected product	long	
productName*	The name of the selected product	string	255
retailCommissionPerc*	Retail commission percentage	decimal	
retailRate*	The recommended retail rate	decimal	
retailRateOverride*	The actual retail rate as overridden by retailer (= API user)	decimal	
retailRef	A freely typed field the API calling retailer can use to record their reference number. Please note: Unlike *Cart.retailRef*, this reference is applied to the individual cart item. Also this reference is currently not passed on to the supplier.	string	**50**
wholesaleCommissionAmount[W]	The agreed wholesale commission	decimal	
wholesaleNetRate[W]	The resulting wholesale net rate = operatorRate-wholesaleCommission	decimal	
P: Always included IF a valid pickup id has been specified. Please note that currently the pickup point address can unfortunately not be provided for all operators (depending on supplier system)			

Enumerated string type: cartStatus	
Value	**Description**
NEW	A brand new cart, which hasn't been posted and validated yet and subsequently has no ID. Next action: POST this cart to the API.
IN_ASYNC_LIVE_CHECKS	A cart currently going through validation and the asynchronous rates and availability checks with the supplier reservation system(s). Next action: Poll cart for status change.
FAILED_LIVE_CHECKS	Signalises a problem during validation or the rates and availability checks. Next action: Cart becomes obsolete.
READY_FOR_CHECKOUT	A cart, that has passed validation and the rates and availability checks and is ready to be taken through the checkout. Next action: Take the cart into checkout before it expires, by following the included checkout link.
EXPIRED	A cart, that passed the initial validation and live checks phase, but was not taken to checkout in the allowed time (default: 15 minutes). Next action: Cart becomes obsolete.
IN_ASYNC_CHECKOUT	Cart is currently going through checkout, i.e. reservations are being written asynchronously to the external booking system(s). Next action: Poll cart for status change.
FAILED_CHECKOUT	A problem occurred during checkout. Next action: Cart becomes obsolete.
PENDING_AUTHORISATION	Confirmed reservations have been made with the external booking systems and the transaction is waiting for your re-confirmation. Failure to re-confirm within the allowed time (default: 40 minutes) will cause the entire transaction to be rolled back and all external reservations to be cancelled. Next action: Re-confirm successful checkout and abort rollback timeout by calling method *authorisePayment*.
PENDING_CREDIT_CARD_PAYMENT	Confirmed reservations have been made with the external booking systems and the transaction is waiting for your credit card payment. Failure to finalise payment within the allowed time (default: 40 minutes) will cause the entire transaction to be rolled back and all external reservations to be cancelled. Next action: Make credit card payment and abort rollback timeout by POST-ing payment details to <code>/carts/{cartId}/ccPayment</code>.
TIMED_OUT_AFTER_CHECKOUT	The transaction has been rolled back and external reservations have been cancelled, because the pending authorisation/re-confirmation was not given in time.

	Next action: Cart becomes obsolete.
COMPLETED	The checkout has been completed (and where necessary re-confirmed). Next action: None as part of checkout. Ready to retrieve reservation details, print tickets, cancel reservations etc.

Complex type: category			
Field Name	**Description**	**Data Type**	**Char. Limit**
id*	ID of this cancellation	long	
name*	Name of the category	string	50

Complex type: country			
Field Name	**Description**	**Data Type**	**Char. Limit**
code*	2 letter ISO 3166-1 alpha-2 country code	string	2
names*	Map of 2 letter ISO 639-1 language codes to their respective country display name	sequence of key-value entry elements	

Complex type: creditCardPayment			
Field Name	**Description**	**Data Type**	**Char. Limit**
amount*	The payment amount. Must match cart.retailTotals.netAmount, which is the cart's grand total retail price in the client's retail currency minus any retail commission	decimal	
currency*	The payment currency, must match the retailer's (i.e. API user) currency	currency	
cvc*	The card security code, depending on the card type 3 or 4 digit numeric string	string	4
expMonth*	Card expiration month (1-12)	integer	
expYear*	Card expiration year (e.g. 2017)	integer	
number*	Credit card number	string	12-19

Extended string type: currency
A currency as represented by its 3-letter ISO 4217 currency code.

Complex type: customerDataFormat		
Field Name	**Description**	**Data Type**
addressFormatFirst*	The scope of address details required from the first pax in a group	addressFormat
addressFormatRest*	The scope of address details required from all remaining pax in a group	addressFormat
dobFirst*	Specifies whether the full date of birth of the first pax in a group is required	boolean
dobRest*	Specifies whether the full date of birth of all remaining pax in a group is required	boolean
emailFirst*	Specifies whether the email address of the first pax in a group is required	boolean

emailRest*	Specifies whether the email address of every remaining pax in a group is required	boolean
languageFirst*	Specifies whether the preferred language spoken by the first pax in a group is required	boolean
languageRest*	Specifies whether the preferred language spoken by all remaining pax in a group is required	boolean
mobileFirst*	Specifies whether the mobile number of the first pax in a group is required	boolean
mobileRest*	Specifies whether the mobile number of every remaining pax in a group is required	boolean
nameFirst*	Specifies whether the full name (salutation, first name and last name) of the first pax in a group is required	boolean
nameRest*	Specifies whether the full name (salutation, first name and last name) of all remaining pax in a group is required	boolean
nationalityFirst*	Specifies whether the nationality (as ISO country code) of the first pax in a group is required	boolean
nationalityRest*	Specifies whether the nationality (as ISO country code) of all remaining pax in a group is required	boolean
passport*	Specified whether the passportNumber, passportExpiry and passportIssued dates are required from all pax	boolean
phoneFirst*	Specifies whether the phone number of the first pax in a group is required	boolean
phoneRest*	Specifies whether the phone number of every remaining pax in a group is required	boolean
Note: The xxxFirst values always relate to the first pax in an individual reservation's group. It may be a different person from the overall first pax in a larger cart, containing multiple products, if not all paxes wish to purchase every product in that cart.		

Enumerated string type: demoMode	
Value	**Description**
BOTH	Show both Demo and Live operators
DEMO_ONLY	Only list Demo operators
NO_DEMO	Hide Demo Operators

Complex type: departure			
Field Name	**Description**	**Data Type**	**Char. Limit**
availableUnits*	The number of available units	integer	
date*	The specific departure date	localdate	
departureRates[R]	A wrapper element containing all pricing information specific to the departure	departureRates element	
status*	An emumeration type field specifying whether the departure can be booked or not	departureStatus	
R: Only included if getDepartures request parameter *includeRates=true*.			

Complex type: departureCheck			
Field Name	**Description**	**Data Type**	**Char. Limit**
childCutoffAge[R]	Age at which pax can no longer be considered to be children. E.g. a value of 12 means pax under the age of 12 will be processed as children (infants if under *infantCutoffAge*), above as adults	integer	
created*	Time and date the availability check as a whole was initiated	timestamp	
currExchangeRate[R]	Currency exchange rate, where *operatorCurrency* and *retailCurrency* differ	decimal	

currExchangeSurchargePerc[R]	Agreed currency conversion surcharge percentage to offset distributor exchange losses	decimal	
departures[*]	List of valid departures	departure sequence	
endDate[*]	End date of checked period	localdate	
infantCutoffAge[R]	Age at which pax can no longer be considered to be infants. E.g. a value of 2 means pax under the age of 2 will be processed as infants, 2 and above as children or adults based on *childCutoffAge*	integer	
minAge[R]	The minimum bookable pax age	integer	
operatorCid[*]	Central CID of the product's operator	long	
operatorCurrency[R]	The Operator's currency	currency	
productId[*]	ID of the requested product	long	
productCid[*]	Central CID of the requested product	long	
productName	Name of the requested product	string	255
productStartTime	Local start/departure time of product	localtime	
productType[*]	The enumerated product type TOUR, FLAVOUR or OPTION	productType	
retailCommissionPerc[R]	Retail commission percentage	decimal	
retailCurrency[R]	Number of required units	integer	
startDate[*]	Start date of checked period	localdate	
R: Only included if getDepartures request parameter *includeRates=true*.			

Complex type: departureRates			
Field Name	**Description**	**Data Type**	**Char. Limit**
levies	A list of levies incurred on top of the retail rate. These are payable by the pax, directly to the operators or their subcontractors (in local currency)	levy sequence	
leviesTotal	The total of all local levies	decimal	
operatorRateAdult[*]	The adult/rack rate supplied by the operator	decimal	
operatorRateChild	The children's rate supplied by the operator, where available	decimal	
operatorRateInfant	The infant's rate supplied by the operator, where available	decimal	
retailCommissionAmountAdult[*]	The retail commission for adults, in *retailCurrency*	decimal	
retailCommissionAmountChild	The retail commission for children, in *retailCurrency*	decimal	
retailCommissionAmountInfant	The retail commission for infant, in *retailCurrency*	decimal	
retailNetRateAdult[*]	The payable retail net rate (=retailRate-commission) for adults, in *retailCurrency*	decimal	
retailNetRateChild	The payable retail net rate (=retailRate-commission) for children, in *retailCurrency*	decimal	
retailNetRateInfant	The payable retail net rate (=retailRate-commission) for infants, in *retailCurrency*	decimal	
retailRateAdult[*]	The recommended retail price for	decimal	

	adults, in *retailCurrency*		
retailRateChild	The recommended retail price for children, in *retailCurrency*	decimal	
retailRateInfant	The recommended retail price for infants, in *retailCurrency*	decimal	
wsCommAmountAdult[*W]	The wholesale commission generated by this product purchase for adults	decimal	
wsCommAmountChild[W]	The wholesale commission generated by this product purchase for children	decimal	
wsCommAmountInfant[W]	The wholesale commission generated by this product purchase for infants	decimal	
wsNetRateAdult[*W]	The resulting adult wholesale net rate (operatorRateAdult-wsCommAmountAdult)	decimal	
wsNetRateChild[W]	The resulting children's wholesale net rate (operatorRateChild-wsCommAmountChild)	decimal	
wsNetRateInfant[W]	The resulting infant wholesale net rate (operatorRateInfant-wsCommAmountInfant)	decimal	
W: Only included if API user is part of the wholesaler company.			

Enumerated string type: departureStatus	
Value	**Description**
OPEN	Product operates and can be booked (sufficient availability provided)
CLOSED	Product/run is closed for booking
REQ_ONLY	Bookings only by request, with no departure guarantee at the time of booking. Not bookable over the API.

Complex type: distributor			
Field Name	**Description**	**Data Type**	**Char. Limit**
address1[*]	Main address of the distributor/wholesaler	string	255
address2	Address continued	string	255
businessNumber	Business number (e.g. ABN)	string	255
city[*]	City the distributor is located	string	255
code	A short code, e.g. LIVN	string	12
companyName	Company name of the distributor, can be different from name	string	255
contactName	Name of the first line contact person at the distributor	string	255
country[*]	Country the distributor is located in. 2 letter ISO 3166-1 alpha-2 code.	string	2
created[*]	Date and time of record creation	timestamp	
email[*]	The distributor's primary email address	string	255
fax	The distributor's fax number	string	255
id[*]	Unique ID of the distributor	long	

language*	The distributor's primary operations language. ISO language code	string	2
modified	Time of last record modification	timestamp	
name*	Name of the distributor	string	50
phone*	The distributor's phone number	string	255
postcode	Postcode of the distributor's address	string	255
resEmail	Reservations email address	string	255
state	State the distributor is located in	string	255
tnc	Terms and conditions	tnc	
tradingName	Name the business is trading under	string	255
website	The distributor's website URL	string	255

Enumerated string type: gender	
Value	**Description**
F	Female
M	Male

Complex type: idCidTuple		
Field Name	**Description**	**Data Type**
cid*	Central CID of the operator or product	long
id*	Unique, distributor specific ID of the operator or product	long

Complex type: language			
Field Name	**Description**	**Data Type**	**Char. Limit**
code*	2 letter ISO 639-1 language code	string	2
names*	Map of 2 letter ISO 639-1 language codes to their respective language display name	sequence of key-value entry elements	

Complex type: levy			
Field Name	**Description**	**Data Type**	**Char. Limit**
amount*	The levy amount	decimal	
description*	A description of this levy	string	∞[1]
1: While we are trying to limit descriptions to a length of 200 characters, technically the field needs to remain unrestricted to allow for unforeseen “spikes” in newly imported products.			

Extended string type: localdate
A date instance without time and timezone information using yyyy-MM-dd ISO notation. E.g.: 2014-12-31

Extended string type: localtime
A time instance with millisecond precision and without timezone information using ISO notation. E.g.: 19:30:00.000

Complex type: loginCredentials		
Field Name	**Description**	**Data Type**
loginName*	Your API user's login name	string
password*	Your password	string

Complex type: loginToken		
Field Name	**Description**	**Data Type**
token*	The JSON Web Token string for use with HTTP authentication in request header: `Authorization: Bearer <token>`	string
expiration*	Expiration timestamp after which the token will no longer be accepted	timestamp

Complex type: logo		
Field Name	**Description**	**Data Type**
fileSize*	The SVG file size in bytes	integer
height*	The print height in mm	decimal
url*	The HTTP URL to this image	URI string
urlSecure*	The HTTPS URL to this image	URI string
width*	The print width in mm	decimal

Extended long type: masterProductId
The id of an eligible master product that can be booked in conjunction with an optional add-on product.

Complex type: operator			
Field Name	**Description**	**Data Type**	**Char. Limit**
accountsEmail*F	Accounts and billing email	string	255
address1*F	Main address of the operator	string	255
address2F	Address continued	string	255
brochureF	Operator brochure PDF	operatorBrochure	
businessNumberF	Business number (e.g. ABN)	string	255
childCutoffAgeF	Age at which pax can no longer be considered to be children. E.g. a value of 12 means pax under the age of 12 will be processed as children (infants if under *infantCutoffAge*), above as adults	integer	
cid*	Central CID of the operator	long	
city*F	City the operator is located in	string	255
code*	A short code commonly used to refer to the operator. NOT the primary key identifying the record in the API	string	12
companyName*F	Company name of the operator, can be different from name	string	255
contactNameF	Name of the first line contact person at the operator	string	255
country*F	Country the operator is located in. ISO country code.	string	2
created*	Date and time of record creation	timestamp	
currency*F	Operator's currency	currency	
customerDataFormat*F	Information on what pax details are required by this	customerDataFormat	

	operator for a reservation, differentiating between the first and all other customers in a group		
customStr1[F]	Customisable field that holds any kind of custom information you wish to associate with this operator	string	255
customStr2[F]	See above	string	255
customStr3[F]	See above	string	255
customStr4[F]	See above	string	255
demo[*F]	Identifies purely fictitious operators that only serve for demonstration purposes	boolean	
disabled[*]	Identifies operators that currently cannot be booked	boolean	
email[*F]	The operator's primary email address	string	255
fax[F]	The operator's fax number	string	255
id[*]	Unique ID of the operator	long	
infantCutoffAge[F]	Age at which pax can no longer be considered to be infants. E.g. a value of 2 means pax under the age of 2 will be processed as infants, 2 and above as children or adults depending on *childCutoffAge*	integer	
language[*F]	The operator's primary language of operations. ISO language code	string	2
latitude[F]	The latitude of the operator's address	double	
logo[F]	Operator's company logo	operatorLogo	
longitude[F]	The longitude of the operator's address	double	
minAge[F]	A minimum age applicable for this operator's entire product portfolio	integer	
modified	Date and time of last record modification	timestamp	
name[*]	Name commonly used for the operator	string	50
phone[*F]	The operator's phone number	string	255
postcode[F]	Postcode of the operator	string	255
resContactName[F]	Name of the contact person for reservations	string	255
resEmail[F]	The operator's reservations email address	string	255
resPhone[F]	The operator's reservations phone number	string	255
state[F]	State the operator is located in	string	255
tags[F]	A list of tags to describe and categorise the operator	tag sequence	255 for all tags
tnc	A hyperlink to a related tnc record (if any). Deprecated with the introduction of tncFull	URI string	depends on API base URL, generally under 100
tncFull[F]	The operator's terms and conditions	tnc	
tncId	ID of related terms and conditions (if any). Deprecated with the introduction of tncFull	long	
tncUrl[F]	The operator's public/external terms and conditions website	string	255
tradingName[F]	Name the business is trading under	string	255
website[F]	The operator's website URL	string	255
F: These fields/sequences can be omitted by specifying the request parameter `includeFullDetails=false` where available, in order to reduce load and response time, when data is not actually required.			

Complex type: operatorBrochure

Field Name	**Description**	**Data Type**	**Char. Limit**
fileSize*	The PDF document file size in bytes	integer	
pages*	The number of pages	integer	
url*	The HTTP URL to this brochure	URI string	generally under 255 characters
urlSecure*	The HTTPS URL to this brochure	URI string	generally under 255 characters

Complex type: operatorLogo

Field Name	**Description**	**Data Type**	**Char. Limit**
width*	The image width in pixels	integer	
height*	The image height in pixels	integer	
fileSize*	The image file size in bytes	integer	
format*	The image format (currently always PNG)	string	Currently always 3
url*	The HTTP URL to this image	URI string	generally under 255 characters
urlSecure*	The HTTPS URL to this image	URI string	generally under 255 characters

Complex type: outlet

Field Name	**Description**	**Data Type**	**Char. Limit**
address1*F	Main address of the outlet	string	255
address2F	Address continued	string	255
businessNumberF	Business number (e.g. ABN)	string	255
city*F	City the outlet is located	string	255
retailCommissionPercF	This outlet's default retail commission percentage if it is to override outlet group and distributor wide values. Deprecated: Use user.retailCommissionPerc	decimal	
companyNameF	Company name of the outlet, can be different from name	string	255
contactNameF	Name of the first line contact person at the outlet	string	255
country*F	Country the outlet is located in. 2 letter ISO 3166-1 alpha-2 code.	string	2
created*	Date and time of record creation	timestamp	
customStr1F	Customisable field that holds any kind of custom information you wish to associate with this outlet	string	255
customStr2F	See above	string	255
customStr3F	See above	string	255
customStr4F	See above	string	255
email*F	The outlet's primary email address	string	255
faxF	The outlet's fax number	string	255
id*	Unique ID of the outlet	long	
language*F	The outlet's primary operations language. ISO language code	string	2
latitudeF	The latitude of the outlet's address	double	
longitudeF	The longitude of the outlet's address	double	
modified	Time of last record modification	timestamp	
name*	Name of the outlet	string	50

phone[*F]	The outlet's phone number	string	255
postcode[F]	Postcode of the outlet	string	255
pseudoCode	Travel agent pseudo code used in SABRE, Galileo...	string	255
state[F]	State the outlet is located in	string	255
tradingName[F]	Name the business is trading under	string	255
website[F]	The outlet's website URL	string	255
Please note: Outlets, the equivalent to a (brick and mortar) travel agency/store, are only interesting for mixed GUI and API use.			
F: When accessing outlet records directly, these fields/sequences can be omitted by specifying the request parameter `includeFullDetails=false` where available, in order to reduce load and response time, when data is not actually required.			

Complex type: outletGroup			
Field Name	**Description**	**Data Type**	**Char. Limit**
address1[*]	Main address of the outlet group HQ	string	255
address2	Address continued	string	255
businessNumber	Business number (e.g. ABN)	string	255
canSellOpenDated	Whether or not users under this group can sell open dated vouchers	boolean	
city[*]	City the outlet group HQ is located	string	255
companyName	Company name of the outlet group, can be different from name	string	255
contactName	Name of the first line contact person at the group	string	255
country[*]	Country the group HQ is located in. 2 letter ISO 3166-1 alpha-2 code.	string	2
created[*]	Date and time of record creation	timestamp	
currency	Currency (as ISO code), used as retail currency for all users under the group	currency	
currSurchargePerc	A currency conversion surcharge applied to the operator supplied rate (where the supplier trades in a different currency). Intended to cover currency exchange losses on retailer to wholesale payments. Note: Only included if calling API user / outlet group is part of the distributor/wholesaler company. See: distributorInternal	decimal	
customStr1	Customisable field that holds any kind of custom information you wish to associate with this group	string	255
customStr2	See above	string	255
customStr3	See above	string	255
customStr4	See above	string	255
distributorInternal	Whether or not the outlet group, and with it its users, are part of the same company as the distributor. This is the case in Livn Direct connections, where retailers do not purchase products through a third party wholesaler (such as Livn Holidays), but act as their own distributor and have their own commercial agreements with the operators.	boolean	
email[*]	The group's primary email address	string	255
fax	The group's fax number	string	255
id[*]	Unique ID of the outletGroup	long	

language*	The group's primary operations language. ISO language code	string	2
latitude	The latitude of the group's HQ address	double	
logo	The SVG logo of the outlet group. This is used in ticket PDFs generated by the system, which can be requested through this API.	logo	
longitude	The longitude of the group's HQ address	double	
modified	Time of last record modification	timestamp	
name*	Name of the outlet group	string	50
paymentMode	Setting that specifies if and how the group's API users pay for, or authorise in-arrears billing and payment for bookings.	paymentMode	
phone*	The group's phone number	string	255
postcode	Postcode of the group's HQ	string	255
pseudoCode	Travel agency consortium pseudo code used in SABRE, Galileo...	string	255
state	State the group HQ is located in	string	255
ticketAccentColor	HTML/CSS hex RGB colour code of the primary colour to be used in ticket PDF generation, e.g. #FEC834	string	7
tradingName	Name the business is trading under	string	255
website	The group's website URL	string	255

Complex type: pax

Field Name	**Description**	**Data Type**	**Char. Limit**
address1	Home address of the customer	string	255
address2	Address continued	string	255
age	Customer age in years, if omitted system assumes pax to be adult automatically	integer	
city	Home city	string	255
country	Home country. 2 letter ISO 3166-1 alpha-2 country code.	string	2
customStr1	Customisable field that holds any kind of custom information you wish to associate with this pax	string	255
customStr2	See above	string	255
customStr3	See above	string	255
customStr4	See above	string	255
dob	Customer date of birth, may be required by the operator, see operator's customerDataFormat	localdate	
email	Pax email address	string	255
firstName	First name	string	255
gender	Pax gender	gender	
id*	Unique ID of the pax	long	
language	The customer's preferred language. ISO language code	string	2
lastName	Last name	string	255
mobile	Mobile number	string	255
nationality	Nationality/Citizenship. 2 letter ISO 3166-1	string	2

	alpha-2 country code.		
notes	Pax notes	string	1000
passportExpiry	Passport expiry date	localdate	
passportIssued	Passport issue date	localdate	
passportNumber	Passport number	string	255
phone	Phone number	string	255
postcode	Home address postcode	string	255
problems	Gives a detailed description of any problems regarding the specific pax record, encountered during cart creation, availability/rates check or checkout	problem sequence	
salutation	Customer's preferred salutation	salutation	
state	Home address state	string	255

Complex type: paxDetails		
Field Name	**Description**	**Data Type**
address*	The scope of address details required	paxDetailLevel
country*	Country of residence (as ISO country code)	paxDetailLevel
dob*	Full date of birth is required	paxDetailLevel
email*	Email address is required	paxDetailLevel
language*	Preferred language is required	paxDetailLevel
mobile*	Mobile number is required	paxDetailLevel
name*	Full name (salutation, first name and last name) is required	paxDetailLevel
nationality*	Nationality (as ISO country code) is required	paxDetailLevel
passport*	Specified whether the passportNumber, passportExpiry and passportIssued dates are required	paxDetailLevel
phone*	Specifies whether the phone number of the first pax in a group is required	paxDetailLevel

Enumerated string type: paxDetailLevel	
Value	**Description**
NOT_REQ	Information is not mandatory for booking
ONCE	Information is required once per booking, i.e. from first pax
ALL	Information is required from all pax in the booking

Enumerated string type: paymentMode	
Value	**Description**
CONFIRM_WITHOUT_PAYMENT	After successfully checking out a cart, its status will be PENDING_AUTHORISATION, and while no actual payments are handled by the system, sales need to be authorised using an arbitrary authorisation code, e.g. your POS transaction number, timestamp, consultant employee number etc.
CREDIT_CARD	After successfully checking out a cart, its status will be PENDING_CREDIT_CARD_PAYMENT, and retailing agents have to make a credit card payment over the net retail total amount to their distributor, using the respective API call.

FCTG_PAYMENT_GATE	Only relevant for Flight Centre Travel Group. After successfully checking out a cart, its status will be PENDING_AUTHORISATION, and the API user has to authorise a payment to the distributor, by supplying a PaymentGate payment authorisation number.
LIVN_OUTLET_CODE	After successfully checking out a cart, its status will be PENDING_AUTHORISATION, and retailing agents have to authorise a payment using their unique payment authorisation code (assigned by Livn).
NO_PAYMENT	No payments are handled by the system and we require no authorisation, i.e. confirmed reservations will not intially be pending authorisation and will never time out. This is normally the case where the calling API user's outlet group belongs to the distribtor. See settings.outletGroup.distributorInternal

Complex type: pickup			
Field Name	**Description**	**Data Type**	**Char. Limit**
id*	Unique ID of the pickup (To be used in cartItem)	long	

operatingDays*	Weekdays pickup is operating on.
	Encoded as bit field integer.
	Monday=2^0=1, Tuesday=2^1=2,…
	Sunday=2^6=64
	Daily=1+2+4+8+16+32+64=127
		integer	

operatingDaysStr*	Textual form of operatingDays. Example: *Mon,Wed,Fri*	string	27
pickupPoint*	Pickup/departure location	pickupPoint	
pickupTime*	Local pickup/departure time	localtime	

Complex type: pickupPoint			
Field Name	**Description**	**Data Type**	**Char. Limit**
address1	Address	string	255
address2	Address continued	string	255
cid*	Central CID of the pickupPoint	long	
city	City the pickup point is located in	string	255
code	Any code this pickup point is commonly referred by	string	40
country	Country the pickup point is located in. ISO country code.	string	2
id*	Unique ID of the pickup point	long	
latitude	The latitude of the pickup point	double	
longitude	The longitude of the pickup point	double	
name*	Name of the pickup point	string	
notes	Any special notes concerning the pickup point	string	1000
postcode	Postcode of the pickup point address	string	255
preferred*	Specifies pickup locations preferred by the operator. These should ideally be offered to pax first	boolean	
state	State the pickup point is located in	string	255

Complex type: problem			
Field Name	**Description**	**Data Type**	**Char. Limit**
code[*]	A standardised problem code	problemCode	
details	A human readable description of the problem	string	∞[1]
1: We are aiming to limit description length to a necessary minimum to describe the underlying issue.			

Enumerated string type: problemCode	
Value	**Description**
UNSPECIFIED	Any problem not elsewhere specified.
CART_CREDIT_CARD_PAYMENT_FAILED	Credit card payment for checked out cart failed.
CART_ITEM_AVAILABILITY_CHECK_FAILED	Could not check the cart item's availability.
CART_ITEM_INSUFFICIENT_AVAILABILITY	There is not enough availability for the selected cart item.
CART_ITEM_PRICE_CHECK_FAILED	Could not retrieve the cart item's up-to-date pricing information.
CART_ITEM_PRICE_CHECK_NO_VALID_RATES	Could not retrieve valid rates for this cart item.
CART_ITEM_PRICE_CHECK_CALCULATING_WHOLESALE_COMMISSION_FAILED	Could not calculate the wholesale commission for this cart item.
CART_ITEM	A problem with at least one of the cart's items. See individual cart items' problems section for full details.
VALIDATION	Cart items could not be grouped into upstream booking requests and validated against the respective operator's requirements.

Complex type: product			
Field Name	**Description**	**Data Type**	**Char. Limit**
bookable[*]	Specifies whether or not this is a bookable product, or merely a parent tour/activity records, used to group its bookable variants (see flavours)	boolean	
cancellationPolicy[F]	Ordered list of conditions, each specifying the cancellation fee percentage, that applies for cancellations made within a certain range of hours till departure. (The list is sorted from shortest to longest time to departure, so the first matching condition would apply at the time of cancellation).	cancellationPolicyCondition sequence	
cid[*]	Central CID of the product	long	
code[*]	A code commonly used to refer to the product. NOT the primary key identifying the record in the API (see *id*)	string	40
created[*]	Date and time of record creation	timestamp	
paxDetails[F]	Passenger information required when booking this product, each field differentiating between not required, required once per booking and required from all pax	paxDetails	
customStr1[F]	Customisable field that can hold any kind of information you wish to associate with this product	string	255
customStr2[F]	See above	string	255
customStr3[F]	See above	string	255

customStr4^{F}	See above	string	255
description^{*F}	A product description	string	∞[1]
disabled*	Identifies products that currently cannot be booked	boolean	
distanceStartToRef^{LF}	Distance of product start location in kilometres from the specified reference point (i.e. *location* search parameter)	double	
dropoffNotes^{F}	Description of any drop off policy. Example: "Tour ends where it starts" or "Drop off at any of the pickup points"	string	1000
duration	Duration of the product activities in milliseconds	long	
fixedWholesaleRateAdult^{WF}	A fixed adult rate agreed between operator and wholesaler	decimal	
fixedWholesaleRateChild^{WF}	A fixed children's rate agreed between operator and wholesaler	decimal	
fixedWholesaleRateInfant^{WF}	A fixed infant's rate agreed between operator and wholesaler	decimal	
flavours*	A list of variations of this product, which have to be specified in the booking process. Only occurs in parent product nodes, where bookable=false.	product sequence	
highlightsStr^{F}	The highlights of this product	string	
id*	Unique ID of the product	long	
images^{F}	A list of product images	productImage sequence	
includes^{F}	What's included with this product	string	
itineraryStr^{F}	The product's itinerary	string	
latitudeStart^{F}	The product's start location latitude. Please note: While we try to pin-point the exact start location of the product as closely as possible, this field may only be an approximation	double	
latitudeEnd^{F}	Analogue to the above for the product's end location	double	
locationStart^{F}	The product's start location as text	string	255
locationEnd^{F}	The product's end location as text	string	255
longitudeStart^{F}	The product's start location longitude. Please note: While we try to pin-point the exact start location of the product as closely as possible, this field may only be an approximation	double	
longitudeEnd^{F}	Analogue to the above for the product's end location	double	
levies^{F}	A list of levies incurred on top of the retail rate. These are paid directly to the operators or their subcontractors	levy sequence	
manualBooking	Products that cannot be booked live via the API, but require the wholesaler to make a reservation with the supplier over the phone. These products are only included in product listings and search results if explicitly requested, by specifying the request parameter *includeManualBooking=true* (default: *false*). At this stage *manualBooking* products only exist for the purpose of displaying such content in Livn's internal wholesale applications.	boolean	

masterProductIds	Ids of suitable master products, one of which is required to be booked along with this product. Example: optional add-on dive, that can only be booked on top of a cruise. Note: This used to be a simple long value *requiredProductId*, but as a requirement of a newly implemented supplier reservation system, as well as to prepare for possible future supply channels, a more flexible one-to-many relationship was devised.	masterProductId sequence	
maxAge*	A maximum pax age for this product	integer	
maxPaxCount*	The maximum number of pax per booked unit. E.g. 2 for a tour with double bed accommodation, that is priced and sold "by the room". Always 1 if product is priced/sold individually per pax.	integer	
maxRateAdult^{*F}	The highest known/cached adult rate	decimal	
maxRateChild^{F}	The highest known/cached child rate	decimal	
maxRateInfant^{F}	The highest known/cached infant rate	decimal	
maxUnits*	The maximum number of units that can be booked. If not specified, our maximum number of pax (not necessarily same as units!) still applies.	integer	
minAge*	A minimum pax age for this product	integer	
minPaxCount*	The minimum number of pax per booked unit. E.g. 1 or 2 for a tour with double bed accommodation, that is priced and sold "by the room", depending on whether it permits single occupancy. Always 1 if products is priced/sold individually per pax.	integer	
minRateAdult^{*F}	The lowest known/cached adult's rate	decimal	
minRateChild^{F}	The lowest known/cached children's rate	decimal	
minRateInfant^{F}	The lowest known/cached infant's rate	decimal	
minUnits*	The minimum number of units that needs to be booked together. Generally only used where the product is priced and sold per pax (i.e. 1 unit/pax) and each pax has to select the product individually.	integer	
modified	Date and time of last record modification	timestamp	
name*	Name commonly used for the product	string	255
onlyVouchers^{*F}	Signalises that the product is (currently) exclusively available for open dated sales	boolean	
operatingDays^{*F}	Weekdays product is operating on. Encoded as bit field integer. Monday=2^0=1, Tuesday=2^1=2, …Sunday=2^6=64 Daily=1+2+4+8+16+32+64=127	integer	
operatingDaysStr^{*F}	Textual form of *operatingDays*. Example: Mon,Wed,Fri	string	27
operatorCid*	Central CID of the product's operator	long	
operatorCurrency*	Currency of the product's operator	currency	
operatorId*	Unique ID of the product's operator	long	
operatorName*	Name of the product's operator	string	50
options*	A list of optional add-ons, that can be purchased in conjunction with this product, or	product sequence	

	one of its flavours. Only occurs in parent product nodes.		
parentCid*	Central CID of the parent product (see below)	long	
parentId*	ID of a parent product if there is one. Note: Multiple products bookable through the Livn API may share a common parent, which itself cannot be purchased. This ID merely serves to help identify and group such related “tour flavours”	long	
parentName*	Name of parent product (if there is one)	string	255
parentCode*	Code of parent product (if there is one)	string	40
parentDescription*	Description of parent (if there is one)	string	∞[1]
pickupNotes^{F}	Any special notes on pickup, departure etc. E.g. "Please be ready to leave at least 15m prior to departure."	string	1000
pickupRequired^{*P}	Indicated whether or not a pickup selection is mandatory for this product. Null indicates a non-pickup-enabled product.	boolean	
pickups^{*PF}	Offered pickups	pickup sequence	
pickupsChanged^{*PF}	Date and time instant when last change to pickups occurred	timestamp	
ratesCached^{*F}	Datetime instant when the min/max display rates were last updated	timestamp	
requiredMultiple*	If this is set and >1, the number of units booked on this particular product must be a multiple of this value. Generally only used where the product is priced and sold per pax (i.e. 1 unit/pax) and each pax has to select the product individually.	integer	
retailCommissionPerc^{*C}	The retail commission percentage, specific to the product and API user (% of retail gross rate, paid as commission to the retailing agent)	decimal	
sellVouchers^{*F}	Specifies whether the product can be purchased as voucher (=open dated ticket)	boolean	
specialNotes^{F}	Any special notes or instructions to the pax	string	∞
startTime^{F}	Local start/departure time	localtime	
tags^{F}	A list of tags to describe and categorise the product	tag sequence	
tnc	A hyperlink to a related tnc record (if any). Deprecated with the introduction of tncFull	URI string	depends on API base URL, generally under 100
tncFull^{F}	Product specific terms and conditions. If defined, these terms apply on top of those defined at the operator level.	tnc	
tncId	ID of related terms and conditions (if any). Deprecated with the introduction of tncFull	long	
type*	The enumerated product type TOUR, FLAVOUR or OPTION	productType	
voucherInstructions^{F}	Special instructions regarding open dated vouchers	string	255
voucherValidity^{*F}	Open dated voucher validity in months	integer	
wholesaleCommissionPerc^{*W}	The applicable wholesale commission percentage. Any fixed wholesale rate	decimal	

	agreements override this value		

F: These fields/sequences can be omitted by specifying the request parameter *includeFullDetails=false* where available, in order to reduce load and response time, when data is not actually required.
P: Only included with pickup enabled products, and if applicable when explicitly included by a resource URL parameter.
L: Only included in search results of *searchProducts* method and if the *location* parameter was specified
C: Only included in single product requests and if explicitly requested by specifying the request parameter *includeRetailCommission=true*.

1: While we are obviously aiming to limit product descriptions to a reasonable length, technically the field needs to remain unrestricted to allow for longer operator supplied blurbs in newly imported products.

Complex type: productCity

Field Name	**Description**	**Data Type**
name*	City name	string
state	State/county the city is located in	string
country*	ISO country code.	string
latitude*	The latitude of the city centre	double
longitude*	The longitude of the city centre	double

Complex type: productImage

Field Name	**Description**	**Data Type**	**Char. Limit**
width*	The image width in pixels	integer	
height*	The image height in pixels	integer	
fileSize*	The image file size in bytes	integer	
format*	The image format (currently PNG or JPEG)	string	currently 4
url*	The HTTP URL to this image	URI string	generally under 255 characters
urlSecure*	The HTTPS URL to this image	URI string	generally under 255 characters
urlPreview*	The HTTP URL to a downscaled preview version. Currently 800x600 or 600x800 px	URI string	generally under 255 characters
urlPreviewSecure*	The HTTPS URL to a downscaled preview	URI string	generally under 255 characters
urlThumbnail*	The HTTP URL to a downscaled thumbnail. Currently 160x120 or 120x160 px	URI string	generally under 255 characters
urlThumbnailSecure*	The HTTPS URL to a downscaled thumbnail	URI string	generally under 255 characters

Enumerated string type: productType

Value	**Description**
FLAVOUR	Flavours are distinct, bookable variants of a TOUR, in the sense that each such variant is subject to its own availability and rates, and will vary in some aspects from its siblings, that are all grouped under a common parent, top-level TOUR. This can include different start times, durations, room configuration of an included accommodation component (Tour X with Single Cabin, Tour X with Double Cabin), included features, special conditions like club memberships (YHA member, ISIC Cardholder), or some combination of these and other differentiating factors. We take care to reflect the differences between various flavours of a tour in the product names alone, and of course in the numerous specific product properties (duration, startTime, includes, ...)
OPTION	An optional add-on product, that can only be booked in conjunction with another bookable product (which can be a FLAVOUR or a TOUR), as specified by the list *product.masterProductIds*.
TOUR	Top level product, which can be directly bookable, where no multiple bookable flavours of the tour exist, or not itself directly bookable, where the operator/supplier reservation system uses flavours, and tours merely serve as grouping, top-level containers.

Complex type: rate			
Field Name	**Description**	**Data Type**	**Char. Limit**
levies	A list of levies incurred on top of the retail rate. These are payable by the pax, directly to the operators or their subcontractors (in local currency)	levy sequence	
leviesTotal	The total of all local levies	decimal	
operatorRate*	The product rate supplied by the operator	decimal	
paxAge*	The requested pax age	integer	
retailCommissionAmount*	The retail commission amount	decimal	
retailNetRate*	Net retail rate (retailRate-retailCommission)	decimal	
retailRate*	The recommended retail price	decimal	
wholesaleCommissionAmount*W	The wholesale commission generated by this product purchase. Only included if API user is part of the wholesaler company.	decimal	
wholesaleNetRate*W	The resulting wholesale net rate (operatorRate-wholesaleCommissionAmount). Only included if API user is part of the wholesaler company.	decimal	

Complex type: ratesDate			
Field Name	**Description**	**Data Type**	**Char. Limit**
checkFailed*	Signalises when the external rates check request failed	boolean	
created*	Time and date the particular rates check completed	timestamp	
date	Specific date the rates apply to, if null rates are for open dated booking	localdate	
hasUsableRates*	Signalises whether usable pricing information was returned by our supplier. If false it is likely the tour/product does not operate on the specified date.	boolean	
rates	List of all usable rates for the specific date, or for open dated reservations	rate sequence	

Complex type: ratesCheck			
Field Name	**Description**	**Data Type**	**Char. Limit**
created*	Time and date the rates check as a whole was initiated	timestamp	
endDate	End date of checked period	localdate	
handlingSurchargePercentage*W	A potential handling surcharge applied to the operator supplied rate. Primarily intended to cover currency exchange losses on retailer to wholesale payments. Only included if API user is part of the wholesaler company	decimal	
operatorCurrency*	The operator's currency	currency	

paxAges*	Array of the required pax ages	integer array	
productId*	ID of the requested product	long	
ratesDates*	List of all ratesDate elements applicable for the specified date range	ratesDate sequence	
retailCommissionPerc*	The retail commission percentage	decimal	
retailCurrency*	The retailer's (i.e. API user) currency	currency	
startDate*	Start date of checked period. If null rates check is for open dated booking	localdate	

Complex type: reservation

Field Name	**Description**	**Data Type**	**Char. Limit**
cancellation^{C}	A hyperlink to a related cancellation (if any)	URI string	depends on API base URL, generally under 255
cancellationId^{C}	ID of related cancellation (if any)	long	
cancellationQuote^{C}	A hyperlink to request a cancellation quote (Only included where possible at the time)	URI string	depends on API base URL, generally under 255
cancellationReason^{C}	A short description of the reason for the cancellation where applicable	string	255
cancelled^{C}	Date and time the reservation was cancelled	timestamp	
cartCustomStr1^{E}	Local copy of the customisable string (*customStr1*) of the cart	string	255
cartCustomStr2^{E}	See above	string	255
cartCustomStr3^{E}	See above	string	255
cartCustomStr4^{E}	See above	string	255
cartId^{*E}	ID of the reservation's original cart	long	
cartRetailRef	Retailer's reference number POSTed with the original cart	string	50
confirmed	Date and time the reservation was confirmed by the operator	timestamp	
created*	Date and time of record creation	timestamp	
expires	Expiry date of open dated vouchers	timestamp	
globalRef*	Globally unique reference number	string	255 (realistically < 20)
grandTotalOperator*	The grand total of the operator supplied product rates	decimal	
grandTotalRetail*	The grand total of the recommended retail prices	decimal	
grandTotalRetailOverride*	The grand total of the retail prices as overridden by the retailer	decimal	
handlingSurchargePercentage^{W}	A potential handling surcharge applied to the operator supplied rate. Primarily intended to cover currency exchange losses on retailer to wholesale payments.	decimal	
id*	ID of this reservation	long	
idExternal	ID of this reservation in the supplier reservation system	string	255
notes	Operator notes	string	1000

numberOfPax*	The number of pax on the reservation	integer	
operatorCurrency*	Operator's currency	currency	
operatorCustomStr1	Copy of the operator's *customStr1* at the time of cart creation	string	255
operatorCustomStr2	See above	string	255
operatorCustomStr3	See above	string	255
operatorCustomStr4	See above	string	255
operatorCid*	Central CID of this reservation's operator	long	
operatorId*	ID of this reservation's operator	long	
productDate	Travel/departure date	localdate	
reservationItems*	The individual reservation items, each representing a product ticket	reservationItem sequence	
retailCommissionTotal*	The sum of retail commission paid on this reservation	decimal	
retailCurrency*	The retailer's (i.e. API user) currency	currency	
status*	The progress/status of this reservation	reservationStatus	
tnc	The operator terms & conditions	string	
type*	The type of this reservation	reservationType	
wholesaleCommissionTotal*[W]	The sum of wholesale commission generated by this reservation	decimal	
E: Omitted as redundant, if reservation element is embedded inside a cart element C: Conditional fields, only included when actually applicable, e.g. the reservation can be or has already been cancelled			

Complex type: reservationItem			
Field Name	**Description**	**Data Type**	**Char. Limit**
cancellationPolicy[F]	Ordered list of conditions, each specifying the cancellation fee percentage, that applies for cancellations made within a certain range of hours till departure. (The list is sorted from shortest to longest time to departure, so the first matching condition would apply at the time of cancellation).	cancellationPolicyCondition sequence	
created*	Date and time of record creation	timestamp	
date	Travel/departure date	localdate	
id*	ID of this reservation item	long	
idExternal	ID of the reservation item in the supplier reservation system	string	255
levies	A list of levies incurred on top of the retail rate. These are payable by the pax, directly to the operators or their subcontractors	levy sequence	
operatorClaimed[W]	Date and time when the ticket sale was claimed by the operator	timestamp	
operatorNote	Operator notes	string	255
operatorRate*	The product rate supplied by the operator	decimal	
paxCustomStr1[E]	Local copy of the customisable string (*customStr1*) of the pax	string	255
paxCustomStr2[E]	See above	string	255
paxCustomStr3[E]	See above	string	255
paxCustomStr4[E]	See above	string	255

paxId*	ID of the pax record	long	
paxIndex*	Index of the pax based on the initial cart, 0 based	integer	
paxName	Full name of the pax	string	255
paxNote	Pax notes	string	255
paxNumber*	Number of the pax as part of the reservation (e.g. pax 2 of 3)	integer	
pickupAddress	Full address of the selected pickup point	string	255
pickupId*P	ID of the selected pickup	long	
pickupLatitudeP	Geographic latitude of the selected pickup point	double	
pickupLongitudeP	Geographic longitude of the selected pickup point	double	
pickupPointCodeP	Code of the selected pickup point	string	255
pickupPointName*P	Name of the selected pickup point	string	255
pickupTime*P	Local time of the selected pickup	localtime	
productCode*	Code of the sold product	string	255
productCustomStr1	Customisable string (*customStr1*) of the product at time cart was created	string	255
productCustomStr2	See above	string	255
productCustomStr3	See above	string	255
productCustomStr4	See above	string	255
productCid*	Central CID of the sold product	long	
productId*	ID of the sold product	long	
productName*	Name of the sold product	string	255
retailCommissionAmount*	The retail commission paid on the reservation item / ticket	decimal	
retailRate*	The recommended retail price of the reservation item / ticket	decimal	
retailRateOverride*	The retail price as overridden by the retailer	decimal	
retailRef	Retailer's reference number POSTed with the original cartItem	string	50
ticket	A hyperlink to the ticket PDF	URI string	depends on API base URL, generally under 255
ticketInfo	Any additional information to be included on the ticket.	string	2000
voucherId*	Ticket/voucher number	string	19
wholesaleCommissionAmount*W	The wholesale commission generated by this reservation item	decimal	
wholesaleNetRate*W	The resulting wholesale net rate = operatorRate-wholesaleCommission	decimal	
E: Omitted as redundant, if reservationItem element is embedded inside a cart element P: Only included if reservation item includes a pickup			

Enumerated string type: reservationStatus	
Value	**Description**
PENDING	Reservation is yet to be confirmed with & by the supplier

CONFIRMED	Successful reservation has been confirmed by the supplier and API user
CANCELLED	Previously confirmed reservation has been cancelled
CANCELLATION_FAILED	Previously confirmed reservation, failed to be cancelled due to supplier limitations (Too close to travel date)
TIMEDOUT	Reservation timed out due to overdue required credit card payment, payment authorisation, or general API user re-confirmation

Enumerated string type: reservationType	
Value	**Description**
FIXED_DATE	A regular date specific confirmed reservation
OPEN_DATED	An open dated ticket sale

Complex type: resFlatView			
Field Name	**Description**	**Data Type**	**Char. Limit**
cartCustomStr1	Local copy of the customisable string (*customStr1*) of the cart	string	255
cartCustomStr2	See above	string	255
cartCustomStr3	See above	string	255
cartCustomStr4	See above	string	255
cartId*	ID of the reservation's original cart	long	
reservationId*	ID of the reservation	long	
reservationItems*	The individual reservation items, each representing a product ticket	resItemFlatView sequence	

Complex type: resItemFlatView			
Field Name	**Description**	**Data Type**	**Char. Limit**
externalResId	ID of the reservation item in the supplier reservation system	string	255
grossRate*	The product rate supplied by the operator	decimal	
id*	ID of this reservation item (i.e. ticket number)	long	
paxCustomStr1	Local copy of the customisable string (*customStr1*) of the pax	string	255
paxCustomStr2	See above	string	255
paxCustomStr3	See above	string	255
paxCustomStr4	See above	string	255
paxId*	ID of the pax record	long	
paxIndex*	Index of the pax based on the initial cart, 0 based	integer	
paxName	Full name of the pax	string	255
productCustomStr1	Customisable string (*customStr1*) of the product at the time cart was created	string	255
productCustomStr2	See above	string	255
productCustomStr3	See above	string	255

productCustomStr4	See above	string	255
productName*	Name of the sold product	string	255
voucherId*	Ticket/voucher number	string	19

Complex type: retailTotals

Field Name	Description	Data Type
commission*	Sum of cart's retail commission to be paid to, or retained by the retailer/API user	decimal
grossAmount*	Sum of cart's retail rates	decimal
netAmount*	Sum of cart's retail net rates, equals grossAmount-commission	decimal

Complex type: sale

Field Name	Description	Data Type	Char. Limit
apiUserId	ID of the API user that made the sale	long	
cancellationId	ID of related cancellation (if any)	long	
cancellationReason	A short description of the reason for the cancellation where applicable	string	255
cancelled	Date and time the sale was cancelled	timestamp	
cartCreated*	Date and time of cart creation	timestamp	
cartCustomStr1	Local copy of the customisable string (*customStr1*) of the cart	string	255
cartCustomStr2	See above	string	255
cartCustomStr3	See above	string	255
cartCustomStr4	See above	string	255
cartId*	ID of the reservation's original cart	long	
cartRetailRef	Retailer's reference number POSTed with the original cart	string	50
confirmed	Date and time the sale was confirmed by the operator	timestamp	
consultantId	ID of the consultant that handled the sale (only interesting for mixed GUI and API use)	long	
created*	Date and time of sale creation	timestamp	
dateCancelled	Date and time the entire cart was cancelled	timestamp	
dateConfirmed	Date and time the entire cart was confirmed	timestamp	
dateSold	Date and time the entire cart was marked as sold	timestamp	
dateTimedOut	Date and time the entire cart timed out	timestamp	
demo*	Sale was made with a demo operator	boolean	
distributorCurrency*	The distributor/wholesales's currency. 3-letter ISO currency code	string	3
distributorId*	ID of the distributor/wholesaler	long	
globalRef*	Globally unique reference number	string	255 (realistically < 20)
operatorClaimed	Date and time the operator has claimed this ticket	timestamp	
operatorCurrency*	The operator's currency. 3-letter ISO code	string	3

operatorName*	Name of the operator	string	255
operatorRate*	The product rate supplied by the operator	decimal	
outletGroupId*	ID of the outlet group that made the sale	long	
outletId	ID of the outlet that made the sale (only interesting for mixed GUI and API use)	long	
paxCustomStr1	Customisable string (*customStr1*) of the pax	string	255
paxCustomStr2	See above	string	255
paxCustomStr3	See above	string	255
paxCustomStr4	See above	string	255
paxEmail	Email address of the pax	string	255
paxFirstName	First name of the pax	string	255
paxId*	ID of the pax record	long	
paxLastName	Last name of the pax	string	255
paxPhone	Phone/mobile number of the pax	string	255
paxSalutation	Preferred salutation of pax	salutation	
paymentAuthorisation	Payment authorisation / confirmation code	string	255
productClass*	Concise description of the product class	string	255
productCode*	Code of the sold product	string	255
productName*	Name of the sold product	string	255
reservationCustomStr1	Copy of the operator's *customStr1* at the time of cart creation	string	255
reservationCustomStr2	*See above*	string	255
reservationCustomStr3	*See above*	string	255
reservationCustomStr4	*See above*	string	255
reservationId*	ID of this sales transaction (*reservationId* or *cancellationId*)	long	
reservationItemCustomStr1	Copy of the product's *customStr1* at the time of cart creation	string	255
reservationItemCustomStr2	See above	string	255
reservationItemCustomStr3	See above	string	255
reservationItemCustomStr4	See above	string	255
reservationItemId*	ID of this sale (*reservationItemId* or *cancellationItemId*)	long	
reservationItemIdExternal	ID of this sale in the supplier’s reservation system	long	
retailCommissionAmount*	The retail commission paid on this sale	decimal	
retailCommissionPercentage*	The retail commission percentage	decimal	
retailCurrency*	Retailer / API user's currency. 3-letter ISO currency code	string	3
retailRate*	The recommended retail price of this sale	decimal	
retailRateNet*	The resulting net retail rate (retailRate-retailCommissionAmount)	decimal	
retailRateOverride*	The retail price as overridden by the retailer	decimal	
retailRef	Retailer's reference number POSTed with the original cartItem	string	50
status*	The progress/status of this sale	string	255
travelDate	Travel/departure date	localdate	
txnType*	Sale transaction type	txnType	

wholesaleCommissionAmount*W	The wholesale commission generated by this sale	decimal	
wholesaleNetRate*W	The resulting wholesale net rate = operatorRate-wholesaleCommission	decimal	

Complex type: salesReport			
Field Name	**Description**	**Data Type**	**Char. Limit**
created*	Time and date the report was generated	timestamp	
startDate*	Start time and date of reporting period	timestamp	
endDate*	End time and date of reporting period	timestamp	
sales	List of all sales completed in the specified time frame (if any)	sale sequence	

Enumerated string type: salutation	
Value	**Description**
MR	
MRS	
MISS	
MS	

Extended string type: tag
An individual tag to describe and categorise an operator or product. The combined length of all tags in an individual operator's or product's sequence never exceeds 255 characters.

Complex type: ticketStyle		
Field Name	**Description**	**Data Type**
accentColor	HTML/CSS hex RGB colour code of the primary colour to be used in ticket PDF generation, e.g. #FEC834	string
logoFileSize	File size in bytes	integer
logoHeight	Print height in millimetres	decimal
logoHeightPx	Screen height in px	integer
logoUrl	HTTPS URL where Livn hosts the SVG logo, or external HTTP/HTTPS URL from where the logo should be sideloaded to our content distribution network (used with updateTicketStyle)	URI-string
logoWidth	Print width in millimetres	decimal
logoWidthPx	Screen width in px	integer
Note: When calling the *PUT* resource *updateTicketStyle*, you only need to include *accentColor*, *logoHeight*, *logoWidth* and *logoUrl* (resend previous *logoUrl* to leave logo unchanged).		

Extended string type: timestamp
A date and time instance with millisecond precision using ISO notation. E.g.: 2012-12-24T17:30:40.043+11:00 Use timezone offset ±hh:mm or Z for UTC/Zulu time, or omit to use server time zone (default: Australia/Sydney)

Complex type: tnc			
Field Name	**Description**	**Data Type**	**Char. Limit**
content*	Terms and conditions body text	timestamp	
created*	Date and time of record creation	timestamp	
id*	ID of this tnc	long	
modified*	Date and time last modified	timestamp	

Enumerated string type: txnType	
Value	**Description**
CANCELLATION	The cancellation of a previous sale
OPEN_DATED	An open dated ticket sale
RESERVATION	A regular date specific confirmed reservation

Complex type: user			
Field Name	**Description**	**Data Type**	**Char. Limit**
address1	Main address of the API user. Not commonly present, but set on the outletGroup	string	255
address2	Address continued	string	255
businessNumber	Business number (e.g. ABN)	string	255
city	City the user is located in	string	255
companyName	Company name specific to this user, can be different from name	string	255
contactName	Name of the first line contact person for this specific user	string	255
country	Country the user is located in. 2 letter ISO 3166-1 alpha-2 code.	string	2
created*	Date and time of record creation	timestamp	
customStr1	Customisable field that holds any kind of custom information you wish to associate with this user	string	255
customStr2	See above	string	255
customStr3	See above	string	255
customStr4	See above	string	255
demoMode*	Dictates whether on not demo operators are included in this user's inventory.	boolean	
distributionChannel	Identifies a special distribution channel used for this user, e.g. AMADEUS	string	255
distributor*	The distributor/wholesaler this user is purchasing/receiving products from.	distributor	
email	The user's primary email address	string	255
fax	The group's fax number	string	255
id*	Unique ID of the outletGroup	long	
language	The user's primary operations language. ISO language code	string	2
latitude	The latitude of the user specific address	double	
longitude	The longitude of the user specific address	double	
modified	Time of last record modification	timestamp	

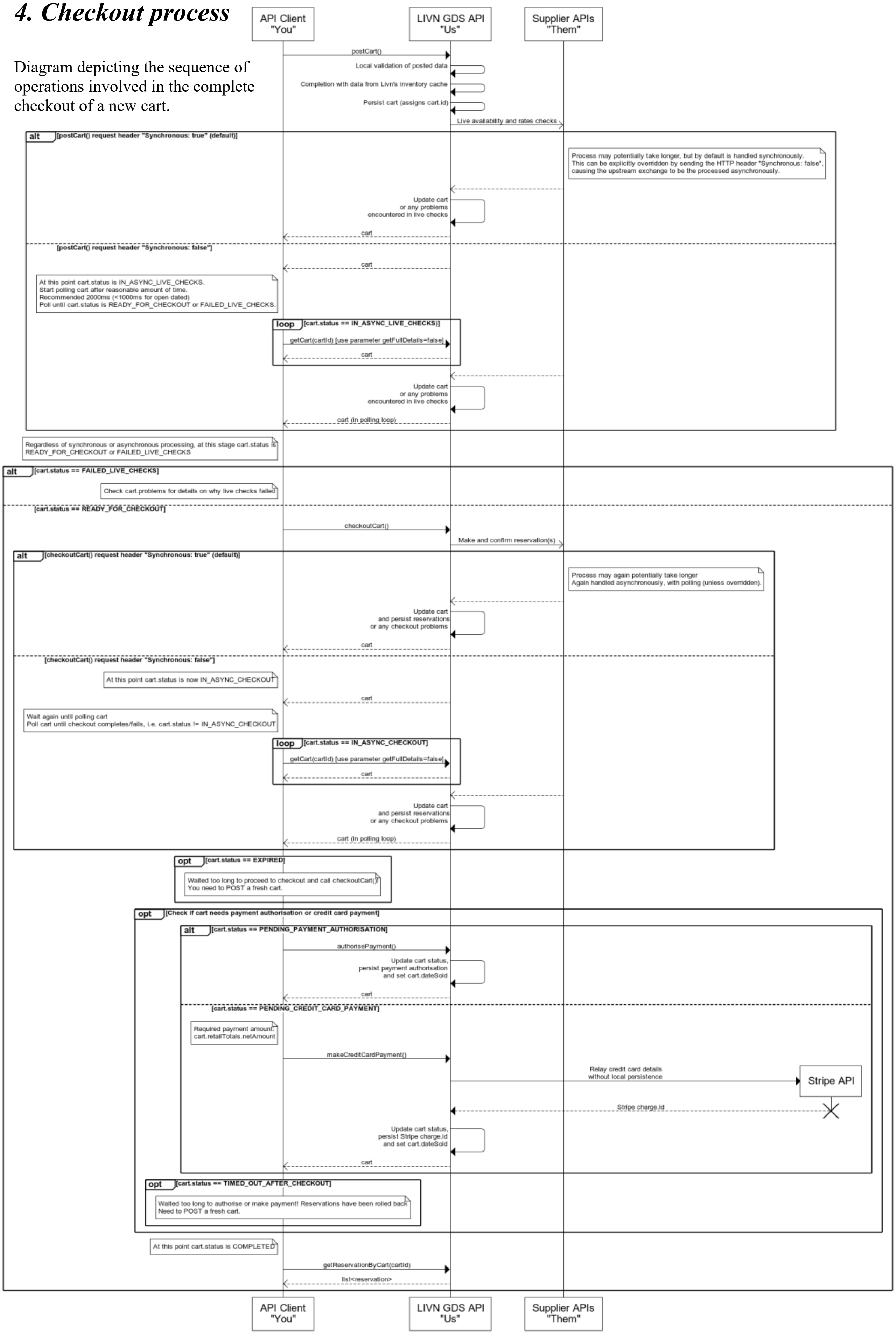
name	Optional display name of the user	string	50
outlet	The (brick and mortar) outlet (i.e. travel agency/shop) this user is associated with. Rarely used.	outlet	
outletGroup*	The agency group or consortium the user is part of.	outletGroup	
phone	The user's phone number	string	255
postcode	Postcode of the user specific address	string	255
pseudoCode	Travel agency pseudo code used in SABRE, Galileo...	string	255
retailCommissionPerc*	The percentage of the retail product value paid to the user as commission	decimal	
state	State the user is located in	string	255
tradingName	Name the user's business is trading under	string	255
website	The user specific website URL	string	255

Complex type: voucherClaim			
Field Name	**Description**	**Data Type**	**Char. Limit**
created*	Time and date this claim was created	timestamp	
currency*	Claim/operator currency. 3-letter ISO currency code	string	3
customStr1	Local copy of the customisable string (*customStr1*) of the cart at the time of claim	string	255
customStr2	*See above*	string	255
customStr3	*See above*	string	255
customStr4	*See above*	string	255
itemCommissionAmount*	Wholesale commission amount	decimal	
itemCommissionPerc*	Wholesale commission percentage	decimal	
itemGross*	Operator supplied product rate	decimal	
itemNet*	The resulting claimable net rate = itemGross-itemCommissionAmount	decimal	
operator*	The claiming operator (limited details only: id, name, email)	operator	
operatorReference*	Reference number supplied by operator when making claim	string	255
pax*	Pax of the claimed sale	pax	
productName*	Name of the claimed product	string	255
travelDate	Travel/departure date	localdate	
txnType*	Type of the claimed transaction	txnType	
voucherId*	Ticket/voucher number	string	19

Complex type: voucherClaimsReport			
Field Name	**Description**	**Data Type**	**Char. Limit**
created*	Time and date the report was generated	timestamp	
invoiceNumber*	Reference/invoice number of this voucher claims report	timestamp	
voucherClaims	List of all vouchers claimed with this report/invoice (if any)	voucherClaim sequence	

4. Checkout process

Diagram depicting the sequence of operations involved in the complete checkout of a new cart.


API Client
"You"
LIVN GDS API
"Us"
Supplier APIs
"Them"
postCart()
Local validation of posted data
Completion with data from Livn's inventory cache
Persist cart (assigns cart.id)
Live availability and rates checks
alt
[postCart() request header "Synchronous: true" (default)]
Process may potentially take longer, but by default is handled synchronously.
This can be explicitly overridden by sending the HTTP header "Synchronous: false",
causing the upstream exchange to be the processed asynchronously.
Update cart
or any problems
encountered in live checks
cart
[postCart() request header "Synchronous: false"]
cart
At this point cart.status is IN_ASYNC_LIVE_CHECKS.
Start polling cart after reasonable amount of time.
Recommended 2000ms (<1000ms for open dated)
Poll until cart.status is READY_FOR_CHECKOUT or FAILED_LIVE_CHECKS.
loop
[cart.status == IN_ASYNC_LIVE_CHECKS)]
getCart(cartId) [use parameter getFullDetails=false]
cart
Update cart
or any problems
encountered in live checks
cart (in polling loop)
Regardless of synchronous or asynchronous processing, at this stage cart.status is
READY_FOR_CHECKOUT or FAILED_LIVE_CHECKS
alt
[cart.status == FAILED_LIVE_CHECKS]
Check cart.problems for details on why live checks failed
[cart.status == READY_FOR_CHECKOUT]
checkoutCart()
Make and confirm reservation(s)
alt
[checkoutCart() request header "Synchronous: true" (default)]
Process may again potentially take longer
Again handled asynchronously, with polling (unless overridden).
Update cart
and persist reservations
or any checkout problems
cart
[checkoutCart() request header "Synchronous: false"]
At this point cart.status is now IN_ASYNC_CHECKOUT
cart
Wait again until polling cart
Poll cart until checkout completes/fails, i.e. cart.status != IN_ASYNC_CHECKOUT
loop
[cart.status == IN_ASYNC_CHECKOUT]
getCart(cartId) [use parameter getFullDetails=false]
cart
Update cart
and persist reservations
or any checkout problems
cart (in polling loop)
opt
[cart.status == EXPIRED]
Waited too long to proceed to checkout and call checkoutCart()!
You need to POST a fresh cart.
opt
[Check if cart needs payment authorisation or credit card payment]
alt
[cart.status == PENDING_PAYMENT_AUTHORISATION]
authorisePayment()
Update cart status,
persist payment authorisation
and set cart.dateSold
cart
[cart.status == PENDING_CREDIT_CARD_PAYMENT]
Required payment amount:
cart.retailTotals.netAmount
makeCreditCardPayment()
Relay credit card details
without local persistence
Stripe API
Stripe charge.id
Update cart status,
persist Stripe charge.id
and set cart.dateSold
cart
opt
[cart.status == TIMED_OUT_AFTER_CHECKOUT]
Waited too long to authorise or make payment! Reservations have been rolled back
Need to POST a fresh cart.
At this point cart.status is COMPLETED
getReservationByCart(cartId)
list<reservation>
API Client
"You"
LIVN GDS API
"Us"
Supplier APIs
"Them"

5. Inventory event notifications using AWS Pub/Sub Messaging

Livn API uses AWS's Simple Notification Service (SNS) to publish or broadcast notifications about changes in its inventory, to any interested party.

Regardless of which of the two basic models of the Livn API you are using, a so called distributor always sits between the operators supplying products and services on the one side, and a hierarchy of retail businesses and individual API users on the other side.
API clients buying product from Livn's wholesale brand Livn Holidays, are set up to sit under this wholesaler on one of Livn Holidays' production endpoints (e.g. gds1). Clients of Livn Direct, utilising the Livn API and curated content to facilitate bookings based on their own, direct relationships and agreements with their suppliers, will generally be configured to act as their own distributor/wholesaler.

Either way, each such distributor or wholesaler uses its own SNS feed, or topic as Amazon calls it, to keep the number of notifications irrelevant to individual API clients limited to cases, where clients only access a subset of their respective distributor's full inventory.

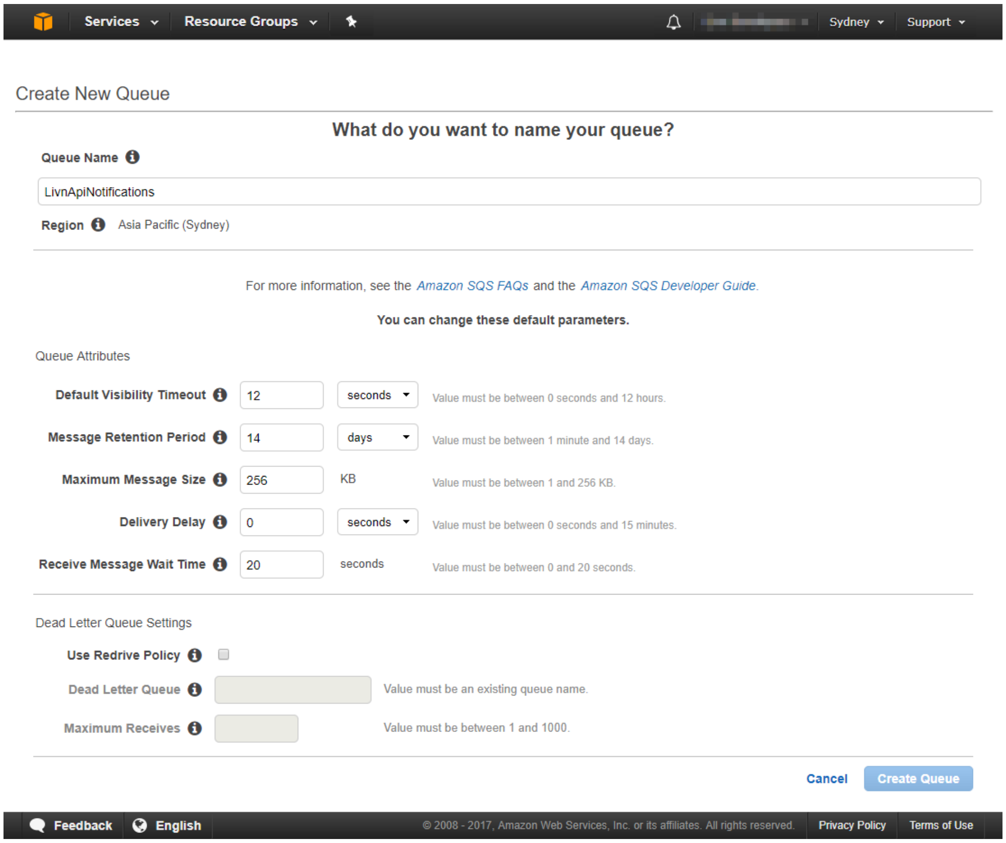
API users can subscribe to this SNS topic with a Simple Queue Service (SQS) queue, that they create within their own AWS account. The purpose of this short chapter is to guide you through all steps required to subscribe such a non-Livn owned and controlled SQS queue, i.e. one not created under Livn's AWS account, to this Livn API inventory notifications SNS topic.

Obviously AWS is a commercial service, so utilising Livn API's live notifications will require signing up for an AWS account, albeit this will not necessarily come at any actual cost. Before you begin, familiarise yourself with AWS offerings in general, their Simple Queue Service in particular, the comprehensive list of SDKs available to connect to AWS in most relevant current programming languages, and not to forget their pricing models (https://aws.amazon.com/sqs/pricing/).

Amazon offer a free tier, which includes up to 1 million requests per month, roughly enough to check the queue for incoming notifications every 3 seconds (hardly practical). Further blocks of 1 million requests are billed monthly at a very reasonable rate, at the time of writing, $0.40 USD (or $0.50 USD if you prefer to use a FIFO queue, details below). The combined volume of data transferred for notifications is negligible for pricing, realistically always falling far short of the included free 1GB per month.

If you decide to proceed, follow these simple steps to receive notifications from your Livn API distributor:

1. Create a new SQS Queue under your AWS account. This can be done via the web based AWS Console, the CLI or any of the AWS SDKs. Standard queues can be in any of AWS numerous regions and it may be most performant to choose the region closest to where you intend to digest messages received by the queue from.



Create New Queue

What do you want to name your queue?

Queue Name: LivnApiNotifications

Region: Asia Pacific (Sydney)

For more information, see the Amazon SQS FAQs and the Amazon SQS Developer Guide.

You can change these default parameters.

Queue Attributes

Default Visibility Timeout: 12 seconds — Value must be between 0 seconds and 12 hours.

Message Retention Period: 14 days — Value must be between 1 minute and 14 days.

Maximum Message Size: 256 KB — Value must be between 1 and 256 KB.

Delivery Delay: 0 seconds — Value must be between 0 seconds and 15 minutes.

Receive Message Wait Time: 20 seconds — Value must be between 0 and 20 seconds.

Dead Letter Queue Settings

Use Redrive Policy: □

Dead Letter Queue: — Value must be an existing queue name.

Maximum Receives: — Value must be between 1 and 1000.

Cancel
Create Queue
Feedback
English
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

In addition to regular Standard Queues, SQS allows the creation of so called FIFO Queues (though only in some AWS regions), that offer guaranteed First-In-First-Out delivery of messages, in strictly the same order messages were published, and further guarantees, that messages will never be delivered more than once, in exchange for a limited throughput of only 300 transactions per second, which is hardly going to affect this application. Since the occasional out of order delivery or duplication of a notification has no significant impact on the functionality of this channel for inventory update notifications, the choice of Standard or FIFO queue, as well as all other settings outlined below are up to the API user's discretion.

We merely recommend setting a long enough message retention period, to cover for the longest realistically expected downtime of your application. AWS allows a value up to 14 days. We further recommend setting the maximum allowed Receive Message Wait Time, meaning AWS will wait 20 seconds for new messages to be received, before returning an empty response, which minimises the amount of wasted queue polling requests.

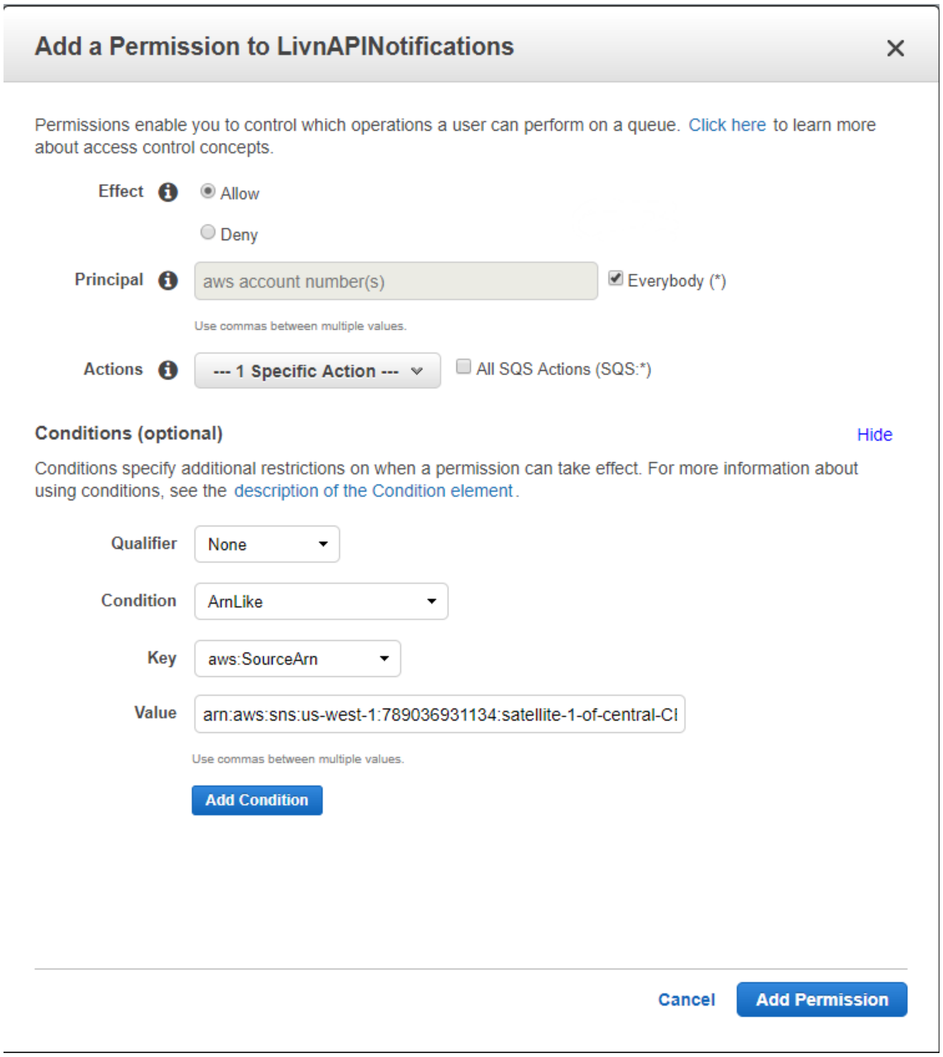
2. Call the API resource *getInventorySNSTopicARN* (GET */aws/inventory*), to retrieve the Amazon Resource Name, short ARN, the URL-like identifier AWS assignes to all of its addressable resources.
   e.g: *arn:aws:sns:us-west-1:789036931134:satellite-1-of-central-CENTRAL-distributor-1-to-apiusers*

3. You, as the SQS queue owner, must give Livn's SNS topic permission to send messages to the queue. This can again be done via the Console, CLI or an SDK. To set the required permission in the Console follow these steps:

   a) Select the SQS queue you wish to connect

   b) Click the *Permissions* tab at the bottom

   c) Choose *Add a Permission*



Add a Permission to LivnAPINotifications

Permissions enable you to control which operations a user can perform on a queue. Click here to learn more about access control concepts.

Effect: ● Allow ○ Deny

Principal: aws account number(s) ☑ Everybody (*)

Use commas between multiple values.

Actions: --- 1 Specific Action --- □ All SQS Actions (SQS:*)

Conditions (optional)

Conditions specify additional restrictions on when a permission can take effect. For more information about using conditions, see the description of the Condition element.

Qualifier: None

Condition: ArnLike

Key: aws:SourceArn

Value: arn:aws:sns:us-west-1:789036931134:satellite-1-of-central-CI

Use commas between multiple values.

   d) Effect: Allow
   Principal: Everybody (i.e. tick the checkbox. no worries, we'll subsequently use a Condition to limit this access to only Livn's SNS Topic)
   Actions: SendMessage

e) Click *Add Conditions*:
Qualifier: None
Condition: ArnEquals or ArnLike
Key: aws:SourceArn
Value: The SNS Topic ARN you retrieved from the Livn API in Step 2

f) Click *Add Condition*, then *Add Permission*

4. Call the API resource *grantPermissionToSubscribeToInventorySNSTopic* (POST */aws/inventory*), posting the ARN of the queue you created above. This sets the correct permissions in Livn's SNS topic, to allow you to subscribe your SQS queue to the topic, without requiring any further confirmation.

5. Subscribe with the SQS queue to the SNS topic. To do this in the online console you must:

   a) Select the correct queue in SQS

   b) Click *Queue Actions* > *Subscribe Queue to SNS Topic*

   c) Paste the SNS topic ARN into the correct field and hit *Subscribe*

At this point the connection is established, and the SQS queue will immediately start receiving all notifications Livn is publishing on your API user's distributor/wholesaler specific SNS topic.

You should next use any of the AWS SKDs to implement a mechanism in whatever application you wish to keep up to date with inventory changes in Livn API, which frequently polls the SQS queue for messages. SQS messages use JSON notation and contain a unique message ID, timestamp when the message was sent, a HMAC-SHA signature and URL to the certificate used to sign the message (as these notifications are not sensitive, there is virtually no sense in validating the signature), a URL to unsubscribe from all further messages from the topic, and of course the actual message body itself.

```
{
   "Type": "Notification",
   "MessageId": "123e4567-e89b-12d3-a456-426655440000",
   "TopicArn": "arn:aws:sns:us-west-1:***",
   "Message": "{\"type\":\"PRODUCT_DELETE\",\"operatorId\":28,\"operatorCid\":1,
              \"productId\":12284,\"productCid\":13,\"parentId\":12282,\"parentCid\":11}",
   "Timestamp": "2017-08-22T01:23:45.678Z",
   "SignatureVersion": "1",
   "Signature": "**************************************************************************************==",
   "SigningCertURL": "https://sns.us-west-1.amazonaws.com/SimpleNotificationService-***.pem",
   "UnsubscribeURL": "https://sns.us-west-1.amazonaws.com/?Action=Unsubscribe&***"
}
```

The payload of all messages broadcast by Livn API, is itself a JSON object (hence it will appear escaped inside the outer message JSON document), of the Livn API model type awsNotification, which essentially consists of an enumerated notification type and the IDs and Central CIDs of the affected record, e.g.:

```
{
   "type": "PRODUCT_DELETE",
   "operatorId": 28,
   "operatorCid": 1,
   "productId": 12284,
   "productCid": 13,
   "parentId": 12282,
   "parentCid": 11
}
```

...which is to inform you and all subscribers of this topic, that the product with ID 12284, which was a flavour or option of parent tour 12282 and belonged to operator 28, has been deleted.

You would subsequently check whether the deletion of product 12284 from the inventory of your API user's distributor, affects you, as not all users may be selling their wholesaler/distributor's full inventory.

Besides *OPERATOR_DELETE* and *PRODUCT_DELETE* notifications, there are similar notifications for *UPDATE* and *INSERT* events, i.e. when and operator or product has been modified in a way that affects consumers of the API, or when a new product (tour, tour flavour or optional add-on) has been added to the inventory:

OPERATOR_UPDATE	An operator has been modified in a way affecting API clients selling its products. Users who cache inventory are advised to update their cache of the specified operator, and ideally all of the operator's products they use.
OPERATOR_DELETE	An operator has been deleted. Users are advised to delete the specified operator from their inventory, along with all of its products.

	Note: You will also receive separate *PRODUCT_DELETE* notifications for each of the operator's products.
PRODUCT_UPDATE	A product has been modified in a way affecting API clients selling it. Users are advised to update their cached record of the specified product, and where applicable, any related child products (flavours/options) they use.
PRODUCT_DELETE	A product has been deleted. Users are advised to delete the specified product, and where applicable, any related child products (flavours/options) they use. Note: You will also receive separate *PRODUCT_DELETE* notifications for any such related child products.
PRODUCT_INSERT_TOUR	A new tour has been added to the distributor's inventory under the specified operator. Users who cache Livn API's inventory are invited to import the new tour into their inventory/cache.
PRODUCT_INSERT_FLAVOUR	A new flavour (=variant) of the specified parent tour has been added to the distributor's inventory. Users who cache inventory are invited to import the new tour flavour into their records.
PRODUCT_INSERT_OPTION	A new option (=add-on service or product) has been added to the specified parent tour. Users who cache Livn API's inventory are invited to import the new optional add-on into their records.

To digest the **_DELETE* and *PRODUCT_INSERT_** type notifications, API clients would subsequently use the regular REST API, to import or update the affected operator and/or product records.

Finally, after you have processed a notification, or determined that the event or entity does not affect your local cache, you must delete the notification from your SQS queue.

6. Change Log

Date	**Change**
2014-10-09	Added field *sale.operatorClaimed* (timestamp)
2014-12-06	Added information on field scope, i.e. mandatory/optional/conditional fields across all data types Added public resource *getTags* Limited access to voucher claims reports resources to API users, internal to the distributor/wholesaler Added resources allowing direct access to products, without going through the operator layer: Added *searchProducts* allowing product search by tags and/or name Added *checkAvailability* call to perform real-time availability checks Added *checkRates* call to perform real-time pricing information checks Added complex data types *ratesCheck*, *ratesDate*, *rate*, *availabilityCheck*, *availability* Added extended string types *currency* and *localdate* Added field *product.operatorId*
2015-04-08	Added field *reservation.tnc* containing the tour operator's, and any product specific terms & conditions (where defined)
2015-05-14	Added field *product.itinerary* Added Tnc to *product* and *operator*
2015-06-01	Added *product.images* and the *productImage* type
2015-06-18	Added *operator.logo* and the *operatorLogo* type Deprecated *operator.logoUrl*
2015-07-08	Added *reservationItem.voucherId*, *resItemFlatView.voucherId* and *cancellationItem.voucherId*. These are to be used as ticket/voucher number instead of *reservationItem.id*, as server may be using a transformed ticket number format.
2015-09-23	Added *product.parentId* and *product.parentName*
2015-10-27	Added *product.parentCode*
2016-02-19	Added *product.includes*, a list of services and activities included in the product. Deprecated sequence/list of extended string types *product.highlights* and *product.itinerary* in favour of plain string *product.hightlightsStr* and *product.itineraryStr*, for reasons of compatibility with our supplier APIs.
2016-03-07	Removed sale.productCodeExternal
2016-04-07	Added field *cid* to numerous data types. Unlike *id* values, which are unique to the specific instance (i.e. server) of the Livn API you are using, these central ID values (short CID), are the same across all servers, and represent the corresponding record in Livn's new central inventory data warehouse. The introduction of a *cid* does not change the existing implementation, please make sure to continue using the regular *id* values specific to your server. But the *cid* can for instance be used to match equal operators and products across multiple API endpoints (e.g. development and production).
2016-06-06	Added service endpoint */public/api-docs-pdf*, which returns the up-to-date version of this PDF document. Added *operator.disabled* for operators that currently cannot be booked.
2016-06-08	Added service endpoint */products/{productId}/departures* to retrieve cached availability and rates. Added model types *departure*, *departureCheck*, *departureRates* and *departureStatus*. Imposing request rate limits on proxied services *checkRates* and *checkAvailability*.
2016-06-14	Added *product.disabled* for products that currently cannot be booked. Added optional parameter *includeDisabled* to include disabled products, and those from disabled operators, in query results of *getProducts* and *searchProducts*.
2016-06-23	Added property *status* to *cart* (of enumerated type *cartStatus*). Added property *retailCurrency* to *cart*, which is to replace the now deprecated *currency*.
2016-06-28	Added service endpoints */public/countries* and */public/languages* to retrieve a list of valid ISO country and language codes, along with display names in several common languages.

2016-07-13	Added *operator.accountsEmail*: Accounts and billing specific email address for operators (where known).
2016-07-14	Added service endpoint */products/categories* and the *category* data type. Added *categories* to *product* type.
2016-07-28	Added a new field *requestOnly* to *availability*. These departures cannot be booked though the API. It may however be possible to make tentative reservations, by direct communication with the operator. Existing implementations require no changes, as *requestOnly* departures will always be marked as not available `if(requestOnly==true) available=false`
2016-07-29	Added new fields *locationStart/End*, *latitudeStart/End* and *longitudeStart/End* to *product*.
2016-08-01	Added new search parameter *location* to *searchProducts* method, which allows limiting search results to products starting within a given distance from a specified reference point. ~~Please note this feature is currently not ready for production, until such time as our full inventory has been geo-coded.~~
2016-08-10	Added new search parameter *airport* to *searchProducts* method, which allows limiting search results to products starting within a given distance from a specified airport (using IATA code). ~~Please note this feature is currently not ready for production, until such time as our full inventory has been geo-coded.~~
2016-08-19	Added fields *nationalityFirst* and *nationalityRest* to model type *customerDataFormat*. Added *nationality*, *passportNumber*, *passportExpiry* and *passportIssued* to *pax*. Fields will only be used by future operators, coming on board in new reservation systems, that are currently being integrated. Operators and product existing at this time will not be affected. Added search parameters *order* and *limit* to product search.
2016-08-24	Added field *passport* to model type *customerDataFormat* to represent the requirement for the recently added *pax* fields *passportNumber*, *passportExpiry* and *passportIssued*.
2016-08-25	Removed deprecated field *operator.logoUrl* as it has been replaced by *operator.logo*. Added *operator.brochure* and the new model type *operatorBrochure* with information on the operator's product brochure PDF document.
2016-08-29	Added resource */products/multi*, which can be used to retrieve multiple records at once by *product.id* or *cid*
2016-08-31	Added fields *maxAge*, *minUnits* and *maxUnits* to *product* type. Added new types *product.paxDetails* and *product.paxDetailLevel* to replace the now deprecated *operator.customerDataFormat*. Added new resource */products/{product.id}/paxDetails* Why this move: The pax information required to book is currently still specified on an operator basis. We intend to move this information to the product level in the near future, as a requirement for new operators and reservation systems coming on board, that have a diverse product portfolio, with varying requirements for different products. Please update your implementation to use *product.paxDetails*, as the operator level information (*operator.customerDataFormat*) is henceforth **deprecated**!
2016-11-03	Added parameter *fts* to *searchProducts* method, which allows full text search in *name*, *tags*, *description* and *highlights*. The existing parameters *name* and *tags* are now **deprecated** and we encourage API users to use *fts* instead! Added fields *operatorName* and *parentDescription* to *product* type, making it easier to display information about bookable products, you might wish to group under their common parent tour.
2016-11-07	Added parameter *treeView* to methods *seachProducts* and *getProducts*. The default API behaviour remains unchanged and requests to these resources will return a flat list, containing only directly bookable products. Specifying *treeView=true* will return the results in an alternative tree structure, with different “flavours” of the same parent tour, nested under a parent node. Analogue optional add-on products, which can only be purchased in conjunction with a full tour or activity, are in this case also listed under their parent tour node. The tree format removes a lot of redundant data, as tour flavours inherit their parent tour's properties, such as descriptions, itineraries, images etc, unless fields are explicitly overridden in the flavour. Also this formatting simplifies the process of creating a listing of main tour/activity products, by moving possible variations of what is essentially the same product, into a lower tier.
2016-11-30	Added new resources: */tickets/{reservationItemId}* and */carts/{cartId}/tickets* to retrieve PDFs for individual reservation items (one ticket per person and product), or a single collated document with all tickets for an entire *cart*.
2017-01-05	Added new type *retailTotals*, field *cart.retailTotals* and resource *cart/{id}/retailTotals*. Type includes the cart's grand total retail commission, net and gross amounts. Added new types/enum values *creditCardPayment*, *CartStatus.PENDING_CREDIT_CARD_PAYMENT* and *ProblemCode.CART_CREDIT_CARD_PAYMENT_FAILED*, as well as the resource *POST /carts/{id}/ccPayment*. Does not affect any of our existing API clients, none of which require credit card payment.

2017-01-24	Added new types *loginCredentials* and *loginToken*, for use as an alternative way of authentication and authorisation with JWT (JSON Web Token). Initial login is done by POST-ing a request body entity type *loginCredentials* to the new resource */public/login*. The returned *loginToken* contains a serialised and encrypted token string, that can subsequently be used instead of basic authentication, with all secured resources.
2017-02-16	Added fields *operator.tncFull* and *product.tncFull* containing the operator's, or product specific terms and conditions (as element of type *tnc*), where full details are requested (or returned by default). For the time being this complements the existing fields *tncId* and *tnc*, which is a link URL to a separate API resource */operators/{id}/tnc* or */products/{id}/tnc*. Please be aware that these old fields are now **deprecated** and may eventually be removed. We kindly advise clients currently using these stand-alone *getTnc* methods, to digest the *tncFull* fields instead, to eliminate unnecessary API requests.
2017-02-28	In the resource *getDepartures* (*/products/{id}/departures*): The parameter *startDate* is now optional. If omitted, the system returns the first, as in soonest, open departure with any availability (*units*>=1). Use the new parameters *minUnits* and *limit* (see below) to further influence this behaviour, e.g. to return the first 3 departures with 2 or more available units use *limit=3* and *minUnits=2*. Added parameter *minUnits*, to exclude departures that don't have the specified minimum availability. Added parameter *limit*, to limit response to the first N departures
2017-03-08	Added new resource: *searchReservationsByPaxNameOrEmail /reservations/search*, to retrieve all reservations belonging to the API user's outlet group, by *pax.name/email*, ignoring case and including partial matches.
2017-03-10	Added new resource: *getAllTickets /reservations/{reservation.id}/tickets*, to retrieve a collated PDF with all pax's tickets for the specified *reservation*. This closes the gap between the resources */tickets/{reservationItemId}* and */carts/{cartId}/tickets* introduced a few months ago.
2017-03-22	Added new resource *getProductAirports /products/airports*, to retrieve a list of all airports, in whose vicinity (radius 100km) there are bookable products. Added new resource *getProductCities /products/cities*, to retrieves a list of the nearest cities (population > 50,000) for all bookable products.
2017-04-04	Added optional parameter *includePickups* to resource *getProduct /products/{productId}*. Value defaults to *true* (preserving existing behaviour), if set to *false* the returned result will not include *pickups*, potentially significantly reducing response time, where pickup information isn't required.
2017-06-05	The previously undocumented fields *sale.pnr*, *reservation.pnr* and *cancellation.pnr* have been renamed to **.globalRef*, and are now officially released into production (on the back of some more exciting changes). The value is simply a globally unique reference number, assigned to every transaction made with a supplier reservation system API, in the sense that it is unique across all instances of the application. Bookings originating from Livn Holidays' wholesale server (gds1.livngds.com) will never have the same reference number as those made on a Livn Direct server operated by one of our clients, through the use of an instance specific prefix. Added new search parameters *startDate*, *endDate* and *requiredUnits* to the *searchProducts* method, which allow limiting results to products that have open, bookable departures in the specified date range. The parameter *endDate* is optional and defaults to the same value as is specified for *startDate*. Parameter *requiredUnits* is also optional and defaults to 1. It allows further restricting results to departures, that meet a minimum number of required bookable units. Please note that this availability is based on the same cache as the number of available units returned by the *getDepartures* method, and may not always be 100% accurate, unlike a live availability check using the *checkAvailability* method (*/product/{productId}/checkAv*). Added *product.startTime* and *departureCheck.productStartTime* containing the local departure time of the product. This is the default start time for products that do not offer courtesy pickups, or for customers that prefer to make their own way to the common meeting and departure point.
2017-06-14	Added the optional custom HTTP request header *Synchronous* to resources *postCart*, *checkoutCart* and *postCancellation*. Requests that include this header, with a value of *true*, *on*, *yes*, *t* or *y* (case insensitive), are processed synchronously, i.e. you will not receive a response until the entire process has been completed, including any forwarded communication with our supplier APIs. This essentially eliminates the need to poll your cart or cancellation, as with the default, asynchronous behaviour.
2017-06-28	Added optional parameter timestamp *modifiedSince* to resource *searchProducts*. Include this to limit results to only those products, that have been added or modified since a specific time instant, including where any change was made on the parent tour or operator.
2017-08-01	Added optional parameter timestamp *modifiedSince* to resource *getProducts /operators/{operatorId}/products*. Similar to its use with the resource *searchProducts*, including this parameter will limit results to only those products, that have been added or modified since a specific time instant, including where any change was made on the parent tour. Note: Unlike with *searchProducts*, this filter does not take into account changes made to the operator itself (as this would obviously affect all products

	in this resource). Added optional parameter timestamp *modifiedSince* to resource *getOperators*. Added resources *getOperatorIdCidTuples /operators/idCidTuples* and *getProductIdCidTuples /products/idCidTuples*. The methods, which were requested by one of our API clients, simply return a list of all operators' or products' *id* and *cid* value pairs or tuples.
2017-08-10	Added retail commission details to resources *checkRates* and *getDepartures* (with parameter *includeRates=true*). New fields: *departureCheck.retailCommissionPerc, departureRates.retailCommissionAmountAdult/-Child/-Infant, departureRates.retailNetRateAdult/-Child/-Infant, ratesCheck.retailCommissionPerc, rate.retailCommissionAmount, rate.retailNetRate*.
2017-08-17	Added field *product.type* with enumerated type *productType.TOUR*, *FLAVOUR* and *OPTION*. Added resource getDeparturesMulti /products/{tour.id}/departures-multi which acts similar to getDepartures, returning our cached availability and optionally rates, but for an entire top level tour, i.e. all bookable products listed under the specified tour, so either the tour itself, or all of its flavours, plus, where applicable, all of the tour's options.
2017-08-22	Added resources *getInventorySNSTopicARN* and *grantPermissionToSubscribeToInventorySNSTopic* and the involved model types *arn*, *awsNotification* and *awsNotificationType*. This new endpoint allows API clients to sign up to receive live notifications about various inventory related events, using an Amazon Web Services message queue subscription. Events such as the deletion of an operator, an existing tour being modified in a way relevant to API users, or the addition of a new tour flavour or option (add-on product) under a specific tour, are now broadcast as short JSON messages, to a distributor specific AWS Simple Notification Service (SNS) topic. The first newly added resource, *getInventorySNSTopicARN*, can be used to retrieve the Amazon Resource Name (ARN), which is the resource identifier commonly used across all of AWS' services, of the SNS topic, relevant to the calling API client. The second resource, *grantPermissionToSubscribeToInventorySNSTopic*, can be used to request the necessary permissions to be set from Livn API's end, to allow API clients to subsequently self-subscribe to that topic with their own AWS Simple Queue Service (SQS) queue. See the new chapters *1.6*, *2.11* and *5* for all details and a step-by-step guide on how to implement these new resources and harness live inventory updates.
2017-08-23	Added optional parameter *includeOptions* to */operators/{operatorId}/products*. The parameter defaults to true, preserving previous behaviour, but can be used to explicitly exclude optional add-on products, that can only be booked in conjunction with a master product.
2017-08-29	Added field *cancellationPolicy* to types *product*, *cartItem* and *reservationItem*. The field is an ordered list of conditions, each specifying the cancellation fee percentage, that applies for cancellations made within a certain range of hours till departure. (The list is sorted from shortest to longest time to departure, so the first matching condition would apply at the time of cancellation). When we receive a new cart, we record the cancellation policy of each product, applicable at that time, with every *cartItem*. That way, these are the terms of cancellation the customer agrees to, when proceeding to check out the cart. And these would be the terms forming the basis of a cancellation quote and applied to the cancellation, regardless of any possible later change to the product's cancellations terms.
2017-09-05	Added new model type *distributor* and resource *getDistributorWholesaler* (*/distributor*), which contains information about the wholesaler/distributor, the calling API user is buying from, e.g. Livn Holidays. Also added a separate resource *getDistributorTncHtml* (*/distributor/tnc*), which only returns the distributor's terms and conditions, as complete HTML document. See chapter *2.12* for more details. Added freely typed fields *cart.retailRef* and *cartItem.retailRef*, which allow the retailing API users to include their own reference number or code with every *cart* and individual *cartItem* they POST. Please note that these values are currently not transmitted to the supplying operators' reservation system, but are carried across into the records of bookings generated for the cart and cartItems, i.e. the newly added fields *reservation.cartRetailRef*, *reservationItem.retailRef*, *sale.cartRetailRef* and *sale.retailRef*. Added resource *getSalesReportCsv* on path */reports/csv*, which simply returns the same results as the existing method *getSalesReport* (*/reports*), when accessed specifying CSV as response MIME type (request header *Accept: text/csv*). This was done due to a limitation of the Swagger / Open API specifications, that do not allow multiple resources on the same path, that only differ by their response MIME type. [https://github.com/swagger-api/swagger-core/issues/935] Note: The old CSV method still exists, for reasons of backwards compatibility, and has merely been hidden from Swagger. Added optional string parameters *customStr1*, *customStr2*, *customStr3* and *customStr4* to the resources *getSalesReport* and *getSalesReportCsv*. If specified, the values are applied as filters, returning only such sales, that were created from a *cart*, that was POSTed with the specified *customStrX* value. Note: It is currently not possible to explicitly search for null values, and a default/null value for either of

	these parameters, is interpreted as the filter being omitted altogether.
2017-09-06	Added new model types *user*, *outletGroup*, *logo*, *demoMode* and *paymentMode*, and new resource *getUser* (*/user*), providing detailed information about the calling API *user*, it's parent *outlet* (travel agency/shop), *outletGroup* (agency consortium) and *distributor* (wholesaler). The resource is currently read only, but we might add a PUT call in the future, to give users the ability to update and self manage certain bits of information (e.g. address and contact details) Deprecated outlet.retailCommissionPerc, use the value user.retailCommissionPerc instead, which may override this value.
2017-10-26	Added field *cancellation.errors*, providing details of why a cancellation failed. So if *cancellation.status=FAILED*, check this new field for details.
2017-12-07	The default behaviour of the *postCart*, *checkoutCart* and *postCancellation* calls has been changed from asynchronous to a synchronous processing. Until now these calls, that all involve some potentially longer running exchange with our upstream supplier reservation systems, would respond immediately with a record id, that would subsequently have to be used to poll the *cart* or *cancellation* record, to monitor for status changes. Instead, these requests will now not return a response until that upstream operation has completed, successfully or otherwise. This synchronous approach simplifies the implementation of the overall booking flow, and was first added with an update on 2017-06-14, by means of including the optional *Synchronous* request header on any of these otherwise asynchronous API calls. The only thing that has changed with this release, is this default behaviour, as it has been embraced by our existing user base. It is still possible to use the calls asynchronously, by explicitly including the request header *Synchronous* with a value of *false*. Not technically an API change, we amended this documentation to clarify the following important point: Across the API, for any resource that includes pricing information, in the absence of any concrete child and/or infant pricing, a product's adult rate will be applicable for all participants regardless of age, and essentially represents the single rack rate for the product. Unless explicitly limited by a minimum age rule (product property *minAge*), this single price, as well as any associated commissions, net rates etc, is to be used for all pax.
2017-12-18	New enumerated type value *ProblemCode.VALIDATION*. We now attempt to group cart items POSTed with a cart into the optimal number of upstream booking requests, and to validate the supplied pax details again the respective operator's requirements, immediately after performing the live availability and rates checks, in the *postCart* call. Until now, this was only done once *checkoutCart* was called, leading to errors if required passenger data was missing, whereas this can now be caught and reported back to our API users at the earliest possible stage. Please note that pax detail validation is no trivial matter, at least once group bookings for multiple products are involved, as the product selection for the individual members of a party, combined with the concrete reservation system's architecture, dictates what number of booking requests needs to be made and which pax becomes the lead participant in each of those bookings. Added optional parameter *includePickups* to *searchProducts* call. Please note that including the full list of pickups with each result item can significantly slow down the search, so please only use this option where it is absolutely necessary for your implementation, and where it actually helps reduce the number of requests you make and the resulting traffic.
2017-12-21	Added new boolean field *product.manualBooking* for products that cannot be booked live via the API, but require the wholesaler to make a reservation with the supplier over the phone. These products are only included in product listings and search results if explicitly requested, by specifying the also newly added request parameter *includeManualBooking=true* (default: *false*). At this stage *manualBooking* products only exist for the purpose of displaying such content in Livn's internal wholesale applications. Added field *product.operatorCurrency*, making every product's respective operator's native currency available directly in product listings and search results, which is for instance used for the locally paid *levies* or, where displayed, *fixedWholesaleRateAdult/Child/Infant*.
2018-01-29	Added optional request parameter *includeRetailCommission* to the product specific *getProduct* requests (*/products/{product.id}* and */operators/{operator.id}/products/{product.id}*). Defaulting to false, this parameter allows clients to explicitly request including the payable retail commission percentage for a product in the response. Please note: As this value may be specific to the exact product and calling API user, and requires a separate database lookup, it is not offered on requests returning potentially large numbers of products, such as *searchProducts* or *getProducts* (*/operators/{operator.id}/products/*) Added new model type *ticketStyle* and resources *getTicketStyle* and *updateTicketStyle* (*GET/PUT /tickets/style*), to retrieve or customise the look of the ticket PDFs we generate (your company logo and an accent/highlight colour).
2018-02-05	Added fields *productName* and *productType* to model type *departureCheck*.