# Contents

## 1 Overview

1.1 Getting Started .............................................. 3
  1.1.1 Prerequisites ........................................... 3
  1.1.2 Where to Start ......................................... 3
  1.1.3 Sample Code ........................................... 3
  1.1.4 Finding Help ........................................... 4
  1.1.5 Rate Limits ........................................... 4
  1.1.6 Terms of Service ....................................... 4

1.2 Connecting to the API ....................................... 4
  1.2.1 Setting Up Your Application ......................... 4
  1.2.2 Authenticating with Token Pairs .................... 5
  1.2.3 Authenticating with OAuth2 ......................... 6

1.3 Request Formatting ........................................ 8

1.4 Response Formatting ....................................... 9

## 2 Developer Guide

2.1 The Developer Console .................................... 11

2.2 Setting Up The SDK ....................................... 11
  2.2.1 With Token Authentication ........................... 11
  2.2.2 With OAuth2 Authentication ......................... 12
  2.2.3 Making Calls to the API ............................... 13

2.3 How To ....................................................... 14
  2.3.1 List Items ............................................... 14
  2.3.2 Add Items to a Booking Session ................... 14
  2.3.3 Alter Items in the Session ......................... 14
  2.3.4 Create a New Booking ................................. 15

2.4 Frequently Asked Questions ............................. 15
  2.4.1 Understanding Errors ................................. 15

## 3 Reference

3.1 Webhooks .................................................. 17
  3.1.1 Booking Modified & Booking Status Change ....... 17
  3.1.2 Item Calendar Edit .................................... 22
  3.1.3 Item Event/Discount Edit ............................ 22
  3.1.4 Item Update ........................................... 27
  3.1.5 New Payment/Refund ................................ 32

3.2 API 3.0 Endpoints ......................................... 34
Note: The v3.0 API documentation is a work in progress and subject to change. If you are developing an application and have additional questions, or find any errors, please contact us for assistance.

The Checkfront API allows developers to expand and build on the Checkfront Booking Platform. The API makes it easy to create web, desktop & mobile applications that directly integrate and interact with your Checkfront account.

This API is built around open standards and secure technologies to streamline development and maintain the integrity of your data.

The documentation is divided into the following sections:

- **Overview**
- **Developer Guide**
- **Reference**
CHAPTER 1

Overview

1.1 Getting Started

Note: Use of the API requires an active Checkfront subscription. Sign up for a free 21 day trial here.

1.1.1 Prerequisites

This documentation assumes you have some knowledge of web programming and API implementations – if not, please see our list of existing plug-ins and extensions, or contact us for an introduction to a qualified Checkfront developer in your area.

1.1.2 Where to Start

Depending on your familiarity with API implementations, your technical skill level, and your language/environment of choice, there may be a number of places for you to get started.

In most cases, you will likely want to start by downloading our Sample Code and running the included examples. You could then go through this documentation to and reference the example code to find out exactly how it works.

If you are comfortable working in your own environment or are working in a language not covered by our SDKs, you may want to begin by experimenting using The Developer Console to get a feel for the system, then review Connecting to the API to begin building your application.

1.1.3 Sample Code

Our SDKs (software development kit) handles most of the complex authentication (OAuth2) issues involved with consuming the Checkfront API. They also include sample code and additional documentation and up and running with the API in no time.
All of our SDKs are open source and are available on Github. If you have a module to contribute, please let us know.

### 1.1.4 Finding Help

If you’re looking for help figuring out how to get your application working, or just have something you want to ask about, there are a few places you can get in touch with us:

- Contact us directly through the support link in your Checkfront account.
- Talk with other developers on our forum.
- Fork us (and contribute!) on GitHub.com/Checkfront.

### 1.1.5 Rate Limits

API throttle limit: We reserve the right to tune the limitations, but they are always set high enough to allow a well-behaving interactive program to do its job.

When the rate limit is exceeded Checkfront will send an HTTP 503 status code. The number of seconds until the throttle is lifted is sent via the “Retry-After” HTTP header, as specified in RFC 2616.

### 1.1.6 Terms of Service

Use of this API is strictly bound by the terms as specified in Checkfront API Terms of Service.

Some functionality documented here may not be available to you based on your plan, or access level of your account.

### 1.2 Connecting to the API

The Checkfront API is accessible via a secure authenticated HTTPS connection to our hosted services, and is isolated to your subscription. This requires your application to have the ability to connect to external servers using SSL and one of the identity token services provided to your Checkfront account.

#### 1.2.1 Setting Up Your Application

To start setting up your application, open your Checkfront account page and use the menu to navigate your browser to Manage > Developer. This page will provide access to a listing of your active application clients, as well as containing your webhook notifications configuration under the “Webhooks” tab, and The Developer Console.

Click the “New Application” button in the upper-left corner, and carefully read the terms of service provided.

#### Choosing Your Workflow

When adding your application, you will be given a choice between two “authentication types”. The method you choose for your application is largely determined by the intended use and scope of your application, and in certain cases you may choose to add more than one access method to your account.
Token Authentication

Token authentication makes use of a simple static private key and secret that can be used for server-to-server communication only. For example, this can be used by an application running on your private web server to create a customer-facing booking page. Applications using this method can only be used privately within your organization, and must not be distributed.

OAuth2 Authentication

OAuth2 is a secure open standard providing a simple transparent avenue of authenticating with the API. Using this method of authentication can allow your application to act on behalf of individual members of your staff by allowing them to “log in” and grant permission to the app. This is ideal for any case where you want to allow your staff to (for example) make or change bookings in the system.

Please see our SDKs or support libraries for OAuth2 in your preferred environment. If we don’t have an example for your language, there are many libraries and documentation available for various programming languages that provide all the functionality you’ll need to implement a custom OAuth2 solution.

Obtaining API Credentials

After you choose an application name (for your own reference), an authentication type, and accept the terms of service, you can click “Create” to save and return to your application listing.

Click on your new application to show the credentials used to connect your application to the API.

| Warning: | Your API credentials provide your application with access to private data stored on your account and the ability to act on your behalf. Treat these as you would your password, and be careful not to distribute or send these to untrusted parties. |

If you are using the SDK, see Setting Up The SDK

| Note: | Much of the following documentation is intended for custom integrations and understanding of the protocols involved in authentication. Our SDK resources abstract the complex authentication processes to simplify development of your application. |

1.2.2 Authenticating with Token Pairs

If you are using token authentication, your application will provide you with two keys:

<table>
<thead>
<tr>
<th>API Key</th>
<th>Will be set as your HTTP BASIC Username</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Secret</td>
<td>Will be set as your HTTP BASIC Password</td>
</tr>
</tbody>
</table>

When sent together in a request header using HTTP Basic authorization, these allow direct access to API endpoints without any additional preamble.

See CURLOPT_USERPWD if working with cURL libraries, or the documentation on HTTP Basic authentication relevant to your chosen framework. If building your requests manually, HTTP Basic credentials are base64 encoded in the sequence “username:password” and sent in the request header in the following format:

1.2. Connecting to the API
### 1.2.3 Authenticating with OAuth2

#### Logging in with a Staff Account

1. Redirect your staff user to your **Authorize Token URL** (e.g. `https://your-company.checkfront.com/oauth/`) with the following query string parameters set:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_id</td>
<td>Your application’s “Consumer Key” (see your application setup).</td>
</tr>
<tr>
<td>response_type</td>
<td>MUST be set to <code>code</code></td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The URI to return your user to after authorization.</td>
</tr>
</tbody>
</table>

2. Upon completion of authorization, the staff user will be sent to the `redirect_uri` specified, passing a `code` parameter in the query string that must be exchanged by your application for an access/refresh token pair within 60 seconds.

   To exchange the code for an access token, make a call to your **Access Token URL** (e.g. `https://your-company.checkfront.com/oauth/token/`) with the following parameters set:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_id</td>
<td>Should be sent as HTTP Basic credentials. See Consumer Key / Consumer Secret.</td>
</tr>
<tr>
<td>client_secret</td>
<td>Should be sent as HTTP Basic credentials. See Consumer Key / Consumer Secret.</td>
</tr>
<tr>
<td>grant_type</td>
<td>MUST be set to <code>authorization_code</code></td>
</tr>
<tr>
<td>code</td>
<td>The authorization code as returned in the client’s GET request to your page.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The URI to return your user to after authorization.</td>
</tr>
</tbody>
</table>

3. Store the token returned by the previous call in a *secure* database along with a field containing the timestamp of most recent update to the token. Your tokens should be refreshed on a regular basis as long as the authorization continues to be used.

   Your application should store and make use of the following fields from the response:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>string See Access Token.</td>
</tr>
<tr>
<td>expires_in</td>
<td>integer The time (in seconds) after which the access token will expire.</td>
</tr>
<tr>
<td>refresh_token</td>
<td>string See Refresh Token.</td>
</tr>
</tbody>
</table>

#### Using and Maintaining OAuth2 Tokens

While your first authentication will provide a usable access token for identifying with the server, the access token has a fixed lifetime and must be refreshed in order to maintain access to the API.

See **Refresh Token** and other details below for information on performing a refresh.

#### OAuth2 Reference

---

---
Authorization Endpoints

There are two important endpoints used in authenticating tokens using OAuth2, which are both displayed when viewing the application key setup on your Checkfront developer page.

- **Your Authorize Token URL** is used when redirecting a user to grant permission to use their account. On success, this will return a code for you to pass to the Access Token URL to grant a token you can use to access the API.

  ```
  https://your-company.checkfront.com/oauth/
  ```

- **Your Access Token URL** is used for granting access tokens from code requests, and refreshing existing access/refresh tokens.

  ```
  https://your-company.checkfront.com/oauth/token/
  ```

Consumer Key / Consumer Secret

These are generated when setting up your application and can be found on your Checkfront developer page. Your consumer key and secret allow your application to grant and refresh tokens on behalf of your users.

**Warning:** You consumer key and secret should only be sent together when making calls to your Access Token URL. When your application is making calls to endpoints requiring a valid access token, the key/secret pair should not be sent.

As with token pairs (see above), these can (and should) be sent as HTTP Basic credentials. However, these can only be sent in this manner to the `/oauth/token/` (code/refresh) endpoint. Your request will be rejected if you attempt to send these to an `/api/` endpoint.

See CURLOPT_USERPWD if working with cURL, or the documentation on HTTP Basic authorization relevant to your chosen framework. If building your requests manually, HTTP Basic credentials are base64 encoded in the sequence “username:password” and sent in the request header in the following format:

```
Authorization: Basic ...
```  

Access Token

This is used by the API server to identify you and allow the application to act on your behalf. When using OAuth2 for your application, an access token is required to be sent with all API calls to secure data endpoints.

Access tokens have a lifetime of **14000 seconds** (this will be returned as expires_in when new tokens are granted), after which they must be refreshed to obtain a new token. Your application should keep track of when this token will be expiring and check if it needs refreshing before attempting a request.

When sending your token with an API request, it can be sent in a header in the following format:

```
Authorization: BEARER f58ef579d0bb5fffb3b5bb0985a85e21a
```  

It will also be accepted in the form of the query string parameter `access_token` in the GET request or POST body if necessary for your application, although this is not recommended in a live application.

1.2. Connecting to the API
Refresh Token

After your current access token has expired, this token can be passed to create a new access/refresh token pair, which must completely replace your previously stored token (which will be invalidated).

To exchange the refresh token for a new access/refresh token pair, make a call to your Access Token URL (eg. https://your-company.checkfront.com/oauth/token/) with the following parameters set:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_id</td>
<td>Should be sent as HTTP Basic credentials. See Consumer Key / Consumer Secret.</td>
</tr>
<tr>
<td>client_secret</td>
<td>Should be sent as HTTP Basic credentials. See Consumer Key / Consumer Secret.</td>
</tr>
<tr>
<td>grant_type</td>
<td>MUST be set to refresh_token</td>
</tr>
<tr>
<td>refresh_token</td>
<td>The current (active) refresh token for this user.</td>
</tr>
</tbody>
</table>

Refresh tokens have a lifetime of 14 days from issue, after which (if allowed to expire) you must generate a new access/refresh token pair to regain application authorization.

1.3 Request Formatting

All requests made to the server are sent over an encrypted SSL connection using standard HTTPS methods such as GET and POST.

When submitting input parameters with your request, these are encoded either in the query string for GET requests, or in the body of a POST request using commonplace url encoding of key-value pairs (application/x-www-form-urlencoded).

The headers of your request should also include the IP of the client you’re forwarding for, as well as the credentials used to authenticate with the API. If you have an open session on the server, you can also pass in a session_id cooking containing the ID to have it automatically reopened.

<table>
<thead>
<tr>
<th>X-Forwarded-For</th>
<th>The IP address of the connecting client.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-On-Behalf</td>
<td>The staff account to act on behalf of. Use “off” to act as a customer.</td>
</tr>
<tr>
<td>Authorization</td>
<td>Credentials used to query the API. See Authenticating with Token Pairs or OAuth2 Reference.</td>
</tr>
</tbody>
</table>

For example, to query a list of bookings with a status of PAID starting today, your full raw request might look like the following:

**GET /api/3.0/**

Example request:

```
GET /api/3.0/booking/index?start_date=today&status_id=PAID HTTP/1.1
Host: your-company.checkfront.com
Accept: application/json
X-Forwarded-For: 66.228.55.142
X-On-Behalf: 1
Authorization: Basic M2JlOTg2NDFmMDc0NWl2ZmU3ZGFjYzJkZjk0N2FkYmMx2GE3MzEy2Do0YzRkNTk4YTvkOTQwZja42mRiN0M1Yj5Y5Y5OD
```

Request Headers
• Authorization – Credentials used to query the API. See Authenticating with Token Pairs or OAuth2 Reference.

• X-On-Behalf – The staff account to act on behalf of. Use “off” to act as a customer.

• X-Forwarded-For – The IP address of the connecting client/customer.

1.4 Response Formatting

All responses from the API are formatted as JSON (JavaScript Object Notation) objects containing information related to the request, and any status. Every modern language should have libraries capable of quickly parsing JSON objects. See json.org for more information.

See also:

For JSON decoding libraries in common environments, see:

- PHP: json_decode
- Python: json.JSONDecoder
- Ruby: JSON.parse
- .Net: Json.NET
- JavaScript: JSON.parse

All responses from the API will include some information related to the processing of the request, such as the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>string</td>
<td>Current API version used to process the request.</td>
</tr>
<tr>
<td>account_id</td>
<td>integer</td>
<td>The staff account that the request was made on behalf of.</td>
</tr>
<tr>
<td>host_id</td>
<td>string</td>
<td>Checkfront host account to which the request was made.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Display name of the company the account belongs to.</td>
</tr>
<tr>
<td>locale</td>
<td>array</td>
<td>Contains the following fields:</td>
</tr>
<tr>
<td>locale -&gt; id</td>
<td>string</td>
<td>Selected locale for the account (eg. en_US)</td>
</tr>
<tr>
<td>locale -&gt; lang</td>
<td>string</td>
<td>Selected language for the account (eg. en)</td>
</tr>
<tr>
<td>locale -&gt; currency</td>
<td>string</td>
<td>Selected currency for the account (eg. CAD)</td>
</tr>
<tr>
<td>request</td>
<td>array</td>
<td>Contains the following fields:</td>
</tr>
<tr>
<td>request -&gt; status</td>
<td>string</td>
<td>Status of the request. Will be “OK” on success.</td>
</tr>
<tr>
<td>request -&gt; resource</td>
<td>string</td>
<td>Resource accessed by the request.</td>
</tr>
<tr>
<td>request -&gt; records</td>
<td>integer</td>
<td>Number of records returned by the request.</td>
</tr>
<tr>
<td>request -&gt; limit</td>
<td>integer</td>
<td>The limit on number of records returned (where applicable).</td>
</tr>
<tr>
<td>request -&gt; page</td>
<td>integer</td>
<td>The current page returned for multi-page requests.</td>
</tr>
<tr>
<td>request -&gt; pages</td>
<td>integer</td>
<td>Number of pages of response records available.</td>
</tr>
<tr>
<td>request -&gt; time</td>
<td>float</td>
<td>The time taken (in ms) to generate the response.</td>
</tr>
</tbody>
</table>

If any critical errors (e.g. failed credentials) occur, the API will not return any of these fields, but will instead return an object containing any available error information.
2.1 The Developer Console

The developer console provides easy, direct access to the API to aid in development or debugging your API calls. This allows you to interact with the API without setting up an environment, where you can manually perform any of the actions available to the API.

Note that all parameters for both GET and POST requests should be URL encoded in the query string.

The console can be found via the “Console” link under Manage > Developer.

**Note:** All calls made in the console are live requests on your account, not examples. Any changes you make to bookings through the console will persist outside of the console.

2.2 Setting Up The SDK

Once you have set up credentials for your application (see *Connecting to the API*), you can set up the SDK and its examples to get a head start on working with the API.

**Warning:** Your API credentials provide your application with access to private data stored on your account and the ability to act on your behalf. Treat these as you would your password, and be careful not to distribute or send these to untrusted parties.

2.2.1 With Token Authentication

Using the PHP-SDK with your extended Checkfront class, you would connect to the API by passing your token key and secrets into the settings, similar to the following:
Checkfront API Documentation, Release 3.0

```php
$Checkfront = new CheckfrontAPI(
    array(
        'host' => 'your-company.checkfront.com',
        'account_id' => 'off', // act on behalf of the customer
        'auth_type' => 'token',
        'api_key' => '3be98641f0745b6fe7dacc2df947adbc1da7312d',
        'api_secret' => 'b300e27b07bfc6fed944a0116b595ec2130e9efc641042d1d12c68c7766f2d861',
    )
);```

Note that the api_key and api_secret fields must be set to the values obtained when setting up your application in your Checkfront account.

Once this is done, the application will be ready to run calls against the API.

### 2.2.2 With OAuth2 Authentication

If you are not familiar with how OAuth2 works, you may want to review [OAuth2 Reference](#) and the preceding authentication overview.

When using OAuth2 authentication, your application will require a place to store access/refresh tokens for your users. While the examples do provide a simple implementation of this, we highly recommend writing your own secure storage procedure suited to the application you’re building. These are typically best stored in a database (such as one you may already be using for other requirements of your application), as they are replaced on a regular basis as long as the application continues to be used.

**Setup**

As with token authentication, a few fields are passed into the object on creation to configure your connection to the API:

```php
$Checkfront = new CheckfrontAPI(
    array(
        'host' => 'your-company.checkfront.com',
        'auth_type' => 'oauth2',
        'consumer_key' => '1d2d5cad60174b5972243693d082e4b4e54979bf',
        'consumer_secret' => '8e3751291c3d0fe090a7d5b28d964407baff96c1028e1e1afb1014d1db85b25a',
        'redirect_uri' => 'oob',
    )
);```

Your consumer_key and consumer_secret should match the values provided when setting up your application. The **redirect_uri** should match the “Callback URL” value you define on the same page.

**Staff Logins**

The SDK includes a simple abstraction process to login staff using OAuth2, easing much of the more complex process of handling logins.

To review the process of handling staff credentials, your application would:
Handle previous logins

Load the previous access_token and refresh_token into your CheckfrontAPI object, likely from a database or a cookie.

To do this, you will at the very least want to write a custom CheckfrontAPI->store() function to save & load tokens. When handling multiple users with tokens stored in your database, remember to account for storing tokens for each.

Whenever a token is prepared for storage by the SDK, it will send an array to your CheckfrontAPI->store($data) function, containing the following fields for storage:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>refresh_token</td>
<td>See Refresh Token.</td>
</tr>
<tr>
<td>access_token</td>
<td>See Access Token.</td>
</tr>
<tr>
<td>expire_token</td>
<td>The unix timestamp after which the access token will expire.</td>
</tr>
<tr>
<td>updated</td>
<td>The current time.</td>
</tr>
</tbody>
</table>

When loading tokens on initialization, the SDK will call the same store() function without any parameters. In this case, the function should return the previously stored values as an array.

If an access_token is expired, the SDK will automatically refresh it on startup. However, if a refresh_token expires, you will need to invalidate the stored tokens and begin the login process for the user. Based on store() input above, you can generally assume the expiry is updated + 14 days, but the tokens can be refreshed at any time before that point.

New logins

1. Make a call to CheckfrontAPI->authorize(boolean $redirect) to either return the login URL or set a redirect header to it. From here, the staff member grants permission to your application, and is redirected back to your specified callback URL.

2. On returning to your callback URL, the client will pass an authentication code in the query string (that’s $_GET['code']), which your application then passes into the fetch_token function of the CheckfrontAPI object as follows:

```
$Checkfront->fetch_token($_GET['code']);
```

3. On completion, a new authenticated token will be returned, and as above, the token will be sent to your store() function.

4. You’re ready to make calls to the API on behalf of the staff member.

2.2.3 Making Calls to the API

The SDK has simple abstractions to common HTTP request methods used with the API – in particular, CheckfrontAPI->get($path, $data); and CheckfrontAPI->post($path, $data);

Using these functions, your application would send the base path of the API element to be accessed in $path, and any options to be sent as an associative array in $data.

For example, instead of manually building a query string in a request such as:

```
GET /api/3.0/booking/index?start_date=today&status_id=PAID HTTP/1.1
Host: your-company.checkfront.com
```

You could instead use:
Checkfront API Documentation, Release 3.0

```php
$data = array(
    'start_date' => 'today',
    'status_id' => 'PAID'
);

$Checkfront->get('booking/index', $data);
```

## 2.3 How To...

### 2.3.1 List Items

You can query a list of available items based on search criteria supplied in an API call to `item`. When no dates are passed in the API call, a full list of enabled items in the inventory. When a date is passed, the API will return a “rated” response that includes pricing and availability, as well as a “SLIP” that will be used to add the item to a session.

In order to build bookings, you’ll need to first retrieve “rated” item details (by declaring the booking length and any additional parameters in an `item` call, see above), which will return a “SLIP” – effectively a token encoding the necessary information to book that item in that time period with the options you’ve selected.

Example rated request to a specific `item`:

```plaintext
GET /api/3.0/item/17?start_date=20141224&end_date=20150101&param[guests]=2 HTTP/1.1
Host: your-company.checkfront.com
```

### 2.3.2 Add Items to a Booking Session

Once you have the `SLIP` for your desired item (or items), this can be passed in to create a new `booking/session`. The session can be thought of as your “cart”, and can be modified until you are ready to submit your booking.

If you have collected multiple items to be added, you can also add them in a batch by sending them as an array.

Example of adding a `SLIP` to a new session:

```plaintext
POST /api/3.0/booking/session HTTP/1.1
Host: your-company.checkfront.com

slip[]=17.20141224X8-guests.2&slip[]=18.20141224X8-guests.2
```

**Note**: For parent/child grouped items, the parent item functions as a container for the child items. As such, the request to obtain the SLIP must be made directly to the child item, rather than the parent.

Once the SLIP has been obtained, it can be added to the booking session as outlined above.

### 2.3.3 Alter Items in the Session

To alter regular items in a `booking/session` make a new rated query for the item, using the altered parameter values and the desired opt-in or out settings for package add-ons. This will provide an updated `SLIP` for the item with the desired quantity. You can then replace the old `SLIP` by referring to the line_id you want it to replace.
POST /api/3.0/booking/session HTTP/1.1
Host: your-company.checkfront.com

/booking/session?session_id={session_id}&line_id={line_id}&slip={new_slip}

If you need to alter simple items in a booking/session, such as changing quantities, or you need to remove any item, you can use the ‘alter’ query parameter of a booking/session call to an existing session.

To change the selected quantity of an added session simple items, pass the quantity in to the alter array entry for the selected line (which will be part of each session response).

To remove an item from the session, use the alter array to specify that the line item should be removed from the order.

GET /api/3.0/booking/session HTTP/1.1
Host: your-company.checkfront.com


POST /api/3.0/booking/session HTTP/1.1
Host: your-company.checkfront.com

{
    "session_id": "rtdv4osethqurlmqgi55mcrkm4",
    "alter": {
        "3": "4",
        "2.1": "optin",
        "1": "remove"
    }
}

2.3.4 Create a New Booking

After you’ve added SLIPs to your session, your application should then capture the customer information needed to make a booking. The fields required for checkout on your account can be retrieved with a GET request to booking/form.

To submit a booking to the system, you’ll then pass your session_id along with the required customer information to a booking/create call, which will return data relating to your booking, such as the booking/customer IDs (which could be recorded in your system) and an invoice/payment URL (if applicable).

2.4 Frequently Asked Questions

2.4.1 Understanding Errors

“The access token was not found.”

When using OAuth2, this error will only appear if the server did not receive the Authenticating with Token Pairs from your application.

“The access token provided is invalid.”

The server received an access token from you, but could not authorize it.
“Malformed auth header.”

Your token pair authorization did not contain all of the required token information in the request header.

“You do not have access to this resource.”

The token pair you attempted to use could not be authorized to use the API. Double check that you are using the correct credentials, and that you are using the correct authentication type. See Connecting to the API
3.1 Webhooks

Your Checkfront account is capable of directly POSTing details of new or modified bookings, items, item events, and transactions to your SSL-secured server endpoint.

Notifications are sent over a standard HTTPS POST request as JSON, XML, or x-www-form-urlencoded data.

Your server must respond with an HTTP 200 class status code immediately upon receipt. If your server does not indicate a successful response in a timely manner, after 5 consecutive failed delivery attempts, your webhook will be automatically disabled.

Webhook setup can be found in the Manage > Developer section of your Checkfront account under the “Webhooks” tab.

3.1.1 Booking Modified & Booking Status Change

The Booking Modified webhook event is triggered whenever a booking is created or modified, and the Booking Status Change webhook event is triggered whenever a booking status is altered. These events may occur simultaneously in some circumstances, and both are triggered whenever a booking status is changed. Using both of these events on the same webhook endpoint URL is discouraged.

A JSON, XML, or x-www-form-urlencoded object containing the following export fields can be found in directly in the raw body of the POST request to your server:
**booking**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>string</td>
<td>The current status code of the booking</td>
</tr>
<tr>
<td>code</td>
<td>string</td>
<td>A unique booking code used to refer to the booking</td>
</tr>
<tr>
<td>created_date</td>
<td>timestamp</td>
<td>The date on which the booking was created</td>
</tr>
<tr>
<td>staff_id</td>
<td>integer</td>
<td>Account ID of the staff account used to create the booking</td>
</tr>
<tr>
<td>source_ip</td>
<td>string</td>
<td>The IP address used to create the booking</td>
</tr>
<tr>
<td>start_date</td>
<td>timestamp</td>
<td>The start date/time of the booking, based on order items</td>
</tr>
<tr>
<td>end_date</td>
<td>timestamp</td>
<td>The end date/time of the booking, based on order items</td>
</tr>
<tr>
<td>customer</td>
<td>array</td>
<td>Customer details attached to the booking (See booking.customer below)</td>
</tr>
<tr>
<td>meta</td>
<td>array</td>
<td>A set of fields containing your custom parameters and other info</td>
</tr>
<tr>
<td>order</td>
<td>array</td>
<td>Details on booking items and payment. See booking.order below</td>
</tr>
<tr>
<td>tid</td>
<td>string</td>
<td>Details on the Tracking ID used for referral &amp; campaign tracking</td>
</tr>
</tbody>
</table>

**booking.customer**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>string</td>
<td>The customer’s unique account code</td>
</tr>
<tr>
<td>customer_name</td>
<td>string</td>
<td>The customer’s full name</td>
</tr>
<tr>
<td>customer_email</td>
<td>string</td>
<td>The customer’s email address</td>
</tr>
<tr>
<td>customer_address</td>
<td>string</td>
<td>The customer’s street address</td>
</tr>
<tr>
<td>customer_region</td>
<td>string</td>
<td>The customer’s province or state</td>
</tr>
<tr>
<td>customer_country</td>
<td>string</td>
<td>The customer’s country of residence</td>
</tr>
<tr>
<td>customerPostal_zip</td>
<td>string</td>
<td>The customer’s postal or zip code</td>
</tr>
<tr>
<td>customer_phone</td>
<td>string</td>
<td>The customer’s phone number</td>
</tr>
</tbody>
</table>

**booking.fields**

Contains each of the form fields associated with a booking.

**booking.order**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub_total</td>
<td>float</td>
<td>The sub-total of all charges added to the order</td>
</tr>
<tr>
<td>paid_total</td>
<td>float</td>
<td>The total amount the customer has paid on the order</td>
</tr>
<tr>
<td>total</td>
<td>float</td>
<td>The total of all charges and taxes added to the order</td>
</tr>
<tr>
<td>tax_total</td>
<td>float</td>
<td>The sum of all taxes applied to the order</td>
</tr>
<tr>
<td>taxes</td>
<td>array</td>
<td>Individual taxes that have been applied to the order, their names, and amounts</td>
</tr>
<tr>
<td>discount</td>
<td>float</td>
<td>The amount that has been discounted from the order total (if applicable)</td>
</tr>
<tr>
<td>items</td>
<td>array</td>
<td>Details on items included in the order (See booking.order.items.item below)</td>
</tr>
</tbody>
</table>

**booking.order.items.item**

An entry for each item in the booking will contain following fields:
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>timestamp</td>
<td>The start date of the booking item</td>
</tr>
<tr>
<td>end_date</td>
<td>timestamp</td>
<td>The end date of the booking item</td>
</tr>
<tr>
<td>sku</td>
<td>string</td>
<td>The unique stock keeping unit of the item</td>
</tr>
<tr>
<td>slip</td>
<td>string</td>
<td>The booking slip attached to the item</td>
</tr>
<tr>
<td>package</td>
<td>integer</td>
<td>The package the item belongs to (if applicable)</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>The booking status of the item</td>
</tr>
<tr>
<td>total</td>
<td>float</td>
<td>The total price of the booking item</td>
</tr>
<tr>
<td>tax_total</td>
<td>float</td>
<td>The sum of taxes applied to this item</td>
</tr>
<tr>
<td>qty</td>
<td>integer</td>
<td>The quantity of this item in the booking</td>
</tr>
</tbody>
</table>

Sample Booking Notification

**JSON**

```json
{
  "@attributes": {
    "version": "3.0",
    "host": "your-company.checkfront.com"
  },
  "booking": {
    "@attributes": {
      "booking_id": "157"
    },
    "status": "HOLD",
    "code": "KMVQ-060314",
    "tid": {},
    "created_date": "1394156618",
    "staff_id": "1",
    "source_ip": "66.228.55.142",
    "start_date": "1394128800",
    "end_date": "1394128800",
    "customer": {
      "code": "BT5-386-579",
      "name": "Your Customer",
      "email": "support@checkfront.com",
      "region": "BC",
      "address": "170-422 Richards Street",
      "city": "Vancouver",
      "country": "CA",
      "phone": "1 (800) 559-0985",
      "postal_zip": "V6B 2Z4"
    },
    "fields": {
      "customer_name": "Your Customer",
      "customer_email": "support@checkfront.com",
      "customer_region": "BC",
      "customer_address": "170-422 Richards Street",
      "customer_city": "Vancouver",
      "customer_country": "CA",
      "customer_phone": "1 (800) 559-0985",
      "customer_postal_zip": "V6B 224",
      "request": "Blue paddleboard if available, please."
    }
  }
}
```
"meta": {  
  "note": "Free icecream!",  
  "mobile_device": {}  
},  
"order": {  
  "@attributes": {  
    "currency_id": "CAD"  
  },  
  "sub_total": "46.00",  
  "tax_total": "2.30",  
  "paid_total": "0.00",  
  "total": "48.30",  
  "taxes": {  
    "@attributes": {  
      "tax_id": "6"  
    },  
    "name": "GST",  
    "amount": "2.30"  
  }  
},  
"items": {  
  "item": [  
    {  
      "@attributes": {  
        "line_id": "1",  
        "item_id": "6"  
      },  
      "start_date": "1394128800",  
      "end_date": "1394128800",  
      "sku": "paddle",  
      "slip": {},  
      "package_id": "1",  
      "status": "HOLD",  
      "total": "6.30",  
      "tax_total": "0.30",  
      "qty": "2"  
    },  
    {  
      "@attributes": {  
        "line_id": "2",  
        "item_id": "8"  
      },  
      "start_date": "1394092800",  
      "end_date": "1394092800",  
      "sku": "paddleboard",  
      "slip": {},  
      "package_id": "1",  
      "status": "HOLD",  
      "total": "42.00",  
      "tax_total": "2.00",  
      "qty": "2"  
    }  
  ]  
}  
(continues on next page)
XML

```xml
<?xml version="1.0" ?>
<checkfront-notify version="3.0" host="your-company.checkfront.com">
  <booking booking_id="157">
    <status>HOLD</status>
    <code>KMVQ-060314</code>
    <tid/>
    <created_date>1394156618</created_date>
    <staff_id>1</staff_id>
    <source_ip>66.228.55.142</source_ip>
    <start_date>1394128800</start_date>
    <end_date>1394128800</end_date>
    <customer>
      <code>BT5-386-579</code>
      <name>Your Customer</name>
      <email>support@checkfront.com</email>
      <region>BC</region>
      <address>170-422 Richards Street</address>
      <city>Vancouver</city>
      <country>CA</country>
      <phone>1 (800) 559-0985</phone>
      <postal_zip>V6B 2Z4</postal_zip>
    </customer>
    <fields>
      <customer_name>Your Customer</customer_name>
      <customer_email>support@checkfront.com</customer_email>
      <customer_region>BC</customer_region>
      <customer_address>170-422 Richards Street</customer_address>
      <customer_city>Vancouver</customer_city>
      <customer_country>CA</customer_country>
      <customer_phone>1 (800) 559-0985</customer_phone>
      <customer_postal_zip>V6B 2Z4</customer_postal_zip>
      <request>Blue paddleboard if available, please.</request>
    </fields>
    <meta>
      <note>Free icecream!</note>
      <mobile_device></mobile_device>
    </meta>
    <order currency_id="CAD">
      <sub_total>46.00</sub_total>
      <tax_total>2.30</tax_total>
      <paid_total>0.00</paid_total>
      <total>48.30</total>
      <taxes>
        <tax tax_id="6">
          <name>GST</name>
          <amount>2.30</amount>
        </tax>
      </taxes>
      <discount>0.00</discount>
      <items>
```

(continues on next page)
3.1.2 Item Calendar Edit

3.1.3 Item Event/Discount Edit

This webhook is triggered whenever any item events or discounts are altered on your account. Because of their similarities, **Item Events and Discounts are both called “rates”**. When fields have specific meanings for either Item Events or Discounts, those differences are defined in the Description column.

A **JSON**, **XML**, or **x-www-form-urlencoded** object containing the following export fields can be found in directly in the **raw body** of the POST request to your server:
### rate

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate_id</td>
<td>integer</td>
<td>A unique code used to refer to this rate</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The name of this rate</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Item Events: “SP” for “Special” events, “SE” for “Seasonal” events. Discounts: “DC” for “Discount”</td>
</tr>
<tr>
<td>start_date</td>
<td>integer</td>
<td>The date (yyyymmdd) when this rate starts to apply</td>
</tr>
<tr>
<td>end_date</td>
<td>integer</td>
<td>The date (yyyymmdd) when this rate stops applying (0 if the rate has no end date)</td>
</tr>
<tr>
<td>span</td>
<td>integer</td>
<td>The “Force item length to the above start and end dates” option (1 if selected, 0 if not selected)</td>
</tr>
<tr>
<td>price_src</td>
<td>string</td>
<td>If applicable, the type of price modification for this rate (“Base Price” is “B”, “Create new Price Point” is “N”, “Dynamic” is “D”, “Yield” is “Yield”)</td>
</tr>
<tr>
<td>dynamic_rate</td>
<td>decimal</td>
<td>If this event is a “Dynamic” price rate (price_src is “D”), this value holds the rate’s percent change or fixed amount price.</td>
</tr>
<tr>
<td>dynamic_type</td>
<td>string</td>
<td>Dynamic Price rates: “P” for percent, “F” for fixed amount</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>Item Events: whether this is an available (“A”) or unavailable (“U”) event. Discounts: always available (“A”)</td>
</tr>
<tr>
<td>repeat_id</td>
<td>string</td>
<td>If the rate is not recurring, this value is blank. Weekly recurring events are “W”, “Always” recurring events are “*”.</td>
</tr>
<tr>
<td>enabled</td>
<td>integer</td>
<td>If the rate is enabled, 1, otherwise 0 when disabled.</td>
</tr>
<tr>
<td>rule_set_id</td>
<td>integer</td>
<td>The ruleset id applying to this event. The default ruleset id is 1.</td>
</tr>
<tr>
<td>vouchers</td>
<td>integer</td>
<td>Discounts: the number of vouchers attached to this discount.</td>
</tr>
<tr>
<td>details</td>
<td>object</td>
<td>See rate.details below.</td>
</tr>
<tr>
<td>repeat</td>
<td>object</td>
<td>See rate.repeat below.</td>
</tr>
<tr>
<td>apply_to</td>
<td>object</td>
<td>See rate.apply_to below.</td>
</tr>
</tbody>
</table>

#### rate.details

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>thresholds</td>
<td>object</td>
<td>If this is rate has a “Yield” price_src, this element will contain a list of objects containing a level and rate pair for each threshold. See rate.details.thresholds below</td>
</tr>
</tbody>
</table>

#### rate.details.thresholds

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshold</td>
<td>object</td>
<td>This contains a pair of level and rate values for a threshold. See rate.details.thresholds.threshold below</td>
</tr>
</tbody>
</table>

### 3.1. Webhooks

23
rate.details.thresholds.threshold

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>integer</td>
<td>The inventory threshold at which this Yield priced rate applies</td>
</tr>
<tr>
<td>rate</td>
<td>decimal</td>
<td>The percentage price multiplier for this threshold</td>
</tr>
</tbody>
</table>

rate.repeat

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>object</td>
<td>If set, this rate repeats on specific days of the week. See rate.repeat.days below.</td>
</tr>
</tbody>
</table>

rate.repeat.days

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;day of week abbreviation&gt; (e.g., mon, tue,..)</td>
<td>integer</td>
<td>The rate repeats for this day of the week.</td>
</tr>
</tbody>
</table>

rate.apply_to

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>items / categories</td>
<td>object</td>
<td>This object contains a list of the items or categories which this rate applies to. See rate.apply_to.items below.</td>
</tr>
</tbody>
</table>

rate.apply_to.items

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>integer</td>
<td>An item_id which this rate applies to</td>
</tr>
</tbody>
</table>

rate.apply_to.categories

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>integer</td>
<td>This contains a category_id which this rate has completely selected. This rate will apply to any new items that are added to this category.</td>
</tr>
</tbody>
</table>

Sample Rate Notification

JSON

```
{
    "version": "3.11.0.0",
    "host": "test.checkfront.com",
    "type": "rate",
    "action": "update",
}
```
"rate": {
    "rate_id": 8,
    "name": "Max 3 Days",
    "start_date": 20161221,
    "end_date": 0,
    "span": 0,
    "price_src": "Yield",
    "dynamic_rate": 10,
    "status": "A",
    "type": "SP",
    "repeat_id": "W",
    "enabled": 1,
    "details": {
        "thresholds": [
            { "threshold": { "level": 5, "rate": 150.5 }
            },
            { "threshold": { "level": 2, "rate": 200 }
            }
        ],
        "repeat": {
            "days": {
                "tue": 1,
                "sat": 1
            }
        },
        "rule_set_id": 1,
        "dynamic_type": "P",
        "apply_to": {
            "items": [
                { "id": 14 },
                { "id": 16 },
                { "id": 35 }
            ],
            "categories": [
                { "id": 3 }
            ]
        }
    }
}
```xml
<?xml version="1.0" encoding="utf-8"?>
<checkfront-notify version="3.11.0.0" host="test.checkfront.com">
    <type>rate</type>
    <action>update</action>
    <rate>
        <rate_id>8</rate_id>
        <name>Max 3 Days</name>
        <start_date>20161221</start_date>
        <end_date>0</end_date>
        <price_src>Yield</price_src>
        <dynamic_rate>10</dynamic_rate>
        <status>A</status>
        <type>SP</type>
        <repeat_id>W</repeat_id>
        <enabled>1</enabled>
        <details>
            <thresholds>
                <threshold>
                    <level>5</level>
                    <rate>150.5</rate>
                </threshold>
                <threshold>
                    <level>2</level>
                    <rate>200</rate>
                </threshold>
            </thresholds>
        </details>
        <repeat>
            <days>
                <tue>1</tue>
                <sat>1</sat>
            </days>
        </repeat>
        <rule_set_id>1</rule_set_id>
        <dynamic_type>P</dynamic_type>
        <apply_to>
            <items>
                <id>14</id>
                <id>16</id>
                <id>35</id>
                <id>36</id>
                <id>37</id>
            </items>
        </apply_to>
    </rate>
</checkfront-notify>
```

(continues on next page)
3.1.4 Item Update

This webhook is triggered whenever any items are altered on your account.

A **JSON**, **XML**, or **x-www-form-urlencoded** object containing the following export fields can be found in directly in the **raw body** of the POST request to your server:
### Item

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
<td>string</td>
<td>The allocation type for the item.</td>
</tr>
<tr>
<td>sku</td>
<td>string</td>
<td>The item’s SKU</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>The Item’s “More Info URL”</td>
</tr>
<tr>
<td>lock</td>
<td>integer</td>
<td>If the item has Date Based Inventory control active. 0 means that Date-based inventory is disabled.</td>
</tr>
<tr>
<td>visibility</td>
<td>string</td>
<td>The visibility of this item on the booking page.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The name of the item.</td>
</tr>
<tr>
<td>pos</td>
<td>integer</td>
<td>The sort order integer.</td>
</tr>
<tr>
<td>meta</td>
<td>object</td>
<td>Contains meta details for the item. This changes based on the add-ons and data for the item.</td>
</tr>
<tr>
<td>stock</td>
<td>integer</td>
<td>The item’s inventory.</td>
</tr>
<tr>
<td>unlimited</td>
<td>integer</td>
<td>This returns if the item’s inventory is unlimited or not. 0 means the inventory is limited.</td>
</tr>
<tr>
<td>video</td>
<td>object</td>
<td>See item.video below.</td>
</tr>
<tr>
<td>image</td>
<td>object</td>
<td>See item.image below.</td>
</tr>
<tr>
<td>category_id</td>
<td>integer</td>
<td>The id of the category this item is in.</td>
</tr>
<tr>
<td>rated</td>
<td>integer</td>
<td>If this item is simple or not. 0 means the item is simple.</td>
</tr>
<tr>
<td>product_group_type</td>
<td>string</td>
<td>The type of item this is in a Product Group. P is for Parent, C is child.</td>
</tr>
<tr>
<td>product_group_children</td>
<td>object</td>
<td>Contains the Item object of the items that are children of the main item.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Determines if this is an Item or a Gift Certificate.</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>This is the availability status for the item.</td>
</tr>
<tr>
<td>alias_id</td>
<td>integer</td>
<td>The ID of the item this is aliased too. This will not show the ID of items aliased to this item.</td>
</tr>
<tr>
<td>len</td>
<td>integer</td>
<td>The fixed length of the item.</td>
</tr>
<tr>
<td>rules</td>
<td>object</td>
<td>Contains the item’s parameters, fixed length, and attribute specific rules such as default length start time, etc.</td>
</tr>
<tr>
<td>category</td>
<td>string</td>
<td>The name of the category this item is in.</td>
</tr>
</tbody>
</table>
item.video

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>This is the identifying string for the Youtube URL.</td>
</tr>
<tr>
<td>start</td>
<td>integer</td>
<td>How many seconds into the video to start.</td>
</tr>
</tbody>
</table>

item.image

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>integer</td>
<td>The index of the image. The images are numbered in the order they were uploaded, but are organized in the order they are displayed on the item.</td>
</tr>
</tbody>
</table>

item.image.images

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>src</td>
<td>string</td>
<td>The identifying string for the image.</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The identifying path to where the image is found relative to your Checkfront Account URL.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>The identifying path to where a full sized version image is found on the server.</td>
</tr>
<tr>
<td>url_medium</td>
<td>string</td>
<td>The identifying path to where a medium version of the image is found on the server.</td>
</tr>
<tr>
<td>url_small</td>
<td>string</td>
<td>The identifying path to where a small version of the image is found on the server.</td>
</tr>
</tbody>
</table>

Sample Item Update Notification

JSON

```json
{
   "version": "3.28.0.7",
   "host": "your-company.checkfront.com",
   "type": "item",
   "action": "update",
   "item": {
      "unit": "30M",
      "item_id": "40",
      "url": "",
      "lock": 1,
      "visibility": "*",
      "name": "Test Item",
      "pos": 0,
      "summary": "",
      "details": "",
   }
}
```

(continues on next page)
"meta": {
  "display_mode": "dropdown",
  "Xero_item_account": "0",
  "item_package_rate": "",
  "delay": 0
},
"stock": 5,
"unlimited": 0,
"video": {
  "id": "",
  "start": 0
},
"image": [],
"category_id": 9,
"rated": 0,
"product_group_type": "P",
"product_group_children": {
  "item_id": "41",
  "sku": "test-item_copy",
  "stock": "1",
  "name": "Test Item Additional Session",
  "category_id": "9",
  "price": {
    "0": {
      "participant": "5.00"
    }
  },
  "price_type": "U",
  "summary": "",
  "url": "",
  "img": "",
  "meta": {
    "display_mode": "dropdown",
    "item_package_rate": "",
    "delay": 0
  },
  "enabled": "1",
  "status": "A",
  "unlimited": "0",
  "alias_id": "40",
  "lock": "1",
  "package": {
    "61": {
      "optin": "O"
    }
  },
  "rated": "1",
  "visible": "1",
  "visibility": "x",
  "taxes": "",
  "video": {
    "id": "",
    "start": 0
  },
  "rules": {
    "param": {
      "inventorycontro": {
        "MIN": "0",
        "MAX": "0"
      }
    },
    "participant": {
      "MIN": "1",
      "MAX": "10"
    }
  },
  "fixed": {
    "start_time": ""
  },
  "unit": "30M",
  "len": "0",
  "param": {
    "inventorycontro": 1,
    "participant": 1
  },
  "ecom": "1",
  "gprice": "",
  "type": "I",
  "price_fixed": "0.00",
  "pos": "0",
  "location": null,
  "details": "",
  "extra_details": "",
  "commission_type": "p",
  "commission_amount": null,
  "product_group_type": "c",
  "product_group_children": ""
}
}
"type": "I",
"status": "A",
"alias_id": 0,
"len": 0,
"rules": "{"inventorycontrol":{"MIN":"4","MAX":"4"},
    "participant":{"MIN":"1","MAX":"10"}},"fixed":{"start_time":""}}
"category": "Programming"}

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<checkfront-notify version="3.29.0.1" host="your-company.checkfront.com">
    <type>item</type>
    <action>update</action>
    <item>
        <unit>30M</unit>
        <item_id>40</item_id>
        <sku>test-item</sku>
        <url></url>
        <lock>1</lock>
        <visibility>*</visibility>
        <name>Test Item</name>
        <pos>0</pos>
        <summary></summary>
        <details></details>
        <meta>
            "display_mode":"dropdown","Xero_item_account":"0","item_package_rate":"",
            "delay":0</meta>
        <stock>5</stock>
        <unlimited>0</unlimited>
        <video>
            <id></id>
            <start>0</start>
            <video/>
            <category_id>9</category_id>
            <rated>0</rated>
        <product_group_type>P</product_group_type>
        <product_group_children>
            <item_id>41</item_id>
            <sku>test-item_copy</sku>
            <stock>1</stock>
            <name>Test Item Additional Session</name>
            <category_id>9</category_id>
            <price>{"0":{"participant":"5.00"}}</price>
            <price_type>U</price_type>
            <summary></summary>
            <url></url>
            <img></img>
            <meta>
            "display_mode":"dropdown","item_package_rate":"","delay":0</meta>
            <enabled>1</enabled>
            <status>A</status>
            <unlimited>0</unlimited>
        </product_group_children>
    </item>
</checkfront-notify>
```

(continues on next page)
3.1.5 New Payment/Refund

This webhook is triggered whenever any transactions are created or refunded on your account.

A JSON, XML, or x-www-form-urlencoded object containing the following export fields can be found in directly in the raw body of the POST request to your server:
payment

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction_id</td>
<td>string</td>
<td>A unique code used to refer to this transaction</td>
</tr>
<tr>
<td>date</td>
<td>timestamp</td>
<td>When this transaction was created</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>Identifies whether the transaction was a payment (PAID) or a refund (REFUND)</td>
</tr>
<tr>
<td>amount</td>
<td>decimal</td>
<td>The transaction’s monetary amount</td>
</tr>
<tr>
<td>gateway_id</td>
<td>string</td>
<td>Identifies which payment gateway made the transaction</td>
</tr>
<tr>
<td>payment_mask</td>
<td>integer</td>
<td>The last four digits of the card, or (POS) for a POS transaction</td>
</tr>
<tr>
<td>payment_type</td>
<td>string</td>
<td>The type of card used, or the type of POS transaction</td>
</tr>
<tr>
<td>payment_customer</td>
<td>string</td>
<td>The payer’s name</td>
</tr>
<tr>
<td>booking_id</td>
<td>string</td>
<td>The code of the booking which received this transaction (See booking)</td>
</tr>
</tbody>
</table>

Sample Payment Notification

**JSON**

```
{
    "version": "3.11.0.0",
    "host": "your-company.checkfront.com",
    "type": "transaction",
    "action": "insert",
    "transaction": {
        "transaction_id": "TEST1482283550",
        "date": 1582283550,
        "status": "PAID",
        "amount": "27.00",
        "gateway_id": "TestPayment",
        "payment_mask": "1111",
        "payment_type": "Visa",
        "payment_customer": "Testy McTestface",
        "booking_id": "LULZ-080916"
    }
}
```

**XML**

```
<?xml version="1.0" encoding="utf-8"?>
<checkfront-notify version="3.11.0.0" host="your-website.checkfront.com">
    <type>transaction</type>
    <action>insert</action>
    <transaction>
        <transaction_id>TEST1482283550</transaction_id>
        <date>1582283550</date>
        <status>PAID</status>
        <amount>27.00</amount>
        <gateway_id>TestPayment</gateway_id>
        <payment_mask>1111</payment_mask>
    </transaction>
</checkfront-notify>
```
<payment_type>Visa</payment_type>
<payment_customer>Testy McTestface</payment_customer>
<booking_id>LULZ-080916</booking_id>
</transaction>
</checkfront-notify>

## 3.2 API 3.0 Endpoints

### 3.2.1 account

**GET /api/3.0/account**

Retrieve a list detailing all staff accounts in your system.

```json
{
  "version": "3.0",
  "account_id": 1,
  "host_id": "your-company.checkfront.com",
  "name": "Your Company",
  "locale": {
    "id": "en_US",
    "lang": "en",
    "currency": "CAD"
  },
  "request": {
    "status": "OK",
    "resource": "account",
    "records": 1,
    "limit": 0,
    "page": 1,
    "pages": 1,
    "time": 0.0013,
    "method": "get"
  },
  "account": [
    {
      "account_id": 1,
      "first_name": "Johnson",
      "last_name": "Smith",
      "email": "support@checkfront.com",
      "mobile_phone": ",",
      "nickname": "John",
      "login_id": "johnsmith",
      "enabled": 1,
      "date_created": 1490573063,
      "date_last_login": 1490573064,
      "ip_last_login": "0.0.0.0",
      "feed_token": "354629ce317f34e6d7208d4847137f371deae824625ee1635009b7b88ed8804",
      "partner": 0
    }
  ]
}
```
3.2.2 booking

A **booking** in the system represents an order and all associated information. It’s possible to list, retrieve, create, and modify bookings in the system through use of the API.

**booking**

**GET /api/3.0/booking**

Retrieve a listing of bookings in the system.

Query timestamps may be Date strings or unix timestamps, and can be prefixed with ‘<’ or ‘>’ to match before or after a date.

**Query Parameters**

- **status_id (string)** – The current status of a booking.
- **customer_id (integer)** – The customer id associated with the booking.
- **customer_email (string)** – The customer email associated with the booking.
- **start_date (string/timestamp)** – The date the booking starts on (i.e. check-in).
- **end_date (string/timestamp)** – The date the booking ends on (i.e. check-out).
- **created_date (string/timestamp)** – The date the booking was created.
- **last_modified (string/timestamp)** – The date the booking was last changed. Useful for getting bookings added or changed since your last call.
- **limit (integer)** – The limit of bookings to return per page (default: 100).
- **page (integer)** – The page of results to return.

**Response JSON Array of Objects**

- **booking_id (integer)** – The internal index of a booking.
- **code (string)** – The unique booking reference code.
- **status_id (string)** – The status code of the booking.
- **status_name (string)** – The display name of the booking status.
- **created_date (timestamp)** – The date/time the booking was created.
- **total (float)** – The full value of the booking.
- **tax_total (float)** – The sum of the taxes applied to the booking.
- **paid_total (float)** – The amount the customer has paid on the booking.
- **customer_name (string)** – The full name of the booking contact.
- **customer_email (string)** – The email address of the booking contact.
- **summary (string)** – A brief description of the contents of the booking.
- **date_desc (string)** – A string describing the booked dates/times.
- **token (string)** – A customer token for the booking, can be used to build links to customer portions of the reservation system.
{
   "booking/index": {
      "3149": {
         "booking_id": 3149,
         "code": "PVCA-050314",
         "status_id": "PAID",
         "status_name": "Paid",
         "created_date": 1394079749,
         "total": "141.90",
         "tax_total": "12.90",
         "paid_total": "141.90",
         "customer_name": "Your Customer",
         "customer_email": "support@checkfront.com",
         "summary": "Islands Snorkeling Tour",
         "date_desc": "Wed Mar 5, 2014",
         "token": "57c2ac0a58dc02a72df50a8841b1c8a190627ff69a2435060a3ed3d469819477"
      }
   }
}

Booking endpoints

booking/[booking_id]

GET /api/3.0/booking/{booking_id}
Retrieve extended information on a specific booking.

Parameters

- booking_id(string) – The unique booking code identifying a booking in the system.

booking/[booking_id]/invoice

GET /api/3.0/booking/{booking_id}/invoice
Return a pre-formatted invoice to display to the customer.

Parameters

- booking_id(string) – The unique booking code identifying a booking in the system.

booking/[booking_id]/checkin

You can check-in and check-out a booking. By default, a note is created under the name of the account when a booking is either checked in or checked out. VOID, and CANCELLED bookings cannot be checked-in or out.

POST /api/3.0/booking/{booking_id}/checkin

Parameters

- booking_id(string) – The unique booking code identifying a booking in the system.
booking/[booking_id]/checkout

POST /api/3.0/booking/{booking_id}/checkout

Parameters
  • booking_id (string) – The unique booking code identifying a booking in the system.

booking/[booking_id]/update

The state of an existing booking can be modified using calls to booking/update. Specifying the status_id is required, but any other update fields are optional.

POST /api/3.0/booking/{booking_id}/update

Parameters
  • booking_id (string) – The unique booking code identifying a booking in the system.

Query Parameters
  • status_id (string) – The status that this booking should be set to. See Manage / Layout / Statuses in your account for a list of all available statuses. The default available statuses are: PEND, HOLD, PART, PAID, WAIT, STOP, and VOID
  • notify (boolean) – Toggle whether to trigger notifications when this booking is changed. (default: false)
  • set_paid (boolean) – When set to 1 (true) on an unpaid booking, and the requested status_id is ‘PAID’, attempt to create a POS transaction covering the remaining cost of the booking (cannot be used with other input).

booking/[booking_id]/bookmark

Bookmarks are made available in the Checkfront mobile apps, and are listed under bookings while logged into the platform. You can add or remove a bookmark to a specific booking for the account you are acting on behalf of.

POST /api/3.0/booking/{booking_id}/bookmark

Parameters
  • booking_id (string) – The unique booking code identifying a booking in the system.

Query Parameters
  • mark (boolean) – Enable or disable the bookmark.

booking/[booking_id]/note

Notes can be added to bookings for reference. These will be logged in the booking as being made by the account you are acting on behalf of, so can identify comments made by individual staff members.

POST /api/3.0/booking/{booking_id}/note

Parameters
  • booking_id (string) – The unique booking code identifying a booking in the system.

Query Parameters
• **body (string)** – The text to include in your booking note. Up to 3000 chars.

**booking/create**

A call to booking/create must pass in at least one *SLIP* - either directly as an array of slips, or by using a **session_id** from a previously created *booking/session* containing at least one *SLIP*. Additionally, the call must pass in any customer input entered to fields from *booking/form*, for all required fields and any optional fields.

**POST /api/3.0/booking/create**

Attempt to create a booking from an existing session, with customer form input sent in the request in the “form” parameter.

**Form Parameters**

- **array form** – An array of fields containing customer details matching the required and optional booking fields (e.g. `form[customer_name]="John Smith"`)
- **string session_id** – The session ID containing the booking items to be committed. *Can also be sent as a cookie.*
- **string/array slip** – A *SLIP* or array of *SLIPS* that can be passed directly to booking/create, bypassing the need to specify a **session_id**

**booking/form**

This call can easily be cached if your configuration does not often change, but depending on your setup, a number of the fields returned in this request may be required to complete a booking.

You will receive an error indicating missing information if you attempt to call booking/create without the required fields, either those required by staff if you’re acting on behalf of a staff member, or by a customer otherwise.

**GET /api/3.0/booking/form**

Retrieve the customer details input form for the booking. Some fields may be required to call booking/create.

**Response JSON Object**

- **booking_form_ui** – An array of form field definitions

```json
{
  "version": "3.0",
  "account_id": 1,
  "host_id": "your-company.checkfront.com",
  "name": "Your Company",
  "locale": {
    "id": "en_US",
    "lang": "en",
    "currency": "CAD"
  },
  "request": {
    "status": ""
  },
  "booking_form_ui": {
    "msg": [],
    "errors": 0,
    "mode": "html",
    "_cnf": [],
    "customer_name": {
```
"value": "",
"define": {
  "required": 1,
  "layout": {
    "staff": {
      "invoice": 1,
      "form": 1,
      "required": 1
    },
    "customer": {
      "invoice": 1,
      "form": 1,
      "required": 1
    },
    "sys": 0,
    "lbl": "Name",
    "type": "text"
  }
},
"booking_policy": {
  "body": "This is your <strong style='color: firebrick'>sample booking policy</strong>. It's included in customer e-mails, your booking receipt page and in PDF print outs.

You can edit this policy by logging into your Checkfront account, and navigating to Manage \ Layout \ Invoice."

"fields": {
  "customer_tos_agree": {
    "value": 0,
    "define": {
      "required": 1,
      "layout": {
        "staff": {
          "invoice": 1,
          "form": 1,
          "required": 0
        },
        "customer": {
          "invoice": 1,
          "form": 1,
          "required": 1
        },
        "lbl": "I have read and agreed to the Terms of Service",
        "type": "checkbox"
      }
    }
  },
  "customer_email": {
    "value": "",
    "define": {
      "mask": "email",
      "layout": {
        "staff": {
          "invoice": 1,
          "form": 1,
          "required": 1
        },
        "customer": {
          "invoice": 1,
          "form": 1,
          "required": 1
        }
      }
    }
  }
}
"required": 0
},
"customer": {
    "invoice": 1,
    "form": 1,
    "required": 1
},
"sys": 0,
"lbl": "E-mail",
"type": "text"
},
"required": 0
}
},
"customer_phone": {
    "value": "",
    "define": {
        "mask": "simple",
        "layout": {
            "staff": {
                "invoice": 1,
                "form": 1,
                "required": 0
            },
            "customer": {
                "invoice": 1,
                "form": 1,
                "required": 0
            },
            "sys": 0,
            "lbl": "Phone",
            "type": "text",
            "customer_address": true
        },
        "required": 0
    }
},
"required": 0
}
},
"customer_address": {
    "value": "",
    "define": {
        "layout": {
            "staff": {
                "invoice": 1,
                "form": 1,
                "required": 0
            },
            "customer": {
                "invoice": 1,
                "form": 1,
                "required": 0
            },
            "sys": 0,
            "lbl": "Address",
            "type": "text",
            "customer_address": true
        },
        "required": 0
    }
}
"customer_city": {
  "value": "",
  "define": {
    "layout": {
      "staff": {
        "invoice": 1,
        "form": 1,
        "required": 0
      },
      "customer": {
        "invoice": 1,
        "form": 1,
        "required": 0
      },
      "sys": 0,
      "lbl": "City",
      "type": "text",
      "customer_address": true
    },
    "required": 0
  }
},
"customer_country": {
  "value": "CA",
  "define": {
    "layout": {
      "staff": {
        "invoice": 1,
        "form": 1,
        "required": 0
      },
      "customer": {
        "invoice": 1,
        "form": 1,
        "required": 0
      },
      "sys": 0,
      "lbl": "Country",
      "type": "select",
      "customer_address": true,
      "options": {
        "AF": "Afghanistan",
        "AL": "Albania",
        "DZ": "Algeria",
        "AS": "American Samoa",
        "AD": "Andorra",
        "AO": "Angola",
        "AG": "Antigua And Barbuda",
        "AR": "Argentina",
        "AM": "Armenia",
        "AU": "Australia",
        "AT": "Austria",
        "AZ": "Azerbaijan",
        "BS": "Bahamas",
        "BH": "Bahrain",
        (continues on next page)
"BD": "Bangladesh",
"BB": "Barbados",
"BY": "Belarus",
"BE": "Belgium",
"BZ": "Belize",
"BJ": "Benin",
"BM": "Bermuda",
"BT": "Bhutan",
"BO": "Bolivia",
"BA": "Bosnia And Herzegovina",
"BW": "Botswana",
"BR": "Brazil",
"BN": "Brunei Darussalam",
"BG": "Bulgaria",
"BF": "Burkina Faso",
"BI": "Burundi",
"KH": "Cambodia",
"CM": "Cameroon",
"CA": "Canada",
"CV": "Cape Verde",
"KY": "Cayman Islands",
"CF": "Central African Republic",
"TD": "Chad",
"CL": "Chile",
"CN": "China",
"CO": "Colombia",
"KM": "Comoros",
"CD": "Congo (DC)",
"CG": "Congo (R)",
"CK": "Cook Islands",
"CR": "Costa Rica",
"HR": "Croatia",
"CU": "Cuba",
"CY": "Cyprus",
"CZ": "Czech Republic",
"DK": "Denmark",
"DJ": "Djibouti",
"DM": "Dominica",
"DO": "Dominican Republic",
"TP": "East Timor",
"EC": "Ecuador",
"EG": "Egypt",
"SV": "El Salvador",
"GQ": "Equatorial Guinea",
"ER": "Eritrea",
"EE": "Estonia",
"ET": "Ethiopia",
"FK": "Falkland Islands",
"FO": "Faroe Islands",
"FJ": "Fiji",
"FI": "Finland",
"FR": "France",
"PF": "French Polynesia",
"GA": "Gabon",
"GM": "Gambia",
"GE": "Georgia",
"DE": "Germany"
"GH": "Ghana",
"GI": "Gibraltar",
"GR": "Greece",
"GL": "Greenland",
"GD": "Grenada",
"GP": "Guadeloupe",
"GU": "Guam",
"GT": "Guatemala",
"GN": "Guinea",
"GW": "Guinea-bissau",
"HT": "Haiti",
"VA": "Holy See",
"HN": "Honduras",
"HK": "Hong Kong",
"HU": "Hungary",
"IS": "Iceland",
"IN": "India",
"ID": "Indonesia",
"IR": "Iran",
"IQ": "Iraq",
"IE": "Ireland",
"IL": "Israel",
"IT": "Italy",
"CI": "Ivory coast",
"JM": "Jamaica",
"JP": "Japan",
"JO": "Jordan",
"KZ": "Kazakhstan",
"KE": "Kenya",
"KI": "Kiribati",
"KW": "Kuwait",
"KG": "Kyrgyzstan",
"LA": "Laos",
"LV": "Latvia",
"LB": "Lebanon",
"LS": "Lesotho",
"LR": "Liberia",
"LI": "Liechtenstein",
"LT": "Lithuania",
"LU": "Luxembourg",
"LY": "Libya",
"MO": "Macao",
"MK": "Macedonia",
"MG": "Madagascar",
"MW": "Malawi",
"MY": "Malaysia",
"MV": "Maldives",
"ML": "Mali",
"MT": "Malta",
"MQ": "Martinique",
"MR": "Mauritania",
"MU": "Mauritius",
"MX": "Mexico",
"MD": "Moldova",
"MC": "Monaco",
"MN": "Mongolia",
"MS": "Montserrat"
"MA": "Morocco",
"MZ": "Mozambique",
"MM": "Myanmar",
"NA": "Namibia",
"NR": "Nauru",
"NP": "Nepal",
"NL": "Netherlands",
"AN": "Netherlands Antilles",
"NC": "New Caledonia",
"NZ": "New Zealand",
"NI": "Nicaragua",
"NE": "Niger",
"NG": "Nigeria",
"KP": "North Korea",
"MP": "Northern Mariana Islands",
"NO": "Norway",
"OM": "Oman",
"PK": "Pakistan",
"PW": "Palau",
"PS": "Palestine",
"PA": "Panama",
"PG": "Papua New Guinea",
"PY": "Paraguay",
"PE": "Peru",
"PH": "Philippines",
"PL": "Poland",
"PT": "Portugal",
"PR": "Puerto Rico",
"QA": "Qatar",
"RE": "Reunion",
"RO": "Romania",
"RU": "Russian Federation",
"RW": "Rwanda",
"SPM": "Saint-Pierre et Miquelon",
"WS": "Samoa",
"SM": "San Marino",
"ST": "Sao Tome And Principe",
"SA": "Saudi Arabia",
"SN": "Senegal",
"YU": "Serbia and Montenegro",
"SC": "Seychelles",
"SL": "Sierra Leone",
"SG": "Singapore",
"SK": "Slovakia",
"SI": "Slovenia",
"SB": "Solomon Islands",
"SO": "Somalia",
"ZA": "South Africa",
"KR": "South Korea",
"ES": "Spain",
"LK": "Sri Lanka",
"VC": "St Vincent And The Grenadines",
"SD": "Sudan",
"SR": "Suriname",
"SZ": "Swaziland",
"SE": "Sweden",
"CH": "Switzerland"
"SY": "Syria",
"TW": "Taiwan",
"TJ": "Tajikistan",
"TZ": "Tanzania",
"TH": "Thailand",
"TG": "Togo",
"TO": "Tonga",
"TT": "Trinidad And Tobago",
"TN": "Tunisia",
"TR": "Turkey",
"TM": "Turkmenistan",
"TV": "Tuvalu",
"UG": "Uganda",
"UA": "Ukraine",
"AE": "United Arab Emirates",
"GB": "United Kingdom",
"US": "United States",
"UY": "Uruguay",
"UZ": "Uzbekistan",
"VU": "Vanuatu",
"VE": "Venezuela",
"VN": "Vietnam",
"VG": "Virgin Islands, British",
"VI": "Virgin Islands, U.S.",
"YE": "Yemen",
"ZM": "Zambia",
"ZW": "Zimbabwe"
}

"required": 0
}

"customer_region": {
  "value": "BC",
  "define": {
    "layout": {
      "staff": {
        "invoice": 1,
        "form": 1,
        "required": 0
      },
      "customer": {
        "invoice": 1,
        "form": 1,
        "required": 0
      }
    },
    "sys": 0,
    "lbl": "Province",
    "type": "select",
    "customer_address": true,
    "options": {
      "AB": "Alberta",
      "BC": "British Columbia",
      "MB": "Manitoba",
      "NB": "New Brunswick",
      "NF": "Newfoundland",
      "NWT": "Northwest Territories",
    }
  }
}
"NS": "Nova Scotia",
"NU": "Nunavut",
"ON": "Ontario",
"PE": "Prince-Edward Island",
"QC": "Quebec",
"SK": "Saskatchewan",
"YK": "Yukon"
}
},
"required": 0
},
"customer_postal_zip": {
  "value": "",
  "define": {
    "mask": "simple",
    "layout": {
      "staff": {
        "invoice": 1,
        "form": 1,
        "required": 0
      },
      "customer": {
        "invoice": 1,
        "form": 1,
        "required": 0
      },
      "sys": 0,
      "lbl": "Postal code",
      "type": "text",
      "customer_address": true
    },
    "required": 0
  }
},

"note": {
  "value": "",
  "define": {
    "layout": {
      "staff": {
        "invoice": 0,
        "form": 1,
        "required": 0
      },
      "customer": {
        "invoice": 0,
        "form": 1,
        "required": 0
      },
      "sys": 1,
      "lbl": "Note",
      "type": "textarea"
    },
    "required": 0
  }
}
booking/index

**GET /api/3.0/booking/index**
Retrieve a listing of bookings in the system.

Query timestamps may be Date strings or unix timestamps, and can be prefixed with `<` or `>` to match before or after a date.

**Query Parameters**
- **status_id** *(string)* – The current status of a booking.
- **customer_id** *(integer)* – The customer id associated with the booking.
- **customer_email** *(string)* – The customer email associated with the booking.
- **start_date** *(string/timestamp)* – The date the booking starts on (i.e. check-in).
- **end_date** *(string/timestamp)* – The date the booking ends on (i.e. check-out).
- **created_date** *(string/timestamp)* – The date the booking was created.
- **last_modified** *(string/timestamp)* – The date the booking was last changed. Useful for getting bookings added or changed since your last call.
- **partner_id** *(string)* – The partner account ID that a booking is attributed to.
- **item_id** *(integer)* – The ID of an item attached to the booking.
- **limit** *(integer)* – The limit of bookings to return per page (default: 100).
- **page** *(integer)* – The page of results to return.

**Response JSON Array of Objects**
- **booking_id** *(integer)* – The internal index of a booking.
- **code** *(string)* – The unique booking reference code.
- **status_id** *(string)* – The status code of the booking.
- **status_name** *(string)* – The display name of the booking status.
- **created_date** *(timestamp)* – The date/time the booking was created.
- **total** *(float)* – The full value of the booking.
- **tax_total** *(float)* – The sum of the taxes applied to the booking.
- **paid_total** *(float)* – The amount the customer has paid on the booking.
- **customer_name** *(string)* – The full name of the booking contact.
- **customer_email** *(string)* – The email address of the booking contact.
- **summary** *(string)* – A brief description of the contents of the booking.
- **date_desc** *(string)* – A string describing the booked dates/times.
- **token** *(string)* – A customer token for the booking, can be used to build links to customer portions of the reservation system.
booking/index

This is a call to your Checkfront account that pulls all the notes from all of your bookings. This defaults to pulling all of the notes from your bookings for the past thirty days.

**POST /api/3.0/booking/notes**

**Query Parameters**
Checkfront API Documentation, Release 3.0

- **date (string/timestamp)** – Filter the booking notes by a date. This can be in ISO-8601 or “YYYY-MM-DD” format. This can also be a month and will show all notes from that month.

Response JSON Array of Objects

- **note_id (integer)** – the ID of the note on that booking.
- **account_id (integer)** – If made by a staff or partner account, this represents their unique identifier, otherwise it represents a customer booking with the value of 0.
- **date (string)** – The date the note was created. It appears in ISO-8601 format.
- **body (string)** – The body of the note.
- **private (bool)** – The visibility of the note on the invoice. True means it is hidden from the invoice, false means it shows.
- **booking_id (string)** – The unique identifier for the booking that contains the note.

```json
{
    "version": "3.0",
    "account_id": 1,
    "host_id": "your-company.checkfront.com",
    "name": "Your Company",
    "locale": {
        "id": "en_CA",
        "lang": "en",
        "currency": "CAD"
    },
    "request": {
        "status": "OK",
        "resource": "booking",
        "id": "notes",
        "records": 26,
        "limit": 0,
        "page": 1,
        "pages": 1,
        "time": 0.003,
        "timestamp": 1524984437.018568,
        "method": "get"
    },
    "notes": [
        {
            "note_id": 1,
            "account_id": 0,
            "date": "2018-04-26T17:40:25-07:00",
            "body": "Quis error quo molestias rerum",
            "private": false,
            "booking_id": "PSGS-260418"
        },
        {
            "note_id": 3,
            "account_id": 1,
            "date": "2018-04-21T23:30:19-07:00",
            "body": "test 2",
            "private": true,
            "booking_id": "LLXF-230318"
        }
    ]
}
```

3.2. API 3.0 Endpoints
booking/session

While creating a booking, the details of the proposed booking are stored in an API session. This allows you to add multiple items to a booking, remove items and inform the server of your intent to book the item(s) in order to prevent over-bookings.

Each session call will return the session_id of the active session, and any attached session information. Your application must use the session_id in order to further modify the session or submit the booking beyond this point, and can pass it back to the API either as a cookie or a request parameter.

To start a new session, you simply pass in the booking SLIPs returned when from earlier “rated” inventory item queries.

POST /api/3.0/booking/session

Create or return information about and stored in the current server session. Once initiated, you can fetch the details of a proposed booking by accessing the session again with the session_id found in the response. The item details for the session will be returned with your request.

Form Parameters

- **string session_id** – The session ID to be loaded or written to. *Can also be sent as a cookie.*
- **string/array slip** – A SLIP, or multiple SLIPs to be added to a session. If need to add or remove more of the same item to a session, use alter below.
- **array alter** – Alterations to be made on the current session, based on the line_id of items in the session and actions to be taken. To change the qty of an item in the session, send an alter for the line_id of that item with the integer value you need to set (e.g alter[3]=5 sets item 3 to a qty of 5). To remove an item, use alter ‘remove’; and to opt-in or out of listed package items use ‘optin’ or ‘optout’ respectively.

GET /api/3.0/booking/session

Query Parameters

- **session_id**(string) – The session ID to read information from. *Can also be sent as a cookie.*
"booking": {
  "session": {
    "id": "60rq7eme0tmhimpkeg2fb2p287",
    "age": 0,
    "summary": "Bungee Jumping from the Top of the O2 Arena",
    "start_date": 1507186800,
    "end_date": 1507186800,
    "date_desc": "Thu Oct 5, 2017",
    "time_desc": "",
    "account_id": 1,
    "partner_id": 0,
    "currency_id": "CAD",
    "sub_total": "130.00",
    "tax_total": "0.00",
    "tax_inc_total": "0.00",
    "discount": "0.00",
    "total": "130.00",
    "due": "130.00",
    "paid_total": "0.00",
    "flat_discount": {
      "total": 0,
      "total_pretax": 0
    },
    "deposit": null,
    "tax": [],
    "qty": 1,
    "item": {
      "1": {
        "sku": "bungeejumpingfromthetopoftheo2arena",
        "item_id": 18,
        "name": "Bungee Jumping from the Top of the O2 Arena",
        "unit": "D",
        "rate": {
          "total": "130.00",
          "item_total": "130.00",
          "qty": 1,
          "summary": "<strong title="Adults">Adults:</strong> 1 Day @ $130.00"
        },
        "date": {
          "summary": "Thu Oct 5, 2017",
          "start_date": 1507186800,
          "end_date": 1507186800
        },
        "slip": "18.20171005X1-adults.1-children.0",
        "available": 9
      },
      "1.1": {
        "sku": "memorabiliaphoto",
        "item_id": 21,
        "name": "Memorabilia - Photo Package",
        "unit": "D",
        "rate": {
          "total": "18.00",
          "item_total": "0.00",
          "qty": 1,
          "summary": "$18.00"
        }
      }
    }
  }
}
{ },
   "date": {
      "summary": "Thu Oct 5, 2017",
      "start_date": 1507186800,
      "end_date": 1507186800
   },
   "slip": "21.20171005X1-1",
   "available": 1000000000,
   "optin": "O",
   "opt": "out"
},
"1.2": {
   "sku": "thelondonexperiencet-hat",
   "item_id": 20,
   "name": "The London Experience Hat",
   "unit": "D",
   "rate": {
      "total": "15.00",
      "item_total": "0.00",
      "qty": 1,
      "summary": "$15.00"
   },
   "date": {
      "summary": "Thu Oct 5, 2017",
      "start_date": 1507186800,
      "end_date": 1507186800
   },
   "slip": "20.20171005X1-1",
   "available": 1000000000,
   "optin": "O",
   "opt": "out"
},
"1.3": {
   "sku": "thelondonexperiencet-shirt",
   "item_id": 19,
   "name": "The London Experience Mini-Bus",
   "unit": "D",
   "rate": {
      "total": "25.00",
      "item_total": "0.00",
      "qty": 1,
      "summary": "$25.00"
   },
   "date": {
      "summary": "Thu Oct 5, 2017",
      "start_date": 1507186800,
      "end_date": 1507186800
   },
   "slip": "19.20171005X1-1",
   "available": 1000000000,
   "optin": "O",
   "opt": "out"
}
}
POST /api/3.0/booking/session/clear
POST /api/3.0/booking/session/end

Query Parameters
- `session_id` (string) – The session ID to empty or close. *Can also be sent as a cookie.*

### 3.2.3 category

GET /api/3.0/category

Retrieve a list of the available categories in the system.

Response JSON Array of Objects
- `category_id` (integer) – The unique ID of the category.
- `name` (string) – The display name of the category.
- `pos` (integer) – The order in which the category will display.

```json
{
  "version": "3.0",
  "account_id": 1,
  "host_id": "your-company.checkfront.com",
  "name": "Your Company",
  "locale": {
    "id": "en_US",
    "lang": "en",
    "currency": "CAD"
  },
  "request": {
    "status": "OK",
    "resource": "category",
    "records": 6,
    "limit": 0,
    "page": 1,
    "pages": 1,
    "time": 0.0028,
    "method": "get"
  },
  "query": [],
  "category": {
    "2": {
      "category_id": 2,
      "name": "Tours",
      "description": "From Walking, to Bicycles, to Double Decker buses, we have the perfect tour for you and your family."
    },
    "4": {
      "category_id": 4,
      "name": "Accommodations",
      "description": "Book a Double or Single room for your overnight stay in Manchester."
    }
  }
}
```

(continues on next page)
3.2.4 company

GET /api/3.0/company

Retrieve company configuration information of the system you are connecting to.

Response JSON Object

- **url (string)** – The Checkfront URL of the account.
- **name (string)** – The display name of the company.
- **plan_id (integer)** – The Checkfront subscription plan ID of the account.
- **email (string)** – The contact email address of the company.
- **address (string)** – The street address of the company.
- **city (string)** – The city where the company is located.
- **postal_zip**(string) – The postal or zip code of the company.
- **region**(string) – The state/province of the company.
- **region_id**(string) – The state/province of the company.
- **country_id**(string) – The country where the company is based.
- **currency_id**(string) – The currency type accepted by the company.
- **timezone**(string) – The timezone of the company.
- **locale_id**(string) – The locale string set for the company.
- **lang_id**(string) – The ISO 639-1 language code of the company.
- **date_format**(string) – The date format set for the company.
- **time_format**(string) – The time format set for the company.

```json
{
    "version": "3.0",
    "account_id": 1,
    "host_id": "your-company.checkfront.com",
    "name": "Your Company",
    "locale": {
        "id": "en_GB",
        "lang": "en",
        "currency": "CAD"
    },
    "request": {
        "status": "OK",
        "resource": "company",
        "records": 19,
        "limit": 0,
        "page": 1,
        "pages": 1,
        "time": 0.0016,
        "method": "get"
    },
    "company": {
        "url": "your-company.checkfront.com",
        "name": "Your Company",
        "plan_id": 1,
        "email": "hello@checkfront.com",
        "address": "",
        "city": "",
        "postal_zip": "",
        "region": "LND",
        "country_id": "CA",
        "currency_id": "CAD",
        "timezone": "Canada/Pacific",
        "locale_id": "en_GB",
        "lang_id": "en",
        "date_format": "%m/%d/%Y",
        "time_format": "%I:%M %p",
        "phone": "",
        "utc_offset": -7,
        "currency": {
            "symbol": ">
```
3.2.5 customer

The customer object provides access to customer details stored in the system. Customer information is automatically stored in the system when a new booking is made.

To retrieve information on a specific customer, you should ideally use the customer ID in a GET call to customer/[customer_id]. In this case, the customer information will be returned as “customer” in the JSON response.

If you do not have the customer ID stored, you can also look up customers based on indexed fields, including email address, name, and phone number. These must be an exact match, but can return multiple results on non-unique fields such as the customer name. The results will be stored in the “customers” object in the JSON response.

GET /api/3.0/customer
Retrieve a list of customers.

Query Parameters

- **customer_id** (**string**) – The unique customer code identifying a customer in the system.
- **customer_email** (**string**) – A unique customer e-mail address to search.
- **customer_name** (**string**) – The full name of the customer to search.
- **customer_phone** (**string**) – The (exact) customer phone number to search.

GET /api/3.0/customer/{customer_id}
Retrieve contact details for a customer.

Parameters

- **customer_id** (**string**) – The unique customer code identifying a customer in the system.
### 3.2.6 event

Events are used to modify the price or availability of your inventory items based on date. The `event` endpoint allows you to retrieve information on events in the system, and see which items they are applied to.

**GET /api/3.0/event**

Retrieve a list of all events in the system.

**Response JSON Array of Objects**

- `events` *(array)* – A list of all events in the system. See *Event Object Structure* below.

**GET /api/3.0/event/{id}**

Retrieve data on a specific event.

**Response JSON Array of Objects**

- `event` *(object)* – Information about the requested event. See *Event Object Structure* below.
Event Object Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate_id</td>
<td>integer</td>
<td>A unique code used to refer to this event</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The name of this event</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Item Events: “SP” for “Special” events, “SE” for “Seasonal” events. Discounts: “DC” for “Discount”</td>
</tr>
<tr>
<td>start_date</td>
<td>integer</td>
<td>The date (yyyyymmdd) when this event starts to apply</td>
</tr>
<tr>
<td>end_date</td>
<td>integer</td>
<td>The date (yyyyymmdd) when this event stops applying (0 if the event has no end date)</td>
</tr>
<tr>
<td>span</td>
<td>integer</td>
<td>The “Force item length to the above start and end dates” option (1 if selected, 0 if not selected)</td>
</tr>
<tr>
<td>price_src</td>
<td>string</td>
<td>If applicable, the type of price modification for this event (“Base Price” is “B”, “Create new Price Point” is “N”, “Dynamic” is “D”, “Yield” is “Yield”)</td>
</tr>
<tr>
<td>dynamic_rate</td>
<td>decimal</td>
<td>If this event is a “Dynamic” price event (price_src is “D”), this value holds the event’s percent change or fixed amount price.</td>
</tr>
<tr>
<td>dynamic_type</td>
<td>string</td>
<td>Dynamic Price events: “P” for percent, “F” for fixed amount</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>Item Events: whether this is an available (“A”) or unavailable (“U”) event. Discounts: always available (“A”)</td>
</tr>
<tr>
<td>repeat_id</td>
<td>string</td>
<td>If the event is not recurring, this value is blank. Weekly recurring events are “W”, “Always” recurring events are “*”.</td>
</tr>
<tr>
<td>enabled</td>
<td>integer</td>
<td>If the event is enabled, 1, otherwise 0 when disabled.</td>
</tr>
<tr>
<td>rule_set_id</td>
<td>integer</td>
<td>The ruleset id applying to this event. The default ruleset id is 1.</td>
</tr>
<tr>
<td>vouchers</td>
<td>integer</td>
<td>Discounts: the number of vouchers attached to this discount.</td>
</tr>
<tr>
<td>details</td>
<td>array</td>
<td>If this is event has a “Yield” price_src, this element will contain a list of objects containing a level and rate pair for each threshold. See event.details objects below</td>
</tr>
<tr>
<td>repeat</td>
<td>array</td>
<td>A list of days (mon, tue, …) for which this event repeats.</td>
</tr>
<tr>
<td>apply_to</td>
<td>object</td>
<td>See event.apply_to below.</td>
</tr>
</tbody>
</table>

**event.details objects**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>integer</td>
<td>The inventory threshold at which this Yield priced event applies</td>
</tr>
<tr>
<td>rate</td>
<td>decimal</td>
<td>The percentage price multiplier for this threshold</td>
</tr>
</tbody>
</table>
### event.apply_to

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_id</td>
<td>array</td>
<td>This array contains the list of item IDs which this event applies to.</td>
</tr>
<tr>
<td>category_id</td>
<td>array</td>
<td>This array contains the list of category IDs which this event applies to. This event will apply to any new items that are added to this category.</td>
</tr>
</tbody>
</table>

#### Sample Event Object

```
{
    "version": "3.0",
    "account_id": 1,
    "host_id": "your-company.checkfront.com",
    "name": "Your Company",
    "locale": {
        "id": "en_US",
        "lang": "en",
        "currency": "CAD"
    },
    "request": {
        "path": [
            "event"
        ],
        "resource": "event",
        "records": 6,
        "total_records": 6,
        "status": "OK",
        "limit": 100,
        "page": 1,
        "pages": 1,
        "time": 0.0028,
        "method": "get"
    },
    "events": {
        "1": {
            "rate_id": 1,
            "name": "Max 3 Days",
            "type": "SP",
            "start_date": 20161221,
            "end_date": 0,
            "span": 0,
            "price_src": "Yield",
            "dynamic_rate": 10,
            "dynamic_type": "P",
            "status": "A",
            "repeat_id": "W",
            "enabled": 1,
            "rule_set_id": 1,
            "details": [
                {
                    "level": 5,
                    "rate": 150.5
                }
            ]
        }
    }
}
```
3.2.7 item

Your Checkfront inventory is made up of ‘items’ that are added to bookings, and have complex availability controls and rules defining their stock and prices.

When no dates are passed in the API call, a full list of enabled items in the inventory is displayed, without specific availability and pricing.

However, if you pass in the relevant booking dates (and optionally parameters, discount codes, etc), the API will return a “rated” response that includes pricing and availability, as well as a “SLIP” that will be used to add the item to a session.

SLIP

The item SLIP is an encoded string returned when making a rated query to a specific item. The slip contains the information needed to make a booking. Don’t attempt to reverse engineer this, as the format changes. It must be optioned via a rated API call.
**Booking Parameters**

Booking parameters are defined globally in your system, and can be configured on a per item basis. Your parameters specify the number of items to book, for example Child and Adult. These are completely configurable in your Checkfront account under **Manage / Settings / Configure**.

To query specific pricing and availability, you need to pass the appropriate parameters in your API call (using the parameter IDs as specified in your configuration).

Say for instance you have 2 parameters configured: Adults (id: adults), and Children (id: children). To get pricing for 2 adults and one child you would pass: `param[adults]=2&param[children]=1` in your API call.

Booking parameters have many options, and can be configured to control inventory in very specific ways. See the Checkfront support centre or ask us for more information.

**Request**

**GET /api/3.0/item**

Retrieve a list of the enabled items in the system.

**Query Parameters**

- **start_date** *(date)* – (rated) Booking start date.
- **end_date** *(date)* – (rated) Booking end date.
- **date** *(date)* – (rated) Alias of **start date** (for same-day bookings).
- **start_time** *(date)* – (rated) Start time (used in hourly bookings).
- **end_time** *(date)* – (rated) End time (used in hourly bookings).
- **category_id** *(integer)* – Filter items by category.
- **packages** *(boolean)* – Show package options.
- **available** *(integer)* – (rated) Filter to items with at least this many left in stock.
- **keyword** *(string)* – Filter to items with a name containing this keyword.
- **item_id** *(string)* – A comma-separated list of items to filter to.
- **discount_code** *(string)* – (rated) The discount code to apply to the price.
- **rules** *(string)* – ‘soft’ will prevent triggering date based rule errors, or ‘off’ will disable rule checking.
- **param** *(array)* – (rated) See **Booking Parameters**.

**GET /api/3.0/item/{item_id}**

Retrieve details for a single item. Shares same params as above for rated requests.

**Parameters**

- **item_id** *(string)* – The unique item_id of the item to query, as found in a response or your Checkfront panel.

**GET /api/3.0/item/{item_id}/cal**

Retrieve calendar availability for a single item.

**Parameters**

- **item_id** *(string)* – The unique item_id of the item to query.

**Query Parameters**
Checkfront API Documentation, Release 3.0

- **start_date** *(date)* – Availability range start date.
- **end_date** *(date)* – Availability range start date.

**GET /api/3.0/item/cal**

Retrieve calendar availability for a set of items.

**Query Parameters**

- **item_id** *(array|integer)* – The unique item_id of the item to query, or an array of item IDs
- **category_id** *(integer)* – A category of items to filter to.
- **start_date** *(date)* – Availability range start date.
- **end_date** *(date)* – Availability range start date.

**Unrated Response**

```
{
  "version": "3.0",
  "account_id": 1,
  "host_id": "your-company.checkfront.com",
  "name": "Your Company",
  "locale": {
    "id": "en_US",
    "lang": "en",
    "currency": "CAD"
  },
  "request": {
    "status": "OK",
    "resource": "item",
    "id": "18",
    "records": 0,
    "limit": 0,
    "page": 1,
    "pages": 1,
    "time": 0.0027,
    "method": "get"
  },
  "query": [],
  "item": {
    "unit": "D",
    "item_id": 18,
    "sku": "bungeejumpingfromthetopoftheo2arena",
    "url": "",
    "lock": 1,
    "visibility": "*",
    "name": "Bungee Jumping from the Top of the O2 Arena",
    "pos": 0,
    "summary": "Take a trip to the top of London's most famous concert venue, the O2 Arena, and right back down again. This exhilarating chance of a lifetime starts with a tour through the Memorabilia Halls of the O2, and finishes with a 160 ft bungee jump. Days and space are limited, so book today!",
    "details": "",
    "meta": "{"display_mode":"dropdown","item_package_rate":"\\\","delay\\\":0}"
  },
  "stock": 15
}
```

(continues on next page)
"unlimited": 0,
"video": {
  "id": "B6wTLgMFM4",
  "start": 0
},
"image": {
  "1": {
    "title": 
    "src": "18-1--1523520394960863",
    "path": "/media/M18-1--1523520394960863.jpg",
    "url_small": "https://storage.googleapis.com/cf-public-us/thelondonexperience-67672/media/S18-1?t=1523520395895660"
  }
},
"category_id": 6,
"rated": 1,
"product_group_type": 
"product_group_children": 
"type": "I",
"status": "A",
"alias_id": 0,
"len": 1,
"rules": "\"param\":{\"adults\":{\"MIN\":\"0\",\"MAX\":\"0\"},\"children\":{\n\"MIN\":\"0\",\"MAX\":\"0\"}},\"fixed\":{\"start_time\":\"\"}}",
"category": "Activities"
}

Rated Response

{
  "version": "3.0",
  "account_id": 1,
  "host_id": "your-company.checkfront.com",
  "name": "Your Company",
  "locale": {
    "id": "en_US",
    "lang": "en",
    "currency": "CAD"
  },
  "request": {
    "status": "OK",
    "resource": "item",
    "id": "18",
    "records": 0,
    "limit": 0,
    "page": 1,
    "pages": 1,
    "time": 0.0101,
    "method": "get"
  }
}
"query": {
  "start_date": "20171005",
  "end_date": "20171005",
  "param": {
    "qty": "1",
    "rooms": "1",
    "adults": "1",
    "children": "0",
    "guests": "1",
    "familyticket2ad": "1",
    "familyticket": "4"
  },
  "local_end_date": "10/05/17",
  "local_start_date": "10/05/17"
},
"item": {
  "unit": "D",
  "item_id": 18,
  "sku": "bungeejumpingfromthetopoftheo2arena",
  "url": "",
  "lock": 1,
  "visibility": "x",
  "name": "Bungee Jumping from the Top of the O2 Arena",
  "pos": 0,
  "summary": "<p>Take a trip to the top of London's most famous concert venue, the O2 Arena, and right back down again. This exhilarating chance of a lifetime starts with a tour through the Memorabilia Halls of the O2, and finishes with a 160 ft bungee jump. Days and space are limited, so book today!</p>",
  "details": "",
  "meta": {
    "display_mode": "dropdown",
    "item_package_rate": "",
    "delay": 0,
    "default_len": 1
  },
  "stock": 15,
  "unlimited": 0,
  "video": {
    "id": "B6MwTLgmFMM",
    "start": 0
  },
  "image": {
    "1": {
      "title": "",
      "src": "18-1--3942",
      "path": "/media/M18-1--3942.jpg"
    },
    "2": {
      "title": "",
      "src": "18-2--9804",
      "path": "/media/M18-2--9804.jpg"
    }
  },
  "category_id": 6,
  "rated": 1,
  "type": "I",
  "visibility": "x"
}
"status": "A",
"alias_id": 0,
"len": 1,
"rules": "\\"\"param\":{"\"adults\":{\\"MIN\":\\"0\\"},\\"children\":{\\"MAX\":\\"0\\"}}\"",
"category": "Activities",
"package": [
  {
    "package_id": 21,
    "optin": "O",
    "select_package_params": 1
  },
  {
    "package_id": 20,
    "optin": "O",
    "select_package_params": 1
  },
  {
    "package_id": 19,
    "optin": "O",
    "select_package_params": 1
  }
],
"param": {
  "adults": {
    "price": 1,
    "lbl": "Adults",
    "lock": 1,
    "qty": 1,
    "hide": 0,
    "customer_hide": 0,
    "req": 1,
    "range": 1,
    "def": 1,
    "MIN": 0,
    "MAX": 0
  },
  "children": {
    "price": 1,
    "lbl": "Children",
    "lock": 1,
    "qty": 0,
    "hide": 0,
    "customer_hide": 0,
    "req": 0,
    "range": 1,
    "def": 0,
    "MIN": 0,
    "MAX": 0
  }
},
"qty": 1,
"rate": {
  "status": "AVAILABLE",
  "available": 9,
  "slip": "18.20171005X1-adults.1-children.0",
  "summary": {
    (continues on next page)
"title": "9 Available",
"details": "<strong title='Adults'>Adults:</strong> 1 Day @ $130.00",
"price": {
  "total": "$130.00",
  "title": "$110.00 - $130.00",
  "unit": "per day",
  "param": {
    "adults": "$130.00",
    "children": "$110.00"
  }
},
"date": "Thu Oct 5, 2017"
},
"sub_total": "130.00",
"event": [
  {
    "adults": {
      "amount": 130,
      "days": 1,
      "type": "q",
      "range": "1-5"
    },
    "adults-1-5": {
      "amount": 130,
      "type": "q"
    },
    "adults-6-15": {
      "amount": 115,
      "type": "q"
    },
    "children": {
      "amount": 110,
      "days": 1,
      "type": "q"
    },
    "children-1-5": {
      "amount": 110,
      "type": "q"
    },
    "children-6-15": {
      "amount": 100,
      "type": "q"
    }
  }
],
"start_date": "20171005",
"end_date": "20171005",
"dates": {
  "20171005": {
    "status": "A",
    "price": {
      "adults": 130,
      "children": 110
    },
    "dow": "4",
    "stock": {
      "T": 15,
"B": 6,
"A": 9,
"MinA": 9
},
"rates": [],
"discount": [],
"sub_total": 130
}
}
",
"days": 1,
"discount": {
  "amount": 0
},
"gprice": {
  "adults": {
    "1-5": "q",
    "6-15": "q"
  },
  "children": {
    "1-5": "q",
    "6-15": "q"
  }
},
"local_start_date": "10/05/17",
"local_end_date": "10/05/17"
},
"events": {
  "9": {
    "name": "Group Booking",
    "start_date": "",
    "end_date": ""
  }
}

### 3.2.8 help

**GET /api/3.0/help**

Retrieve a list of API endpoints.
3.2.9 ping

**GET /api/3.0/ping**

Show API connection details and latency.
"provider": "API Console"
```
HTTP Routing Table

/api
GET /api/3.0/*, 8
GET /api/3.0/account, 34
GET /api/3.0/booking, 35
GET /api/3.0/booking/form, 38
GET /api/3.0/booking/index, 47
GET /api/3.0/booking/session, 50
GET /api/3.0/booking/{booking_id}, 36
GET /api/3.0/booking/{booking_id}/invoice, 36
GET /api/3.0/category, 53
GET /api/3.0/company, 54
GET /api/3.0/customer, 56
GET /api/3.0/customer/{customer_id}, 56
GET /api/3.0/event, 57
GET /api/3.0/event/{id}, 57
GET /api/3.0/help, 67
GET /api/3.0/item, 61
GET /api/3.0/item/cal, 62
GET /api/3.0/item/{item_id}, 61
GET /api/3.0/item/{item_id}/cal, 61
GET /api/3.0/ping, 68
POST /api/3.0/booking/create, 38
POST /api/3.0/booking/notes, 48
POST /api/3.0/booking/session, 50
POST /api/3.0/booking/session/clear, 52
POST /api/3.0/booking/session/end, 53
POST /api/3.0/booking/{booking_id}/bookmark, 37
POST /api/3.0/booking/{booking_id}/checkin, 36
POST /api/3.0/booking/{booking_id}/checkout, 37
POST /api/3.0/booking/{booking_id}/note, 37
POST /api/3.0/booking/{booking_id}/update, 37