PARTNER DEVELOPER GUIDE

Integrations Best Practices

15 January 2020

This document may contain statements of future direction concerning possible functionality for Checkr's software products and technology. All functionality and software products will be available for license and shipment from Checkr only if and when generally commercially available. Checkr disclaims any express or implied commitment to deliver functionality or software unless or until actual shipment of the functionality or software occurs. The statements of possible future direction are for information purposes only, and Checkr makes no express or implied commitments or representations concerning the timing and content of any future functionality or releases.

This document is subject to change without notice, and Checkr does not warrant that the material contained in this document is error-free. If you find any problems with this document, please report them to Checkr in writing.

TABLE OF CONTENTS

1 INTRODUCTION

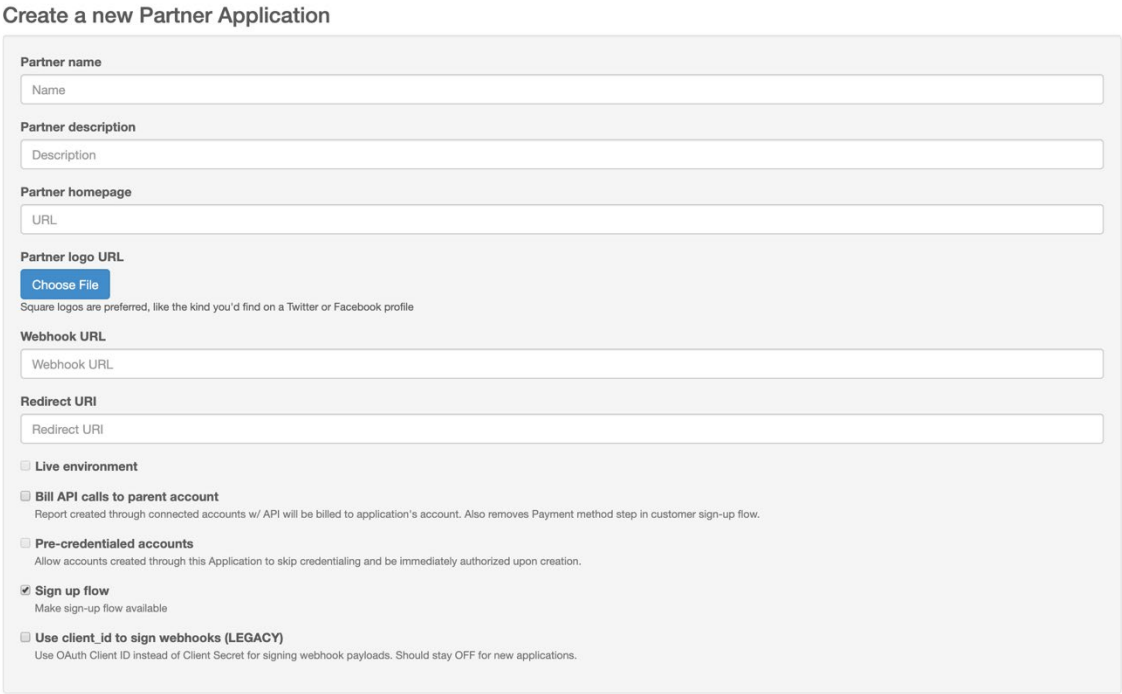
This guide describes using the Checkr API to build a background check workflow that works for your and your customers' needs.

If you are new to the regulatory aspects of background screenings, please review the Checkr Help Center. This site includes sections on Compliance, Adjudication and Review, and Checkr Screening Types. If you can't find an answer to your question, please contact your Partner Manager directly or reach out to our Customer Success team at clients@checkr.com.

2 GETTING STARTED

The first step to building a Checkr Partner integration is to set up your Checkr account with a "Partner Application". Partner Applications allow you to connect your customers' Checkr accounts to yours.

Your first Partner Application will be created in the Checkr Test environment and can be accessed from the Checkr Dashboard through **Account Settings > Application Settings**. If you do not see the **Application Settings** tab, ask your Partner Manager to enable this setting for you.



*Create a New Partner Application from the Checkr Dashboard at **Account Settings > Application Settings***

To create this Partner Application, you will be asked to provide the following information:

- **Partner name:** Your application's name or brand. This name will be displayed in the Connect to Checkr flow.
- **Partner description:** A short description of your application. This will be used in Checkr Marketplace listings.
- **Partner homepage:** Your application's URL. This will be used in Checkr Marketplace listings.
- **Partner logo URL:** A URL or a file of your application's logo or brand. This image will appear in the Connect to Checkr flow. Square logos are preferred.
- **Webhook URL:** An endpoint to which webhooks will be transmitted. This endpoint will receive all webhook events transmitted for your connected customer accounts.
- **Redirect URI:** A page in your application to which your customers will be redirected after connecting, or failing to connect, their Checkr account using the Connect to Checkr flow. This URL must be HTTPS. It is used to secure your customers' authentication and prevent Cross-Site Request Forgery (CSRF) attacks.
- **Live environment:** This setting determines whether your Partner Application generates Test or Live resources. This setting is disabled by default. You will not be able to create a Partner Application in the Live environment until your account has been credentialed. This cannot be changed once the Partner Application has been created.
- **Bill API calls to parent account:** Also known as "master billing", this setting determines whether your connected customer accounts' background check reports will be billed to your account as a consolidated invoice, or individually to each customer. If enabled, the "Payment Method" step will also be removed from the Connect to Checkr flow. Work with your Partner Manager to determine which billing workflow makes the most sense for your application. This setting is disabled by default.
- **Pre-credentialed accounts:** This setting determines whether your connected customer accounts must comply with the Checkr credentialing process (~1–2 business days) or will receive an automatic credentialing upon account connection. This setting is only available for partners with a strict Know Your Customer (KYC) process and can only be enabled by your Partner Manager.
- **Sign up flow:** This setting determines whether to make the Checkr-hosted Sign Up form available as part of the Connect to Checkr flow. If disabled, Checkr will bypass the Sign Up flow and immediately redirect your customers to Sign In. Disable this setting if you plan on self-hosting the Checkr Sign Up flow using the Account API.

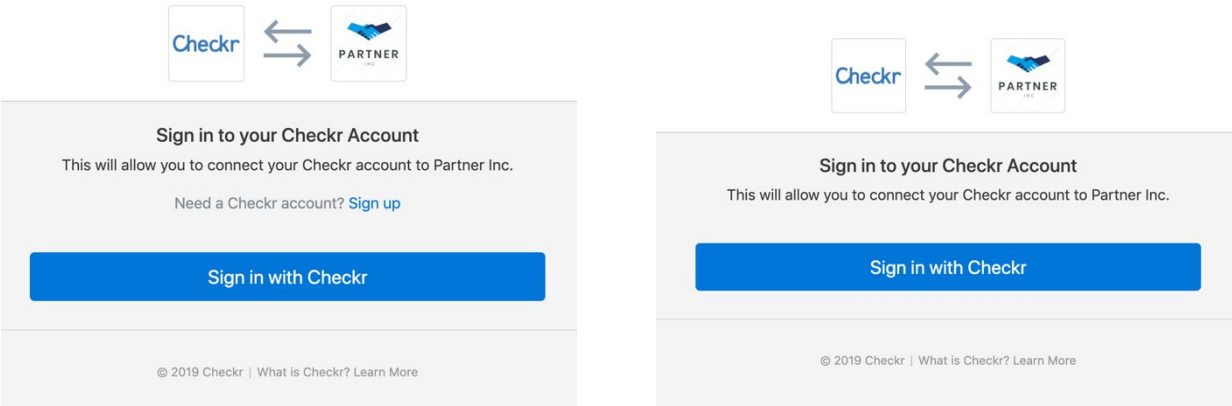
Once you've created your Partner Application, Checkr will generate a `client_id` and a `client_secret` to use as your application credentials. These credentials allow you to make API calls as a Partner. Keep them safe! (Particularly your `client_secret`: this is a secret key that should be stored securely in your application and not shared with anyone.)

3 CONNECT TO CHECKR

Your `client_id` is the unique identifier used to identify your Partner Application. Checkr uses this `client_id` to compose a unique link to embed in your application for your customers to use to either sign in to an existing Checkr account, or sign up for a new Checkr account. With an active Partner Application and the "Sign Up flow" setting enabled, these links take the format:

- `https://partners.checkr.com/authorize/{client_id}/signup`
- `https://partners.checkr.com/authorize/{client_id}/signin`

If the "Sign Up flow" setting is disabled, the `/signup` URL will always redirect to `/signin`, and the "Need a Checkr account? Sign up" option will be hidden from the Sign In page.



The Sign In page with the "Sign up flow" setting enabled vs. disabled

3.1 USING THE CHECKR-HOSTED SIGN UP FLOW

The Checkr-hosted Sign Up flow is the simplest way for customers to connect a new Checkr account and requires the least development effort on your part. Embed your unique link into your application and Checkr will collect the necessary user and company information to set up and credential the customer's Checkr account.

The "Connect to Checkr" call-to-action works well in an integration or vendor marketplace, within workflows where background checks are ordered, in a settings page, or within a combination of the three. Work with your Partner Manager to understand which works best for your use case.

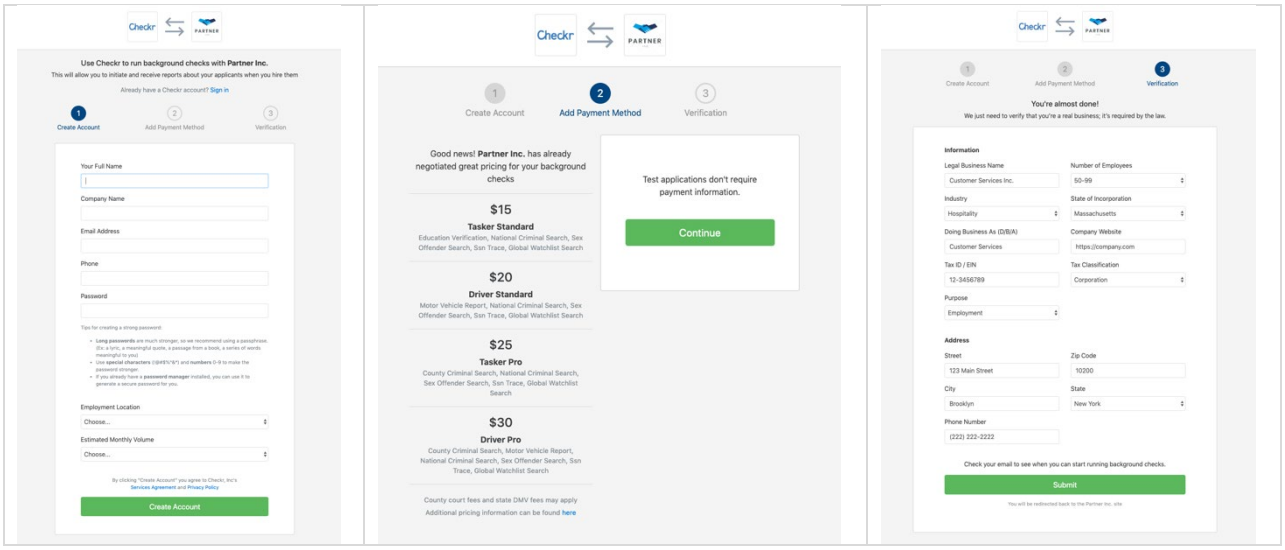
When embedding the “Connect to Checkr” call-to-action in your application, you may also elect to use the following URL parameters to pass additional information and secure your customers’ authentication.

- **`redirect_uri`** (optional): The URL to redirect your user to upon flow completion. This must match the [configured redirect_uri in your Partner Application] settings and must use the HTTPS protocol. This must be a static string. Wildcards (*) are not supported.
- **`state`** (optional): A string to be passed back as a URL parameter on `redirect_uri` upon flow completion. We recommend using state to pass through either the ID of the user completing the flow, or a tokenized string for which you can compute the hash.

For example, your "Connect to Checkr" call-to-action may look like:

```
https://partners.checkr.com/authorize/{client_id}?redirect_uri=https://par
tnerinc.com/checkr/callback&state=79a3ead9-2768-477f-8eca-724890dcf8d6
```

Following this link will direct a user to a Checkr-hosted Sign Up flow. This flow is 3 steps and requires the end user to supply information about themselves, their company, and the reason they are running background checks (also known as "permissible purpose"). If "master billing" is enabled, the user will not be required to supply a payment method; else, a valid payment method must be provided. Checkr accepts either credit/debit card or ACH information, which can be updated from the Checkr Dashboard at any time. Customers are only charged for the background checks they run.



Checkr-hosted Sign Up flow: [click here] to access a live demo

Once the end user has completed the “Connect to Checkr” flow and successfully connected an account, they will be redirected to your defined `redirect_uri` with both a `code` and `state` parameters (if provided).

When your users have completed the flow described in the previous example, they will be redirected to:

```
https://partnerinc.com/checkr/callback?code={JWT}&state=79a3ead9-2768-
477f-8eca-724890dcf8d6
```

At this point please check that the `state` string matches what you passed initially. If it doesn't match or doesn't exist, you should treat it as a failed connection, surface the error to the user and redirect them to the start of the flow to try the process again.

Next, use the `code` parameter to call the Tokens endpoint and retrieve an `access_token` for your authorized customer. This is a one-time process and the `access_token` grants your application the right to make API calls to Checkr on behalf of the customer account.

Retrieve a customer's access token:

```
$ curl -X POST https://api.checkr.com/oauth/tokens \
   -d client_id={client_id} \
   -d client_secret={client_secret} \
   -d code={JWT}
```

Example response:

```
{
   "access_token": "{access_token}", // customer's access token
   "scope": "read_write",
   "checkr_account_id": "5d78dfa52ea938723b2f2ba3" //customer's account ID
}
```

> **Note:** The authorization code that is passed as a parameter on the `redirect_uri` is specifically used to retrieve an access token for the authorized customer. It can be used only once and expires 5 minutes after creation.

Access tokens are long-lived, account-level API keys that are not tied to specific user access. They are valid until revoked, so treat them with care. We recommend storing them encrypted in your application's data store along with the customer account ID (`checkr_account_id`) returned in the response payload.

3.2 USING THE CREATE ACCOUNT API

If you collect the information listed above on your customers' behalf, use the Account endpoint to provide a more integrated, seamless experience. The Account endpoint can be used to create customer accounts without requiring the customer to complete a Checkr-hosted Sign Up flow.

The `oauth_authorize` parameter gives you the flexibility to determine how your customer authorizes your application.

- For explicit authorization, set `oauth_authorize` to `false` or simply don't pass the flag; this will return the created account object in the response. From here, redirect your end user to the Sign In flow to sign in and explicitly grant authorization to your application. The authorization code will be returned as a URL parameter on your defined `redirect_uri`.
- For implicit authorization, set `oauth_authorize` to `true`; the authorization code will be returned in the body of the response.

Create a customer account that requires end-user authorization:

```
$ curl -X POST https://api.checkr.com/v1/accounts -u {API_KEY}: \
   -d client_id={client_id} \
      -d oauth_authorize=false \
   -d See request body ↗
```

Example response:

```
{
    "id": "dwe2u29j7gg47p8ed7wa",
    "object": "account",
    "name": "Customer Services Inc.",
    "default_compliance_state": "CA",
    "authorized": false,  // false means account is not yet credentialed
    "purpose": "employment",
    "user": {
        "email": "user@email.com",
        "full_name": "Jane Doe"
    },
    "company": {
        "industry": "72",
        "incorporation_state": "MA",
        "dba_name": "Customer Services",
        "website": "https://company.com",
        "tax_id": "123456789",
        "incorporation_type": "corporation",
        "street": "123 Main Street",
        "zipcode": "10200",
        "city": "Brooklyn",
        "state": "NY"
        "phone": "222-222-2222"
    }
    ...
}
```

Authorize a customer account that implies end-user authorization:

```
$ curl -X POST https://api.checkr.com/v1/accounts -u {API_KEY}: \
   -d client_id={client_id} \
      -d oauth_authorize=true \
   -d See request body ↗
```

Example response:

```
{
    "user": {
        "code": "{JWT}"
    }
}
```

> **Note:** For the Accounts endpoint only, authenticate using an API key generated using **Account Settings > Developer Settings** from within the Checkr Dashboard. Logs generated from this API call can be accessed from the **Logs** tab in your Partner Checkr account.

Regardless of the method used to retrieve the authorization `code`, you'll then use the Tokens endpoint to retrieve the customer's access token.

3.3 CUSTOMER ACCOUNT CREDENTIALING

New accounts must be credentialed by Checkr's Customer Success team before they will be allowed to request background checks. We use the information provided by the customer to assess the validity of the business and its permissible purpose. This process generally takes less than 1 business day.

Once the credentialing process is complete, Checkr will issue an `account.credentialed` webhook to the `webhook_url` configured during Partner Application setup. We will also notify your customer by email (the technical contact, if present, otherwise the first admin user).

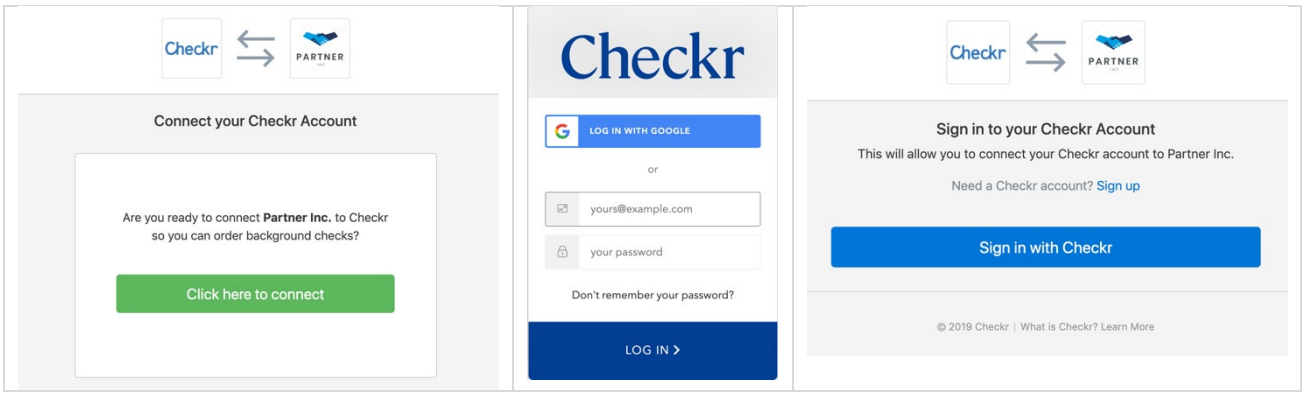
If your customer's Checkr account is already credentialed, the `account.credentialed` webhook will be issued immediately after the Connect to Checkr flow is completed.

If your Partner Application has the "Pre-credentialed accounts" setting enabled, customer accounts created using your Connect to Checkr flow will receive an automatic credentialing upon account connection. This setting is available only for partners with a strict KYC process, and can be enabled only by your Partner Manager.

To check the account credentialing status for a customer account, use their `access_token` to call `GET /v1/account`. The account is credentialed if the `authorize` parameter is set to `true`.

3.4 FOR EXISTING CHECKR CUSTOMERS

The Sign In flow prompts users to sign into their Checkr account to authorize the connection. Only Checkr users with an Admin role can perform this action.



Checkr-hosted Sign In and authorization flow

3.5 DISPLAYING THE CONNECTED STATE AND DEAUTHORIZATION

After a customer has connected their Checkr account once, there is no need to perform the action again unless or until the customer's access token is deauthorized. We recommend storing and displaying this connected state to prevent your end users from attempting to create more than one Checkr account or connect more than once.

You may also elect to provide your end users the ability to disconnect their Checkr account from your application. Use the `Deauthorize` endpoint to deprecate a customer's access token. Customers may also deauthorize your application from the Checkr Dashboard. Listen for the `token.deauthorized` webhook for notification of these events.

Deauthorize a customer's access token:

```
$ curl -X POST https://api.checkr.com/oauth/deauthorize -u {access_token}:
```

Example response:

```
{
    "access_token": "{access_token}"
}
```

Once an `access_token` is deauthorized and the customer account is disconnected from your application, we recommend reflecting this state in your application so that customers can attempt to connect again.

4 SELECTING PACKAGES

Once your customer's account is connected and has been credentialed, they may begin to order background checks from your application. The first step to ordering a background check is to select which background check package to run. A package is a collection of screenings, with screenings being different types of checks like a criminal check, motor vehicle record, credit report, etc. For additional information on which screenings comprise basic package types, refer to the Checkr Help Center.

Which package to order for a candidate can be determined by candidate (select a package for a candidate when ordering the background check), or by job position (select a package for a position, to be applied to all candidates placed against that position). Your use case, and the volume of background checks that your end users may run, will help determine which of these two options to consider.

Work with your Partner Manager to define the set of background check packages and their pricing for your Partner account. Your connected customer accounts will inherit these packages and prices.

4.1 RETRIEVING A CUSTOMER'S PACKAGE LIST

In some cases a connected customer account will have additional packages that differ from those defined at the Partner account level. They may be accounts that already exist and are connected through the Sign In flow, or your customers may contact Checkr to add additional screening types required for their business. Because of this, we recommend using the customers' `access_token` to retrieve the package list that will populate your package selection interface, instead of relying on your Partner account package list.

The response is paginated and contains 25 objects at a time. If the account contains more than 25 packages, you will need to iterate through the paginated list or specify the **per_page** limit as described in the Pagination section of the API documentation.

Retrieve a customer's package list:

```
$ curl -X GET https://api.checkr.com/v1/packages -u {access_token}:
```

Example response:

```
{
  "data": [
    {
      "id": "c6759e59e807618f8bcbd37a",
      "object": "package",
      "price": 2500,
      "apply_url": "https://apply.checkr.com/apply/customer-services-
inc/532c20ea819b",
      "created_at": "2019-08-07T22:17:50Z",
      "deleted_at": null,
      "name": "Tasker Standard",
      "screenings": [
        {
          "type": "county_criminal_search",
          "subtype": "current"
        },
        {
          "type": "national_criminal_search",
          "subtype": "standard"
        },
        {
          "type": "sex_offender_search",
          "subtype": null
        },
        {
          "type": "ssn_trace",
          "subtype": null
        },
        {
          "type": "global_watchlist_search",
          "subtype": null
        }
      ],
      "slug": "tasker standard",   // used for subsequent API calls
      "uri": "/v1/packages/c6759e59e807618f8bcbd37a"
    }
  ],
  "object": "list",
  "next_href": null,
  "previous_href": null,
  "count": 1
}
```

You may also choose to cache the package list and listen for each customers' `package.*` webhook events for updates. Checkr will transmit a webhook event for `package.created`, `package.updated`, and `package.deleted`. See the Webhooks section for more information on consuming Checkr webhooks.

5 REQUESTING BACKGROUND CHECKS

There are two methods of requesting a background check using the Checkr API: the invitation flow and the self-hosted reports flow. The invitation flow requires the least amount of development effort and all Checkr screening types are supported. A self-hosted reports flow may be desirable if you would like to have more control over the end-to-end process, though it requires that you take on some compliance burden, such as surfacing the right compliance forms and collecting consent from the candidate. For more on the benefits and disadvantages of each approach, take a look at Designing your workflow.

5.1 CREATING OR RE-USING CANDIDATE OBJECTS

Before creating an invitation or a report, you must retrieve the ID of the Candidate for which you want to order the background check. You can do so by retrieving an existing Candidate, or creating a new one if it doesn't exist. We recommend re-using Checkr Candidate objects instead of creating a new one for each report, as it consolidates the Candidate experience and makes the support process much simpler. The most surefire way to retrieve an existing Checkr Candidate is with its unique resource ID ("id"). Store this ID against the representation of the Candidate in your application, and use this value for all Invitation or Report orders for that Candidate.

Create a new Candidate:

```
$ curl -X POST https://api.checkr.com/v1/candidates -u {access_token}:
      -d email=candidate@email.com
```

Example response:

```
{
    "id": "e44aa283528e6fde7d542194",
    "object": "candidate",
    "email": "candidate@email.com",
    ...
}
```

> **Note:** Use the returned `id` (Checkr Candidate ID) for all Invitation or Report orders for that Candidate.

Retrieve an existing Candidate by ID:

```
$ curl -X GET https://api.checkr.com/v1/candidates/{id} -u
{access_token}:
```

Example response:

```
{
    "id": "e44aa283528e6fde7d542194",
    "object": "candidate",
    "email": "candidate@email.com",
    ...
}
```

If you do not know or have the Checkr Candidate ID, you can also use query parameters to retrieve a Candidate object by other identifiers. Typically we see this work well with the query parameter's `email` (if you have this data) and/or `custom_id` (a string that you can use to store your application's identifier against the Checkr candidate resource). For the full list of possible query parameters, see the [List existing Candidates] method.

Retrieve an existing Candidate by query parameter:

```
$ curl -X GET
https://api.checkr.com/v1/candidates?email=candidate@email.com -u
{access_token}:
```

Example response:

```
{
  "data": [
    {
      "id": "e44aa283528e6fde7d542194",
      "object": "candidate",
      "email": "candidate@email.com",
      ...
    }
  ]
  "object": "list",
  "next_href": null,
  "previous_href": null,
  "count": 1
}
```

Note: The returned object is a paginated list, as the call is not for a specific object but for a list of objects. See Retrieve existing Candidate to retrieve a Candidate object by its ID.

5.2 USING THE INVITATION FLOW (RECOMMENDED)

The easiest method to integrating background checks into your application is with the invitation flow. In this flow, you use the Invitation resource to order the background check. Checkr sends an invitation email to the candidate to provide their information and consent, and once the invitation is completed a Report is automatically created. The invitation is valid for 7 days, in which Checkr will send a follow-up notice to the candidate to complete the invitation every 24 hours. If 7 days pass and the candidate has not completed the invitation, the invitation will expire and you will need to create a new invitation.

Use the Candidate ID you have retrieved or created via the previous method (see Creating or re-using Candidate objects), the Package "slug" (as selected in step Selecting Packages), and the Candidate's work location to create an Invitation.

Checkr uses the candidate work location to apply the appropriate state- and city-based fair hiring laws, disclosures, and adverse action procedures. If a city is not provided, Checkr utilizes the state-based regulation.

Create an Invitation:

```
$ curl -X POST https://api.checkr.com/v1/invitations -u {access_token}:
      -d candidate_id=e44aa283528e6fde7d542194 \
      -d package=tasker_standard \
      -d work_locations[][state]=CA \   // state required, city optional
      -d work_locations[][city]=San+Francisco
```

Example response:

```
{
    "id": "551564b7865af96a28b13f36",
    "object": "invitation",
    "uri": "/v1/invitations/551564b7865af96a28b13f36",
    "invitation_url":
"https://apply.checkr.com/invite/try-checkr/290f9d6d6e46/test",
    "status": "pending",
    "created_at": "2015-05-14T17:45:34Z",
    "expires_at": "2015-05-21T17:45:34Z",
    "completed_at": null,
    "deleted_at": null,
    "package": "tasker_standard",
    "candidate_id": "e44aa283528e6fde7d542194",
    "report_id": null
}
```

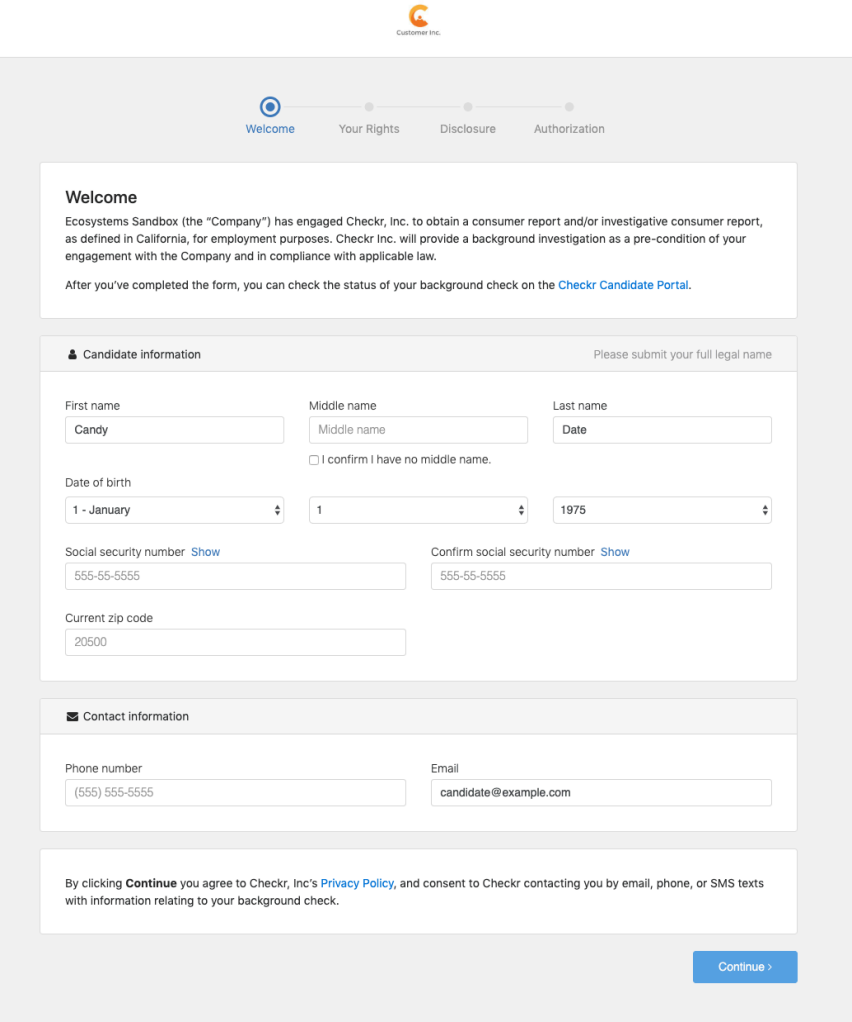
Note: Checkr only requires the Candidate to provide information that is required for the screenings contained in the package. (For example: SSN for criminal screenings, driver's license number and state for MVR) If you already collect some of this information in your application, you can choose to pre-fill these fields in the invitation by creating or updating the Candidate object with this data prior to creating the Invitation.

Update the Candidate object with known information:

```
$ curl -X POST https://api.checkr.com/v1/candidates/{id} -u
{access_token}:
      -d first_name=Candy \
      -d last_name=Date \
      -d dob=1975-01-01
```

Example response:

```
{
    "id": "e44aa283528e6fde7d542194",
    "object": "candidate",
    "email": "candidate@email.com"
    "first_name": "Candy",
    "last_name": "Date",
    "dob": "1975-01-01",
    ...
}
```



Checkr-hosted Invitation flow

> **Note:** When using a Test API key or test `access_token` generated via a Test Partner Application, Checkr will not send an email to the test candidate email address. To access the invitation flow, retrieve the `invitation_url` from the Create Invitation response.

When the Candidate completes the invitation, the Invitation status is updated to **Complete** and the `report_id` value is updated with the created Report resource ID.

There is also the option to suppress the Checkr invitation email and reminders and still leverage all the benefits of the Checkr-hosted Invitation flow. Reach out to your Partner Manager for more information about this feature.

5.3 CREATING A SELF-HOSTED REPORTS FLOW

A more advanced method to integrating background checks into your application is by building a self-hosted Reports flow. In this flow, you are responsible for collecting the candidate information and consent in your own application, and you'll use the Report resource to order the background check. Once all required information is present on the Candidate resource, creating a Report will kick off the background check immediately.

As an end user ordering consumer reports, you have certain responsibilities under the Fair Credit Reporting Act (FCRA). As your partner in background check screening, Checkr helps facilitate your compliance with the FCRA in a few ways. Building a self-hosted Reports flow requires that you take on these obligations on behalf of your customers, including providing Candidates the appropriate state- and city-specific disclosures for each screening type. For more information on your obligations under FCRA, and Checkr's responsibilities as a Consumer Reporting Agency (CRA), check out our helpful Compliance resources in the Help Center, particularly our article about obligations under FCRA and disclosures and authorizations.

There are some screening types that are not supported with the Reports flow, such as credit checks and health checks, and others that require significant data entry like employment and education verifications. If you are interested in building a self-hosted Reports flow, chat with your Partner Manager to understand the required disclosures and authorizations and PII that you must collect from the Candidate before creating a report.

Your Partner Manager will need to review your workflow before you will be approved to use the Reports API in production.

6 DISPLAYING RESULTS

Once a report is created, it can take anywhere from a few hours to a few weeks for a background check to be completed. The average time to completion is 2–3 days, depending on the screenings included in the report. Once a background check report is completed, Checkr will update the report with a status of `clear`, `consider`, or `complete`. We recommend taking a look at the description of all report statuses and their meanings to understand the types of statuses you may receive and how they are displayed in the Checkr Dashboard.

Between the time of report creation and report completion, some events can occur like exceptions, whereby Checkr will request more information from the candidate to validate the accuracy of the information they've provided. If the candidate does not respond within 7 days, the report is placed in a `suspended` status. Suspended reports can be updated and processed if the required document is provided within 30 days of the report's creation. If 30 days passes and the candidate has not provided the requested information, the report will remain in the `suspended` status and you will need to create a new report.

Once a report has been completed, you may choose to display report information within your application. While you may retrieve all the report information via the API, displaying report information in your application is another area governed by FCRA where Checkr can help facilitate compliance. As a partner, the more you change or modify the report information, the more risk you take on in becoming a CRA yourself. Here is guidance for displaying report information, though we recommend you work with your Legal team to determine how much consumer information you want to report from within your platform:

- On the safest end of the spectrum, a partner displays nothing but a direct link to the report in the Checkr Dashboard. There, you have only provided access to the information. Great!
- Further, a partner displays the status of the report, ETA, etc. in addition to a direct link to the Checkr Dashboard. There you have provided slightly more—though not consumer report information—audit information about the completion of the report. Good.
- Further, a partner displays a PDF render of the report in your own application. This is slightly more risky because you are providing access to the report information itself: information bearing on the consumer as opposed to information on the status or timing of the report. As long as you do not modify the information, and present it as rendered by Checkr, you have a strong argument that you are still only providing access. Not bad.

Retrieve the report status and link:

```
$ curl -X GET https://api.checkr.com/v1/reports/{id} -u {access_token}:
```

Example response:

```
{
      "id": "4722c07dd9a10c3985ae432a",
      "object": "report",
      "uri": "/v1/reports/4722c07dd9a10c3985ae432a",
      "status": "clear",
      ...
}
```

To embed a direct link to the report in the Checkr Dashboard, trim "/v1/" from the `uri` and use the following in your application:

- If `object` is `test_report`, URL is https://dashboard.checkr.com/reports/{id}?test=true
- If `object` is `report`, URL is https://dashboard.checkr.com/reports/{id}

If you would like to provide a PDF copy of the report to your customer, we recommend using the document `url` in the context of a **Download PDF** button. To retrieve a report PDF in a single call, take advantage of the Embedded Resource feature: use the **include** parameter to expand the **documents** object. The object type you're looking for is `pdf_report`.

Retrieve the report PDF:

```
$ curl -X GET https://api.checkr.com/v1/reports/{id}?include=documents -u
{access_token}:
```

Example response:

```
{
      "id": "4722c07dd9a10c3985ae432a",
      "object": "report",
      "uri": "/v1/reports/4722c07dd9a10c3985ae432a",
      "status": "clear",
      "documents": [
            {
              "id": "e44aa283528e6fde7d542194",
              "object": "document",
              "type": "pdf_report",  // this type refers to the report PDF
              "created_at": "2015-02-11T20:01:50Z",
              "download_uri": "{s3_download_uri}",
              ...
            }
      ]
}
```

Note: The document PDFs are hosted in an S3 bucket that has no expiration (lifecycle) policy, so they are available forever unless a Candidate requests to have their data removed from the Checkr system. The `download_uri` links expire after an hour. You can retrieve a refreshed `url` from the Retrieve existing Report endpoint at any time.

7 ADJUDICATION

If any information contained on a candidate's background check precludes them from working with your customer, it is your customer's obligation to carry out the Adverse Action process. Although responsibility and liability for the Adverse Action process ultimately lies with your end user, Checkr helps make it easier to maintain a compliant Adverse Action process. We recommend directing your customers directly to the report in the Checkr Dashboard to adjudicate a report (Engage or Adverse Action).

Any updates to the report as a result of the customer's adjudication process can be received by your application in the form of webhooks. See the Webhooks section for more information on event types and typical workflows in which you might expect to receive them.

8 WEBHOOKS

Checkr uses webhooks to communicate asynchronous changes on objects created with the API. Each time an event that you subscribed to occurs, Checkr submits a POST request to the webhook URL designated in your Partner Application with information about the event. For webhooks configured through Partner Applications, the `include_object` is enabled by default, which means that the object referenced in the event will be returned as part of the payload.

8.1 SUPPORTED WEBHOOK URLS

Checkr supports the use of HTTPS as well as AWS Simple Notification System (SNS).

HTTPS

The endpoint must be public, and Live environment webhooks must be HTTPS. While we do accept HTTP in the Test environment, as a general rule we recommend using the HTTPS protocol. In addition, while it's not required, we do support the Basic Auth method of authentication by adding **username:password@** in front of the hostname. These credentials must be URL escaped.

```
https://{user}:{password}@{hostname}/{path}
```

For example:

```
https://dw69ds8zg7yt:tmdghtwer999p2q3@partnerinc.com/webhooks/checkr
```

SNS

We also support webhook transmittal using Amazon SNS. To use SNS, your Access Key must only have the "Publish to SNS" IAM permission policy configured.

```
sns://{key_id}:{access_key}@{region}/{topic_owner}/{topic_name}
```

For example:

```
sns://AKI95AMUAD5K:a2n66fVKX7%2BYJKid3@us-east-1/12048/checkr
```

8.2 RESPONDING TO AND SECURING WEBHOOKS

Your endpoint should respond to Checkr webhooks as quickly as possible. To acknowledge receipt of a webhook, your endpoint must return a 2xx HTTP status code. This status code should only indicate receipt of the message, not acknowledgment that it was successfully processed by your system. Any other information returned in the response headers or response body is ignored.

If a webhook is not successfully received for any reason, Checkr will continue trying to send it every minute for 10 minutes, then every hour for 24 hours. Webhooks failing for more than 7 consecutive days are automatically deleted.

We pass along a hash signature with each request in a header X-Checkr-Signature. The hash signature is generated with the HMAC algorithm, using your `client_secret` as a key and a SHA256 digest. When you receive a request, you can compute a hash and ensure that the one from Checkr matches.

Example hash signature computation:

```
echo -n "${request_body}" | openssl dgst -sha256 -hmac "${client_secret}"
```

> **Note:** The key used to compute the hash is your `client_secret`, not an account-level API key or a customer's `access_token`.
> Any webhook event transmitted for an object requested using a customer's `access_token` will contain a signature in its header that can be verified using your `client_secret`.

8.3 TYPICAL EVENT FLOWS

Your Partner Application is subscribed to all webhook events by default, which include notifications for the resources Account, Candidate, Invitation, Verification, Report, Adverse Action, and Package. While the webhook event type is generally descriptive of how the object has been updated, we recommend consuming the object payload instead of relying on the event type itself. Each event describes the creation or update of its contained object, so it's good practice to consume that payload as if you had made a call to retrieve the resource yourself.

While webhooks are helpful for updates, they are not foolproof. In some cases, report updates can be sent in rapid succession based on multiple events within the Checkr environment, and may be "mis-heard". There are some additional recommendations for guarding against duplicate and missed notifications in the API documentation.

The following table provides the most common sequence for the most common webhook events. This list is not exhaustive and does not describe all sequences. In any given cycle, some events may not occur, and others may occur in an order different than that listed here.

Most webhook events proceed in the following order.

Event	Description
`candidate.created`	A new Candidate has been created.
`invitation.created`	An Invitation has been created.
`invitation.completed`	An Invitation has been completed.
`report.created`	A new Report has been created. Status: **pending**
`candidate.driver_license_required`	An exception has been raised requiring a copy of the Candidate's driver license.
`verification.created`	A verification has been created and a request to upload a document or enter data has been forwarded to the candidate.
`report.suspended`	A Report has been suspended. Checkr is waiting for the candidate to provide additional documentation. Status: **suspended**
`verification.completed`	A document has been uploaded or data has been entered by the candidate.
`verification.processed`	The data gathered by the verification has been processed manually or automatically and the background check can proceed.
`candidate.updated`	A Candidate has been updated.
`report.resumed`	A Report has resumed. (The candidate has provided documentation to a previously "suspended" Report.) Status: **pending**
`report.completed`	A Report has been completed. Status: **clear**, **consider**

`report.engaged`	A Report has been adjudicated as "engaged". Use this event to track either all candidates you have officially engaged, or simply those candidates with a "consider" background check report that you have engaged. This can be triggered either from an API call or from the dashboard ("Engage" button). Status: **pending**, **clear**, **consider**, **suspended**
`candidate.engaged`	A Candidate has been marked "Engaged".
or	
`report.pre_adverse_action`	The Pre-Adverse Action notice has been sent to the candidate of that report. Status: **consider**
`candidate.pre_adverse_action`	An Adverse Action has been initiated on the Candidate.
`report.post_adverse_action`	The Post-Adverse Action notice has been sent to the candidate of that report. Status: **consider**
`candidate.post_adverse_action`	An Adverse Action has been completed on the Candidate.
or	
`report.disputed`	A Report has been disputed by a candidate. Once a dispute is completed, Checkr will trigger the `report.completed` webhook again with the appropriate Report status. Status: **dispute**

8.4 ACCESSING WEBHOOK LOGS

All webhook events transmitted for objects requested using a customer's `access_token` will be stored within the customer's Checkr account webhook logs. Checkr does not yet grant partners direct access to customer webhook logs. If you have a question about a particular event or series of events, work with your Partner Manager to help you troubleshoot.

9 ADVANCED FEATURES

9.1 ETA

Report ETAs predict when County Criminal Checks will complete for each background check report. This ETA provides a date for the estimated completion of a specific report, helping both you and candidates plan ahead. The process of searching at the county level varies from county to county. Some county searches are returned the same day, while some take several days or more, depending on the search methodology. While our predictions are highly accurate, they are not a guarantee. Estimates provided by Report ETA are correct within one business day for more than 9 out of 10 requested reports.

Retrieve a Report ETA:

```
curl -X GET https://api.checkr.com/v1/reports/{id}/eta -u {access_token}:
```

> **Note:** Report ETA is calculated only for packages that include a County Criminal Check. If the package does not include a Country Criminal Check, the endpoint will respond with a 404.

9.2 SCREENING-LEVEL STATUSES

In addition to providing a high-level report status (Pending, Clear, or Consider), you may also wish to expose the status of individual screenings within the package. The most straightforward way to do this is to use the Embedded Resource feature. Use the `include` parameter to expand the screening objects in the Report resource in order to fetch the individual statuses.

Retrieve screening statuses:

```
$ curl -X GET
https://api.checkr.com/v1/reports/{id}?include=ssn_trace,county_criminal_s
earches,county_civil_searches,municipal_criminal_searches,drug_screening,e
ducation_verification,employment_verification,eviction_search,federal_civi
l_search,federal_criminal_search,global_watchlist_search,international_cri
minal_searches,national_criminal_search,personal_reference_verifications,p
rofessional_reference_verifications,sex_offender_search,state_criminal_sea
rches,pointer_state_criminal_searches,terrorist_watchlist_search,credit_re
port,facis_search,photo_verification,arrest_search,motor_vehicle_report,id
entity_document_verification -u {access_token}:
```

Example response:

```
{
      "id": "4722c07dd9a10c3985ae432a",
      "object": "report",
      "uri": "/v1/reports/4722c07dd9a10c3985ae432a",
      "status": "pending",  // overall report status
      "ssn_trace": {
          "id": "e44aa283528e6fde7d542194",
          "object": "ssn_trace",
          "uri": "/v1/ssn_traces/539fd88c101897f7cd000001",
          "status": "clear",  // ssn trace status
          ...
      },
      "county_criminal_searches": [
          {
            "id": "58845a3ea0fcd97136763136",
            "object": "county_criminal_search",
            "uri":
"/v1/county_criminal_searches/58845a3ea0fcd97136763136",
            "status": "clear",  // county criminal search status
            ...
          },
          {
            "id": "58845a3ea0fcd97136763137",
            "object": "county_criminal_search",
            "uri":
"/v1/county_criminal_searches/58845a3ea0fcd97136763137",
            "status": "pending",  // county criminal search status
            ...
          }
      ]
      ...
}
```