OpeningsJson

`POST /JR4.0/OPENINGS/OPENINGSJSON`

Returns available classes for a given organization that can be **registered** for or **waitlisted**.

POST JSON

`https://app.jackrabbitclass.com/jr4.0/Openings/OpeningsJson`

AUTH REQUIRED
No
Valid `OrgId` required

REQUEST FORMAT
JSON or form fields

RESPONSE FORMAT
application/json

RECORD LIMITS
None

Parameters

All optional unless marked as required

CORS Note: Only "simple" CORS requests are supported. Pre-flight (OPTIONS) requests are not honored. See examples below for usage.

Name	Type	Required	Description	Example
`OrgId`	int	Required	The organization ID requesting classes (usually a 5–6 digit number).	`507172`
`ShowClosed`	bit	Optional	If `1`, include classes where `openings.calculated_openings = 0`. Default: `0`.	`1`
`Exact`	bit	Optional	Controls how `Cat1`, `Cat2`, and `Cat3` are matched: • `0` – `Contains` (default) • `1` – `Equals` All other filters always use `Contains`.	`1`
`ShowLocName`	bit	Optional	Deprecated Previously controlled whether location code or name was returned. The response now always includes `location_code` and `location_name`. This parameter will be phased out.	`1`
`Cat1`	string	Optional	Filter by **Category 1**. Matching behavior is controlled by `Exact`.	`Dancing`
`Cat2`	string	Optional	Filter by **Category 2**. Matching behavior is controlled by `Exact`.	`Sprinting`

Name	Type	Required	Description	Example
`Cat3`	string	Optional	Filter by **Category 3**. Matching behavior is controlled by `Exact`.	`Yoga`
`InstructorId`	int	Optional	Filter by **primary instructor** only (Instructor 1). Uses the Instructor ID and does not apply to instructors 2–4.	`1087683`
`ClassDays`	string	Optional	Comma-separated list of days. Returns classes that meet **only** on the specified days. Accepted values: `Mon` `Tue` `Wed` `Thu` `Fri` `Sat` `Sun`	`Mon,Wed,Fri`
`Session`	string	Optional	Filter by session name.	`Summer 2015`
`Loc`	string	Optional	Filter by **Location Code** (not the full location name).	`North`
`Gender`	string	Optional	Filter by gender. Accepted values: `Male`, `Female`, `Both`.	`Both`
`Room`	string	Optional	Filter by room name or identifier.	`Studio A`
`Status`	string	Optional	Filter by class status.	`Active`
`Sort`	string	Optional	SQL-style sort expression using response property names. Supports complex names such as `Meeting_Days.Mon`. See Sorting and Output sections.	`Name DESC, openings.calculated_openings`
`InitEmpty`	bit	Optional	**Registration embed only.** If `1`, the registration "class search" page initializes with **no results**. Default: `0`.	`1`
`HdrColor`	string	Optional	**Registration embed only.** Hex color for the "class search" header.	`#fff`
`Hc`	string	Optional	**Registration embed only.** Comma-separated list of column indexes to hide on the "class search" page.	`1,5`

> **Location fields & deprecation:**
> The response always includes `location_code` and `location_name`. The legacy `location` field and `ShowLocName` parameter are deprecated and will be phased out. Use `location_code` and `location_name` going forward.

Example Requests

Server-side (cURL, JSON body)

```
curl -H "Content-Type: application/json" \
     -X POST \
     -d '{"orgid":"9209","session":"2017-2018"}' \
     https://app.jackrabbitclass.com/jr4.0/Openings/OpeningsJson
```

Client-side (CORS-compliant)

> **Important:** Only “simple” CORS requests are supported. Pre-flight (OPTIONS) requests will not be honored. Use basic POST with standard headers as shown below.

Native XHR

```
// Setup XHR
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://app.jackrabbitclass.com/jr4.0/Openings/OpeningsJson');

xhr.onload = function () {
    if (xhr.status === 200) {
        var data = JSON.parse(xhr.responseText);
        console.log(data.rows);
    }
};

// Build form data
var formData = new FormData();
formData.append('orgid', '12345');

// Send request
xhr.send(formData);
```

jQuery

```
$.post(
    'https://app.jackrabbitclass.com/jr4.0/Openings/OpeningsJson',
    { orgid: '12456' }
).done(function (resp) {
    console.log(resp.rows);
});
```

Sorting

Use the `Sort` parameter to control the order of returned classes. Property names must match the properties in the JSON response (see Output for a full list).

SYNTAX

```
PropertyName [ASC|DESC]
```

- `ASC` / `DESC` are optional; default is `ASC`.
- Separate multiple fields with commas.
- Complex properties use dot-notation, e.g. `Meeting_Days.Mon`.

EXAMPLES

```
// Sort by Name ascending
Name ASC

// Sort by End_Date descending, then Name ascending
End_Date DESC, Name ASC

// Sort by classes meeting Monday (complex property) descending, then Name ascending
Meeting_Days.Mon DESC, Name ASC
```

> If you need more advanced logic than simple property ordering (for example, conditional sorting based on data values), implement that in the consuming application after retrieving the results.

Output / Response Structure

The endpoint returns a JSON object with high-level metadata and a list of class records in the `rows` array.

Top-level Properties

Property	Type	Description
`rows`	array<object>	Collection of class records matching the request filters. Each element is a class object (see "Class Object Fields" below).
`success`	bool	Indicates whether the request completed successfully. `true` if results were returned; `false` otherwise.
`message`	string	Human-readable status or message, e.g. `"Class listings returned successfully!"`.

Class Object Fields (`rows[]`)

Property	Type	Description
`id`	int	Unique identifier for the class.
`category1`	string	Category 1 value for the class (e.g., program or discipline).
`category2`	string	Category 2 value for the class.

Property	Type	Description
`category3`	string	Category 3 value for the class.
`description`	string	Text description of the class.
`end_date`	string	Class end date in ISO 8601 date format: `YYYY-MM-DD`.
`end_time`	string	Class end time in 24-hour format: `hh:mm`.
`gender`	string	Gender restriction for the class. Typical values: `Male`, `Female`, `Both`.
`instructors`	array<string>	List of instructor display names associated with the class. The first element is the primary instructor.
`location`	string	Deprecated Legacy location identifier. Use `location_code` or `location_name` instead.
`location_code`	string	Code for the location where the class is held. (Short code, typically used in filtering and URLs.)
`location_name`	string	Full name of the location where the class is held.
`master_class`	bool	Indicates if this is a "per-day enrollment" class. `true` = per-day enrollment, using additional details in `openings.days` and `tuition.days`.
`max_age`	string	Maximum age allowed for the class, in ISO 8601 duration format, e.g. `P15Y0M` (15 years).
`meeting_days`	object	Boolean flags indicating which days of the week the class meets. See "Meeting Days Object" below.
`min_age`	string	Minimum age allowed for the class, in ISO 8601 duration format, e.g. `P8Y0M` (8 years).
`name`	string	Display name of the class.
`online_reg_link`	string	URL-encoded and HTML-encoded link to web registration for this specific class. If `openings.calculated_openings <= 0`, the link will be marked for waitlist.
`openings`	object	Openings information for normal and per-day enrollment classes. See "Openings Object" below.
`reg_start_date`	string	Date when online registration opens for this class, in ISO 8601 format: `YYYY-MM-DD`.
`room`	string	Room where the class is held (may be blank).
`session`	string	Session name associated with the class (e.g., "Openings").
`start_date`	string	Class start date in ISO 8601 date format: `YYYY-MM-DD`.
`start_time`	string	Class start time in 24-hour format: `hh:mm`.

Property	Type	Description
`waitlist`	bool	Indicates if the class is in waitlist-only mode: • `true` – calculated openings = 0 and waitlists are allowed. • `false` – openings are available *or* waitlists are not allowed.
`location_addr1`	string	Primary address line for the class location.
`location_addr2`	string	Secondary address line for the class location (suite, unit, etc.).
`location_city`	string	City for the class location.
`location_state`	string	State or province for the class location.
`location_postalcode`	string	Postal/ZIP code for the class location.
`location_phone`	string	Primary contact phone number for the class location.
`BillingCycle`	string	Billing cycle for the class tuition (e.g., `Monthly`).
`tuition`	object	Tuition information for normal and per-day enrollment classes. See "Tuition Object" below.

Meeting Days Object (`meeting_days`)

The `meeting_days` object contains a boolean flag for each day of the week indicating whether the class meets on that day.

Property	Type	Description
`mon`	bool	`true` if the class meets on Monday.
`tue`	bool	`true` if the class meets on Tuesday.
`wed`	bool	`true` if the class meets on Wednesday.
`thu`	bool	`true` if the class meets on Thursday.
`fri`	bool	`true` if the class meets on Friday.
`sat`	bool	`true` if the class meets on Saturday.
`sun`	bool	`true` if the class meets on Sunday.

Openings Object (`openings`)

The `openings` object represents seat availability for the class overall and, in per-day scenarios, per meeting day.

Property	Type	Description
`calculated_openings`	int	Calculated openings for standard enrollment, factoring in web registration settings (e.g., max enrollments, waitlist rules).

Property	Type	Description
`days`	object	Per-day openings when `master_class = true` (per-day enrollment). See "Openings Per-Day Object" below.

Openings Per-Day Object (`openings.days`)

Property	Type	Description
`mon`	int	Available openings for Monday.
`tue`	int	Available openings for Tuesday.
`wed`	int	Available openings for Wednesday.
`thu`	int	Available openings for Thursday.
`fri`	int	Available openings for Friday.
`sat`	int	Available openings for Saturday.
`sun`	int	Available openings for Sunday.

Tuition Object (`tuition`)

The `tuition` object represents billing information for normal and per-day enrollment classes.

Property	Type	Description
`fee`	float	Tuition fee for a standard enrollment class or the "day 1" fee for a per-day enrollment class.
`days`	object	Additional tuition fees for multi-day enrollment in per-day classes. See "Tuition Per-Day Object" below.

Tuition Per-Day Object (`tuition.days`)

Property	Type	Description
`day_2`	float	Tuition fee for enrolling in 2 days.
`day_3`	float	Tuition fee for enrolling in 3 days.
`day_4`	float	Tuition fee for enrolling in 4 days.
`day_5`	float	Tuition fee for enrolling in 5 days.
`day_6`	float	Tuition fee for enrolling in 6 days.
`day_7`	float	Tuition fee for enrolling in 7 days.

Example Response

```
{
  "rows": [
    {
      // type: int
      // Unique ID for the class
      "id": 16858445,

      // type: string
      "category1": "Boys Flip-Tastics",
      // type: string
      "category2": "",
      // type: string
      "category3": "4Thu",

      // type: string
      "description": "This class will teach you all about the openings listings!",

      // type: string, Date ISO 8601: YYYY-MM-DD
      "end_date": "2015-10-30",
      // type: string, Time ISO 8601: hh:mm (24-hour)
      "end_time": "14:15",

      // type: string
      "gender": "Both",

      // type: string[]
      "instructors": [
        "Justin Tyme",
        "Barbara Foster",
        "Ashley Lineberry"
      ],

      // type: string
      // DEPRECATED: use location_code or location_name instead.
      "location": "JRT",

      // type: string
      "location_code": "JRT",
      // type: string
      "location_name": "Jackrabbit Technologies",

      // type: bool
      // true if class is a 'Per Day' enrollment class
      "master_class": false,

      // type: string, Duration ISO 8601: P12Y6M = 12 years 6 months
      "max_age": "P15Y0M",

      // description: days of the week the class meets
      "meeting_days": {
        // type: bool
        "mon": true,
```

```
    "tue": true,
    "wed": true,
    "thu": true,
    "fri": true,
    "sat": false,
    "sun": false
  },

  // type: string, Duration ISO 8601
  "min_age": "P8Y0M",

  // type: string
  "name": "Openings Example",

  // type: string
  // URL encoded & HTML encoded link to web registration for this class.
  // Will be marked for waitlist if openings.calculated_openings <= 0
  "online_reg_link": "https://app3.jackrabbitclass.com/reg.asp?id=9209\u0026amp;hc=\u0026amp;initEm

  // description: openings for normal and 'per day' enrollment classes
  "openings": {
    // type: int
    // calculated openings for normal enrollment classes; honors web reg settings
    "calculated_openings": 1,

    // description: openings for each day (used if master_class = true)
    "days": {
      // type: int
      "mon": 0,
      "tue": 0,
      "wed": 0,
      "thu": 0,
      "fri": 0,
      "sat": 0,
      "sun": 0
    }
  },

  // type: string, DateTime ISO 8601
  "reg_start_date": "2015-06-04",

  // type: string
  "room": "",

  // type: string
  "session": "Openings",

  // type: string, Date ISO 8601
  "start_date": "2015-06-04",
  // type: string, Time ISO 8601: hh:mm (24-hour)
  "start_time": "10:30",

  // type: bool
```

```
        // false if calculated_openings > 0 or waitlists are not allowed
        // true if calculated_openings = 0 and waitlists are allowed
        "waitlist": true,

        // type: string
        // Description: Location where class is held
        "location_addr1": "123 East Street",
        "location_addr2": "Suite 105",
        "location_city": "Cleveland",
        "location_state": "OH",
        "location_postalcode": "22378",
        "location_phone": "555-555-1212",

        // type: string
        "BillingCycle": "Monthly",

        // description: tuition for normal and 'per day' enrollment classes
        "tuition": {
          // type: float
          // tuition fee for normal enrollment or "day 1" fee for per-day enrollment
          "fee": 50.0,

          // description: tuition fees for additional days in a per-day enrollment class
          "days": {
            // type: float
            "day_2": 0,
            "day_3": 0,
            "day_4": 0,
            "day_5": 0,
            "day_6": 0,
            "day_7": 0
          }
        }
      }
    ],
    "success": true,
    "message": "Class listings returned successfully!"
}
```

How to Use This Endpoint

For school owners, admins, and power users

This guide is meant for small business and school owners who may not be full-time developers, but want to show live class openings on their website or in a simple app. If you're a power user (or using AI to build apps), the same concepts apply — you'll just take them further.

> **Big picture:** You send a small request with your `OrgId` (and optional filters), and you get back a structured list of classes in JSON that you can display however you like.

---

1. What This Endpoint Actually Does

The `OpeningsJson` endpoint answers questions like:

- "What classes are currently available for my organization?"
- "Which classes still have open spots and which are waitlist only?"
- "What are the times, days, tuition, and registration links for those classes?"

It returns a JSON object with:

- `rows` – a list of class objects (one per class)
- `success` – whether the call worked
- `message` – a short status message

---

2. What You Need Before You Start

- Your Jackrabbit **OrgId** (5–6 digit number).
- Somewhere to use the data:
  - A simple web page where you can paste some JavaScript, or
  - A REST client such as Postman / Insomnia / Thunder Client, or
  - An app (or AI-generated app) that can make HTTP requests.
- Optional filters, such as:
  - `session` (e.g. `Summer 2025`)
  - `ClassDays` (e.g. `Mon,Wed,Fri`)
  - `Cat1` / `Cat2` / `Cat3` (program categories)

---

3. Try It First Without Writing Any Code

Before putting this on your website, it's helpful to see the raw JSON the endpoint returns. You can do this with a tool like Postman.

1. Create a new request and set:
   - **Method:** `POST`
   - **URL:** `https://app.jackrabbitclass.com/jr4.0/Openings/OpeningsJson`
2. In the request body, choose **form-data** or **x-www-form-urlencoded** and add:
   - `orgid` = your OrgId (e.g. `507172`)

   Optional:
   - `session = Summer 2025`
   - `ShowClosed = 1` (include full/waitlist classes)
3. Send the request.
4. You should see a JSON response with `rows`, `success`, and `message`.

> **If you get no data:** Try again with just `orgid` and no other filters. If that works, add filters one at a time until you get the result you want.

4. Understanding the Response (Just Enough to Be Dangerous)

The response is JSON that looks like this:


```
{
  "rows": [
    {
      "id": 16858445,
      "name": "Openings Example",
      "meeting_days": { "mon": true, "wed": true, "fri": true, ... },
      "start_time": "10:30",
      "end_time": "11:15",
      "openings": {
        "calculated_openings": 3,
        "days": { "mon": 1, "wed": 1, "fri": 1, ... }
      },
      "waitlist": false,
      "online_reg_link": "https://app3.jackrabbitclass.com/..."
    }
  ],
  "success": true,
  "message": "Class listings returned successfully!"
}
```


Each item in `rows` is one class. Useful fields:

- `name` – the class name students will see
- `meeting_days` – which days (Mon–Sun) it meets
- `start_time / end_time` – class time
- `openings.calculated_openings` – how many spots remain
- `waitlist` – if the class is waitlist only
- `online_reg_link` – direct link to register for that specific class

> **Simple rule of thumb:** show classes where `openings.calculated_openings > 0` as "Open" and `waitlist = true` as "Waitlist Only".

---

5. Embedding Live Class Listings on Your Website

Below is a minimal example you can drop into an HTML page. It:

- Calls the endpoint with your `OrgId` and optional `session`
- Builds a simple list of classes
- Shows the register link for each class

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
```

```
  <title>My Class Openings</title>
</head>
<body>
  <h1>Available Classes</h1>
  <div id="class-list">Loading classes...</div>

  <script>
    // 1. Build the form data to send
    var formData = new FormData();
    formData.append('orgid', 'YOUR_ORG_ID_HERE');   // e.g. 507172
    formData.append('session', 'Summer 2025');      // optional

    // 2. Send the request (simple CORS-friendly POST)
    fetch('https://app.jackrabbitclass.com/jr4.0/Openings/OpeningsJson', {
      method: 'POST',
      body: formData
    })
    .then(function (response) { return response.json(); })
    .then(function (data) {
      var container = document.getElementById('class-list');

      if (!data.success) {
        container.textContent = 'Error: ' + (data.message || 'Unable to load classes.');
        return;
      }

      if (!data.rows || data.rows.length === 0) {
        container.textContent = 'No classes found for the selected filters.';
        return;
      }

      var list = document.createElement('ul');

      data.rows.forEach(function (cls) {
        var li = document.createElement('li');

        // Determine meeting days text
        var days = [];
        if (cls.meeting_days.mon) days.push('Mon');
        if (cls.meeting_days.tue) days.push('Tue');
        if (cls.meeting_days.wed) days.push('Wed');
        if (cls.meeting_days.thu) days.push('Thu');
        if (cls.meeting_days.fri) days.push('Fri');
        if (cls.meeting_days.sat) days.push('Sat');
        if (cls.meeting_days.sun) days.push('Sun');

        var label = cls.name + ' – ' + days.join('/') +
                    ' ' + cls.start_time + '–' + cls.end_time;

        if (cls.openings && typeof cls.openings.calculated_openings === 'number') {
          label += ' (Open spots: ' + cls.openings.calculated_openings + ')';
        }
```

```
          if (cls.waitlist) {
            label += ' [Waitlist]';
          }

          li.textContent = label;

          // Add registration link
          if (cls.online_reg_link) {
            var link = document.createElement('a');
            link.href = cls.online_reg_link;
            link.textContent = ' Register';
            link.style.marginLeft = '0.5rem';
            li.appendChild(link);
          }

          list.appendChild(li);
        });

        container.innerHTML = '';
        container.appendChild(list);
      })
      .catch(function (error) {
        document.getElementById('class-list').textContent =
          'An error occurred: ' + error;
      });
    </script>
  </body>
  </html>
```

> **Customize this:** Replace `YOUR_ORG_ID_HERE` with your actual OrgId, tweak the `session` value, and adjust the label text to match your branding (“spots left”, “seats available”, etc.).

---

6. Common Filter Recipes

Here are some easy combinations you can use in your request:

- **Show only a specific session**
  Add `session = "Summer 2025"` .
- **Show only classes on certain days**
  Add `ClassDays = "Mon,Wed"` to see Monday/Wednesday classes only.
- **Hide full classes**
  Keep `ShowClosed = 0` (the default) and optionally only display classes where `openings.calculated_openings > 0` .
- **Filter by category**
  Add `Cat1` , `Cat2` , or `Cat3` with the program name you use in Jackrabbit (for example, `Cat1 = "Dance"`).

---

7. Sorting Without Extra Code

You can control the sort order using the `Sort` parameter so you don't have to sort manually in your page.

- **Sort by name (A → Z)**
  `Sort = "Name ASC"`
- **Sort by start date, then name**
  `Sort = "Start_Date ASC, Name ASC"`
- **Bring Monday classes to the top**
  `Sort = "Meeting_Days.Mon DESC, Name ASC"`

> Valid sort property names come from the fields described in the **Output / Response Structure** section.

---

8. Troubleshooting

- **No classes returned:** Try calling with only `orgid`. If that works, add filters one by one.
- **Browser CORS errors:** Make sure you:
  - Use `POST`, not `GET`,
  - Don't add custom headers like `Authorization`,
  - Send data as `FormData` just like the example.
- **"success" is false:** Check the `message` field for details and confirm your `OrgId` and parameter names are spelled correctly.

---

9. Ideas for Power Users & AI-Generated Apps

If you're comfortable with APIs (or using AI to generate code), you can:

- Build a "Class Finder" with filters for age, day, and category that passes those values through to `OpeningsJson`.
- Create a small backend service (Node.js, Python, etc.) that:
  - Caches responses briefly to reduce traffic, and
  - Provides a simplified JSON API to your website or mobile app.
- Use AI to generate React/Vue/Angular components that:
  - Call this endpoint,
  - Display the `rows` in a table or cards, and
  - Add client-side filters and search.

No matter how advanced your setup is, the pattern is always the same: **send a request with your OrgId and filters in, use the `rows` list out.**