ChatGPT

Integrating Highlightly's College Baseball API into BlazeSportsIntel.com

This guide explains how to integrate **Highlightly's MLB/NCAA Baseball API** into your site to provide deep college baseball coverage. The API offers rich data for schedules, live and historical scores, team rosters, standings, highlights, and more. Each section below explains how to acquire credentials, configure requests, and implement the major features needed for **Blaze Sports Intel**.

1. Get API credentials and choose a platform

1. **Create an account** at Highlightly or via RapidAPI. Both platforms let you manage billing and view usage, but RapidAPI does not support custom/discount plans. Accounts are **not shared across platforms** and you must obtain your own API key.
2. **Retrieve your API key** from your chosen platform. Each request must include an `x-rapidapi-key` header containing this key [1]. When calling through RapidAPI you must also include an `x-rapidapi-host` header set to `mlb-college-baseball-api.p.rapidapi.com` [1].
3. **Secure your key**. Do **not** hard-code it in client-side code. Instead, store the key in a server-side environment variable (e.g., `HL_API_KEY`) or a secure secret manager. Your backend requests should read the key from this environment variable and add it to the HTTP headers.

2. Select the base URL and configure your HTTP client

- If you signed up directly with Highlightly, call the API at `https://baseball.highlightly.net`.
- If you subscribed through RapidAPI, use `https://mlb-college-baseball-api.p.rapidapi.com`. Both hosts expose the same endpoints; RapidAPI requires the extra `x-rapidapi-host` header [1].

All endpoints return JSON and require `Content-Type: application/json`. Use a robust HTTP client (e.g., axios in Node.js or Python's `requests` module) and set a timeout (e.g., 10 seconds) to handle network hiccups. Include the two mandatory headers on every request:

```
x-rapidapi-key: <your-secret-key>
x-rapidapi-host: mlb-college-baseball-api.p.rapidapi.com   # only when using RapidAPI
```

3. Core endpoints for college baseball coverage

Highlightly's API divides functionality into several resources. The table below summarizes the main routes needed for NCAA Division I baseball, which uses the abbreviation `NCAA` [2]. When filtering by league, set

the `league` or `leagueName` parameter to `NCAA` (or `NCAA Division I (CBASE)` in the `leagueName` field) to restrict results to college baseball.

3.1 Upcoming and finished games

Use the `GET /matches` endpoint to list games. This route returns basic information about each match (date, teams, state, scores). It refreshes once a minute [3]. You **must specify at least one primary query parameter** such as `league`, `date`, `season`, `homeTeamId` or `awayTeamId`; the optional `timezone`, `limit`, and `offset` parameters control pagination [4]. Relevant query parameters include [5]:

Parameter	Purpose	Example
`league`	Filter by MLB or NCAA league	`league=NCAA` [6]
`date`	Match date in `YYYY-MM-DD` format	`date=2025-02-14` [7]
`timezone`	IANA timezone to localize dates and times	`timezone=America/Chicago` (the user's timezone)
`season`	Season year	`season=2025`
`homeTeamId` / `awayTeamId`	Filter by specific team ID	`homeTeamId=10291199` [8]
`limit` / `offset`	Pagination; defaults are 100 and 0 respectively [9]	`limit=50&offset=0`

Sample request (Node.js with `fetch`):

```
const apiHost = 'https://baseball.highlightly.net';
const headers = {
  'x-rapidapi-key': process.env.HL_API_KEY,
  // Only include the host header if using RapidAPI
  //'x-rapidapi-host': 'mlb-college-baseball-api.p.rapidapi.com'
};

// Fetch upcoming NCAA games on a specific date in America/Chicago time
async function fetchMatches(date) {
  const url = new URL(apiHost + '/matches');
  url.searchParams.set('league', 'NCAA');
  url.searchParams.set('date', date);        // e.g., '2025-02-14'
  url.searchParams.set('timezone', 'America/Chicago');
  url.searchParams.set('limit', '50');

  const res = await fetch(url.toString(), { headers });
  if (!res.ok) throw new Error(`Matches request failed: ${res.status}`);
  const data = await res.json();
```

```
  return data.data;  // array of match objects
}
```

For each game, the API returns fields such as `id`, `date`, `league`, `season`, `homeTeam`, `awayTeam`, and `state` [10]. Use the `state.description` field (e.g., `Finished`, `Scheduled`, `In Progress`) to display match status [11].

3.2 Detailed match information

When the user clicks on a game, call `GET /matches/{id}` to fetch detailed data: venue, weather forecast, statistics, rosters, plays and referees [12]. The only parameter is the `id` path variable. Example:

```
async function fetchMatchDetails(matchId) {
  const res = await fetch(`${apiHost}/matches/${matchId}`, { headers });
  if (!res.ok) throw new Error(`Match details failed: ${res.status}`);
  return res.json();
}
```

3.3 Highlights and recaps

The `GET /highlights` endpoint returns video clips, including live plays, recaps and interviews. Videos may be **VERIFIED** (official sources) or **UNVERIFIED** (user-uploaded) [13]. At least one primary query parameter—such as `leagueName`, `date`, `season`, `matchId`, `homeTeamId` or `awayTeamId`—must be specified [14]. Useful parameters include:

Parameter	Description	Example
`leagueName`	Name of the league (use `NCAA`) [15]	`leagueName=NCAA`
`date`	Return highlights for a day (`YYYY-MM-DD`) [16]	`date=2025-02-14`
`season`	Season year [17]	`season=2025`
`matchId`	Fetch highlights for a specific game [18]	`matchId=394`
`homeTeamId` / `awayTeamId`	Narrow to a specific team [19]	`homeTeamId=10291525`
`limit` / `offset`	Pagination (max `limit` is 40) [20]	`limit=20&offset=0`

Example request:

```
async function fetchHighlights(matchId) {
  const url = new URL(apiHost + '/highlights');
  url.searchParams.set('matchId', matchId);
```

```
  url.searchParams.set('leagueName', 'NCAA');
  url.searchParams.set('limit', '20');
  const res = await fetch(url.toString(), { headers });
  if (!res.ok) throw new Error(`Highlights failed: ${res.status}`);
  return (await res.json()).data;
}
```

Each highlight contains an `id`, `title`, `source`, `channel`, `embedUrl`, and an optional `description` [21]. To embed a video in your site, use the `embedUrl` field. Some clips may have geo-restrictions; call `GET /highlights/geo-restrictions/{highlightId}` before embedding to verify availability.

3.4 Standings

To display the NCAA Division I standings (conference/division tables), call `GET /standings`. Specify either `leagueName` (e.g., `leagueName=NCAA Division I (CBASE)`) or the abbreviation `NCAA` [2]. Optional parameters include `year` (season year), `limit` and `offset` for pagination [22]. Example:

```
async function fetchStandings(year) {
  const url = new URL(apiHost + '/standings');
  url.searchParams.set('abbreviation', 'NCAA'); // or leagueName
  url.searchParams.set('year', year.toString());
  url.searchParams.set('limit', '10');
  const res = await fetch(url.toString(), { headers });
  if (!res.ok) throw new Error(`Standings failed: ${res.status}`);
  return res.json();
}
```

3.5 Teams and team details

- **List teams:** `GET /teams` returns all teams. Filter by `league` (`NCAA`), `name`, `displayName` or `abbreviation` to search for specific schools.
- **Team profile:** `GET /teams/{id}` returns static information (logo, name, abbreviation, league) for a team.
- **Team statistics:** `GET /teams/statistics/{id}` provides season statistics (games played, wins/losses, points for/against). You **must** specify a `fromDate` query parameter in `YYYY-MM-DD` format; optionally set `timezone` to `America/Chicago` [23]. The endpoint refreshes immediately after a match finishes [24].

Example to fetch Baylor's team stats since 1 February 2025:

```
async function fetchTeamStats(teamId, fromDate) {
  const url = new URL(`${apiHost}/teams/statistics/${teamId}`);
  url.searchParams.set('fromDate', fromDate); // e.g., '2025-02-01'
  url.searchParams.set('timezone', 'America/Chicago');
```

```
  const res = await fetch(url.toString(), { headers });
  if (!res.ok) throw new Error(`Team stats failed: ${res.status}`);
  return res.json();
}
```

3.6 Recent form and head-to-head

- **Last five games:** `GET /last-five-games` returns the five most recent completed games for a team. Provide the `teamId` query parameter [25]. Use this endpoint to show how each team has performed lately.

```
async function fetchLastFive(teamId) {
  const url = new URL(apiHost + '/last-five-games');
  url.searchParams.set('teamId', teamId);
  const res = await fetch(url.toString(), { headers });
  return (await res.json()).data;
}
```

- **Head-to-head:** `GET /head-2-head` compares two teams by returning their last ten meetings [26]. Supply `teamIdOne` and `teamIdTwo` as query parameters. The order of IDs does not matter [26].

```
async function fetchHeadToHead(teamIdOne, teamIdTwo) {
  const url = new URL(apiHost + '/head-2-head');
  url.searchParams.set('teamIdOne', teamIdOne);
  url.searchParams.set('teamIdTwo', teamIdTwo);
  const res = await fetch(url.toString(), { headers });
  return (await res.json()).data;
}
```

3.7 Lineups

If you wish to display projected starting lineups, call `GET /lineups/{matchId}` a few hours before first pitch [27]. It returns separate `home` and `away` sections with player IDs, names, jersey numbers, positions and whether the player is a starter [28].

```
async function fetchLineups(matchId) {
  const res = await fetch(`${apiHost}/lineups/${matchId}`, { headers });
  return res.json();
}
```

3.8 Odds and bookmakers (optional)

The `GET /odds` and `GET /bookmakers` endpoints provide prematch and live betting odds as well as information about bookmakers. **These routes are not available on the Basic/Free plan** [29], so you need a

paid tier. Use `oddsType=live` or `prematch` and filter by `leagueName`, `matchId`, `bookmakerId`, `date` and other fields [30]. This information is optional if your site does not display betting lines.

4. Pagination and rate limiting

Most endpoints support pagination via `limit` and `offset`. Always set `limit` to a sensible number (e.g., 20 or 50) and increase `offset` to fetch subsequent pages. The API returns a `pagination` object with `totalCount`, `offset` and `limit` fields to help you build "Load more" controls [31].

Highlightly enforces per-plan rate limits. Inspect the `x-ratelimit-requests-remaining` header in each response to see how many requests you have left [32]. If you exhaust your quota, wait until the next reset before making more calls. Implement simple caching on your server (e.g., store responses in memory or Redis for one minute) to reduce duplicate requests, especially since endpoints like `matches` and `highlights` refresh at most once per minute [3] [33].

5. Practical integration tips for Blaze Sports Intel

- **Use server-side API calls.** To prevent exposure of your secret key and reduce latency, call Highlightly from your backend. Then return only the necessary data to the client (e.g., via your own REST endpoints).
- **Timezone handling.** Because your user base is in San Antonio (America/Chicago), set `timezone=America/Chicago` on all date-sensitive endpoints (e.g., `matches`, `highlights`, `team statistics`) [4] [23].
- **Filtering for NCAA only.** Always pass `league=NCAA` (for `/matches`) or `leagueName=NCAA` / `abbreviation=NCAA` (for `/highlights` and `/standings`) [6] [2]. This avoids mixing MLB data with college baseball.
- **Handling match states.** Use the `state` object to decide how to display each game: "Scheduled" games need lineups and start times; "In Progress" games need live scores; and "Finished" games should show the final score and link to recaps. Recognise states like `Postponed`, `Canceled`, `Rain Delay` and `Suspended` to handle unusual situations [11].
- **Highlights embedding.** Always check for an `embedUrl` and verify geo-restrictions (via `/highlights/geo-restrictions/{id}`) before embedding videos [34]. Provide fallback text when a clip cannot be embedded.
- **Team/Player imagery.** The API returns `logo` URLs for teams; serve these directly or proxy them through your CDN. Player images are not yet provided—augment your database with official roster photos if needed.
- **Respect user preferences.** Austin requested to focus on sports ignored by mainstream media; emphasise college baseball stories. Avoid references to sports he does not care about (e.g., soccer) and ensure your site covers **every NCAA team**, not just the big names.

6. Example environment configuration

Create a `.env` file on your server and define your secret key:

```
# .env
HL_API_KEY=your-very-long-secret-api-key
HL_API_BASE=https://baseball.highlightly.net
```

In your backend code, read these variables and set headers accordingly. Use a configuration module to avoid repeating header definitions across requests.

7. Testing and error handling

During development, test your integration with sample requests for different dates and teams. Handle HTTP status codes:

- **200** – Success; parse the JSON and update your UI.
- **400** – Bad request; likely missing required parameters (check that you specify at least one primary query parameter for `matches` or `highlights`) [14] [4].
- **500** – Internal server error; retry after a short delay or contact Highlightly support at support@highlightly.net.

8. Support and next steps

If you encounter issues or need extra features (e.g., additional statistics or extended quotas), email Highlightly at **support@highlightly.net** [35]. Their team can tailor custom plans or provide guidance on new functionality.

Once the API calls work in your staging environment, integrate them into the production version of **Blaze Sports Intel**, design attractive UI components for schedules, standings, team pages and highlight reels, and ensure the site remains performant and within rate limits.

This guide provides all of the technical details necessary to implement comprehensive NCAA Division I baseball coverage using Highlightly's API. Use the sample code as a starting point and adapt it to the frameworks and languages used by your development team.

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] MLB, NCAA Baseball API | Highlightly
https://highlightly.net/documentation/baseball/