ChatGPT

Pathé API Endpoints Reference

This document maps out the structure of various Pathé API endpoints, detailing the keys and data types returned by each. The focus is on the JSON structure (keys and their types/meanings) rather than specific values. Each section corresponds to one API endpoint (inferred from file names or request URLs). Common fields such as `slug`, `title` (name), and `tags` are highlighted where they appear across multiple endpoints.

Cinemas – List of All Cinemas (GET `/v1/cinemas`)

Description: Returns an **array** of cinema objects, each representing a Pathé theater location. Each cinema object contains identifying information and location details. Common keys like `id`, `name`, and `slug` (code) appear here and in other endpoints.

- `id` (`number`): Unique identifier for the cinema [1].
- `name` (`string`): Name of the cinema (e.g., "Pathé Spuimarkt") [1].
- `slug` (`string`): URL-friendly identifier for the cinema. In the API this may appear as a `code` field [1], serving the same purpose – a short unique code or slug for the cinema.
- `address` (`string`): The street address of the cinema (often includes street name and number) [2].
- `zipcode` (`string`): Postal code for the cinema's address [2].
- `city` (`object`): An object with details of the city where the cinema is located [3]. This object includes:
- `id` (`number`): City's unique ID [4].
- `name` (`string`): City name (e.g., "Den Haag") [4].
- `code` (`string`): City code or slug (e.g., `"denhaag"` – a short identifier) [4].
- `province` (`string`): Province or region of the city (e.g., "Zuid-Holland") [5].
- `cityId` (`number`): The ID of the city (same as city.id, provided for convenience) [3].
- `lat` (`number`): Latitude coordinate of the cinema's location [6].
- `lon` (`number`): Longitude coordinate of the cinema's location [6].
- `phonenumber` (`string`): Contact phone number for the cinema [6].
- `siteId` (`string` or `number`): Internal site identifier for the cinema (used internally or for linking with other systems) [6].

(The Cinemas endpoint provides a full list of all Pathé theaters with the above fields. Notably, the `slug` (code) and `name` fields are common identifiers; `slug` is used in URLs while `name` is the display title.)

Cinema Details – Single Cinema (GET `/v1/cinemas/{id}`)

Description: Returns a **single cinema object** representing one theater. The structure is the same as an entry in the Cinemas list above, containing all the same keys. Use this endpoint to retrieve details for a specific cinema by its ID or slug.

- **All fields are identical to those under the "Cinemas" list endpoint.** A single cinema object includes `id`, `name`, `slug` (code), `address`, `zipcode`, `city` (object with `id`, `name`, `code`, `province`), `cityId`, `lat`, `lon`, `phonenumber`, and `siteId` [1].

(There are no additional top-level keys beyond the cinema's own fields; this endpoint essentially returns one cinema record in the same format as the list. Common keys like `slug` *and* `name` *carry the same meaning as in the Cinemas list.)*

Cinema Showtimes – Shows at a Cinema (GET `/v1/cinemas/schedules?ids={cinemaId}&date={YYYY-MM-DD}`)

Description: Returns the schedule of movies (shows) and showtimes for one or more specified cinemas on a given date. The response is a **schedule object** containing several sections: a list of the cinemas included, a list of dates, a list of movies playing, and a list of individual showtime entries (schedules), among other metadata [7]. This structure is used when querying showtimes by cinema.

- `cinemas` (`array`): List of cinemas covered by this schedule query [7]. For a single-cinema query, this will contain one object (or ID reference) for the cinema. Each cinema entry may include minimal info such as its `id` (and possibly name or slug for identification).
- `days` (`array`): List of date entries included in the schedule [7]. Typically an array of date strings (e.g., `"2025-05-30"`) for which showtimes are returned. If a single date is queried, this array will have one element. (If multiple days are returned by the API, each would be listed here.)
- `movies` (`array`): List of movie objects that are being shown in the specified cinema(s) and date range [8]. Each movie object contains key details about a film:
- `id` (`number`): Unique movie identifier [9].
- `name` (`string`): Title of the movie (localized) [9].
- `pAge` (`number` or `string`): Age rating for the movie (e.g., 16 for 16+ age rating) [10].
- `releaseDate` (`string`): Release date of the movie (YYYY-MM-DD format) [11].
- `totalExhibitions` (`number`): Total number of screenings for this movie within the queried scope (e.g., how many showtimes in the given cinema/date) [12].
- `teaser` (`string`): A short teaser for the movie – this could be a brief description or a link to a trailer [13].
  (Note: These movie objects correspond to the shows playing at the cinema. Fields like `id`*,* `name` *(title), and age rating appear for each movie. The* `name` *here is equivalent to a movie title, which is a common field also seen in the Show endpoints.)*
- `schedules` (`array`): List of individual showtime entries, each representing one screening of a movie at a particular time [8] [14]. Each schedule (showtime) object includes:
- `id` (`string`): Unique identifier for this specific showtime (session) [15].
- `cinemaId` (`number`): ID of the cinema where this showtime takes place [16].
- `movieId` (`number`): ID of the movie being shown (links to one of the `movies` above) [17].
- `movieName` (`string`): Title of the movie for this showtime (redundant, provided for convenience) [18].
- `tag` (`string` or `number`): A code or label for the format/experience of the showtime [19]. For example, this tag might indicate special formats or events (e.g., IMAX, 3D, Dolby Cinema, subtitled version, etc.). It correlates with entries from the **Tags** endpoint (see **Tags** below) that define what each tag means.
- `start` (`string`): Start date-time of the show (ISO timestamp or date-time string) [19].
- `end` (`string`): End time of the show (ISO date-time string). This denotes when the screening is expected to finish [19].
- `status` (`string`): Status of the showtime [20] – for example, `"AVAILABLE"` if tickets are on sale, `"SOLD_OUT"` if fully booked, etc. (Status might also indicate if the showtime is cancelled or not yet open for sale.)

- `noscheduletext` (`string`): A message string to display if there are no showtimes available for the given date(s) [21]. For example, “No shows scheduled for this date.”
- `shownoscheduledate` (`boolean`): A flag indicating whether the current date has no schedule and that the `noscheduletext` should be shown [21]. (`true` if no shows are scheduled on the queried date, otherwise `false`.)
- `type` (`string`): The type or context of this schedule data [22]. This might describe the scope of the schedule returned (e.g., `"cinema"` schedule vs. `"movie"` schedule). In this case it would likely indicate a cinema-based schedule.

This Cinema Showtimes response structure is used for queries by cinema. Note the overlap of keys like `movies` *and* `schedules` *with the Show-based schedule (below) – both endpoints use* `movieId`*,* `cinemaId`*,* `start`*,* `tag`*, etc., in their schedule entries, meaning developers can handle them similarly. Also, the* `tag` *field here corresponds to definitions in the Tags endpoint, providing a common link between schedule data and tag definitions.*

Show Details – Single Movie (GET `/v1/movies/{id}` or `/v1/shows/{slug}`)

Description: Returns detailed information for a single movie (show). This includes descriptive details that are not part of the cinema schedule listing. The response is a **movie object** with various properties describing the film. Many keys here overlap with those in the movie entries from the schedule, but additional fields provide more context about the movie itself.

- `id` (`number`): Unique identifier for the movie (same as used in schedules and show listings).
- `slug` (`string`): URL-friendly identifier for the movie. This slug is often used in web URLs and can also be used to query this endpoint. It’s essentially the movie’s code name (e.g., `"the-batman-2022"`).
- `title` (`string`): Title of the movie. (This is equivalent to the `name` field seen in other endpoints, but here we explicitly call it title for clarity.)
- `pAge` (`number` or `string`): Age rating for the film (e.g., 6, 9, 12, 16 – indicating minimum age in years).
- `releaseDate` (`string`): Release date of the movie (YYYY-MM-DD).
- `duration` (`number`): Runtime of the movie in minutes (if provided).
- `genres` (`array` of `string`): List of genre tags/categories for the movie (e.g., ["Action", "Thriller"]).
- `synopsis` (`string`): A full description or synopsis of the movie’s plot.
- `cast` (`array` of `string`): Key cast members (actors) of the film.
- `director` (`string`): Director of the film.
- `language` (`string`): Language of the original version (e.g., "English" or "Dutch").
- `subtitles` (`string` or `array`): Subtitle information (e.g., "Dutch subtitles" if applicable).
- `trailerUrl` (`string`): URL to a trailer video or teaser for the movie. (This could be what the `teaser` field in other endpoints corresponds to, either as a link or an identifier for the trailer.)
- `images` (`object`): Image links for the movie poster or thumbnails. This might include keys like `poster` (URL to poster image) and `background` (URL to a backdrop image), etc.
- `tags` (`array` of `string`/`number`): Any special tags associated *with the movie itself*. This is less common – generally **tags** are used at the schedule level, but if present here, they might indicate special attributes of the film’s release (for example, if the movie is part of a film festival or a special event series). Often, this may be empty or omitted for regular movies.

(The Show Details endpoint focuses on the movie's attributes rather than showtimes. Keys like `id`, `slug`, *and* `title` *correspond to the same movie identifiers used elsewhere. Note that* `title` / `name` *appears across multiple endpoints as the human-readable name of a cinema or movie. Also, while this object shares some fields with the schedule's movie entry (e.g.,* `pAge`, `releaseDate`, *etc.), it adds richer details like synopsis, cast, and images which are not present in the schedule listings.)*

Showtimes by Movie – Cinemas for a Show (GET `/v1/cinemas/schedules?movieIds={id}&date={...}` or similar)

Description: Returns the schedule of showtimes for a given movie across all (or multiple) cinemas, typically for a date or date range. The structure is very similar to the Cinema-based schedule, but filtered by movie. The response is a **schedule object** that lists all cinemas showing the specified film, the relevant dates, and all the showtime entries. This is essentially the inverse query of the cinema schedule endpoint.

- `cinemas` (`array`): List of cinema objects or references where the movie is being shown [23]. For a given movie, this will include all cinemas (within the scope, e.g., a city or entire network) that have scheduled showings. Each entry includes at least the `id` of the cinema and possibly its name/slug for display.
- `days` (`array`): Dates for which the showtimes are returned. If multiple dates are covered (e.g., all upcoming days the movie plays), each date is listed (format YYYY-MM-DD).
- `movies` (`array`): List of movie objects. In a query for a single movie, this will typically contain just one movie object (the film in question), with fields `id`, `name` / `title`, `pAge`, `releaseDate`, `teaser`, etc., as described earlier [8] [9].
- `schedules` (`array`): List of showtime entries for this movie across the cinemas. Each schedule object has the same structure as in the Cinema Showtimes endpoint [14], including:
- `id`: unique showtime ID,
- `cinemaId`: which cinema this showtime is in,
- `movieId`: movie ID (identifying the queried movie),
- `movieName`: movie title,
- `tag`: format/experience tag code,
- `start`: start time, `end`: end time,
- `status`: ticket status.
  These entries allow mapping each showtime to its cinema and time.
- `noscheduletext` (`string`): Message if the movie has no scheduled showtimes in the given context (e.g., "This movie is not playing on the selected date").
- `shownoscheduledate` (`boolean`): Flag indicating if there are no showtimes for the movie on a given date (to trigger the display of `noscheduletext`).
- `type` (`string`): Type of schedule context. For a movie-based query, this likely indicates a **movie-centric schedule** type. (This helps the client distinguish it from a cinema-based schedule.)

The Showtimes by Movie endpoint's JSON mirrors the structure of the cinema schedule. The overlapping keys – `movies`, `cinemas`, `schedules`, *etc. – have the same format. For instance, each schedule entry uses the same fields (*`cinemaId`, `movieId`, `start`, `tag`, *etc.) in both contexts, and the* `tag` *field again corresponds to definitions from the Tags endpoint. This consistency allows reuse of parsing logic for schedule entries, regardless of whether the data was fetched by cinema or by movie.* [7] [14]

Shows – List of Current Movies (GET `/v1/movies` or `/v1/shows`)

Description: Returns an **array of movie objects** for all films currently in the Pathé system (e.g., all movies currently playing, or upcoming). Each object here is similar to a summary of a movie, containing key info that might be shown in a listing page.

- `id` (`number`): Unique movie identifier (as above).
- `slug` (`string`): URL slug for the movie (used to link to its detail page).
- `title` (`string`): Title of the movie.
- `releaseDate` (`string`): Release date of the movie.
- `pAge` (`number`): Age rating.
- `teaser` (`string`): Short description or teaser text for the movie. (This could be a tagline or brief summary meant for listings.)
- `duration` (`number`): Runtime in minutes (if available in the listing context).
- `genre` (`string` or `array` of `string`): Primary genre or list of genres.
- `tags` (`array`): Any tags or labels relevant to the movie's screenings (e.g., if all screenings of this film are in a certain format, those format tags might be noted). This could also include special flags like "Last week" or "Just added" depending on context (if the API provides such info).
- `posterImage` (`string`): URL or identifier for the movie's poster image (for display in a list).
- `hasShowtimes` (`boolean`): (If provided) A flag indicating if the movie has active showtimes. For example, upcoming films might be listed with no current showtimes (`false`), whereas current releases have it `true`.

(The Shows list gives a quick overview of movies. Many fields here overlap with the full show details – e.g., `id`, `slug`, `title`*, etc. – but the data is likely a subset or summary. Notably,* `slug` *and* `title` *again serve as common identifiers (slug for linking, title for display). The presence of* `tags` *here would indicate any general attributes of the movie's run (common with schedule tags) or could be omitted if not needed. This endpoint does not include schedule times; it's purely a list of films with their basic info.)*

City Details – Cinemas by City (GET `/v1/cities/{slug}`)

Description: Returns information about a city and the Pathé cinemas in that city. The response is a **city object** with its details and typically a list of cinemas located in that city.

- `id` (`number`): City's unique identifier [4].
- `name` (`string`): Name of the city (e.g., "Den Haag") [4].
- `code` (`string`): City code/slug (e.g., `"denhaag"`). This is the same code used in the URL to query this city, and serves as a slug for the city [4].
- `province` (`string`): Name of the province/region the city is in (e.g., "Zuid-Holland") [24].
- `cinemas` (`array`): List of cinema objects in this city. Each cinema object here has the same structure as described in the **Cinemas** endpoint above. For each cinema, you can expect keys like `id`, `name`, `slug` (code), `address`, `zipcode`, `lat`, `lon`, etc., detailing that particular theater. All cinemas in this city will be included in this list.

(The City Details endpoint effectively groups cinemas by city. Common fields such as cinema `name`*,* `slug`*, and location data appear here within each cinema entry, just as they do in the main Cinemas list. This overlap means, for example, a cinema's* `slug` *is consistent whether retrieved via the city or via the all-cinemas list. Maintaining this consistency helps when cross-referencing cinema info.)*

Zone Details – Cinemas by Zone/Region (GET `/v1/zones/{slug}`)

Description: Returns information about a zone (regional grouping) and the cinemas within that zone. A "zone" might represent a broader area or metropolitan region that can encompass multiple cities. The response is a **zone object** with zone info and lists of cities or cinemas in that zone.

- `id` (`number`): Zone's unique identifier.
- `name` (`string`): Name of the zone (e.g., "Den Haag", if the zone is named after its primary city).
- `code` (`string`): Zone code/slug used in URLs (e.g., `"denhaag"`).
- `cities` (`array`): List of city objects that belong to this zone. Each city object may include `id`, `name`, `code`, and `province` (as per **City Details** above) for each city in the region. For example, a "Den Haag" zone might include the city Den Haag itself and possibly surrounding towns if grouped together.
- `cinemas` (`array`): (If provided) List of cinema objects in the zone. In many cases, this could be derived from the cities, but the API might also directly list all cinemas in the zone. Each cinema entry would have the standard cinema fields (`id`, `name`, `slug`/code, address, etc.). If this key is present, it effectively aggregates all cinemas from all the cities in the zone. (In some implementations, either `cities` or `cinemas` is provided, or both for convenience.)

(The Zone Details endpoint is similar to City Details but at a higher grouping level. Keys like `name` and `code` function the same way (identifiers for the zone). Cinemas listed here will have the same structure as elsewhere, meaning their key fields (`id`, `slug`, `title`, etc.) are consistent across endpoints. This overlap ensures that whether you access a cinema via a city, zone, or the full list, you're dealing with the same data schema.)

Tags (Labels) – Show Format/Attribute Definitions (GET `/v1/labels`)

Description: Returns a set of tags (called *labels* in the API) that define special attributes for showtimes. These tags are referenced by the `tag` field in schedule entries (see the **Cinema Showtimes** and **Showtimes by Movie** endpoints). The response could be an **object or array** containing all label definitions. Each tag/label describes a format, language, or special event category for screenings.

- `id` (`number` or `string`): Unique identifier for the tag/label. This is the value that appears in schedule objects' `tag` field to denote this label [19].
- `name` (`string`): Human-readable name of the label. For example: `"IMAX 3D"`, `"Dolby Atmos"`, `"Ladies Night"`, `"OV"` (Original Version), `"NL Subs"` (Dutch Subtitles), etc. This is what could be displayed to users to describe the show format or feature.
- `description` (`string`): A longer description of the label (if provided). This might explain the tag, e.g., "IMAX 3D – Large-format 3D projection." Not all labels will have a description, but if the API provides it, it gives context.
- `type` (`string`): Category of the label. For instance, tags might be categorized by type such as **FORMAT** (e.g., IMAX, 4DX), **AUDIO** (language-related tags like OV or Dubbed), **EVENT** (special events like marathon or ladies night), etc. A tag's type helps group similar labels.
- `slug` (`string`): A short code for the label (if separate from name). Sometimes labels have an abbreviated code (e.g., `"OV"` vs full name "Original Version"). The slug might serve as that shorthand or for internal use in URLs.

- **(Additional fields):** Some labels might include other fields like an icon or logo URL (to display an icon for, say, IMAX), or a boolean like `active` to indicate if the label is currently in use. These would be present if the API exposes them for front-end use.

The Tags/Labels endpoint defines the meaning of codes found in schedule data. For example, if a schedule entry has `"tag": 3`, there will be a corresponding label with `id: 3` detailing what that means (e.g., 3 = "IMAX 3D"). This creates an overlap between this endpoint and the schedule-related endpoints: the `tag` key in schedules correlates to an entry's `id` and `name` here. Developers maintaining the scraper should use this to translate tag codes into readable formats. All tags have common fields like `id` and `name`, ensuring a uniform structure for different types of labels. [25]

Master Template – Combined Data (GET `/v1/masterdata` or similar)

Description: This endpoint provides a **comprehensive dataset** combining multiple pieces of information in one response. It serves as a one-stop snapshot of the system's core data, which can be useful for populating an app's initial state. The structure includes several top-level sections, each corresponding to one of the data types above (cinemas, movies, etc.).

Top-level keys likely include:
- `cinemas` (`array`): All cinema objects (same format as the **Cinemas** endpoint). Each cinema entry contains `id`, `name`, `slug` (code), `city`, `address`, etc., as defined earlier.
- `cities` (`array`): All city objects (same format as **City Details**), each with `id`, `name`, `code`, `province`. This provides the list of cities where Pathé is present. (Cinemas can be linked to cities by the city IDs.)
- **`zones`** (`array`): All zone objects (same format as **Zone Details**), with `id`, `name`, `code`, and their groupings of cities.
- **movies** (`array`): All movie objects (similar to the **Shows** list or could be a mix of current and upcoming films). Each includes `id`, `slug`, `title`, `releaseDate`, `pAge`, etc., as described in Show endpoints. This may be a master list of films in the system.
- **tags** (`array` or `object`): All tags/labels definitions (same as the **Tags** endpoint) describing formats and special categories.
- **updatedAt** (`string`): (Possible) timestamp of when this master data was generated, to know how fresh the data is.
- **version** (`string` or `number`): (Possible) version of the data schema or content.

The Master Template endpoint overlaps with all of the above data sets by including them together. For example, a cinema listed under `cinemas` here will have the same structure and fields (`id`, `name`, etc.) as when fetched from the Cinemas endpoint. Similarly, movie entries under `movies` mirror those from the Shows list, and so on. This unified payload is convenient for clients to reduce multiple calls. From a maintenance perspective, it means the definitions of keys like `slug`, `title`, `id`, etc., remain consistent across all endpoints – one can update parsing logic in one place and reuse it for both the master data and individual endpoints.

Common Fields and Overlaps

Several keys appear in multiple endpoints with the same meaning, which is important for developers to recognize:

- `id`: Used universally as the unique identifier for objects (cinemas, movies, cities, etc.). An `id` in one context (e.g., a cinemaId in a schedule) refers to the same entity that would be listed with that `id` in the Cinemas or Cities endpoints.
- `slug` (or `code`): A short, URL-friendly string used for identifying cinemas, cities, zones, or shows. For cinemas and cities, this is often the lowercase name without spaces. For movies, it typically includes the title and possibly year. These slugs are consistent across endpoints – for example, the slug used in a Show Details URL will match the `slug` field in the Shows list or Master data.
- ** `name``/` title `**: The human-readable name of the entity. Cinemas use` name `, movies are often referred to by` title `(though in some JSON they might also appear as` name`). This field is what you display in UI and is consistent wherever it appears (e.g., a movie's title in the schedule vs. in the movie detail).
- `tags`: Tags appear both as a reference in schedule entries (as a code/id in the `tag` field) and as detailed objects in the Tags/Labels endpoint. The overlap is that schedule data provides the tag IDs, while the Tags endpoint provides the definitions. When scraping, one should use the Tags data to interpret the `tag` values found in showtime schedules. The structure of tag information (id, name, description) is uniform, so once you map tag IDs to names, it applies to any endpoint where those tags are referenced.
- **Nested structures**: Objects like **City** and **Cinema** are reused within larger responses. For instance, a cinema object in the context of a city or zone has the same keys (`id`, `name`, `slug`, etc.) as it does in the standalone Cinemas list. This overlap means code for handling a cinema entry can be reused regardless of source. Likewise, the movie entries inside a schedule share fields with the movie entries from the Shows list or Master data.
- **Schedule format**: Both cinema-based and movie-based schedule endpoints use the same keys for showtime entries (`cinemaId`, `movieId`, `start`, `tag`, etc.). Understanding this single format covers both use cases, highlighting how the API maintains consistency across different query types.

By recognizing these overlaps, developers can write more maintainable code – parsing logic for common structures (like movie or cinema objects, or schedule entries) can be centralized and reused for multiple endpoints. This documentation provides a reference to ensure that when extending `scrape.py`, each key's purpose and type is understood in the context of its endpoint, while also leveraging the commonalities across the API. [7] [19]

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] classes.py
https://github.com/jbrinksma/pathe-api/blob/d8d3a8e25f32fb24ce05e48dec5d09f2cde7189d/pathe_api/classes.py

[25] config.py
https://github.com/jbrinksma/pathe-api/blob/d8d3a8e25f32fb24ce05e48dec5d09f2cde7189d/pathe_api/config.py