GBIF API: Agent Usage

1. Occurrence Data Retrieval

Use Case 1.1: Occurrence Record Search
Purpose: Discover occurrence records based on taxonomic, geographic, temporal, and methodological criteria.

The occurrence search functionality utilizes the `GET /v1/occurrence/search` endpoint. The system supports comprehensive filters available via GBIF API. API parameters are enabled via pydantic model GBIFOccurrenceBaseParams.

As of now, we do not support for the following parameters: bed, biostratigraphy, checklistKey, coordinateUncertaintyInMeters, crawlId, degreeOfEstablishment, endDayOfYear, establishmentMeans, fieldNumber, formation, gadmGid, gadmLevel0Gid, gadmLevel1Gid, gadmLevel2Gid, gadmLevel3Gid, group, hostingOrganizationKey, institutionKey, isInCluster, iucnRedListCategory, latestAgeOrHighestStage, latestEonOrHighestEonothem, latestEpochOrHighestSeries, latestEraOrHighestErathem, latestPeriodOrHighestSystem, lowestBiostratigraphicZone, modified, organismId, organismQuantity, organismQuantityType, otherCatalogNumbers, parentEventId, pathway, programme, protocol, publishedByGbifRegion, recordedByID, relativeOrganismQuantity, repatriated, sampleSizeUnit, sampleSizeValue, samplingProtocol, startDayOfYear, taxonConceptId, taxonId, taxonomicIssue.

Example Query: "Find records of birds in Gainesville, Florida, United States from 2014-2016"

Use Case 1.2: Occurrence Record Statstics
Purpose: Search and breakdowns of occurrence data

Aggregation uses the same `GET /v1/occurrence/search` endpoint but with GBIF faceting parameters. The request pattern mirrors standard search functionality but automatically sets `limit=0` and includes one or more facet parameters to generate aggregated counts. The faceting system supports any valid occurrence parameter as a grouping dimension available via the GBIF API for faceting operations (except `scientificName` - using this as facet parameter - results in a 404 error.).

Example Query: "How many species occurrences does GBIF have for Temperate Asia?"

Use Case 1.3: Individual Occurrence Record Retrieval
Purpose: Retrieve detailed information about specific biodiversity observations or specimens from GBIF.

Agent uses the `GET /v1/occurrence/{gbifId}` endpoint. This follows a direct ID-based lookup pattern where the response provides a single occurrence record with complete metadata.

Example Query: "Find occurrence record 5292056925"

2. Species and Taxonomic Information Services

Use Case 2.1: Scientific Name Resolution and Matching
Purpose: Convert scientific names to standardized GBIF identifiers

Scientific name resolution utilizes the `GET /v2/species/match` endpoint. If there is a scientific name strings in input and it returns matched taxa with usage keys and taxonomy. The agent automatically invokes this service internally whenever scientificName parameters are parsed and identified in user queries. It is similar to using `rgbif` package search

The resolution process follows a systematic approach beginning with the extraction of scientific names from user queries using LLM parsing. The system then calls the species match API for each identified name and validates that the returned taxonomic rank matches the expected taxonomic level. Successfully resolved names result in the replacement of scientificName parameters with their corresponding taxonKeys for improved search performance. It is possible that GBIF `/match` returns multiple records for a scientificName, in that case, the agent uses LLM call to pick the records that matches closest to the user query. Throughout this process, the system generates artifacts for tracking resolution results and maintaining data provenance.

Example Query: N/A. Agent uses this internally.

Use Case 2.2: Species Search
Purpose: Discover species and taxonomic entities using species search

Species discovery employs the `GET /v1/species/search` endpoint. This supports species search queries with filtering capabilities, returning paginated species results.

The search system provides flexible query capabilities supporting both scientific names and vernacular names as query fields. Filtering options also include taxonomic rank, conservation status, habitat preferences, threat levels, and dataset-specific searches.

The system supports comprehensive filters available via GBIF API. API parameters are enabled via pydantic model GBIFSpeciesSearchParams. As of now, we do not have support for the following parameters: scientificNameID, sourceId, taxonConceptID, taxonID, usageKey.

Example Query: "Search for species named Quercus"

Use Case 2.3: Taxonomic Name Statistics
Purpose: search and breakdown species data

This utilizes the same `GET /v1/species/search` endpoint with GBIF faceting parameters. It applies standard search parameters while automatically setting

`limit=0` and enabling faceting functionality to generate aggregated results rather than individual species records.

Example Query: "How many species are in the family Rosaceae?"

Use Case 2.4: Species Taxonomic Information Retrieval
Purpose: Gather information about a taxonomic name including classification, relationships, and synonymy.

Comprehensive taxonomic information retrieval employs a sophisticated multi-endpoint approach utilizing parallel API calls to gather information about a taxonomic name. It queries multiple endpoints including the primary species endpoint `GET /v1/species/{key}`, taxonomic classification through `GET /v1/species/{key}/parents`, child taxa via `GET /v1/species/{key}/children`, synonymous names through `GET /v1/species/{key}/synonyms`, and detailed name information using `GET /v1/species/{key}/name`. The agent determine the API calls based on user query intent.

Example Query: "Get taxonomic information for Rattus including synonyms and children"

3. Dataset Search

Use Case 3.1: Dataset Search
Purpose: Find and analyze data sources and collections within the GBIF network.

Dataset search utilizes the `GET /v1/dataset/search` endpoint. The filters are parsed through the user request and locate relevant datasets, returning dataset information.

The search system provides extensive filtering capabilities across multiple dimensions. Metadata filtering encompasses title, description, and keyword searches for content-based discovery. Geographic filtering includes publishing country specifications and geographic coverage parameters. Technical filtering supports dataset type, licensing information, and format specifications. Taxonomic filtering enables discovery based on taxonomic coverage using taxonKey parameters. Organizational filtering covers publishing organizations, network affiliations, and installation-specific searches.

Example Query: "Find datasets about marine biodiversity from Nordic countries"

GBIF integration

- For each usage; agent generates the API request url as well as portal integration url which maintains continuity by providing direct links to

corresponding dataset portal pages for detailed exploration to the users Source Code: gbif/api.py.

API URLs are automatically converted to corresponding portal URLs:

```
# API URL
https://api.gbif.org/v1/occurrence/search?taxonKey=212&country=US

# Portal URL
https://gbif.org/occurrence/search?taxonKey=212&country=US
```

Parameter Processing

GBIF Enums

Enums parameter values taken from the GBIF OpenAPI Specifications JSON Source Code: models/enums/

```
# Enum values extracted using .value attribute
BasisOfRecord.PRESERVED_SPECIMEN → "PRESERVED_SPECIMEN"
```

Parameter List

Multiple values for the same parameter are handled using URL encoding.

```
taxonKey=2435098&taxonKey=2435099&taxonKey=2435100
```

Status Codes

All API requests include error handling to reflect the correct error to the iChatBio UI: - HTTP status code validation - Error logging with request context - User-friendly error messages - Artifact creation even for failed requests (for tracking)

Automatic Parameter Resolution When users provide taxonomic names without specific keys, the system: 1. Extracts taxonomic names using LLM analysis 2. Resolves names to GBIF keys via species match API 3. Maps resolved keys to appropriate parameter fields 4. Updates request parameters before API calls
Source: gbif/resolve_parameters.py

Example Usage: Find occurrence records

Source: entrypoints/occurrences/find_occurrence_records.py

The diagram above illustrates the high-level flow of how the GBIF agent creates a query to search occurrence records from the user request

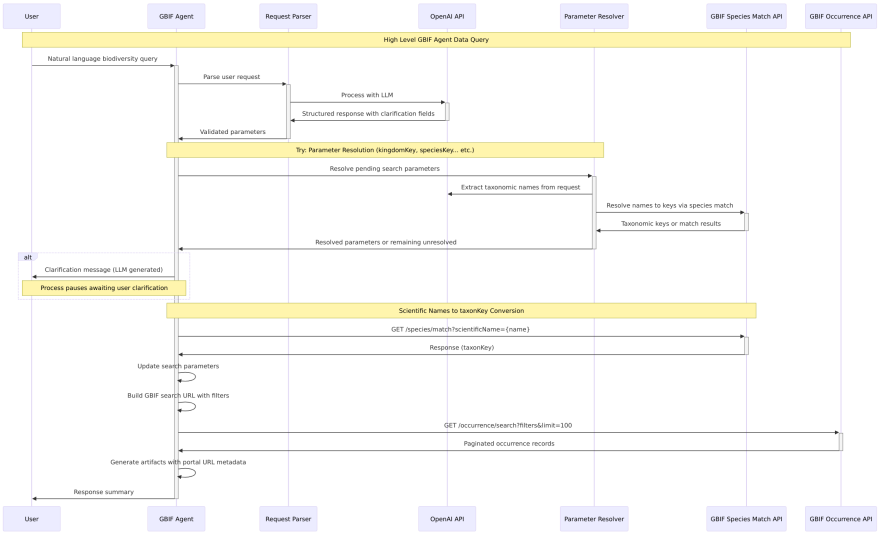

Figure 1: GBIF Agent Architecture