IATI Datastore version 3 API Contract

Published by the IATI Technical Team: 11 January 2022

Overview

This document defines the API contract for version 3 of the IATI Datastore. It sets out how the API will behave and acts as a contract between the IATI Technical Team (the API provider) and the developers that will use it.

As was the case with version 2 of the IATI Datastore, version 3 uses Apache Solr to index IATI data and provide search capabilities.

Also in line the version 2, version 3 features three distinct collections:

- The **Activity** collection, which holds the data in one document per IATI Activity element
- The **Transaction** collection, which explodes the data to include one document per Transaction element, and
- The **Budget** collection, which explodes the data to include one document per Budget element

New for Version 3 is the ability to return an Activity resultset as an IATI Activities XML document.

The Datastore v3 API contract will comprise a subset of the Solr Standard Query Parser and its Faceting Parameters. Whilst the chances are high that future versions of the Datastore will retain Solr as its search engine and for the majority of its API implementation, naturally technical environments, requirements and options constantly progress. Therefore it might be the case that a more appropriate alternative emerges for those futures. If this transpires, we will make every effort to continue to support this API contract beyond the use of Solr and ensure that any breaking change is subject to adequate deprecation procedures.

Regarding updates to Solr itself, the IATI Technical Team will assess Solr's release notes before updating. Any changes to the Solr API contract declared by Apache which in turn would break this API contract would necessitate a major version change to the Datastore.

In addition, we will make the full power of the Solr query APIs available for those that wish to use them - including, for example, the JSON APIs. We hope this will support a wider community of data users to query IATI data with greater flexibility. However, the IATI Technical Team will only commit to long term support of the API as defined in this contract, and so use of any other features must accept the possibility of change or removal at shorter notice.

The API contract is adapted from the Solr 8.10 API documentation and abbreviated to form a summary - for more detailed documentation on any of the parameters supported by the IATI Standard Query Parser please refer to the Solr 8.10 API documentation at https://solr.apache.org/guide/8_10/

Access and Security

The Datastore v3 API is accessible via the IATI API Gateway and follows the same conventions as established with the Validator v2 API.

The base URL will be api.iatistandard.org/datastore

As with the Validator API, API keys will be required and will be freely available via the IATI Developer Portal.

Traffic to the previous version of the Datastore API will be redirected to the v3 API without the requirement of an API Key for a period of six months from its launch date of 11 January 2022, to allow existing users time to migrate.

IATI Standard Query Parser

Parameters

q
Defines a query using standard query syntax. This parameter is mandatory.

q.op
Specifies the default operator for query expressions. Possible values are "AND" or "OR".

df
Specifies a default searchable field.

sow
Split on whitespace. If set to true, text analysis is invoked separately for each individual whitespace-separated term. The default is false.

sort
The sort parameter arranges search results in either ascending (asc) or descending (desc) order. The parameter can be used with either numerical or alphabetical content. The directions can be entered in either all lowercase or all uppercase letters (i.e., both asc and ASC are accepted).

Pagination

start
When specified, the start parameter specifies an offset into a query's result set and instructs Solr to begin displaying results from this offset.

rows
You can use the rows parameter to paginate results from a query. The parameter specifies the maximum number of documents from the complete result set that Solr should return to the client at one time.

The default value is 10. That is, by default, Solr returns 10 documents at a time in response to a query.

The hard limit is 1000. Should your response contain greater than 1000 results, you can use the **start** parameter to offset.

fq
The fq (Filter Query) parameter defines a query that can be used to restrict the superset of documents that can be returned, without influencing score. It can be very useful for speeding up complex queries, since the queries specified with fq are cached independently of the main query. When a later query uses the same filter, there's a cache hit, and filter results are returned quickly from the cache.

fl
The fl parameter limits the information included in a query response to a specified list of fields. The fields must be either stored="true" or docValues="true"`.`

The field list can be specified as a space-separated or comma-separated list of field names. The string "score" can be used to indicate that the score of each document for the particular query should be returned as a field.

The wildcard character * selects all the fields in the document which are either stored="true" or docValues="true" and useDocValuesAsStored="true" (which is the default when docValues are enabled). You can also add pseudo-fields, functions and transformers to the field list request.

timeAllowed

This parameter specifies the amount of time, in milliseconds, allowed for a search to complete. If this time expires before the search is complete, any partial results will be returned, but values such as numFound, facet counts, and result stats may not be accurate for the entire result set.

Response Formatting

wt
The wt parameter selects the format of the response. The accepted values are either *JSON*, *CSV,* or *XML* for all collections, and they return the standard Solr response from those response writers.

Additionally, for the Activity collection the API supports a custom response, *iati*. This returns an XML document containing each Activity in exactly the form it was originally published.

CSV Response Writer (wt=csv)

The CSV response writer returns a list of documents in comma-separated values (CSV) format. Other information that would normally be included in a response, such as facet information, is excluded.

CSV Parameters
These parameters specify the CSV format that will be returned. You can accept the default values or specify your own.

csv.encapsulator
encapsulator character for CSV response. default: “

csv.escape
escape character for CSV response. default: None

csv.separator
separator character for CSV response. default: ,

csv.header
Defaults to true. If false, Solr does not print the column headers

csv.newline
newline character for CSV response. default: \n

csv.null
Defaults to a zero length string. Use this parameter when a document has no value for a particular field.

Multi-Valued Field CSV Parameters

These parameters specify how multi-valued fields are encoded. Per-field overrides for these values can be done using f.<fieldname>.csv.separator=|

csv.mv.encapsulator
encapsulator character for multi-valued fields in CSV response. default: None

csv.mv.escape
escape character for multi-valued fields in CSV response. default: \

csv.mv.separator
separator character for multi-valued fields in CSV response. Defaults to the csv.separator value.

Grouping

group
If true, query results will be grouped.

group.field
The name of the field by which to group results. The field must be single-valued, and either be indexed or a field type that has a value source and works in a function query, such as ExternalFileField. It must also be a string-based field, such as StrField or TextField

As per the nature of the architecture of Solr as inherited from version 2, many fields are by necessity multi-valued. For example, an Activity can have many Transactions, and therefore in the Activity collection the *transaction* field is multi-valued, and as such can not be used to group by.

Faceting

facet
If set to true, this parameter enables facet counts in the query response. If set to false, a blank or missing value, this parameter disables faceting. None of the other parameters listed below will have any effect unless this parameter is set to true. The default value is blank (false).

facet.field
The facet.field parameter identifies a field that should be treated as a facet. It iterates over each Term in the field and generates a facet count using that Term as the constraint. This parameter can be specified multiple times in a query to select multiple facet fields.

> If you do not set this parameter to at least one field in the schema, none of the other facet parameters will have any effect.

facet.prefix

The facet.prefix parameter limits the terms on which to facet to those starting with the given string prefix. This does not limit the query in any way, only the facets that would be returned in response to the query.

facet.contains
The facet.contains parameter limits the terms on which to facet to those containing the given substring. This does not limit the query in any way, only the facets that would be returned in response to the query.

facet.contains.ignoreCase
If facet.contains is used, the facet.contains.ignoreCase parameter causes case to be ignored when matching the given substring against candidate facet terms.

facet.matches
If you want to only return facet buckets for the terms that match a regular expression.

facet.sort
This parameter determines the ordering of the facet field constraints.

There are two options for this parameter.

1. count
   Sort the constraints by count (highest count first).

2. index
   Return the constraints sorted in their index order (lexicographic by indexed term). For terms in the ASCII range, this will be alphabetically sorted.

The default is count if facet.limit is greater than 0, otherwise, the default is index.

facet.limit
This parameter specifies the maximum number of constraint counts (essentially, the number of facets for a field that are returned) that should be returned for the facet fields. A negative value means that Solr will return an unlimited number of constraint counts.
The default value is 100.

facet.offset
The facet.offset parameter indicates an offset into the list of constraints to allow paging.
The default value is 0.

facet.mincount
The facet.mincount parameter specifies the minimum counts required for a facet field to be included in the response. If a field's counts are below the minimum, the field's facet is not returned.
The default value is 0.

facet.missing
If set to true, this parameter indicates that, in addition to the Term-based constraints of a facet field, a count of all results that match the query but which have no facet value for the field should be computed and returned in the response.
The default value is false.

facet.method
The facet.method parameter selects the type of algorithm or method Solr should use when faceting a field.
The following methods are available.

> **enum**
> Enumerates all terms in a field, calculating the set intersection of documents that match the term with documents that match the query.
> This method is recommended for faceting multi-valued fields that have only a few distinct values.
>
> **fc** (default)
> Calculates facet counts by iterating over documents that match the query and summing the terms that appear in each document.

facet.exists
To cap facet counts by 1, specify facet.exists=true. This parameter can be used with facet.method=enum or when it's omitted. It can be used only on non-trie fields (such as strings). It may speed up facet counting on large indices and/or high-cardinality facet values.

facet.excludeTerms
If you want to remove terms from facet counts but keep them in the index, the facet.excludeTerms parameter allows you to do that.

facet.range
The facet.range parameter defines the field for which Solr should create range facets.

facet.range.start
Specifies the lower bound of the ranges.

facet.range.end
Specifies the upper bound of the ranges.

facet.range.gap
The span of each range expressed as a value to be added to the lower bound. You can specify this parameter on a per-field basis with the syntax of f.<fieldname>.facet.range.gap.

facet.range.other
The facet.range.other parameter specifies that in addition to the counts for each range constraint between facet.range.start and facet.range.end, counts should also be computed for these options:

- before: All records with field values lower than the lower bound of the first range.
- after: All records with field values greater than the upper bound of the last range.
- between: All records with field values between the start and end bounds of all ranges.
- none: Do not compute any counts.
- all: Compute counts for before, between, and after.

facet.pivot
Pivoting is a summarisation tool that lets you automatically sort, count, total or average data stored in a table. The facet.pivot parameter defines the fields to use for the pivot. Multiple facet.pivot values will create multiple "facet_pivot" sections in the response. Separate each list of fields with a comma.

facet.pivot.mincount
The facet.pivot.mincount parameter defines the minimum number of documents that need to match in order for the facet to be included in results. The default is 1.

Specifying Terms for the IATI Standard Query Parser

A query to the standard query parser is broken up into terms and operators. There are two types of terms: single terms and phrases.

- A single term is a single word such as "test" or "hello"
- A phrase is a group of words surrounded by double quotes such as "hello world"

Multiple terms can be combined together with Boolean operators to form more complex queries (as described below).

Wildcard Searches

Solr’s standard query parser supports single and multiple character wildcard searches within single terms. Wildcard characters can be applied to single terms, but not to search phrases.

Wildcard Search Type	Special Character	Example
Single character (matches a single character)	?	The search string te?t would match both test and text.
Multiple characters (matches zero or more sequential characters)	*	The wildcard search: tes* would match test, testing, and tester. You can also use wildcard characters in the middle of a term. For example: te*t would match test, text and teapot. *est would match pest and test.

Boolean Operators Supported by the Standard Query Parser

Boolean operators allow you to apply Boolean logic to queries, requiring the presence or absence of specific terms or conditions in fields in order to match documents. The table below summarises the Boolean operators supported by the standard query parser.

Boolean Operator	Alternative Symbol	Description
AND	&&	Requires both terms on either side of the Boolean operator to be present for a match.
NOT	!	Requires that the following term not be present.

OR	||	Requires that either term (or both terms) be present for a match.
	+	Requires that the following term be present.
	-	Prohibits the following term (that is, matches on fields or documents that do not include that term). The - operator is functionally similar to the Boolean operator !. Because it's used by popular search engines such as Google, it may be more familiar to some user communities.

Defining ranges

For either date or numeric fields you can match against a range of values, and use a wildcard ("*") to denote an unlimited upper or lower range.

For example:

```
transaction_value:[1000000 TO *]
```

would match all transaction values upwards of one million.

```
last_updated_datetime:[* TO 2021-11-01T00:00:00Z]
```

would match all last updated datetimes from the earliest to midnight of 1st November 2021. Solr is very strict in its datetime formats, and datetimes must be to that exact form.

Please note that Solr does not support greater or less than operators, so using ranges is the way to achieve such functionality.

Response Formats

As mentioned earlier, by using the *wt* parameter you can select the format in which you wish to receive your response.

Each of the Activity, Transactions and Budget collections support the standard Solr response writer json, xml and csv formats. CSV responses are returned in the order specified by the **fl** parameter. JSON and XML responses are unordered.

Additionally, the Activity collection supports a custom response format - iati. Use *iati* in the path instead of *select* (for example, /datastore/activity/iati?q=...) allows you to return each Activity in their original XML format, exactly as originally published, and contained within an *iati-activities* element.