DataONE Metrics Service Documentation

Release 0.0.2

DataONE

Oct 02, 2019

Contents:

1 Log Processing Architecture **3**
- 1.1 Component Diagram . . . 3

2 Metrics Service Class Diagram **5**
- 2.1 Class Diagram . . . 5

3 Example Elasticsearch Queries **7**
- 3.1 Metrics Service . . . 7
- 3.2 Event Processing . . . 9

4 Examples of Event Records **11**
- 4.1 Log tail . . . 11
- 4.2 Event Example . . . 13

5 Log Formatting **15**

6 Metrics Service Implementation **19**
- 6.1 Solr Event Processor . . . 20

7 Indices and tables **23**

Python Module Index **25**

Index **27**

The DataONE metrics service provides information about the use of content across the entire DataONE federation and makes those metrics available at various levels of detail. A high level overview of the infrastructure is provided in Figure 1.

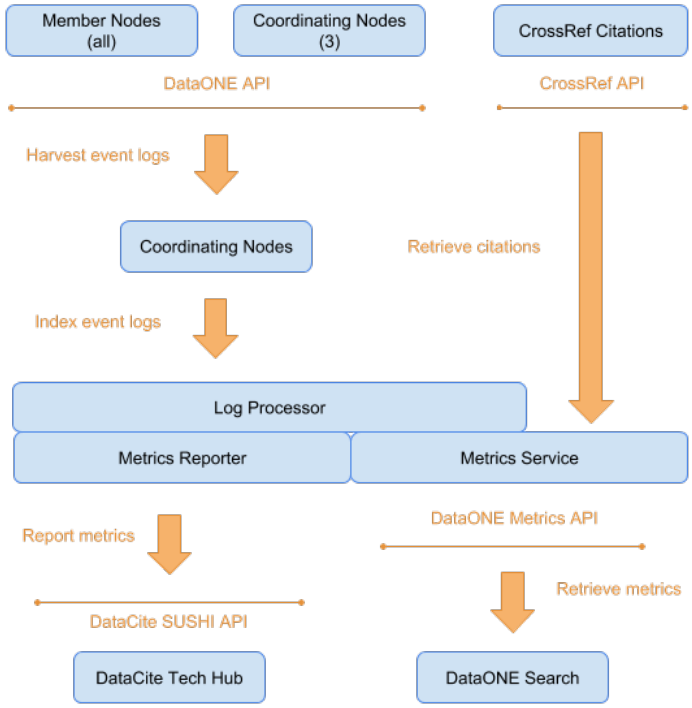

Figure 1. High level implementation of the DataONE Metrics Service.

Events such as READ activity are logged by Member (MN) and Coordinating (CN) Nodes and aggregated by the log aggregation service on the CNs. Further processing is performed done based on the COUNTER Code of Practice at the Log Processor. The Metrics Reporting service generates reports for the DataCite SUSHI API and sends them to the DataCite HUB. The DataONE Metrics API uses these processed logs to drive metrics display on the DataONE search interface.

CHAPTER 1

Log Processing Architecture

1.1 Component Diagram

General flow of event information in the DataONE system. Events occur at Member and Coordinating Nodes and are aggregated by the Log Aggregation process that runs within the `d1-processing` application. Events are written to both a solr index and to JSON format log files. The log files are monitored by `filebeat` and forwarded to a `logstash` service. The `logstash` service applies geocoding, robots identification, and user agent matching before adding to an `elasticseach` instance.

CHAPTER 2

Metrics Service Class Diagram

- The Falcon *d1_metrics_service* class is where the Falcon API creates a WSGI application and the Gunicorn server runs this application.
- Resource object is created from the class *MetricsReader* which is eventually mapped to the end-points into the *d1_metrics_service*.
- The JSON *MetricsRequest* object from the HTTP requests is passed on to the *MetricsReader* class for further processing.
- The *MetricsReader* class parses the *MetricsRequest* object and based on the filtering properties calls the appropriate method of the *MetricsElasticSearch* and the *MetricsDatabase* class from the *d1_metrics* package.
- The *MetricsElasticSearch* class has various methods to aggregate the metrics based on the filters specified in the *MetricsReader* class. The *MetricsReader* class forms a *MetricsResponse* object which is sent back to the client as HTTP Response.
- The *MetricsDatabase* class has methods that establishes connection and returns a cursor that could perform *CRUD* operations on the Citations database. This class is reponsible for gathering Citations for the DataONE objects from the Crossref Citations API and updating the metadata on timely basis.
- The *MetricsReporter* class queries *solr* to get the metadata for the generation of reports on a scheduled basis. These reports are send to the hub on month-to-month basis.

2.1 Class Diagram

CHAPTER 3

Example Elasticsearch Queries

The service can be accessed at https://logproc-stage-ucsb-1.test.dataone.org/app/kibana

Elasticsearch can be accessed locally by opening a ssh tunnel:

```
ssh -L9200:localhost:9200 logproc-stage-ucsb-1.test.dataone.org
```

3.1 Metrics Service

3.1.1 Landing Page Query Request

```
{
  "metrics": [
      "Citations",
      "Unique_Dataset_Requests",
      "Total_Dataset_Requests",
      "Total_Dataset_Investigations",
      "Unique_Dataset_Investigations"
    ],
    "filterBy": [
      {
        "filterType": "dataset",
        "values": [
          "Dataset002"
        ],
        "interpretAs": "list"
      }
    ],
    "groupBy": [
      "dataset"
    ]
}
```

3.1.2 User Profile Charts

```
{
  "metrics": [
      "Citations",
      "Unique_Dataset_Requests",
      "Total_Dataset_Requests",
      "Total_Dataset_Investigations",
      "Unique_Dataset_Investigations"
    ],
    "filterBy": [
      {
        "filterType": "uid",
        "values": [
          "User01"
        ],
        "interpretAs": "list"
      },
      {
        "filterType": "month",
        "values": [
          "2017-01",
          "2018-04"
        ],
        "interpretAs": "range"
      }
    ],
    "groupBy": [
      "uid", "month", "year"
    ]
}
```

3.1.3 User Profile Summary

```
{
  "metrics": [
      "Citations",
      "Unique_Dataset_Requests",
      "Total_Dataset_Requests",
      "Total_Dataset_Investigations",
      "Unique_Dataset_Investigations"
    ],
    "filterBy": [
      {
        "filterType": "uid",
        "values": [
          "User01"
        ],
        "interpretAs": "list"
      }
    ],
    "groupBy": [
      "uid"
    ]
}
```

3.2 Event Processing

3.2.1 Unprocessed events

```
GET eventlog-1/_search
{
 "query": {
   "bool": {
     "must": [
       {
         "term": {"event.key": "read"}
       }
     ],
     "must_not": [
       {
         "exists":{"field":"sessionId"}
       }
     ]
   }
 },
 "size":0
}
```

View in kibana

CHAPTER 4

Examples of Event Records

4.1 Log tail


```
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:TERN", "subject": "public
↪", "geohash_7": "9zvxyg9", "size": 51808, "readPermission": ["New South Wales␣
↪Office of Environment and Heritage", "New South Wales Office of Environment and␣
↪Heritage"], "location": "44.9759, -93.2166", "_version_": 1600860164592762887,
↪"country": "United States", "isPublic": false, "dateLogged": "2018-05-18T23:55:19.
↪29Z", "pid": "aekos.org.au/collection/nsw.gov.au/nsw_atlas/vis_flora_module/HCR_
↪VMP3.20150515", "formatId": "eml://ecoinformatics.org/eml-2.1.1", "id":
↪"urn:node:TERN.4154806", "event": "replicate", "geohash_6": "9zvxyg", "geohash_5":
↪"9zvxy", "geohash_3": "9zv", "geohash_9": "9zvxyg9ex", "geohash_8": "9zvxyg9e",
↪"formatType": "METADATA", "rightsHolder": "New South Wales Office of Environment␣
↪and Heritage", "dateAggregated": "2018-05-19T03:06:34.091Z", "userAgent": "pyd1/1.1.
↪11 +http://dataone.org/", "isRepeatVisit": false, "geohash_1": "9", "geohash_2": "9z
↪", "versionCompliance": "v1", "geohash_4": "9zvx", "city": "Minneapolis", "entryId
↪": "4154806", "region": "Minnesota", "ipAddress": "160.94.0.146", "inFullRobotList
↪": false, "inPartialRobotList": false}
{"nodeId": "urn:node:DRYAD", "geohash_5": "9q4gc", "inPartialRobotList": true,
↪"geohash_7": "9q4gch3", "size": 0, "dateUpdated": "1900-01-01T00:00:00Z", "location
↪": "34.4329, -119.8371", "_version_": 1600856433431150596, "country": "United States
↪", "isPublic": false, "dateLogged": "2018-05-18T23:55:23.024Z", "pid": "https:/doi.
↪org/10.5061/dryad.1215m?ver=2018-05-18T23:49:16.201+00:00", "geohash_2": "9q", "id
↪": "urn:node:DRYAD.1028656", "userAgent": "Apache-HttpClient/4.3.6 (java 1.5)",
↪"geohash_6": "9q4gch", "geohash_3": "9q4", "geohash_9": "9q4gch361", "geohash_8":
↪"9q4gch36", "event": "read", "dateAggregated": "2018-05-19T02:07:39.741Z",
↪"isRepeatVisit": false, "geohash_1": "9", "versionCompliance": "v1", "geohash_4":
↪"9q4g", "city": "Santa Barbara", "entryId": "1028656", "region": "California",
↪"ipAddress": "128.111.54.80", "inFullRobotList": true, "subject": "DC=dataone,␣
↪DC=org"}
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:TERN", "subject": "public
↪", "geohash_7": "9zvxyg9", "size": 109440, "readPermission": ["New South Wales␣
↪Office of Environment and Heritage", "New South Wales Office of Environment and␣
↪Heritage"], "location": "44.9759, -93.2166", "_version_": 1600860164592762888,
↪"country": "United States", "isPublic": false, "dateLogged": "2018-05-18T23:55:23.
↪943Z", "pid": "aekos.org.au/collection/nsw.gov.au/nsw_atlas/vis_flora_module/CMT.
↪20150515", "formatId": "eml://ecoinformatics.org/eml-2.1.1", "id": "urn:node:TERN.
↪4154807", "event": "replicate", "geohash_6": "9zvxyg", "geohash_5": "9zvxy",
↪"geohash_3": "9zv", "geohash_9": "9zvxyg9ex", "geohash_8": "9zvxyg9e", "formatType
↪": "METADATA", "rightsHolder": "New South Wales Office of Environment and Heritage",
↪ "dateAggregated": "2018-05-19T03:06:34.172Z", "userAgent": "pyd1/1.1.11 +http://
↪dataone.org/", "isRepeatVisit": false, "geohash_1": "9", "geohash_2": "9z"
```


(continues on next page)

(continued from previous page)


```
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:TERN", "subject": "public
↪", "geohash_7": "9zvxyg9", "size": 74153, "readPermission": ["New South Wales␣
↪Office of Environment and Heritage", "New South Wales Office of Environment and␣
↪Heritage"], "location": "44.9759, -93.2166", "_version_": 1600860164592762889,
↪"country": "United States", "isPublic": false, "dateLogged": "2018-05-18T23:55:28.
↪37Z", "pid": "aekos.org.au/collection/nsw.gov.au/nsw_atlas/vis_flora_module/BWNP.
↪20150515", "formatId": "eml://ecoinformatics.org/eml-2.1.1", "id": "urn:node:TERN.
↪4154808", "event": "replicate", "geohash_6": "9zvxyg", "geohash_5": "9zvxy",
↪"geohash_3": "9zv", "geohash_9": "9zvxyg9ex", "geohash_8": "9zvxyg9e", "formatType
↪": "METADATA", "rightsHolder": "New South Wales Office of Environment and Heritage",
↪ "dateAggregated": "2018-05-19T03:06:34.254Z", "userAgent": "pyd1/1.1.11 +http://
↪dataone.org/", "isRepeatVisit": false, "geohash_1": "9", "geohash_2": "9z",
↪"versionCompliance": "v1", "geohash_4": "9zvx", "city": "Minneapolis", "entryId":
↪"4154808", "region": "Minnesota", "ipAddress": "160.94.0.146", "inFullRobotList":␣
↪false, "inPartialRobotList": false}
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:TERN", "subject": "public
↪", "geohash_7": "9zvxyg9", "size": 24921, "readPermission": ["New South Wales␣
↪Office of Environment and Heritage", "New South Wales Office of Environment and␣
↪Heritage"], "location": "44.9759, -93.2166", "_version_": 1600860164592762890,
↪"country": "United States", "isPublic": false, "dateLogged": "2018-05-18T23:55:33.
↪103Z", "pid": "aekos.org.au/collection/nsw.gov.au/nsw_atlas/vis_flora_module/JTH_NP.
↪20150515", "formatId": "eml://ecoinformatics.org/eml-2.1.1", "id": "urn:node:TERN.
↪4154809", "event": "replicate", "geohash_6": "9zvxyg", "geohash_5": "9zvxy",
↪"geohash_3": "9zv", "geohash_9": "9zvxyg9ex", "geohash_8": "9zvxyg9e", "formatType
↪": "METADATA", "rightsHolder": "New South Wales Office of Environment and Heritage",
↪ "dateAggregated": "2018-05-19T03:06:34.286Z", "userAgent": "pyd1/1.1.11 +http://
↪dataone.org/", "isRepeatVisit": false, "geohash_1": "9", "geohash_2": "9z",
↪"versionCompliance": "v1", "geohash_4": "9zvx", "city": "Minneapolis", "entryId":
↪"4154809", "region": "Minnesota", "ipAddress": "160.94.0.146", "inFullRobotList":␣
↪false, "inPartialRobotList": false}
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:KNB", "subject":
↪"CN=urn:node:NMEPSCOR,DC=dataone,DC=org", "geohash_7": "9whpkee", "size": 15500059,
↪"readPermission": ["UID=jkbalch,O=unaffiliated,DC=ecoinformatics,DC=org"], "location
↪": "35.0443, -106.6729", "_version_": 1600852628511653897, "country": "United States
↪", "isPublic": false, "dateLogged": "2018-05-18T23:55:35.449Z", "pid": "jkbalch.16.1
↪", "formatId": "application/octet-stream", "id": "urn:node:KNB.14370028", "event":
↪"replicate", "geohash_6": "9whpke", "geohash_5": "9whpk", "geohash_3": "9wh",
↪"geohash_9": "9whpkee5v", "geohash_8": "9whpkee5", "formatType": "DATA",
↪"rightsHolder": "uid=jkbalch,o=unaffiliated,dc=ecoinformatics,dc=org",
↪"dateAggregated": "2018-05-19T01:07:01.379Z", "userAgent": "pyd1/1.1.11 +http://
↪dataone.org/", "isRepeatVisit": false, "geohash_1": "9", "geohash_2": "9w",
↪"versionCompliance": "v1", "geohash_4": "9whp", "city": "Albuquerque", "entryId":
↪"14370028", "region": "New Mexico", "ipAddress": "129.24.63.59", "inFullRobotList":␣
↪false, "inPartialRobotList": false}
{"nodeId": "urn:node:DRYAD", "geohash_5": "9q4gc", "inPartialRobotList": true,
↪"geohash_7": "9q4gch3", "size": 0, "dateUpdated": "1900-01-01T00:00:00Z", "location
↪": "34.4329, -119.8371", "_version_": 1600856433431150597, "country": "United States
↪", "isPublic": false, "dateLogged": "2018-05-18T23:55:37.342Z", "pid": "https:/doi.
↪org/10.5061/dryad.1215m?format=d1rem&ver=2018-05-18T23:49:16.201+00:00", "geohash_2
↪": "9q", "id": "urn:node:DRYAD.1028657", "userAgent": "Apache-HttpClient/4.3.6␣
↪(java 1.5)", "geohash_6": "9q4gch", "geohash_3": "9q4", "geohash_9": "9q4gch361",
↪"geohash_8": "9q4gch36", "event": "read", "dateAggregated": "2018-05-19T02:07:39.
↪869Z", "isRepeatVisit": false, "geohash_1": "9", "versionCompliance": "v1",
↪"geohash_4": "9q4g", "city": "Santa Barbara", "entryId": "1028657", "region":
↪"California", "ipAddress": "128.111.54.80", "inFullRobotList": true, "subject":
↪"DC=dataone, DC=org"}
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:TERN", "subject": "public
↪", "geohash_7": "9zvxyg9", "size": 32718, "readPermission": ["New South Wales␣
↪Office of Environment and Heritage", "New South Wales Office of Environment and␣
↪Heritage"], "location": "44.9759, -93.2166", "_version_": 1600860164592762891,
↪"country": "United States", "isPublic": false, "dateLogged": "2018-05-18T23:55:37.
↪695Z", "pid": "aekos.org.au/collection/nsw.gov.au/nsw_atlas/vis_flora_module/
↪NANGFLOR.20150515", "formatId": "eml://ecoinformatics.org/eml-2.1.1", "id":
↪"urn:node:TERN.4154810", "event": "replicate", "geohash_6": "9zvxyg", "geohash_5":
```


(continues on next page)

(continued from previous page)


```
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:KNB", "subject": "public",
↪ "geohash_7": "bdvs4ch", "size": 70427, "readPermission": ["CN=SASAP,DC=dataone,
↪DC=org", "CN=SASAP,DC=dataone,DC=org", "CN=arctic-data-admins,DC=dataone,DC=org"],
↪"location": "61.1777, -149.6384", "_version_": 1600852628511653898, "country":
↪"United States", "isPublic": false, "dateLogged": "2018-05-18T23:55:38.641Z", "pid
↪": "urn:uuid:a3c58bd6-481e-4c64-aa93-795df10a4664", "formatId": "text/csv", "id":
↪"urn:node:KNB.14370029", "event": "read", "geohash_6": "bdvs4c", "geohash_5": "bdvs4
↪", "geohash_3": "bdv", "geohash_9": "bdvs4ch9u", "geohash_8": "bdvs4ch9",
↪"formatType": "DATA", "rightsHolder": "CN=SASAP,DC=dataone,DC=org", "dateAggregated
↪": "2018-05-19T01:07:01.438Z", "userAgent": "R (3.5.0 x86_64-apple-darwin15.6.0 x86_
↪64 darwin15.6.0)", "isRepeatVisit": false, "geohash_1": "b", "geohash_2": "bd",
↪"versionCompliance": "v1", "geohash_4": "bdvs", "city": "Anchorage", "entryId":
↪"14370029", "region": "Alaska", "ipAddress": "12.17.185.172", "inFullRobotList":␣
↪false, "inPartialRobotList": false}
{"dateUpdated": "1900-01-01T00:00:00Z", "nodeId": "urn:node:KNB", "subject":
↪"CN=urn:node:NMEPSCOR,DC=dataone,DC=org", "geohash_7": "9whpkee", "size": 42662,
↪"readPermission": ["UID=rudstam,O=unaffiliated,DC=ecoinformatics,DC=org", "UID=gss1,
↪O=unaffiliated,DC=ecoinformatics,DC=org", "UID=kholeck,O=unaffiliated,
↪DC=ecoinformatics,DC=org", "UID=teb1,O=unaffiliated,DC=ecoinformatics,DC=org",
↪"UID=rudstam,O=unaffiliated,DC=ecoinformatics,DC=org"], "location": "35.0443, -106.
↪6729", "_version_": 1600852628512702464, "country": "United States", "isPublic":␣
↪false, "dateLogged": "2018-05-18T23:55:39.958Z", "pid": "cbfs.127.22", "formatId":
↪"eml://ecoinformatics.org/eml-2.1.0", "id": "urn:node:KNB.14370030", "event":
↪"replicate", "geohash_6": "9whpke", "geohash_5": "9whpk", "geohash_3": "9wh",
↪"geohash_9": "9whpkee5v", "geohash_8": "9whpkee5", "formatType": "METADATA",
↪"rightsHolder": "uid=rudstam,o=unaffiliated,dc=ecoinformatics,dc=org",
↪"dateAggregated": "2018-05-19T01:07:01.545Z", "userAgent": "pyd1/1.1.11 +http://
↪dataone.org/", "isRepeatVisit": false, "geohash_1": "9", "geohash_2": "9w",
↪"versionCompliance": "v1", "geohash_4": "9whp", "city": "Albuquerque", "entryId":
↪"14370030", "region": "New Mexico", "ipAddress": "129.24.63.59", "inFullRobotList":␣
↪false, "inPartialRobotList": false}
```


4.2 Event Example


```
{
  "dateUpdated": "1900-01-01T00:00:00Z",
  "nodeId": "urn:node:TERN",
  "subject": "public",
  "geohash_7": "9zvxyg9",
  "size": 51808,
  "readPermission": [
    "New South Wales Office of Environment and Heritage",
    "New South Wales Office of Environment and Heritage"
  ],
  "location": "44.9759, -93.2166",
  "_version_": 1600860164592763000,
  "country": "United States",
  "isPublic": false,
  "dateLogged": "2018-05-18T23:55:19.29Z",
  "pid": "aekos.org.au/collection/nsw.gov.au/nsw_atlas/vis_flora_module/HCR_VMP3.
↪20150515",
  "formatId": "eml://ecoinformatics.org/eml-2.1.1",
  "id": "urn:node:TERN.4154806",
  "event": "replicate",
```


(continues on next page)

(continued from previous page)


```
  "geohash_6": "9zvxyg",
  "geohash_5": "9zvxy",
  "geohash_3": "9zv",
  "geohash_9": "9zvxyg9ex",
  "geohash_8": "9zvxyg9e",
  "formatType": "METADATA",
  "rightsHolder": "New South Wales Office of Environment and Heritage",
  "dateAggregated": "2018-05-19T03:06:34.091Z",
  "userAgent": "pyd1/1.1.11 +http://dataone.org/",
  "isRepeatVisit": false,
  "geohash_1": "9",
  "geohash_2": "9z",
  "versionCompliance": "v1",
  "geohash_4": "9zvx",
  "city": "Minneapolis",
  "entryId": "4154806",
  "region": "Minnesota",
  "ipAddress": "160.94.0.146",
  "inFullRobotList": false,
  "inPartialRobotList": false
}
```

CHAPTER 5

Log Formatting

Apache2 log formatting for the search UI. The DataONE Search UI is a single page application served from an Apache web server. All requests issued by the Search UI are proxied back through the apache server, hence the logs for that server provide a good location for logging requests issued by the service.

Table 1: Apache search log properties

Property	Entry	Notes
ver	1.0	Version flag, indicating the revision of the log information. Must be incremented when format changes.
time	%{%Y-%m-%d}tT%{%T}t.%{msec_frac}tZ	Time that the request started
remoteIP	%a	IP address of the requestor
method	%m	HTTP method used for the request
request	%U	The request portion of the URL, i.e. after host and before query
query	%q	The query portion of the URL, i.e. after the "?"
userAgent	%{User-agent}i	Name of the client user agent
remoteUser	%u	Remote user identity, only if HTTP authentication used.
referer	%{Referer}i	URL of the page that requested the resource
status	%>s	HTTP status of the response
responseTime	%D	The time in microseconds taken by the server to respond

accessToken	%{Authorization}i	The accessToken can be decoded using pyjwt for example:
		`accessToken = "Bearer eyJhbGciOiJSUzI1NiJ ..."`
		`junk, token = accessToken.split(" ")`
		`print( jwt.decode(token, verify=False) )`
		`{'sub': 'http://orcid.org/0000-...',`
		`'fullName': 'Dave Vieglais',`
		`'issuedAt': '2018-10-06T12:28:44.156+00:00',`
		`'consumerKey': '...',`
		`'exp': 1538893724,`
		`'userId': 'http://orcid.org/0000-...',`
		`'ttl': 64800,`
		`'iat': 1538...}`

ga_cookie	%{_ga}C	The value of the Google Analytics _ga cookie. More details

Apache configuration for logging search events:

```
LogFormat "{ \"ver\":\"1.0\", \"time\":\"%{%Y-%m-%d}tT%{%T}t.%{msec_frac}tZ\", \
↪"remoteIP\":\"%a\",
            \"request\":\"%U\", \"query\":\"%q\", \"method\":\"%m\", \"status\":\"%>
↪s\",
            \"responseTime\":\"%T\", \"userAgent\":\"%{User-agent}i\",
            \"accessToken\":\"%{Authorization}i\", \"referer\":\"%{Referer}i\",
            \"remoteuser\":\"%u\", \"ga_cookie\":\"%{_ga}Ci\"}" searchstats
```

Example of output, reformatted for readability. Each log message appears as a single line in the log file.

```
{
  "ver": 1.0,
  "time": "2018-10-08T07:56:41.600Z",
  "remoteIP": "73.128.224.157",
  "request": "/cn/v2/meta/solson.18.1",
  "query": "",
  "method": "GET",
  "status": "200",
  "responseTime": "7504774",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36␣
↪(KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36",
  "accessToken": "-",
```

(continues on next page)

(continued from previous page)

```
  "referer": "https://search.dataone.org/view/doi:10.5063/F1HT2M7Q",
  "remoteUser": "-"
}
```

See also:

- Apache log files for general information on configuring Apache logs
- mod_log_config for log format options

CHAPTER 6

Metrics Service Implementation

Generic linux daemon base class for python 3.x.

class d1_metrics.daemon.**Daemon**(*pidfile*)

A generic daemon class.

Usage: subclass the daemon class and override the run() method.

daemonize()

Deamonize class. UNIX double fork mechanism.

delpid()

restart()

Restart the daemon.

run()

You should override this method when you subclass Daemon.

It will be called after the process has been daemonized by start() or restart().

start()

Start the daemon.

stop()

Stop the daemon.

class d1_metrics.solrclient.**SolrClient**(*base_url*, *core_name*, *select='/'*)

doGet(*params*)

getFieldValues(*name*, *q='*:*'*, *fq=None*, *maxvalues=-1*, *sort=True*, ***query_args*)

Retrieve the unique values for a field, along with their usage counts. :param sort: Sort the result :param name: Name of field for which to retrieve values :type name: string :param q: Query identifying the records from which values will be retrieved :type q: string :param fq: Filter query restricting operation of query :type fq: string :param maxvalues: Maximum number of values to retrieve. Default is -1,

which causes retrieval of all values.

Returns dict of {fieldname: [[value, count], ...], }

class d1_metrics.solrclient.**SolrSearchResponseIterator**(*base_url*, *core_name*, *q*, *select='select'*, *fq=None*, *fields='*'*, *page_size=10000*, *max_records=None*, *sort=None*, ***query_args*)

Performs a search against a Solr index and acts as an iterator to retrieve all the values.

process_row(*row*)

Override this method in derived classes to reformat the row response.

d1_metrics.solrclient.**escapeSolrQueryTerm**(*term*)

6.1 Solr Event Processor

Echo DataONE aggregated logs to disk

Requires python 3

This script reads records from the aggregated logs solr index and writes each record to a log file on disk, one record per line. Each line is formatted as:

JSON_DATA

where:

JSON_DATA = JSON representation of the record as retrieved from solr

Output log files are rotated based on size, with rotation scheduled at 1GB. A maximum of 150 log files are kept, so the log directory should not exceed about 150GB.

JSON loading benchmarks: http://artem.krylysov.com/blog/2015/09/29/benchmark-python-json-libraries/ Note performance difference under python3 are much reduced.

One particular challenge is that the dateLogged time in the log records has precision only to the second. This makes restarting the harvest challenging since there may be multiple records on the same second.

The strategy employed here is to retrieve the last set of n records (100 or so) and ignore any retrieved records that are present in the last set.

Each log record is on the order of 500-600 bytes, assume 1000bytes / record. The last 100 or so records would be the last 100k bytes of the log record.

class d1_logagg.eventprocessor.**LogFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)

converter()

timestamp[, tz] -> tz's local time from POSIX timestamp.

formatTime(*record*, *datefmt=None*)

Return the creation time of the specified LogRecord as formatted text.

This method should be called from format() by a formatter which wants to make use of a formatted time. This method can be overridden in formatters to provide for any specific requirement, but the basic behaviour is as follows: if datefmt (a string) is specified, it is used with time.strftime() to format the creation time of the record. Otherwise, an ISO8601-like (or RFC 3339-like) format is used. The resulting string is returned. This function uses a user-configurable function to convert the creation time to a tuple. By default, time.localtime() is used; to change this for a particular formatter instance, set the 'converter' attribute to a

function with the same signature as time.localtime() or time.gmtime(). To change it for all formatters, for example if you want all logging times to be shown in GMT, set the 'converter' attribute in the Formatter class.

class d1_logagg.eventprocessor.**OutputLogFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)

converter()
timestamp[, tz] -> tz's local time from POSIX timestamp.

formatTime(*record*, *datefmt=None*)
Return the creation time of the specified LogRecord as formatted text.

This method should be called from format() by a formatter which wants to make use of a formatted time. This method can be overridden in formatters to provide for any specific requirement, but the basic behaviour is as follows: if datefmt (a string) is specified, it is used with time.strftime() to format the creation time of the record. Otherwise, an ISO8601-like (or RFC 3339-like) format is used. The resulting string is returned. This function uses a user-configurable function to convert the creation time to a tuple. By default, time.localtime() is used; to change this for a particular formatter instance, set the 'converter' attribute to a function with the same signature as time.localtime() or time.gmtime(). To change it for all formatters, for example if you want all logging times to be shown in GMT, set the 'converter' attribute in the Formatter class.

class d1_logagg.eventprocessor.**SolrSearchResponseIterator**(*select_url*, *q*, *fq=None*, *fields='*'*, *page_size=10000*, *max_records=None*, *sort=None*, ***query_args*)
Performs a search against a Solr index and acts as an iterator to retrieve all the values.

process_row(*row*)
Override this method in derived classes to reformat the row response.

d1_logagg.eventprocessor.**escapeSolrQueryTerm**(*term*)

d1_logagg.eventprocessor.**getLastLinesFromFile**(*fname*, *seek_back=100000*, *pattern='^{'*, *lines_to_return=100*)
Returns the last lines matching pattern from the file fname

Args: fname: name of file to examine seek_back: number of bytes to look backwards in file pattern: Pattern lines must match to be returned lines_to_return: maximum number of lines to return

Returns: last n log entries that match pattern

d1_logagg.eventprocessor.**getOutputLogger**(*log_file*, *log_level=20*)
Logger used for emitting the solr records as JSON blobs, one record per line.

Only really using logger for this to take advantage of the file rotation capability.

Parameters

- **log_file** –
- **log_level** –

Returns

d1_logagg.eventprocessor.**getQuery**(*src_file='d1logagg.log'*, *tstamp=None*)
Returns a query that would retrieve the last entry in the log file

Args: src_file: name of the log file to examine tstamp: timestamp of last point for query. Defaults to the value of utcnow if not set

Returns: A solr query string that returns at least the last record from the index and the record data that was retrieved from the log

d1_logagg.eventprocessor.**getRecords**(*log_file_name*, *core_name*, *base_url='http://localhost:8983/solr'*, *test_only=False*)

Main method. Retrieve records from solr and save them to disk.

Args: log_file_name: Name of the destination log file core_name: Name of the solr core to query base_url: Base URL of the solr service

Returns: Nothing

d1_logagg.eventprocessor.**logRecordInList**(*record_list*, *record*)

Returns True if record is in record_list Args:

record_list: record:

Returns: Boolean

d1_logagg.eventprocessor.**main**()

d1_logagg.eventprocessor.**setupLogger**(*level=30*)

Logger used for application logging

Parameters **level** –

Returns

d1_logagg.eventprocessor.**trimLogEntries**(*records*, *field*)

Shrink the list so only entries that match the last record are returned Args:

records: records to test field: field to evaluate

Returns: records that match the field of the last record

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

d1_logagg.eventprocessor, 20
d1_metrics, 19
d1_metrics.daemon, 19
d1_metrics.solrclient, 19

Index

C

converter() (*d1_logagg.eventprocessor.LogFormatter method*), 20
converter() (*d1_logagg.eventprocessor.OutputLogFormatter method*), 21

D

d1_logagg.eventprocessor (*module*), 20
d1_metrics (*module*), 19
d1_metrics.daemon (*module*), 19
d1_metrics.solrclient (*module*), 19
Daemon (*class in d1_metrics.daemon*), 19
daemonize() (*d1_metrics.daemon.Daemon method*), 19
delpid() (*d1_metrics.daemon.Daemon method*), 19
doGet() (*d1_metrics.solrclient.SolrClient method*), 19

E

escapeSolrQueryTerm() (*in module d1_logagg.eventprocessor*), 21
escapeSolrQueryTerm() (*in module d1_metrics.solrclient*), 20

F

formatTime() (*d1_logagg.eventprocessor.LogFormatter method*), 20
formatTime() (*d1_logagg.eventprocessor.OutputLogFormatter method*), 21

G

getFieldValues() (*d1_metrics.solrclient.SolrClient method*), 19
getLastLinesFromFile() (*in module d1_logagg.eventprocessor*), 21
getOutputLogger() (*in module d1_logagg.eventprocessor*), 21
getQuery() (*in module d1_logagg.eventprocessor*), 21
getRecords() (*in module d1_logagg.eventprocessor*), 22

L

LogFormatter (*class in d1_logagg.eventprocessor*), 20
logRecordInList() (*in module d1_logagg.eventprocessor*), 22

M

main() (*in module d1_logagg.eventprocessor*), 22

O

OutputLogFormatter (*class in d1_logagg.eventprocessor*), 21

P

process_row() (*d1_logagg.eventprocessor.SolrSearchResponseIterator method*), 21
process_row() (*d1_metrics.solrclient.SolrSearchResponseIterator method*), 20

R

restart() (*d1_metrics.daemon.Daemon method*), 19
run() (*d1_metrics.daemon.Daemon method*), 19

S

setupLogger() (*in module d1_logagg.eventprocessor*), 22
SolrClient (*class in d1_metrics.solrclient*), 19
SolrSearchResponseIterator (*class in d1_logagg.eventprocessor*), 21
SolrSearchResponseIterator (*class in d1_metrics.solrclient*), 20
start() (*d1_metrics.daemon.Daemon method*), 19
stop() (*d1_metrics.daemon.Daemon method*), 19

T

trimLogEntries() (*in module d1_logagg.eventprocessor*), 22