MARKLOGIC DATA ANALYZER

Installation Instructions

Abstract

Installation of MarkLogic Data Analyzer

Biju George

biju.george@marklogic.com

Contents

Objective of Analyzer

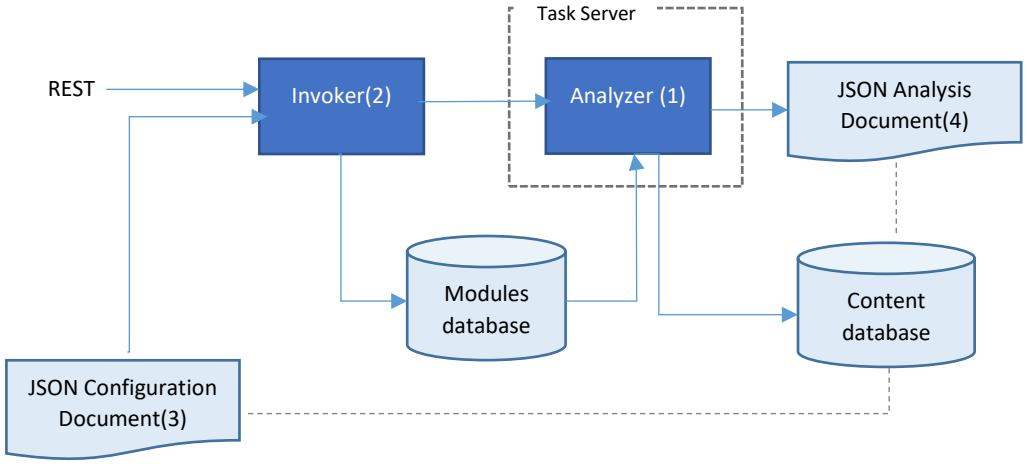
The MarkLogic Database Analyzer is a module that does in-database analysis of documents stored in MarkLogic and generates information required for Business Analysts for cataloging the data. The Analyzer runs as a background job in MarkLogic Task Server using MarkLogic resources.

Refer the section 'Limitations and Limits' for additional information. If sufficient resources cannot be provided for the Analyzer, then the document has to be

Components in the Analyzer

The Analyzer has 4 components

1. The main backend component which runs in the Task Server and generates the Analysis document. The backend component should be installed as a Javascript extension module.
2. The invoker component which invokes the backend component. The invoker component should be installed as a REST extension.
3. A JSON configuration file which is read by the invoker component. The JSON configuration document should be available in the database that need to be analyzed.
4. The JSON Analysis document which is the output document generated in the target content database that is analyzed.



Ways of invoking the Analyzer

The Analyzer can be invoked in two ways 1

1. Call the 'Invoker' through a REST call
2. Invoke the Analyzer through a Scheduled Task

Note: This version of the document explains only the REST option of the analyzer. The document will be updated with instructions on configuring scheduled tasks

Installation Instructions

For gradle based installation, gradle should be installed and gradle command (or gradlew) should be available to run from the command prompt

Installing Analyzer Module

Gradle

1. Unpack the gradle project
2. Update the `gradle.properties` for the below attributes
   `mlAppName`=*<How you want to call the app>*
   `mlHost`=*<The host where the analyzer would be run>*
   `mlUsername`=*<The user under which the installation will be done. Will require adequate permissions to install modules>*
   `mlPassword`=*<Password>*
   `mlRestPort`=*<The REST port of the app server, can be 8000 and will be installed in Modules database>*
   `mlRestAuthentication`=*<the authentication type of the server>*
   `mlModulesDatabaseName`=*<The modules database where the Analyzer will be installed>*
3. From the root project run '`gradle mlLoadModules`'
4. Verify the installation of the module in the modules database configured in `gradle.properties`. The installed module will have the uri `/analyzer/mlDatabaseAnalyzer.sjs`

Note: To change the URI of the analyzer, update the path relative to `<project-root>/src/main/ml-modules/root`

Example:

Location	**URI**
`<project-root>/src/main/ml-modules/root/analyzer/mlDatabaseAnalyzer.sjs`	`/analyzer/mlDatabaseAnalyzer.sjs`

<project-root>/src/main/ml-modules/root/abc/analyzer/mlDatabaseAnalyzer.sjs	/abc/analyzer/mlDatabaseAnalyzer.sjs
<project-root>/src/main/ml-modules/root/analyzer/abc/mlDatabaseAnalyzer.sjs	/analyzer/abc/mlDatabaseAnalyzer.sjs

REST API

For installing the module using REST,

1. Unpack the gradle project
2. Copy the content of `<project-root>/src/main/ml-modules/root/analyzer/mlDatabaseAnalyzer.sjs`
3. From your preferred REST Client (cUrl, Postman, SoapUI etc), call a PUT to `http(s)://<host>:<port>/LATEST/ext/mlAnalyze/mlDatabaseAnalyzer.sjs` with the content of `<root>/src/main/ml-modules/root/analyzer/mlDatabaseAnalyzer.sjs` as the BODY. Ensure that a header `Content-type: application/vnd.marklogic-javascript` is added.
4. On successful installation, 201 – Created or 204 - Updated will be received.
5. The module will be installed with the URI `/ext/mlAnalyze/mlDatabaseAnalyzer.sjs`

Note: The URI of the module can be updated by appropriately changing in the PUT call, relative to /ext

Location	URI
/LATEST/ext/mlAnalyze/mlDatabaseAnalyzer.sjs	/ext/mlAnalyze/mlDatabaseAnalyzer.sjs
/LATEST/ext/abc/mlAnalyze/mlDatabaseAnalyzer.sjs	/ext/abc/mlAnalyze/mlDatabaseAnalyzer.sjs
/LATEST/ext/abc/mlDatabaseAnalyzer.sjs	/ext/abc/mlDatabaseAnalyzer.sjs

Installing Invoker

Gradle

The same installation instructions for Analyzer module apply for Invoker also. The Invoker source is in `<project-root>/ \src\main\ml-modules\services`. '`gradle mlLoadModules`' installs the resource extension by the name the source file '`InvokeAnalyzer.sjs`' (minus extension).
For a different name for the extension, just rename the file.

REST API

The Invoker can also be installed via REST API.

1. Copy the content of `<project-root>\src\main\ml-modules\services\InvokeAnalyzer.sjs`

2. From your preferred REST Client (cUrl, Postman, SoapUI etc), call a PUT to `http(s)://<host>:<port>/LATEST/config/resources/InvokeAnalyzer`. Ensure that a header `Content-type: application/vnd.marklogic-javascript` is added.
3. On successful installation, 201 – Created or 204 - Updated will be received.

Configuring the Analyzer

Configuration

The Analyzer reads a json configuration file of below structure. This configuration file is expected in the content database that is being analyzed. The URI and collection of the configuration file can be chosen by the administrator (or whoever configures). The URI should be provided to the user who is calling the Invoker.

Attribute	Mandatory	Purpose	Example
collections	No	The documents that will be analyzed will be from these collections only. Should be provided as comma separated values. The collections configured here and the collections configured while calling the invoker are combined for the final document search.	"collections" : "Order,Supplies"
docCount	Yes	The number of sample URIs to be returned for each unique path. Can be zero if the URIs are not to be returned.	"docCount" : 2
outputCollections	Yes	The collections where the output analysis documents should be stored. Can be comma separated values for specifying multiple collections	"outputCollections" : "analysis, analysis-output"
limit	No	The number of random documents to be sampled.	"limit" : 2000
query	No	The serialized cts query that should be applied to extract the documents. The configured cts query is 'ANDed' with configured collections to generate the final query.	"query": { "andQuery": { "queries": [{ "jsonPropertyValueQuery": { "property": ["stateCode"], "value": ["IL"], "options": ["lang=en"] } }]

			} }
uriPrefix	Yes	The prefix of the URI of the output analysis document	"uriPrefix": "/ml/abc/analysis/data-hub-FINAL-"

Attention ! The Analyzer configuration is a document so that invoker (typically generated by a non-MarkLogic knowledgeable system) does not need to understand the MarkLogic query syntax. If the query used in the configuration is dynamic (ex. between two timestamps), then generation of the document should be synchronized with the invoker. The preferred approach will be to have a static configuration document that constraints the output. Ex. Documents updated in last 24 hrs instead of documents updated between two timestamps.

Example configuration

```
{
    "taskConfig": {
        "limit": 2,
        "collections": "Order",
        "docCount": 2,
        "outputCollections": "analysis,analysis-output",
        "query": {
            "andQuery": {
                "queries": [
                    {
                        "jsonPropertyValueQuery": {
                            "property": [
                                "stateCode"
                            ],
                            "value": [
                                "IL"
                            ],
                            "options": [
                                "lang=en"
                            ]
                        }
                    }
                ]
            }
        },
```

```
        "uriPrefix": "/ml/abc/analysis/data-hub-FINAL-"
    }
}
```

Running the Invoker and Analyzer

The Analyzer can be invoked directly only for debugging/troubleshooting purposes. In all other cases, Analyzer should be invoked from Invoker. The Invoker can be invoked using a REST POST call to

`http(s):<domain>:<port>//LATEST/resources/InvokeAnalyzer` with a JSON document in the BODY as described in below section. If successfully invoked, a response like below will be returned.

```
{"result": {
   "status": 201,
   "message": "Task Submitted",
   "analysisUri": "/ml/abc/analysis/data-hub-FINAL-15716293509463080.json"
}}
```

The `analysisUri` will have the URI of the analysis document that will be generated on completing the analysis. The calling application should wait for the URI to be available.

Configuring the Invoker

Attribute	Mandatory	Purpose	Example
`collections`	No	The documents that will be analyzed will be from these collections only. Should be provided as comma separated values. The collections configured here and the collections configured while calling the invoker are combined for the final document search.	See example document below
`configUri`	Yes	The URI of the configuration document. The `configUri` is expected to be available in `contentDb`	See example document below
`analyzerModule`	Yes	The URI of the Analyzer module	See example document below
`contentDb`	Yes	The database targeted for analysis. The `configUri` is expected to be available in `contentDb`	See example document below
`modulesDb`	Yes	The modules database where the `analyzerModule` is installed	See example document below
`user`	Yes	The user under which the Invoker is run. Appropriate permissions are required in the `contentDb` to run the `analyzerModule`.	See example document below

Example

```
{
    "config": {
       "collections" : "sales",
        "configUri": "/analysis/config/task-config.json",
        "analyzerModule": "/ext/mlAnalyze/mlDatabaseAnalyzer.sjs",
        "contentDb": "ml9-unit-test-demo-1",
        "modulesDb": "abc-hub-MODULES",
        "user": "admin"
    }
}
```

Understanding the Analysis document

The analysis document has 4 main elements which are described below

`analysis`	This is the main part where the analysis information is available. *paths* in array of *path* and *uris* and *collections* `"paths": [ { "path": "/shipTo/text('orderNumber')", "uris": [ { "uri": "/sales/shipment.json", "collections": [ "my-sales" ] }, { "uri": "/sales/shipment1.json", "collections": [ "my-sales" ] } ] } ]` Note that the Analyzer only analyzes xml and json documents.
`stats`	This section provides information about the size of documents analyzed.
`typeStats`	This section provides the count information of the documents by document type.
`audit`	This section provides the user, time taken and the actual query used for analysis. I also provides the configuration that was used for generating the analysis.

Limitations and Limits

The following limitations and limit are worth knowing while running the Analyzer

1. The Analyzer identifies only the leaf nodes (nodes not having any children) in documents that can be represented as trees (JSON, XML). For cataloging purposes only leaf nodes should be sufficient.
2. The Analyzer produces only structural information about the documents and do not bring in any values in the documents. In other words, the Analyzer as it is never results in a data silo.
3. The Analyzer is run in the MarkLogic Task Server. The Task Server has to be configured for a higher maximum timeout based on the volume of documents to be analyzed.
4. The Task Server by design does not recover from a failed task or does not continue from a point of failure.
5. The Analyzer consumes MarkLogic resources to complete the analysis and is a heavy task. The Analyzer has to be run in a dedicated host (if available with appropriate permissions to the content database) or should be run at non-business hours to avoid a competition for resources.
6. The optimal way to use the Analyzer to configure the *query* in the configuration document to select a reasonable size of documents to be analyzed. Some options to consider are
    a. Use *'limit'* option to limit the selection of documents to a sample rather than a full set of documents
    b. Use queries with constraints like *'all documents updated in last 24 hours'* , *'all documents in X collections'* only
7. The paths in analysis section of the output documents are generated only for JSON and xml documents which have tree structures. Binary and text documents are not analyzed. However, the typeStats section does include all the type of documents.
8. This version of the analyzer does not analyze attributes in XML documents. If there are attributes in XML documents, they are ignored.
9. This version of the analyzer does not analyze in the context of name spaces. This means, if there are same attributes in a xml, but in different context of name spaces, they will not appear separately in the output analysis document. And sample URIs will not be generated for the paths. For example, a document like below with two name spaces

```
<?xml  version="1.0" encoding="UTF-8"?>
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
<h:tr>
<h:td>Apples</h:td>
<h:td>Bananas</h:td>
</h:tr>
</h:table>
<f:table xmlns:f="https://www.w3schools.com/furniture">
<f:name>African Coffee Table</f:name>
<f:width>80</f:width>
```

```
            <f:length>120</f:length>
            </f:table>
            </root>
```

will identify below 4 unique paths

```
            /root/*:table/*:tr/*:td
            /root/*:table/*:name
            /root/*:table/*:width
            /root/*:table/*:length
```

If there is another document like below without namespace,

```
<?xml  version="1.0" encoding="UTF-8"?>
<root>
<table>
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
<table>
<name>African Coffee Table</name>
<width>80</width>
<length>120</length>
</table>
</root>
```

Below unique paths will be identified

```
            /root/table/tr/td
            /root/table/name
            /root/table/width
            /root/table/length
```

Troubleshooting

Analyzer

- When the Invoker successfully submit the Analyzer task, there will be an entry as below in the `TaskServer_ErrorLog.txt`

```
2019-10-20 22:41:48.871 Info: ---mlDatabaseAnalyzer::Starting DB Analysis with
configuration::{"limit":1,"collections":"undefined,my-
sales","docCount":2,"outputCollections":"analysis,analysis-
output","query":{"andQuery":{"queries":[{"jsonPropertyValueQuery":{"property":["stateCode"],"value":[
"WI"],"options":["lang=en"]}}]}},"uriPrefix":"/ml/abc/analysis/data-hub-FINAL-
","returnUri":"/ml/abc/analysis/data-hub-FINAL-15716293088584640.json"}
```

- When the Analyzer is completed successfully, there would be an entry in `TaskServer_ErrorLog.txt`

```
2019-10-20 22:41:48.890 Info: ---mlDatabaseAnalyzer::Completed DB Analysis.Analysis URI
is::/ml/abc/analysis/data-hub-FINAL-15716293088584640.json in PT0.023859S
```

- If the Analyzer times out or errors out, the error will be in `TaskServer_ErrorLog.txt`. Using the Error Code, the normal Marklogic troubleshooting can be performed by the MarkLogic administrator.

Invoker

- If the Invoker cannot submit the Analyzer job, then response to the REST API call will have the required error details. For example, if the configuration document does not exist or a non-existing (or not allowed by permissions) content or module database are provided, then Error 500 will be returned by the Invoker. Using the Error Code, the normal Marklogic troubleshooting can be performed by the MarkLogic administrator.

Examples

The examples are only illustrations on how to use the analyzer with different configurations and does not provide a real production scenario

Documents setup

uri: /person/person.json

```
{"Person": {
  "Name": {
    "first-name": "Sam",
    "last-name": "Jose"
  }
 }}
```

uri: /person/person1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<person> <name>
  <first-name>Sam</first-name>
  <last-name>Peter</last-name>
 </name></person>
```

uri: /person/person.txt

```
first-name|last-name
Sam|Keller
```

Example 1:

Requirement: Analyze all documents in *Person* Collection, generate two sample uris per attribute and store the analysis in collections 'analysis'. Use the configuration file `/analysis/config/task-config.json` in database `ml9-unit-test-demo-1` using `/analyzer/mlDatabaseAnalyzer.sjs` in `abc-hub-MODULES`

```
{
    "taskConfig": {
        "collections": "Person",
        "docCount": 2,
        "outputCollections": "analysis",
        "query": {
            "andQuery": {
                "queries": [
                    {
                        "trueQuery": {}
                    }
                ]
            }
        },
        "uriPrefix": "/ml/abc/analysis/data-hub-FINAL-"
    }
`
```

```
{
  "config": {
    "configUri": "/analysis/config/task-config.json",
    "analyzerModule": "/analyzer/mlDatabaseAnalyzer.sjs",
    "contentDb": "ml9-unit-test-demo-1",
    "modulesDb": "abc-hub-MODULES",
    "user": "analyzer-admin"
    }
}
```

Example 2:

Requirement: Analyze randomly selected one document in *Person* Collection, do not generate sample uris per attribute and store the analysis in collections 'analysis'. Use the configuration file `/analysis/config/task-config.json` in database `ml9-unit-test-demo-1` using `/analyzer/mlDatabaseAnalyzer.sjs` in `abc-hub-MODULES`

```
{
    "taskConfig": {
        "limit" : 1,
        "collections": "Person",
        "docCount": 0,
        "outputCollections": "analysis",
        "query": {
            "andQuery": {
                "queries": [
                    {
                        "trueQuery": {}
                    }
                ]
            }
        },
        "uriPrefix": "/ml/abc/analysis/data-hub-FINAL-"
    }
}
```

```
{
  "config": {
    "configUri": "/analysis/config/task-config.json",
    "analyzerModule": "/analyzer/mlDatabaseAnalyzer.sjs",
    "contentDb": "ml9-unit-test-demo-1",
    "modulesDb": "abc-hub-MODULES",
    "user": "analyzer-admin"
    }
}
```

Example 3:

Requirement: Analyze all documents in Person Collection with attribute last-name as 'Peter' and generate 1 sample uris per attribute. Store the analysis in collections 'analysis'. Use the configuration file `/analysis/config/task-config.json` in database `ml9-unit-test-demo-1` using `/analyzer/mlDatabaseAnalyzer.sjs` in `abc-hub-MODULES`

```
{
    "taskConfig": {
        "collections": "Person",
        "docCount": 1,
        "outputCollections": "analysis,analysis-output",
        "query": {
            "andQuery": {
                "queries": [
                    {
```

```
                        "jsonPropertyValueQuery": {
                            "property": [
                                "last-name"
                            ],
                            "value": [
                                "Peter"
                            ],
                            "options": [
                                "lang=en"
                            ]
                        }
                    }
                ]
            }
        },
        "uriPrefix": "/ml/abc/analysis/data-hub-FINAL-"
    }
}
```

References:

Below information will be useful in configuring the Analyzer.

ml-gradle : https://github.com/marklogic-community/ml-gradle

Serializing cts queries: https://docs.marklogic.com/guide/search-dev/cts_query#id_29308

==========