DigitalEdge™

Search API Guide

Version 1.2.1

July 2014

leidos

DISCLAIMER OF WARRANTY AND LIMITATION OF LIABILITY

The Software accompanying this Documentation is provided with the Limited Warranty contained in the License Agreement for that Software. Leidos, its affiliates and suppliers, disclaim all warranties that the Software will perform as expected or desired on any machine or in any environment. Leidos, its affiliates and suppliers, further disclaim any warranties that this Documentation is complete, accurate, or error-free. Both the Software and the Documentation are subject to updates or changes at Leidos' sole discretion. LEIDOS, ITS LICENSORS AND SUPPLIERS MAKE NO OTHER WARRANTIES, WRITTEN OR ORAL, EXPRESS OR IMPLIED RELATING TO THE PRODUCTS, SOFTWARE, AND DOCUMENTATION. LEIDOS, ITS LICENSORS AND SUPPLIERS DISCLAIM ALL IMPLIED WARRANTIES, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, USE, TITLE, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. In no event shall Leidos, its affiliates or suppliers, be liable to the End User for any consequential, incidental, indirect, exemplary, punitive, or special damages (including lost profits, lost data, or cost of substitute goods or services) related to or arising out of the use of this Software and Documentation however caused and whether such damages are based in tort (including negligence), contract, or otherwise, and regardless of whether Leidos, its affiliates or suppliers, has been advised of the possibility of such damages in advance. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAWS, END USER ACKNOWLEDGES AND AGREES THAT LEIDOS AND ITS AFFILIATES AND SUPPLIERS IN NO EVENT SHALL BE RESPONSIBLE OR LIABLE TO THE END USER FOR ANY AMOUNTS IN EXCESS OF THE FEES PAID BY THE END USER TO LEIDOS. LEIDOS SHALL NOT BE RESPONSIBLE FOR ANY MATTER BEYOND ITS REASONABLE CONTROL.

LEIDOS PROPRIETARY INFORMATION

This document contains Leidos Proprietary Information. It may be used by recipient only for the purpose for which it was transmitted and will be returned or destroyed upon request or when no longer needed by recipient. It may not be copied or communicated without the advance written consent of Leidos. This document contains trade secrets and commercial or financial information which are privileged and confidential and exempt from disclosure under the Freedom of Information Act, 5 U.S.C. § 552.

TRADEMARKS AND ACKNOWLEDGMENTS

Private installations of DigitalEdge are powered by Eucalyptus®.

Public cloud installations of DigitalEdge are powered by Amazon Web Services™.

The following list includes all trademarks that are referenced throughout the DigitalEdge documentation suite.

Adobe, Flash, PDF, and Shockwave are either registered trademarks or trademarks of Adobe Systems Incorporate in the United States and/or other countries.

Amazon Web Services, AWS, Amazon Elastic Compute Cloud, Amazon EC2, EC2, Amazon Simple Storage Service, Amazon S3, Amazon VPC, Amazon DynamoDB, Amazon Route 53, the "Powered by Amazon Web Services" logo, are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.

Apache, Archiva, Cassandra, Hadoop, Hive, HBase, Hue, Lucene, Maven, Apache Phoenix, Solr, Zoie, ActiveMQ are all trademarks of The Apache Software Foundation.

ArcSight is a registered trademark of ArcSight, Inc.

CAS is copyright 2007, JA-SIG, Inc.

CentOS is a trademark of the CentOS Project.

Cloudera is a registered trademark of Cloudera, Inc.

CloudShield is a registered trademark of CloudShield Technologies, Inc. in the U.S. and/or other countries.

CTools are open-source tools produced and managed by Webdetails Consulting Company in Portugal.

Drupal is a registered trademark of Dries Buytaert.

Elasticsearch is a trademark of Elasticsearch BV, registered in the U.S. and in other countries.

Eucalyptus and Walrus are registered trademarks of Eucalyptus Systems, Inc.

Firefox is a registered trademark of the Mozilla Foundation.

The Groovy programming language is sustained and led by SpringSource and the Groovy Community.

H2 is available under a modified version of the Mozilla Public License and under the unmodified Eclipse Public License.

Hybridfox is developed and maintained by CSS Corp R&D Labs.

JUnit is available under the terms of the Common Public License v 1.0.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft, Windows, and Word are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

MongoDB and Mongo are registered trademarks of 10gen, Inc.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Pentaho is a registered trademark of Pentaho, Inc.

PostgreSQL is a trademark of The PostgreSQL Global Development Group, in the US and other countries.

PuTTY is copyright 1997-2012 Simon Tatham.

Sonatype Nexus is a trademark of Sonatype, Inc.

Tableau Software and Tableau are registered trademarks of Tableau Software, Inc.

Twitter is a registered trademark of Twitter, Inc.

All other trademarks are the property of their respective owners.

CONTACT INFORMATION

Leidos Franklin Center
6841 Benjamin Franklin Drive
Columbia, Maryland 21046

Email: DigitalEdgeSupport@Leidos.com

DigitalEdge Technical Support: 443-367-7770

DigitalEdge Sales Support: 443-367-7800

To submit ideas or feedback: https://www9.v1ideas.com/digitaledge/welcome

Contents

Introduction

DigitalEdge offers the ability to search and retrieve data from a Lucene platform in near real time. Implementing search can be done several ways to interact with Lucene:

- Using the DigitalEdge Search App (see the DigitalEdge *Configuration* or *Operations* documentation). The Search app operates on top of the Lucene index and uses the Search API to access and query the index.
- Using code which creates and sends an `HTTP GET` command (see See "Sample Code" on page 5); or using a web browser with the appropriate URL

The Search API provides search services to all these options. It performs distributed searches, which means that multiple Lucene nodes can be searched at the same time. This is accomplished by using the Solr distributed search functionality.

You can access the Search API on the webapps.main machine at: `http://default.<system_name>/searchapi`

The Lucene index is located on the datasink.lucene machine in `/mnt/rdafs/lucene/index`.

Product documentation

DigitalEdge is a complex big data platform. The system comes with a complete set of documentation in PDF and HTML5 formats to help you master DigitalEdge:

Document	Use	Audience
Overview Guide	Basic information about the DigitalEdge platform, including architecture, concepts, and terminology; a must-read before working with any aspect of DigitalEdge	Anyone working with DigitalEdge in any capacity
Configuration Guide	Instructions for defining data models and building processing pipelines	Data Specialists, DigitalEdge Administrators
Operations Guide	Daily administration information, covering monitoring, managing, and modifying the platform	DigitalEdge Administrators
DigitalEdge SDK Guide	Reference for building custom plug-in components	Developers
Alerts API Guide	Reference for specifying data triggers and notifications for an alerting capability	Developers
Search API Guide	Reference for providing search services on a Lucene data sink node	Developers

Typographical conventions

The following style conventions are used throughout this documentation:

Type of Information	Style in Documentation
Code, commands, filenames	`code`
Cross references	Click to see this topic
Emphasis	*important point*
Hyperlinks	Click to go to this site
Notes, warnings, tips	
References to other documents	*Document Title*
Sample code blocks	`code`
Troubleshooting issue or problem	
Troubleshooting solution	
User input	*Italics*
User interface labels and controls	**Bold**
Variables	<*change-this-name*>

What the API Includes

The Search API includes the following components:

- Java library
- Javadoc
- Sample code

Using the API

The API is accessible via an HTTP call to the external Lucene process. The parameters in the GET command determine how the response is configured. For example, if the webapps.main node uses the public DNS entry of:

```
http://ec2-107-22-137-229.compute-1.amazonaws.com
```

the URL required to search for the word *Houston* would be:

```
http://ec2-107-22-137-229.compute-
1.amazonaws.com/searchapi/nearrealtime?q=Houston
```

However, appending `+Space` to the end will make the API return matches for both *Space* and *Houston*.

By default, only the top ten matches will be returned. You can change the default by appending a rows parameter, such as: `&rows=100` to the end of the URL.

In sum, here is the URL that is needed to search for *Houston* and *Space*, and to return the top 100 matches:

```
http://ec2-107-22-137-229.compute-
1.amazonaws.com/searchapi/nearrealtime?q=Houston+Space&rows=100
```

There are many other modifications which can be made to the search parameters. The solr wiki contains a list of available parameters and the correct syntax.

The Search API resides on the webapps.main node and searches data stored in the datasink.lucene group on the Lucene node.

Sample Code

The sample code creates an HTTP GET command, and sends it to the node running the external Lucene process. It will store the value of the response in a `response` object and display the results of the query on your console. This code is useful because it demonstrates one of the three primary ways to use the Search API.

This example shows how to make a request to the Search API on a non-secure port. A secure request requires the appropriate security artifacts.

SearchExample.java

```
package com.saic.rtws.webapp.searchapi.example;

import static org.junit.Assert.fail;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpException;
import org.apache.commons.httpclient.HttpMethod;
import org.apache.commons.httpclient.methods.GetMethod;
import org.junit.After;
import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;
import org.springframework.http.HttpStatus;

public class SearchExample {

                @Before
                public void setUp() throws Exception { }

                @After
                public void tearDown() throws Exception { }

                // Remove the Ignore annotation to run as a junit test
                @Ignore("for example uses")
                @Test
                public void testInternalSearchApi() {
                                String scheme = "http";
                                String host = "54.205.109.174";             // internal searchapi host / ip address
                                String port = "80";                        // internal searchapi port
                                String path = "searchapi/select";         // internal searchapi resource path
                                String params = "q=*:*&wt=json";         // Parameters for the query
```

```
                        // Construct the location of the internal Lucene URL
                        String url = String.format("%s://%s:%s/%s?%s", scheme, host, port, path, params);

                        StringBuilder response = new StringBuilder();

                        BufferedReader br = null;

                        try {
                                // Construct a http client and a get request
                                HttpMethod get = new GetMethod(url);
                                HttpClient client = new HttpClient();

                                // Submit the searchapi get request to the server
                                int statusCode = client.executeMethod(get);

                                // Search was successfully a 200 response is returned
                                // !200 response is considered a failure
                                if (HttpStatus.valueOf(statusCode) == HttpStatus.OK) {
                                        br = new BufferedReader(new InputStreamReader
                                             (get.getResponseBodyAsStream()));

                                        // Parse results and output it to standard out
                                        String line = null;
                                        while ((line = br.readLine()) != null) {
                                                response.append(line);
                                        }
                                        System.out.println("Response: " + response.toString());
                                } else {
                                        fail("SearchApi returned a status of " + statusCode + ".");
                                }
                        } catch (HttpException e) {
                                fail("Exception thrown: " + e.getMessage());
                        } catch (IOException e) {
                                fail("Exception thrown: " + e.getMessage());
                        } finally {
                                if (br != null) {
                                        try { br.close(); } catch (Exception ignore) { }
                                }
                        }
                }
```

```
}
```

How to Run the Sample

This sample runs on one Lucene shard and processes one search statement. To run the sample code:

1. Verify that you have a system up and running with a Lucene data sink and the SearchAPI tools.
2. Open `SearchExample.java`.
   a. Comment out the `@Ignore` annotation.
   b. Change the `String host` and `String port` parameters for your site.
   c. Replace the `params` variable with whatever you are searching for, and add any other parameters as needed.
   d. Save the file.
3. Right-click the file and select **Run As/JUnit Test**.

   Or, use the following commands to run the sample using Java:

```
cd <my-project-path>
```

   Build your project which contains the `SearchExample.java` file.

```
java -cp.;<junit-jar-path>;<other-jarpath> org.junit.runner.JUnitCore
                        <package>.SearchExample
```

Programming and Testing Environment

To test a webapp in-line with DigitalEdge, the webapp must be installed with your system. Currently, a webapp can only be integrated with your DigitalEdge system by an DigitalEdge Engineer.

Tips and tricks

- If you are using Amazon EC2™, the public DNS can be modified through Amazon Route 53™ to make it shorter and more practical.

Other resources

You can find more detailed information about the Search API in the following resources:

- Javadoc: `webapps-searchapi/doc`