

Endeca® Latitude

Administrator's Guide

Version 2.1.0 • June 2011



Contents

Preface.....	7
About this guide.....	7
Who should use this guide.....	7
Conventions used in this guide.....	8
Contacting Endeca Customer Support.....	8
Chapter 1: Introduction.....	9
Taking ownership of your Latitude implementation.....	9
Overview of administrator tasks.....	9
Chapter 2: Using the Administration Web Service.....	11
About the Administration Web Service.....	11
Accessing the Administration Web Service.....	12
Using the Administration Web Service.....	12
Chapter 3: Job Monitoring.....	15
About job monitoring.....	15
About jobs.....	15
Requesting a list of jobs.....	16
Chapter 4: Capturing Snapshots.....	17
About snapshots.....	17
Restrictions for taking a snapshot.....	18
Creating a snapshot.....	18
Restoring an MDEX Engine from a snapshot.....	19
cpmdex syntax.....	19
Chapter 5: Dgraph Administrative Tasks.....	21
Checking the Dgraph with the ping command.....	21
About connecting Web browsers to your MDEX Engine.....	21
Managing Dgraph core dump files.....	22
Managing Dgraph crash dump files on Windows.....	22
Managing Dgraph core dump files on Linux.....	22
Collecting debugging information.....	23
Logs created by the Dgraph.....	23
Troubleshooting socket and port errors with Dgraph.....	24
Running multiple Dgraphs on the same Windows machine.....	24
Troubleshooting baseline update failures.....	24
Identifying connection errors	25
Chapter 6: Administrative Operations and Logging Variables.....	27
About administrative and configuration operations.....	27
List of administrative operations.....	27
About MDEX Engine logging variables.....	33
Logging variable operation syntax.....	33
List of configuration operations.....	34
List of supported logging variables.....	34
Chapter 7: Managing the Merge Policy.....	37
Using a merge policy for incremental updates.....	37
Types of merge policies.....	37
Setting or changing the merge policy.....	38
Setting the merge policy with the Configuration Service API.....	38
Changing the merge policy of a running MDEX Engine.....	40

Forcing a merge.....	40
Chapter 8: MDEX Engine Process Management.....	43
Running the MDEX Engine as a Windows service.....	43
SC Create command syntax.....	43
Creating the MDEX Engine Windows service.....	46
Setting a service description.....	46
Modifying the service configuration.....	47
Deleting the MDEX Engine Windows service.....	48
Using the Windows Services utility.....	48
Logging in service mode.....	50
Starting the MDEX Engine from initab.....	51
Chapter 9: Deploying Latitude in a Cluster.....	53
Cluster overview.....	53
Latitude cluster architecture.....	54
Important cluster concepts.....	55
Before you begin.....	57
System and hardware requirements.....	57
Downloading, installing and starting the Cluster Coordinator.....	59
Planning cluster nodes.....	62
Cluster behavior.....	62
Building a cluster.....	64
Starting the MDEX Engine as the leader node.....	64
Starting a follower node.....	65
Summary of operations handled by the leader node and any node.....	66
Connecting the leader node with the Data Integrator.....	67
Connecting a cluster with Latitude Studio.....	67
Running a cluster.....	70
Removing a follower node.....	70
Chapter 10: Using Endeca SSL Certificate Utilities.....	71
Certificate files used by Endeca components.....	71
Generating SSL certificates.....	71
Generating standard SSL certificates on UNIX.....	72
Generating standard SSL certificates on Windows.....	72
Generating custom certificates.....	72
Copying the SSL certificates to other machines.....	73
Importing SSL certificates in Internet Explorer.....	74
Configuring the MDEX Engine for SSL mutual authentication.....	74
Converting PEM-format keys to JKS format.....	76
Chapter 11: Latitude Studio Administrative Tasks.....	79
About Latitude Studio administrative tasks.....	79
About the Latitude Studio Control Panel.....	79
Overview of the Control Panel sections.....	79
Accessing the Control Panel.....	80
Installing a new theme.....	80
Setting up the email server for Bookmarks support.....	81
Appendix A: Endeca Flag Reference.....	83
Dgraph flags.....	83



Copyright and disclaimer

Product specifications are subject to change without notice and do not represent a commitment on the part of Endeca Technologies, Inc. The software described in this document is furnished under a license agreement. The software may not be reverse engineered, decompiled, or otherwise manipulated for purposes of obtaining the source code. The software may be used or copied only in accordance with the terms of the license agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Endeca Technologies, Inc.

Copyright © 2003-2011 Endeca Technologies, Inc. All rights reserved. Printed in USA.

Portions of this document and the software are subject to third-party rights, including:

Corda PopChart® and Corda Builder™ Copyright © 1996-2005 Corda Technologies, Inc.

Outside In® Search Export Copyright © 2008 Oracle. All rights reserved.

Rosette® Globalization Platform Copyright © 2003-2005 Basis Technology Corp. All rights reserved.

Teragram Language Identification Software Copyright © 1997-2005 Teragram Corporation. All rights reserved.

Trademarks

Endeca, the Endeca logo, Guided Navigation, MDEX Engine, Find/Analyze/Understand, Guided Summarization, Every Day Discovery, Find Analyze and Understand Information in Ways Never Before Possible, Endeca Latitude, Endeca InFront, Endeca Profind, Endeca Navigation Engine, Don't Stop at Search, and other Endeca product names referenced herein are registered trademarks or trademarks of Endeca Technologies, Inc. in the United States and other jurisdictions. All other product names, company names, marks, logos, and symbols are trademarks of their respective owners.

The software may be covered by one or more of the following patents: US Patent 7035864, US Patent 7062483, US Patent 7325201, US Patent 7428528, US Patent 7567957, US Patent 7617184, US Patent 7856454, US Patent 7912823, Australian Standard Patent 2001268095, Republic of Korea Patent 0797232, Chinese Patent for Invention CN10461159C, Hong Kong Patent HK1072114, European Patent EP1459206, European Patent EP1502205B1, and other patents pending.

Preface

Endeca® Latitude applications guide people to better decisions by combining the ease of search with the analytic power of business intelligence. Users get self-service access to the data they need without needing to specify in advance the queries or views they need. At the same time, the user experience is data driven, continuously revealing the salient relationships in the underlying data for them to explore.

The heart of Endeca's technology is the MDEX Engine.™ The MDEX Engine is a hybrid between an analytical database and a search engine that makes possible a new kind of Agile BI. It provides guided exploration, search, and analysis on any kind of information: structured or unstructured, inside the firm or from external sources.

Endeca Latitude includes data integration and content enrichment tools to load both structured and unstructured data. It also includes Latitude Studio, a set of tools to configure user experience features including search, analytics, and visualizations. This enables IT to partner with the business to gather requirements and rapidly iterate a solution.

Related Links

[About this guide](#) on page 7

This guide describes the administrative tasks related to Endeca Latitude.

[Who should use this guide](#) on page 7

This guide is intended for system administrators who administer and maintain an Endeca Latitude implementation.

[Conventions used in this guide](#) on page 8

This guide uses the following typographical conventions:

[Contacting Endeca Customer Support](#) on page 8

The Endeca Support Center provides registered users with important information regarding Endeca software, implementation questions, product and solution help, training and professional services consultation as well as overall news and updates from Endeca.

About this guide

This guide describes the administrative tasks related to Endeca Latitude.

Who should use this guide

This guide is intended for system administrators who administer and maintain an Endeca Latitude implementation.

This guide assumes that the Endeca software is already installed on a development server. It may be already installed in a production environment. It also assumes that you, or your Endeca Services representatives, have already used the Deployment Template to configure the application on the development server.

You can choose to read specific topics from this guide individually as needed while maintaining your Endeca implementation after it has been initially deployed.

Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in monospace font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: `~`

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

Contacting Endeca Customer Support

The Endeca Support Center provides registered users with important information regarding Endeca software, implementation questions, product and solution help, training and professional services consultation as well as overall news and updates from Endeca.

You can contact Endeca Standard Customer Support through the Support section of the Endeca Developer Network (EDeN) at <http://eden.endeca.com>.



Chapter 1

Introduction

This section describes the stage at which you take control of the operation and maintenance of your Endeca implementation.

Taking ownership of your Latitude implementation

As a system administrator, you take ownership of the Latitude implementation at a certain stage. This topic describes the context in which you will perform administrative tasks to maintain the stable operation of a properly functioning Latitude implementation.

This guide assumes that by this point in using the Endeca Latitude software, you or your team have done the following:

- Planned and provisioned the hardware needed for the staging and production environments.
- Installed the Endeca components, including the MDEX Engine, Latitude Studio, and Latitude Data Integrator.
- Read the *Latitude Getting Started Guide*.

Planned the user-facing details of your application, such as the Endeca attributes that will be displayed in Latitude Studio, the search interfaces to be used in the Latitude **Search Box** component, and so on. The *Latitude Studio Power User's Guide* is especially useful in helping you plan your user interface.

In addition, the guide assumes that you have performed the following application-building tasks:

- You have completed the process of extracting source information from your incoming data sources.
- You have completed the process of using Latitude Data Integrator to load your configuration schema and your source the data into the MDEX Engine, thus creating the Endeca index files.
- You have created a working prototype of your Latitude Studio front-end application for your end users. This front-end application can be used to issue requests to the running MDEX Engine in a production environment.
- You have deployed your Endeca Latitude solution in a staging environment, and are either preparing to deploy it in production, or have already deployed it in production.

Overview of administrator tasks

This topic provides a brief overview of the administrator tasks described in this guide.

This guide assumes that you are performing administrator tasks on both the MDEX Engine and Latitude Studio. The types of task that are described in this guide are the following (as grouped by their chapter):

Chapter	Tasks
Using the Administration Web Service	Use the Administration Web Service for MDEX Engine administrative tasks.
Job Monitoring	Obtain information about the jobs that are being currently processed by the MDEX Engine.
Capturing Snapshots	Create snapshots of a running MDEX Engine and use them as a part of your backup and archiving strategy.
Dgraph Administrative Tasks	<ul style="list-style-type: none"> Check the Dgraph with the ping command. Manage Dgraph core dump files. Collect debugging information to help solve problems. Troubleshoot Dgraph socket and port errors. Identify Dgraph connection errors.
Administrative Operations and Logging Variables	<ul style="list-style-type: none"> Shut down a running MDEX Engine. Flush the dynamic cache. Force a query log roll. Merge update generations and sets the system's merge policy. Check the MDEX Engine Statistics page. Rebuild the aspell dictionary for spelling correction. Modify the logging configuration for the MDEX Engine.
Managing the Merge Policy	<ul style="list-style-type: none"> Merge update generations. Set and manage the merge policy for the MDEX Engine.
MDEX Engine Process Management	<ul style="list-style-type: none"> Create a Windows service for running the MDEX Engine in service mode. Add an <code>inittab</code> entry so that <code>init</code> can start the MDEX Engine on a Linux machine.
Deploying Latitude in a Clustered Environment	Set up and manage a cluster of MDEX Engine nodes.
Using Endeca SSL Certificate Utilities	<ul style="list-style-type: none"> Generate standard and custom SSL certificate files to be used for SSL connections to the MDEX Engine. Convert PEM-format certificates to the standard Java KeyStore (JKS) format. Configure the MDEX Engine for SSL mutual authentication.
Latitude Studio Administrative Tasks	<ul style="list-style-type: none"> Perform administrative functions of Latitude Studio from the Control Panel. Install a new theme. Set up the email server for Bookmarks support.



Chapter 2

Using the Administration Web Service

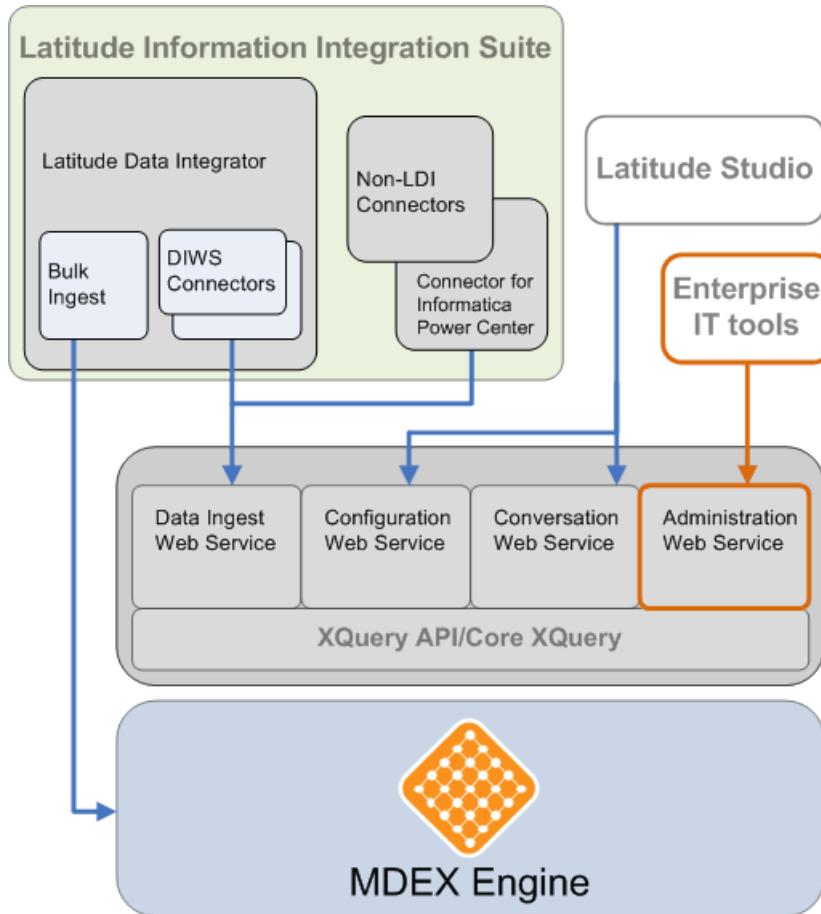
This section describes how to use the Administration Web Service with the MDEX Engine.

About the Administration Web Service

The Administration Web Service enables IT engineers to administer the MDEX Engine server.

The Administration Web Service provides an interface to the MDEX Engine that accesses the server outside of the MDEX Engine data layer (as opposed to the other Endeca Web services, which communicate through XQuery with the data layer of the MDEX Engine). As such, this Web service is designed to serve the various needs of IT engineers and server administrators responsible for the administrative and IT maintenance of the MDEX Engine server.

The following diagram shows how the Administration Web Service fits into the larger picture of packaged Web services that serve the MDEX Engine:



Accessing the Administration Web Service

The Administration Web Service is declared in `admin.wsdl`.

You can access the Administration Web Service at the following URL:

```
http://localhost:<port>/ws/admin
```

Using the Administration Web Service

The Administration Web Service contains administrative options.

For example `createSnapshotOperation($name, $path)` creates a snapshot of the MDEX Engine state as a tree of hard links under `$name` in directory `$path`.

Function description

The Administration Web Service takes as its input parameters to the functions it contains and performs the requested operations.

Request

The input to the Administration Web Service depends on the function. For example, if your goal is to create the MDEX Engine snapshot, you specify a name and a directory path to the snapshot file; if you want to send a request for job monitoring in the MDEX Engine, you specify an operation that lists all currently running jobs.

Response

The Administration Web Service returns:

- An `<operation successful>` response element if there are no problems.
- A `<fault>` element if an exception was thrown internally.

Operations

The Administration Web Service contains the following operations:

Operation	Description
createSnapshotOperation	Create a snapshot representing a consistent view of the state of the MDEX Engine at a specific point in time. As an argument, specify the name for a snapshot, such as <code>NewSnapshot</code> , and an absolute path to the snapshot directory in the URI format, such as <code>file:///mydirectory/home/snapshots/</code> .
listJobsOperation	List the jobs that are currently running in the MDEX Engine, such as queries, updating operations or administrative services.

Example

The following examples show the Administration Web Service request and response bodies for creating a snapshot.

To access the Administration Web Service, send a soap request to the following URL:

```
http://localhost:<port>/ws/admin
```

This example shows the Post body of the Administration Web Service request that creates a snapshot:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <admin:Request>
      <admin:createSnapshotOperation path="file:///mydirectory/home/snapshots/" name="NewSnapshot" />
    </admin:Request>
  </soapenv:Body>
</soapenv:Envelope>
```

This example shows the response body of the Administration Web Service request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <admin:Response xmlns:admin="http://www.endeca.com/XQuery/admin/lib/2010" />
  </soapenv:Body>
</soapenv:Envelope>
```

```
<admin:createSnapshotSuccess/>
</admin:Response>
</soapenv:Body>
</soapenv:Envelope>
```



Note: For more information about the functions used in the Administration Web Service, see the Administration API section of the *MDEX Engine API Reference*.



Chapter 3

Job Monitoring

This section describes how you, as a system administrator, can obtain information, monitor and control various long-running jobs in the MDEX Engine — such as updates or long-running queries — using the Administration Web Service.

About job monitoring

In many instances, it is useful to have more information about the jobs that are being currently processed by the MDEX Engine.

When the MDEX Engine processes record updates, all other operations are temporarily stopped, waiting for the update operations to complete, and then restarted after the updates are finished. In such instances a system administrator needs to have more information about which operations are currently being processed by the MDEX Engine.

When administering an MDEX Engine, it is useful to manage long-running jobs in the following scenarios:

- The data architect updates the configuration, for example by issuing a request to make an attribute (in your records schema) value searchable, and the MDEX Engine becomes unresponsive because it is running an update operation. The data architect can make a request to see when the MDEX Engine had started running the update.
- An administrator of the Endeca application sends a query to the MDEX Engine and the MDEX Engine becomes unresponsive because it is already running a long-running query. The administrator can make a request to see when the MDEX Engine had started processing the query.
- An administrator of the Endeca application would like to send an update and needs to verify whether any other updates are already being processed or are queued up before submitting a new update. Making an Administration Web Service request allows the administrator to understand whether a new update will begin processing immediately.
- An administrator of the Endeca application wants to check which updates have been submitted recently.

About jobs

You can monitor several types of jobs.

You can monitor the following types of jobs:

- A query. This can be any type of a web service request that submits a query to the MDEX Engine.

- An update. This is an update to the records sent through the Data Ingest Web Service.
- An administrative operation. This can be a request for any administrative operation.

Requesting a list of jobs

Using the `listJobsOperation` of the Administration Web Service, you can make a job monitoring request for a list of jobs that are currently being processed by the MDEX Engine or are waiting in the queue.

To issue a job monitoring request:

Specify the `listJobsOperation` to the Administration Web Service, as in the following example:

```
<admin:request xmlns:admin="http://www.endeca.com/MDEX/admin/2010">
    <admin:listJobsOperation/>
</admin:request>
```

The response contains a list of currently running jobs, and includes the following information:

- The job ID. This is an internal ID assigned by the MDEX Engine.
- The job start time. It indicates the time at which the MDEX Engine received a request for this job, and has an outstanding request for processing it. The start time does not indicate that the job had actually started at that time.
- Job type. The job type indicates the type of job that is being monitored. It can be `Admin`, `Query`, or `Update`.

Example

In this example of the Administration Web Service response, you can see that a query with the job ID 10 is currently running. You can also observe its start time. In addition, the response indicates that a request of type `Admin` has been issued as well, with the Job ID 11 (this job represents the job monitoring request itself).

```
<admin:response xmlns:admin="http://www.endeca.com/MDEX/admin/2010">
    <admin:jobs>
        <admin:job jobId="10">
            <admin:startTime>2011-04-18T10:26:41.449Z</admin:startTime>
            <admin:jobType>Query</admin:jobType>
        </admin:job>
        <admin:job jobId="11">
            <admin:startTime>2011-04-18T10:26:41.449Z</admin:startTime>
            <admin:jobType>Admin</admin:jobType>
        </admin:job>
    </admin:jobs>
</admin:response>
```



Chapter 4

Capturing Snapshots

You can create snapshots of a running MDEX Engine and use them as a part of your backup and archiving strategy. This section describes the snapshot process.

About snapshots

A snapshot represents a consistent view of the state of the MDEX Engine index at a specific point in time. By taking a snapshot, you can capture the state of the index without shutting down the MDEX Engine.

Snapshots operate at the data layer level of the MDEX Engine index. The data layer implements a versioned data store in the MDEX Engine index, which includes a collection of files such as data structures and indices. When you create a snapshot, the data layer identifies the set of files that comprise a version, and captures the state of the system as it exists at that moment.

A backup operation without taking the snapshot would involve the need to stop the MDEX Engine and copy its index, which can take a long time. In contrast, you can create a snapshot while the MDEX Engine is handling updates and queries, without downtime. After a snapshot is complete, you can plan and create a backup at your convenience.

A snapshot contains all the files needed to restore the MDEX Engine to a specific state.

To create a snapshot, you issue a request to the MDEX Engine through `createSnapshotOperation` in the Administration Web Service. In a cluster of MDEX Engine nodes, this operation should be performed on the leader node only.

After you take the snapshot, you can back up the state in a manner compatible with your archiving strategy, whether you have an elaborate backup infrastructure or a simpler solution based on CIFS or NFS protocols.

If the need arises, you can restore an MDEX Engine from a snapshot with the `cpmdex` command.



Important: Because snapshots represent internal files needed to restore the MDEX Engine data structures, they are not human-readable and should be treated as read-only. Modifying a snapshot can corrupt the MDEX Engine.

Restrictions for taking a snapshot

The following restrictions apply when taking snapshots.

- The `createSnapshotOperation` cannot be combined with other operations in the same Web service request.
- Avoid submitting snapshot requests when the MDEX Engine is running updates. The MDEX Engine processes update requests and snapshot requests one at a time. If an update takes a long time to process, it can block a snapshot and cause the snapshot to time out. If a snapshot request times out, an error message displays. Try submitting it again once the update is completed.
- The `createSnapshotOperation` requires a URI absolute path indicating where the snapshot should be recorded; it should have the following format: `file:///localdisk/username/dir`. Specifying a relative path causes the operation to fail.
- If you are running a cluster of MDEX Engine nodes (as opposed to running the MDEX Engine on a single server that is not part of the cluster), run the `createSnapshotOperation` on the leader node.
- Do not modify snapshot files. Because snapshots represent internal files needed to restore the MDEX Engine data structures, they are not human-readable and should be treated as read-only. Modifying a snapshot can corrupt the MDEX Engine.

Creating a snapshot

You create a snapshot with the `createSnapshotOperation` interface in the Administration Web Service.

Before taking the snapshot, ensure that you have reviewed the list of restrictions.

To create a snapshot:

1. Run the client application that will invoke the Administration Web Service.
2. Specify the `snap_URI` and the `snap_name` in the following XML snippet:

```
<admin:createSnapshotOperation path="${snap_URI}" name="${snap_name}" />
```

- `snap_URI` represents an absolute URI path to the file system location, and is located on the same file system as the MDEX Engine. It should be of the format `file:///localdisk/username/dir`.
- `snap_name` represents the name of the snapshot.

The Web service returns a confirmation message if the snapshot was successfully captured.



Note: You should treat snapshots as read-only. Modifying a snapshot could corrupt your running MDEX Engine.

After you capture the snapshot, copy it to a safe location using the archiving method of your choice. Deleting a snapshot does not affect the MDEX Engine.

Restoring an MDEX Engine from a snapshot

You can restore an MDEX Engine from an archived snapshot using the `cpmdex` command. The command copies files from the archived snapshot into the index of the MDEX Engine.

The MDEX Engine `bin` directory contains `cpmdex.cmd` (Windows) and `cpmdex.sh` (Linux) versions of this command.

The `cpmdex` command takes as input the path to the archived snapshot and the path to the MDEX Engine instance which will be restored.



Important: Before running the `cpmdex` command, use `dgraph --version` to ensure that the version of MDEX Engine index to which you are restoring from the snapshot matches the version of the MDEX Engine index from which the snapshot was captured.

To restore the MDEX Engine index from a snapshot:

1. Stop the MDEX Engine that are you are about to restore using `/admin?op=exit`.
If you are running multiple MDEX Engine instances in a cluster, stop all MDEX Engine nodes.
2. From a command prompt, run the `cpmdex` command.
If you are restoring the MDEX Engine index in a cluster, run this command on a leader node.

An example on Windows is:

```
cpmdex -a backup\2010-07-20 -m endeca\myapp\my_mdex
```

An example on Linux is:

```
$ cpmdex.sh -a mnt/backup/2010-07-20 -m home/endeca/myapp/my_mdex
```

3. Start the MDEX Engine.

cpmdex syntax

This topic contains syntax for the `cpmdex` command.

The syntax for the `cpmdex` command is as follows:

```
cpmdex -a <archive_path> -m <mindex_path> -t <transfer_path>
```

The `cpmdex` command uses the following parameters:

Options	Description
<code>-a <archive_path></code>	Required. The absolute file path to the directory containing the archived snapshot.
<code>-m <mindex_path></code>	Required. The absolute file path to the directory where the snapshot should be restored. The end of this path should match the value passed to the Dgraph executable when it is started.
<code>-t <transfer_path></code>	The file path to a directory to which the snapshot should be moved. This option uses a move operation, instead of a copy, to restore the files to the MDEX Engine.

Options	Description
	You may want to use this option if the backup has already been copied from the archive to the local file system and you want to save considerable I/O bandwidth.
-h	The help for this command.

In this example, the `cpmdex` command copies the snapshot from the `backup\2011-03-20` directory and restores it to the `endeca\myapp\my_mdex` directory on Windows:

```
cpmdex -a backup\2011-03-20 -m endeca\myapp\my_mdex
```



Chapter 5

Dgraph Administrative Tasks

This section describes some basic administrative tasks for the Dgraph. In addition, it contains Dgraph troubleshooting tips and describes the Dgraph logs.

Checking the Dgraph with the ping command

A quick way of checking the health of a Dgraph is to ping it.

To check the aliveness of a Dgraph:

Issue the following command:

```
http://<DgraphServerNameOrIP:DgraphPort>/admin?op=ping
```

It returns a lightweight HTML response page with the following content:

```
dgraph <host:port> responding at <date/time>
```



Note: You can also view the MDEX Engine Statistics page to check whether the MDEX Engine is running and accepting queries.

About connecting Web browsers to your MDEX Engine

For security reasons, you should never allow user Web browsers to connect directly to your MDEX Engine server (although an administrator may choose to connect directly to the MDEX Engine server using proper precautions).

Browsers started by non-administrators should always connect to your application through an application server.

IPv4 and IPv6 address support

The MDEX Engine supports both IPv4 (Internet Protocol Version 4) and IPv6 (Internet Protocol Version 6) addressing schemes for connections. This IPv4 and IPv6 addressing support is configured automatically in the MDEX Engine, so there is no need for the administrator to do any explicit addressing configuration.

Managing Dgraph core dump files

In the rare case of a Dgraph crash, the Dgraph writes its core dump files on disk.

When the Dgraph runs on a very large data set, its in-memory representation of the index size may exceed the size of the physical RAM. If such a Dgraph process fails, it may need to write out potentially very large core dump files on disk.

To troubleshoot the Dgraph, it is often useful to preserve the entire set of core files written out as a result of such failures. When there is not enough disk space, only a portion of the files is written to disk until this process stops. Since the most valuable troubleshooting information is contained in the last portion of core files, to make these files meaningful for troubleshooting purposes, it is important to provision enough disk space to capture the files in their entirety.

Two situations are possible, depending on your goal:

- To troubleshoot a Dgraph crash, provision enough disk space to capture the entire set of core files. In this case, the files will be saved at the expense of potentially filling up the disk.
- To prevent filling up the disk, you can limit the size of these files on the operating system level. In this case, with large Dgraph applications, only a portion of core files is saved on disk. This may limit their usefulness for debugging purposes.

Related Links

[Managing Dgraph crash dump files on Windows](#) on page 22

On Windows, all Dgraph crash dump files are saved on disk by default.

[Managing Dgraph core dump files on Linux](#) on page 22

Endeca recommends using the `ulimit -c unlimited` setting for Dgraph core dump files. Non-limited core files contain all Dgraph data that is resident in memory (RSS of the Dgraph).

Managing Dgraph crash dump files on Windows

On Windows, all Dgraph crash dump files are saved on disk by default.

The MDEX Engine uses the `MiniDump` function from the Microsoft `DbgHelp` library.

Provision enough disk space to accommodate core files based on this estimate:

- The projected upper limit for the size of these files is equal, at a maximum, to the size of the physical memory used by the MDEX Engine plus index size. Often the files take up less space than that.

Managing Dgraph core dump files on Linux

Endeca recommends using the `ulimit -c unlimited` setting for Dgraph core dump files. Non-limited core files contain all Dgraph data that is resident in memory (RSS of the Dgraph).

Since large MDEX applications may take up the entire amount of available RAM, the core dump files can also grow large and take up the space equal to the size of the physical RAM on disk plus index size.

Provision enough disk space to accommodate core files based on this estimate:

- The projected upper limit for the size of these files should be equal, at a maximum, to the size of the physical RAM. Often the files take up less space than that.



Note: If you are not setting `ulimit -c unlimited`, you could be seeing the MDEX Engine crashes that do not write any core files to disk, since on some Linux installations the default for `ulimit -c` is set to 0.

Alternatively, it is possible to limit the size of core files with the `ulimit -c <size>` command, although this is not recommended. If you set the limit size in this way, the core files cannot be used for debugging, although their presence will confirm that the Dgraph had crashed. To be able to troubleshoot the crash, change this setting to `ulimit -c unlimited`, and reproduce the crash while capturing the entire core file. Similarly, to enable Endeca Support to troubleshoot the crash, you will need to reproduce the crash while capturing the full core file.

Collecting debugging information

Before attempting to debug an issue with the MDEX Engine, collect the following information.

- Hardware specifications and configuration.
- Description of the Endeca topology (servers, number of Dgraphs).
- The data from the MDEX Engine Statistics page.
- The contents of the pipeline directory.
- Dgraph input.
- Partial update files.
- Description of typical partial updates.
- Description of which Dgraphs are affected.

Related Links

[Logs created by the Dgraph](#) on page 23

The Dgraph creates several logs, although some of these logs depend on your implementation and the Endeca components that you may be using. This topic provides a summary of these logs.

Logs created by the Dgraph

The Dgraph creates several logs, although some of these logs depend on your implementation and the Endeca components that you may be using. This topic provides a summary of these logs.

You can use these Dgraph logs to troubleshoot MDEX Engine queries, or to track performance of particular queries or updates.

Dgraph request log

The Dgraph request log is always created. You can use it to debug both requests and update processing. It contains one entry for each request processed. The requests are sorted by their timestamp.

If you are using the Dgraph from the command line, create the path to the request log in the Dgraph working directory with the filename `dgraph.reqlog`.

By default, the Dgraph truncates the contents of the body for POST requests at 64K. This default setting saves disk space in the log, especially during the process of adding large numbers of records to the MDEX Engine. If you need to review the log for the full contents of the POST request body, contact Endeca Support.

Dgraph error log

The Dgraph error log is created only if you redirect `stderr` to a file, using a command line or a `dgraph --out` flag. Otherwise, error messages appear in `stderr`.

The Dgraph error log includes startup messages as well as warning and error messages. It can be configured via Dgraph flags (such as `-v`). Also, the `/admin?op=logroll` command forces a query log roll, with the side effect of remapping `stdout`.

Troubleshooting socket and port errors with Dgraph

The Dgraph cannot start if its process cannot bind to a socket and its port cannot initialize. This error tends to occur when you upgrade the MDEX Engine and attempt to use a port that is already occupied by another process on your server.

The following errors appear in the Dgraph log:

```
ERROR (date and time)
DGRAPH {dgraph,baseline}: Unable to bind
to socket [err='Result too large',errno=34]
FATAL (date and time)
DGRAPH {dgraph,baseline}: Unable to initialize the
main server port: 8000
```

The "Unable to bind to socket" errors usually indicate that the port in question is already in use by another process.

The Windows command-line utility `netstat -ano` lists all ports in use along with the process ID of the process using them. Use this utility to identify the process ID occupying port 8000, and locate that process in the Windows Task Manager to confirm that it is used by another process. This prevents the Dgraph from starting.

To identify ports in use on your Windows system:

1. Run `netstat -ano`
This command lists ports and process IDs of all processes that are running.
2. Examine which process occupies the port that the Dgraph is trying to use. In this example, it is port 8000.
3. Run the Dgraph on another port, or ensure that the previously occupied port can be freed to be used by the MDEX Engine.

Running multiple Dgraphs on the same Windows machine

If you have more than one Dgraph starting on a single Windows machine, each Dgraph constructs its port in isolation.

This prevents multiple Dgraphs running on a single machine from presenting inconsistent behavior.

Troubleshooting baseline update failures

To debug baseline update failures, examine the Dgraph request log.

Review Dgraph request logs. Review the logs around the time of the baseline update failure, to rule out issues in the Dgraph.

Notice the times when health checks were sent to the Dgraph, the Dgraph was restarted, the partial updates were issued, and the last query was issued. For example, this modified abstract from the Dgraph request log shows activity for a period of time:

```
12096521815/1/09 14:29 last search query
12096522265/1/09 14:30 health check
12096526095/1/09 14:36 last health check for x time
12096571605/1/09 15:52 health checks resume
12096574435/1/09 15:57 last empty health check
12096601195/1/09 16:41 Dgraph startup
12096601435/1/09 16:42 first query
```

Notice that the Dgraph did not receive any requests besides health checks for a period of time from 14:29 to 15:57. The log does not include error messages. The Dgraph was not restarted during this time. These observations indicate that the problem that led to the baseline update failure in this example possibly occurred outside of the Dgraph.

Identifying connection errors

If the Dgraph standard out log contains connection broken messages, although it may look like the problem occurred with the Dgraph, the actual cause of the problem is usually a broken connection between the server that hosts the front-end application and the server that hosts the Dgraph.

In the case of connection errors, various parts of the Endeca implementation issue the following error and warning messages:

- The Dgraph standard out log contains warnings similar to the following:

```
WARN [DATE TIME] UTC (1239830549803)
DGRAPH {dgraph}: Aborting request: connection broken: client 10.10.21.21
```

- And finally, the Dgraph request log contains an abnormal status 0 message similar to the following:

```
1239830549803 10.6.35.35 - 349 0 19.35 0.00 0 - 0 0 - -
```

Typically, the connection broken message means that the Dgraph encountered an unexpected failure in the connection between the client and the Dgraph. This type of error may occur outside the Dgraph, such as in the network, or be caused by the timeout of the client application session.

Investigate the connection between the client and the Dgraph. For example, to prevent timeouts of the client application sessions, you may decide to implement front-end application retries.



Chapter 6

Administrative Operations and Logging Variables

The MDEX Engine supports many administrative and configuration operations that you can access through simple URLs. You can use these operations and their logging variables to control the behavior of the MDEX Engine cleanly from within the system.

About administrative and configuration operations

Administrative and configuration operations make it possible to check Dgraph statistics, and enable or disable diagnostic flags without having to stop a running Dgraph. They also let you stop and restart the Dgraphs. This section lists URLs exposed by the Dgraph, describes the functions of each URL, and defines the syntax of those URLs.

The syntax of administrative and configuration operations

In the following listings, `<host>` refers to the hostname or IP address of the MDEX Engine and `<port>` refers to the port on which the MDEX Engine is listening. Queries to these URLs are handled in the MDEX Engine's request queue like any other request—that is, they are handled on a first-come, first-served basis. They are also reported in the MDEX Engine request log like any other request.

For administrative operations, the syntax is:

```
http://<host>:<port>/admin?op=<supported-operation>
```

For configuration operations, the syntax is:

```
http://<host>:<port>/config?op=<supported-operation>
```



Note: If you are using HTTPS mode, use `https` in the URL.

List of administrative operations

Administrative (or admin) operations listed in this topic allow you to control the behavior of the MDEX Engine from within the system.

The MDEX Engine recognizes the following admin operations:

Admin operation	Description
/admin?op=exit	<p>Shuts down a running MDEX Engine after completing all in-progress requests and background merges.</p> <p>You can also specify an optional timeout limit that immediately shuts down the MDEX Engine if the shutdown operation takes longer than the specified timeout limit.</p>
/admin?op=flush	Specifies when the MDEX Engine should flush its dynamic cache.
/admin?op=help	Returns the usage page for all of the admin operations.
/admin?op=logroll	Forces a query log roll, with the side effect of remapping stdout.
/admin?op=merge	Merges update generations and sets the system's merge policy.
/admin?op=ping	Checks the aliveness of an MDEX Engine and returns a lightweight message.
/admin?op=reload-services	A Web services operation that reloads the application's main and library modules.
/admin?op=stats	Returns the MDEX Engine Statistics page.
/admin?op=statsreset	Resets the MDEX Engine Statistics page.
/admin?op=updateaspell	Rebuilds the aspell dictionary for spelling correction from the data corpus while continuing to issue queries and partial updates to the MDEX Engine and without stopping and restarting it.

Related Links

[exit](#) on page 29

/admin?op=exit gracefully shuts down a running MDEX Engine.

[flush](#) on page 31

/admin?op=flush flushes the Dgraph cache.

[help](#) on page 31

/admin?op=help returns the usage page for all of the administrative operations.

[logroll](#) on page 31

/admin?op=logroll forces a query log roll, with the side effect of remapping stdout.

[merge](#) on page 31

/admin?op=merge forces a merge, and (optionally) changes the merge policy of a running MDEX Engine. In a cluster of MDEX Engine nodes, this command should be used on the leader node only.

[ping](#) on page 31

/admin?op=ping checks the aliveness of an MDEX Engine and returns a lightweight message.

[reload-services](#) on page 31

/admin?op=reload-services is a Web services operation that reloads the application's main and library modules.

[stats](#) on page 32

/admin?op=stats returns the MDEX Engine Statistics page.

[statsreset](#) on page 32

/admin?op=statsreset resets the MDEX Engine Statistics page.

[updateaspell](#) on page 32

The `admin?op=updateaspell` administrative operation lets you rebuild the aspell dictionary for spelling correction from the data corpus while continuing to issue queries and updates to the MDEX Engine and without stopping and restarting it.

exit

`/admin?op=exit` gracefully shuts down a running MDEX Engine.

The `/admin?op=exit` command has two formats:

- The base version does not use the `timelimit` option.
- The timeout version uses the `timelimit` option.

Using the base version

The format of the base version is:

```
/admin?op=exit
```

The `exit` operation works as follows:

- Any new non-admin request will get an HTTP response code 503 (Service Unavailable).
- Any in-progress request will finish normally (including updates).
- The MDEX Engine will wait to exit until the following conditions have been met:
 - All requests have finished (including updates).
 - All background merging has been completed.

The `exit` operation's output to the browser looks similar to the following:

```
Dgraph admin, OK
Dgraph shutting down at Thu Feb 17 13:12:54 2011
```

The command also writes shutdown information to the Dgraph error log, as in this example:

```
Shutdown request with received at Thu Feb 17 13:12:54 2011.
Shutdown will complete when all outstanding jobs are complete.
All dgraph transactions completed at Thu Feb 17 13:12:54 2011, exiting
normally (pid=4128)
```

The base `exit` operation is the recommended way to shut down the MDEX Engine because it gracefully completes all transactions and exits cleanly. However, note that because the MDEX Engine waits until all background merging has completed, the shutdown process could potentially take several hours if the request occurs during a major merge. Therefore, if the speed of the shutdown is more important than the completing a merge, you should consider using the `timelimit` option to set a time limit for the shutdown operation.

Using the `timelimit` option

The timeout version lets you specify a time limit, in seconds, of the shutdown procedure. The format of the timeout version is:

```
/admin?op=exit&timelimit=seconds
```

where `seconds` is a positive integer.

This example uses a time limit of 30 seconds:

```
/admin?op=exit&timelimit=30
```

The `exit&timelimit` operation works as follows:

- A time limit of 0 (zero) will shut down the MDEX Engine immediately.
- Any queries still in progress when the time limit is reached will not return a result to the client (i.e., the client will observe a closed connection).
- Any queries still in progress when the time limit is reached will not be logged in the Dgraph log.
- Any updates still in progress when the time limit is reached will not be applied.
- Any background merges still in progress when the time limit is reached will be aborted at the end of the timeout.
- The number of in-progress queries is written to the Dgraph error log just before exiting, along with a message stating that the shutdown time limit was reached.

Issuing an `exit` command with a time limit ensures that the MDEX Engine shuts down within that time limit, regardless of prior or following `exit` queries (i.e., `exit` commands with a time limit can only shorten the MDEX Engine's time to live). These examples demonstrate what happens when successive `exit` commands are issued:

- If `exit&timelimit=30` is issued and 10 seconds later `exit&timelimit=0` is issued, the MDEX Engine will exit immediately when the second request is issued (if it hasn't already exited).
- If `exit&timelimit=30` is issued and 10 seconds later `exit&timelimit=5` is issued, the MDEX Engine will exit 5 seconds after the second request, or when all queries are drained (whichever comes first).
- If `exit&timelimit=30` is issued and 10 seconds later `exit&timelimit=30` is issued, the MDEX Engine will exit 30 seconds after the first request, or when all queries are drained (whichever comes first).
- If `exit&timelimit=30` is issued and 10 seconds later the base `exit` command is issued, the MDEX Engine will exit 30 seconds after the first request, or when all queries are drained (whichever comes first).
- If `exit` is issued and 10 seconds later `exit&timelimit=30` is issued, the MDEX Engine will exit 30 seconds after the second request, or when all queries are drained (whichever comes first).

An `exit&timelimit=30` operation's output to the browser looks like this:

```
Dgraph admin, OK
Dgraph shutting down within 30 seconds at Thu Feb 17 14:09:38 2011
```

The command also writes shutdown information to the Dgraph error log, as in this example for an `exit&timelimit=30` command:

```
Shutdown request with time limit of 30 seconds received at Thu Feb 17
14:09:38 2011.
Shutdown will complete when all outstanding jobs are complete, or within
30 seconds, whichever happens earlier.
All dgraph transactions completed at Thu Feb 17 13:12:54 2011, exiting
normally (pid=4128)
```

If command were for an immediate shutdown (`exit&timelimit=0`) and queries were still in progress, the Dgraph error log would contain a message similar to this example:

```
Shutdown request with time limit of 0 seconds received at Thu Feb 17 14:18:46
2011.
Shutdown will complete when all outstanding jobs are complete, or within 0
seconds, whichever happens earlier.
Shutdown time limit reached at Thu Feb 17 14:18:46 2011, exiting with jobs
still in progress:
1 request is still active or queued, and will not be logged to the request
log
1 job is currently executing, and will be killed.
0 jobs are queued, and will not be executed.
```

flush

/admin?op=flush flushes the Dgraph cache.

The flush operation clears all entries from the Dgraph cache. It returns the following message:

```
flushing cache...
```

help

/admin?op=help returns the usage page for all of the administrative operations.

logroll

/admin?op=logroll forces a query log roll, with the side effect of remapping stdout.

The logroll command returns a message similar to the following:

```
rolling log... Successfully remapped stdout/stderr to specified
path "C:\Endeca\apps\JanWine\logs\dgraphs\Dgraph2\Dgraph2.log".
Successfully rolled log file.
```

merge

/admin?op=merge forces a merge, and (optionally) changes the merge policy of a running MDEX Engine. In a cluster of MDEX Engine nodes, this command should be used on the leader node only.

Related Links

[Managing the Merge Policy](#) on page 37

This chapter describes how to set and manage an MDEX Engine's merge policy.

ping

/admin?op=ping checks the aliveness of an MDEX Engine and returns a lightweight message.

You can view the MDEX Engine Statistics page to check whether the MDEX Engine is running and accepting queries, but that comes with some overhead. A quicker way to check the aliveness of a Dgraph is by running the ping command.

Because ping requests are given the highest priority and are processed synchronously (as they are received), a ping response time is independent of the number of outstanding requests in the MDEX Engine.

The ping command returns a lightweight page that lists the MDEX Engine, the current date and time, such as the following:

```
dgraph example.endeca.com:8000 responding at Wed Oct 27 15:35:27 2010
```

You can use this operation to monitor the health or heartbeat of the MDEX Engine, and as a health check for load balancers.

reload-services

/admin?op=reload-services is a Web services operation that reloads the application's main and library modules.

The admin?op=reload-services operation causes the Dgraph to process all existing preceding queries, temporarily stop processing other queries and begin to process admin?op=reload-ser-

vices. After it finishes processing this operation, the Dgraph resumes processing queries that queued up temporarily behind this request.

In a cluster of MDEX Engine nodes, this command should be run on the leader node only.



Note: `admin?op=reload-services` can be a time-consuming operation, depending on the number of XQuery modules that are present and that have to be compiled.

stats

`/admin?op=stats` returns the MDEX Engine Statistics page.

The MDEX Engine Statistics page provides a detailed breakdown of what the Dgraph is doing, and is a useful source of information about your Endeca implementation's configuration and performance. It provides information such as startup time, last data indexing time, and indexing data path. This lets you focus your tuning and load-balancing efforts. By examining this page, you can see where the Dgraph is spending its time. Begin your tuning efforts by identifying the features on the **Details** tab Hotspots section with the highest totals.

statsreset

`/admin?op=statsreset` resets the MDEX Engine Statistics page.

The `statsreset` operation returns the following message:

```
resetting server stats...
```

updateaspell

The `admin?op=updateaspell` administrative operation lets you rebuild the aspell dictionary for spelling correction from the data corpus while continuing to issue queries and updates to the MDEX Engine and without stopping and restarting it.

Run this command after you have added data records to the MDEX Engine, to enable spelling correction in the MDEX Engine.

During the data ingest process, you can run the `admin?op=updateaspell` command periodically to update the spelling dictionary used by the MDEX Engine for Automatic Spelling Correction and DYM.

In a cluster of MDEX Engine nodes, this command should be run on the leader node only.

The `admin?op=updateaspell` operation performs the following actions:

- Crawls the text search index for all terms which meet the constraint settings.

The constraint settings include minimum word occurrences and maximum and minimum number of characters, for records and attribute values. The MDEX Engine uses these constraints to update the spelling dictionary. You can change them in the Global Configuration Record.

- Compiles a temporary text version of the `aspell` word list, `<db_prefix>.worddat`.
- Converts this word list to the binary format required by `aspell`
- Writes the generated binary file into the current index representation in the MDEX Engine.
- Makes the updated `aspell` spelling dictionary available in the MDEX Engine for processing of all queries arriving after this index update. The MDEX Engine uses this updated dictionary when processing all future queries.



Note: Because of the nature of continuous query, once the MDEX Engine processes this administrative request, it will start using the updated spelling dictionary after a certain point

in its processing, and all newly incoming queries will be answered against the updated spelling dictionary. However, it is not possible to identify after which particular partial update or after which query the MDEX Engine will start using the newly updated spelling dictionary.

The Dgraph applies the updated settings while continuing to run queries and without needing to restart. Only one `admin?op=updateaspell` operation can be processed at a time.

The `admin?op=updateaspell` operation returns output similar to the following in the Dgraph error log:

```
...
spellengine aspell ran successfully.
```

If you start the Dgraph with the `-v` flag, the output also contains a line similar to the following:

```
Time taken for updateaspell, including wait time on any
previous updateaspell, was 290.378174 ms.
```

About MDEX Engine logging variables

You can use logging variables with config operations. This lets you obtain detailed information about Dgraph processing, to help diagnose unexpected application behavior or performance problems, without stopping and restarting the Dgraph or requiring a configuration update.

Although you can also specify general verbose logging at the Dgraph command line with the `-v` flag, it requires a Dgraph restart to take effect.

Related Links

[Logging variable operation syntax](#) on page 33

MDEX Engine logging variables are toggled using the `/config?op=log-enable&name=<variable-name>` and `/config?op=log-disable&name=<variable-name>` operations.

[List of configuration operations](#) on page 34

Configuration (or config) operations listed in this topic allow you to modify configuration and logging information for the MDEX Engine from within the system.

[List of supported logging variables](#) on page 34

The following table describes the supported logging variables that you can use with related config operations to toggle logging verbosity for specified features.

Logging variable operation syntax

MDEX Engine logging variables are toggled using the `/config?op=log-enable&name=<variable-name>` and `/config?op=log-disable&name=<variable-name>` operations.

You can include multiple logging variables in a single request. Unrecognized logging variables generate warnings.

For example, this operation:

```
/config?op=log-enable&name=requestverbose
```

turns on verbose logging for queries, while this operation:

```
config?op=log-enable&name=textsearchrelrankverbose&name=textsearchspellverbose
```

turns on verbose logging for both the text search relevance ranking and spelling features.

However, this operation:

```
config?op=log-enable&name=allmylogs
```

returns an unsupported logging setting message.

In addition, the following operations are supported:

- /config?op=log-status returns a list of all logging variables with their values (true or false).
- The special name `all` can be used with `/config?op=log-enable` or `/config?op=log-disable` to set all logging variables.

List of configuration operations

Configuration (or config) operations listed in this topic allow you to modify configuration and logging information for the MDEX Engine from within the system.

The Dgraph recognizes the following config operations:

Config operation	Description
<code>/config?op=help</code>	Returns the usage page for all of the config operations.
<code>/config?op=log-enable</code>	Enables verbose logging for one or more specified variables.
<code>/config?op=log-disable</code>	Disables verbose logging for one or more specified variables.
<code>/config?op=log-status</code>	Returns verbose logging status.

List of supported logging variables

The following table describes the supported logging variables that you can use with related config operations to toggle logging verbosity for specified features.

Logging variable names are not case sensitive.

Variable	Description
<code>verbose</code>	Enables verbose mode.
<code>requestverbose</code>	Prints information about each request to stdout.
<code>updateverbose</code>	Show verbose messages while processing updates.
<code>recordfilterperfverbose</code>	Enables verbose information about record filter performance.
<code>textsearchrelrankverbose</code>	Enables verbose information about relevance ranking during search query processing.
<code>textsearchspellverbose</code>	Enables verbose output for spelling correction features.
<code>dgraphperfverbose</code>	Enables verbose performance debugging messages during core Dgraph navigation computations.
<code>dgraphrefinementgroupverbose</code>	Enables refinement verbose/debugging messages.

Related Links

[log-enable](#) on page 35

The log-enable operation lets you turn on verbose logging.

[log-disable](#) on page 35

The log-disable operation turns off verbose logging.

[log-status](#) on page 35

The log-status operation returns a list of all logging variables with their values (true or false).

[help](#) on page 31

/admin?op=help returns the usage page for all of the administrative operations.

log-enable

The log-enable operation lets you turn on verbose logging.

You can include multiple logging variables in a single request. Unrecognized logging variables generate warnings.

For example, this operation:

```
/config?op=log-enable&name=requestverbose
```

turns on verbose logging for queries, while this operation:

```
config?op=log-enable&name=textsearchrelrankverbose&name=textsearchspellverbose
```

turns on verbose logging for both the text search relevance ranking and spelling features.

However, this operation:

```
config?op=log-enable&name=allmylogs
```

returns an “Unsupported logging setting” message.

log-disable

The log-disable operation turns off verbose logging.

/config?op=log-disable with no arguments returns the same output as log-status.

log-status

The log-status operation returns a list of all logging variables with their values (true or false).

For example, if you have enabled verbose logging on two of the features, you would see a message similar to the following:

Logging settings:

```
verbose - FALSE
requestverbose - TRUE
updateverbose - FALSE
recordfilterperfverbose - FALSE
textsearchrelrankverbose - TRUE
textsearchspellverbose - FALSE
dgraphperfverbose - FALSE
dgraphrefinementgroupverbose - FALSE
```

help

/config?op=help returns the usage page for all of the config operations.



Chapter 7

Managing the Merge Policy

This chapter describes how to set and manage an MDEX Engine's merge policy.

Using a merge policy for incremental updates

A merge policy for the MDEX Engine determines how frequently it merges incremental update generations in its index.

The data layer that stores the index of the MDEX Engine as a versioning data store. As a result:

- Old versions can be accessed while new versions are created.
- Old versions are garbage-collected when no longer needed.

A version is persisted as a sequence of generation files. A new version appends a new generation file to the sequence. Query latency depends, in part, on the number and size of generation files used to store the index.

Generation files are combined through a process called *merging*. Merging is a background task that does not affect MDEX Engine functionality, but may affect its performance. Because of this, you can set a *merge policy* that dictates the aggressiveness of the merges. In a clustered environment, merge policy can be set on the leader node only.

Types of merge policies

You can set the merge policy to one of two settings: balanced or aggressive.

- **Balanced:** This policy strikes a balance between low latency and high throughput. This is the default policy of the MDEX Engine.
- **Aggressive:** This policy merges frequently and completely to keep query latency low at the expense of average throughput.

The balanced policy is recommended for the majority of applications. However, aggressive merging may help those applications that meet the following criteria:

- Query latency is the primary concern.
- A large fraction of the records (for example, 20%) are either modified or deleted by incremental updates before re-baselining.
- The time to perform an aggressive merge is less than the time between incremental updates.



Note: Under normal conditions, you do not need to change the default balanced policy. However, you may need to change to an aggressive policy based on a recommendation from Endeca Support.

Setting or changing the merge policy

The `mdex-config_MergePolicy` attribute in the system's Global Configuration Record (or GCR) sets the merge policy for the MDEX Engine.

You can set the merge policy with the Configuration Web Service API.

In addition, you can use the URL `merge` command to change the merge policy of a running MDEX Engine or to force a merge.

If you are running a cluster of MDEX Engine nodes, changing the merge policy (either through a Configuration Web Service request or with the `merge` command) can be performed on the leader node only.

Both of these methods are discussed in the following topics.

Setting the merge policy with the Configuration Service API

You can get and set the merge policy with API calls.

The following two topics describe how to use the Configuration Web Service to programmatically get and set the merge policy in the GCR.

Getting the merge policy programmatically

You can retrieve the MDEX Engine's Global Configuration Record to see the current setting for the merge policy.

To programmatically retrieve the Global Configuration Record:

1. Use a URL command similar to the following example to make certain that the Configuration Web Service is running on the MDEX Engine. You should see `config` as one of the available Web services.

```
http://localhost:5555/ws
```

2. Use the `getGlobalConfigRecord` function retrieve the Global Configuration Record via the Configuration Web Service, as in this example:

In a cluster of MDEX Engine nodes, this request should be sent to the leader node only.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <config:configTransaction
      xmlns:config="http://www.endeca.com/MDEX/config/services/config/2010">
      xmlns:mdex="http://www.endeca.com/MDEX/XQuery/2009/09">
        <config:getGlobalConfigRecord />
      </config:configTransaction>
    </soap:Body>
  </soap:Envelope>
```

The `results` response from the Conversation Web Service should look like this example (the SOAP elements have been removed):

```

<config-service:results
  xmlns:config-service="http://www.endeca.com/MDEX/config/services/config/2010">
  <mdex:globalConfigRecord xmlns:mdex="http://www.endeca.com/MDEX/XQuery/2009/09">
    <mdex:record xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      ...
      <mdex-config_Key type="mdex:string">global</mdex-config_Key>
      <mdex-config_MergePolicy type="mdex:string">balanced</mdex-config_MergePolicy>
      ...
    </mdex:record>
  </mdex:globalConfigRecord>
</config-service:results>

```

In this example, the merge policy is set to balanced for this MDEX Engine.

Setting the merge policy programmatically

You can programmatically set the merge policy for the MDEX Engine by updating the Global Configuration Record.

To set the merge policy in the Global Configuration Record:

1. Use a URL command (similar to the following example) to make certain that the Configuration Web Service is running on the MDEX Engine. You should see `config` as one of the available Web services.

```
http://localhost:5555/ws
```

2. Use the `putGlobalConfigRecord` function to set the value of the `mdex-config_MergePolicy` attribute in the Global Configuration Record, as in this example that changes the merge policy to aggressive (note that all attributes must be put, but the example omits most of them for the sake of clarity):

In a cluster of MDEX Engine nodes, this request should be sent to the leader node only.

```

<config:configTransaction
  xmlns:config="http://www.endeca.com/MDEX/config/services/config/2010"
  xmlns:mdex="http://www.endeca.com/MDEX/XQuery/2009/09">
  <config:putGlobalConfigRecord>
    <mdex:record xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      ...
      <mdex-config_Key type="mdex:string">global</mdex-config_Key>
      <mdex-config_MergePolicy type="mdex:string">aggressive</mdex-config_MergePolicy>
      ...
    </mdex:record>
  </config:putGlobalConfigRecord>
</config:configTransaction>

```

The `results` response from the Configuration Web Service should look like this example (the SOAP elements have been removed):

```
<config-service:results
  xmlns:config-service="http://www.endeca.com/MDEX/config/services/config/2010"/>
```

Changing the merge policy of a running MDEX Engine

The URL `merge` command can be used to change the merge policy of a running MDEX Engine.

The sticky version of the `merge` command is intended to change the merge policy of a running MDEX Engine. The duration of the policy change is for the life of the current Dgraph process (that is, until the MDEX Engine is restarted) or until another sticky change is performed during the current Dgraph process.

The format of the sticky version of the command is:

```
/admin?op=merge&mergepolicy=<policy>&stickymergepolicy
```

where `policy` is either balanced or aggressive.

The command also performs a merge operation if warranted.

This example:

```
http://localhost:8000/admin?op=merge&mergepolicy=aggressive&stickymergepolicy
```

forces a merge operation (if one is needed) and changes the current merge policy to an aggressive policy.

Forcing a merge

The URL `merge` command can also be used to force a merge.

Manually forcing a merge is considered a one-time version, because after the merge operation is performed (via a temporary aggressive change to the merge policy), the merge policy reverts to its previous setting.

The one-time version of the `merge` command is used to perform a complete merge of all generations without making a change to the default merge policy.

In a cluster of nodes, you can use this command on the leader node only.

The format of the one-time version of the command is:

```
/admin?op=merge&mergepolicy=<version>
```

The following example assumes that the MDEX Engine is using a balanced merge policy, and you want to temporarily apply an aggressive policy so that the merging can be performed.

```
http://localhost:8000/admin?op=merge&mergepolicy=aggressive
```

When you issue the command, the resulting Web page will look like this example:

```
Dgraph admin, OK.
Dgraph Manual merge started at Sat March 26 09:52:47 2011
```

After the merging is performed, the merge policy reverts to its previous setting.



Chapter 8

MDEX Engine Process Management

This chapter describes how to control the MDEX Engine process from the Windows Services utility or the Linux `inittab`.

Running the MDEX Engine as a Windows service

You can create a Windows service for running the MDEX Engine in service mode.

The Windows SC tool (`sc.exe`) communicates with the Windows Service Controller and installed Windows services. The SC tool allows you to create a Windows service for the MDEX Engine. You can then start and stop the MDEX Engine from the Windows Services utility, as well as make configuration changes (such as configuring the service to automatically restart in case of a failure).



Note: The SC tool (`sc.exe`) is case-insensitive.

For more information, refer to these Web pages on the Microsoft site:

- For more information on creating Windows services: <http://support.microsoft.com/kb/251192>
- For more information on the `sc.exe` command:
<http://technet.microsoft.com/en-us/library/bb490995.aspx>

SC Create command syntax

This topic describes the various options of the SC command with the Create command option.

The SC command communicates with the Windows Service Controller and installed services. When used with its `create` command option, you can use it to create a Windows service under which the MDEX Engine will run.

The SC `Create` command uses the following format:

```
sc [remoteServername] create Servicename
  binpath= "path\to\dgraph.exe dgraphFlags path\to\mdex_db"
  [Optionname= Optionvalue...]
```

where:

- `remoteServername` is an optional parameter that specifies the name of the server if you want to run the command on a remote computer. The name must start with two backslash (\) characters. Do not use this parameter if you are running SC on the local computer.

- `create` is the command to be run by `SC` (this command name is mandatory to create a service).
- `Servicename` is the name of the Windows service to be created. This is the name given to the service key in the registry. Note that this name is different from the display name.
- `binpath` is a mandatory parameter that specifies information for the `dgraph.exe` command.
- `Optionname` specifies optional parameters, which are described in the table below.

The `binpath` parameter specifies this information for the `dgraph.exe` command:

- The absolute path to the `dgraph.exe` command.
- The Dgraph flags used when the MDEX Engine is started. Note that you must use the `Dgraph --out` flag when the MDEX Engine is run in service mode.
- The absolute path to the Dgraph database (that is, the database created by the `mkmdex` utility). Be sure to use the same database name that was supplied to `mkmdex` (that is, do not use the `_indexes` suffix that was added by `mkmdex`).

A space must be used between the `binpath` parameter and its argument. You should also use double quotes around the argument.

SC Create options

You can use these `SC Create` options to further customize the Windows service. Note that the option name includes the equal sign, and a space is required between the equal sign and the option value.

Option Name/Values	Meaning
<code>type= <serviceType></code>	The type of service to be created. Use the <code>own</code> parameter value, which means the service runs in its own process. It does not share an executable file with other services. This is the default for the <code>sc create</code> command. Note that other service types are available, but you should use the <code>own</code> value.
<code>start= <startType></code>	The start type for the service: <ul style="list-style-type: none"> • <code>auto</code> – A service that automatically starts each time the computer is restarted. • <code>demand</code> – A service that must be manually started. This is the default value if <code>start=</code> is not specified. <code>demand</code> maps to <code>Manual</code> in the Services Control Manager. • <code>delayed-auto</code> – The SCM supports delayed auto-start services to improve system performance at boot time without affecting the user experience. The SCM makes a list of delayed auto-start services during boot and starts them one at a time after the delay has passed, honoring dependencies. There is no specific time guarantee as to when the service will be started. • <code>disabled</code> – A service that cannot be started. To start a disabled service, change the start type to another start value.
<code>error= <errorSeverity></code>	The severity of error if the service does not start during boot: <ul style="list-style-type: none"> • <code>normal</code> – The error is logged and a message box is displayed informing the user that a service has failed to start. System startup will continue. This is the default setting. • <code>severe</code> – The error is logged (if possible). The computer attempts to restart with the last-known-good configuration.

Option Name/Values	Meaning
	<p>This could result in the computer being able to restart, but the service may still be unable to run.</p> <ul style="list-style-type: none"> critical – The error is logged (if possible). The computer attempts to restart with the last-known-good configuration. If the last-known-good configuration fails, system startup also fails, and the boot process halts with a Stop error. ignore – The error is logged and startup continues. No notification is given to the user beyond recording the error in the Event Log.
group= <loadOrderGroup>	Name of group of which this service is a member. The list of groups are stored in the registry under the ServiceGroupOrder key. Default is null.
tag= yes no	Do not use this parameter as tags are used only for device driver service types.
depend= <dependencies>	Names of services or groups that must start before this service. Each name is separated by / (forward slash).
obj= <accountName>	Name of the account under in the service will run. The specified account must exist and must be a valid account. Default is LocalSystem .
password= <password>	Password of the obj account. A password is required if an account other than the LocalSystem account is used.
displayname= <displayName>	A friendly, meaningful name that can be used in user-interface programs to identify the service to users. For example, if the service name is MService , you can specify Endeca MDEX Engine as the display name so that will be more meaningful when shown in the Windows Services Control Manager.

SC Create example

The following SC Create example creates a Windows service for the MDEX Engine (note that the command is on one line, but is indented here for ease of reading):

```
sc create MDEXService displayname= "Endeca MDEX Engine"
  type= own error= severe obj= "CORPDEV\EndecaUser" password= banx912
  binpath= "c:\endeca\latitude\2.1.0\mdex\bin\dgraph.exe --port 5555
            --threads 4
            --pidfile c:\mdex_db\dgraph.pid
            --log c:\mdex_db\dgraph.log
            --out c:\mdex_db\dgraph.out c:\mdex_db\mdexdb"
```

The sample command does the following:

- Creates a Windows service named **MDEXService**.
- Uses **Endeca MDEX Engine** as the display name for the service.
- Sets the service type as **own** (which means the service runs in its own process).
- Sets **severe** as the severity of error if the service does not start during the boot process.
- Specifies that the service run under the **CORPDEV\EndecaUser** user account, which has **banx912** as its password.

- Sets the binary path of the `dgraph.exe` executable and specifies `c:\mdex_db\mdexdb` as the MDEX Engine database prefix. Also specifies the locations of the `dgraph` and `error` logs and the `Dgraph` PID file.

For ease of use, you can place the command in a batch script.

Creating the MDEX Engine Windows service

Use the `SC` command's `Create` option to create the MDEX Engine Windows service.

Before running this create-service procedure, make sure that you have Administrator rights.

When creating the service, you must specify the `Dgraph --out` flag as part of the `SC Create` command's `binpath` parameter. Failure to do so will result in the MDEX Engine not being able to start.

To create a Windows service for the MDEX Engine:

1. Click on the **Start** button in the Windows taskbar.
2. Locate the Command Prompt menu item and right-click on the Command Prompt.
3. On the pop-up right click context menu, select **Run as administrator**.
4. In the Command Prompt, enter the `SC Create` command with the appropriate options.

If the command was successful, the SCM will return this message:

```
[SC] CreateService SUCCESS
```

If the command was not successful, the SCM may return this message:

```
[SC] OpenSCManager FAILED 5:
```

```
Access is denied.
```

If you do receive this error, verify that you are a member of the Administrators group on the machine (for example, by using the Microsoft Management Console). If you do have Administrator rights, check that you are opening the Command Prompt with the **Run as administrator** option.

After the service has been created, you can use the `SC Config` command to change any parameter set by the `SC Create` command.

Setting a service description

Use the `SC` command's `Description` option to set a description for the MDEX Engine Windows service.

Before adding a description to the service, make sure that you have Administrator rights.

When you create a service with the `SC Create` command, you cannot set a service description. However, after creating the service, you can use the `SC` command's `Description` option to add a new description or to modify an existing description.

The format of the `SC Description` command is:

```
sc description Servicename descriptionText
```

where `Servicename` is the name of the service to modify and `descriptionText` is the new description within double quotes. There is no limit to the number of characters that can be contained in the service description.

To add or modify the description of the MDEX Engine Windows service:

1. Stop the MDEX Engine Windows service.
2. Click the **Start** button in the Windows taskbar.
3. In the menu, right-click **Command Prompt**.
4. On the pop-up right click context menu, select **Run as administrator**.
5. At the command prompt, enter the `sc description` command with the service name and new description, as in this example, which sets a description for the MDEXService:

```
sc description MDEXService "Provides high-performance search."
```

If the command was successful, the SCM will return this message:

```
[SC] ChangeServiceConfig2 SUCCESS
```

Modifying the service configuration

Use the `SC` command's `Config` option to modify the configuration of the MDEX Engine service.

Before attempting to modify the service configuration, make sure that you have Administrator rights.

After you create the MDEX Engine service with the `SC Create` command, you can use the `SC Config` command to modify the service configuration. Because both commands use the exact same set of parameters, any parameter that you set with `SC Create` can be modified with `SC Config`. The command is especially useful when you want to add or remove Dgraph flags from the current `binpath` setting.

The format of the `SC Config` command is:

```
sc [remoteServername] config Servicename Optionname= Optionvalue...
```

where `Servicename` is the name of the existing MDEX Engine Windows service to be modified.

When using the `SC Config` command, you specify only the parameter settings that will be changed. Any parameter setting that is not specified will remain as-is in the service configuration. Note that to change the service description, you must use the `SC Description` command.

To modify the MDEX Engine Windows service:

1. Stop the MDEX Engine Windows service.
2. Click the **Start** button in the Windows taskbar.
3. In the menu, right-click **Command Prompt**.
4. On the pop-up right click context menu, select **Run as administrator**.
5. At the command prompt, enter the `SC Config` command with the service name to be modified and the parameters to be changed, as in this example that adds a Dgraph flag to the `binpath` configuration:

```
sc config MDEXService binpath= "c:\endeca\latitude\2.1.0\mdex\bin\graph.exe
--port 5555 --ancestor_counts --pidfile c:\mdex_db\dgraph.pid --threads
4
--log c:\mdex_db\dgraph.log --out c:\mdex_db\dgraph.out c:\mdex_db\mdexdb"
```

If the command was successful, the SCM will return this message:

```
[SC] ChangeServiceConfig SUCCESS
```

Deleting the MDEX Engine Windows service

Use the SC command's Delete option to remove the MDEX Engine Windows service.

Before deleting the service, make sure that you have Administrator rights.

The format of the SC Delete command is:

```
sc delete Servicename
```

where *Servicename* is the name of the service to be deleted.

To delete the MDEX Engine Windows service:

1. Stop the MDEX Engine Windows service.
2. Click the **Start** button in the Windows taskbar.
3. In the menu, right-click **Command Prompt**.
4. On the pop-up right click context menu, select **Run as administrator**.
5. At the command prompt, enter the SC Delete command with the service name to be deleted, as in this example:

```
sc delete MDEXService
```

If the command was successful, the SCM will return this message:

```
[ SC] DeleteService SUCCESS
```

If the command was not successful, the SCM may return this message:

```
[ SC] OpenService FAILED 5:
```

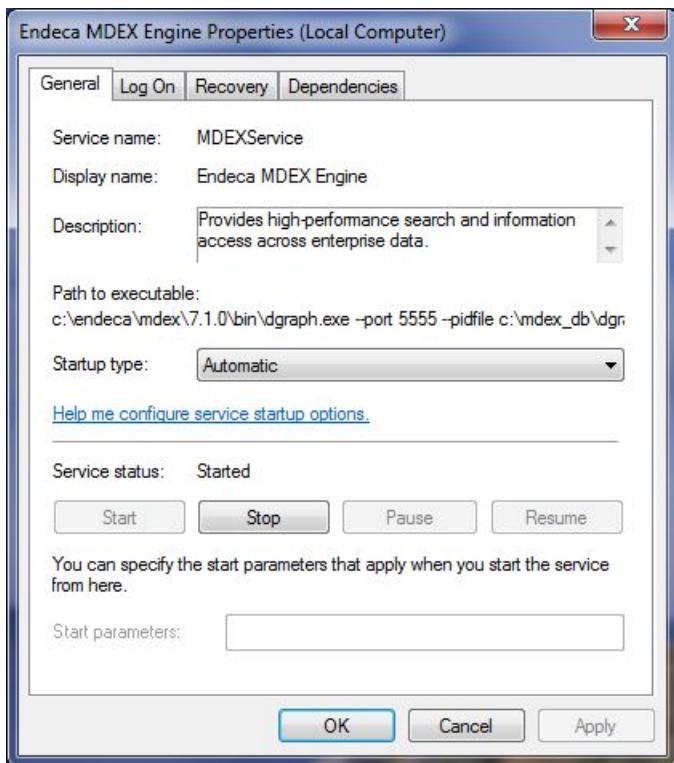
```
Access is denied.
```

If you do receive this error, first verify that you are a member of the Administrators group on the machine. If you do have Administrator rights, check that you are opening the **Command Prompt** with the **Run as administrator** option.

Using the Windows Services utility

The Windows Services utility allows you to control and configure the MDEX Engine service.

The MDEX Engine service, when selected in the Windows Services utility, looks like this example:



General tab

The **General** tab allows you to start and stop the MDEX Engine service. Either operation will log an appropriate message to the MDEX Engine's stdout/stderr log.

Clicking the **Stop** button sends a shutdown request (with time limit of 120 seconds) to the MDEX Engine. The shutdown will complete when all outstanding jobs are complete, or within 120 seconds, whichever happens first. Unfinished jobs are handled as follows:

- A query still in progress when the time limit is reached will not return a result to the client and will not be logged in the Dgraph log.
- An update still in progress when the time limit is reached will not be applied.
- A background merge still in progress when the time limit is reached will be aborted at the end of the timeout.
- The number of in-progress queries is written to the Dgraph error log just before exiting, along with a message stating that the shutdown time limit was reached.

You can use the **Startup type** drop-down menu to change the startup type to Automatic, Automatic (Delayed Start), Manual, or Disabled. Clicking the help link displays usage information for this option.

Note that the **Start parameters** field has no effect on the service.

Log On tab

The **Log On** tab allows you to change the account under which the MDEX Engine service runs. This option is especially useful if you created the service to run under the Local System account and want to change to a user account. The tab has a help link that provides detailed information on configuring the user account log on options.

Recovery tab

The **Recovery** tab is used to configure recovery actions when a service fails. You can configure the MDEX Engine service for automatic restart. That is, the MDEX Engine service will restart in the case of a crash or machine reboot.

To obtain information on how to configure the computer's response if the MDEX Engine fails, click the "Help me set up recovery actions" link on the tab.

Logging in service mode

MDEX Engine logging is supported in service mode.

When the MDEX Engine is run in service mode, it will log startup and shutdown messages to its stdout/stderr log, which is specified by the Dgraph --out flag.

When the service is started:

```
INFO 03/01/11 20:55:44.578 UTC (1299012944577) DGRAPH {dgraph,baseline, service} Starting in service mode.
```

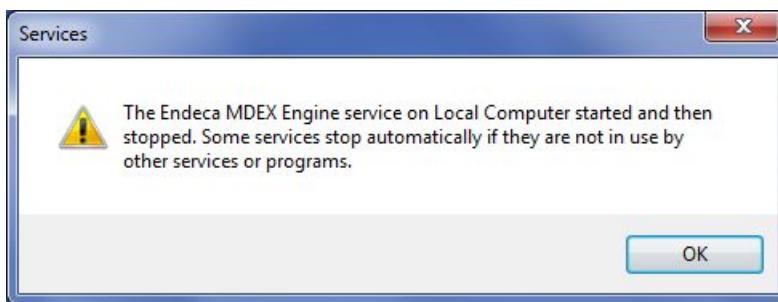
When the service is stopped by the user (such as from the Windows Services utility):

```
INFO 03/01/11 21:32:14.500 UTC (1299015134500) DGRAPH {dgraph,service} Stopping on user request. Shutdown request with time limit of 120 seconds received at Tue Mar 01 16:32:14 2011. Shutdown will complete when all outstanding jobs are complete, or within 120 seconds, whichever happens earlier.
```

When the service is stopped as part of a system shutdown:

```
INFO 03/01/11 21:53:41.469 UTC (1299016421469) DGRAPH {dgraph,service} Stopping on system shutdown. Shutdown request with time limit of 15 seconds received at Tue Mar 01 16:53:41 2011. Shutdown will complete when all outstanding jobs are complete, or within 15 seconds, whichever happens earlier.
```

If Dgraph stdout/stderr is not redirected to a file in service mode, all log messages will be lost. Therefore, when the MDEX Engine is run in service mode, the Dgraph --out flag is required. If the Dgraph --out flag is not supplied, the service will not start. If you try to start it from the Windows Services utility, Windows displays this error message:



To recover from this situation, use the SC Config command to modify the binpath parameter of the service and add the --out flag.

Starting the MDEX Engine from inittab

In a Linux production environment, the MDEX Engine can be started by `init` from `inittab`.

In a Linux development environment, the MDEX Engine can be started from the command line. In a Linux production environment, however, Endeca recommends that it be started by `init` from `inittab`. If the service crashes or is terminated, `init` automatically restarts it.

The `inittab` entry should be formatted like this:

```
dg:2345:respawn:/bin/su - <dgraph_user> -c "/absolute/path/to/bin/dgraph <dgraph_flags> <mdex_indices>"
```

where:

- **dg** is the `inittab` entry identifier.
- **2345** lists the runlevels for which the specified action should be taken.
- **respawn** is the action to be taken, which is that the process will be restarted whenever it terminates.
- **/bin/su** specifies the process to be executed. In this case, a non-root user will run the `dgraph` command. It is a best practice to run the Dgraph as a user other than root.
- **-c dgraph <dgraph_flags> <mdex_indices>** specifies that the `dgraph` command will be run with the specified Dgraph flags, using the absolute path to the Dgraph database (that is, the database created by the `mkmdex` utility).

Note that you must use the Dgraph `--out` flag to direct `stdout/stderr` output to a log file.



Chapter 9

Deploying Latitude in a Cluster

This section discusses how to deploy a Latitude application in a cluster with multiple nodes hosting the MDEX Engine instances.

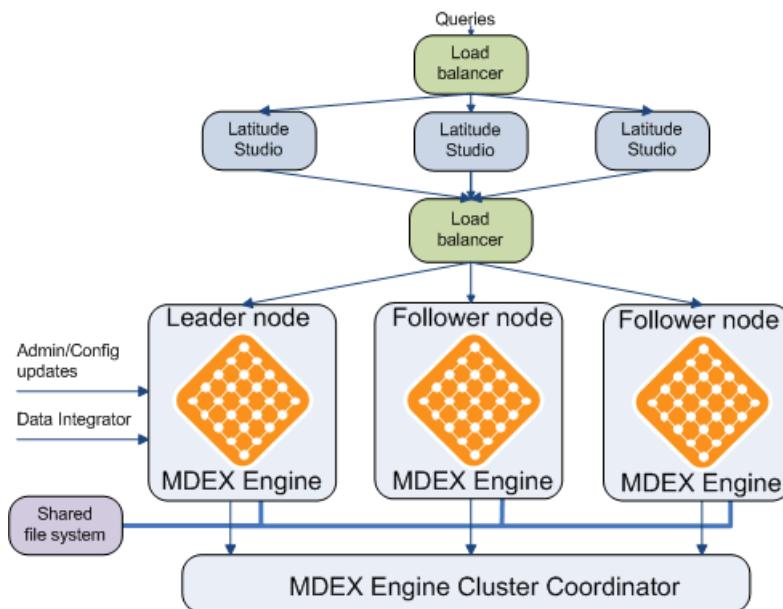
Cluster overview

This topic introduces the cluster of MDEX Engine nodes and describes its capabilities.

About the cluster

A *cluster* is composed of a set of MDEX Engine nodes. All nodes can serve query requests. Only one node is identified as the leader node; All other nodes are follower nodes. All of the MDEX Engine nodes share and use one copy of the on-disk representation of the MDEX Engine index.

The Cluster Coordinator provides communication between the nodes in the cluster. The Cluster Coordinator is also used to notify the follower nodes about index updates and updates to the configuration.



Cluster capabilities

A cluster of MDEX Engine nodes provides the following capabilities:

- **Enhanced availability of query processing by the MDEX Engine.** In a cluster, if one of the nodes fails, queries continue to be processed by other nodes in the cluster.
- **Increased throughput by the MDEX Engine.** In a cluster, you change throughput capacity by adding or removing nodes. By adding nodes you can spread the query load across multiple MDEX Engine instances without the need to increase storage requirements at the same rate. You can add or remove nodes dynamically, without having to stop the cluster.

In a cluster, you can perform the following administrative tasks:

- Add one or more MDEX Engine instances to a cluster.
- Remove MDEX Engine instances while allowing the cluster to continue running.
- Identify a single node to which you can send data during an initial index data load and subsequent updates. (The administration and configuration updates must also be sent to this node.) All types of updates are automatically propagated to all MDEX Engine nodes while one or more MDEX Engine nodes continue to process queries.

Latitude cluster architecture

This topic discusses cluster architecture in the development and production environments.

In the development environment, you can start with a simple single-node cluster configuration and expand it by adding more nodes. When you move to a production environment, you can duplicate a multi-node development cluster.

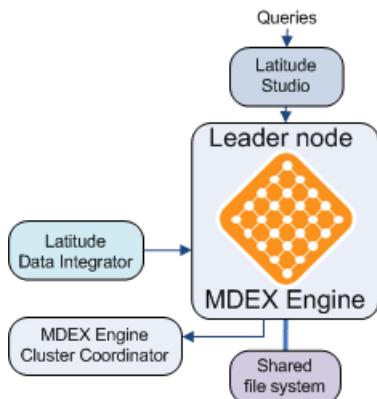
A single-node cluster in the development environment

In a development environment, the simplest version of a cluster may consist of just one node hosting an instance of the MDEX Engine. This node is by definition the leader node — in a single-node cluster, the only node is considered the leader node by default.



Note: You are not required to run a single instance of the MDEX Engine in a cluster. Without the cluster services, having a single running MDEX Engine instance is a valid configuration for starting in the development environment.

This diagram represents a single-node cluster in a development environment:



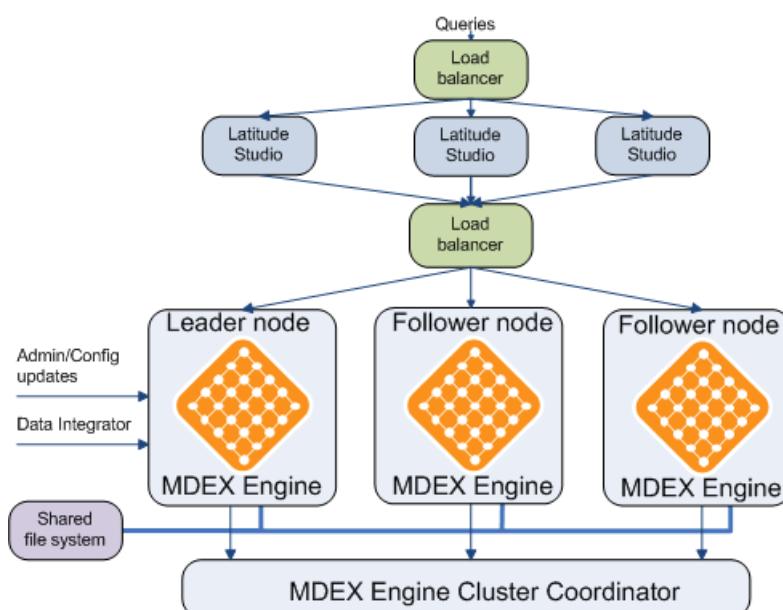
In this diagram:

- The leader node is hosting an MDEX Engine and is receiving query requests from Latitude Studio.
- The leader node's host and port are included in the configuration for connectors from the Latitude Data Integrator, which can send various kinds of data and updates to the MDEX Engine index.
- The leader node (and any additional nodes that you may add later) must have access to a shared file system on which the MDEX Engine index is stored.
- Finally, the cluster is managed by the MDEX Engine Cluster Coordinator. In a single-node cluster, the Cluster Coordinator service runs on the same server on which the MDEX Engine is running.

A multiple-node cluster in the development environment

In the development environment, many cluster configurations are possible; they depend on the requirements for your application. For example, while a single leader node is always required, the number of additional follower nodes hosting the MDEX Engine instances may vary.

This diagram represents a possible multiple-node cluster in a production environment:



In this diagram, starting from the top, the following actions take place:

- The queries are sent to the load balancer that is configured in front of several servers hosting Latitude Studio instances.
- The instances of Latitude Studio point to a second load balancer between Latitude Studio and the MDEX Engine cluster. This load balancer is configured to recognize the leader node and all follower nodes.
- The Latitude Data Integrator is configured to communicate with the host and port of the leader node to ensure a point of communication for sending data and updates.
- All nodes in the cluster communicate with each other through the MDEX Engine Cluster Coordinator. The Cluster Coordinator service must be running on the leader node.
- All nodes in the cluster have access to a shared file system on which is stored a shared index for the MDEX Engine nodes.

Important cluster concepts

This topic introduces the leader and follower nodes and the Cluster Coordinator.

In a cluster composed of nodes, each of which hosts an MDEX Engine instance, the following definitions are used:

- Leader node
- Follower node
- Cluster Coordinator

Leader node

A single node in the cluster responsible for processing queries and for receiving updates to the index and to the configuration. This node is responsible for obtaining information about the latest index and propagating this information to the follower nodes through the Cluster Coordinator.

The leader node has the following characteristics:

- Each cluster must have one and only one leader node.
- The leader node must have write access to the same shared file system on which the MDEX Engine index is stored and to which all follower nodes also have access.
- The Cluster Coordinator service must be running on the leader node.
- The entities outside the cluster of MDEX Engine nodes (such as connectors in the Latitude Data Integrator and components of Latitude Studio) must be able to access the leader node.

The leader node periodically receives full or incremental index updates from the Latitude Data Integrator. It also receives administration or configuration updates. It is the only node in the cluster that has access with "write" permissions to the on-disk representation of the MDEX Engine index.

Once the leader node acquires access to the new version of the MDEX Engine index, it updates the index, adding new information to it and deleting information that has become obsolete.

The Cluster Coordinator notifies all follower nodes, alerting them to start using the updated index. The follower nodes acquire read-only access to an updated index.

Follower node

A node in the cluster responsible for processing queries. The follower node does not update the index, although it has read-only access to its latest copy.

The follower node has the following characteristics:

- Each cluster can have more than one follower node.
- In a single-node cluster, a leader is also a follower.
- Each follower node must have a unique name across the cluster. The name also must be a valid directory name (characters such as slashes (/) are not allowed).
- All follower nodes must reference the host name and port of the Cluster Coordinator service.
- All nodes (including the follower nodes) must have write access to the same shared file system on which the MDEX Engine index is stored.

During the process of acquiring access to the recently updated index, both the follower and the leader nodes continue to serve queries. Each query is processed against a specific version of the index that is available to a cluster node at any given time. Query processing performance may slow down as the follower nodes acquire read-only access to the updated index.

Cluster Coordinator

An entity that provides a mechanism for the MDEX Engine nodes to communicate with each other.

The Cluster Coordinator controls the heartbeat function, the file sharing function between the cluster nodes, and the propagation of updates to the follower nodes once the leader node acquires access to an updated MDEX Engine index.

Before you begin

This section discusses requirements for installation related to deploying a cluster, as well as tips for planning your cluster architecture.

System and hardware requirements

This section outlines the operating system and hardware requirements for deploying Latitude in a clustered environment.

Operating system requirements

A cluster of MDEX Engine nodes can be deployed on either Windows or Linux.

You cannot create a cluster in which some nodes are running on Windows while other nodes are running on Linux.

Shared file system requirements

A cluster of MDEX Engine nodes must use the index stored on a shared file system. This topic describes the requirements for the shared file system.

The requirements and best practices for a shared file system are the following:

- All nodes in the cluster must have write access to a shared file system. The on-disk representation of the MDEX Engine index must reside on this file system. On Linux, any shared file system that supports hard links, such as NFS, can be used.
- File system size. You can start a cluster with a single node that serves both as the leader and a follower node. As you add additional follower nodes, file system size requirements (as measured by the high-water mark parameters for shared storage) increase modestly and do not increase proportionally to the number of follower nodes.
- Performance. Even in a single-node cluster, using an index on remote storage affects node startup time and performance associated with processing of index updates. In a multi-node cluster, all MDEX Engine nodes are accessing the index at the same time. This coordinated access may affect performance for the network or shared file system, especially when large updates are accessed for the first time.

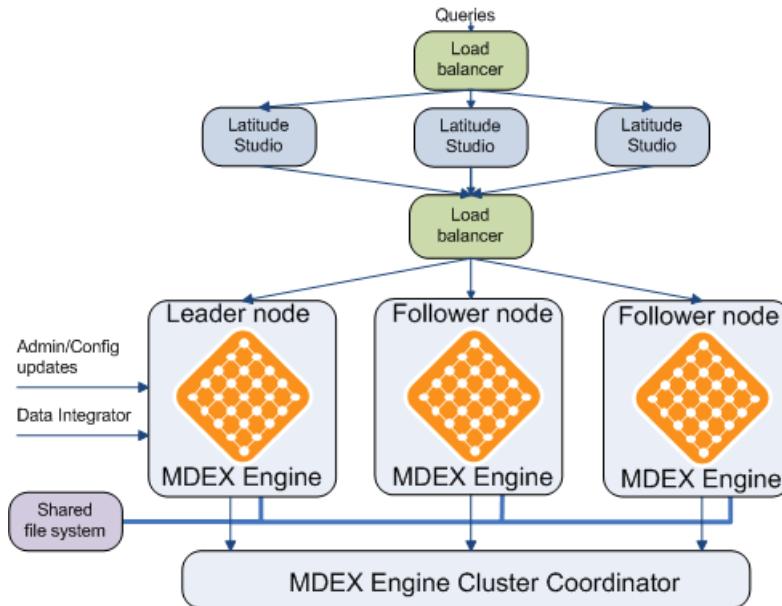
Load balancer requirements

In most production deployments, it is desirable to configure a load balancer between Latitude Studio and a cluster of the MDEX Engine nodes. This topic discusses the load balancer considerations associated with the Latitude cluster.

The following diagram of a typical multi-node cluster shows two load balancers:

- A load balancer in front of a number of servers hosting Latitude Studio
- A load balancer between Latitude Studio and the cluster of MDEX Engine nodes

This topic discusses the load balancer between Latitude Studio and the MDEX Engine cluster. In the diagram, it is the load balancer which is second from the top. (This diagram is also used in other locations in the cluster section of this guide).



The following considerations apply to the load balancer between Latitude Studio and a cluster of MDEX Engine nodes:

- Include host names and ports of all nodes into the load balancer configuration. This ensures that regular query-type (non-updating) requests from Latitude Studio are sent to any of the nodes in the cluster.



Note: Latitude Studio data sources that send both updating and non-updating requests to a cluster should include not only the load balancer's host name and port, but also an update host name and update port that reference the leader node. If the leader node changes, the configuration of the data source must be updated, but the load balancer's configuration does not need to change.

- If you add nodes to the cluster, you must update the configuration of the load balancer with the host names and ports of the added nodes.
- You may optionally configure the load balancer to use session affinity. In this case, all MDEX Engine queries from a given Latitude Studio session end up on the same MDEX Engine. This allows the MDEX Engine to use its cache to avoid redundant processing on related queries from one page view.

A series of page views also benefits from cache contents. For instance, a record search that remains in the filter state through multiple page views will not have to be calculated repeatedly for each click.

Configuring session affinity also helps minimize consistency problems as updates propagate to nodes in the cluster.

Related Links

[Connecting a cluster with a load balancer](#) on page 67

The load balancer between a cluster and one or more Latitude Studio servers must be aware of all nodes in the cluster. In addition, all data sources in Latitude Studio must reference the host name and port of the load balancer server.

Downloading, installing and starting the Cluster Coordinator

The cluster coordinator provides a mechanism for the MDEX Engine nodes to communicate with each other while ensuring high availability of the MDEX Engine.

The Cluster Coordinator has the following characteristics:

- It is a shared information repository that provides a set of distributed coordination services. It ensures that all systems in the cluster coordinate their actions relative to all other systems running in your environment. If one of the nodes communicates any cluster information, all other nodes recognize it and react to it in manner that ensures synchronization, event notification, and coordination between the nodes.

The communication and coordination mechanisms continue to work in the case when connections or cluster nodes fail.

- The Cluster Coordinator logs messages using the `log4j` file included with its installation in the `/conf` directory.

When the Cluster Coordinator service is running on a node, log messages can be logged to the console (default) and/or a log file depending on the `log4j` configuration.

Deployment strategy for the Cluster Coordinator service and MDEX Engine nodes

To create a cluster, the Cluster Coordinator service must be running on at least one node. Endeca recommends that you start a single Cluster Coordinator service on the node that will be designated as the leader node, and then start the MDEX Engine on all nodes by referencing the host and port of the Cluster Coordinator.



Note: The port for the Cluster Coordinator is a dedicated port used for cluster node communication.

Downloading the Cluster Coordinator package

You can download the Cluster Coordinator package from the Downloads section of the Endeca Developer Network (EDeN).

Download and install the Cluster Coordinator package on one server you plan to include in a cluster as a leader node.

To download the Cluster Coordinator package:

1. If you have not previously done so, establish a Support account with download access through the Support section of the Endeca Developer Network (EDeN) at <http://eden.endeca.com>. This enables the Endeca Support and Customer Care groups to track which versions of the software you are using.
2. On the EDeN homepage, click **Downloads**.
3. On the **Tools and Utilities** page, find the **Product Downloads** section, then click **View and download purchased products**.

4. On the **Product Downloads** page, click **Latitude <version>**.
5. In the **Current Releases** table, click **Latitude <version>**.
6. In the **Product Download** page, download the **MDEX Engine Cluster Coordinator** software for Windows or Linux.

Now that you have downloaded this package, you can install it on the node in the cluster that you will designate as the leader node.

Installing the Cluster Coordinator package

To run in a cluster, the Cluster Coordinator package must be installed and the Cluster Coordinator service must be running on the one cluster node that is designated to be the leader node.

It is assumed that you have downloaded the Cluster Coordinator package and also have installed the MDEX Engine on the node.

You install the Cluster Coordinator in the `endeca/Latitude` directory. If you installed the MDEX Engine, this directory should already exist.

To install the Cluster Coordinator on the leader node:

Unzip the following packages to your `endeca/Latitude` directory:

Windows	<code>Latitude_<version>_cluster_coordinator_x86_64pc-win.tgz</code>
Linux	<code>Latitude_<version>_cluster_coordinator_x86_64pc-linux.tgz</code>

where `<version>` is the version of the Endeca Latitude software package.

The Cluster Coordinator package is installed in `endeca/Latitude/<version>/ClusterCoordinator` on Linux. For Windows, the path is `Endeca\Latitude\<version>\ClusterCoordinator`.

Once the Cluster Coordinator package is installed on the leader node, its service can be started.

Starting and stopping the Cluster Coordinator service

To ensure that MDEX Engine nodes can function together in a cluster, the Cluster Coordinator service must be running on the leader node and all the nodes must be started with references to the host name and port of the Cluster Coordinator service.

You start the Cluster Coordinator service on the leader node only.

It is assumed that the Cluster Coordinator package has been downloaded and installed on the leader node.

To start or stop the Cluster Coordinator service:

1. On the server that will serve as the leader node, go to the `Latitude\<version>\ClusterCoordinator\bin` directory and locate the `clusterCoordinator` script.

The script has a different extension for Windows and Linux.

2. From this directory, run the script as follows:

- To start, run:

```
clusterCoordinator start
```

- To stop, run:

```
clusterCoordinator stop
```



Note: If you provide a full path to the script on the command line, you can also run it from another directory. In this case, the Cluster Coordinator creates its `dataDir` in the path from which you run the script.

When the Cluster Coordinator is started on a server, it uses the host name of this server, and the port 2181. (You can change the Cluster Coordinator configuration to use another port, using its configuration file.)

After the Cluster Coordinator service has been started on a node, you can start the MDEX Engine on this node. If you start the MDEX Engine without the `--replica-name` flag, and with the flags that specify the Cluster Coordinator host name and port, this node becomes the leader node in the cluster.

To ensure that follower nodes can establish a connection with the Cluster Coordinator service, when you start the MDEX Engine on follower nodes, in addition to the `--replica-name` flag, specify the host name and port of the Cluster Coordinator service to the `dgraph` command.

The configuration file for the Cluster Coordinator

The Cluster Coordinator service uses a configuration file which specifies the settings for it.



Note: This topic provides reference information about this file. For the cluster to work, this file does not require any modifications.

After the installation of the Cluster Coordinator, the configuration file is placed in the `Latitude\<version>\ClusterCoordinator\conf` directory. When you run the `clusterCoordinator` script, it detects the configuration file.

Format

The configuration file should have the following format:

```
# Modified and renamed from source by
# Endeca Technologies

# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgment
syncLimit=5
# the directory where the Cluster Coordinator snapshot is stored.
dataDir=
# the port at which the clients will connect
clientPort=2181
```

Where:

Entry	Description
tickTime	<p>The basic time unit in milliseconds used by the Cluster Coordinator. It is used for heartbeats.</p> <p>For example, <code>tickTime</code> can be 2000 milliseconds. The minimum session timeout is twice the <code>tickTime</code>.</p>
initLimit	<p>The number of ticks that the initial synchronization phase can take.</p> <p>This number specifies the length of time the nodes have to connect to the leader node.</p>
syncLimit	<p>The number of ticks that can take place between one node sending a request for an update and receiving an acknowledgment from the leader node.</p>
dataDir	<p>The directory where the in-memory database snapshots for the Cluster Coordinator and the transaction log of updates to its database are stored.</p> <p>This directory is created in the same file system location from which you run the Cluster Coordinator script.</p> <p>You can specify to store the log of updates to the database in another directory, using the <code>log4j</code> file stored in the <code>/conf</code> directory of the Cluster Coordinator installation.</p>
clientPort	<p>The port at which clients should connect to the Cluster Coordinator service.</p> <p>As configured after the initial installation, this port is 2181.</p>

Planning cluster nodes

To plan cluster nodes, you specify which ones will serve as follower nodes. The one node in a cluster that is not a follower is the leader node.

To plan your cluster nodes:

1. Write down each node's host and port.

You will need this information to identify which of your nodes should serve as a leader node and to designate other nodes as follower nodes.

2. Provision a shared file system on which the MDEX Engine index will be stored.

When you will install and start MDEX Engine instances, they should each point to the index on this file system and have write access to it.

3. On the node that you plan to designate as the leader node, reserve the port 2181.

This port is used by the Cluster Coordinator service (unless you configure it to use another port). You will also specify this port when starting all nodes in your cluster.

Cluster behavior

The Cluster Coordinator ensures that the MDEX Engine nodes in the cluster provide query processing that is highly available and stable in the face of individual node failures. This topic discusses cluster

behavior in various scenarios, such as cluster startup, updates to the index, and response to a node failure.

Bringing a cluster online

On startup, the following actions take place:

- Any MDEX Engine node can be started in either a leader or follower mode, and in any order. Any number of follower nodes and exactly one leader node can be added to a cluster.

If you attempt to start two leader nodes in the same cluster, you will receive an error.

- Once started, each node registers with the Cluster Coordinator that manages the distributed state of the cluster.

One node for which you do not specify that it must be a follower is the leader; you identify all other nodes as follower nodes.

- The leader node determines the current version of the index and informs the Cluster Coordinator.

After all follower nodes have started, each of them acquires read-only access to the current version of the index.

- Follower nodes do not alter the index files in any way; they continue answering queries based on the index version to which they have read-only access at startup, even if the leader node is in the process of updating, merging or deleting index files on disk.

Follower nodes refuse all updating web service and HTTP requests (such as `admin?op=updatea-spell`) with a 403 HTTP status code (forbidden).

Processing an update

The leader node is responsible for processing updates. It processes the update and commits it to the on-disk index. The Cluster Coordinator informs all follower nodes that a new index version is available. The leader node and all follower nodes can continue to use files from the previous version of the index to finish query processing that had started against that version.

As each node finishes processing queries on the previous version, it releases references to it. Once the follower nodes are notified of the new version, they acquire read-only access to it and start using it.

Updates take time to propagate across the nodes in the cluster. Because of this, at any given time when updates are running, portions of page views in Latitude Studio could be processed against different versions of the index and thus may be inconsistent. To minimize inconsistency in page views, configure session affinity on the load balancer between a cluster of MDEX Engine nodes and the servers running Latitude Studio.

Responding to a node failure

In a cluster, a follower or a leader node may fail:

- Failure of the leader node. Responding to the expected or unexpected leader node failure is critical for system availability and data consistency.

When the leader node goes off-line, the Cluster Coordinator detects this event, and the follower nodes stop receiving notifications about the new versions of the index. You need to restart the node.

- Failure of a follower node. When one of the follower nodes goes off-line, it is removed from the cluster. The other nodes do not need to keep track of this event.

If the node is restarted, it joins the cluster as a new follower node.

Responding to network failures of the Cluster Coordinator service

If a network connection fails between the nodes in the cluster that connect to the Cluster Coordinator service, the MDEX Engine on those nodes will shut down.

A node will rejoin the cluster once the MDEX Engine on the node is restarted (this will happen automatically if it is run as a service) and is able to establish a connection with the Cluster Coordinator service.

Building a cluster

This section discusses how to build a cluster by starting the leader node and adding follower nodes.

Starting the MDEX Engine as the leader node

When you start any MDEX Engine without specifying that it is a follower node, this node serves as the leader node.

Before starting an MDEX Engine that will serve as the leader node, ensure that the Cluster Coordinator service is running on this node. Note the host name and port of your Cluster Coordinator service, so that you can specify them when starting the MDEX Engine as the leader node.

You configure one and only one leader node in a cluster. Therefore, if you want to start the MDEX Engine as the leader, do not specify the `--replica-name` flag for it. The MDEX Engine instance that is started without the `--replica-name` flag becomes the leader node. You also do not have to specify a name for the leader node.

If you want to change which node is the leader node, you need to:

1. Stop the existing cluster by stopping the nodes and the Cluster Coordinator service.
2. Start the Cluster Coordinator service on another node.
3. Recreate the cluster with another node as the leader, by referencing the new location of the Cluster Coordinator to all nodes.

To start the MDEX Engine as the leader node:

1. Start the MDEX Engine on the node with the `dgraph` command as in the following example:

```
dgraph
--port 5555
--coordinator_port 2181
--coordinator_host My_cluster_coordinator_server.com
--threads 16
--log c:\mdex_db\dgraph1.log
--out c:\mdex_db\dgraph1.out
z:\shared_mdex_db\mdexdbdgraph
```

In this example:

- `z:\shared_mdex_db\mdexdbdgraph` is the location of the MDEX Engine index, which resides on a shared file system. All nodes in your cluster must point to the same location of the index on a shared file system and have write access to it.

- 2181 is the port used by the Cluster Coordinator (if you haven't changed it after installing the Cluster Coordinator).
- `My_cluster_coordinator_server.com` is the host name used by the Cluster Coordinator. This is the host name of the leader node.

Once the MDEX Engine is running on the leader node, this node receives updates to the index and configuration. The Cluster Coordinator propagates updates to the follower nodes.

2. Note the leader node's port and host name.

You will need to reference this information in the configuration for the Latitude Data Integrator, so that the integrator can send data and updates to the leader node. This information should be also useful when configuring the data sources in Latitude Studio.

Now that you have started the leader node, you can add one or more follower nodes.

Starting a follower node

You can add a follower node to the cluster at any time by starting the MDEX Engine with the `--replica-name <node_name>` flag.

Before starting an MDEX Engine that will serve as a follower node, ensure that the Cluster Coordinator service is running on the leader node. Note the host name and port of your Cluster Coordinator service, so that you can specify them when starting the MDEX Engine as a follower node.

The `Dgraph` flag `--replica-name <node_name>` specifies the follower node, where `<node_name>` is the name of the follower node. This name must be unique across the cluster. The name must also be a valid directory name (characters such as slashes (/) are not allowed).



Note: If you start a node without this flag, the Cluster Coordinator assumes this is the leader node. Since there can be only one leader node in the cluster, it is important to start just one node without the `--replica-name <node_name>` flag. In fact, the MDEX Engine will not start if it is asked to be the leader node when a leader node already exists.

To start an MDEX Engine as a follower node:

1. Issue the command as in the following example, specifying a unique name of the follower node in the `--replica-name` flag:

```
dgraph
--port 5556
--threads 16
--replica-name FollowerNode1
--coordinator_port 2181
--coordinator_host My_cluster_coordinator_server.com
--log c:\mdex_db\dgraph2.log
--out c:\mdex_db\dgraph2.out
z:\shared_mdex_db\mdexdbdgraph
```

In this example:

- `z:\shared_mdex_db\mdexdbdgraph` is the location of the MDEX Engine index, which resides on a shared file system.

All nodes in your cluster must point to the same location of the index on a shared file system and have write access to it.

- The `--coordinator_host` and `--coordinator_port` reference the host name and port of the Cluster Coordinator service.

This service uses the host name of the leader node, and the port 2181 (unless you configure the Cluster Coordinator to use another port).

The node `FollowerNode1` is now known to the Cluster Coordinator as a follower node — when changes occur to the on-disk MDEX Engine index, the Cluster Coordinator notifies this node to start using the new version of the index.

2. Proceed by adding additional follower nodes if needed.

For each follower node:

- Specify a different name.
- Reference the host name and port of the Cluster Coordinator service.
- Ensure that the index is referenced in the same location on a shared file system as for other nodes in the cluster.



Note: Since there is one Cluster Coordinator service running on the leader node, you can add more than one follower node, all referencing the same Cluster Coordinator service host name and port.

Summary of operations handled by the leader node and any node

This topic summarizes which specific requests to the MDEX Engine should be directed to the leader node and which can be handled by any node.

Operations on the leader node only

These operations should be directed to the leader node only:

- Updates to data records. If you are adding more records to the MDEX Engine cluster, they should be sent to the leader node.

This means that operations from the Data Ingest Web Service and the bulk load interface should be directed to the leader node only.

- Snapshot operations from the Administrative Web Service. Operations for taking and applying a snapshot should be directed to the leader node only.
- Updating operations from the Configuration Web Service. All requests to the MDEX Engine that require changing schema for the MDEX Engine records or the configuration of the MDEX Engine features should be directed to the leader node.

If such requests are sent by Latitude Studio components, this requirement is achieved by configuring data sources that include the update host name and update port that reference the leader node.

- Administrative operations for the MDEX Engine. The following administrative operations should be directed to the leader node:
 - `/admin?op=merge`
 - `/admin?op=reload-services`
 - `/admin?op=updateaspell`

Operations on any node

The following operations can be directed to any node in the cluster (including any of the follower nodes):

- Any request from the Conversation Web Service (this means any request from Latitude Studio asking for read-only queries against the data).
- Any request from the Administrative Web Service other than snapshot-related operations.
- Any request utilizing the read-only version of the Configuration Web Service.
- Some administrative operations for the MDEX Engine. The following administrative operations can be directed to any node in the cluster:
 - `/admin?op=exit`
 - `/admin?op=flush`
 - `/admin?op=logroll`
 - `/admin?op=stats`
 - `/admin?op=statsreset`

Connecting the leader node with the Data Integrator

The connectors in the Latitude Data Integrator that send data to the MDEX Engine must be configured to reference the host name and port of the leader node.

The MDEX Engine instance running on this node is capable of receiving updates, since it has "write" permissions to the MDEX Engine index.

It is assumed that by this point, you have configured a leader node in the cluster.

To reference the leader node in the Data Integrator configuration:

In the connector's configuration, specify the host name and port of the leader node.

Connecting a cluster with Latitude Studio

In a typical implementation, a cluster of MDEX Engine nodes is connected to one or more Latitude Studio servers through a load balancer. The host and port of this load balancer must be referenced in the data sources for Latitude Studio components. In addition, for any updating operations or queries, Latitude Studio data sources need to include an update host name and update port that reference the leader node.

Connecting a cluster with a load balancer

The load balancer between a cluster and one or more Latitude Studio servers must be aware of all nodes in the cluster. In addition, all data sources in Latitude Studio must reference the host name and port of the load balancer server.

Configuration of the load balancer involves taking care of these high-level tasks:

- To connect the load balancer with a cluster of MDEX Engine nodes, reference the host names and ports of all nodes in the load balancer configuration.
- To connect the load balancer with Latitude Studio servers, each data source in Latitude Studio must specify the host name and port of the load balancer.

Related Links

[Load balancer requirements](#) on page 57

In most production deployments, it is desirable to configure a load balancer between Latitude Studio and a cluster of the MDEX Engine nodes. This topic discusses the load balancer considerations associated with the Latitude cluster.

Examples of data sources

Latitude Studio supports two different scenarios for integration with the MDEX Engine cluster: read-only access, and read-only access with a specified host and port name for a leader node in the cluster (to enable updating operations). This topic contains examples of data sources for each scenario.



Note: The examples in this topic are specific to the cluster requirements. For complete information on how to configure data sources, see the *Latitude Studio Power User's Guide*.

Example of a read-only data source

A component may have a backing data source that allows read-only access to a cluster of MDEX Engine nodes.

If a data source allows read-only access to a cluster, all viewing actions are enabled for a component, along with editing of component's preferences. However, you cannot edit attribute groups for a read-only data source using the **Attribute Settings** component in the **Control Panel**. This is because the Attribute Settings component only lets you edit attribute groups for data sources that have update operations enabled.

Here is an example of a data source with read-only access to a cluster:

```
{  
  "server": "cluster_loadbalancer_server.company.com",  
  "port": "15000",  
  "name": "cluster read-only",  
  "apiVersion": "DISCOVERY_SERVICE"  
}
```

Note that this configuration looks identical to a standard non-clustered data source definition file. However, since the MDEX Engine is read-only, only read operations will be enabled in Latitude Studio.

Example of a read-only data source with updating access

A component may have a backing data source with read-only access to a cluster that also has an updating access (leader node) host name and port specified.

In this case, all viewing actions are enabled for a component, along with editing of component's preferences. In addition, you can change attribute settings for this data source the Attribute Settings component. Because the data source references the leader node that is responsible for handling updates, these changes are sent to the leader node. The index changes from the leader node are propagated to the other nodes in the cluster, but this may not happen immediately.

Here is an example of a read-only data source with the leader's node host name and port specified:

```
{  
  "server": "cluster_loadbalancer_server.company.com",  
  "port": "15000",  
  "name": "cluster updatable",  
  "apiVersion": "DISCOVERY_SERVICE",  
  "updateServer": "leaderNode.company.com",  
}
```

```

        "updatePort": "18000",
    }
}
```

Configuring a data source for cluster access

This procedure describes the format of the data source that allows both the read-only access to any node in the cluster through a load balancer and the updating access to the leader node.

Before configuring data sources so that they can connect to your cluster, ensure that you have already configured a load balancer between the Latitude Studio servers and the cluster. You will need to specify the load balancer's host name and port in the data sources.

In a non-clustered environment, after you install Latitude Studio and the MDEX Engine, the default data source references the port and host name of the server on which the MDEX Engine must be running once it has been installed. In particular, the `endeca-portal\data\endeca-data-sources` directory in Latitude Studio includes a `default.json` data source file, which has an implicit id of `default`. This file includes host and port information for the default installation of a single MDEX Engine server.

In a clustered environment, which type of data source you should configure for a Latitude Studio component depends on the type of information that will be sent from this component to the MDEX Engine:

- Latitude Studio components that make standard read-only queries (without sending any updating requests) can send requests to any node in the cluster.

Data sources for such components must reference only the host and port of the load balancer configured between the Latitude Studio servers and the cluster of MDEX Engine nodes. This is achieved by configuring in the data source the host name and port of the load balancer: `"server": "loadBalancerHost", "port": "loadBalancerPort"`.

- Those components that in addition to read-only queries must make updating requests for changing the configuration should direct their updating queries to the leader node.

Data sources for such components must reference the host and port of the leader node, in addition to referencing the load balancer server. This is achieved by configuring in the data source the host name and port of the leader node: `"updateServer": "leaderNodeHost", "updatePort": "leaderNodePort"`.

To configure a data source that allows both read-only and updating access to a cluster:

Specify in the data sources the host name and port of the load balancer, as well as the host name and port of the leader node, as shown in this example:

```

{
    "server": "loadBalancerHost",
    "port": "loadBalancerPort",
    "name": "cluster loadbalanced datasource",
    "apiVersion": "DISCOVERY_SERVICE",
    "updateServer": "leaderNodeHost",
    "updatePort": "leaderNodePort"
}
```

This configuration allows instances of Latitude Studio to distribute all of the normal queries across all MDEX Engine nodes in the cluster, while sending all update operations to the leader node only.



Note: When the leader node changes, you should update the data source definition file in Latitude Studio to specify a host name and port of the new leader node, but the load balancer configuration does not need to be changed.

Running a cluster

This section contains tasks you need to perform for cluster maintenance.

Removing a follower node

To remove a follower node, stop the MDEX Engine on this node with the `/admin?op=exit` command. This command gracefully shuts down a running MDEX Engine.

When you stop the MDEX Engine on one of the follower nodes, this node is removed from the cluster.

Related Links

[exit](#) on page 29

`/admin?op=exit` gracefully shuts down a running MDEX Engine.



Chapter 10

Using Endeca SSL Certificate Utilities

This section describes how to use the Endeca `enecerts` utility to generate standard and custom SSL certificate files to be used for SSL connections to the MDEX Engine. It also documents how to convert PEM-format certificates to the standard Java KeyStore (JKS) format.

Certificate files used by Endeca components

You configure SSL among the standard Endeca components by using a set of certificate files.

The certificate files are listed in the following table:

Certificate file	Description
<code>eneCert.pem</code>	Certificate file used by all Endeca clients and servers to specify their identity when using SSL. This certificate should be thought of as the identity of the Endeca system, or as the identity of all components of the Endeca system.
<code>eneCA.pem</code>	Certificate authority file used by all Endeca clients and servers to authenticate the other endpoint of a communication channel.
<code>eneCA.key</code>	Private key that is used by the <code>enecerts</code> certificate authority program to sign the <code>eneCert.pem</code> certificate.
<code>eneCA.cer</code>	Certificate authority file for import into browsers such as Microsoft Internet Explorer.
<code>eneCert.p12</code>	Personal Information Exchange (PKCS12-format) key file for import into browsers such as Microsoft Internet Explorer.

Because these certificate files are not provided in the Endeca Latitude packages, you must use the Endeca-provided `enecerts` utility (documented in the next topic) to generate them.

Generating SSL certificates

You can use the `enecerts` utility program to generate new SSL certificate files.

The two typical scenarios for generating SSL certificates are:

- You are setting up SSL for the first time and need to generate the set of standard certificates.

- You want to generate custom certificates, such as those with a private key size greater than the default 1024 bits.

The `enecerts` utility resides in the `bin` directory (located in the MDEX Engine root directory) under the name `enecerts.exe` on Windows.

Generating standard SSL certificates on UNIX

This procedure shows how to generate the set of standard certificates with a 1024-bit private key size on UNIX platforms.

To generate the SSL certificates on a UNIX machine:

1. Make sure that the `bin` directory (located in the MDEX Engine root directory) is in your `$PATH` environment variable.
2. Change to the directory in which the certificate files should reside.
3. Run the `enecerts` utility that creates the certificates:
4. Enter an export password of your choice.

If the programs finishes successfully, it displays the list of certificates that it generated.

Generating standard SSL certificates on Windows

This procedure shows how to generate the set of standard certificates with a 1024-bit private key size on Windows platforms.

To generate the SSL certificates on a Windows machine:

1. Open a command prompt.



Note: Make sure you are using a new command prompt window, not one that is left over from earlier tasks.

2. To ensure that the MDEX Engine environment variables are set for this user process, change to the MDEX Engine root directory, and then run the `mdex_setup.bat` script.
3. Change to the directory in which the certificate files should reside.
4. Run the `enecerts` utility that creates the certificates:

```
enecerts
```

5. Enter an export password of your choice.

If the programs finishes successfully, it displays the list of certificates that it generated.

Generating custom certificates

You can use the `enecerts` utility to generate customized certificates.

You can generate two types of customized certificates by:

- Specifying a private key size larger or smaller than the default 1024-bit size.
- Using your own CA file and private key to generate the `eneCert.pem` certificate.

The next two sections describe these operations.

Specifying a different certificate key size

The `--keysize` flag of the `enecerts` utility lets users specify the size of the generated private key. The flag syntax is:

```
--keysize bits
```

where *bits* is the private key size in bits (default value is 1024).

For example, the following Windows command creates certificates with a private key size of 2048 bits:

```
enecerts --keysize 2048
```

Keep in mind that using larger keys will slow system performance. A recommended alternative to the default 1024-bit size is a key size of 512 bits, which will give you a good balance between security and performance considerations.

Using your CA file to generate certificates

By default, the `enecerts` utility produces the `eneCert.pem` certificate (used by all clients and servers to specify their identity when using SSL) and the `eneCA.pem` CA certificate (used by all clients and servers that wish to authenticate the other endpoint of a communication channel).

If you have your own CA certificate and private-key files, you can use the `--CAkey` and `--CACert` flags to generate the `eneCert.pem` certificate. The private-key file (.key extension) is used to digitally sign the public key that is generated by the `enecerts` utility. Both flags must be used for this operation.

The syntax for the `--CAkey` flag is:

```
--CAkey private-key
```

where *private-key* is your own .key file with the private key for the CA that should be used to sign the generated certificate.

The syntax for the `--CACert` flag is:

```
--CACert cert-pem
```

where *cert-pem* is your CA certificate (.pem extension). This file is the same type of file as the default `eneCA.pem` CA certificate.

For example, the following Windows command creates a signed certificate file using your own CA certificate and private-key files:

```
enecerts --CAkey myCA.key --CACert myCA.pem
```

You would then use the resulting `eneCert.pem` certificate and your CA file (`myCA.pem` in the example) to configure SSL for your Endeca components. If you have multiple machines in your deployment, you must also copy these files to the other machines.

Copying the SSL certificates to other machines

All machines that are running your deployment must use the same SSL certificates.

If you have multiple machines in your deployment, the standard or custom SSL certificates should be created only once, on one machine. You must then copy them to the directories (on all other machines) from which the MDEX Engine is started. All of the machines must use the same SSL certificates.

Importing SSL certificates in Internet Explorer

Depending on the details of your deployment, you may have to import the SSL certificates to your browser.

If the MDEX Engine is SSL-enabled, you must import the certificates in Internet Explorer if you want to connect directly to it via your browser (for example, to issue administrative operation commands).

To import the SSL certificates in Internet Explorer:

1. If you created the SSL certificates on a UNIX machine, copy the `eneCert.p12` and `eneCA.cer` files to the Windows machine.
2. Open Internet Explorer. From the **Tools** menu, choose **Internet Options**.
3. In the **Internet Options** dialog box, click the **Content** tab. To display the **Certificates** dialog box, click the **Certificates** button.
4. If imported certificates are listed from any previous Endeca installations, delete them.
5. In the **Certificates** dialog box, click **Import** to launch the **Certificate Import** wizard. This allows you to import the standard or custom SSL certificates that you created using the `enecerts` utility. Follow these steps:
 - a) In the **Welcome** screen, click **Next**.
 - b) In the **File to Import** screen, browse to `eneCA.cer`, which is located in the `%ENDECA_CONF%\etc` directory or the directory to which you copied the file. You may have to change the **File of Type** option to `X.509 Certificate` to see the `eneCA.cer` file.
 - c) In the **Certificate Store** screen, choose `Automatically select the certificate store based on the type of certificate`.
 - d) In the **Completing the Certificate Import Wizard** screen, click **Finish**. (If you receive a Security Warning, click **Yes**.) When you see the confirmation message, click **OK**.
 - e) Relaunch the **Certificate Import** wizard.
 - f) In the **File to Import** screen, browse to the `eneCert.p12` file, which is located in the same directory as the `eneCA.cer` file.
 - g) In the **Password** screen, enter the password you used when you created the SSL certificates.
 - h) In the **Certificate Store** screen, choose `Automatically select the certificate store based on the type of certificate`.
 - i) In the **Completing the Certificate Import Wizard** screen, click **Finish**. When you see the confirmation message, click **OK**.
6. Close the **Certificates** window.
7. In the **Internet Options** window, click **OK**.

Configuring the MDEX Engine for SSL mutual authentication

This topic describes high level steps required to configure an SSL mutual authentication between the MDEX Engine server and an external server. The authentication uses certificates signed by a certificate authority (CA). This setup may apply if your MDEX Engine and external servers are hosted outside the firewall, or if a two-way authentication is required between them.

When using Web services and XQuery with the MDEX Engine, client servers running non-Endeca software may need to access the MDEX Engine server securely. In such cases, a secure connection may need to be established between these servers by configuring the MDEX Engine server for authentication with SSL certificates.

This procedure is an example of how you can establish a mutual (two-way) authentication. Treat this procedure as a high-level recommendation rather than the only way to establish a secure connection. Other steps may be required depending on your specific security requirements.

In this procedure, you create two signed certificates. First, you create a private key and send a Certificate Signing Request (CSR) to a CA from the external server. Next, you create a private key and send a CSR from the server hosting the MDEX Engine. You can then start the MDEX Engine referencing the `sslcertfile` which contains the MDEX Engine private key and the signed certificate.

To configure an SSL mutual authentication between the MDEX Engine and an external server:

1. Create a private key and send a Certificate Signing Request (CSR) from the external server. You can create a private key and issue a CSR by using one of these methods:
 - Use the server's certificate management utility (if applicable)
 - Use `openssl` commands
 - Consult your security and server administrator for assistance.



Note: Some CA vendors require that the CSR be generated from 2046-bit length private keys and not from 1024-bit length keys. Please confirm with your CA vendor before issuing the CSR.

2. Send the CSR to a CA for signing.
A CA provides a bundled key file (including intermediate keys) along with a signed certificate.



Note: You will need the bundled key file for the Dgraph `--sslcafile` startup flag later on in this procedure.

3. Add the signed certificate to the keystore of the external server.

For information, refer to the server's documentation or your security administrator.

4. Create a private key and certificate for the MDEX Engine server using the `openssl` utility.
For example, the following command creates a 2048-bit RSA key that is valid for a year:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout MDEXCert.pem
-out MDEXCert.pem
```

The resulting `MDEXCert.pem` file stores both the private key and the certificate.

5. Create a Certificate Signing Request (CSR) from the `MDEXCert.pem` file, as follows:

```
openssl req -new -key MDEXCert.pem -out MDEXCertCSR.pem
```

6. Send the `MDEXCertCSR.pem` file to the CA for signing.

7. Obtain from the CA a bundled key file (including intermediate keys) along with a signed certificate.

8. Create an empty file, such as `MDEXSSLCert.pem` to store the combination of the MDEX Engine private key and the signed certificate.

You can do this by copying the entry for the private key (step 4) into the empty file, and appending the contents of the signed certificate (Step 7) underneath the private key entry in the new file.

9. Reference the file `MDEXSSLCert.pem` created in the previous step in the `--sslcertfile` startup flag for the Dgraph.

10. Add both the `sslcafile` and `sslcertfile` flags to your Dgraph startup options, as follows:

```
--sslcafile <full_path_to_location_of_bundled_key_file_in_step2>
```

```
--sslcertfile <full_path_to_location_of_MDEXSSLCert.pem>
```

11. (Optional) If the external server requires the bundled keys for the MDEX Engine that you obtained in step 7, add them accordingly to its keystore.

Converting PEM-format keys to JKS format

This topic describes how to convert PEM-format certificates to the standard Java KeyStore (JKS) format.

The Java KeyStores can be used for communication between Endeca components that are configured for SSL (for example, between Latitude Studio and the MDEX Engine, if both are SSL-enabled).

Two utilities are referenced in the instructions below:

- `openssl`, which is located in the `bin` directory of the MDEX Engine distribution.
- `keytool`, which is located in the `bin` directory of the JDK distribution.

This procedure assumes the following:

- You have run the appropriate version of the `mdex_setup` script for your operating system.

This script adds the `utilities` directory and the MDEX Engine binaries to the search path, and allows you to run the `openssl` utility from the directory of your choice.

It is documented as part of the MDEX Engine installation in the *Latitude Installation Guide*.

- Your path will allow you to use the `keytool` utility from the directory of your choice.
- You have already generated the set of standard SSL certificates with the `enecerts` command, as documented earlier in this section.
- All of the input files are located in the local directory.

To convert the PEM-format keys to Java KeyStores:

1. Convert the certificate from PEM to PKCS12, using the following command:

```
openssl pkcs12 -export -out eneCert.pkcs12 -in eneCert.pem
```

You may ignore the warning message this command issues.

2. Enter and repeat the export password (endeca).

3. Create and then delete an empty truststore for Tomcat, using the following commands:

```
keytool -genkey -keyalg RSA -alias "endeca" -keystore truststore.ks
keytool -delete -alias endeca -keystore truststore.ks
```

The `-genkey` command creates the default certificate shown below. (This is a temporary certificate that is subsequently deleted by the `-delete` command, so it does not matter what information you enter here.)

```
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]:
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
```

```
What is the two-letter country code for this unit?
[Unknown]:
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
correct?
[no]: yes

Enter key password for <endeca>
(RETURN if same as keystore password):
Re-enter new password:
```

4. Import the CA into the truststore, using the following command:

```
keytool -import -v -trustcacerts -alias endeca-ca -file eneCA.pem -keystore
truststore.ks
```

5. Enter the keystore password (endeca).
6. At the prompt, "Trust this certificate?" type yes.
7. Create an empty Java KeyStore, using the following commands:

```
keytool -genkey -keyalg RSA -alias "endeca" -keystore keystore.ks
keytool -delete -alias endeca -keystore keystore.ks
```

The -genkey command creates the default certificate shown below. (This is a temporary certificate that is subsequently deleted by the -delete command, so it does not matter what information you enter here.)

```
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]:
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]:
Is CN="Unknown", OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
correct?
[no]: yes
```

8. Import your private key into the empty JKS, using the following command:

```
keytool -v -importkeystore -srckeystore eneCert.pkcs12 -srcstoretype
PKCS12 -destkeystore keystore.ks -deststoretype JKS
```




Chapter 11

Latitude Studio Administrative Tasks

This section describes some of the administrative tasks performed in Latitude Studio.

About Latitude Studio administrative tasks

The Latitude Studio administrator generally controls the installation and setup of Latitude Studio and manages its users.

For full documentation on administering the underlying Liferay Portal, see the *Liferay Portal Administrator's Guide* version 5.2.

To access a free PDF download of this guide:

1. Go to <http://www.liferay.com>.
2. Click the **Documentation** link.
3. Select the version.
4. Click the **Administration** link.

About the Latitude Studio Control Panel

You use the Latitude Studio **Control Panel** to perform administrative functions in Latitude Studio.

The **Control Panel** contains configuration options such as layout controls, attribute group settings, portal settings, and server settings.

It also provides access to a wide range of administrative controls, including managing accounts, adding new users, and monitoring performance.

Overview of the Control Panel sections

The **Control Panel** consists of five sections, each of which contains a number of tools.

User:	The logged-in user's personal space. It allows users to manage their accounts and pages.
--------------	---

Latitude:	Provides access to Latitude Studio administrative components, including: <ul style="list-style-type: none"> • Data Sources • Data Source Bindings • Attribute Settings • Framework Settings • Performance Metrics
Portal:	Intended for portal administrators to manage the user community.
Server:	Provides access to server administration tools such as resource usage, logging, and server shutdown It also allows administrators to manage instances of Latitude Studio and to install plugins (including custom components and themes).
Layout Control:	Allows the administrator to manage the integration of Web content into Latitude Studio applications.

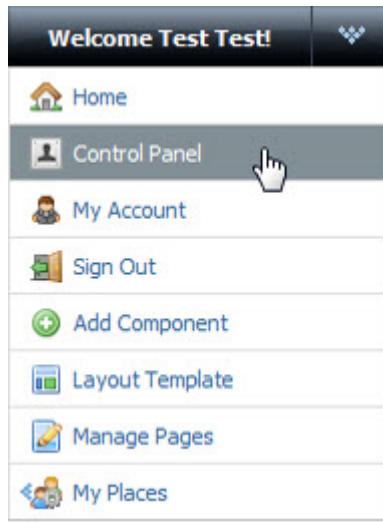
Accessing the Control Panel

The Latitude Studio **Control Panel** is available from the Dock menu.

After logging in to Latitude Studio, to display the **Control Panel**:

1. Point the cursor at the Dock in the upper-right corner of the page.

The Dock is labeled "Welcome <user name>"



2. From the drop-down menu, choose **Control Panel**.

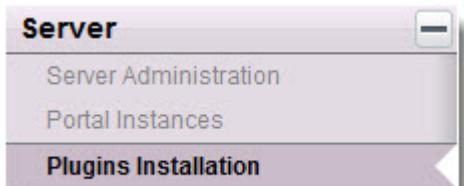
Installing a new theme

Themes define the look and feel of a Latitude Studio application. A Web developer can create a new theme for your application.

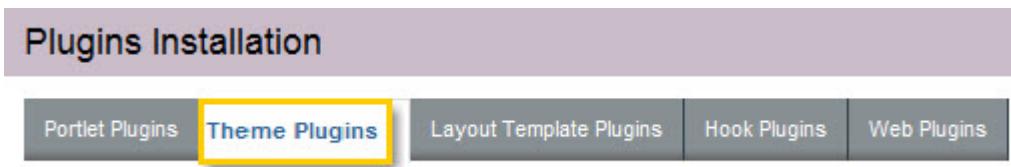
For more information about developing themes, see <http://www.liferay.com/web/guest/community/wiki/-/wiki/Main/Themes>.

To install a new theme:

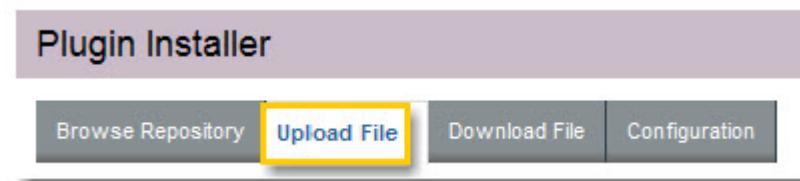
1. In the Dock, click **Control Panel**.
2. In the **Server** section of the **Control Panel**, click **Plugins Installation**.



3. In the **Plugins Installation** panel, click the **Theme Plugins** tab.



4. Click the **Install More Themes** button.
5. In the **Plugin Installer** panel, click **Upload File**.



6. Browse to select the theme's .war file, and then click **Open**.
7. Click **Install**.

Setting up the email server for Bookmarks support

The Latitude Studio contains a **Bookmarks** component. Before end users email bookmarks to other users, the Latitude Studio administrator must configure the mail server in the **Control Panel**.

To set up the mail server:

1. In the Dock, click **Control Panel**.
2. In the **Server** section of the **Control Panel Portal** menu, click **Server Administration**.
3. In the **Server Administration** panel, click the **Mail** tab.
4. On the **Mail** panel, fill out the following outgoing email settings:
 - **Outgoing SMTP Server**
 - **Outgoing Port**
 - **User Name**
 - **Password**

For example:

Outgoing SMTP Server	acme.com.s7a1.pstmp.com
Outgoing Port	25
Use a Secure Network Connection	<input type="checkbox"/>
User Name	user_user@acme.com
Password	12345

5. Click **Save**.



Appendix A

Endeca Flag Reference

This appendix provides a description of the flags (options) used by the Dgraph program.

Dgraph flags

The Dgraph program starts the MDEX Engine.

You start the MDEX Engine by running a program called Dgraph, which you point at a set of indices loaded into the MDEX Engine by the Data Ingest Web Service. The Dgraph has a number of options that allow you to adjust the MDEX Engine.

The usage of Dgraph is as follows:

```
dgraph [-?Adv] [--flags] <db_prefix>
```

where `<db_prefix>` specifies the path to the directory, and the prefix used for the files in your Endeca application.

Flag	Description
<code>?</code>	Print the help message and exit.
<code>-v</code>	Verbose mode. Print information about each request to <code>stdout</code> .
<code>--ancestor_counts</code>	Compute counts for root managed attribute values and any intermediate managed attribute value selections. By default, the Dgraph only computes refinement counts for proper refinements (in other words, for actual managed attribute values). It does not compute counts for root managed attribute values or for any intermediate managed attribute value selections.
<code>--backlog-timeout <seconds></code>	Specify the wait limit (in seconds) for a query that has been read and queued for processing. This is the maximum number of seconds that a query is allowed to spend waiting in the processing queue before the Dgraph responds with a timeout message. The default value is 0 seconds.
<code>--bulk_load_port <num></code>	Specify the port for bulk load ingest operations. This port number must be different from the port specified by the <code>--port</code> flag. If this flag is not used when starting the MDEX Engine, then the default bulk load port is either 5556 (if the <code>--port</code> flag is not used) or is the number specified by the <code>--port</code> flag plus one.

Flag	Description
--cmem <MB>	Specify an absolute value in MB for the MDEX Engine cache. When an absolute value is not specified with the --cmem flag, the default Dgraph cache size is computed as 10% of the amount of RAM available in the system.
--coordinator_host <host name>	Specifies the host name of the server on which the Cluster Coordinator service is running. You specify this flag along with the --coordinator_port flag when you start the MDEX Engine as one of the nodes in the cluster.
--coordinator_port <num>	Specifies the port of the server on which the Cluster Coordinator service is running. The Cluster Coordinator expects that you specify the port 2181 (if you specify another port, changes to the Cluster Coordinator configuration file are required). You specify this flag along with the --coordinator_host flag when you start the MDEX Engine as one of the nodes in the cluster.
--disable_fast_aspell	Disable fast mode for the aspell spelling module. If you disable fast mode, it decreases the performance of the spelling correction, but may allow additional queries to be corrected. When the fast mode is enabled, it can significantly speed up applications that use spelling correction features with the aspell module. The fast mode is used by default.
--esampmin <num>	Specify the minimum number of records to sample during refinement computation. The default is 0. Tuning recommendations: <ul style="list-style-type: none">For most applications, larger values reduce performance without improving dynamic refinement ranking quality.For some applications with extremely large, non-hierarchical managed attributes (if they cannot be avoided), larger values can meaningfully improve dynamic refinement ranking quality with minor performance cost.
--help	Print the help message and exit.
--implicit_exact	Disable approximate computation of implicit refinements. Use of this option is not recommended. If this option is not enabled, managed attribute values without full coverage of the current result record set may sometimes be returned as implicit refinements, although. The probability of such "false" implicit refinements is minuscule.
--implicit_sample	Set the maximum number of records to sample when computing implicit refinements (which are a performance tuning parameter). Default value is 1024.
--latin1	Ignore character accents when handling search requests, and use ISO Latin 1 character mappings when processing search requests.
--log <path>	Specify the path for the Dgraph request log file. The default log file is named <code>dgraph.reqlog</code> .

Flag	Description
--net-timeout	Specify the maximum number of seconds the Dgraph waits for the client to download data from queries across the network. The default network timeout value is 30 seconds.
--out <stdout/stderr file>	Specify file path to which stdout/stderr should be remapped (the default is to use default stdout/stderr for the process).
--pidfile <pidfile-path>	Specify the file to which to write the process ID (pid). If unspecified, the default name of the pid file depends on how the Dgraph starts. Running the Dgraph from the command line creates a default named <code>dgraph.pid</code> .
--port <num>	Specify the port to use in server (non-interactive) mode. The default is 5555.
--replica-name <name>	<p>Specify the name of the MDEX Engine node that should serve as one of the follower nodes in the cluster. This name must be unique across the cluster. The name must also be a valid directory name (characters such as slashes (/) are not allowed). You can start more than one node in the cluster with this command, thus designating more than one follower node.</p> <p>Before starting the MDEX Engine with this command flag, ensure that the Cluster Coordinator service is running on the server that serves as the leader node.</p> <p>All nodes must be able to connect to the Cluster Coordinator. Therefore, when you specify a follower node with the <code>--replica-name</code> flag, also specify for the follower node the host name and port of the Cluster Coordinator service using the <code>--coordinator_host</code> and <code>--coordinator_port</code> commands.</p> <p> Note: If you start a node without the <code>--replica-name</code> flag, the Cluster Coordinator assumes this is the leader node. Since there could be one and only one leader node in the cluster, the MDEX Engine will not start if it is asked to be the leader node but a leader node already exists.</p>
--search_max <num>	Specify the maximum number of terms for text search. Default is 10.
--snip_cutoff <num>	<p>Limit the number of words in an attribute that the MDEX Engine evaluates to identify the snippet. If a match is not found within <code><num></code> words, the MDEX Engine does not return a snippet, even if a match occurs later in the attribute value.</p> <p>If the flag is not specified, or <code><num></code> is not specified, the default is 500.</p>
--snip_disable	Globally disable snippetting.
--sslcafile <CA-certfile-path>	Specify the path of the <code>eneCA.pem</code> Certificate Authority file that the Dgraph will use to authenticate SSL communications with other

Flag	Description
	Endeca components. If not given, SSL mutual authentication is not performed.
--sslcertfile <certfile-path>	Specify the path of the <code>eneCert.pem</code> certificate file that will be used by the Dgraph to present to any client for SSL communications. If not given, SSL is not enabled for Dgraph communications.
--sslcipher <cipher-list>	Set one or more cipher names (such as RC4-SHA) that specify the minimum cryptographic algorithm that the Dgraph will use during the SSL negotiation. If multiple ciphers are specified, the names must be separated by colons.
--stat-abins	<p>Enable refinement counts for aggregated records. A refinement count is the number of records that would be in the result set if you were to refine on a managed attribute value. An aggregated record is a record that represents several records that are rolled up into a single record for display purposes.</p> <p>If you use this flag, the refinement counts reflect how many aggregated records the MDEX Engine would return in a result set if you were to refine on a managed attribute value.</p> <p>In general, the MDEX Engine calculates refinement counts as follows:</p> <ul style="list-style-type: none"> • When returning regular (non-aggregated) record results, the MDEX Engine calculates refinement counts per refinement. • When returning aggregated record results, the <code>--stat-abins</code> flag lets the MDEX Engine return the refinement counts for aggregated records. These counts accurately reflect the number of aggregated records per refinement. (You enable refinement counts for aggregated records by using this flag.) <p>Note that dynamic statistics on aggregated records is an expensive computation for the MDEX Engine. Use this flag only if you intend to display the refinement counts for aggregated records in your front-end application.</p>
--stat-all	Enable all available dynamic attribute value characteristics. Note that this option has performance implications and is not intended for production use.
--stat-brel	Create dynamic record attributes indicating the relevance rank assigned to fulltext search result records.
--syslog	Direct all output to syslog.
--thesaurus_cutoff <limit>	<p>Set a limit on the number of words in a user's search query that are subject to thesaurus replacement.</p> <p>The default value of <code><limit></code> is 3. This means that up to 3 words in a user's search query can be replaced with thesaurus entries. If there exist more terms in the query that match thesaurus entries than the number set by this flag, none of the terms are thesaurus expanded.</p>

Flag	Description
	This option is intended as a performance guard against very expensive thesaurus queries. Lower values improve thesaurus engine performance.
--thesaurus_multiword_nostem	<p>Specify that words in a multiple-word thesaurus form should be treated like phrases and should not be stemmed, which increases performance for some query loads. Single-word terms will be subject to stemming regardless of whether this flag is specified.</p> <p>This flag prevents the Dgraph from expanding multi-word thesaurus forms by stemming. Thesaurus entries continue to match any stemmed form in the query, but multi-word expansions only include explicitly listed forms. To get the multi-word stemmed thesaurus expansions, the various forms must be listed explicitly in the thesaurus.</p>
--threads <num>	Specify the number of threads in the MDEX Engine threading pool. The value of <num> must be a positive integer (that is, 1 or greater). The default for <i>num</i> is 2. The recommended number of threads for the MDEX Engine is typically equal to the number of cores on the MDEX Engine server.
--unctrct	<p>Specify to the Dgraph not to compute implicit managed attributes, and to only compute and present explicitly specified managed attributes, when displaying refinements in navigation results. Specifying this flag does not reduce the size of the resulting record set that is being displayed; however, it improves run-time performance of the MDEX Engine.</p> <p>Be aware that if you use this flag, in order to receive meaningful navigation refinements, you need to make top-level precedence rules work for ALL outbound queries.</p>
--validate_data	Validate that all indexed data loads and then exit.
--version	Print version information and exit. This includes both the Latitude version and the internal MDEX Engine identifier and index format version.
--wildcard_max <count>	Specify the maximum number of terms that can match a wildcard term in a wildcard query that contains punctuation, such as ab*c.def*. The default is 100.
--whymatch	Enable computation of "Why Did It Match" dynamic record attributes returned as results of full-text search queries. These dynamic attributes contain a copy of the attribute key and value that caused the match, along with query interpretation notes (spelling, thesaurus, and so on).
--whymatchConcise	Similar to --whymatch, but produces more concise dynamic attribute values containing only the attribute key and query interpretation notes. This is useful when the attribute value might include large amounts of text, such as document contents.

Flag	Description
--wordinterp	Enable computation of word interpretation dynamic supplement (or see-also) objects, which report on alternate forms of user query terms considered by the text search engine while processing full-text (record) search requests.
--xquery_fn doc <mode>	Specifies the handling of the <code>fn:doc()</code> function within XQuery. The following three values are supported: <ul style="list-style-type: none"> • <code>none</code> causes all calls to <code>fn:doc()</code> to fail. • <code>sandbox</code> allows <code>fn:doc()</code>, but interprets its argument as a relative path within the XML subdirectory of the XQuery service directory. • <code>open</code> allows <code>fn:doc()</code> and interprets its argument as a URL. If not specified, defaults to <code>none</code> . Note that <code>open</code> is not supported for use in deployed applications.
--xquery_path <path>	Specify the directory in which XQuery Web service resources are located. XQuery main modules and WSDL files are loaded from this directory. Library modules are loaded from the <code>lib</code> subdirectory. If not specified, a user XQuery path is not used.

Index

A

accessing the Control Panel 80
admin operations
 about 27
 exit 29
 flush 31
 help 31
 list of 27
 logroll 31
 merge 31
 ping 31
 reload-services 31
 stats 32
 statsreset 32
 updateaspell 32
Administration Web Service
 about 11
 accessing 12
 using 12
administrative tasks
 admin and config operations 27
 Latitude Studio 79
 overview 10
aggressive merge policy 37
architecture
 of a cluster of MDEX Engine nodes 54

B

balanced merge policy 37

C

Certificate Authority file
 eneCA.pem 71
certificates
 copying to other machines 73
 eneCA.cer 71
 eneCA.key 71
 eneCA.pem 71
 eneCert.p12 71
 eneCert.pem 71
 generating from own private key 73
 importing in Internet Explorer 74
changing the merge policy of the MDEX Engine 38, 40
cluster 56
 about 53
 and Latitude Data Integrator configuration 67
 and Latitude Studio configuration 69
 architecture 54
 behavior 63
 examples of data sources 68

cluster (*continued*)

 file system requirements 57
 installation requirements 59
 leader node 56
 load balancer configuring 67
 load balancer requirements 57
 operating systems requirements 57
 planning nodes 62
 sending data updates 67

Cluster Coordinator

 configuration file 61

cluster Coordinator package

 installing 60

config operations

 about 27, 34
 about 27, 34
 for logging verbosity 33
 help 36
 log-disable 35
 log-enable 35
 log-status 35
 logging variables 34

configuration file

 cluster 61

connecting a Web browser to your MDEX Engine 21

connecting Latitude Studio with cluster nodes 69

connection errors

 MDEX Engine and client 25

Control Panel

 about 79
 accessing 80
 parts of 79

core dump files

 in the Dgraph 22
 managing 22

D

Dgraph

 checking aliveness of 21
 flags 83
 what to collect for debugging 23

E

edit controls, turning on 80

eneCA.cer

 description 71
 importing in Internet Explorer 74

eneCA.key, description of 71

eneCA.pem

 description 71

eneCert.p12
 description 71
 importing in Internet Explorer 74
eneCert.pem
 description 71
 generating with own private key 73
enecerts utility
 changing key size 73
 generating certificates with own private key 73
 overview 72
exit admin operation 29

F

flags, Dgraph 83
flush admin operation 31
follower node 56
 adding, to a cluster 65
forcing a merge 40

G

Global Configuration Record
 retrieving with API 38
 setting merge policy 39

H

help admin operation 31
help config operation 36
high availability 53

I

incremental updates, merge policy for 37
inittab, starting MDEX Engine from 51
Internet Explorer, importing certificates in 74
IPv4 and IPv6 address support in MDEX Engine 21

J

Java keystore
 converting to 76
Java KeyStores
 converting PEM-format keys to 76
job monitoring
 job start time 16
 listing jobs 16
 types 15
 when to use 15

K

key size, changing private 73

L

Latitude Studio
 cluster integration 68
 Control Panel 79
 installing a new theme 81
 Liferay documentation 79
 setting up the mail server for 81
leader node 56
 adding, to a cluster 64
 list of updating operations to send to it 66
load balancer
 in a cluster, configuring 67
log-disable config operation 35
log-enable config operation 35
log-status config operation 35
logging variables
 MDEX Engine 33
 operation syntax 33, 35
 supported variables for 34
logroll admin operation 31

M

mail server for Latitude Studio, setting up 81
MDEX Engine
 admin operations 27
 config operations 34
 configuring automatic restart 50
 connecting Web browsers to 21
 crash dump files on Linux 22
 crash dump files on Windows 22
 creating as a Windows service 46
 deleting Windows service 48
 flags 83
 identifying connection errors 25
 IPv4 and IPv6 address support 21
 logging as a Windows service 50
 logging variables for 33
 logs 23
 modifying Windows service 47
 running multiple instances on a single machine 24
 setting description for Windows service 46
 started from inittab on Linux 51
 starting or stopping from Services utility 49
merge admin operation 31
merge policy
 changing in a running MDEX Engine 40
 for incremental updates 37
 forcing a merge 40
 getting programmatically 38
 setting 38
 setting programmatically 39
 types of 37
multiple-node cluster
 development environment 55
 mutual authentication for MDEX Engine 74

O

operation syntax for MDEX Engine logging variables 33

P

ping admin operation 31
 pinging components 21
 private key for certificates
 changing size of 73
 description 71

R

reload-services admin operation 31

S

security
 mutual authentication for MDEX Engine 74
 single-node cluster
 development environment 54
 snapshot
 about 17
 cpmdex command 19
 creating 18
 deleting 18
 restoring an MDEX Engine 19
 restrictions 18
 spelling, enabling 32
 SSL certificates
 converting PEM-format keys to JKS format 76

SSL certificates (continued)

 mutual authentication 74

stats admin operation 32

statsreset admin operation 32

T

themes, installing Latitude Studio 81
 troubleshooting
 baseline updates 25
 Dgraph port and socket 24
 truststore conversion from eneCA.pem 76
 turning on edit controls 80

U

updateaspell admin operation 32
 URL operations, about 27

V

variables for MDEX Engine logging 34

W

Who should use this guide 7
 Windows service
 creating MDEX Engine as 46, 50
 deleting MDEX Engine 48
 modifying MDEX Engine configuration 47
 setting description 46

