Endeca® Latitude

Migration Guide

Version 2.2.2 • December 2011

Contents

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 © 2011 Oracle. All rights reserved.

Rosette® Linguistics Platform Copyright © 2000-2011 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, US Patent 8005643, US Patent 8019752, US Patent 8024327, US Patent 8051073, US Patent 8051084, 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.

About this guide

This guide helps you upgrade your Latitude implementation by describing the major changes between versions 2.1.0 and 2.2.x of Latitude.

Who should use this guide

This guide is intended for system administrators and developers who are upgrading Endeca Latitude on Windows or Linux.

For Latitude Studio, this guide is intended for users whose Latitude Studio applications are built using standard Latitude components. It does not necessarily apply to Latitude Studio applications built with custom components.

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

Upgrading to Latitude Version 2.2.x

This section provides instructions for upgrading to Latitude 2.2.x. Before you start the upgrade processes, check the remaining sections of this guide to learn about the changes that will affect you during or after an upgrade.

Required reading

In addition to reading this document, Endeca recommends that you read the following documents for important information about the release.

Release Announcement for Latitude

The Release Announcement provides a brief explanation of the new features that were added in Version 2.2.x.

You can download the Release Announcement from the Knowledge Base section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*.

Release Notes

The Release Notes for each package provide information about new features, changed features, and bug fixes for this release.

You can download the release notes from the Knowledge Base section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*.

Installation Guide

The *Latitude Installation Guide* gives an overview of the Latitude components and includes information about configuration scenarios. After installing all the components in your Endeca deployment, read this guide for information on verifying your installation.

You can download the *Latitude Installation Guide* from the Downloads section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*.

High-level upgrade procedure

This topic provides the recommended order for upgrading to version 2.2.x.

The upgrade process assumes you need to install Latitude components on dedicated servers. Because of this, the upgrade process references installation instructions for each component, and does not utilize the Latitude All-in-One Installer. The All-in-One installer is intended for single-server development environments only, and should be used with the Quick Start project.

The upgrade process includes separate upgrade procedures for the MDEX Engine, Latitude Data Integrator and Latitude Studio. In addition, as part of the upgrade, you need to make changes to the existing pre-upgrade graphs so that you can run these graphs in LDI after the upgrade, and load data and configuration to the MDEX Engine index.

The following procedure provides high-level steps for the entire upgrade process. See the individual topics in this section for detailed instructions in each step.

To upgrade to Latitude release version 2.2.x:

1. Upgrade the MDEX Engine.
   As a result of this step, you should have the MDEX Engine installed, started and provisioned. Note that after this step, the MDEX Engine index is empty.
2. Upgrade the Latitude Data Integrator Designer.
   As a result of this step, you should have the LDI Designer installed and pointing to the workspace you used for Latitude graphs before the upgrade.

   Note: Do not run any existing graphs in LDI before you make changes to them.

3. In LDI, make changes to the graphs you have created in the previous release.
4. Run the upgraded graphs.
5. Upgrade Latitude Studio.

Related Links

Upgrading the MDEX Engine on page 10
To upgrade the MDEX Engine, you uninstall the existing version of the MDEX Engine and install the new version. Next, you provision the MDEX Engine using the `mkmdex` command.

Upgrading the LDI Designer on page 11
To upgrade the LDI Designer, you uninstall the existing version of LDI and install the new one. You can still use your existing workspace after you install the LDI Designer version 2.2.x.

Upgrading Latitude Studio on page 13
This section describes how to upgrade to Latitude Studio 2.2.x.

Upgrading the MDEX Engine

To upgrade the MDEX Engine, you uninstall the existing version of the MDEX Engine and install the new version. Next, you provision the MDEX Engine using the `mkmdex` command.

To upgrade the MDEX Engine:

1. Uninstall the previous version of the MDEX Engine, using instructions from the *Latitude Installation Guide* version 2.1.
2. Install the MDEX Engine, using instructions from the *Latitude Installation Guide* version 2.2.x. If required, uninstall and install the Cluster Coordinator.
3. Start and provision the MDEX Engine by running the `dgraph` and `mkmdex` commands. For information, see the *Latitude Installation Guide* version 2.2.x.

After you are finished, you should have an MDEX Engine running with an empty MDEX Engine index.

Proceed to upgrading the Latitude Data Integrator (LDI) Designer.

Upgrading the LDI Designer

To upgrade the LDI Designer, you uninstall the existing version of LDI and install the new one. You can still use your existing workspace after you install the LDI Designer version 2.2.x.

If you have custom plug-ins deployed to LDI 2.1, you should back up the plug-ins before beginning the LDI upgrade. You can reload them after the LDI upgrade.

To upgrade the LDI Designer:

1. Uninstall the LDI Designer version 2.1. For information, see the *Latitude Installation Guide*, version 2.1.
   After you uninstall, the workspace you used for creating your Latitude graphs should remain in place on the server.
2. Install the LDI Designer version 2.2.x. For information, see the *Latitude Installation Guide*, version 2.2.x.
3. Start the LDI Designer and point it to use the workspace you used for your Latitude graphs.

Now you have upgraded to the new version of the LDI Designer. However, do not run your existing graphs until you have made changes to them that are required for the Latitude release version 2.2.x.

After the upgrade, you can proceed to making changes to your existing Latitude graphs.

If you are reloading a custom plug-in, keep in mind that the installation path of LDI has changed. For example, on Windows the 2.1.x default installation path has changed from

```
C:\Program Files (x86)\CloverETL Designer
```

to this 2.2.2 default path

```
C:\Endeca\Latitude\2.2.2\DataIntegrator
```

Because LDI would still be referencing the `customcomponents.xml` at the previous location, it will have to be updated to the new location in the 2.2 version of the Designer.

Related Links

List of required changes for existing LDI graphs on page 11
This topic lists changes you should make in existing LDI graphs before attempting to run them after you upgrade the LDI Designer and the MDEX Engine to version 2.2.x.

Running the upgraded graphs on page 13
After you have made changes to the existing graphs, you can run them in LDI.

List of required changes for existing LDI graphs

This topic lists changes you should make in existing LDI graphs before attempting to run them after you upgrade the LDI Designer and the MDEX Engine to version 2.2.x.

If you have existing graphs created in LDI that worked with the MDEX Engine in the Latitude 2.1 release, the following changes are required in these graphs so that they work after you upgrade to this release:

Making changes to the namespace of the Configuration Web Service

Change the namespace in the **Request Structure** of those components in LDI that use calls to the Configuration Web Service.

The namespace of the Configuration Web Service has changed to the following:

```
<config-service:configTransaction
xmlns:config="http://www.endeca.com/MDEX/config/services/types"
xmlns:mdex="http://www.endeca.com/MDEX/XQuery/2009/09">
```

This means that if you used the LDI **WebServiceClient** for sending requests through the Configuration Web Service, you need to change the way the **Request Structure** is specified. Use the following example:

```
<config-service:configTransaction
xmlns:config-service="http://www.endeca.com/MDEX/config/services/types"
outerTransactionId="${MDEX_TRANSACTION_ID}">
...
</config-service:configTransaction>
```

In this example, the line:

```
xmlns:config-service="http://www.endeca.com/MDEX/config/services/types"
```

correctly identifies the namespace.

Making Latitude components transaction-friendly

Change your existing Latitude components so that they are aware of transactions and can run either outside of transactions, or within them.

All Latitude-specific components automatically reference the outer transaction ID if it is specified in the `workspace.prm` file for your project.

Therefore, to make it possible to run your Latitude-specific components within transactions (as well as without transactions), add this ID to an existing project in its `workspace.prm` file, as in the following example (notice the empty value):

```
MDEX_TRANSACTION_ID=
```

With this change, you can continue to run your components outside of transactions (as in previous releases), and also run them within transactions, without making any additional changes to them, or to `workspace.prm`.

Note: Setting the `MDEX_TRANSACTION_ID` parameter is optional. It allows you to have flexibility in your project if you ever need to run transactions. If you don't set it, you will still be able to run all graphs that don't use transactions.

Making non-Latitude components transaction-friendly

To use any non-Latitude components, such as WebServiceClient or HTTPConnector) inside a graph that starts and commits a transaction, do the following:

1. Specify an outer transaction ID parameter in the `workspace.prm` file for your project, with an empty value:

   ```
   MDEX_TRANSACTION_ID=
   ```

2. Include a reference to this parameter in the **Request Structure** for your component, as in this example:

   ```
   <config-service:configTransaction
   xmlns:config-service="http://www.endeca.com/MDEX/config/services/types"

   outerTransactionId="${MDEX_TRANSACTION_ID}">
   ```

```
...
</config-service:configTransaction>
```

These steps make it possible for the components to run both with and without transactions, without requiring any additional modifications to the components configuration.

For information on Latitude-specific parameters in the `workspace.prm` file, see the *Before You Begin* section in the *LDI MDEX Engine Components Guide*.

Changing the way precedence rules are loaded

If you are upgrading an existing graph that loaded the configuration document describing precedence rules, this graph will no longer work. The `precedence_rules` document has been removed from the MDEX Engine in this release.

Replace your existing graph with the new graph that includes components for reading the precedence rules configuration from a file, reformatting the file, and then loading the rules as records, using the **WebServiceClient** component in LDI.

For information on loading precedence rules records into the MDEX Engine, see the section about precedence rules in the *LDI MDEX Engine Components Guide*.

Running the upgraded graphs

After you have made changes to the existing graphs, you can run them in LDI.

After running the graphs and troubleshooting any errors, you can upgrade Latitude Studio, and proceed to working with your records using Latitude Studio.

Upgrading Latitude Studio

This section describes how to upgrade to Latitude Studio 2.2.x.

Related Links

Requirement to use the same context root on page 13
The upgrade process assumes that you are using the same installation path and URL. You cannot change the context root during the upgrade process.

Backing up your Latitude Studio 2.1 files on page 14
The first step in migrating to Latitude Studio 2.2.x is to back up the files from Latitude Studio 2.1.

Installing Latitude Studio 2.2.x and restoring backups on page 14
You next install Latitude Studio 2.2.x. After installing, you use your backed-up files to restore your data and custom settings. You delete the directory containing your previous version, then start Latitude Studio 2.2.x.

Reconfiguring existing components after the upgrade on page 15
Upgrading to Latitude Studio 2.2.x may cause your Latitude Studio components to lose some or all of their configuration.

Requirement to use the same context root

The upgrade process assumes that you are using the same installation path and URL. You cannot change the context root during the upgrade process.

Backing up your Latitude Studio 2.1 files

The first step in migrating to Latitude Studio 2.2.x is to back up the files from Latitude Studio 2.1.

To back up your Latitude Studio 2.1 files:

1. Stop your Latitude Studio server.
2. Back up your database as follows:
   - If you are using HSQL, skip this step. Your database will be backed up as part of step 4 below.
   - If you are using MYSQL, DB2, or some other RDBMS, follow the backup procedures from your vendor.
3. Back up any Latitude Studio customizations you have made, such as database connectivity options in `portal-ext.properties`.
4. Back up the entire `endeca-portal\data` directory to a safe backup location.

   If you have changed the location of anything normally kept in `endeca-portal\data`, such as your data source definitions, JCR repository, or Lucene search indexes, back up your custom location(s) as well.
5. After backing up the files, rename the `endeca-portal` directory.

   This ensures that the Latitude Studio 2.2.x installation will not overwrite any of the existing files.

   You will remove this directory when you complete the Latitude Studio 2.2.x installation.

Installing Latitude Studio 2.2.x and restoring backups

You next install Latitude Studio 2.2.x. After installing, you use your backed-up files to restore your data and custom settings. You delete the directory containing your previous version, then start Latitude Studio 2.2.x.

To complete the upgrade to Latitude Studio 2.2.x:

1. Install Latitude Studio 2.2.x, following the steps in the *Latitude Installation Guide*. Do not start the server.
2. Restore the entire `endeca-portal\data` directory from your safe backup location. Overwrite the `endeca-portal\data` files that were installed with the Latitude Studio 2.2.x.

   If you have changed the location of anything normally kept in `endeca-portal/data`, such as your data source definitions, JCR repository, or Lucene search indexes, restore and re-verify your custom locations as well.
3. Restore your database, as follows:
   - If you are using HSQL, skip this step, since your database was restored as part of step 2.
   - If you are using MYSQL, DB2, or some other RDBMS, your database should still be intact. If it is not, follow the restore procedures from your vendor.
4. Configure Latitude Studio to connect to your database.

   If you are not using HSQL, you should have created a backup of your database connection strings earlier.
5. Delete the renamed directory containing your earlier version of the Latitude Studio.
6. Place the Latitude Studio license file in the `endeca-portal\deploy` directory.

The license file is available from the Latitude Studio section of EDeN (*http://eden.endeca.com*).

7. Start Latitude Studio 2.2.x.

   Before you start to use the application, make sure that you wait until all of the components have been deployed and registered. To verify that the upgrade is truly complete, you can monitor the log files to see when all activity has stopped.

Reconfiguring existing components after the upgrade

Upgrading to Latitude Studio 2.2.x may cause your Latitude Studio components to lose some or all of their configuration.

The components will still display on your pages, but may for example revert to default settings.

After you complete the upgrade, review all of your components and reconfigure them as needed to restore your preferred settings.

Chapter 2

Required Changes

You must make the changes listed here if they apply to your application.

Latitude Data Integrator platform support and installation information

To obtain the most accurate platform support and installation information for the Latitude Data Integrator, see the *Latitude Installation Guide*, and not the *LDI Designer Guide* or *LDI Server Guide*.

The information in the *Latitude Installation Guide* supersedes what is found in the *LDI Designer Guide* and *LDI Server Guide*.

Changes to supported platforms (as of version 2.2.x)

As of version 2.2.x, Latitude Studio has eliminated support for some older platforms and operating systems.

Latitude Studio no longer supports:

- Windows 2003
- Java 1.5
- Tomcat 5.5
- WebSphere Application Server 6.1

Clarification of Latitude Data Integrator platform support

In version 2.2.x, the LDI Designer must be installed on 64-bit platforms. That is, it is not supported on 32-bit platforms.

Changes to Latitude Studio log files (as of version 2.2.x)

The Latitude Studio log files have been renamed to reflect the name of the application. The default location of the log files also has been changed.

Change to the log file names

The Latitude Studio log files have been renamed as follows:

Previous Name	New Name	Description
`df.log`	`LatitudeStudio.log`	Main log file for Latitude Studio
`df-metrics.log`	`LatitudeStudio-metrics.log`	Performance metrics log file for Latitude Studio.

Change to the default log file location

As of version 2.2.x, the logger tries to create the log files in the following directory:

Tomcat:	The default location is:
	`<value of catalina.home>/logs`
	For example, if `catalina.home` is set to `C:\endeca-portal\tomcat-6.0.29`, then the log files are located in:
	`C:\endeca-portal\tomcat-6.0.29\logs`

WAS 7:	The default location is either the value of `SERVER_LOG_ROOT` or the value of `LOG_ROOT`.
	1. If `SERVER_LOG_ROOT` is defined, then the log files are located in:
	`<value of SERVER_LOG_ROOT>`
	`SERVER_LOG_ROOT` might be set to something like `/usr/local/WAS/AppServer/profiles/AppSrv01/logs/server1`
	2. If `SERVER_LOG_ROOT` is not defined, then the log files are located in:
	`<value of LOG_ROOT>`
	`LOG_ROOT` might be set to something like `/usr/local/WAS/AppServer/profiles/AppSrv01/logs`

If the logger can't place the file in the default directory, then you typically can find the log files in one of the following locations:

Tomcat - startup script:	If you start Tomcat by running a startup script, the log files are located where the script was run. For example, if you ran the startup script from `tomcat-<version>/bin`, the log file also is in `tomcat-<version>/bin`.
Tomcat - Windows service:	If you register and start Tomcat as a Windows service, the log files may be in `C:\Windows\System32` or `C:\Windows\SysWOW64`.
Tomcat - Eclipse server:	If Tomcat is a server inside of Eclipse, the log files may be located in the root of the Eclipse directory.

WAS 7	For WAS 7, the log files are located relative to the profile's working directory. For example, `/usr/local/WAS/AppServer/profiles/AppSrv01`.

Changes to the State Manager interface (as of version 2.2.x)

The State Manager interface has changed as of version 2.2.x. If you are using a custom State Manager, you will need to update your code.

The specific changes are as follows:

- State Managers must now implement the new `MDEXStateManager` interface.

  `AbstractMDEXStateManager` implements `MDEXStateManager`. If your current State Manager extends `AbstractMDEXStateManager`, then you do not need to make any changes for this new requirement.

- State Managers must now specify a `handleStateReset(PortletRequest request, MDEXState mdexState)` method.

  Previously this was supplied by `AbstractMDEXStateManager`.

  To use the previous implementation, you can copy it from `DefaultMDEXStateManager`.

For details on creating and implementing custom State Managers, see the *Latitude Developer's Guide*.

Transaction-related changes

This topic lists changes that are required in your existing implementation if you are planning to utilize transactions.

This topic highlights the changes you need to make in your project if you choose to run transactions; it does not describe the details for running transactions.

In general, you are not required to use transactions. However, Endeca recommends to use them for updates, if you are running a cluster of MDEX Engine nodes. Even in non-clustered environments, using transactions may be beneficial — all operations that run within a transaction succeed or fail as a unit, providing you with greater operational control over your application.

For information on how to wrap your existing graphs inside a transaction, and how to create and run a transaction graph, see the *LDI MDEX Engine Components Guide*.

If you want to make it possible to run both transaction-aware and transaction-free graphs, a few changes are required in your project:

- Set the `MDEX_TRANSACTION_ID` parameter in `workspace.prm` to an empty value.
- Specify the `MDEX_TRANSACTION_ID` parameter in component's configuration.
- If a transaction fails to commit, use one of the sample graphs from the Quick Start project to manually commit it.

Each of these steps is described below.

Specifying the transaction ID

The default `workspace.prm` file does not contain any references to the outer transaction ID, since the `MDEX_TRANSACTION_ID` parameter is specific to the MDEX Engine and is used to control transactions.

When you start a new Latitude project in LDI Designer, if you want to have an option to use transactions, add the line similar to the following to your `workspace.prm` file:

```
MDEX_TRANSACTION_ID=
```

If you are not planning to use transactions, this line does not affect your project. However, it allows to use transactions without any further modifications to the project's parameter file.

Specifying an outer transaction ID parameter

This example illustrates how to specify an outer transaction ID parameter in the component's configuration.

Latitude-specific components automatically reference this ID, if it is specified in the `workspace.prm` file for your project. However, you need to configure non-Latitude components (that use MDEX Engine web services or bulk ingest interface) to reference this ID, if you plan to use these components in graphs that run transactions.

For example, if you are using a **WebServiceClient** component for running any of the MDEX Engine web services, and plan to use this component inside an outer transaction, the **Request Structure** field for the component must include an attribute `outerTransactionId` with an ID of an outer transaction.

Specify the following request In the **Request Structure** field for your component:

```
<config-service:configTransaction
xmlns:config-service="http://www.endeca.com/MDEX/config/services/types"
outerTransactionId="${MDEX_TRANSACTION_ID}">
<config-service:putGroups
xmlns:config-service="http://www.endeca.com/MDEX/config/services/types"
xmlns:mdex="http://www.endeca.com/MDEX/XQuery/2009/09">
...
</config-service:putGroups>
</config-service:configTransaction>
```

where the string `outerTransactionId="${MDEX_TRANSACTION_ID}"` specifies the ID of the outer transaction listed in the `workspace.prm` file for your project.

Even if you do not use transactions, Endeca still recommends to include `outerTransactionID` attribute in the component's request:

- When a component is used inside a **Transaction RunGraph**, this graph overrides the empty ID value with the string value `transaction`, for the duration of the transaction.
- When a component is used in a transaction-free graph, the request to the MDEX Engine uses the empty value of the ID from `workspace.prm`. The MDEX Engine ignores the transaction ID attribute if its value is empty — this allows to run the web service requests outside transactions.

Manually committing failed transactions

When building and troubleshooting a project in LDI that utilizes transactions, you may need to manually commit a transaction.

Normally, a graph that starts a transaction should finish by successfully committing it, however, in some instances, especially during the development stage, the transaction does not commit and you

need to manually commit it. For example, you may need to manually commit a transaction if the **Transaction RunGraph** is configured to **Do Nothing** upon failure.

Two sample graphs from the Quick Start project let you explicitly commit a transaction — the **RollBackTransaction**, and the **CommitTransaction** graphs.

Alternatively, to manually commit a transaction that remained opened, you can issue an `/admin?op=rollback&outerTransactionId="ID"` command on the MDEX Engine server, specifying the transaction ID as the string `transaction`.

Change of the namespace for the Configuration Web Service

The namespace for the Configuration Web Service has changed.

The namespace is now specified as in the following example:

```
<config-service:configTransaction
xmlns:config="http://www.endeca.com/MDEX/config/services/types"
xmlns:mdex="http://www.endeca.com/MDEX/XQuery/2009/09">
```

You need to be aware of this change if you have used requests to the Configuration Web Service in the configuration of components in LDI, or if you are using other methods, such as SOAPUI, to issue configuration requests to the MDEX Engine.

Changes to the precedence rules

The way precedence rules are configured in the MDEX Engine has changed.

In this release, precedence rules are configured as records in the MDEX Engine. Therefore, like all other types of records, they can be loaded into the MDEX Engine index.

Two scenarios are possible:

- New projects in which you would like to define precedence rules.

  If you are starting a new project, you can create a graph in LDI that would read the rules configuration from a file, reformat it, and then load the rules as records, using the **WebServiceClient** component in LDI. This component should be configured to use the Configuration Web Service operations for loading precedence rules. (The MDEX Engine Configuration Web Service has been extended to allow loading precedence rule records.)

- Upgrading existing projects that loaded precedence rules as configuration documents.

  If you are upgrading an existing project to this release, your existing graph that loaded the configuration document describing precedence rules will no longer work. Replace it with the new graph that includes components for reading the rules from a file, reformatting it, and then loading the rules as records, using the **WebServiceClient** component in LDI.

For information on loading precedence rules records into the MDEX Engine, see the section about precedence rules in the *LDI MDEX Engine Components Guide*.

Specifying an outer transaction ID in web service requests

All MDEX Engine web services, as well as the Bulk Ingest interface, require specifying an ID of an outer transaction as an attribute, if one is open.

Any request to any of the MDEX Engine web services can contain an optional attribute `outerTrans¬actionId` that specifies the ID of an outer transaction (if it has been started by the Transaction Web Service).

This attribute must be specified only if a request made by the web service is started after a request to start a transaction has been made by the Transaction Web Service.

If no transactions have been started, the `outerTransactionId` attribute can still be present but its value should be empty.

The outer transaction ID is specified as an attribute on the operation sent to the web service, as in the following example:

```
<ingest:clearMdex outerTransactionId="myID"/>
```

where `myID` is the ID of the outer transaction, if the outer transaction has been started previously with this ID.

Note that if you specify an invalid outer transaction ID, the operation will fail with the notification of the ID mismatch. In addition, if no outer transactions have been started, but you specify an ID, this request will also fail.

In general, web service operations should rarely be used directly through the web services tools, such as SOAPUI. In most instances, you use web service operations by utilizing one of the existing Latitude connectors in LDI. All Latitude connectors that utilize calls to the MDEX Engine web services or the Bulk Ingest interface automatically reference the ID of an outer transaction, if it is specified in the `workspace.prm` file for your project.

In the rare instances when you issue web service requests directly to the MDEX Engine (and don't use LDI for running transactions), to identify an outer transaction ID, you should use the MDEX Engine request log (`dgraph.log`). The successful response to the `StartTransactionOperation` that starts an outer transaction returns an outer transaction ID. This is the ID that must be specified in all subsequent web service requests that run while the transaction remains in progress.

Specifying an outer transaction ID for admin?op=updateaspell

If `admin?op=updateaspell` is started after an outer transaction has been started, it must reference a transaction ID.

You can specify the ID as in the following example:

```
admin?op=updateaspell&outerTransactionId="transaction"
```

where `transaction` is the ID of the outer transaction that has been started in the MDEX Engine.

Note: For this release, if you are using transactions outside of LDI, it is important to note the outer transaction ID. The transaction ID is returned in a Transaction Web Service response to the successful `StartTransactionOpearation`. When you use transactions through LDI, you

should specify the ID in the `workspace.prm` file for your project in LDI, and reference that ID for all subsequent requests to the MDEX Engine that require it.

Clustering changes

The following changes are required if you run a cluster of MDEX Engine nodes.

The topics in this section highlight the changes made in this release in relation to clustering. These topics do not include complete information about running a cluster of MDEX Engine nodes. For more information on running a cluster, see the *Latitude Administrator's Guide*.

The flag for configuring follower nodes has changed

In this release, to start a follower node, run `--follower <node_name>`.

In the previous release, you used the `--replica-name <node_name>` to start a follower node. In this release, the command is:

```
--follower FollowerNode1
```

where `FollowerNode1` 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).

Before starting an MDEX Engine that will serve as a follower node, ensure that the leader node has been started. (If the leader node was not started, the follower nodes will not start.)

In addition, confirm 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.

Running baseline updates in a clustered environment

Endeca recommends that you wrap updates in outer transactions when using a cluster, although you may run updates on the leader node without using transactions.

Updates that run within an outer transaction succeed or fail as a unit. As a result, page views in Latitude Studio reflect either the pre-update state of the index, or the state after all updates have been committed.

Changes to load balancer configuration

If you are using a load balancer between the Latitude Studio servers and the cluster of MDEX Engine nodes, change the load balancer configuration so that it does not direct query requests to the leader node while that node is running updates.

In a cluster, updates are sent to the leader node only. During an outer transaction, the leader node responds to any queries, including the `admin?op=ping` command, with an HTTP status code 403.

This way, the load balancer can be configured to automatically detect whether a transaction is in progress and remove the leader node from answering queries, while other nodes in the cluster continue to respond to user requests in Latitude Studio.

Chapter 3

Behavioral Changes

The changes listed here do not require any action on your part, but will affect how your application behaves after you upgrade.

Additional Liferay functions disabled (as of version 2.2.x)

The default version of `portal-ext.properties` has been updated to disable some additional functionality. These functions have been disabled in order to improve performance.

If you are not currently using any of these functions, then you will not be affected by this change.

If you are currently using any of these functions, then you can continue to use them. When you restore your own version of `portal-ext.properties` during the upgrade, your current settings will be restored.

The disabled functions are:

Description	Entry in portal-ext.properties
Disables the **Monitoring** component of the Control Panel.	`session.tracker.memory.enabled=false`
Prevents the style sheet from being updated for an individual component. Removes the **Look and Feel** menu option for components.	`portlet.css.enabled=false`
Disables audit trails for compliance.	`com.liferay.portal.servlet.filters.audit.AuditFil¬ter=false`
Disables automatic logins, usually used for single sign-on.	`com.liferay.portal.servlet.filters.autologin.Au¬toLoginFilter=false`
Removes support for CAS, NTLM, and OpenSSO single sign-on/authentication providers.	`com.liferay.portal.servlet.filters.sso.cas.CASFil¬ter=false` `com.liferay.portal.servlet.filters.sso.ntlm.Ntlm¬Filter=false`

Description	Entry in portal-ext.properties
	`com.liferay.portal.servlet.filters.sso.ntlm.Ntlm¬ PostFilter=falsecom.liferay.portal.servlet.fil¬ ters.sso.opensso.OpenSSOFilter=false`
Removes support for exposing Liferay's Document Library from within Sharepoint or Microsoft Office.	`com.liferay.portal.sharepoint.SharepointFil¬ ter=false`
Disables performance monitoring.	`com.liferay.portal.servlet.filters.monitoring.Mon¬ itoringFilter=false`

Change to the default value for the data source version (as of 2.2.x)

Previously, Latitude data sources were required to include the parameter `"apiVersion" : "DIS¬ COVERY_SERVICE"`. If this parameter was not included, then the data source was assumed to be for the previous version of the MDEX Engine.

As of 2.2.x, if the `apiVersion` parameter is not included in the data source, the default value is `DISCOVERY_SERVICE`.

You no longer need to include the `apiVersion` parameter in your data source files.

Note that if you do include the `apiVersion` parameter, it must be set to `DISCOVERY_SERVICE`.

Changes to the documentation set

This section lists changes that were made to the contents of the documentation set for this release.

The Latitude Migration Guide (this guide)

The *Latitude Migration Guide* is new in this release. It provides information on how to upgrade your existing project from Latitude release version 2.1 to the Latitude release version 2.2.x.

Changes to the set of Latitude Data Integrator Guides

The documentation for using the Latitude Data Integrator has been significantly expanded and now includes the following guides:

- The *Latitude Data Integrator Getting Started Guide*. Use this guide for information on how to get started with the Latitude Data Integrator Designer. This guide provides a high-level overview of the LDI Designer, and then outlines how to use the LDI Designer on a single Windows machine to create your first LDI project that loads data into the MDEX Engine.
- The *LDI MDEX Engine Components Guide*. Use this guide for information on how to use the Latitude-specific components (also known as connectors) in LDI for loading the data and configuration into the MDEX Engine index, as well as for running operational tasks, such as updates. This guide also describes in detail all Latitude sample graphs that are available in the Latitude Quick Start project.

Note: This guide has been renamed compared with the previous release. In the previous release, this guide was known as the *Latitude Data Integrator Guide*.

- The *Latitude Data Integrator Designer Guide*. Use this guide for information on how to use the LDI Designer for data management and loading tasks. This guide describes how to use those LDI components that are not specific to Latitude and the MDEX Engine.
- The *Latitude Data Integrator Server Guide*. Use this guide if you are implementing a data loading solution in an enterprise-wide environment. Such an environment requires using both the LDI Designer and the LDI Server. The LDI Server provides centralized ETL job management and enables integration into enterprise workflows.

The Liferay Portal Administrator's Guide

The documentation set includes the *Liferay Portal Administrator's Guide*. This guide, together with the *Latitude Studio User's Guide* and the *Latitude Administrator's Guide* addresses administrative tasks for running Latitude Studio in a Liferay portal.