Endeca® Latitude

Installation Guide

Version 2.1.0 • June 2011

Contents

Copyright and disclaimer

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 contains installation instructions for setting up Endeca Latitude on Windows and Linux.

Installing Endeca Latitude consists of installing its three parts:

- MDEX Engine
- Latitude Data Integrator
- Latitude Studio

Who should use this guide

This guide is intended for system administrators installing Endeca Latitude on Windows or Linux, as well as for developers who are building applications using Endeca Latitude.

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*.

Part 1

Before You Begin

Chapter 1

Endeca Latitude Installation Overview

This chapter provides a brief overview of the Endeca Latitude components and the installation process.

About this release

Read this section to understand the assumptions under which you can use this release. This section lists those aspects of the release that may change in the future. These aspects may represent limitations in the configuration process or feature availability.

Consider the following characteristics of this release:

- **Support for the MDEX Engine, Latitude Studio, and Latitude Data Integrator is limited to those versions that are included in this release of Latitude**. This release relies on the index created with the version of the MDEX Engine that supports Endeca Latitude. Similarly, this release only supports Latitude Studio and Latitude Data Integrator that are compatible with Endeca Latitude.
- **Options for loading the data sources**. In this release, you have several options for loading the data. The recommended way of loading the data is through the Latitude Data Integrator that is part of the Latitude Information Integration Suite. If your implementation already includes Informatica, the Endeca Integration for Informatica PowerCenter should be used to load data into the MDEX Engine. Other options for loading data include using the Data Ingest Web Service directly or the Java Loader utility that is available for download from the Endeca Services organization.
- **Notes about the configuration and development process**. In this release, the configuration and development process takes place in the Latitude Data Integrator and is complemented by options available to the power users in Latitude Studio.

Overview of the Latitude modules

The Endeca Latitude product consists of three major modules.

For Latitude 2.1, these modules are:

- Endeca MDEX Engine
- Endeca Latitude Data Integrator
- Endeca Latitude Studio

Each of these modules has its own installation package.

Note: These Latitude 2.1 modules are compatible with one another. However, modules shipped as part of Latitude 2.1 are not compatible with earlier versions.

About the MDEX Engine

The Endeca MDEX Engine is the indexing and query engine that provides the foundation for all Endeca solutions.

The MDEX Engine uses proprietary data structures and algorithms that allow it to provide real-time responses to client requests. The MDEX Engine stores the indices that were created from the Data Ingest Web Service. After the indices are stored, the MDEX Engine receives client requests via the application tier, queries the indices, and then returns the results.

The MDEX Engine is designed to be stateless. This design requires that a complete query be sent to the MDEX Engine for each request. The stateless design of the MDEX Engine facilitates the addition of MDEX Engine servers for load balancing and redundancy. Because the MDEX Engine is stateless, any replica of an MDEX Engine on one server can reply to queries independently of a replica on other MDEX Engine servers.

The Dgraph is the name of the process for the MDEX Engine. Because the Dgraph is key to every Endeca implementation, its performance is critical. A typical Endeca implementation includes one or more Dgraphs.

About the cluster of MDEX Engine nodes

A cluster is composed of a set of MDEX Engine nodes, all of which can serve query requests. Only one node is identified as the leader node; All other nodes are follower nodes. There is one copy of the on-disk representation of the MDEX Engine index that is shared and used by all MDEX Engine nodes. The Cluster Coordinator provides communication between the nodes in the cluster. It also notifies the reader nodes about index updates and updates to the configuration.

With a cluster of nodes, access to the MDEX Engine and its index becomes highly available. If one node fails, queries continue to be processed by other nodes in the cluster. A cluster also provides increased throughput by the MDEX Engine. By adding nodes to a cluster you can spread the query load across multiple MDEX Engine instances without the need to increase storage requirements at the same rate. Nodes can be added or removed dynamically, without having to stop the cluster.

In the development environment, you can start with a single instance of the MDEX Engine that is not part of a cluster. (Without the cluster services, having a single running MDEX Engine instance is a valid configuration for starting in the development environment.) You can then expand your single instance MDEX Engine implementation by first building a simple single-node cluster configuration and then adding more MDEX Engine nodes. When you move to a production environment, you can duplicate a multi-node cluster that you built in the development environment.

In this release, the cluster implementation requires that you download and install the Cluster Coordinator package available from the **Downloads** section of the Endeca Developer Network (EDeN). For information on downloading and installing the Cluster Coordinator package, see the chapter *"Deploying Latitude in a Cluster"* in the *Latitude Administrator's Guide*.

About the Latitude Data Integrator

The Latitude Data Integrator is a high-performance data integration platform that lets you extract source records from a variety of source types (from flat files to databases).

The Latitude Data Integrator then sends that data to the MDEX Engine via the Data Ingest Web Service or the Bulk Load Interface.

From a high level, the Latitude Data Integrator suite consists of:

- The LDI Designer. With its powerful graphical interface, you can build graphs that can load source data into the MDEX Engine, as well as the schema for your attributes and the index configuration documents for various MDEX Engine features.
- The Latitude connectors. Developed by Endeca, these connectors are Designer components that are easily configured to perform various data ingest operations.
- The LDI Server. The Server is used to run graphs in an enterprise-wide environment. In this environment, different users and user groups can access and run the graphs.

Note: For those customers wishing to use Informatica PowerCenter for their Endeca ETL needs, they can purchase PowerCenter directly from Endeca. Usage of Informatica in this case is restricted to the context of the Endeca Latitude 2.1 application license, and includes the MDEX Engine Connector for Informatica PowerCenter.

About Latitude Studio

Latitude Studio enables rapid configuration of dashboard applications that offer the highly interactive Guided Navigation® user experience across a full range of structured and unstructured enterprise data.

Latitude Studio is easy to deploy and ideal for the agile development of enterprise-quality applications. Due to the component-based nature of Latitude Studio, these applications are simple to control, adapt, and extend. It provides granular layout and configuration control to enable users to manage and personalize their own experiences.

Latitude Studio consists of an enterprise-class portal framework and a library of UI components that embody best practices in Endeca applications.

About the Component SDK

The Component SDK is a packaged development environment for portlets, themes, layout templates, and other portal elements. Endeca has modified Liferay's version of its Plugins SDK to include the Endeca enhancements, such as the `EndecaPortlet` core class.

The installation and use of the Component SDK is covered in the *Latitude Developer's Guide*.

Interaction with Liferay Portal

Latitude Studio is built upon the Liferay Portal Enterprise Edition.

Liferay Portal is an open-source JSR-286 portal technology. Latitude Studio extends basic Liferay functionality to provide enhanced user management, security, and cross-component interaction, as well as performance-optimized communication with Endeca MDEX Engines.

This version of Latitude Studio is built upon Liferay Portal 5.2 Enterprise Edition Service Pack 5.

Installation order

Following the recommended order of installation helps you minimize component dependencies.

Endeca recommends that you install the components in this order:

1. Latitude MDEX Engine (see *Installing the MDEX Engine* on page 25). After installing, to verify the installation, run the `mkmdex` command, and then start the MDEX Engine.

   Note that you can start the MDEX Engine without loading any of your source data records.

2. Latitude Data Integrator (see *Installing the Latitude Data Integrator* on page 39). After installing, open the LDI Designer.
3. Latitude Studio (see *Installing Latitude Studio* on page 45). After installing, to verify the installation, log in to Latitude Studio. Use the running MDEX Engine as the data source.

The verification procedures for each component are described in the installation section for that component.

Chapter 2

Latitude System Requirements

This chapter describes the requirements for each component of an Endeca Latitude installation.

MDEX Engine system requirements

This version of the Endeca MDEX Engine has the following requirements:

Hardware requirements

Endeca software has the following hardware requirements. These requirements apply to all of the Latitude components.

Note: In this guide, the term “x64” refers to any processor compatible with the AMD64/EM64T architecture. You might need to upgrade your hardware, depending on the data you are processing. All run-time code must fit entirely in RAM. Likewise, hard disk capacity must be sufficient based on the size of your data set. Please contact your Endeca representative if you need more information on sizing your hardware.

Windows and Linux on x64

Minimum hardware requirements:

- x64 processor, minimum 1.8 GHz
- At least 2 GB of RAM, depending on the size of the application data set
- 80 GB hard drive, depending on the size of the application data set

Recommended hardware requirements:

- x64 3.0+ GHz processors; Endeca recommends Intel Xeon (including Nehalem) or AMD Opteron processors
- 8 GB of RAM or more, depending on the size of the application data set
- High performance network-attached storage (for example, attached via a dedicated iSCSI or fibre channel network) or high performance locally-attached RAID storage (for example, a RAID 6 or RAID 0+1 array with battery-backed write caching, operating on 72GB or 146 GB spindles at 10k or 15k RPM spindle speed)
- Gigabit Ethernet

Supported operating systems

The Endeca software supports the following 64-bit operating systems running on servers with x64 capabilities:

Platform	Description
LINUX RHEL 5	• Red Hat Enterprise Linux Server (version 5 for x64) running on x64 processors. • Red Hat Enterprise Linux Advanced Platform (version 5 for x64) running on x64 processors. For best performance on Red Hat Linux version 5 (Server and Advanced), Endeca recommends the latest version of RHEL 5.
Windows 2008	Windows Server 2008 R2 Enterprise running on x64 processors.

Note: Windows 7 is not supported for production deployment, but operates sufficiently to enable training and small-scale development work.

Note: Windows XP is not supported.

Support for VMware

The MDEX Engine is supported on VMware ESX 3.5 for the following guest operating system platforms: Windows 2008 and RHEL 5 for Linux.

The recommended MDEX Engine server configuration for Windows and Linux are:

- Configure four VCPUs on a virtual machine.
- Allocate a single Dgraph per virtual machine.
- Specify four threads for each Dgraph.

The number of threads should not exceed the number of VCPUs. Endeca does not recommend running more than one MDEX Engine per virtual machine.

Linux utilities dependencies

The MDEX Engine installer requires several Linux utilities.

The following Linux utilities must be present in the `/bin` directory:

```
basename
cat
chgrp
chown
date
dd
df
mkdir
more
rm
sed
```

```
tar
true
```

The following Linux utilities must be present in the `/usr/bin` directory:

```
awk
cksum
cut
dirname
expr
gzip
head
id
printf
tail
tr
wc
which
```

If these utilities are not in the specified locations, the installation fails with a message similar to the following:

```
Required dependency is not executable: /bin/df. Aborting.
```

Disk space requirements

You should ensure that adequate disk space is available before installing the MDEX Engine.

On Windows, the installation process requires a minimum of 400 MB in the system partition and 200 MB in the target partition. To avoid an "out of drive space" error during the installation process, you should allow the minimum of memory required on the system and target partitions.

The Windows installation process unpacks its `.MSI` installation file, and other temporary and log files, to a location on the system drive (typically the user's %TEMP% folder). These files add another 400MB during the installation. When the process completes the installation, it deletes the temporary files and frees the 400 MB space it consumed.

On Linux, the MDEX Engine unpacks to approximately 430 MB. Because multiple versions may eventually be stored, a destination in a large disk partition is recommended.

Required Endeca components

The MDEX Engine installation does not require any other Endeca components to be previously installed.

Latitude Data Integrator system requirements

The Latitude Data Integrator has the same installation requirements as the MDEX Engine.

The Latitude Data Integrator is compatible only with the Latitude 2.1 version of the MDEX Engine.

In addition, you need the Java version 6 (also called 1.6) JDK, which is included in the Latitude Data Integrator Designer installer.

Latitude Studio system requirements

This version of Endeca Latitude Studio has the following requirements:

Hardware requirements

The hardware requirements for Latitude Studio are the same as those for Endeca MDEX Engine, which are listed in the "Hardware requirements" topic under "MDEX Engine system requirements."

Supported operating systems

Latitude Studio is supported on the same Windows and Linux operating systems as the Endeca MDEX Engine, which are listed in the "Supported operating systems" topic under "MDEX Engine system requirements."

Software requirements

Latitude Studio is a Web-based application that runs in an application server. It supports the following software:

Software	Supported Versions
Application server	• Tomcat 6 • Tomcat 5.5 • WebSphere Application Server (WAS) 6.1.0.1 or higher • WebSphere Application Server (WAS) 7
Java	• Tomcat 6 is supported with Sun Java 6 • Tomcat 5.5 is supported with Sun Java 5 • WAS 6.1 is supported with IBM Java 5 • WAS 7 is supported with IBM Java 6  **Important:** For Sun Java 6, update 18 or greater is required.
Database system	• MySQL 5.1 • DB2 9.5
Browser	• Firefox 3.6 on Windows • Internet Explorer 8 (with compatibility mode disabled) on Windows  **Tip:** Firefox is recommended.  **Important:** Running Internet Explorer 8 in compatibility mode is not supported.
Browser plugin	Adobe Flash 10.0

Alternative database support

The Liferay Portal server uses a relational database to store configuration and state. By default, Liferay uses Hypersonic, but this is not recommended for production use due to performance issues. Endeca tests Latitude Studio on MySQL and DB2. However, many other databases are expected to work.

Customers should feel free to use any database, including shared systems they may already have in place. As with application servers, customers who choose to deploy on un-tested databases will always be supported on any issue that can be traced back to core Latitude Studio code and can be reproduced on a supported database.

The Latitude Studio section of this guide, combined with the *Liferay Portal Administrator's Guide*, provides detailed instructions on how to switch to another database system.

Changing the JavaScript time-out value on Internet Explorer 8

Internet Explorer 8 keeps track of the number of JavaScript lines executed. After a fixed value, the browser issues an error message, prompting the user to decide whether he or she would like to continuing running the script. Because Latitude Studio is a rich Internet application that leverages JavaScript heavily in all components, it can trigger this error during normal usage.

Microsoft describes this issue in Knowledge Base Article 175500 and specifies a fix. More detail can be found in the Knowledge Base Article (*http://support.microsoft.com/kb/175500*). The following is a condensed version of Microsoft's fix for the Internet Explorer 8 issue.

Important: If you intend to run Latitude Studio on Internet Explorer 8, Endeca strongly recommends that you apply the fix outlined in this topic.

To change the script time-out value:

1. Using a registry editor such as `Regedt32.exe`, open this key:
   ```
   HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Styles
   ```
   

   Note: If the `Styles` key is not present, create a new key that is called `Styles`.
2. Create a new `DWORD` value called "MaxScriptStatements" under this key, and set the value to the desired number of script statements. You will have to try different values for your application environment, but the suggested starting point is a `DWORD` value of 0x1CFFFFF.

   Note: You can turn off this Internet Explorer 8 feature using a `DWORD` value of 0xFFFFFFFF.

Compatibility with other Latitude modules

This document assumes that you already have a running MDEX Engine at which you can point Latitude Studio.

This version of Latitude Studio is compatible with the version of the MDEX Engine that is available with this Latitude release.

Chapter 3

Downloading the Latitude Software

Downloading the Latitude software

You can download Endeca Latitude modules from the Downloads section of the Endeca Developer Network (EDeN).

To download the Latitude software:

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 2**.
5. In the **Current Releases** table, click **Latitude 2.1**.
6. In the **Product Downloads** page, download the appropriate version of the MDEX Engine for your platform:

To install:	Download:	Resulting downloaded file:
MDEX Engine on Windows	**MDEX Engine for Windows 64-bit**	Latitude_2.1.0_mdex_ x86_64pc-win.exe
MDEX Engine on Linux	**MDEX Engine for Intel Linux 64-bit**	Latitude_2.1.0_mdex_ x86_64pc-linux.sh

Note: You will notice the MDEX Engine Cluster Coordinator package, which is required only if you would like to run a number of MDEX Engine nodes in a clustered environment. For information on downloading, installing, and using a cluster, see the *Latitude Administrator's Guide*.

7. In the **Product Downloads** page, download the appropriate set of the Latitude Data Integrator files for your platform:

To install:	Download:	Resulting downloaded file:
LDI Designer for Linux	**Data Integrator Designer for Linux**	Latitude_2.1.0_data-integrator-designer_linux.tgz
LDI Designer for Windows	**Data Integrator Designer for Windows**	Latitude_2.1.0_data-integrator-designer.exe
LDI Components for Windows	**Data Integrator Components for Windows**	Latitude_2.1.0_data-integrator-components.win.zip
LDI Server	**Data Integrator Server**	clover.war
LDI Server License	**Data Integrator Server License**	clover-license.war

8. In the **Product Downloads** page, download the appropriate Latitude Studio files, depending on your installation environment:

To install:	Download:	Resulting downloaded files:
Tomcat 6 bundle for Windows	• **Studio for Windows** • **Studio Components** • **Studio License**	• Latitude_2.1.0_endeca-portal.zip • Latitude_2.1.0_components.zip • latitude_2.1.0_studio_license.xml
Tomcat 6 bundle for Linux	• **Studio for Linux** • **Studio Components** • **Studio License**	• Latitude_2.1.0_endeca-portal.tgz • Latitude_2.1.0_components.zip • latitude_2.1.0_studio_license.xml
Latitude Studio for Tomcat 5.5 or Websphere Application Server version 6.1 or 7	• **Studio Standalone WAR** • **Studio Standalone WAR Dependencies** • **Studio Components** • **Studio License**	• Latitude_2.1.0_endeca-portal-war.zip • Latitude_2.1.0_endeca-portal-dependencies.zip • Latitude_2.1.0_components.zip • latitude_2.1.0_studio_license.xml
Chart component (for use with any of the above installations, see the "Installing Corda" section)	• **Studio Charting and Visualization Servlet**	• Latitude_2.1.0_Corda.zip

Part 2

Installing the MDEX Engine

Chapter 4

Installing the MDEX Engine

This section contains instructions for installing the Endeca MDEX Engine.

About User Account Control in Windows Server 2008

User Account Control in Windows Server 2008 R2 limits which tasks Standard Users can run.

User Account Control divides users into two groups - Standard Users and Administrators:

- **Standard Users** have the least amount of privileges required to perform basic tasks. They cannot install or uninstall applications to or from %SYSTEMROOT%, change system settings, or perform other administrative tasks.
- **Administrators** have full permissions for adding, removing, or modifying programs and user accounts.

By default, users are created as Standard Users. Although User Account Control allows Standard Users to temporarily elevate permissions in order to perform administrative tasks, doing so requires administrative credentials.

For more information regarding User Access Control and the permissions granted to each user type, see the Microsoft documentation at *http://technet.microsoft.com/en-us/library/cc731416%28WS.10%29.aspx*.

Effects on MDEX Engine Installation

The MDEX Engine installation process has changed with the introduction of User Account Control in Windows Server 2008 R2. Because of the resulting security restrictions on tasks that run at elevated privilege, running a per-machine installation of the MDEX Engine now requires administrator permissions. A user may still install a per-user installation without these permissions, but this is not supported in a production environment.

Note: UAC behavior and installation steps may differ if installing on Windows 7 for development purposes. Consult the Microsoft documentation for details.

Installer file names

Endeca installation packages and executables are named according to a common convention.

The installer file names follow the format:

```
latversion_mdex_arch-OS
```

For example:

```
latitude_2.1.0_mdex_x86_64pc-linux.sh
```

The *latversion* is the Latitude version. In the example, the `latitude_2.1.0` identifier indicates that this installer is part of the Latitude 2.1.0 product package.

The *arch-OS* is the architecture and operating system identifier for the component being installed. In the example installer, `x86_64pc-linux` identifies the file as an installer for the 64-bit Linux platform. The following table lists the *arch-OS* identifiers and their platforms:

arch-OS identifier	Installation platform
x86_64pc-linux	Linux running on 64-bit Intel processors
x86_64pc-win32	Windows running on 64-bit Intel processors

Installing a per-user MDEX Engine installation on Windows

A per-user installation of the MDEX Engine may be used if administrator permissions are unavailable. This is primarily useful for training and for small-scale development environments.

Note: Installing a per-user installation of the MDEX Engine on Windows Server 2008 is only permitted when it has been configured as a managed application, or the `DisableMSI` registry key has been set to zero. Contact an administrator if you require a per-user installation under Windows Server 2008.

Before installing, make sure to uninstall any previous versions of the MDEX Engine using the **Uninstall a program** utility in the Control Panel.

If a per-machine installation of the MDEX Engine is already present, attempting to install a per-user installation fails and the installer instead attempts to uninstall the per-machine installation.

Important: If you are setting up your MDEX Engine for a production environment, you must use a per-machine installation.

To install the Endeca MDEX Engine as a per-user installation:

1. In your local environment, locate the Endeca MDEX Engine software that you downloaded from the Endeca Developer Network (EDeN).
2. Double-click the installer file `latitude_2.1.0_mdex_x86_64pc-win32.exe` to start the wizard.

   The wizard verifies the contents of the installation package and confirms that no previous version is installed.

   Note: If the installer identifies that the previous version is still installed, cancel the installation and uninstall the previous version using the **Uninstall a program** utility in the Control Panel.

3. Click **Next** to begin the installation process.
4. In the **Copyright and Legal** screen, click **Next**.
5. In the **License Agreement** screen, select **I accept the terms of the license agreement**, then click **Next**.
6. In the **Select Program Folder** screen, select the **Only for me (current user)** radio button, then click **Next**.
7. In the **Destination Folder** screen, either accept the default installation folder (`%USERPROFILE%\Endeca\Latitude\2.1.0\MDEX`) or select another installation location. Then click **Next**.

   You cannot install the MDEX Engine into a directory that contains content.

   The wizard displays both the required and available disk space for the target directory chosen. The MDEX Engine requires approximately 200 MB of disk space. The installer requires approximately 400 MB of space on the system drive for temporary files. These files are cleared after the installation process completes.

   Note: If you install to a non-default location, the installation does not create the subdirectory structure `MDEX\<version>` unless you specify this structure explicitly. Additionally, clicking the **Back** button in the installation wizard resets the installation path to the default directory.

8. In the **Completing the Setup Wizard** screen, click **Next**.

   The wizard begins to install the MDEX Engine files.

9. When the wizard confirms that you have successfully completed the installation, click **Finish**.
10. Open a command prompt and change to the root of your `\MDEX` installation folder.
11. Run the `mdex_setup.bat` file.

    This batch file adds the `bin` and `utilities` directories to the `PATH` environment variable.

    The script itself is optional and provided as a convenience, although it only affects the `PATH` variable for the current user in the current context.

Installing a machine-wide MDEX Engine installation on Windows

If you are setting up your MDEX Engine for a production environment, you must use a machine-wide installation. Additionally, Endeca recommends this method of installation any time administrator permissions are available.

Before installing, make sure to uninstall any previous versions of the MDEX Engine using the **Uninstall a program** utility in the Control Panel.

To install the Endeca MDEX Engine on Windows:

1. In your local environment, locate the Endeca MDEX Engine software that you downloaded from the Endeca Developer Network (EDeN).
2. Double-click the installer file `latitude_2.1.0_mdex_x86_64pc-win32.exe` to start the wizard.

   The wizard verifies the contents of the installation package and confirms that no previous version is installed.

When running an installation with administrator permissions, User Account Control will check the digital signature of the installer. Digital signatures provide system administrators with a higher level of confidence in the authenticity of the installation package.

Note: If the installer identifies that the previous version is still installed, cancel the installation and uninstall the previous version using the **Uninstall a program** utility in the Control Panel.

3. Click **Next** to begin the installation process.
4. In the **Copyright and Legal** screen, click **Next**.
5. In the **License Agreement** screen, select **I accept the terms of the license agreement**, then click **Next**.
6. In the **Select Program Folder** screen, select the **Anyone who uses this computer (all users)** radio button, then click **Next**.
7. In the **Destination Folder** screen, either accept the default installation folder (`C:\Endeca\Latitude\2.1.0\MDEX`) or select another installation location. Then click **Next**.

   You cannot install the MDEX Engine into a directory that contains content.

   The wizard displays both the required and available disk space for the target directory chosen. The MDEX Engine requires approximately 200 MB of disk space. The installer requires approximately 400 MB of space on the system drive for temporary files. These temporary files are cleared after the installation process completes.

   Note: If you install to a non-default location, the installation does not create the subdirectory structure `MDEX\<version>` unless you specify this structure explicitly. Additionally, clicking the **Back** button in the installation wizard resets the installation path to the default directory.

8. In the **Completing the Setup Wizard** screen, click **Next**.

   The wizard begins to install the MDEX Engine files.

9. When the wizard confirms that you have successfully completed the installation, click **Finish**.
10. Open a command prompt and change to the root of your MDEX installation folder. `C:\Endeca\Latitude\2.1.0\MDEX` is the default.
11. Run the `mdex_setup.bat` file.

    This batch file adds the `bin` and `utilities` directories to the `PATH` environment variable.

    The script itself is optional and provided as a convenience, although it only affects the `PATH` variable for the current user in the current context.

Installing silently on Windows

Running the silent installer on Windows has different effects depending on whether or not the user has administrator permissions.

If the silent installer is run with administrator permissions, it creates a per-machine installation. Otherwise, it creates a per-user installation. Variables on the command line can be used to override this default behavior.

To install silently on Windows:

1. From a command prompt, navigate to the directory where you downloaded the installer.

2. Issue the following command:

```
start /wait latitude_2.1.0_mdex_x86_64pc-win32.exe
   /s TARGETDIR=C:\Endeca\Latitude\2.1.0\MDEX
```

   You can replace the TARGETDIR path location in the example with the location to which you want to install. However, if you set the install location to a non-empty directory or to a drive that does not exist, the silent installation will fail with a non-zero status code.

   Additionally, an administrator can override the default behavior and create a per-user installation by setting ALLUSERS=FALSE.

3. Run the `mdex_setup.bat` file in the MDEX Engine root directory.

   This script adds the `utilities` directory and the MDEX Engine binaries to the PATH environment variable. The script itself is optional and provided as a convenience, although it only affects the PATH variable for the current user in the current context.

Turning on logging for the Windows silent installer

When running the silent installer on Windows, you can turn on logging.

This can be useful, for example, if you need to debug a failed silent installation.

To turn on logging during a silent installation on Windows, add `/l=<path>`. An absolute path is required.

Installation steps for Linux

The Endeca software is distributed as a self-extracting tar file and install script. It can be installed at any location.

Note: The MDEX Engine unpacks to approximately 200 MB. Because multiple versions may eventually be stored, a destination in a large disk partition is recommended.

To install the MDEX Engine:

1. Determine where you will install the Endeca system. Verify that the target directory on which you plan to install has enough available disk space, and has write permissions (is not read-only).

   For example, in this procedure we assume that the target directory is `/usr/local/endeca` and that you have write permissions for it.

   If you do not set these permissions, the install script will not run.

2. Locate the MDEX Engine installation file. This procedure assumes the location is `/downloads` and that the name of the installation file is `latitude_2.1.0_mdex_x86_64pc-linux.sh`.
3. Assuming the locations used in steps 1 and 2, run the Endeca installation script with the following command:

```
/downloads/latitude_2.1.0_mdex_x86_64pc-linux.sh
 --target /usr/local
```

   The Endeca license agreement displays.

4. Scroll to the end of the license agreement, then type `Y` to accept the agreement and continue with the installation.

   After you enter `Y` to accept the agreement, the MDEX Installer displays a message that it is about to extract files in the specified directory.

   The installer also checks that the directory has enough available disk space, and that it can write to this directory.

   If these conditions are met, the installer proceeds with the installation and completes it. If they are not met, the installer issues an error and discontinues the installation.

At the completion of the installation, the installer prompts you to run the `mdex_setup` script.

After you install the MDEX Engine, run the script (in the `endeca/Latitude/2.1.0/MDEX` directory) that is appropriate to your shell:

- For Bourne, Bash, or Korn shells, run the `mdex_setup_sh.ini` script.
- For csh or tcsh shells, run the `mdex_setup_csh.ini` script.

For example:

```
source endeca/Latitude/2.1.0/MDEX/mdex_setup_sh.ini
```

This script adds the `utilities` directory and the MDEX Engine binaries to the search path. It is provided as a convenience.

Installing silently on Linux

The silent installer is useful if you want to add the installation of the MDEX Engine to your own install script, or push out the installation on multiple machines.

The silent installer is not interactive.

To install silently on Linux:

1. From a command prompt, navigate to the directory where you downloaded the installer.
2. Issue the following command (on a single line):

   ```
   echo Y | ./latitude_2.1.0_mdex_x86_64pc-linux.sh
      --silent --target /usr/local
   ```

   if you enter `N`, or any string other than `Y`, for the license agreement, the installer issues an error and exits.

   Note: `--target` must be the last parameter specified.

   Optionally, you can replace `/localdisk/username` with the location to which you want to install.

After you install the MDEX Engine, run the script (in the `endeca/Latitude/2.1.0/MDEX` directory) that is appropriate to your shell:

- For Bourne, Bash, or Korn shells, run the `mdex_setup_sh.ini` script.
- For csh or tcsh shells, run the `mdex_setup_csh.ini` script.

For example:

```
source endeca/Latitude/2.1.0/MDEX/mdex_setup_sh.ini
```

This script adds the `utilities` directory and the MDEX Engine binaries to the search path. It is provided as a convenience.

Package contents and directory structure

The MDEX Engine installation creates the following directory structure.

The default root directory for the MDEX Engine is:

- For Linux: `endeca/Latitude/2.1.0/MDEX`
- For Windows: `C:\Endeca\Latitude|2.1.0\MDEX`

The root directory contains files and software modules for all of the MDEX Engine components:

Directory	Contents
root directory	The release notes (`README.txt`) and the `mdex_setup` script that you run after the installation, which adds the `utilities` directory and the MDEX Engine binaries to the PATH environment variable.
`/bin`	Executables for various components, such as Dgraph, along with additional libraries.
`/conf`	Stemming, schema, and DTD files.
`/doc`	The Endeca Licensing Guide and the MDEX Engine API Reference.
`/lib`	Object file libraries.
`/utilities`	Executable files for various utilities, such as GZIP, touch, and grep (Windows only).
`/lib64`	64-bit libraries (Linux only).
`/xquery`	XQuery Web services.

Chapter 5

After You Install

After you install the MDEX Engine, use the following procedures to verify your installation.

Testing your installation with the mkmdex and dgraph commands

To verify the installation, you can create an instance of the MDEX Engine by running the `mkmdex` and `dgraph` commands.

Before running the `mkmdex` command, you must ensure that you added the MDEX binaries and utilities to your path. An error results when you attempt to run `mkmdex` outside of its home directory and have not set the variables. You can either set the path variables manually or set them by running the `mdex_setup.bat` file. For example, if you installed the MDEX Engine in the default location on a Windows machine, at the command prompt, enter:

```
C:\Endeca\Latitude\2.1.0\MDEX\mdex_setup.bat
```

Note that if you use `mdex_setup.bat`, the batch file only sets the variables for the user in the current context.

The `mkmdex` command creates the index database used by the MDEX Engine. The `dgraph` command starts the MDEX Engine.

Note: The instructions in this topic are based on Windows. If you installed on Linux, the steps will be similar, though you will need to substitute executables and paths.

To test that the installation succeeded and create a running instance of the MDEX Engine, perform the following steps:

1. Open a Command Prompt window.
2. At the prompt, enter `mkmdex` followed by a prefix name for the MDEX Engine index. (Be sure to specify an absolute path if the index is to be created in a directory other than the current one).

   The command creates an initial index that the MDEX Engine will use to store its data.

   For example, if you enter:

   ```
   mkmdex C:\testapp
   ```

   The `mkmdex` command appends `_indexes` to the `testapp` directory name (i.e., `testapp_in¬dexes`) and adds subdirectories.

3. To start the MDEX Engine, enter the `dgraph` command and specify the directory name that you provided above.

   For example, enter:

   ```
   dgraph C:\testapp
   ```

   The command loads XQuery and Web services into the MDEX Engine, and starts the MDEX Engine on the default port of 5555.

   The output from running `dgraph` displays as follows:

   ```
   dgraph C:\testapp
   XQuery fn:doc() URL loading is disabled
   Loading XQuery web services...
   XQuery web service created: admin
   XQuery web service created: config
   XQuery web service created: config_read_only
   XQuery web service created: conversation
   XQuery web service created: ingest
   XQuery web service created: mdex
   Finished loading XQuery web services (6958 ms).
   Starting HTTP server on port: 5555
   [Thu Jun 09 09:05:39 2011] Dynamic graph server "dgraph" version
   "7.2.0.537785",

   pid=4448 listening for HTTP connections on port 5555, and bulk ingest
   connections
   on port 5556 at Thu Jun 09 09:05:39 2011
   ```

Note: To point your Latitude Studio application to the MDEX Engine as a data source, see the Latitude Studio installation section of this guide. Ensure that you make note of the Dgraph's port and hostname, and specify them in the Latitude Studio's data source files.

mkmdex command

The `mkmdex` command creates an initial index that the MDEX Engine uses to store its data.

The `mkmdex` command takes the following parameters:

```
mkmdex [--port port] [--verbose] [--help] [-t secs] [db_prefix]
```

db_prefix specifies the prefix to be used for the MDEX Engine database directory. If the database is to be created in a directory other than the current one, be sure to specify the absolute path to the directory.

The flags provide the following functionality:

Flags	Description
`--port` *port*	Specifies the port to use during the provisioning of the MDEX Engine. If you do not use this flag, the default port of 5554 is used.
`-t` *secs*	Specifies the maximum time (in seconds) to wait for the MDEX Engine to be fully started. The MDEX Engine must be started as part of the provisioning operation. The default startup time is 60 seconds.

Flags	Description
	Note that unlike the other flags, the `-t` flag has only one dash.
`--verbose`	Enables verbose mode.
`--help`	Prints usage information and exits.

For example, use this command to create a database (using the prefix `mdexdb`) in the current directory:

```
mkmdex -t 90 mdexdb
```

The command will create a database directory named `mdexdb_indexes` and will provision the MDEX Engine using the default 5554 port. The command will wait up to 90 seconds for the MDEX Engine to start up.

Checking that the Dgraph process is running

Use the `ping` URL command to verify that the Dgraph process is running on a given host name and port number.

To check that the Dgraph process is running, open a browser window and enter the following URL in the Address bar:

```
http://hostname:port/admin?op=ping
```

For example, enter:

```
http://localhost:5555/admin?op=ping
```

The browser window displays a lightweight HTML page that lists the MDEX Engine's host name and port, followed by a timestamp, as in this example:

```
dgraph Web07-WIN:5555 responding at Fri Apr 15 15:26:54 2011
```

Another useful verification command is the following:

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

This command lists the available Web services running on the MDEX Engine, which should be:

- admin
- config
- config_read_only
- conversation
- ingest
- mdex

Loading the correct resolver library

If, when running the Dgraph on a Linux machine, you get the error `Couldn't resolve host host`, your system might be loading the wrong resolver library at run time.

This can happen if `ld.so.cache` contains an entry for a different version than the one you need first.

To load the correct resolver library:

- Set `LD_LIBRARY_PATH` as follows so that it will be searched before `ld.so.cache`.
  - For csh and similar shells:
    ```
    setenv LD_LIBRARY_PATH /lib:${LD_LIBRARY_PATH}
    ```
  - For bash:
    ```
    export LD_LIBRARY_PATH=/lib:${LD_LIBRARY_PATH}
    ```

Part 3

Installing the Latitude Data Integrator

Chapter 6

Installing the Latitude Data Integrator

This chapter describes how to install the Latitude Data Integrator on Linux and Windows platforms.

Installation packages

There are Latitude Data Integrator Designer installation packages for Linux and Windows clients, and one for the Server.

Linux client installation package

The Linux client installation package is named **Data Integrator Designer for Linux**.

The package consists of a tar file named `Latitude_2.1.0_data-integrator-designer_linux.tgz`. The tar file contains a complete version of the LDI Designer utility, including the Latitude connectors.

Windows client installation package

The Windows client installation package consists of two components:

- The **Data Integrator Designer for Windows** component installs the base LDI Designer. The install file is named `Latitude_2.1.0_data-integrator-designer.exe` and must be run first.
- The **Data Integrator Components for Windows** component adds the Latitude connectors to the Designer. The ZIP file is named `Latitude_2.1.0_data-integrator-components.win.zip` and will be overlain on the base Designer install directory.

Server installation package

The `clover.war` file contains the server version of the Latitude Data Integrator, including the Latitude connectors.

Windows client installation

This topic describes how to install Latitude Data Integrator Designer on a Windows client machine.

This procedure assumes that you have downloaded the Latitude Data Integrator packages and have these two files:

- `Latitude_2.1.0_data-integrator-designer.exe`

- `Latitude_2.1.0_data-integrator-components.win.zip`

The procedure also assumes that if you are re-installing, you have deleted all previous LDI workspaces, as well as the `.eclipse` directory that may be created when you first start the LDI utility.

To install Latitude Data Integrator on a Windows client:

1. Right-click on the `Latitude_2.1.0_data-integrator-designer.exe` installer and choose **Run as administrator**.

   The Setup Wizard is displayed.

2. From the Setup Wizard, click **Next**.
3. Click **I Agree** to accept the license agreement.
4. Choose the Install Location.
5. At the Designer Settings screen, make these selections and then click **Next**:
   a) Select **Install a separate Java Development Kit**.
   b) Choose one of the install location shortcuts.
6. Click **I Agree** to accept the Java SDK license agreement.
7. Choose either a Start Menu folder for the shortcut or check **Do not create a new folder in start menu**. Then click **Install**.
8. When the installation completes, first uncheck the **Start CloverETL** box and then click **Finish** to exit the Setup Wizard.

   Important: It is important that you do not start the program before completing the next step. If you do start the Designer at this point, the Endeca components will not be available after step 9. In this case, you must completely uninstall the Designer and re-install it.

9. Unzip `Latitude_2.1.0_data-integrator-components.win.zip` and overlay it into the CloverETL Designer installation directory.

   The directory's pathname is `C:\Program Files (x86)\CloverETL Designer` if you used the default install location.

   Note that if you are having problems overlaying the files from your zip utility, you can unzip the package to a temporary directory and then copy the files using Windows Explorer.

The default installation creates a `C:\Program Files (x86)\CloverETL Designer` directory.

When you run the Latitude Data Integrator Designer, make sure you right-click its start icon and select **Run as administrator**.

Linux client installation

This topic describes how to install Latitude Data Integrator Designer on a Linux client machine.

To install Latitude Data Integrator Designer on a Linux client:

1. Download the `Latitude_2.1.0_data-integrator-designer_linux.tgz` file.
2. Use the `tar` command to extract the file's contents. For example, use this command to extract the contents to the current directory:
   tar -zxvf Latitude_2.1.0_data-integrator-designer_linux.tgz

As a result, a directory named `cloveretl-designer` is created.

Run the `cloveretl-designer` executable file to bring up the Latitude Data Integrator Designer.

Server installation

This topic provides information on installing the Latitude Data Integrator Server.

To install the Latitude Data Integrator Server:

1. Download the `clover.war` file.
2. Use the "Installation" chapter of the *CloverETL Server Reference Manual* for requirements and installation instructions. The manual is available at this URL:

   http://server-demo-ec2.cloveretl.com/clover/docs/installation.html

After installation, refer to the *CloverETL Server Reference Manual* for configuration and usage information.

Part 4

Installing Latitude Studio

Chapter 7

Installing Latitude Studio

This section contains the Latitude Studio installation procedures for the supported application servers.

About the installation process

After downloading the Latitude Studio software, you can install it on your development server.

You have the following options for installing this release of Latitude Studio:

- Latitude Studio with the Windows Tomcat bundle. This is based on Tomcat 6 and Java 1.6.
- Latitude Studio with the Linux Tomcat bundle. This is based on Tomcat 6 and Java 1.6.
- Latitude Studio as a standalone application on Tomcat 5.5 application server.
- Latitude Studio as a standalone application on Websphere Application Server 6.1.
- Latitude Studio as a standalone application on Websphere Application Server 7.

Note: The following steps will deploy the portal using the default embedded Hypersonic database, which is not intended for production use. In production, you must deploy using an alternate database. More information about this process can be found in the section "Using a different database" in the "Other Installation Tasks" chapter. Briefly, to deploy an alternate database, you can modify the `portal-ext.properties` file to specify the appropriate JDBC connection information for the desired database. Alternatively, you can follow the instructions in the *Liferay Portal Administrator's Guide* to set up a JDBC provider and data source in your application server, and then configure `portal-ext.properties` to look up the data source by JNDI name.

Important: To start up, Latitude Studio requires the Endeca Theme. Even if you do not intend to use the Endeca Theme in production, you should not uninstall the Endeca Theme (`endeca-theme-<version>.war`) from the `endeca-portal\deploy` directory.

Installing the Windows Tomcat bundle

This topic provides the steps for installing the Latitude Studio Windows Tomcat bundle on your development server. In this version, Tomcat 6 and the JVM 1.6 are embedded.

Note: Among the data sources in your Latitude Studio application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. Latitude Studio comes with a `default.json` file. For details on configuring this data source to point to your server, see "Updating default.json to point to your server".

To install the Latitude Studio Tomcat bundle:

1. Unzip `Latitude_2.1.0_endeca-portal.zip` to the directory of your choice.

   Latitude Studio creates a directory called `endeca-portal`. For example, if you unzip into `C:`, Latitude Studio installs into `C:\endeca-portal`.

2. Extract the `.war` files from `Latitude_2.1.0_components.zip` and place them into the `endeca-portal\deploy` directory. The `.war` files go in the root of `endeca-portal\deploy`. There should be no subdirectories.

   Note: This directory already contains themes, hooks, and layouts required by the portal. It is safe to overwrite these files with the versions in `Latitude_2.1.0_components.zip`.

3. Install the Latitude Studio license file (`latitude_2.1.0_studio_license.xml`) in the `endeca-portal/deploy` directory.
4. If the environment variables `CATALINA_HOME` or `JAVA_HOME` are already set, update them to point to your newly installed Tomcat directory and a valid 1.6 JRE.

   For example, `set CATALINA_HOME=C:\path\to\endeca-portal\tomcat-6.0.29`. (If you do not have these environment variables set, you can leave them un-set.)

5. Start the portal's Tomcat instance by running `endeca-portal\tomcat-6.0.29\bin\start¬up.bat`.

   Note: Server startup can take several minutes. You can follow the log messages to ascertain when the process is complete. Do not shut down the Tomcat window while Latitude Studio is running.

6. To test that the application is running, go to the portal (`http://localhost:8080/`) in your browser. Log in using the following default credentials:

Login:	`test@endeca.com`
Password:	`test`

7. Optionally, you can set up *log4j* logging. `log4j` provides configurable, Java-based logging in an open-source utility.

   Note: For more information about Latitude Studio logging, see the *Power User's Guide*.

Before you can begin building an application, you need to add your data sources, including a default data source.

Related Links

Getting Started with Latitude Studio on page 75

This section describes how to launch and configure Latitude Studio and begin to work with it.

Other Latitude Studio Installation Tasks on page 79
This section discusses some other installation tasks related to your Latitude Studio installation.

Changing the context root for the Windows Tomcat bundle

Optionally, you can change the context root after installing the Windows Tomcat bundle.

To change the context root:

1. Rename `endeca-portal\tomcat-6.0.29\conf\Catalina\localhost\ROOT.xml` file to `<context root>.xml`.

   For example, if your context root is `sales`, the file name should be `sales.xml`.

   For multi-level context paths, separate the name with #. For example, for a context path of `/sales/east`, the file name should be `sales#east.xml`.

2. Modify the XML file created in the previous step as needed:

   - For a root context: `<Context path="" />`
   - For a context of `/sales`: `<Context path="/sales"/>`
   - For a context of `/sales/east`: `<Context path="/sales/east"/>`

3. Rename the `endeca-portal\tomcat-6.0.29\webapps\ROOT` directory to `endeca-portal\tomcat-6.0.29\webapps\<context root>`.

   For multi-level context paths, use a multi-level path like the following: `endeca-portal\tomcat-6.0.29\webapps\sales#east`.

4. Edit the `endeca-portal\portal-ext.properties` file. Find the `portal.ctx` property at the beginning of `portal-ext.properties`. Change the value of this setting to be the same context root value you used above. However, do not include a trailing slash in the `portal.ctx` value.

   For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

   Do not use this value:

   ```
   portal.ctx=/mycompany/portal/
   ```

Running Latitude Studio as a Windows service

If you have installed the Windows Tomcat bundle, then you can run Latitude Studio as a Windows service.

About running Latitude Studio as a Windows service

Running Latitude Studio as a Windows service requires the Tomcat service installer files.

The Latitude Studio bundle does not include the Tomcat service installer files. You will need to obtain those files from the Tomcat download, which is available from the Apache web site.

After you obtain the files, you then configure and install the service.

You also should install the Tomcat service monitor. The monitor is used to configure and monitor the Windows service, and is useful for troubleshooting. The service monitor executable also is available from the Tomcat download.

Obtaining the service installer files

The service installer and monitor files are part of the Tomcat download.

You must use the files for Tomcat version 6.0.29.

To obtain the files and add them to Latitude Studio:

1. From the Apache Tomcat website (*http://tomcat.apache.org*), download the Tomcat file `apache-tomcat-6.0.29-windows-x86.zip` .

   A sample URL for the archive directory is: *http://archive.apache.org/dist/tomcat/tomcat-6/v6.0.29/bin/*

   If you are not using the bundled JVM, and your JVM is 64-bit, then you must download `apache-tomcat-6.0.29-windows-x64.zip`. This is the 64-bit version of the Tomcat download.

2. Extract the file to a temporary directory.
3. In the `bin` subdirectory of the temporary download directory, locate the following files:
   - `service.bat`
   - `tomcat6.exe`
   - `tomcat6w.exe`. This is the Tomcat service monitor.
4. Copy these files to the `bin` directory of Latitude Studio:

   `endeca-portal\tomcat-6.0.29\bin`

Configuring the service

In the `service.bat` file, you need to configure the service name, description, and memory allocation. The Tomcat monitor file name also must be updated to reflect the change to the service name.

After you copy the Tomcat service installation files, before you can start the service, you need to update `service.bat` to:

- Edit the service name and descriptions to reflect your Latitude Studio installation
- Increase the memory allocation. Latitude Studio requires more memory than is set in the default values.

If you change the service name, then you also must change the name of the Tomcat monitor executable.

To update the configuration:

1. Open the file `service.bat`.
2. In the file, find the following lines:

```
set SERVICE_NAME=Tomcat6
set PR_DISPLAYNAME=Apache Tomcat 6
```

3. Change the name and display name to reflect your Latitude Studio installation. For example:

```
set SERVICE_NAME=LS15
set PR_DISPLAYNAME=Latitude Studio 2.1
```

4. Next, find the following line:

```
set PR_DESCRIPTION=Apache Tomcat 6.0.29 Server -
http://tomcat.apache.org/
```

5. Change the service description to reflect your Latitude Studio installation. For example:

```
set PR_DESCRIPTION=Endeca Latitude Studio server, version 2.1
```

6. Next, find the following line:

```
"%EXECUTABLE%" //US//%SERVICE_NAME% ++JvmOptions "-Djava.io.tmpdir=%CATALI¬
NA_BASE%\temp;-Djava.util.logging.manager=org.apache.juli.ClassLoaderLog¬
Manager;-Djava.util.logging.config.file=%CATALINA_BASE%\conf\logging.prop¬
erties" --JvmMs 128 --JvmMx 256
```

7. Replace the last part of the line:

```
-Djava.util.logging.config.file=%CATALINA_BASE%\conf\logging.properties"
 --JvmMs 128 --JvmMx 256
```

   with:

```
-Djava.util.logging.config.file=%CATALINA_BASE%\conf\logging.properties;-
XX:MaxPermSize=256m" --JvmMs 256 --JvmMx 1024
```

   Make sure that there are no manual line breaks or extra spaces.

8. Save and close the file.
9. Rename the Tomcat monitor file (`tomcat6w.exe`) to be <*value of SERVICE_NAME*> `w.exe`.

   For example, if you set `SERVICE_NAME=DF21` in `service.bat`, then you must rename `tomcat6w.exe` to `DF21w.exe`.

Installing and starting the service

To install the service, you run the `service.bat` file. You also must update the Tomcat monitor to point to the JVM.

Before you install the service, make sure that you have updated the configuration.

Also, if you are not using the bundled JVM, then make sure that the `JAVA_HOME` environment variable is set to the location of your JDK. By default, `service.bat` looks for `%JAVA_HOME%\jre\server\jvm.dll`.

To install and start the Latitude Studio service:

1. From the command line, navigate to the Latitude Studio Tomcat `bin` directory.

```
endeca_portal\tomcat-6.0.29\bin\
```

2. Run the following command:

```
service.bat install
```

3. Configure the Tomcat monitor to point to the JVM:
   a) Double-click the monitor executable.
   b) On the properties dialog, click the **Java** tab.
   c) Uncheck **Use default**.
   d) In the **Java Virtual Machine** field, set the full path to `jvm.dll`.

For the bundled JVM, the file is `endeca-portal\tomcat-6.0.29\jre1.6.0_21\win\bin\server\jvm.dll`.

If you are not using the bundled JVM, then set the path to your JVM.

e) Click **OK.**

4. When you install the service, it is set up to be started manually. To configure the service to start automatically:
   a) Display the **Services** list (**Control Panel** > **Administrative Tools** > **Services**).
   b) In the list, double-click the Latitude Studio service.

      The properties dialog for the service is displayed.

   c) From the **Startup type** drop-down list, select **Automatic**.
   d) Click **OK**.
5. From the **Services** list, to start the service for the first time, right click the service, then click **Start**.

Troubleshooting the service installation

If the service installs properly, but fails to start, you can use the steps provided here to troubleshoot.

For additional details on using the Tomcat service and service monitor, see *http://tomcat.apache.org/tomcat-6.0-doc/windows-service-howto.html*.

As you are troubleshooting, check the log files (`endeca-portal\tomcat-6.0.29\logs\jakarta_service*.log`) for the relevant messages.

If the service will not start:

1. Make sure that you have used the correct version of the Tomcat download:
   - Tomcat version 6.0.29
   - For a 32-bit JVM (including the bundled JVM), `apache-tomcat-6.0.29-windows-x86.zip`
   - For a 64-bit JVM, `apache-tomcat-6.0.29-windows-x64.zip`
2. If you are not using the bundled JVM, make sure that the `JAVA_HOME` environment variable is set to the location of your JDK.

   By default, `service.bat` looks for `%JAVA_HOME%\jre\server\jvm.dll`.

   To change `JAVA_HOME` after the service is installed:
   a) Uninstall the service. To uninstall the service, run the following command:

      ```
      service.bat remove
      ```

   b) Update `JAVA_HOME`.
   c) Reinstall and restart the service.
3. Make sure the Tomcat service monitor is configured to point to the location of your JVM.
   a) Double-click the monitor executable.
   b) On the properties dialog, click the **Java** tab.
   c) Uncheck **Use default**.
   d) In the **Java Virtual Machine** field, specify the path to `jvm.dll`.

      For the bundled JVM, the file is `endeca-portal\tomcat-6.0.29\jre1.6.0_21\win\bin\server\jvm.dll`.

      If you are not using the bundled JVM, then set the path to your JVM.

e) Click **OK.**

4. Use the Tomcat service monitor to set the startup and shutdown modes to Java.
   a) Double-click the monitor executable.
   b) On the properties dialog, click the **Startup** tab.
   c) From the **Mode** drop-down list, select **Java**.
   d) Click the **Shutdown** tab.
   e) From the **Mode** drop-down list, select **Java**.
   f) Click **OK**.

Installing the Linux Tomcat bundle

This topic provides the steps for installing the Latitude Studio Linux Tomcat bundle on your development server. In this version, Tomcat 6 is embedded.

Note: Among the data sources in your Latitude Studio application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. Latitude Studio comes with a `default.json` file. For details on configuring this data source to point to your server, see "Updating default.json to point to your server".

To install the Latitude Studio Tomcat bundle:

1. Extract `Latitude_2.1.0_endeca-portal.tgz` to the directory of your choice.
2. Extract the `.war` files from `Latitude_2.1.0_components.zip` and place them into the `endeca-portal/deploy` directory. The `.war` files go in the root of `endeca-portal/deploy`. There should be no subdirectories.

   Note: This directory already contains themes, hooks, and layouts required by the portal. It is safe to overwrite these files with the versions in `Latitude_2.1.0_components.zip`.

3. Install the Latitude Studio license file (`latitude_2.1.0_studio_license.xml`) in the `endeca-portal/deploy` directory.
4. If the environment variable `CATALINA_HOME` is already set, update it to point to your newly installed Tomcat directory.
5. Make sure that the `JAVA_HOME` environment variable is set to point to a valid 1.6 JRE.
6. To start the portal's Tomcat instance, run `endeca-portal/tomcat-6.0.29/bin/startup.sh`.

   Note: Server startup can take several minutes. You can follow the log messages to ascertain when the process is complete.

7. To test that the application is running, go to the portal (`http://localhost:8080/`) in your browser. Log in using the following default credentials:

Login:	`test@endeca.com`
Password:	`test`

8. Optionally, you can set up *log4j* logging. `log4j` provides configurable, Java-based logging in an open-source utility.

   Note: For more information about Latitude Studio logging, see the *Power User's Guide*.

Before you can begin building an application, you need to add your data sources, including a default data source.

Related Links

Getting Started with Latitude Studio on page 75
This section describes how to launch and configure Latitude Studio and begin to work with it.

Other Latitude Studio Installation Tasks on page 79
This section discusses some other installation tasks related to your Latitude Studio installation.

Changing the context root in the Linux Tomcat bundle

Optionally, you can change the context root used by your Latitude Studio application.

To change the context root:

1. Rename `endeca-portal/tomcat-6.0.16/conf/Catalina/localhost/ROOT.xml` file to `<context root>.xml`.

   For example, if your context root is `sales`, the file name should be `sales.xml`.

   For multi-level context paths, separate the name with #. For example, for a context path of `/sales/east`, the file name should be `sales#east.xml`.

2. Modify the XML file created in the previous step as needed:
   - For a root context: `<Context path="" />`
   - For a context of `/sales`: `<Context path="/sales"/>`
   - For a context of `/sales/east`: `<Context path="/sales/east"/>`

3. Rename the `endeca-portal/tomcat-6.0.29/webapps/ROOT` directory to `endeca-portal/tomcat-6.0.29/webapps/<context root>`.

   For multi-level context paths, use a multi-level path like the following: `endeca-portal/tomcat-6.0.29/webapps/sales#east`.

4. Edit the `endeca-portal/portal-ext.properties` file.

   Find the `portal.ctx` property at the beginning of `portal-ext.properties`.

   Change the value of this setting to be the same context root value you used above. However, do not include a trailing slash in the `portal.ctx` value.

   For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

   Do not use this value:

   ```
   portal.ctx=/mycompany/portal/
   ```

Installing Latitude Studio on Tomcat 5.5

You can deploy Latitude Studio as a standalone application on Tomcat 5.5.

These instructions assume that you have obtained the `apache-tomcat-5.5.x.zip` or `tar.gz` file from the *Apache Foundation* but that you have not yet installed it.

The rest of these instructions will refer to the installation directory as `apache-tomcat-5.5.x`, leaving off the minor version number.

Before following the steps here, consult the *Liferay Portal Administrator's Guide*, which contains portal deployment instructions and examples for Tomcat 5.5.

Note: The examples in this section are based on a Windows server Tomcat deployment. If you are installing on Linux, the steps will be similar, though you will need to substitute Linux binaries and paths. Where there is a significant difference, this is called out.

Note: Among the data sources in your Latitude Studio application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. See "Updating default.json to point to your server". For additional information on creating and configuring data sources, see the *Latitude Studio Power User's Guide*.

High-level overview of Tomcat 5.5 deployment

This topic provides an overview of the steps you need to take to deploy Latitude Studio as a standalone application on Tomcat 5.5.

Details on each of these steps appear in the topics that follow.

To deploy Latitude Studio on Tomcat 5.5:

1. Install Tomcat and deploy the Latitude Studio dependency libraries.
2. Modify Tomcat configuration to work with Latitude Studio.
3. Deploy and start the Latitude Studio application.

Installing Tomcat 5.5 and deploying the dependency libraries

Latitude Studio requires the deployment of several Java libraries.

To install the Tomcat software and deploy the Latitude Studio dependency libraries:

1. Create an `endeca-portal` directory. This will be the home directory for your Latitude Studio installation.
2. Create an `apache-tomcat-<version>` directory under the `endeca-portal` directory.
3. Unzip `apache-tomcat-5.5.x.zip` into `endeca-portal/apache-tomcat-5.5.x`, where `x` indicates the minor version number.

   Unzipping this file creates much of the directory structure mentioned below.

4. Unzip `Latitude_2.1.0_endeca-portal-dependencies.zip` into a temporary directory.

   This zip file contains a collection of `.jar` files and other dependency files.

5. From the temporary directory, copy the following `.jar` files into the `endeca-portal/apache-tomcat-5.5.x/common/endorsed` directory:

```
ccpp.jar
jutf7.jar
log4j.jar
log4j.properties.jar
```

6. Under the `endeca-portal/apache-tomcat-5.5.x/common/lib` directory, create an `ext` directory.
7. From the temporary directory you created in step 4, copy the following `.jar` files into the `endeca-portal/apache-tomcat-5.5.x/common/lib/ext` directory that you just created:

```
activation.jar
annotations.jar
commons-lang.jar
config_bindings_cxf.jar
config_ro_bindings_cxf.jar
cs_bindings.jar
cxf-2.2.8.jar
cxf-rt-databinding-jaxb-2.2.8.jar
endeca-images.jar
endeca_navigation.jar
endeca-portal.jar
ext-service.jar
geronimo-activation_1.1_spec-1.0.2.jar
geronimo-annotation_1.0_spec-1.1.1.jar
geronimo-jaxws_2.1_spec-1.0.jar
geronimo-saaj_1.3_spec-1.0.1.jar
geronimo-stax-api_1.0_spec-1.0.1.jar
geronimo-ws-metadata_2.0_spec-1.1.2.jar
hsql.jar
jabsorb.jar
jackson-core-lgpl-1.7.2.jar
jackson-mapper-lgpl-1.7.2.jar
jaxb-api-2.1.jar
jaxb-impl-2.1.7.jar
jms.jar
jsr173_1.0_api.jar
jta.jar
jtds.jar
mail.jar
mysql.jar
portal-kernel.jar
portal-service.jar
portlet.jar
postgresql.jar
stax-1.2.0.jar
wsdl4j-1.6.2.jar
wstx.jar
XmlSchema-1.4.3.jar
```

Modifying Tomcat configuration to work with Latitude Studio

Before proceeding further, you must modify some Tomcat configuration files.

1. In the `endeca-portal/apache-tomcat-5.5.x/bin/` directory, modify `catalina.bat` (on Windows) or `catalina.sh` (on Linux) by adding the JAVA_OPTS line.

This line should appear under the line `Execute The Requested Command` as follows:

```
set JAVA_OPTS=%JAVA_OPTS% -Xmx1024m -XX:MaxPermSize=256m -Dfile.encod¬
ing=UTF8 -Duser.timezone=GMT -Dorg.apache.catalina.loader.WebappClassLoad¬
er.ENABLE_CLEAR_REFERENCES=false
```

This increases the memory size for the server and establishes security configuration for Latitude Studio.

2. Modify the `endeca-portal/apache-tomcat-5.5.x/conf/catalina.properties` file as follows to add the `ext` directory to the common class loader:

```
common.loader=
     ${catalina.home}/common/classes,\
     ...\
     ${catalina.home}/common/lib/ext/*.jar
```

3. To deploy Latitude Studio in the root context, create a new file called `ROOT.xml` and place it in `endeca-portal/apache-tomcat-5.5.x/conf/Catalina/localhost/`.

   To deploy Latitude Studio into any other context, create a new file called `<context root>.xml` and place it in `endeca-portal/apache-tomcat-5.5.x/conf/Catalina/localhost/`.

   For multi-level context paths, separate the name with #. For example, for a context path of `/sales/east`, the file name should be `sales#east.xml`.

4. Modify the XML file created in the previous step as needed:

   - For a root context: `<Context path="" />`
   - For a context of `/sales`: `<Context path="/sales"/>`
   - For a context of `/sales/east`: `<Context path="/sales/east"/>`

5. To support UTF-8 URI encoding, edit the `server.xml` file located in the `endeca-portal/apache-tomcat-5.5.x/conf` directory as follows:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
   <Connector
      port="8080"
      maxHttpHeaderSize="8192"
      maxThreads="150"
      minSpareThreads="25"
      maxSpareThreads="75"
      enableLookups="false"
      redirectPort="8443"
      acceptCount="100"
      connectionTimeout="20000"
      disableUploadTimeout="true"
      URIEncoding="UTF-8"
   />
```

Deploying and starting the Latitude Studio application

Once Tomcat configuration is complete, the Latitude Studio application can be deployed and started.

To deploy and start the Latitude Studio application:

1. Delete the contents of the `endeca-portal/apache-tomcat-5.5.x/webapps/ROOT` directory.

This directory contains the standard Web application that is installed with Tomcat by default. We will replace this standard web application with the Latitude Studio application in the next step.

2. Unzip `Latitude_2.1.0_endeca-portal-war.zip` into a temporary directory.

   This zip file contains the Latitude Studio .war file and the `copyright.txt` file.

3. Read the `copyright.txt` file and then save it to the location of your choice.
4. Unzip the contents of the .war file into the `endeca-portal/apache-tomcat-5.5.x/webapps/ROOT` directory.
5. If using a non-root context for your deployment, rename the `endeca-portal\apache-tomcat-5.5.x\webapps\ROOT` directory to `endeca-portal\apache-tomcat-5.5.x\webapps\<context root>`.

   For multi-level context paths, use the multi-level path. For example: `endeca-portal\apache-tomcat-5.5.x\webapps\ROOT` directory to `endeca-portal\apache-tomcat-5.5.x\webapps\mycompany\sales`.

6. Copy the `portal-ext.properties` file from the temporary directory you created for the `Latitude_2.1.0_endeca-portal-dependencies.zip` file in step 4 to the `endeca-portal` directory.
7. Edit the `endeca-portal\portal-ext.properties` file.

   Find the `portal.ctx` property at the beginning of `portal-ext.properties`.

   Change the value of this setting to be the same context root value you set earlier. However, do not include a trailing slash in the `portal.ctx` value.

   For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

   Do not use this value:

   ```
   portal.ctx=/mycompany/portal/
   ```

8. Under the `endeca-portal` directory, create a `data` directory, and then create an `endeca-data-sources` directory below that.
9. Create a data source to place in the `endeca-portal/data/endeca-data-sources` directory.

   For information about data sources, see the *Latitude Studio Power User's Guide*.

   In addition, you can reference the sample data source files, which are located in the `endeca-data-sources` directory in the temporary directory you created for the `endeca-portal-dependencies-<version>.zip` file in a previous step.

10. In the `endeca-portal` directory, create a `deploy` directory.
11. Extract the `.war` files from `Latitude_2.1.0_components.zip` and place them into the `endeca-portal\deploy` directory. The `.war` files go in the root of `endeca-portal\deploy`. There should be no subdirectories.
12. Install the Latitude Studio license file (`latitude_2.1.0_studio_license.xml`) in the `endeca-portal/deploy` directory.
13. To start the portal's Tomcat instance, run `endeca-portal\tomcat-5.5.x\bin\startup.bat`.

Note: Server startup can take several minutes. You can follow the log messages to ascertain when the process is complete. Do not shut down the Tomcat window while Latitude Studio is running.

14. To test that the application is running, go to the portal (`http://localhost:8080/`) in your browser. Log in using the following default credentials:

Login:	`test@endeca.com`
Password:	`test`

Before you can begin building an application, you need to add your data sources, including a default data source.

Related Links

Getting Started with Latitude Studio on page 75
This section describes how to launch and configure Latitude Studio and begin to work with it.

Other Latitude Studio Installation Tasks on page 79
This section discusses some other installation tasks related to your Latitude Studio installation.

Installing Latitude Studio on the WebSphere Application Server version 6.1

You can deploy Latitude Studio as a standalone application on WebSphere Application Server (WAS) version 6.1. (6.1.0.1 or higher).

Before following the steps here, consult the *Liferay Portal Administrator's Guide*, which contains portal deployment instructions and examples for WebSphere Application Server 6.1.

Note: The examples in this section are based on a Linux server WAS deployment. If you are installing on Windows, the steps will be similar, though you will need to substitute Windows executables and paths. In certain examples, backslashes are used to escape the dollar sign ($) character on Linux, because the shell would otherwise attempt a variable substitution for this character. These backslashes should not be required on a Windows system.

Note: Among the data sources in your Latitude Studio application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. For details, see the topic "Updating default.json to point to your server" in the next chapter.

Note: When Latitude Studio is running on WAS 6.1 on Linux, some of its administrative components require the `compat-libstdc++-33` library to function. If this library is not already installed, you can install it with a command such as `yum install compat-libstdc++-33.` For more information, see the *IBM WebSphere documentation*.

High-level overview of WebSphere Application Server 6.1 deployment

This topic provides an overview of the steps you need to take to deploy Latitude Studio on WAS 6.1.

Details on each of these steps appear in the topics that follow.

To deploy Latitude Studio on WAS 6.1:

1. Deploy dependency `.jar` files.

   The exact list of required files and destination directories appears below.

2. Start (or restart) the WAS server.
3. Install the Latitude Studio `.war` file as an enterprise application.
4. Edit and deploy `portal-ext.properties`.
5. Create the `endeca-data-sources/*.json` data source configuration files.

   For more information, see the *Latitude Studio Power User's Guide*.

6. Install the Endeca theme, portlet components, and other framework `.war` files.
7. Install the Liferay license.

   The instructions for obtaining and installing the license are provided later in this section.

8. Start the Latitude Studio enterprise application.
9. Optionally, repeat step 6 for any additional plugins you want to add.

Deploying Latitude Studio dependency libraries on WAS 6.1

Latitude Studio requires the deployment of several Java libraries.

These libraries are deployed to a global class loader, making them available to multiple applications.

To deploy the Latitude Studio dependency libraries on WAS 6.1:

1. Unzip the `.jar` files found in `Latitude_2.1.0_endeca-portal-dependencies.zip`.
2. Upload the following `.jar` files from the `.zip` file to the WAS server's external library directory. (For example, if WAS is installed in `/usr/local/WAS/AppServer`, you would deploy the selected `.jar` files into `/usr/local/WAS/AppServer/lib/ext/`.)

```
annotations.jar
ccpp.jar
commons-lang.jar
config_bindings_cxf.jar
config_ro_bindings_cxf.jar
cs_bindings.jar
cxf-rt-databinding-jaxb-2.2.8.jar
endeca-images.jar
endeca_navigation.jar
endeca-portal.jar
ext-service.jar
hsql.jar
jabsorb.jar
jackson-core-lgpl-1.7.2.jar
jackson-mapper-lgpl-1.7.2.jar
jsr173_1.0_api.jar
log4j.jar
portal-kernel.jar
portal-service.jar
slf4j-api.jar
slf4j-log4j12.jar
stax-1.2.0.jar
wstx.jar
```

3. Move the following list of `.jar` files from the `.zip` file to `/usr/local/WAS/AppServer/java/jre/lib/ext/`:

```
cxf-2.2.8.jar
geronimo-activation_1.1_spec-1.0.2.jar
geronimo-annotation_1.0_spec-1.1.1.jar
geronimo-jaxws_2.1_spec-1.0.jar
geronimo-saaj_1.3_spec-1.0.1.jar
geronimo-stax-api_1.0_spec-1.0.1.jar
geronimo-ws-metadata_2.0_spec-1.1.2.jar
jaxb-api-2.1.jar
jaxb-impl-2.1.7.jar
portlet.jar
wsdl4j-1.6.2.jar
XmlSchema-1.4.3.jar
```

4. Restart the WAS server so that it can pick up the newly available `.jar` files.

Extracting the standalone portal WAR on WAS 6.1

Before you can install the standalone portal WAR, you must extract it from its download package.

To extract the standalone portal WAR on WAS 6.1:

1. Unzip `Latitude_2.1.0_endeca-portal-war.zip` into a temporary directory.

   This zip file contains`endeca-portal-<version>.war` and the `copyright.txt` file.

2. Read the `copyright.txt` file, and then save it to the location of your choice.
3. Note the location of `endeca-portal-<version>.war`, as you will need to navigate to it in the next step.

Deploying the standalone portal WAR on WAS 6.1

After downloading and extracting the necessary files, you can deploy Latitude Studio as an enterprise application in WebSphere Application Server, and then install components, themes, and other plugins as modules in that enterprise application.

The following steps document the installation procedure by using the IBM Integrated Solutions Console for a WebSphere Application Server installed and maintained without the use of the Deployment Manager, and consisting of one cell with one node and one server.

The instructions may need to be adjusted for

- Clustered environments
- Environments maintained with the Deployment Manager
- Environments where administration is performed by using tools like `wsadmin`, rather than the Integrated Solutions Console

The following steps assume that no other applications are deployed in the same application server. If there are other applications, ensure that no applications are bound to context root `/` (or that any such applications are stopped during the Latitude Studio deployment). After following these steps, you will be able to adjust the context root for the Latitude Studio application, to ensure it does not conflict with other applications.

To deploy the Latitude Studio standalone portal WAR on WebSphere Application Server version 6.1:

1. Start the WAS server.
2. Log in to the WAS Integrated Solutions Console, using the appropriate administrator credentials.
3. In the WAS Integrated Solutions Console, select **Applications** > **Install New Application**.
4. Click to browse to and select the Endeca Latitude Studio WAR you downloaded earlier.
5. Set the context root to the desired path for your Latitude Studio installation.

   Note: Make a note of your context root, as you will need to reference it several times during the deployment process.

6. Select **Show me all installation options and parameters**, and then click **Next**.
7. In the **Choose to generate default bindings and mappings** section, check **Generate Default Bindings**, and then select the following options:

   - **Override existing bindings**
   - **Use default virtual host name for Web and WIP modules** (enter `default_host` in the text field)

   Click **Next**.
8. In the **Select installation options** step, the default application name is **endeca-portal-<*version*>_war**. Set the application name to a more relevant name (for example, `LatitudeStudio`). All other installation options can remain unchanged. Click **Next**.

   Note: Do not use spaces in the application name. For example, use **LatitudeStudio** instead of **Latitude Studio**.

9. Keep the defaults for all other options and then click **Finish**.
10. Wait for installation and, if it is successful, click **Save directly to master configuration**.

Creating the Liferay Home directory for WAS 6.1

The remaining instructions in this section refer to a directory called Liferay Home. The Liferay Home directory is created relative to the user's home directory.

Manually create a Liferay Home directory (`/home/endeca/liferay/`), along with the following subdirectories:

- `/home/endeca/liferay/data`
- `/home/endeca/liferay/data/endeca-data-sources`
- `/home/endeca/liferay/websphere-deploy`

Editing the portal-ext.properties file for WAS 6.1 deployment

Before deploying your `portal-ext.properties` file, you must edit it.

Endeca's default version of `portal-ext.properties` is included in the package `Latitude_2.1.0_endeca-portal-dependencies.zip`.

1. Open the `portal-ext.properties` file.
2. Add the following lines to the end of the file:

```
# Specify a directory where Liferay will "deploy" processed plugins.
# From this directory, WAS users will deploy WARs as modules in the
```

```
# Latitude Studio enterprise application. Replace ${liferay.home}
# with the appropriate directory path, such as /home/endeca.
#
auto.deploy.dest.dir=${liferay.home}/websphere-deploy
#
# Set this to true to enable JMX integration in
# com.liferay.portal.cache.EhcachePortalCacheManager.
#
ehcache.portal.cache.manager.jmx.enabled=false
```

Note: The destination directory (specified by the `auto.deploy.dest.dir` setting) must exist before the plugin is hot-deployed. In the above example, you must manually create the `websphere-deploy` directory if it does not exist.

3. Find the `portal.ctx` property at the beginning of `portal-ext.properties`. Change the value of this setting to be the same context root value you used when deploying the standalone portal WAR.

   However, do not include a trailing slash in the `portal.ctx` value.

   For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

   Do not use this value:

   ```
   portal.ctx=/mycompany/portal/
   ```

4. Save the file.

Configuring portal-ext.properties for WAS 6.1 deployment

After you edit your `portal-ext.properties` file, you must deploy it.

To deploy the file in WAS 6.1, you can either:

- Update the application to include the `portal-ext.properties` file.
- Upload the `portal-ext.properties` file to the Liferay Home directory on the server.

Updating the application to include the portal-ext.properties file on WAS 6.1

After you edit the `portal-ext.properties` file, you can use the IBM Integrated Solutions Console to include `portal-ext.properties` in the `endeca-portal.war` module.

These steps may be performed with the `wsadmin` tool instead of the Integrated Solutions Console, and may need to be adjusted for alternate WAS configurations.

Note: In order to make changes to the `portal-ext.properties` file, users will need to repeat these steps to update the application with updated versions of the `portal-ext.properties` file. In some environments, it may be more appropriate to deploy the `portal-ext.properties` file to the Liferay Home directory, where it can be updated without updating the deployed application. That option is described in another topic.

To deploy a `portal-ext.properties` file in the Integrated Solutions Console:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications** and .
2. Select the enterprise application created when you deployed the portal WAR, then click **Update**.

3. Select **Replace or add a single file**.
4. Specify the path to deploy the file into the `WEB-INF/classes` directory of the portal Web application.

   For example:

   ```
   endeca-portal-<version>.war/WEB-INF/classes/portal-ext.properties
   ```

5. Browse to where you created the file on your computer.
6. When the file has successfully updated, click **Save directly to master configuration**.

Uploading portal-ext.properties to Liferay Home on the server on WAS 6.1

After you create the `portal-ext.properties` file, you can manually upload it to WAS 6.1.

To manually upload the `portal-ext.properties` file:

1. Upload the `portal-ext.properties` file to the Liferay Home directory.

   For example: `/home/endeca/liferay/portal-ext.properties`

2. When the Latitude Studio application is started, Liferay reads these properties.

Example settings for portal-ext.properties on WAS 6.1

The Endeca installation includes a default version of `portal-ext.properties`.

This file serves as a useful starting point for configuration of the portal properties, and should be deployed to the application server according to the steps described in a previous topic.

Note: Most of the settings in the default `portal-ext.properties` file are not specific to deployment on WAS 6.1. However, the following additional settings (which you must add to the file as described in the topic "Configuring portal-ext.properties for WAS 6.1 deployment") are important for portlet deployment on WAS:

```
auto.deploy.dest.dir=/home/endeca/liferay/websphere-deploy
ehcache.portal.cache.manager.jmx.enabled=false
```

Keep in mind that the destination directory (specified by the `auto.deploy.dest.dir` setting) must exist before the plugin is hot-deployed. In the above example, you must manually create the `websphere-deploy` directory if it does not exist.

Deploying Endeca data source configuration on WAS 6.1

To configure one or more MDEX Engines as data sources for Latitude Studio, you must deploy a JSON configuration file for each MDEX Engine.

These files should be deployed relative to the Liferay Home directory.

Sample data source configuration files are provided as `.json.sample` files in the `Latitude_2.1.0_endeca-portal-dependencies.zip` file you downloaded.

To deploy the Endeca data source configuration, upload the data source files to the `data/endeca-data-sources/` subdirectory.

For example, `/home/endeca/liferay/data/endeca-data-sources/default.json`.

Starting the application on WAS 6.1

Once the Latitude Studio application has been deployed, and the `portal-ext.properties` file has been configured and deployed, the application needs to be started.

The following steps describe this process in the IBM Integrated Solutions Console.

To start the application:

1. Go to **Applications** > **Enterprise Application** .
2. Select the enterprise application created when you deployed the portal WAR.
3. If the application is not already running, click **Start** to start it.
4. View your deployed application at the root context of the server.
5. Restart WAS 6.1.

Deploying components and other plugins in WAS 6.1

This topic describes how to deploy components, themes, hooks, and other plugins in WAS 6.1.

These plugins are located in the `Latitude_2.1.0_components.zip` package.

About Liferay component pre-processing on WAS 6.1

WAS does not support the hot deployment of components. However, Liferay's deployment code must update plugins by adding necessary libraries and configuration files.

For example, Liferay's portlet deployment code adds the following important piece of configuration to a portlet component's `web.xml` file:

```
<context-param>
  <param-name>com.ibm.websphere.portletcontainer.PortletDeploymentEn¬
abled</param-name>
  <param-value>false</param-value>
</context-param>
```

This context parameter is important for WAS deployment, as it ensures that WAS's portal server does not attempt to load the new portlet, and instead allows Latitude Studio to load the newly deployed portlet.

For this reason, Liferay must be allowed to pre-process components before they are deployed to WAS. You upload your `.war` files to Liferay's `deploy` directory so that Liferay's deployer can find and process them.

Deploying components in WAS 6.1

Liferay must pre-process Latitude Studio components before they can be deployed in WAS 6.1.

Important: To start up, Latitude Studio requires the Endeca Theme. Even if you do not intend to use the Endeca Theme in production, you should deploy the Endeca Theme (`endeca-theme-<version>.war`).

To deploy Latitude Studio components in WAS 6.1:

1. Copy all component `.war` files to `${liferay.home}/deploy`.

2. Wait while Liferay pre-processes the .war files and places them in the `${liferay.home}/websphere-deploy` directory.
3. Deploy the `.war` files generated in step 2 as modules in the Latitude Studio enterprise application. To do this, you can use either:
   - The WebSphere Integrated Solutions Console.
   - The command line, using `wsadmin`.

Deploying generated .war files on WAS 6.1 with the Integrated Solutions Console

You can use the IBM Integrated Solutions Console to deploy the `.war` files it finds in the `websphere-deploy` directory.

Important: The procedure in this topic does not work in versions of WAS prior to 6.1.0.9, due to a known problem in WebSphere Application Server (for details, see *http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK36365*). If you are using a service pack prior to 6.1.0.9, use `wsadmin` to deploy the generated `.war` file.

Note: These steps may need to be adjusted for alternate WAS configurations.

To manually deploy a generated `.war` file:

1. Go to **Applications** > **Enterprise Applications**.
2. Select the enterprise application created when you deployed the portal `.war` file, then click **Update**
3. Select **Replace or add a single module**.
4. Specify the path to deploy the file as the display name of the new module.

   For example, if you are adding `endeca-navigation-portlet.war`, specify the path as `endeca-navigation-portlet`.

5. Browse the remote file system to the newly created `.war` file in the Liferay deploy output directory.

   Continuing the example above, this might be `/home/endeca/liferay/websphere-deploy/endeca-navigation-portlet.war`.

6. After selecting the directory, click **Ok**.
7. Select the detailed install path. Keep the defaults on all screens except the context root. Set the context root to match the display name of the new plugin (in this example, `/endeca-navigation-portlet/`).
8. Once it has successfully updated, click **Save directly to master configuration**.

Using wsadmin to deploy the generated .war file on WAS 6.1

You can also use the `wsadmin` tool to deploy the generated `.war` file from the command line .

Note: These steps may need to be adjusted for alternate WAS configurations.

In the `wsadmin` tool, enter a command similar to the example below.

In this example:

- The enterprise application is named `LatitudeStudio`.
- The file name for the module being added is `endeca-navigation-portlet.war`.

- The module display name is `endeca-navigation-portlet`.

This command is executed from the Liferay deploy output directory (that is, the directory containing the `endeca-navigation-portlet.war` file). In our example, this command is executed in `/home/endeca/liferay/websphere-deploy/` .

```
[WAS]/AppServer/bin/wsadmin.sh -c "\$AdminApp update LatitudeStudio
modulefile {-operation addupdate -contents endeca-navigation-port¬
let.war -contextroot /endeca-navigation-portlet/ -contenturi endeca-navi¬
gation-portlet -usedefaultbindings}" -c "\$AdminConfig save"
```

Installing the Latitude Studio license

Before you can start Latitude Studio, you must install the license, which is available from the Endeca Developer Network (EDeN).

To install the license:

1. Download the Latitude Studio license (`latitude_2.1.0_studio_license.xml`) from the Latitude section of EDeN (*http://eden.endeca.com*).

   For details on navigating to this section, see the section on downloading the Latitude software.

2. Save the file to the `${liferay.home}/deploy` directory of your Latitude Studio installation.

When you start Latitude Studio, the license is installed.

Troubleshooting WAS 6.1 deployment

Keep the following issue in mind when deploying Latitude Studio on WAS 6.1.

Updating the Latitude Studio .war file

If you need to update the Latitude Studio `.war` file (not any individual plugin, but the portal `.war` itself), you must restart the WAS server. If you only restart the module, the restart might not be successful.

Installing Latitude Studio on the WebSphere Application Server version 7

You can deploy Latitude Studio as a standalone application on WebSphere Application Server (WAS) version 7.0.

Before following the steps here, consult the *Liferay Portal Administrator's Guide*, which contains portal deployment instructions and examples for WebSphere Application Server 7.0.

Note: The examples in this section are based on a Linux server WAS deployment. If you are installing on Windows, the steps will be similar, though you will need to substitute Windows executables and paths. In certain examples, backslashes are used to escape the dollar sign ($) character on Linux, because the shell would otherwise attempt a variable substitution for this character. These backslashes should not be required on a Windows system.

Note: Among the data sources in your Latitude Studio application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. For details, see the topic "Updating default.json to point to your server" in the next chapter.

High-level overview of WebSphere Application Server 7 deployment

This topic provides an overview of the steps you need to take to deploy Latitude Studio on WAS 7.

Details on each of these steps appear in later topics.

To deploy Latitude Studio on WAS:

1. Deploy dependency `.jar` files.

   The exact list of required files and destination directories is provided below.

2. Start (or restart) the WAS server.
3. Install the Latitude Studio `.war` file as an enterprise application.
4. Edit and deploy `portal-ext.properties`.
5. Create the `endeca-data-sources/*.json` data source configuration files.
   For more information, see the *Power Users Guide*.
6. Install the Endeca theme, portlet components, and other framework `.war` files.
7. Install the Liferay license.

   The instructions for obtaining and installing the license are provided later in this section.

8. Start the Latitude Studio enterprise application.
9. Optionally, repeat step 6 for any additional plugins you want to add.

Deploying Latitude Studio dependency libraries on WAS 7

Latitude Studio requires the deployment of several Java libraries.

These libraries are deployed to a global class loader, making them available to multiple applications.

To deploy the Latitude Studio dependency libraries:

1. Unzip the `.jar` files found in `Latitude_2.1.0_endeca-portal-dependencies.zip`.
2. Upload the following `.jar` files from the `.zip` file to the WAS server's external library directory.

   (For example, if WAS is installed in `/usr/local/WAS/AppServer`, you would deploy the selected `.jar` files into `/usr/local/WAS/AppServer/lib/ext/`.)

```
annotations.jar
commons-lang.jar
config_bindings_cxf.jar
config_ro_bindings_cxf.jar
cs_bindings.jar
cxf-2.2.8.jar
cxf-rt-databinding-jaxb-2.2.8.jar
endeca-images.jar
endeca_navigation.jar
endeca-portal.jar
```

```
ext-service.jar
geronimo-activation_1.1_spec-1.0.2.jar
geronimo-annotation_1.0_spec-1.1.1.jar
geronimo-jaxws_2.1_spec-1.0.jar
geronimo-saaj_1.3_spec-1.0.1.jar
geronimo-stax-api_1.0_spec-1.0.1.jar
geronimo-ws-metadata_2.0_spec-1.1.2.jar
hsql.jar
jabsorb.jar
jackson-core-lgpl-1.7.2.jar
jackson-mapper-lgpl-1.7.2.jar
jaxb-api-2.1.jar
jaxb-impl-2.1.7.jar
jsr173_1.0_api.jar
log4j.jar
portal-kernel.jar
portal-service.jar
portlet.jar
slf4j-api.jar
slf4j-log4j12.jar
stax-1.2.0.jar
wsdl4j-1.6.2.jar
wstx.jar
XmlSchema-1.4.3.jar
```

3. Restart the WAS server so that it can pick up the newly available `.jar` files.

Extracting the standalone portal WAR on WAS 7

Before you can install the standalone portal WAR, you must extract it from its download package.

To extract the standalone portal WAR on WAS 7:

1. Unzip `Latitude_2.1.0_endeca-portal-war.zip` into a temporary directory.

   This zip file contains the `endeca-portal-<version>.war` file and the `copyright.txt` file.

2. Read the `copyright.txt` file, then save it to the location of your choice.
3. Note the location of the `endeca-portal-<version>.war` file, as you will need to navigate to it in the next step.

Deploying the standalone portal WAR on WAS 7

After downloading and extracting the necessary files, you can deploy Latitude Studio as an enterprise application in WebSphere Application Server, and then install portlets, themes, and other plugins as modules in that enterprise application.

The following steps document the installation procedure by using the IBM Integrated Solutions Console for a WebSphere Application Server installed and maintained without the use of the Deployment Manager, and consisting of one cell with one node and one server.

The instructions may need to be adjusted for clustered environments, environments maintained with the Deployment Manager, or for environments where administration is performed by using tools like `wsadmin`, rather than the Integrated Solutions Console.

The following steps assume that no other applications are deployed in the same application server. If there are other applications, ensure that no applications are bound to context root `/` (or that any such applications are stopped during the Latitude Studio deployment).

After following these steps, you will be able to adjust the context root for the Latitude Studio application, to ensure it does not conflict with other applications.

To deploy the Latitude Studio standalone portal WAR on WAS 7:

1. Start the WAS server.
2. Log in to the WAS Integrated Solutions Console, using the appropriate administrator credentials.
3. In the WAS Integrated Solutions Console, select **Applications** > **New Application** > **New Enterprise Application**.
4. Click to browse to and select the Endeca Latitude Studio WAR you downloaded earlier (`endeca-portal-<version>.war`), and then click **Next**.
5. Select **Choose to generate default bindings and mappings**. Check the following options:
   - **Generate default bindings**
   - **Override existing bindings**
6. Still in the **Choose to generate default bindings and mappings** section, check **Use default virtual host name for Web and SIP modules**, and then enter `default_host` in the text field.

   Click **Next**.
7. By default, the application name is **endeca-portal-<*version*>_war**. Set the application name to a more relevant name (for example, `LatitudeStudio`). All other installation options can remain unchanged. Click **Next**.

   Note: Do not use spaces in the application name. For example, use **LatitudeStudio** instead of **Latitude Studio**.
8. In **Map modules to servers**, accept the default settings, and then click **Next**.
9. In **Map context roots for Web modules**, set the context root to the desired path for your Latitude Studio installation, and then click **Next**.

   Note: Make a note of your context root, as you will need to reference it several times during the deployment process.
10. In **Install New Application**, confirm that your settings are correct, and then click **Finish**.
11. Wait for installation. If the installation is successful, click **Save directly to master configuration**.

Creating the Liferay Home directory on WAS 7

The remaining instructions in this section refer to a directory called Liferay Home. The Liferay Home directory is created relative to the user's home directory.

Manually create a Liferay Home directory (`/home/endeca/liferay/`), along with the following subdirectories:

- `/home/endeca/liferay/data`
- `/home/endeca/liferay/data/endeca-data-sources`
- `/home/endeca/liferay/websphere-deploy`

Editing the portal-ext.properties file for WAS 7 deployment

Before deploying your `portal-ext.properties` file, you must edit it.

Endeca's default version of `portal-ext.properties` is included in the package `Latitude_2.1.0_endeca-portal-dependencies.zip`.

1. Open the `portal-ext.properties` file
2. Add the following lines to the end of the file:

```
# Specify a directory where Liferay will "deploy" processed plugins.
# From this directory, WAS users will deploy WARs as modules in the
# Latitude Studio enterprise application.
#
auto.deploy.dest.dir=${liferay.home}/websphere-deploy
#
# Set this to true to enable JMX integration in
# com.liferay.portal.cache.EhcachePortalCacheManager.
#
ehcache.portal.cache.manager.jmx.enabled=false
```

Note: The destination directory (specified by the `auto.deploy.dest.dir` setting) must exist before the plugin is hot-deployed. In the above example, you must manually create the `websphere-deploy` directory if it does not exist.

3. Find the `portal.ctx` property at the beginning of `portal-ext.properties`.

   Change the value of this setting to be the same context root value you used when deploying the standalone portal WAR. However, do not include a trailing slash in the `portal.ctx` value.

   For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

   Do not use this value:

   ```
   portal.ctx=/mycompany/portal/
   ```

4. Save the file.

Configuring portal-ext.properties for WAS 7 deployment

After you edit your `portal-ext.properties` file, you must deploy it in WAS.

To deploy the file, you can either:

- Update the application to include the `portal-ext.properties` file.
- Upload the `portal-ext.properties` file to the Liferay Home directory on the server.

Updating the application to include the portal-ext.properties file on WAS 7

After you create the `portal-ext.properties` file, you can use the IBM Integrated Solutions Console to include `portal-ext.properties` in the `endeca-portal.war` module.

These steps may be performed with the `wsadmin` tool instead of the Integrated Solutions Console, and may need to be adjusted for alternate WAS configurations.

Note: In order to make changes to the `portal-ext.properties` file, users will need to repeat these steps to update the application with updated versions of the `portal-ext.properties` file. In some environments, it may be more appropriate to deploy the `portal-ext.properties` file to the Liferay Home directory, where it can be updated without updating the deployed application. That option is described in another topic.

To deploy a `portal-ext.properties` file in the Integrated Solutions Console:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications**.
2. Select the enterprise application created when you deployed the portal WAR, then click **Update**.
3. Select **Replace or add a single file**.
4. Specify the path to deploy the file into the `WEB-INF/classes` directory of the portal Web application.

   For example: `endeca-portal-<version>.war/WEB-INF/classes/portal-ext.properties`

5. Browse to where you created the file on your computer.
6. When the file has successfully updated, click **Save directly to master configuration**.

Uploading portal-ext.properties to Liferay Home on the server on WAS 7

After you create the `portal-ext.properties` file, you can manually upload it to WAS.

To manually upload the `portal-ext.properties` file:

1. Upload the `portal-ext.properties` file to the Liferay Home directory. For example: `/home/endeca/liferay/portal-ext.properties`.
2. When the Latitude Studio application is started, Liferay reads these properties.

Example settings for portal-ext.properties on WAS 7

Endeca's default version of `portal-ext.properties` is included in the package `Latitude_2.1.0_endeca-portal-dependencies.zip`.

This file serves as a useful starting point for configuration of the portal properties, and should be deployed to the application server according to the steps described in a previous topic.

Note: Most of the settings in the default `portal-ext.properties` file are not specific to deployment on WAS 7. However, the following additional settings (which you must add to the file as described in the topic "Configuring portal-ext.properties for WAS 7 deployment") are important for portlet deployment on WAS:

```
auto.deploy.dest.dir=/home/endeca/liferay/websphere-deploy
ehcache.portal.cache.manager.jmx.enabled=false
```

Deploying Endeca data source configuration on WAS 7

To configure one or more MDEX Engines as data sources for Latitude Studio, you must deploy a JSON configuration file for each MDEX Engine.

These files should be deployed relative to the Liferay Home directory.

Sample data source configuration files are provided as `.json.sample` files in the `Latitude_2.1.0_endeca-portal-dependencies.zip` file you downloaded.

To deploy the Endeca data source configuration, upload the files to the `data/endeca-data-sources/` subdirectory.

For example, `/home/endeca/liferay/data/endeca-data-sources/default.json`.

Starting the application on WAS 7

Once the Latitude Studio application has been deployed, and the `portal-ext.properties` file has been configured and deployed, the application needs to be started.

The following steps describe this process in the IBM Integrated Solutions Console.

To start the application:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications**.
2. Select the enterprise application created when you deployed the portal WAR.
3. If the application is not already running, click **Start** to start it.
4. View your deployed application at the root context of the server.
5. Restart WAS 7.

Deploying components and other plugins in WAS 7

This section describes how to deploy components, themes, hooks, and other plugins in WAS 7.

These plugins are located in the `Latitude_2.1.0_components.zip` package.

About Liferay component pre-processing in WAS 7

WAS does not support the hot deployment of components. However, Liferay's deployment code must update plugins by adding necessary libraries and configuration files.

For example, Liferay's portlet deployment code adds the following important piece of configuration to a portlet component's `web.xml` file:

```
<context-param>
  <param-name>com.ibm.websphere.portletcontainer.PortletDeploymentEn¬
abled</param-name>
  <param-value>false</param-value>
</context-param>
```

This context parameter is important for WAS deployment, as it ensures that WAS's portal server does not attempt to load the new portlet, and instead allows Latitude Studio to load the newly deployed portlet.

For this reason, Liferay must be allowed to pre-process components before they are deployed to WAS. You upload your `.war` files to Liferay's `deploy` directory so that Liferay's deployer can find and process them.

Deploying components in WAS 7

Liferay must pre-process Latitude Studio components before they can be deployed in WAS 7.

Important: To start up, Latitude Studio requires the Endeca Theme. Even if you do not intend to use the Endeca Theme in production, you should deploy the Endeca Theme (`endeca-theme-<version>.war`).

To deploy Latitude Studio components in WAS 7:

1. Copy all component `.war` files to `${liferay.home}/deploy`.
2. Wait while Liferay pre-processes the .war files and places them in the `${liferay.home}/websphere-deploy` directory.
3. Deploy the `.war` files generated in step 2 as modules in the Latitude Studio enterprise application. To do this, you can either use:
   - The WebSphere Integrated Solutions Console.
   - The command line, using `wsadmin`.

Deploying generated .war files on WAS 7 with the Integrated Solutions Console

You can use the IBM Integrated Solutions Console to deploy the `.war` files it finds in the `websphere-deploy` directory.

 **Note:** These steps may need to be adjusted for alternate WAS configurations.

To deploy a generated `.war` file with the Integrated Solutions Console:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications**.
2. Select the enterprise application created when you deployed the portal `.war` file, then click **Update**.
3. Select **Replace or add a single module**.
4. Specify the path to deploy the file as the display name of the new module.

   For example, if you are adding `endeca-navigation-portlet.war`, specify the path as `endeca-navigation-portlet`.

5. Browse the remote file system to the newly created `.war` file in the Liferay deploy output directory.

   Continuing the example above, this might be `/home/endeca/liferay/websphere-deploy/endeca-navigation-portlet.war`.

6. After finding the file, click **Ok**.
7. Select the detailed install path. Keep the defaults on all screens except the context root.

   Set the context root to match the display name of the new plugin (in this example, `/endeca-navigation-portlet/`).

8. Once it has successfully updated, click **Save directly to master configuration**.

Using wsadmin to deploy the generated .war file on WAS 7

You can also use the `wsadmin` tool to deploy the generated `.war` file from the command line.

 **Note:** These steps may need to be adjusted for alternate WAS configurations.

In the `wsadmin` tool, enter a command similar to the example below, where the command is executed from the Liferay deploy output directory (that is, the directory containing the `endeca-navigation-portlet.war` file):

```
[WAS]/AppServer/bin/wsadmin.sh -c "\$AdminApp update LatitudeStudio
modulefile {-operation addupdate -contents endeca-navigation-port¬
```

```
let.war -contextroot /endeca-navigation-portlet/ -contenturi endeca-navi¬
gation-portlet -usedefaultbindings}" -c "\$AdminConfig save"
```

In this example:

- The enterprise application is named `LatitudeStudio`.
- The file name for the module being added is `endeca-navigation-portlet.war`.
- The module display name is `endeca-navigation-portlet`
- The command is executed in `/home/endeca/liferay/websphere-deploy/` .

Installing the Latitude Studio license

Before you can start Latitude Studio, you must install the license, which is available from the Endeca Developer Network (EDeN).

To install the license:

1. Download the Latitude Studio license (`latitude_2.1.0_studio_license.xml`) from the Latitude section of EDeN (*http://eden.endeca.com*).

   For details on navigating to this section, see the section on downloading the Latitude software.

2. Save the file to the `${liferay.home}/deploy` directory of your Latitude Studio installation.

When you start Latitude Studio, the license is installed.

Troubleshooting WAS 7 deployment

When deploying Latitude Studio on WAS 7, keep the following issue in mind .

Updating the Latitude Studio .war file

If you need to update the Latitude Studio `.war` file (not any individual plugin, but the portal `.war` itself), you must restart the WAS server. If you only restart the module, the restart might not be successful.

Using the Presentation API with Latitude 2.1

In order to support the latest enhancements to Analytics statement parsing, Endeca requires the use of the Endeca IAP 6.1.4 Presentation API with Latitude 2.1.

The 6.1.4 Presentation API can be downloaded from the MDEX Engine 6.1.4 downloads page of EDeN.

Configuring Latitude Studio on Tomcat to work with the 6.1.4 Presentation API

This topic describes how to configure a Latitude Studio implementation running on Tomcat to work with the 6.1.4 Presentation API.

To configure Latitude Studio on Tomcat to work with the 6.1.4 Presentation API:

Copy the 6.1.4 `endeca_navigation.jar` file to the `tomcat-<version>/lib/ext` directory, overwriting the existing `endeca_navigation.jar`.

Configuring Latitude Studio on WAS to work with the Presentation API

This topic describes how to configure a Latitude Studio implementation running on WebSphere Application Server to work with the Presentation API 6.1.4.

To configure Latitude Studio on WAS to work with the 6.1.4 Presentation API:

Copy the 6.1.4 `endeca_navigation.jar` file to the `WAS/AppServer/lib/ext` directory, overwriting the existing `endeca_navigation.jar`.

Chapter 8

Getting Started with Latitude Studio

This section describes how to launch and configure Latitude Studio and begin to work with it.

Starting Latitude Studio

After you complete the Latitude Studio installation, you can start and log in to the application.

Before logging in to Latitude Studio:

- Make sure you have a data source in place.
- Edit the host and port settings in `default.json` to match your data source.

To start Latitude Studio:

1. Start your application server.
2. In your Web browser, go to the portal.
3. Log in using the default login and password:

Login:	`test@endeca.com`
Password:	`test`

Updating default.json to point to your server

When you first install Latitude Studio, all components that require a backing data source are bound to a data source called `default.json`.

The `endeca-portal\data\endeca-data-sources` directory contains a `default.json` file.

Before you can start working with components in Latitude Studio, you must either:

- Update this file to point to the correct server and port.
- From the **Framework Settings** component, select a different data source to use as the default.

To update the `default.json` file:

1. In the `endeca-portal\data\endeca-data-sources` directory, open `default.json` in a text editor.

2. In the file, edit the server and port lines to reflect your server. For example:

```
{
   "server":"localhost",
   "port":"5555",
   "apiVersion":"DISCOVERY_SERVICE"
}
```

   Do not change the "apiVersion" line.

3. Save the file.

After you have updated the `default.json` file, you can log in to Latitude Studio and begin adding components to your application.

Changing Control Panel settings

After logging in to Latitude Studio, you may also want to use the **Control Panel** to configure Latitude Studio settings.

Accessing the Control Panel

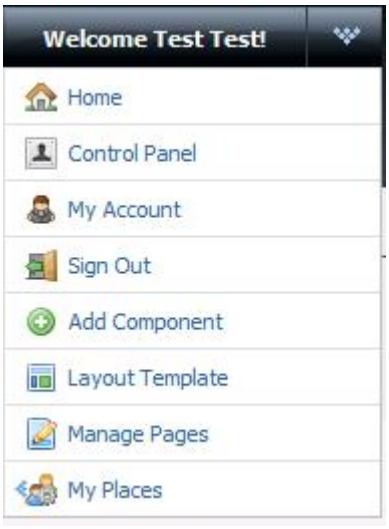
The **Control Panel** provides access to a wide range of edit controls, including managing accounts, adding new users, and monitoring performance.

For full documentation on **Control Panel** capabilities, see the *Liferay Portal Administrator's Guide*. To access a free PDF download of this guide, go to *http://www.liferay.com* and navigate to Documentation.

To access the **Control Panel**:

1. Click the Dock in the upper-right corner of the page.

   The Dock is labeled "Welcome *<user name>*!"



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

About Framework Settings

The **Framework Settings** component on the **Control Panel** provides access to state, security, and other settings.

For details about **Framework Settings**, see the *Latitude Studio Power User's Guide*.

Adding Latitude standard components

Latitude Studio contains several Latitude standard components. These components make it possible for you to add Latitude functionality to your application.

For information about building applications with Latitude Studio, see the *Latitude Studio Power User's Guide*.

To add a Latitude component to your Latitude Studio application:

1. Click the Dock in the upper-right corner of the page.
2. In the drop-down menu, select **Add Component**.

   The **Add Component** dialog box opens.
3. In the **Add Component** dialog box, expand the **Latitude** category.

   A list of the available Latitude components is displayed.
4. To add a component to the main page layout, either:
   - Click the **Add** link for the component.
   - Drag the component from the **Add Component** dialog to the page.

Chapter 9

Other Latitude Studio Installation Tasks

This section discusses some other installation tasks related to your Latitude Studio installation.

Using a different database

The Liferay portal server uses a relational database to store configuration and state, such as portlet preferences, user permissions, system settings, and more.

By default, Liferay uses Hypersonic (HSQL), which is an embedded database running inside the Java virtual machine. HSQL is useful for standing up a Liferay instance very quickly, but must NOT be used in production due to performance issues and its inability to support clustered Liferay instances.

For instructions on switching to another supported database system, see the *Liferay Portal Administrator's Guide*. Keep the following details in mind:

- Latitude Studio ships with a `portal-ext.properties` file (in the portal distribution's root directory). You can modify this file instead of creating a new one.
- Endeca has tested Latitude Studio on MySQL and DB2. Other databases are expected to work but have not been explicitly tested.

Overview of switching to a different database

This topic provides a high-level overview of the steps involved in switching from the default Hypersonic database to the production RDBMS of your choice.

Note: Because the details vary from database to database, this topic only provides a high-level overview of this process. For detailed information, see the *Liferay Portal Administrator's Guide*.

To switch to a different database:

1. Install and verify that your database is working.
2. Create a new empty database or schema for the Liferay portal.
3. Create a database user for the Liferay portal.
4. Grant that user access to the appropriate database/schema, with privileges to create tables, alter schemas, and so on in that database. Ensure that the user has remote access from the Liferay application servers.
5. Stop Liferay if it is running.

6. Edit the `portal-ext.properties` file. In the JDBC section, comment out the settings for Hypersonic, and uncomment the settings for your database.
7. Edit the settings for your database of choice, adding the appropriate username and password and editing the JDBC connection string as necessary.
8. Start Latitude Studio. Monitor its logs to ensure for any error messages while connecting to the database and creating tables.
9. After tables have been created and you have validated Liferay is running, you may remove the liferay user's alter table privileges.

   Note that you may have to add these back later if you upgrade Liferay or install components that require schema changes.

Installing Corda

The **Chart** component requires the installation of Corda charting software. Endeca recommends deploying Corda Server as a servlet when using it with Latitude Studio. If you plan to use the **Chart** component, make sure to download `Latitude_2.1.0_Corda.zip` and deploy the Corda Server servlet in that package.

Note: The Corda image authorization feature is in Beta for this release and is not supported.

Obtaining the Corda software

You download the Corda Server servlet package from the EDeN downloads page.

The Corda Server servlet software is packaged in the `Latitude_2.1.0_Corda.zip` file.

To obtain this file, download the **Latitude Studio Corda Servlet** package from the **Product Download** page on EDeN.

About the Corda Server servlet

This topic describes the Corda Server servlet shipped with Latitude Studio.

The Corda Server servlet is a Java servlet version of the Corda Server. It is designed to run on, and be accessed through, Java-enabled application servers such as Tomcat and WAS.

Because it is packaged as a servlet, you do not need to run this version of the Corda Server as a separate process over a separate server port.

For details on how the Corda Server servlet is packaged, see the *Corda documentation*.

Important: Deploying the Corda servlet on the same Tomcat server as Latitude Studio is intended for development purposes only. You should install Corda on a separate application server (or as a standalone server) for production use. If you purchased the Corda charting module (also known as the Advanced Visualization for Java and .NET module), your license entitles you to run a single production instance of the Corda server (whether deployed as a servlet or as a standalone server).

About the Latitude Studio implementation of Corda

All communication between Latitude Studio and Corda occurs through the Corda Redirector. The location of the Corda Server servlet (or Corda Server) is configurable, allowing it to be deployed locally for development and remotely for production.

The Corda Redirector module is a proxy that allows the browser requests that embed generated chart images to be routed through the application server (in this case Liferay), rather than going directly to the Corda server. All requests for Corda images are made to the Corda Redirector module.

This new redirector-based deployment model ensures a greater level of control over network infrastructure. It enables system administrators to control network access, by allowing direct access *only* to the application server(s).

For more information about the Corda Redirector, see the *Corda documentation*.

Deploying the Corda Server servlet in an application server

The Corda Server servlet can be deployed by following the standard servlet deployment procedure for the application server in question.

Before deploying Corda, you must configure it.

Note: Deploying the `corda.war` requires the use of an archiving tool to expand the archive file. This Java archive can be expanded and re-packaged with Java's jar tool or with a zip utility.

Extracting the Corda zip file

After you download the `Latitude_2.1.0_Corda.zip` file, you need to extract it to a temporary location in order to access the `corda.war` file.

To prepare the `corda.war` file for modifications:

1. To make the `corda.war` file available, unzip the `Latitude_2.1.0_Corda.zip` file to your hard drive. .
2. Expand the `corda.war` file.

   You can expand and re-package this Java archive with Java's jar tool or with a zip utility.

Configuring Corda to allow connections from other hosts

The Corda Server servlet is set up by default to accept incoming requests from any host. You must configure it to allow only incoming requests from the application server that hosts Latitude Studio.

Note: In the instructions below, the paths are relative to the location to which the `.war` was extracted.

To change the Corda configuration:

In `corda-web.war`, update the `WEB-INF/classes/Corda60/config/path.xml` configuration file to include entries for the hosts that need to embed charts powered by the Corda Server.

For example, you may add entries like the following to enable access from a specific host, from a range of domain names, or from a range of IP addresses, respectively:

```
<PathMaps Version="1.0">
   <Map Name="DefaultRead" Path="./*" Action="Load"/>
   <Map Name="DefaultSave" Path="./images/*" Action="Save"/>
   <Map Name="ValidDomain" Path="127.0.0.1" Action="allowDomain"/>
   <Map Name="ValidDomain" Path="localhost" Action="allowDomain"/>
   <Map Name="ValidDomain" Path="*" Action="allowDomain"/>
</PathMaps>
```

The last line, highlighted in bold, leaves Corda open to requests from anywhere. For this reason, you should replace **"*"** with the IP address (or range of addresses) allowed to access it (such as, for example, “192.168.*”).

Specifically, you should add the location of the application server hosting Latitude Studio.

Refer to *Corda’s documentation* for details about this configuration file.

Adding or removing PCXML templates

PCXML templates are XML-based templates that describe and define the charts and maps used by Corda. You must deploy the PCXML templates distributed with the Corda Server servlet.

Note: In the instructions below, the paths are relative to the location to which the WAR was extracted.

To deploy the PCXML templates, update the servlet with the new PCXML templates. The steps to update the servlet may differ, depending on the application server and configuration used when deploying the servlet.

In all cases, this can be accomplished by updating `corda.war` with the required changes and repeating the steps in the topic "Deploying the corda.war file" to deploy the modified `.war` file.

The PCXML chart templates are located in the following location in `corda.war`: `WEB-INF/classes/Corda60/chart_root/apfiles`.

Note: Adding new PCXML chart templates requires updates to the `Chart` component to use the newly deployed PCXML files. For more information, see the topic "Configuring Chart to use PCXML templates."

Deploying the corda.war file

After configuring Corda, you must repackage the `.war` file and then deploy it.

Note: Hot-deploying the Corda Server servlet into Liferay's deploy directory is not supported.

To deploy on a Tomcat server:

1. Re-zip the modified `corda.war` file.
2. Deploy the file into Tomcat's `webapps` directory (such as `/path/to/tomcat-<version>/webapps`).
3. Restart Tomcat.

Depending on your Tomcat configuration, the servlet container may unpack the `.war` archive, or it may operate directly from the archive. If you plan to modify Corda configuration files further, or to

deploy or modify PCXML chart templates, you may prefer to unpack the `corda.war` archive, to provide easier access to files inside the archive.

To deploy on WebSphere Application Server:

1. Re-zip the modified `corda.war` file.
2. Use the IBM Integrated Solutions Console, Deployment Manager, or `wsadmin` utility to deploy `corda.war`. The servlet should be deployed as an enterprise application with context root `/corda`.

Confirming the Corda Server servlet deployment

After deploying the Corda Server servlet, you should ensure that it is running.

The server log should display messages similar to the following example when the Corda Server servlet starts successfully:

```
Corda Server (PopChart) Version 6.0.735
  PopChart: Valid Key, OEM build for: ENDECA.
  OptiMap: No key entered, or key invalid.
  Highwire: No key entered, or key invalid.
  Cluster: No key entered, or key invalid.

Copyright 1997 - 2006, Corda Technologies, Inc. (www.corda.com) Protected
by U.S
. Patent 5,933,830. Other patents pending.

server_root: /Corda60
chart_root: chart_root
Cache Segment Size: 0
Password is Enabled, Required for Save
Maximum Threads: 64
Default Image Type is: Flash
Auto Detect PNG Support. Compression Mode: DEFAULT
```

In addition, to test the Corda Server servlet in a browser:

1. Access the following URL, using the name of your server rather than `server.example.com`:

   ```
   http://server.example.com:8080/corda/server
   ```

   A empty white box with a gray background displays.
2. Confirm that the title bar says **Corda Server (PopChart) Servlet Version 6.0.735**.
3. Access the following URL, using the name of your server rather than `server.example.com`:

   ```
   http://server.example.com:8080/endeca-corda-chart-portlet/ctRedirect
   ```

   The same empty white box and title bar mentioned above should appear.

Updating the Chart component with changes to the Corda Server servlet

By default, the `Chart` component is configured to look for the local instance of the Corda Server servlet, and fails if the servlet is not deployed.

You can change where the `Chart` component looks for Corda, if you are installing the servlet on a non-localhost machine or you have a Corda server already running elsewhere.

By default, the `Chart` component is configured to use a Corda Server deployed as a servlet on the same application server as Latitude Studio. This is a convenient configuration for single-server

deployments and development and demonstration environments. However, production environments (especially those with clustered application servers) may require alternate configuration to specify a separate location for the Corda Server.

The internal and external hosts used by Corda Server differ in cases where Corda is deployed as a standalone server and cases where Corda is deployed as a servlet.

- When deployed as a servlet, the internal and external URL typically take a form similar to `http://server.example.com:[app server port]/corda/server`.
- When deployed as a standalone server, the internal and external hosts typically take a form similar to `http://cordaserver.example.com:[corda server port]/` (where the Corda server port differs for the internal and external URLs). For details, refer to the *Corda documentation*.

Configuring the location of the Corda Server

You can configure Corda Server servlet location in Latitude Studio's **Control Panel**.

Important: Because the **Control Panel** only shows settings that exist, in order to be able to edit the Corda Server URL properties, you must first put the `Chart` component on a page.

To configure the Corda Server URL:

1. Point the cursor at the **Dock** in the upper-right corner of the page.
2. In the drop-down menu, choose **Control Panel**.
3. In the **Portal** section of the **Control Panel** navigation panel, select **Framework Settings**.
4. Update the internal, external, and redirector URLs - `df.cordaServerInternalUrl`, `df.cordaServerExternalUrl`, and `df.cordaServerRedirectorUrl` - to point to the location (host and port) of the Corda Server servlet.

   For example:

   ```
   df.cordaServerInternalUrl = http://server.example.com:8080/corda/server
   ```

5. Click **Update Settings**.
6. Restart Latitude Studio.

Configuring Chart to use PCXML templates

During deployment, you added PCXML templates to your Corda implementation. You must configure the `Chart` component to use these templates.

The `endeca-corda-chart-portlet-<version>.war` archive file is included in `Latitude_2.1.0_components.zip`.

To configure the `Chart` component's PCXML templates:

1. Open the file `WEB-INF/analytics-portlet-config.xml` in `endeca-corda-chart-portlet-<version>.war`.

   Note: This Java archive file can be expanded and re-packaged with Java's jar tool or with a zip utility.

2. Add or remove the `CordaChartConfiguration` elements.

For example, to add a new chart called 3D Pie with PCXML template `3DPie.pcxml`, update the file to include the following XML:

```
<bean id="3dPieChart" class="com.endeca.portlet.corda.CordaChartConfigu¬
ration">
  <property name="chartDisplayName" value="3D Pie" />
  <property name="pcxml" value="3DPie.pcxml" />
</bean>
```

This configuration causes the `Chart` component to display an additional option on its **Preferences** panel, allowing the use of 3D Pie as a chart style.

Troubleshooting Corda

The following section provides troubleshooting information for Corda.

Change in Corda behavior in Latitude Studio

Starting in Latitude Studio version 1.3.1, the Corda splash screen no longer renders upon entering the base URL.

In addition, URLs such as @_FILE URL are now restricted, to prevent cross-site scripting vulnerabilities.

Problems rendering a chart

When attempting to render a chart, the `Chart` component may fail to reach the Corda Server at the specified host and port.

If the host and port configuration is correct, you may need to configure Corda to allow connections from the application server hosts that are hosting Latitude Studio.

For details, see the topic *Configuring Corda to allow connections from other hosts* on page 81.

Known issue with JVM crashes related to Corda Server servlet and JIT compilation

There is a known issue with JVM crashes that is related to the Corda Server servlet and JIT compilation on the latest version of Sun's JVM (specifically, version 1.6_21).

When this JVM crash occurs, the JVM error log typically notes that the active thread is a compiler thread, similar to the following:

```
Current thread (0x00002aab04077800):  JavaThread "CompilerThread1" daemon
 [_thread_in_native, id=2353, stack(0x0000000041123000,0x0000000041224000)]
```

The compilation task being performed is related to the method `com.corda.b.dc.a()`, similar to the following:

```
Current CompileTask:
C2:4112      com.corda.b.dc.a(B)Z (559 bytes)
```

To resolve this issue, disable JIT compilation with the following JVM argument:

```
-XX:CompileCommand=exclude,com/corda/a/dc,a -XX:CompileCommand=ex¬
clude,com/corda/b/dc,a
```

There is an expected, albeit minimal, performance degradation associated with disabling JIT compilation. You should be aware of the change and may want to perform additional testing to ensure adequate performance.

Note: This issue has not been reproduced on the latest version of Sun's 1.5 JVM.

Note: This issue is not known to exist on IBM's JVM and has not been reproduced on it.

Chapter 10

Notes on Upgrading from Latitude 2.0

If you are upgrading from Latitude 2.0, then for Latitude Studio (previously the Discovery Framework), you need to be aware of the following additional tasks.

Reconfiguring existing components (Required)

Upgrading to Latitude 2.1 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.

Replacing RecordSpecListFilter in custom components (Recommended)

As of version 2.1, the `RecordSpecListFilter` function has been deprecated. In the standard components developed by Endeca, it has been replaced by the new `RecordDetailsConfig` function.

If you are using `RecordSpecListFilter` in any custom components that you have developed, we strongly recommend that you update the component as soon as possible to use `RecordDetailsConfig`.

For more information on `RecordDetailsConfig` and on component development in general, see the *Latitude Developer's Guide*.

Part 5

Uninstallation Tasks

Chapter 11

Uninstalling the Latitude Components

This chapter describes how to uninstall the various components of an Endeca Latitude installation.

Uninstalling the MDEX Engine

This section contains the procedures for uninstalling the MDEX Engine package.

Steps to uninstall the MDEX Engine on Windows

Follow these steps to uninstall the MDEX Engine from your Windows machine.

Before you begin the uninstall process, back up files that you want to retain from the MDEX Engine root directory.

You also must:

- Stop all Endeca processes, including the Dgraph.
- Close any open PDF files.

To uninstall the MDEX Engine from your Windows machine:

1. From the Windows Control Panel, select **Programs** > **Programs and Features** > **Uninstall a program**.
2. Select **Endeca Latitude 2.1.0 MDEX Engine 7.2.0 x64 Edition** from the list of installed software.
3. Click the **Uninstall** option.

   The setup wizard is launched.
4. In the wizard, make sure that the **Uninstall** radio button is selected and then click **Next**.
5. In the following dialog, click **Next** to begin the uninstall procedure.
6. Click **Finish** to exit the wizard.

Steps to uninstall the MDEX Engine on Linux

Follow these steps to uninstall the MDEX Engine from your Linux machine.

Before you begin the uninstall process, back up files that you want to retain from the MDEX Engine directory.

Make sure that you stop all Endeca processes, including the Dgraph, before uninstalling the Endeca software.

To uninstall the MDEX Engine from your Linux machine:

1. Change to the parent directory of the Latitude install directory.
2. Issue an `rm` command as in this example:

```
rm -rf endeca/Latitude/2.1.0/MDEX
```

This command removes the Endeca MDEX Engine.

Uninstalling the Latitude Data Integrator

This section contains the procedures for uninstalling the Latitude Data Integrator Designer package.

Steps to uninstall LDI on Windows

Follow these steps to uninstall the LDI Designer from your Windows client machine.

Although the uninstall procedure does not delete the LDI workspace folder, it is a good practice to back up your LDI projects before uninstalling.

To uninstall the Latitude Data Integrator Designer from your Windows machine:

1. From the Windows Control Panel, select **Programs** > **Programs and Features** > **Uninstall a program**.
2. Select **CloverETL Designer** from the list of installed software.
3. Click the **Uninstall** option.
4. In the Uninstall wizard, click **Uninstall**.

   When the uninstallation procedure is in progress, the **Back**, **Close**, and **Cancel** buttons are disabled (grayed out).

5. When the uninstallation procedure is completed, click **Close**.
6. Delete your LDI workspace folder.

As mentioned above, the LDI workspace folder is not deleted after Step 5. However, if you re-install the LDI Designer, it is recommended that you back up the LDI projects (in the workspace folder) and then delete the workspace folder. After re-installation, you can import the LDI projects from their saved location.

Steps to uninstall LDI on Linux

Follow these steps to uninstall the LDI Designer from your Linux client machine.

Although the uninstall procedure does not delete the LDI workspace folder, it is a good practice to back up your LDI projects before uninstalling.

To uninstall the Latitude Data Integrator Designer from your Linux machine:

1. Change to the parent directory of the LDI install directory.

2. Issue an `rm` command as in this example:
   ```
   rm -rf cloveretl-designer
   ```
3. Delete your LDI workspace folder.

As mentioned above, the LDI workspace folder is not deleted after Step 2. However, if you re-install the LDI Designer, it is recommended that you back up the LDI projects (in the workspace folder) and then delete the workspace folder. After re-installation, you can import the LDI projects from their saved location.

Uninstalling Latitude Studio

To uninstall Latitude Studio, remove the packages and directories that you installed.

Index

M

O

P

R

S

T

U

W