Endeca® Content Acquisition System

JSR-170 Compliant Repositories Connector Guide

Version 3.0.1 • December 2011

ENDECA
INFRONT™

Contents

Copyright and disclaimer

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

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

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

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

Outside In® Search Export Copyright © 2011 Oracle. All rights reserved.

Rosette® Linguistics Platform Copyright © 2000-2011 Basis Technology Corp. All rights reserved.

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

Trademarks

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

The software may be covered by one or more of the following patents: US Patent 7035864, US Patent 7062483, US Patent 7325201, US Patent 7428528, US Patent 7567957, US Patent 7617184, US Patent 7856454, US Patent 7912823, US Patent 8005643, US Patent 8019752, US Patent 8024327, US Patent 8051073, US Patent 8051084, Australian Standard Patent 2001268095, Republic of Korea Patent 0797232, Chinese Patent for Invention CN10461159C, Hong Kong Patent HK1072114, European Patent EP1459206, European Patent EP1502205B1, and other patents pending.

Preface

Endeca® InFront enables businesses to deliver targeted experiences for any customer, every time, in any channel. Utilizing all underlying product data and content, businesses are able to influence customer behavior regardless of where or how customers choose to engage — online, in-store, or on-the-go. And with integrated analytics and agile business-user tools, InFront solutions help businesses adapt to changing market needs, influence customer behavior across channels, and dynamically manage a relevant and targeted experience for every customer, every time.

InFront Workbench with Experience Manager provides a single, flexible platform to create, deliver, and manage content-rich, multichannel customer experiences. Experience Manager allows non-technical users to control how, where, when, and what type of content is presented in response to any search, category selection, or facet refinement.

At the core of InFront is the Endeca MDEX Engine,™ a hybrid search-analytical database specifically designed for high-performance exploration and discovery. InFront Integrator provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. InFront Assembler dynamically assembles content from any resource and seamlessly combines it with results from the MDEX Engine.

These components — along with additional modules for SEO, Social, and Mobile channel support — make up the core of Endeca InFront, a customer experience management platform focused on delivering the most relevant, targeted, and optimized experience for every customer, at every step, across all customer touch points.

About this guide

This guide describes the tasks necessary to configure the JSR-170 Compliant Repositories CMS connector.

It assumes familiarity with the concepts of the Endeca Content Acquisition System and the Endeca Information Transformation Layer. For more information, see the *Endeca CAS Developer's Guide* and the *Endeca Forge Guide*.

Who should use this guide

This guide is intended for application developers who are building applications using the Endeca Content Acquisition System, and are responsible for gathering, crawling, joining and feeding the data in different source formats into the Endeca pipeline to transform them into Endeca records.

Conventions used in this guide

This guide uses the following typographical conventions:

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

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

Contacting Endeca Customer Support

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

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

Chapter 1

Configuration steps for JSR-170 compliant repositories

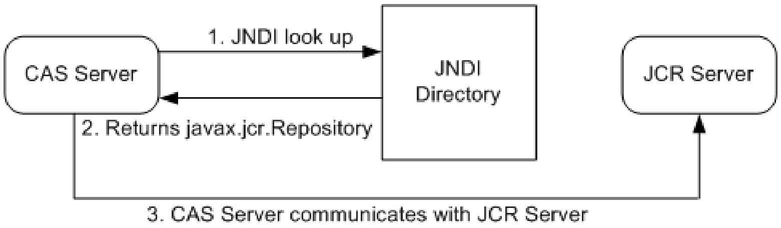
Set up the CAS Server for crawling JSR-170 compliant repositories, and set options specific to the repository in the CAS Console for Endeca Workbench.

Exposing a remote JCR server to the CAS Server

The instructions below provide an overview of how to expose a Java Content Repository (JCR) server to the CAS Server. This configuration enables communication among the CAS Server, JNDI, and the JCR server components in an implementation.

The CAS Server requires that the `javax.jcr.Repository` instance be exposed in the JNDI directory in order for the CAS Server to communicate with the remote JCR server. If a `javax.jcr.Repository` instance is not available in the JNDI directory, a communication problem occurs when the CAS Server attempts to contact the JCR server.

The following diagram shows the communication process among the components:



1. JNDI look up
CAS Server
JNDI Directory
JCR Server
2. Returns javax.jcr.Repository
3. CAS Server communicates with JCR Server

1. The CAS Server calls JNDI to look up the name of the remote JCR repository.
2. The JNDI directory returns an instance of a `javax.jcr.Repository`.
3. Using the `javax.jcr.Repository` handle, the CAS Server communicates directly with the remote JCR repository.

The implementation of the steps to expose a remote JCR server are specific to your environment. The steps below provide high-level guidance.

To expose a remote JCR Server to the CAS Server:

1. Configure the JCR server to make the `javax.jcr.Repository` handle available in the JNDI directory.

For more information, see the JNDI documentation at *http://java.sun.com/products/jndi/docs.html* and your JCR-compliant CMS documentation.

2. Configure the Endeca CAS Service to point at the appropriate JNDI directory.

   For more information about how to configure the client side CAS JNDI libraries, see the JNDI InitialContext documentation at *http://java.sun.com/javase/6/docs/api/javax/naming/InitialContext.html*. This configuration usually consists of modifying `cas-service.bat`, `cas-service-wrapper.conf` (Windows) or `cas-service.sh` (UNIX) to pass additional system properties to the JVM running the CAS Service.

   Here are several examples: On UNIX, you modify `cas-service.sh` by adding an additional `JVM_ARGS` setting:

   ```
   JVM_ARGS="$JVM_ARGS <your JNDI setting>"
   ```

   On Windows, if you are running the Endeca CAS Service from the Microsoft Services console, you modify `cas-service-wrapper.conf` by adding an addtional JVM setting:

   ```
   wrapper.java.additional.14=-<your JNDI setting>
   ```

   On Windows, if you are running the Endeca CAS Service from the command prompt, you modify `cas-service.bat` by adding an additional JVM setting:

   ```
   SET JVM_ARGS=%JVM_ARGS% -<your JNDI setting>
   ```

Setting up the CAS Server for JSR-170

To crawl a JSR-170 compliant repository, perform additional configuration steps on the machine where the CAS Server will be running.

It is assumed that your CAS Server is set up by including the libraries for your implementation of the JSR-170 API (Content Repository for Java^{TM} technology API).

To configure the CAS Server for a JSR-170 compliant repository:

1. Add any custom Java libraries that are being used by the repository to the `<install path>\CAS\`*`version`*`\lib\server` directory (on Windows) and to `<install path>/CAS/`*`version`*`/lib/server` (on UNIX).
2. Restart the Endeca CAS Service.

To run a crawl on your repository, you also specify options specific to the repository in the CAS Console.

Configuration properties for a JSR-170 connector

To configure a JSR-170 connector, specify the configuration properties listed below.

Note: In addition to configuring the connector-specific property listed below, you must enter values for the data source username and password.

Create the following configuration properties using either CAS Console or the CAS Server Command-line Utility.

CAS Property Display Name	CAS Property Name	Property Description
JNDI name or RMI address	`jndiName`	(Required). Specify the JNDI name or RMI address that CAS uses to retrieve the repository implementation.
JCR repository retrieval method	`JNDIOrRMI`	(Required). Specify `JNDI` if you specified a JNDI name above or specify `RMI` if you specified an RMI address.

Note: Properties are case sensitive.

About obtaining ACL properties

The CAS Server does not generate ACL properties from JSR-170 compliant repositories, because the JSR-170 specification does not yet provide standard mechanisms for retrieving ACLs (Access Control Lists).

If ACL properties are required in your pipeline, consider the following method:

- If the CAS Server generates ACLs as generic name-value pairs for the repository items that have ACLs, you can use record manipulators to process them in the pipeline.

For additional information on retrieving ACLs, please contact Endeca Customer Support.