Oracle Endeca® Commerce

Security Guide

Version 11.0 • January 2014

Contents

Copyright and disclaimer

Preface

Oracle Endeca Commerce is the most effective way for your customers to dynamically explore your storefront and find relevant and desired items quickly. An industry-leading faceted search and Guided Navigation solution, Oracle Endeca Commerce enables businesses to help guide and influence customers in each step of their search experience. At the core of Oracle Endeca Commerce is the MDEX Engine™, a hybrid search-analytical database specifically designed for high-performance exploration and discovery. The Endeca Content Acquisition System provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. Endeca Assembler dynamically assembles content from any resource and seamlessly combines it into results that can be rendered for display.

Oracle Endeca Experience Manager is a single, flexible solution that enables you to create, deliver, and manage content-rich, cross-channel customer experiences. It also enables non-technical business users to deliver targeted, user-centric online experiences in a scalable way — creating always-relevant customer interactions that increase conversion rates and accelerate cross-channel sales. Non-technical users can determine the conditions for displaying content in response to any search, category selection, or facet refinement.

About this guide

This guide describes the Endeca security features and the major tasks involved in using them to develop a secure Endeca implementation.

Who should use this guide

This guide is for developers who are responsible for implementing security features in Endeca applications.

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 Oracle Support

Oracle Support provides registered users with important information regarding Oracle Endeca software, implementation questions, product and solution help, as well as overall news and updates.

You can contact Oracle Support through Oracle's Support portal, My Oracle Support at *https://support.oracle.com*.

Chapter 1

Introduction

Oracle Endeca software is designed to provide a high level of security for your applications. However, you can enhance its security by following the recommended measures described in this guide.

Chapter 2

Basic Security Measures

This section describes basic measures that you can follow to enhance the security of your Endeca Commerce implementation.

Firewall Security

Your MDEX Engine and all other components of the Endeca Commerce implementation – except for your web application – must be placed behind a secure firewall.

Ensure that ports are open to enable the web application to communicate through the firewall with the MDEX Engine and Workbench.

Installing Endeca Commerce on its own machine

To protect your Oracle Endeca implementation against viruses, malware, and other forms of malicious interference, Oracle recommends that you host your Oracle Endeca software on a machine on which the only other software is the operating system. Running additional software on the same machine – for example, email applications – can compromise the security of your Oracle Endeca implementation.

Securing Your Oracle Endeca Components

Be sure to protect your Oracle Endeca components against unauthorized access by installing them in restricted parts of your local network.

In particular, be sure to place the following components in restricted parts of your network:

- Endeca Application Controller (EAC)
- Content Acquisition System (CAS)
- Log Server
- MDEX Engine
- Workbench

In addition, you can restrict access to Workbench by defining profiles of the users who will be authorized to access Workbench. See *Configuring Workbench for Security* on page 13.

Chapter 3

Configuring Workbench for Security

This section describes how to authenticate Workbench users against credentials stored on an LDAP server or an Active Directory server, or to create user profiles for authenticating and managing Workbench users.

Integrating Workbench with an LDAP Server for User Authentication

A Workbench administrator can configure Workbench to authenticate users against user profiles stored on an LDAP server. The roles and permissions associated with each user profile are stored in the Workbench database.

The user profiles in the LDAP directory must be created independently of Workbench, which cannot write anything to LDAP directories.

For information about how to integrate LDAP with Oracle Endeca Workbench, refer to the *Oracle Endeca Commerce Administrator's Guide*.

Managing Workbench Users Through Profiles

You can also control access to Workbench by creating Workbench user profiles.

Endeca Workbench users, roles, and permissions can be defined by an Endeca Workbench administrator. Endeca Workbench users log in to an application in Endeca Workbench with basic user name and password authentication. Before a business user can log in to an application in Endeca Workbench, an Endeca Workbench administrator or a user with the settings role must create a profile for the user that includes the following:

- user name
- password

Note: Take care not to lose passwords. Passwords cannot be looked up or reset.

- roles and permissions
- user identity information such as first name, last name, and email address

For information about how to create a Workbench user profile, refer to the *Workbench Administrator's Guide*.

Securing the Built-in Workbench Admin Account

Be sure to create a strong password for the Workbench Admin account.

Use a generated password whenever possible. If you create a password yourself, follow these standard guidelines to make the password as strong as possible:

- Use passwords that are at least 12 characters long.
- Do not use passwords that contain repeated elements, dictionary words, sequences of letters or numbers, user names, names of relatives, friends, or pets, or biographical information.
- Include numbers, and symbols if possible.
- Use a mixture of upper-case and lower-case letters, if your can system distinguish them from each other.
- Do not use the same password for different sites or for different purposes.
- Do not use passwords that refer to any of your personal preferences or dislikes.

For more information, consult as standard reference on the topic of password strength.

Chapter 4

Configuring the MDEX Engine to run under SSL

This section describes how to configure an MDEX Engine to run under SSL.

Generating SSL certificates

Oracle Endeca implementations that do *not* use the Assembler can run the MDEX Engine under SSL.

Use the `enecerts` utility program to generate new SSL certificate files. The `enecerts` utility resides in the `$ENDECA_MDEX_ROOT/bin` directory (`%ENDECA_MDEX_ROOT%\bin` on Windows) under the name `enecerts` (`enecerts.exe` on Windows).

The two typical scenarios for generating SSL certificates are:

- You are setting up SSL for the first time and need to generate the set of standard certificates.
- You want to generate custom certificates, such as those with a private key size greater than the default 1024 bits.

Generating custom certificates

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

You can generate two types of customized certificates by:

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

The next two sections describe these operations.

Specifying a different certificate key size

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

```
--keysize bits
```

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

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

```
enecerts --keysize 2048
```

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

Using your Certificate Authority file to generate certificates

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

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

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

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

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

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

```
--CAcert cert-pem
```

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

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

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

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

Selecting an Encryption Algorithm

The MDEX Engine can optionally be configured to require encryption when other components of your Endeca Commerce implementation communicate with it.

Encryption is required whenever the the `-sslcertfile` option is used with the `dgidx` command that starts the MDEX Engine. For information about the `dgidx` command and its options, refer to the *Endeca Commerce Administrator's Guide*.

Whenever encryption is required, the MDEX Engine and the client with which it is negotiating a connection together choose an appropriate encryption algorithm from Oracle's approved list of algorithms.

However, you may want to limit the available choices to a specific algorithm or algorithms on the approved list. To do this, you can specify the algorithm or algorithms in configurations or on the command line where encryption algorithms are accepted. If you specify more than one algorithm, the component and the MDEX Engine will negotiate and decide which one to use.

You specify the algorithms by their standard names, such as DHE-RSA-AES256-SHA. If you specify more than one algorithm, you must separate their names with colons; for example: `DHE-RSA-AES256-SHA:DHE-DSS-AES256-SHA`.

Each Endeca Commerce component uses its own syntax for accepting specific algorithms as input. For example, the dgraph command uses the `--sslcipher` option, as follows:

```
dgraph --sslcipher DHE-RSA-AES256-SHA:DHE-DSS-AES256-SHA
```

Note: You can also specify encryption algorithms not on the Oracle-approved list in Appendix A, provided that these algorithms are supported by the client with which the MDEX Engine is communicating. In this case, however, the encryption is not guaranteed to be secure.

Chapter 5

Securing the Assembler Admin Servlet

This section describes how to provide secure communications with your web application's Admin servlet using the BASIC authentication mechanism.

Configuring Secure Access to the Admin Servlet

You can use the BASIC authentication mechanism to provide secure communication between your web application's Admin servlet and the AssemblerUpdateComponent and the Usage collector components of the Deployment Template.

To secure access to the Admin servlet, follow these steps:

1. Before modifying the configuration of the Endeca Application Controller (EAC), be sure to secure the Admin servlet with BASIC authentication using the standard J2EE mechanism.
2. Add the BASIC authentication credentials to the credential store. EAC components reference these credentials from the credential store to authenticate the Admin servlet. For example, if you specified `webAp¬pAdmin` (user name) and `complexP@ssword` (password) as the BASIC credentials, you can follow these steps to add them to the credentials store:
   a) Go to the `credential_store\bin` directory of your Tools and Framework installation.
   b) Run following command: `manage_credentials add --key webAppAdminCredentialsKey --user webAppAdmin`
   c) When prompted, enter the password `complexP@ssword` from Step 2. The following output appears on the console:

```
manage_credentials add --key webAppAdminCredentialsKey --user webAppAdmin

Enter password for user webAppAdmin :

Re-enter password to conform :
21 Oct 2013 12:43:51,547  INFO CSFHandler:139 - Credential successfully cre¬
ated for map : endecaToolsAndFrameworks.
```

3. Modify LiveAppServerCluster.xml to reference credentials. To do this, follow these steps:
   a) Open `LiveAppServerCluster.xml` in the `config\script` directory of your EAC application.
   b) Add the following code to LiveAppServerCluster.xml, to reference the credentials that you created: `<basic-credentials id="webAppAdminCredentials" credentialsStore="csfManager" credentialsKey="webAppAdminCredentialsKey"/>`

c) Modify a <web-app> element to enable your web application to reference these BASIC credentials :

```
<web-app id="MyWebApp"
contextPath="/my-web-app"
adminCredentials="webAppAdminCredentials" />
```

4. Follow these steps to verify that the usage collection and promotion mechanism are able to authenticate access to the Admin servlet:
   a) Go to the control directory of your EAC application.
   b) Run the `collect_usage` command.
   c) Verify that usage information is collected in the `logs\usage` directory of your EAC application.
   d) Verify that the promotion mechanism works correctly by making changes in your Authoring environment and running the `promote_content` command from the control directory of your EAC application. Verify that your changes are successfully promoted to the live environment.

Appendix A

Default Encryption Algorithms

Endeca Commerce components by default use an encryption algorithm chosen from a list of approved algorithms for secure communication with other Endeca Commerce components.

Default Algorithms

The following list contains the currently approved algorithms. The order in which the items appear in the list is not indicative of any priority among the items.

- ADH-AES128-SHA
- ADH-AES256-SHA
- AES128-SHA
- AES256-SHA
- DHE-DSS-AES128-SHA
- DHE-DSS-AES256-SHA
- DHE-RSA-AES128-SHA
- DHE-RSA-AES256-SHA

Index