Endeca® IAP Workbench

Administrator's Guide

Endeca Workbench Administrator’s Guide • September 2008

Contents

Preface

The Endeca® Information Access Platform is the foundation for building applications based on Endeca MDEX Engine® technology. With the Endeca Information Access Platform, you can build Guided Navigation® functionality into your Web applications. The Endeca Guided Navigation solution puts the results of all navigation, search, and analytic queries in an organized context that shows users how to refine and explore further. This helps solve the problems associated with information overload by guiding users as they quickly and precisely navigate through large data sets.

About this guide

This guide describes the tasks involved in the configuration and administration of an Endeca implementation running in an Endeca Control System environment.

This chapter provides an overview of the administrative aspects of the Endeca tools. For a general introduction to the broad capabilities, usage, and workflow of Endeca tools, see the *Endeca Getting Started Guide*.

Endeca IAP Workbench contains configuration and administrative functionality for system administrators as well as business logic functionality for business users. IAP Workbench provides the primary means to administer your Endeca implementation in a Tools environment.

Who should use this guide

This guide is intended for system administrators and others who are managing the day-to-day operations of the Endeca Information Access Platform. It may also be of interest to developers while they are deploying an Endeca implementation.

Endeca IAP Workbench is a Web-based tool intended for business users and system administrators. For business user information, see the *Endeca Business User's Guide*.

With IAP Workbench, system administrators can perform any of the following tasks:

- Provision the hosts available to an Endeca implementation.
- Provision the applications available to an Endeca implementation.
- Provision the scripts, such as the report generator script, or a baseline update script to an Endeca implementation.
- Configure SSL settings, report generation, and set up a preview application for dynamic business rule testing.
- Perform system operations such as running baseline updates or starting and stopping the MDEX Engine or Log Server.
- Monitor the status of system components such as Forge, Dgidx, MDEX Engine, Log Server, and Report Generator.

IAP Workbench and Developer Studio require the Endeca Application Controller (EAC) to control and communicate with other components and hosts in an Endeca implementation.

Contacting Endeca Standard Customer Support

You can contact Endeca Standard Customer Support through the online Endeca Support Center (https://support.endeca.com).

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.

Chapter 1

Working with IAP Workbench

This chapter discusses a variety of administrative tasks associated with Endeca IAP Workbench. This chapter includes the following sections:

- The IAP Workbench login page
- Backing up and restoring an Endeca project
- Viewing system logs
- Specifying which Dgraphs to update with configuration changes
- Changing Endeca HTTP service ports
- Encoding of workflow emails in IAP Workbench

The IAP Workbench login page

The default URL of the IAP Workbench login page is:

`http://`*`WebStudioHost`*`:8888`

If you used a different HTTP Connector port when you configured IAP Workbench, substitute that port's number for 8888.

Logging in to IAP Workbench as an administrator

Upon installation, IAP Workbench has a predefined administrator user with full administration privileges.

To log in to IAP Workbench:

1. In a Web browser, navigate to the IAP Workbench login page.
2. Specify a username and password. The username and password for the predefined administrator are both **admin**.
3. If you have an application provisioned, select the application to access. An admin user can also log to IAP Workbench without any applications provisioned in the system.
4. Click Log In.

After your initial login, you can change the password of the predefined admin user or create additional users and administrators. For details, see "Managing Users in IAP Workbench".

Hiding the list of applications on the login page

By default, IAP Workbench shows all available Endeca applications in a drop-down list on the login page. Business users can log in to any application that an administrator has added them to. In cases where you do not want IAP Workbench users to see all available IAP Workbench applications on the login page, you can hide the drop-down list of applications displayed in IAP Workbench.

To hide the drop-down list of applications on the login page of IAP Workbench:

1. Stop the Endeca HTTP service.
2. Open the webstudio.properties file located in **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF\conf** (on UNIX).
3. Locate the `com.endeca.webstudio.hide.login.application.dropdown` property, for example:

   ```
   # Hides the dropdown for selecting an application
   # on the login page.
   com.endeca.webstudio.hide.login.application.dropdown=false
   ```
4. Set the value of the property to **true**, for example:

   ```
   com.endeca.webstudio.hide.login.application.dropdown=true
   ```
5. Save and close the webstudio.properties file.
6. Start the Endeca HTTP service.

Application-specific login pages

The URL for an application-specific login page is http://*WebStudioHost*:8888/login/*AppName.* The value of *AppName* is the name you provided when creating the application using IAP Workbench (or eaccmd or the custom Web services interface). For example, if you created an application named “wine” on localhost, the URL is http://localhost:8888/login/wine.

Backing up and restoring an Endeca project

The backup process allows you to take a snapshot of your project including its users, rule groups, and permissions data. This process does not include the provisioning information for an application.

For backup purposes, an Endeca project is composed of three pieces:

- Instance configuration — the Endeca project files created by Developer Studio.

- Web Studio store — a directory that contains a database of users, rule groups, and associated permission information.
- Configuration files — XML and properties files that customize the behavior of a IAP Workbench installation.

Together, the instance configuration and the Web Studio store are the backup. The two are a snapshot of your project and all its associated user and permission information.

Backing up the instance configuration

The instance configuration is created in Developer Studio and consists of pipeline components, Endeca properties and dimensions, precedence rules, dynamic business rules, and user profiles.

To back up the instance configuration:

1. Stop the Endeca HTTP service.
2. Copy the **emanager** directory, including all its subdirectories, from **%ENDECA_CONF%\state** (on Windows) or **$ENDECA_CONF/state/** (on UNIX)to another location.

   Note: *Recall that the default location of* **%ENDECA_CONF%** *on Windows is* **C:\Endeca\MDEXEngine\workspace.**
3. Start the Endeca HTTP service.

Restoring a backup of the instance configuration

You can only restore backups of the instance configuration within the same major.minor release version, for example between 1.0.0 and an installation of a later 1.0.x version, but not between a 1.0.x version and a 1.1.x version.

To restore a backup of the instance configuration:

1. Stop the Endeca HTTP service.
2. Delete the **emanager** directory from **%ENDECA_CONF%\state\** (on Windows) or **$ENDECA_CONF/state/** (on UNIX).

3 Copy the **emanager** directory that you backed up earlier to **%ENDECA_CONF%\state\** (on Windows) or **$ENDECA_CONF/state/** (on UNIX).

4 Start the Endeca HTTP service.

Downloading the instance configuration

How is this different from the backup procedure above? Why would you use one over the other?

The Instance Configuration page under Application Settings in IAP Workbench allows you to view and download the instance configuration that is currently being used by IAP Workbench. The project XML files that make up the instance configuration are zipped into one file. This feature is intended primarily for debugging and support purposes. See the Endeca IAP Workbench Help for how to download an instance configuration.

For information on transferring your instance configuration from staging to production environment, and using the **emgr_update** utility, see NEED_NEW_REFERENCE_HERE.

Backing up the Web Studio store

The Web Studio store contains information such as users and permissions, as well as preview application settings.

To back up the Web Studio store:

1 Stop the Endeca HTTP service.

2 Copy the **webstudiostore** directory, including all its subdirectories, from **%ENDECA_CONF%\state** (on Windows) or **$ENDECA_CONF/state/** (on UNIX) to another location.

   Note: *Recall that the default location of* **%ENDECA_CONF%** *on Windows is* **C:\Endeca\MDEXEngine\workspace.**

3 Start the Endeca HTTP service.

Restoring a backup of the Web Studio store

You can restore backups of the Web Studio store to an installation of the same version or later.

To restore a backup of the instance configuration:

1. Stop the Endeca HTTP service.
2. Delete the **webstudiostore** directory from **%ENDECA_CONF%\state\** (on Windows) or **$ENDECA_CONF/state/** (on UNIX).
3. Copy the **webstudiostore** directory that you backed up earlier to **%ENDECA_CONF%\state\** (on Windows) or **$ENDECA_CONF/state/** (on UNIX).
4. Start the Endeca HTTP service.

Backing up the IAP Workbench configuration files

IAP Workbench uses several configuration files located in **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX) to customize the behavior of various aspects of IAP Workbench.

File name	Description
Login.conf	Configuration for user authentication using LDAP
webstudio.properties	Miscellaneous configuration parameters for IAP Workbench
webstudio.log4j.properties	Configuration for the IAP Workbench system log and audit log
ws-extensions.xml	Definitions of Workbench extensions
ws-mainMenu.xml	Definitions of the IAP Workbench navigation menu and launch page
ws-roles.xml	Definitions of custom IAP Workbench user roles

To preserve the settings controlled by each of these files, simply copy them to another location.

Restoring a backup of the IAP Workbench configuration files

In general, you should only restore backups of configuration files to the same exact version of IAP Workbench, for example, from 1.0.1 to 1.0.1, but not from 1.0.1 to any other 1.0.x version. Upgrading your installation may introduce configuration changes that require you to manually merge your configuration files. For more details about configuration changes, see the *Endeca Migration Guide*.

To restore a backup of the IAP Workbench configuration files:

1 Stop the Endeca HTTP service.

2 Copy the files that you backed up earlier to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).

3 Start the Endeca HTTP service.

Viewing system logs

In addition to viewing component logs, you can also check the Endeca Application Controller and IAP Workbench logs that are located in **%ENDECA_CONF%\logs** (on Windows) or **$ENDECA_CONF/logs** (on UNIX). The following logs can be found in this directory:

File name	Description
webstudio.log	System log for IAP Workbench, including activity such as user logins, updates to instance configuration, and IAP Workbench errors.
webstudio_audit.log	Audit log for activity such as business rule and search configuration changes. Business rule logging records the name of the rule being modified, when it was modified, who modified it (according to IAP Workbench user name), and any note associated with the change.

Log file naming and rolling

By default, the IAP Workbench log and audit file have a maximum size of 1MB. Each of the logs is part of a four-log rotation.

Configuring the IAP Workbench logs

Both the IAP Workbench system log and audit log are configured by the **webstudio.log4j.properties** file, located in **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).

By editing the configuration file, you can control the log level, the maximum file size, and the number of files in the log rotation. You can also optionally direct the output of any IAP Workbench logger to the console or to another file.

To configure the behavior of the IAP Workbench logs:

1. Stop the Endeca HTTP service.
2. Navigate to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).
3. Open the **webstudio.log4j.properties** file.
4. Modify the configuration file as needed. For more information, see the comments in **webstudio.log4j.properties** and the log4j documentation at http://logging.apache.org/log4j/.
5. Save and close the file.
6. Start the Endeca HTTP service.

Specifying which Dgraphs to update with configuration changes

By default, IAP Workbench updates all Dgraphs that are defined in your application whenever a user saves changes to the instance configuration (including changes to dynamic business rules, keyword redirects, thesaurus entries, and so on). Beginning in 5.1.5, it is possible to specify which Dgraphs are updated with configuration changes made in IAP Workbench.

Omitting some Dgraphs from the update process can offer performance improvements when saving changes. It also allows you to control which servers can be updated directly by business users working in IAP Workbench.

You specify that a Dgraph should not be updated by IAP Workbench by defining a custom EAC property of **WebStudioSkipConfigUpdate** set to **true** on the appropriate component. You can do this using one of the following methods:

- Specify the property on the Dgraph component in the EAC provisioning file. For details, see the *Endeca EAC Guide*.
- Specify the property on the Dgraph component using the EAC Admin Console in IAP Workbench. For details, see the *Endeca IAP Workbench Help*.

Note that only Dgraphs are updated with configuration changes, so this property does not apply to Agraph components.

If you omit one or more Dgraphs from the update process, you may want to specify which MDEX Engine is used for preview to ensure that the information displayed in the preview status messages reflects the configuration changes made in IAP Workbench. For details, see “Specifying which MDEX Engine to use for preview” on page 33.

Changing Endeca HTTP service ports

The port on which the Endeca HTTP service and also IAP Workbench listen are specified in the **server.xml** file, which is located in the **%ENDECA_CONF%\conf** directory (**$ENDECA_CONF/conf** for UNIX). The file also specifies the default server port. The default values are:

- Port 8090 for the Endeca HTTP service shutdown port.
- Port 8888 for the Endeca HTTP service port.

You can change either or both of these ports, as long as you choose a new port that is not being used.

To change the Endeca HTTP service shutdown port:

1 Open the **server.xml** file in a text editor.

2 Find the **Server** element in the file:

```
<!--  NOTE: ENDECA HAS MODIFIED TOMCAT'S DEFAULT SERVER PORT OF 8005.
      ENDECA'S USES A DEFAULT SERVER PORT OF 8090
-->
<Server port="8090" shutdown="SHUTDOWN" debug="0">
```

3 Change the number in the **port** attribute to the new port you want to use.

4 Save and close the **server.xml** file.

5 Restart the Endeca HTTP service.

On UNIX:

a Stop the Endeca HTTP service using:

```
$ENDECA_ROOT/tools/server/bin/shutdown.sh
```

b Restart the Endeca HTTP service using:

```
$ENDECA_ROOT/tools/server/bin/startup.sh
```

On Windows:

a From the Windows Control Panel, select Administrative Tools, and then select Services.

b In the right pane of the Services window, right-click Endeca HTTP service and choose Restart.

c Close the Services window.

To change the Endeca HTTP service port:

1 Open the **server.xml** file in a text editor.

2 Find the non-SSL HTTP/1.1 **Connector** element:

```
<!-- NOTE: ENDECA HAS MODIFIED THE DEFAULT TOMCAT NON-SSL HTTP PORT OF 8080.
      ENDECA' USES A DEFAULT NON-SSL HTTP PORT OF 8888
-->
<!-- Define a non-SSL HTTP/1.1 Connector on port 8888 -->
    <Connector className="org.apache.catalina.connector.http.HttpConnector"
               port="8888" minProcessors="5" maxProcessors="75"
               enableLookups="true" redirectPort="8443"
               acceptCount="10" debug="0" connectionTimeout="60000"/>
```

3. Change the number in the **port** attribute to the new port you want the Endeca Application Controller and Web Studio to use.
4. Save and close the **server.xml** file.
5. Restart the Endeca HTTP service, as documented in step 5 of the previous procedure.

Encoding of workflow emails in IAP Workbench

Any time a user makes a change to the workflow state of a dynamic business rule and clicks Save Changes, the Add a note page displays. On the Add a note page, the user can choose to click Add to store a note or Add and Email to launch an email client to send a change notification as well as the text of the note. For more information about workflow for business rules in IAP Workbench, see the *Endeca Business User's Guide*.

To support non-ASCII characters in workflow emails, you can configure IAP Workbench to use UTF-8 encoding. Note that some email clients, including Microsoft Outlook 2003, do not support UTF-8 encoding in mailto URLs, which causes extended characters not to display properly. You should only enable UTF-8 encoding if you are certain that it is supported on all email clients in your organization.

The default setting in IAP Workbench encodes workflow email notifications using the escape function in JavaScript. On most systems this results in ISO-8859-1 encoding (which is supported by Outlook), but the actual encoding may depend on system settings on the client machine.

To enable UTF-8 URL encoding in workflow emails:

1. Stop the Endeca HTTP service.
2. Navigate to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).
3. Open the **webstudio.properties** file, and locate the **com.endeca.webstudio.useUTF8InMailToUrls** property, for example:

   ```
   # URL encoding for workflow emails
   com.endeca.webstudio.useUTF8InMailToUrls=false
   ```
4. Change the value of the property to **true**, for example:

   ```
   com.endeca.webstudio.useUTF8InMailToUrls=true
   ```
5. Save and close the file.
6. Start the Endeca HTTP service.

Note that although UTF-8 support varies depending on the default email client on each user's machine, this setting applies to all workflow email messages created by IAP Workbench.

Chapter 2

Managing Users in IAP Workbench

This chapter describes the IAP Workbench user and permissions model, and how to manage users within IAP Workbench.

The chapter includes the following sections:

- Users, roles, and permissions in IAP Workbench
- IAP Workbench user roles
- Assigning rule group permissions to IAP Workbench users

Users, roles, and permissions in IAP Workbench

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

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

Roles dictate which IAP Workbench features are available to users. User identity information provides a way to associate name and contact information with user names in IAP Workbench.

If you have IAP Workbench configured to use LDAP for user authentication, an administrator can create a user profile where the password and identity information is stored and managed in an LDAP directory. LDAP integration also allows you to assign roles and permissions across an entire LDAP group rather than configuring each user individually. For more information about configuring IAP Workbench with LDAP, see “LDAP Integration with IAP Workbench”.

Each business user profile is associated with a specific application. A business user profile cannot span multiple applications. In cases where you might want the same user in multiple applications, an administrator can create a number of identical business user profiles for any number of applications. Administrators, on the other hand, span applications across IAP Workbench. For the process to add users and modify user names, passwords, and roles, see the *Endeca IAP Workbench Help*.

IAP Workbench predefined admin user

IAP Workbench has a predefined administrator with full administration privileges. An administrator is granted all roles in the system. The user name for the predefined IAP Workbench administrator is **admin** and the default password is **admin**. After signing in as the admin user, you can modify the password but not the user name.

The admin user can create additional users and administrators in IAP Workbench. Only an administrator can create other administrators. An administrator can also delete other administrators, including the predefined admin user, as long as there is always at least one administrator in the system. If you have LDAP authentication enabled, see also “Administrators in IAP Workbench with LDAP” on page 38.

An administrator is not associated with an application in the same way that business users are. Each business user is associated with a particular application. Administrators span applications, so an administrator can add or remove applications without being affected by that addition or removal.

IAP Workbench user roles

A IAP Workbench administrator can assign users any of the following roles in the table below. A user who does not have any roles assigned is unable to log in to IAP Workbench. For information about how to add and configure users, see the *Endeca IAP Workbench Help*.

Role name	Role description
dimorder	Provides access to the Dimension Order page.
eacconsole	Provides access to the EAC Admin Console page. Users with this role cannot modify provisioning information on the EAC Admin Console. However, they can start and stop Endeca components and EAC scripts.
phrases	Provides access to the Phrases page.
redirects	Provides access to the Keyword Redirects page.
reporting	Provides access to the Reporting page.
rules	Provides access to the Rule Manager page.

Role name	Role description
settings	Provides access to all pages under the Application Settings section. This includes the following pages: Instance Configuration, Resource Locks, User Management, Rule Group Permissions, Preview App Settings.
stopwords	Provides access to the Stop Words page.
thesaurus	Provides access to the Thesaurus page.
admin	This is a cumulative role that provides access to pages enabled by all the predefined user roles in IAP Workbench. This role cannot be assigned to users, but is automatically assigned to any administrators that you create. It is possible to disable admin users from modifying provisioning information. For more information, see “Disabling the admin role from modifying provisioning information” on page 23.

Custom user roles in IAP Workbench

In addition to the predefined user roles in IAP Workbench, you can also define custom roles, for instance to control access to IAP Workbench extensions. For more information about extensions, see “Workbench extensions” on page 65.

Like the predefined user roles, custom roles span applications. Administrators are automatically granted all roles including custom roles.

Custom roles are defined in the **ws-roles.xml** file in **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).

The default **ws-roles.xml** file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<roles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="roles.xsd">

</roles>
```

Each role is defined in a **role** element within **roles**. You can specify as many additional roles as you need by adding more **role** elements. The following attributes must be defined for each role:

Attribute name	Attribute value
id	A unique string identifying this role. Do not define a custom role with the same id as one of the predefined user roles: admin, crawler, dimorder, eacconsole, phrases, redirects, reporting, rules, settings, stopwords, thesaurus. Roles are listed in alphabetical order by id in the User Management page in IAP Workbench. ***Note:*** *Modifying this value after the rule is created deletes the original role and creates a new role.*
defaultName	The display name for this role that appears on the User Management page in IAP Workbench.
defaultDescription	A brief description of this role that appears on the User Management page in IAP Workbench.

Example

This example of a **ws-roles.xml** file defines two custom roles.

```
<?xml version="1.0" encoding="UTF-8"?>

<roles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="roles.xsd">
    <role id="roleA" defaultName="roleA"
      defaultDescription="Provides access to an extension
       page" />
    <role id="roleB" defaultName="roleB"
      defaultDescription="Provides access to another extension
       page" />
</roles>
```

Role names and descriptions for multiple locales

If you support multiple locales in IAP Workbench, you can optionally specify localized names and descriptions for custom roles.

Localized names are defined in a **names** element within **role** that contains one or more **name** elements. Localized descriptions are defined in a **descriptions** element within **role** that contains one or more **description** elements.

The **name** and **description** elements require a locale attribute whose value is a valid ISO language code.

Example

This example of a **ws-roles.xml** file defines a custom role with separate names and descriptions for English and French.

```
<?xml version="1.0" encoding="UTF-8"?>

<roles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="roles.xsd">
  <role id="localized" defaultName="localized"
    defaultDescription="A role with localized names" >
    <names>
      <name locale="en">localized</name>
      <name locale="fr">localisé</name>
    </names>
    <descriptions>
      <description locale="en">A localized role</description>
      <description locale="fr">Un rôle localisé</description>
    </descriptions>
  </role>
</roles>
```

IAP Workbench checks for a name and description that matches the locale defined in the current installation of IAP Workbench. If no matching localized name or description is found, the defaultName and defaultDescription values are used.

Enabling custom roles in IAP Workbench

To update IAP Workbench to use custom user roles:

1. Make a backup of your Endeca project, including the IAP Workbench customization files (especially **ws-roles.xml**). See “Backing up and restoring an Endeca project” on page 3.
2. Stop the Endeca HTTP service.

3. Navigate to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).
4. Open **ws-roles.xml** in a text editor and add or modify roles as necessary. See “Custom user roles in IAP Workbench” on page 16.
5. Save and close the file.
6. Start the Endeca HTTP service.

IMPORTANT: Deleting a role causes all the user assignments to that role to be deleted across all applications. Modifying the id attribute of a role deletes the original role (and its corresponding user assignments) and creates a new role with the new id. Modifications to any other attributes are saved when you update IAP Workbench and user assignments are preserved. To recover a deleted role along with its user assignments, restore the backups made in Step 1. See “Backing up and restoring an Endeca project” on page 3.

Assigning rule group permissions to IAP Workbench users

Rule group permissions control how IAP Workbench users access rule groups and the rules contained in the groups. An administrator uses IAP Workbench to assign rule group permissions in either of two ways:

- Assign by group on the Rule Group Permissions page
- Assign by user name on the User Management page

There are four permission levels available for rule group access. A user may have one of the following permissions for each rule group:

- Approve—The user has permission to view, edit, and approve rules in the group.
- Edit—The user has permission to view and edit rules in the group but no permission to approve rules.
- View—The user has permission to view rules in the group but no permission to edit or approve rules.
- None—The user has no permission to view, edit, or approve rules in the group. Users with this permission will not see the rule group displayed in IAP Workbench.

Administrators are automatically assigned Approve permissions in all rule groups.

See the *Endeca IAP Workbench Help* for the procedures to assign rule group permissions to IAP Workbench users.

Chapter 3

Managing System Operations with IAP Workbench

This chapter describes the system administration and maintenance tasks available in the Administration section of IAP Workbench.

The chapter includes the following sections:

- About the EAC Administration Console of IAP Workbench
- Provisioning an application using IAP Workbench
- Performing system operations
- Monitoring the system status

About the EAC Administration Console of IAP Workbench

The EAC Administration Console page provides a way for administrators to establish and modify system provisioning, start and stop system components, and run EAC scripts. The Administration Console of IAP Workbench is divided into three sections:

- Hosts – shows a view of your application organized by the hosts you provision. This view indicates the host name, host alias, port and configuration options. You can modify the hosts configuration options, start or stop a component on a host, and see the status of a component on a host.
- Components – shows a view of your application organized by the Endeca components provisioned for an application. You can create components on this tab but not hosts.
- Scripts – show the EAC scripts available to an application and allows you to add, remove, run, and monitor EAC scripts. You can stop and start system operations run by EAC scripts, such as baseline updates.

Provisioning an application using IAP Workbench

Provisioning is the task of defining the location and configuration of the Endeca resources (such as Forge and the MDEX Engine) that control your Endeca application. You can use EAC Administration Console page to provision an Endeca application.

Note: *Endeca recommends using the Endeca Deployment Template to provision your application. For more information about the Endeca Deployment Template, see NEED_NEW_REFERENCE_HERE.*

To provision an application:

1. Using a Web browser, log in to Endeca IAP Workbench, as described in “About the EAC Administration Console of IAP Workbench” on page 22.
2. Add one or more applications. The procedure to add or remove an application is described in the Endeca IAP Workbench Help.

3. Add one or more hosts. The procedure to add or remove a host is described in the Endeca IAP Workbench Help.
4. Add Endeca components to the hosts. You should set one Forge, and at least one Indexer (Dgidx) and one MDEX Engine (Dgraph). The procedure to add components is described in the Endeca IAP Workbench Help.
5. Add EAC scripts. The procedure to add EAC scripts is described in the Endeca IAP Workbench Help.

Disabling the admin role from modifying provisioning information

By default, the admin role allows an administrator to modify provisioning information on the EAC Admin Console page. If necessary, you can disable the admin role from modifying provisioning information.

To disable the admin role from modifying provisioning:

1. Stop the Endeca HTTP service.
2. Navigate to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).
3. Open **webstudio.properties** in a text editor.
4. Change the `com.endeca.webstudio.allow.eac.provisioning` property from true to false as shown:

   ```
   com.endeca.webstudio.allow.eac.provisioning=false
   ```
5. Save and close the file.
6. Start the Endeca HTTP service.

Performing system operations

System operations include running updates, starting and stopping Endeca components, backing up projects, and so on. The Scripts tab of the EAC Administration Console is where you run baseline updates that control Endeca components such as Forge, Dgidx, the MDEX Engine (both Dgraph and Agraph), the Report Generator, and the Log Server. The Hosts and Components tab is where you run individual Endeca components.

For information on other system operations, such as transferring your instance configuration from staging to production environment, and using the **emgr_update** utility, see NEED_NEW_REFERENCE_HERE.

Running a baseline update from IAP Workbench

A baseline update completely rebuilds the Endeca application, including running Forge on the source data, running Dgidx to produce the Endeca records and indices, and starting one or more MDEX Engines with the new indices. See the Endeca IAP Workbench Help for the process to run a baseline update using your EAC script.

Starting and stopping the MDEX Engine

If the status of the MDEX Engine is Running, the Start link (next to the MDEX Engine label) is disabled and only the Stop link is available. If the status is Stopped, only the Start link can be used.

When you start an MDEX Engine, it starts with any options that you specified in Arguments field of component configuration. See the Endeca IAP Workbench Help for the process to start and stop the MDEX Engine.

Starting and stopping the Log Server

If the status of the Log Server is Running, the Start link (next to the Log Server label) is disabled and only the Stop link is available. If the status is Pending or Stopped, only the Start link can be used. See the Endeca IAP Workbench Help for the process to start and stop the Log Server.

Rolling Log Server logs

From the Administration page, you cannot roll the logs created by the Log Server. However, you can roll the logs with this URL command:

```
http://logserverhost:logserverport/roll
```

For example, this command:

```
    http://web002:8002/roll
```

rolls the Log Server that is running on port 8002 on the host named **web002**.

Monitoring the system status

Each host and component that you provision in an Endeca application displays its system status on the EAC Administration Console page of IAP Workbench. IAP Workbench displays a summary of the component's status in the collapsed view of the Hosts tab and Components tab. You can access details about each component (except the Log Server, which does not log its own activities) via the status link next to each component. Clicking the status link displays start time, duration (how long the component has been running), and the last time IAP Workbench checked the component's status. With Auto-Refresh selected, IAP Workbench automatically refreshes status at frequent intervals.

Viewing component logs

To view the latest log for an Endeca component (except for the Log Server which does not log its own actions), check the value of **Log File** for a component as indicated on the Components tab of the Administration Console page. Then browse to the **Log File** directory and open the component log.

Refreshing the status information

You can manually refresh the status display by clicking the Refresh Status button. You can also set the page to be refreshed automatically (at a pre-set interval) by checking the Auto Refresh checkbox. This option is useful when a baseline update is in progress and the system state changes frequently. By default, this option is turned off because the overall system state changes infrequently.

Chapter 4

Setting Up the Preview Application for IAP Workbench

This chapter describes how to set up a custom Endeca application so that it functions as the IAP Workbench's preview application. It includes the following sections:

- Preview application overview
- Preview application requirements
- Instrumenting your application
- Configuring the preview application
- Specifying which MDEX Engine to use for preview

Preview application overview

The preview application is the end-user application that displays in the bottom frame of the Rule Manager page in IAP Workbench. Business users search and navigate to specific locations in the preview application that then become the basis for configured dynamic business rules.

It is important to remember that the only purpose of the preview application is to *present the data* that you are changing via the IAP Workbench. It is not necessary for the preview application to be an exact representation of your final front-end application, as long as it is using the correct data. The business logic that is built into IAP Workbench is not tied to the physical representation of the front-end application. It is good practice, however, to make sure that your preview application represents your final application closely enough so that business users know if their changes are correct.

By default, IAP Workbench is configured to use a copy of the JSP reference implementation as the preview application. This chapter describes how to set up your own custom application to be the preview application.

IAP Workbench communicates with the preview application via settings you specify on the Preview App Settings page. The URL Mapping subsection lets you change the default preview application to your own custom preview application.

Note: *The JSP reference implementation that is used as the preview application for IAP Workbench is stored in* ***$ENDECA_ROOT/tools/server/webapps/endeca_jspref*** *(****%ENDECA_ROOT%\tools\server\webapps\endeca_jspref*** *on Windows). Do not confuse this with the regular JSP reference implementation in* ***$ENDECA_REFERENCE_DIR/endeca_jspref*** *(****%ENDECA_REFERENCE_DIR%\endeca_jspref****. on Windows).*

Preview application requirements

In order to use a custom Endeca application as your IAP Workbench preview application, the custom application must meet the following requirements.

Domain

The preview application and IAP Workbench must reside in the same domain (for example, **endeca.com**).

Javascript domain

If IAP Workbench and your custom application do not reside on the same host, you must declare the Javascript domain in two locations inside the custom application's code:

- Navigation results page (the page that shows the set of records that correspond to a user's query).
- Record page (the page that displays information about a single record).

IAP Workbench communicates with and controls the preview application via Javascript. As a result, both IAP Workbench and the preview application must have the same Javascript **domain** property. The **domain** property provides security for scripts that run in different browser windows but need to communicate with one another.

When you enter the Javascript domain, you can also include the port number of the application server. This will ensure that you are referring to the exact host machine and port number. For example, if the custom application is on an application server running on port 8080, you can enter the Javascript domain as:

```
10.0.0.61:8080
```

or

```
web004:8080
```

The first format uses the host machine's IP address, while the second uses the machine name.

IMPORTANT: In addition, IAP Workbench's Configuration page provides a field where you must enter this information. This is analogous to declaring the `domain` in your Javascript headers.

Embedded hidden form

You must embed small hidden HTML forms on the preview application's navigation results and record pages. The Application Instrumentation Library offers convenient methods to do this. See "Instrumenting your application" on page 30 for more information.

No frames

The preview application must not use frames, because they are likely to collide with the frames of IAP Workbench itself.

URL-based state

The preview application must use URLs to handle navigation and search requests, as opposed to a hidden cookie or session state. The URLs should allow the substitution of search terms and navigation components. See "Using pre-existing applications" on page 32 for more information.

Cookie name

IAP Workbench uses cookies to maintain a user's session. The name of the session cookie used by IAP Workbench is **ESESSIONID**.

In rare cases it is possible for the cookie name to collide with a cookie of the same name on the same application server. This conflict can occur if you are running your application on an application server on the *same* host as IAP Workbench and using **ESESSIONID** for two purposes. In this situation, a user may have their session unexpectedly terminated. To resolve this issue, you can either run the application on another host (that is, a host other than the one IAP Workbench is on), or customize your application server to use a different cookie name (other than **ESESSIONID**) through custom directives on the specific application server.

Note: *If your application does not meet the above requirements, Endeca recommends that you use the default JSP reference implementation available in IAP Workbench.*

Instrumenting your application

To use a custom application as the preview application in IAP Workbench, you must embed small, hidden HTML forms in two places within the preview application pages:

- Navigation results page (the page that shows the set of records that correspond to a user's query).
- Record page (the page that displays information about a single record).

Endeca provides an Application Instrumentation Library with convenient methods to do this. The Application Instrumentation Library is a simple library, consisting of two functions, one for the navigation results page and one for the record page. A version is provided for both supported languages—Java and .NET.

Instrumenting the navigation results page

You use the **htmlInstrumentNavigation()** function to instrument the navigation results page.

The following is a Java example of this function:

```
<%
  ETInstrumentor eti = new ETInstrumentor();
%>
<%= eti.htmlInstrumentNavigation(nav) %>
```

The following is a .NET example in C#:

```
ETInstrumentor eti = new ETInstrumentor();
eti.htmlInstrumentNavigation(nav);
```

In the examples, *nav* is the Navigation object for the page.

The code above produces an HTML form that looks similar to this example:

```
<form name="eti_navigation">
  <input type="hidden" name="nav" value="0">
  <input type="hidden" name="srchTerms" value="é">
  <input type="hidden" name="srchKey" value="Wine Types">
</form>
```

Instrumenting the record page

You use the **htmlInstrumentRecord()** function to instrument the record page.

The following is a Java example of this function:

```
<%
  ETInstrumentor eti = new ETInstrumentor();
  eti.htmlInstrumentRecord(rec, "NameProp",
    "UniqueProp") %>
```

The following is a .NET example in C#:

```
ETInstrumentor eti = new ETInstrumentor();
eti.htmlInstrumentRecord(rec, "NameProp", "UniqueProp");
```

In the examples, *rec* is the ID of the Endeca record displayed on the page, *NameProp* is the name of the property that represents the record's name, and *UniqueProp* is the name of the property that uniquely identifies the record. (*UniqueProp* is typically the Record spec property that you set in Developer Studio > Properties view > Property editor > General tab.)

The code above produces an HTML form that looks similar to this example:

```
<form name="eti_recordeti_navigation">
  <input type="hidden" name="displayName" value="Mustilli, Non-Vintage">
  <input type="hidden" name="recordSpecKey" value="WineID">
  <input type="hidden" name="recordSpecValue" value="1">
</form>
```

Configuring the preview application

After instrumenting your custom application, you must provide URL mappings on the Preview App Settings page of Endeca IAP Workbench. For the procedure on adding URL mappings on the Preview App Settings page, see the *Endeca IAP Workbench Help*.

Using pre-existing applications

If you are using a pre-existing application that uses parameters other than the standard Endeca parameters (**N**, **Ntk**, **Ntt**, **Nmpt**, **Nmrf**, and **R**) as your preview application, you can still map the URLs.

There are two requirements:

- The URLs must contain parameters that map to navigation, search key, and search term parameters.
- The navigation, search key, search term parameters, record ID, preview time, and rule filter parameters must use the same encoding as the standard Endeca **N**, **Ntk**, **Ntt**, **Nmpt**, and **Nmrf** parameters, respectively.

Enabling and disabling the display of the preview application

By default, the URL mappings are filled in with URLs for the preview application of the JSP reference implementation. This enables IAP Workbench to display the preview application for the JSP reference implementation.

If you clear out the default URL settings, the preview application does not display, and the preview-related options, such as Show in Preview, do not appear in the Rule List page of the Rule Manager, in IAP Workbench.

If the display of the preview application is disabled because you previously removed the settings for the URL mappings, you may enable it again.

To enable the display of the preview application, you can use either of the two options:

- Enter the URLs for the preview application of the reference implementation (these URLs originally were filled in as default settings), *or*
- Enter the URL settings for your own application.

For information on the default URL settings used for the JSP reference implementation, see the *Endeca IAP Workbench Help*.

Specifying which MDEX Engine to use for preview

By default, IAP Workbench queries the first MDEX Engine returned by the EAC to update the status messages in the Rule List when in preview mode. Beginning in 5.1.5, you can now designate a specific MDEX Engine that IAP Workbench uses for preview status messages.

You designate a specific Dgraph or Agraph to use for preview status messages by defining a custom EAC property of **WebStudioMDEX** set to **true** on the appropriate component. You can do this using one of the following methods:

- Specify the property on the Dgraph or Agraph component in the EAC provisioning file. For details, see the *Endeca EAC Guide*.

- Specify the property on the Dgraph or Agraph component using the EAC Admin Console in IAP Workbench. For details, see the *Endeca IAP Workbench Help*.

If the property is set to **true** for more than one MDEX Engine component, IAP Workbench uses the first component with the property that is returned by the EAC.

To ensure that the preview application and the status messages are in sync, the preview application should also be configured to point to the same MDEX Engine using the **eneHost** and **enePort** parameters within the preview application URL.

Note that you can use this property in combination with the **WebStudioSkipConfigUpdate** property on other Dgraphs in your application. In this case:

- If you set the **WebStudioMDEX** property on a Dgraph, ensure that it does not also have the **WebStudioSkipConfigUpdate** property set to **true**.
- If you set the **WebStudioMDEX** property on an Agraph, ensure that none of its child Dgraphs have the **WebStudioSkipConfigUpdate** property set to **true**.

For more information about the **WebStudioSkipConfigUpdate** property, see “Specifying which Dgraphs to update with configuration changes” on page 8.

Chapter 5

LDAP Integration with IAP Workbench

This chapter NEEDS BETTER DESC

The chapter includes the following sections:

- LDAP integration with IAP Workbench

LDAP integration with IAP Workbench

If you have IAP Workbench configured to use LDAP for user authentication, an administrator can create a user profile in IAP Workbench that is associated with a user in an LDAP directory. LDAP integration also allows you to assign roles and permissions across an entire LDAP group rather than configuring each user individually.

For users who are configured in IAP Workbench to authenticate via LDAP, the password and identity information such as name and email address are maintained in the LDAP directory. IAP Workbench does not write any data to the LDAP directory. Any roles and permissions assigned to an LDAP user profile in IAP Workbench are stored in the IAP Workbench database.

LDAP user and group profiles can be used in combination with the traditional IAP Workbench user profiles that an administrator configures manually. Users can authenticate via either method on the same instance of IAP Workbench and in the same application.

Optionally, you can enable SSL for communication between IAP Workbench and your LDAP server. For more information on using LDAP with SSL, see the *Endeca Security Guide*.

IAP Workbench supports integration with LDAP servers that comply with LDAP version 3.

Authentication of users in IAP Workbench with LDAP enabled

User authentication via LDAP can be used in combination with the traditional method of authentication for users that are configured manually in IAP Workbench.

IAP Workbench follows this order of events when a user attempts to log in:

1. IAP Workbench checks whether the user name matches the name of any manually configured IAP Workbench user profile in the current application. If such a user exists, IAP Workbench attempts to authenticate the user against the password stored in the IAP Workbench user profile.
2. If no manually configured user of that name exists, IAP Workbench attempts to authenticate the user against the LDAP directory.

3. If the user also has a profile configured as an LDAP user in IAP Workbench, then any associated roles and permissions are applied. If the user is an administrator or if the user profile has the Override LDAP Group Permissions option selected, then the user enters IAP Workbench with the roles and permissions specified in the user profile.
4. Otherwise, IAP Workbench checks the LDAP directory for any groups of which the user is a member. If any of these groups have a profile configured in IAP Workbench, then any roles and permissions associated with all the groups are applied to the user. For more details about inheritance of LDAP group roles and permissions, see “Roles and permissions for LDAP users and groups” on page 38.

User profiles for LDAP users and groups

If LDAP authentication is enabled for IAP Workbench, you have the option of creating user profiles in IAP Workbench for individual users or groups managed in an LDAP directory. For more information on creating user profiles in IAP Workbench, see the *Endeca IAP Workbench Help*.

A user profile is uniquely identified in IAP Workbench by the combination of the user name and user type (IAP Workbench user, LDAP user, or LDAP group) in each application. Administrators are uniquely identified by the combination of user name and user type across all of IAP Workbench. In the case that an LDAP directory defines a user and a group with the same name, this allows profiles to exist in IAP Workbench for both the user and the group. Once a user profile is created and saved in IAP Workbench, the user type cannot be changed.

Note that because of the order in which IAP Workbench handles logins, a IAP Workbench user always takes precedence over an LDAP user. For example, if there is a manually configured IAP Workbench user named lsmith, a user with the name lsmith in the LDAP directory will not be able to log in with the credentials stored in LDAP, even if there is a user profile in IAP Workbench for lsmith as an LDAP user or as a member of an LDAP group.

However, there is no conflict between manually configured users and LDAP groups. For example, if IAP Workbench has a manually configured user named Marketing, and also has a profile for an LDAP group named Marketing, members of the Marketing group in LDAP are able to log in to IAP Workbench as long as there are no conflicts between the LDAP user name and the name of a manually configured IAP Workbench user. (For example, if one of the users in the group has the name Marketing in the LDAP directory, the IAP Workbench user named Marketing will still take precedence.)

Roles and permissions for LDAP users and groups

The user profiles you create in IAP Workbench allow you to assign roles and rule group permissions to an LDAP user or group. Users that exist in the LDAP directory but do not have a profile and associated roles specified in IAP Workbench, either as an individual or as a member of an LDAP group, cannot log in to IAP Workbench.

A user who authenticates via LDAP is assigned the union of all roles associated with all groups of which that user is a member. For each rule group in the application, a user who is a member of multiple LDAP groups defined in IAP Workbench is assigned the broadest permission associated with any of the LDAP groups of which that user is a member.

If you create an LDAP user profile in IAP Workbench for an individual who is also a member of one or more LDAP groups defined in IAP Workbench, that user is assigned any roles you specify on the User Management page in addition to any roles that the user inherits from membership in LDAP groups. If you specify rule group permissions for an LDAP user who is also a member of an LDAP group, then for each rule group, the user is assigned either the permission specified on the User Management page or the broadest permission associated with any of the user's LDAP groups, whichever is broader. You can override this behavior by specifying Override LDAP Group Permissions when creating the profile in IAP Workbench. If you select this option, the user is assigned only the roles and permissions you specify in the user profile, and does not inherit any roles or permissions from LDAP groups.

Administrators in IAP Workbench with LDAP

If you have LDAP enabled, you can create profiles for both LDAP users and LDAP groups as administrators in IAP Workbench. Note that the same precedence rules apply when logging in to IAP Workbench as for non-administrators, so that if a manually configured user profile exists for either an administrator or non-administrator in IAP Workbench, a user will not be able to log in via LDAP with the same user name.

Note that administrators can delete other administrators, including the predefined admin user, but there must be at least one manually configured IAP Workbench administrator. This is to ensure that changes to the LDAP directory or disabling of LDAP authentication for IAP Workbench cannot disable all administrator logins.

Workflow notifications for LDAP users and groups

For users who authenticate via LDAP, IAP Workbench uses the email address that is stored in the LDAP directory for workflow notification messages.

Whenever a rule is modified, IAP Workbench saves the user name of the editor who modified the rule for notification purposes, whether the user is defined as an IAP Workbench user, an LDAP user, or a member of an LDAP group. When workflow notifications are sent out, IAP Workbench looks up the user’s email address in the user profile or in the LDAP directory as appropriate. If an approver for a rule group is an LDAP group, then IAP Workbench attempts to find an email address associated with the group in LDAP.

When a user changes a rule’s workflow state (by activating, deactivating, requesting activation, requesting deactivation, cancelling a request for a rule, or rejecting a request) and clicks Save Changes, IAP Workbench writes a message to the log similar to the following:

```
   INFO: User mmartin made a workflow state change.
   INFO: Email addresses were retrieved for the following users or
groups: Web Studio User batkins, LDAP User lsmith
   INFO: Email addresses could not be found for the following users
or groups: LDAP Group rule_approvers, Web Studio User admin
```

This information is only captured in the log; the user in IAP Workbench will not see any message about whether email addresses could be found.

Because IAP Workbench launches another application to send the email and the user can edit the list of recipients before sending the message, the IAP Workbench log cannot record whether an email was sent, or the actual recipients of the message.

Enabling LDAP authentication in IAP Workbench

Because LDAP configuration is unique to each LDAP server and directory, enabling LDAP authentication for IAP Workbench is a manual process.

To enable LDAP authentication in IAP Workbench:

1 Stop the Endeca HTTP service.

2 Navigate to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).

3 Open the **webstudio.properties** file, and locate the **com.endeca.webstudio.useLdap** property, for example:

```
# LDAP Authentication
com.endeca.webstudio.useLdap=false
```

4 Change the value of the property to **true**, for example:

```
com.endeca.webstudio.useLdap=true
```

5 Save and close the file.

6 Open the **Login.conf** file. This file contains a sample configuration for LDAP authentication.

Note: *By default, IAP Workbench uses the authentication profile in this location. You can specify an alternate configuration file. For more information, see "Specifying the location of the configuration file" on page 41.*

7 Uncomment and modify the **Webstudio** profile according to your LDAP configuration. For details about profile parameters, see "Configuration of the Webstudio login profile for LDAP" on page 41.

8 Save and close the file.

9 Start the Endeca HTTP service.

Disabling LDAP authentication for IAP Workbench

If you disable LDAP authentication for IAP Workbench by setting the property **com.endeca.webstudio.useLdap=false** in the **webstudio.properties** file, the options to create a user profile for an LDAP user or an LDAP group do not display in IAP Workbench. All new user profiles you create must be manually configured in IAP Workbench. Any users who were configured as LDAP users or as members of an LDAP group are no longer able to log in to IAP Workbench. Although they are inactive, any existing user profiles for LDAP users or LDAP groups remain in IAP Workbench and can be edited by an administrator.

Configuration of the Webstudio login profile for LDAP

IAP Workbench uses the Java Authentication and Authorization Service (JAAS) to authenticate users against an LDAP directory. The configuration information that IAP Workbench uses for LDAP authentication is stored in a profile named Webstudio in **%ENDECA_CONF%\conf\Login.conf** (on Windows) or **$ENDECA_CONF/conf/Login.conf** (on UNIX). A sample profile is included in this location by default, but you should modify its parameters as needed for your LDAP configuration. You can also specify an alternate location for the configuration file.

If you want to configure JAAS authentication for other applications running in the Endeca HTTP service, for example, for the Standard Application or your own Endeca implementation, create additional profiles with unique names in this same **Login.conf** file. For more information on configuring JAAS authentication for your Endeca application using LDAP or a local password file, see the *Endeca Security Guide*.

Note: *A **Login.conf** file exists in **%ENDECA_CONF%\etc** (on Windows) and **$ENDECA_CONF/etc** (on UNIX). This file contains a sample profile for file-based authentication and is not used by IAP Workbench.*

Specifying the location of the configuration file

By default, IAP Workbench uses **%ENDECA_CONF%\conf\Login.conf** (on Windows) or **$ENDECA_CONF/conf/Login.conf** (on UNIX) as its configuration file. You can substitute any configuration file that includes a JAAS profile named Webstudio. The file does not have to be named **Login.conf**, but it should be saved in UTF-8 format.

If you want to store the configuration file in a different location, you can pass this location to the Java JVM. How you specify the location depends on how you run the Endeca HTTP service.

If you are running the Endeca HTTP service as a Windows service:

1 Open the Registry Editor and look for the following key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\
  Procrun 2.0\EndecaHTTPservice\Parameters\Java\Options
```

Note: *In the Registry Editor Explorer pane, expand the folders until you reach Java. Then click on the Java folder and look for the Options setting in the right pane.*

2 Right click **Options**.

3 Choose **Modify**. The **Edit Multi-String** dialog box displays.

4 Locate the following parameter:

```
-Djava.security.auth.login.config=%ENDECA_CONF%/conf/Login.conf
```

5 Change the path to point to the location of your configuration file.

If you are running the Endeca HTTP service on Windows from the command line:

1 Navigate to the **%ENDECA_ROOT%\tools\server\bin** directory.

2 Open the **setenv.bat** file.

3 Locate the line that begins with **set JAVA_OPTS**, for example:

```
    set JAVA_OPTS=-Xmx1024m -XX:MaxPermSize=128m
-Djava.security.auth.login.config=%ENDECA_CONF%/conf/Login.conf
```

4 Change the path of the **-Djava.security.auth.login.config** parameter to point to the location of your configuration file.

If you are running the Endeca HTTP service on UNIX:

1 Navigate to the **$ENDECA_ROOT/tools/server/bin** directory.

2 Open the **setenv.sh** file.

3 Locate the line that begins with **JAVA_OPTS=**, for example:

```
    JAVA_OPTS="-Xmx1024m -XX:MaxPermSize=128m
-Djava.security.auth.login.config=$ENDECA_CONF/conf/Login.conf"
```

4 Change the path of the **-Djava.security.auth.login.config** parameter to point to the location of your configuration file.

Templates used in the Webstudio profile

IAP Workbench allows templates to be supplied for certain configuration parameters in the Webstudio JAAS profile. These templates, indicated by %{} escapes, allow values from the authentication operation, such as a user or group name entered in IAP Workbench, or specific values from the user or group objects in LDAP, to be substituted into the parameter value. Templates also allow you to extract information from the LDAP user or group object such as the exact user or group name as specified in the LDAP directory, or identity information that is stored in LDAP. The %{} escapes are expanded as follows:

Escape	Description
%{#username}	The name of the LDAP user as entered in the Add User page in IAP Workbench, or the user name entered by a user at the IAP Workbench Login page.
%{#groupname}	The name of the LDAP group as entered in the Add User page in IAP Workbench.
%{#dn}	The distinguished name of the user or group object in the LDAP directory.
%{#dn:*n*}	The value of the path field at index *n* in the distinguished name of the user or group object in LDAP. For example, if the value in the **%{#dn}** field is **cn=joe,ou=People,dc=foo,dc=com**, then the value “People” will be substituted for %{#dn:1}, while “joe” will be substituted for %{#dn:0}. Note that unlike the value of %{#dn}, which is the raw value returned from the LDAP server, the values returned by this template are not LDAP escaped.
%{#*fieldname*}	The value in the specified field of the user object (or group object when used in the **groupTemplate** or **findGroupTemplate** parameter) under consideration.

Configuration parameters for the Webstudio profile

You specify the values of configuration parameters for LDAP authentication as quoted strings. If there are any quotation marks (**"**) or backslashes (****)

in the string, they must be escaped. For example, if you have the following string:

```
"A string with an "embedded quote" and a \backslash"
```

In the profile, it should be specified as follows:

```
"A string with an \"embedded quote\" and a \\backslash"
```

For most parameter values, single quotation marks (**'**) do not need to be escaped and the values you specify for the parameters can include UTF-8 characters. For additional restrictions on the userPath, groupPath, and findGroupPath parameters, see “LDAP path parameters” on page 48.

The following parameters can be specified in the profile:

Parameter	Description
serverInfo	A URL specifying the name and port of the LDAP server to be used for authentication. You can specify multiple LDAP servers.
userPath	The query that is passed to the LDAP server to find an individual user. You can use the **%{#username}** template to insert the name entered in the Add User page (when using the Check Name function), or in the IAP Workbench Login page, into the query.
userTemplate	A template that specifies how to produce the username from the user object returned by the **userPath** query. This template allows IAP Workbench to automatically correct the case (capital or lowercase) of the username to match the name exactly as specified in the LDAP directory. The correction occurs when you add an LDAP user to IAP Workbench. Therefore, the value returned by this template should match the name entered on the Add User page, except for possible differences in case.

Parameter	Description
groupPath	The query that is passed to the LDAP server to find all the groups of which a user is a member. This query is executed when a user logs in to IAP Workbench after looking up the user with the **userPath** query. Thus, you can use templates to insert any information from the user object that is returned by the previous query, such as the distinguished name of the user or any other LDAP attributes, into the groupPath query. You can specify multiple values for groupPath.
groupTemplate	A template that specifies how to produce individual group names from the set of groups returned by the **groupPath** query. The value returned by this template should match the name of the LDAP group as defined in the IAP Workbench user profile. You can specify multiple values for groupTemplate.
findGroupPath	The query that is passed to the LDAP server to find a specific group. You can use the **%{#groupname}** template to insert the name of the group as entered in the Add User page into the query.
findgroupTemplate	A template that specifies how to produce the group name from the group object returned by the **findGroupPath** query. Like the userTemplate, this template is used to correct the case of a group name when you add LDAP group profiles in IAP Workbench. Therefore, the value returned by this template should match the name entered on the Add User page, except for possible differences in case.

Parameter	Description

serviceUsername	The user name of an administrator login to the LDAP server specified in the **serverInfo** parameter. For example:
	`"Manager@example.com"`
	or
	`"cn=Manager,dc=example,dc=com"`
	If no value is specified for this option, IAP Workbench attempts to authenticate anonymously.

servicePassword	The password to use in conjunction with the **serviceUsername** value.
serviceAuthentication	Specifies the method of authentication that should be used in connecting to the LDAP server as the administrator account. The permitted values are **none**, **simple**, or **EXTERNAL**.
authentication	Specifies the method of authentication that should be used in rebinding to the LDAP server as a user account. The permitted values are **none**, **simple**, or **EXTERNAL**.
ldapBindAuthentication	Optional. By default this is set to **true**, and IAP Workbench authenticates users by rebinding as the user to the LDAP system, thereby employing the LDAP system's own authentication mechanism.
loginName	Optional. A template login name that will be used to rebind to the LDAP server if **ldapBindAuthentication** is **true**. Default value is **%{dn}**.
passwordAttribute	Optional. The name of the attribute on the user object that contains the user's password. Used only if **ldapBindAuthentication** is set to **false**. The field specified must contain the user's password in clear text. By default this is set to **userPassword**.

Parameter	Description
checkPasswords	Optional. Determines whether IAP Workbench checks passwords during logins. Default value is **true**. If set to **false**, IAP Workbench uses only the user name to authenticate from the LDAP directory.
useSSL	Optional. Default value is **false**. If set to **true**, IAP Workbench will make mutually authenticated SSL connections to the LDAP server. If you set the parameter, ensure that you have configured the LDAP server to use SSL and that the value of **serverInfo** has the protocol specified as **ldaps://** with an SSL port.
keyStoreLocation	Used only if **useSSL=true**. The location of the Java keystore, which stores keys and certificates. The keystore is where Java gets the certificates to be presented for authentication. The location of the keystore is OS-dependant, but is often stored in a file named **.keystore** in the user's home directory. ***Note:*** *Even if this location is on a Windows system, the path uses forward slashes, (**/**) not backslashes (\).*
keyStorePassphrase	Used only if **useSSL=true**. The passphrase used to open the keystore file.

Configuration parameters for identity information stored in LDAP

IAP Workbench does not store any identity information such as first name, last name, or email address for LDAP users or groups. Instead, IAP Workbench looks up this information in the LDAP directory when needed, for example, when sending workflow email notifications. The LDAP configuration profile allows you to specify templates to extract identity information from LDAP user or group objects, but they are not required for authentication via LDAP.

Parameter	Description
firstNameTemplate	A template that specifies how to produce the user's first name from the user object, for example, **%{#*firstNameAttribute*}**.
lastNameTemplate	A template that specifies how to produce the user's last name from the user object, for example, **%{#*lastNameAttribute*}**.
emailTemplate	A template that specifies how to produce the user's email address from the user object, for example, **%{#*emailAttribute*}**, or **%{*usernameField*}@*companydomain.com***.
groupEmailTemplate	A template that specifies how to produce the email address associated with a group in LDAP from the group object. This information is used for workflow notifications in the case where an LDAP group is specified as an approver for a rule group.

IAP Workbench looks up the identity information for a user or group when you use the Check Name function on the Add User page to confirm that you are adding the correct LDAP user or group. If you do not specify templates for retrieving identity information, the fields are not filled in when you use Check Name and workflow email notifications are not automatically addressed to the approvers and editors related to a rule.

LDAP path parameters

The userPath, groupPath, and findGroupPath parameters, when appended to the URL in the **serverInfo** parameter, must conform to RFC 2255. This means that certain characters must be encoded in order for the path parameters to form a valid LDAP URL when appended to the value of the serverInfo parameter. Both LDAP and URL encoding may apply to these strings depending on your data. If possible, verify the URL by passing it to your LDAP server before specifying it in the configuration for IAP Workbench.

LDAP encoding affects reserved characters such as the comma (**,**), equals sign (**=**), and question mark (**?**). These characters must be escaped by prepending a backslash (\) when they are not used for their reserved purpose, for example if they appear within a common name or organizational unit.

URL encoding affects characters that are invalid for URLs, such as non-ASCII characters and any unsafe characters as defined in RFC 1738. This includes reserved LDAP characters when they are not used for their reserved purpose. These characters must be replaced with the % sign followed by the appropriate hex code.

For example, if you have the following string as part of your userPath:

```
ou=Endeca Technologies, Inc.
```

Applying LDAP encoding produces the following result:

```
ou=Endeca Technologies\, Inc.
```

And applying URL encoding to the LDAP-encoded string produces:

```
ou=Endeca%20Technologies%5C%2C%20Inc.
```

Any non-ASCII characters or any other characters that are not valid in an LDAP URL must also be properly encoded in the string that you specify in the Webstudio profile.

Specifying multiple values for parameters in the Webstudio profile

You can specify multiple LDAP servers with multiple instances of the **serverInfo** parameter, by using the format:

```
serverInfo.n = ”ldap://server_url:port_number”
```

For example:

```
serverInfo.0="ldap://web01.endeca.com:1234"
serverInfo.1="ldap://web02.endeca.com:1230"
serverInfo.2="ldap://web03.endeca.com:1334"
```

If you specify multiple LDAP servers, the servers are assumed to be equivalent.

The choice of which LDAP server to contact is made randomly. If an LDAP server cannot be reached, the LoginModule plug-in proceeds through the remaining servers in order of configuration, wrapping if necessary. For example, if five servers are configured and Server 3 is the first to be contacted, the remaining order of contact is Server 4, Server 5, Server 1, and finally Server 2.

You can also specify multiple values for the **groupPath** attribute by using the same format, for example:

```
    groupPath.0="/ou=groups,dc=endeca,dc=com??sub?(member=%{#dn})"
    groupPath.1="/dc=endeca,dc=com?memberOf?sub?(AccountName=%{#use
rname})"
```

If you specify more than one **groupPath**, IAP Workbench sends all the queries to the LDAP server to discover the groups of which a user is a member.

You can specify corresponding values for **groupTemplate** for each **groupPath**. In this case, the value for **groupTemplate.0** is applied to the results of the **groupPath.0** query, **groupTemplate.1** is applied to the results of **groupPath.1**, and so on.

Troubleshooting user authentication in IAP Workbench with LDAP enabled

If a user cannot log in to IAP Workbench, one of the following messages displays:

Incorrect Username or Password

If the user is entering the correct LDAP user name and password, there may be a manually configured IAP Workbench user in the same application with the same user name or a IAP Workbench administrator with the same user name.

A user with a manually configured profile always takes precedence over a user authenticating via LDAP. For more details about the behavior of users with the same name, see “User profiles for LDAP users and groups” on page 37.

You have no roles to access IAP Workbench

This can mean that a profile was created for this user, or for a group of which this user is a member, that was not assigned any roles. This message also displays when a user who exists in the LDAP directory attempts to log in to IAP Workbench, but no profile exists in IAP Workbench for the user or for any group of which the user is a member. A user who does not have any associated roles cannot log in to IAP Workbench.

An error occurred while trying to validate your credentials

This error displays when any error occurs other than a user name-password mismatch or an absence of roles. It can indicate anything from a connectivity issue with the LDAP server to a mistake in the configuration in the LDAP login profile located in **%ENDECA_CONF%\conf\Login.conf** (on Windows) or **$ENDECA_CONF/conf/Login.conf** (on UNIX). For more information about the login profile, see "Configuration of the Webstudio login profile for LDAP" on page 41.

The message "An error occurred while querying the LDAP server" when using the Check Name function in the Add User page indicates the same set of possible issues.

Check the IAP Workbench log, which is located in **%ENDECA_CONF%\logs\webstudio.#.log** (on Windows) or **$ENDECA_CONF/logs/webstudio.#.log** (on UNIX) for more information about the causes of authentication failures. In most cases, the solution is to adjust the LDAP query strings to return the desired results. If possible, test the query URLs against your LDAP server using an indepedent tool in order to confirm that they behave as expected and that each query for a user or group that exists in the directory returns a unique user or group object.

Chapter 6

SSL Configuration

This chapter describes how to configure your IAP Workbench to use SSL for Web browser connections.

This chapter contains the following sections:

- Configuring SSL in Endeca IAP Workbench
- SSL considerations for the preview application and Workbench extensions

Note: *For information about configuring IAP Workbench to use SSL to communicate with the EAC Central Server, see the* Endeca Security Guide*.*

Configuring SSL in Endeca IAP Workbench

SSL is disabled by default for IAP Workbench as a server. To enable SSL security between IAP Workbench and its clients, you need to do the following:

- Enable the SSL version of the IAP Workbench war.
- Set up a certificate for the IAP Workbench server. For details, see the *Endeca Security Guide*. The server certificate for IAP Workbench must be issued to the fully qualified domain name of the server.
- Modify the **server.xml** file for the Endeca Tools Service to enable the HTTPS connector and point to the new keystore.

Clients can make secure connections to IAP Workbench either by taking advantage of a redirect from the non-SSL port or, if you have disabled the non-SSL port or do not wish to use the redirect, by making an HTTPS connection directly to the SSL port.

Enabling the SSL version of IAP Workbench

The non-SSL version of IAP Workbench is installed by default.

To enable the SSL version of IAP Workbench:

1. Stop the Endeca Tools Service.
2. Navigate to **%ENDECA_TOOLS_CONF%\conf\Standalone\localhost** (on Windows) or **$ENDECA_TOOLS_CONF/conf/Standalone/localhost** (on UNIX).
3. Open the **ROOT.xml** file.
4. Locate the line in which the **docBase** is defined, for example:

```
docBase="C:\Endeca\Workbench\version\server/
  ewkbench-1.0.1.123456.war"
```

5. Change this to point to the SSL version of the war, for example:

```
docBase="C:\Endeca\Workbench\version\server/
  ewkbench-1.0.1.123456-ssl.war"
```

6. Save and close the file.

7 Start the Endeca Tools Service.

Note: *If you want to restore the non-SSL version at a later date, you can reverse the process by editing the* ***ROOT.xml*** *file accordingly.*

Modifying the server.xml for the Endeca Tools Service

Before you can use SSL with IAP Workbench, you must edit its **server.xml** file as described below.

This procedure assumes you have already generated server certificates for IAP Workbench as described in the *Endeca Security Guide* and uploaded them to the IAP Workbench server.

To enable the HTTPS connector:

1 Stop the Endeca Tools Service.

2 Navigate to **%ENDECA_TOOLS_CONF%\conf** (on Windows) or **$ENDECA_TOOLS_CONF/conf** (on UNIX).

3 Open the **server.xml** file.

4 Remove the comments around the following lines:

```
<!-- Define a SSL HTTP/1.1 Connector on port 8446 -->
  <!--
  <Connector port="8446" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="conf/eac.ks" keystorePass="eacpass"
    truststoreFile="conf/ca.ks" truststorePass="eacpass"
    URIEncoding="UTF-8"/>
-->
```

5 Optionally, change the port number to something other than 8446 if you do not want to use that default. If you do not use the default port, update the **redirectPort** attribute on the non-SSL HTTP connector to point to the new port.

6 If you want to require users to connect to the SSL port directly over HTTPS, you can comment out the non-SSL connector in the **server.xml** file.

7 To enable mutually authenticated connections, set **clientAuth="true"**. Note that mutually authenticated connections are not recommended for IAP Workbench because they require each user’s browser to be configured with its own private certificates.

8 Update the **keystoreFile**, **keystorePass**, **truststoreFile**, and **truststorePass** with the appropriate values for your certificates.

   The **keystoreFile** and **truststoreFile** values should be the paths to the location where you uploaded your keystore and truststore files. These paths can be specified as absolute paths, or paths relative to ENDECA_TOOLS_CONF, although the files themselves can be located anywhere on the server.

9 Save and close the file.

10 Start the Endeca Tools Service.

SSL considerations for the preview application and Workbench extensions

If you have IAP Workbench running under SSL and your preview application or any Workbench extensions do not use SSL, a browser message displays when you navigate to IAP Workbench warning you that the site contains both secure and nonsecure items. In order to prevent this message from displaying, you must do the following:

- Configure the URLs of the preview application to use HTTPS and an SSL port. For information about specifying the URLs for the preview application, see the *Endeca IAP Workbench Help*.
- Configure any Workbench extensions to use HTTPS and an SSL port. For information about configuring Workbench extensions, see “Workbench extensions” on page 65.

Note that even if you have an internal redirect set up to forward connections from a non-SSL port to a secure SSL port, the browser warning will appear because the initial connection is being made over HTTP.

Configuring the default preview application to require SSL

connections

If you want all communication with the preview application to use SSL, you can add a security constraint to the **web.xml** file. The configuration file for the default preview application is found in **%ENDECA_TOOLS_ROOT%\server\webapps\endeca_jspref\WEB-INF** (on Windows) or **$ENDECA_TOOLS_ROOT/server/webapps/endeca_jspref/WEB-INF** (on UNIX) and includes a commented-out security constraint as follows:

```
   <!--
   Uncomment the following section if you want to force the jsp_ref
app to use ssl
   -->
   <!--
   <security-constraint>
     <web-resource-collection>
       <web-resource-name>Entire Site</web-resource-name>
       <url-pattern>/*</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
     </web-resource-collection>
     <user-data-constraint>
       <transport-guarantee>CONFIDENTIAL</transport-guarantee>
     </user-data-constraint>
   </security-constraint>
   -->
```

To enable this constraint, remove the comments around the **<security-constraint>** element. This forces connections to the application on the non-SSL port to be redirected to the SSL port defined in **server.xml**.

Chapter 7IAP Workbench

Customizing IAP Workbench

This chapter describes how to customize the IAP Workbench interface and how to add extensions to IAP Workbench.

The chapter includes the following sections:

- The navigation menu and launch page
- Workbench extensions

The navigation menu and launch page

You can configure the items in the navigation menu on the left and on the launch page of IAP Workbench by modifying the **ws-mainMenu.xml** file in **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).

By editing **ws-mainMenu.xml**, you can do any of the following:

- Add a new menu item.
- Remove an item from the menu.
- Specify the order in which the menu items display.
- Specify whether an item is in the top-level menu or in a submenu.
- Specify whether a menu item displays on the launch page.

Navigation menu nodes

A menu item is either a *leaf* or a *node*. A node is a top-level menu item that does not link directly to any pages. Instead it has children that are leaf items and are displayed in a submenu. Nodes cannot be displayed on the launch page. Each node is defined in a **menunode** element in **ws-mainMenu.xml** that takes the following attributes:

Attribute name	Attribute value
id	The id of a predefined node in IAP Workbench or a unique string identifying a custom node. For more information on predefined nodes, see “Predefined menu nodes in IAP Workbench” on page 62.
defaultTitle	The display name for this node that appears in the navigation menu. This attribute is required for all custom nodes.

A **menunode** element requires one or more child **menuitem** elements (see “Navigation menu leaf items” on page 62).

Example

This example of a **ws-mainMenu.xml** file defines a custom menu node with extensions as its child items.

```
<?xml version="1.0" encoding="UTF-8"?>

<mainmenu xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mainMenu.xsd">
  <menunode id="myextensions" defaultTitle="My Extensions">
    <menuitem id="extensionA"/>
    <menuitem id="extensionB"/>
  </menunode>
</mainmenu>
```

Node titles for multiple locales

If you customize a menu for multiple locales in IAP Workbench, you can optionally specify localized titles for custom menu nodes in a **titles** element within **menunode** that contains one or more **title** elements. The **title** element requires a locale attribute whose value is a valid ISO language code.

Example

This example of a **ws-mainMenu.xml** file defines a custom menu node with titles in both English and French.

```
<?xml version="1.0" encoding="UTF-8"?>

<mainmenu xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mainMenu.xsd">
  <menunode id="myextensions" defaultTitle="My Extensions">
    <titles>
      <title locale="en">Access Extensions</title>
      <title locale="fr">Accéder aux extensions</title>
    </titles>
    <menuitem id="extensionA"/>
    <menuitem id="extensionB"/>
  </menunode>
</mainmenu>
```

IAP Workbench checks for a title that matches the locale defined in the current installation of IAP Workbench. If no matching localized title is found, the defaultTitle value is used.

Predefined menu nodes in IAP Workbench

There are several predefined menu nodes in IAP Workbench. You can specify the placement of the predefined nodes in the menu and what items display under them, but you cannot modify the titles or specify localized titles.

The predefined nodes in IAP Workbench are as follows:

Node id	Node description
searchConfig	Search Configuration
reporting	View Reports
settings	Application Settings
eacconsole	EAC Administration

Navigation menu leaf items

A leaf is a menu item that links to a page, and that can also have an entry on the launch page. A leaf can be either in the top-level menu or in a submenu as the child of a node. Leaf items cannot have child items. Menu items display in the order in which they are listed in **ws-mainMenu.xml**.

Each leaf in the menu is defined in a **menuitem** element in **ws-mainMenu.xml** that takes the following attributes:

Attribute name	Attribute value	Required?
id	The id of a predefined page in IAP Workbench or the id of an extension as defined in **ws-extensions.xml**. For more information about extensions, see “Workbench extensions” on page 65.	yes
onLaunchPage	If set to **true**, the menu item displays on the launch page in the order in which it is listed in **ws-mainMenu.xml**. Default value is **false**.	no

The predefined pages and their corresponding ids are as follows:

IAP Workbench page	Menu item id
Rule Manager	rules
Keyword Redirects	redirects
Thesaurus	thesaurus
Phrases	phrases
Stop Words	stopwords
Dimension Order	dimorder
Current Report (Daily)	reporting.currentDaily
Current Report (Weekly)	reporting.currentWeekly
Daily Reports	reporting.daily
Weekly Reports	reporting.weekly
EAC Monitor	eacMonitor
User Management	settings.users
Rule Group Permissions	settings.permissions
Resource Locks	settings.locks
Report Generation	settings.reporting
Preview App Settings	settings.previewApp
Instance Configuration	settings.instanceConfig
User Settings (for non-admin users)	userSettings
EAC Admin Console	eacconsole.console
EAC Settings	eacconsole.settings

Example

This example of a **ws-mainMenu.xml** file defines a menu that shows top-level leaf items, items nested within a predefined node, and items nested within a custom node. Items that have **onLaunchPage="true"** display in the launch page regardless of whether they are in the top-level menu or in a submenu.

```
<?xml version="1.0" encoding="UTF-8"?>

<mainmenu xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mainMenu.xsd">
  <menuitem id="rules" onLaunchPage="true"/>
  <menuitem id="redirects" onLaunchPage="true"/>
  <menunode id="searchConfig">
    <menuitem id="thesaurus" onLaunchPage="true"/>
    <menuitem id="phrases"/>
    <menuitem id="stopwords"/>
  </menunode>
  <menunode id="myextensions" defaultTitle="My Extensions">
    <menuitem id="extensionA" onLaunchPage="true"/>
    <menuitem id="extensionB"/>
  </menunode>
</mainmenu>
```

Updating the IAP Workbench menu and launch page

To update the navigation menu and launch page:

1. Stop the Endeca HTTP service.
2. Navigate to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).
3. Open **ws-mainMenu.xml** in a text editor and add or modify menu items as necessary. See “The navigation menu and launch page” on page 60.
4. Save and close the file.
5. Start the Endeca HTTP service.

Workbench extensions

Extensions enable you to incorporate web applications related to your Endeca implementation as plug-ins to IAP Workbench. An extension can be as simple as a static web page or it can provide sophisticated functionality to control, monitor, and configure your Endeca applications. Extensions can be hosted on the same server as IAP Workbench or on another server.

IAP Workbench provides the ability to customize the navigation menu and launch page to include links to extensions. The extension itself is presented within an iFrame in IAP Workbench and can be themed to inherit the look and feel of the IAP Workbench interface. The extensibility framework allows extensions to leverage IAP Workbench user authentication and role-based permissions. IAP Workbench can also pass information to extensions, such as the EAC Host and applications that it is connected to.

Configuration of extensions in IAP Workbench

Extensions are defined in the **ws-extensions.xml** file in **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).

The default **ws-extensions.xml** file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<extensions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="extensions.xsd">

</extensions>
```

Each extension is defined in an **extension** element within **extensions**. You can specify as many additional extensions as you need by adding more **extension** elements.

The **extension** element takes the following attributes:

Attribute name	Attribute value	Required?
id	A unique string identifying this extension. Do not define an extension with the same id as one of the predefined IAP Workbench pages. For a list of predefined IAP Workbench pages and their ids, see the table in “Navigation menu leaf items” on page 62.	yes
defaultName	The display name for this extension that appears in the navigation menu and launch page in IAP Workbench.	yes
defaultDescription	A brief description of this extension that appears on the launch page in IAP Workbench.	yes
url	The fully specified URL to this extension. The extension must be a Web application reachable through HTTP or HTTPS, but it does not have to run on the same server as IAP Workbench.	yes
launchImageUrl	The fully specified URL to a custom image for this extension’s entry on the launch page.	no
role	The id of the role that is allowed to access this extension. This can be one of the predefined IAP Workbench user roles, or any custom role. For more information on user roles, see “IAP Workbench user roles” on page 15. Each extension can have a maximum of one role, although a single role can allow access to many extensions. If no role is specified, the extension is available to all IAP Workbench users.	no
height	The height in pixels of the frame in which the extension is displayed. The default value is 500 pixels.	no
sharedSecret	A shared key that IAP Workbench uses to calculate the authentication token. For more information on the authentication token, see “Token-based authentication for Workbench extensions” on page 70.	no

Example

This example of a **ws-extensions.xml** file defines a simple extension that enables a link to the Endeca Web site for all admin users.

```
<?xml version="1.0" encoding="UTF-8"?>

<extensions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="extensions.xsd">
  <extension id="endecaHome"
      defaultName="Endeca home page"
      defaultDescription="Visit the Endeca home page"
      url="http://www.endeca.com"
      role="admin" />
</extensions>
```

Extension names and descriptions for multiple locales

If you deploy extensions to multiple locales in IAP Workbench, you can optionally specify localized names and descriptions for extensions.

You can define localized names in a **names** element within **extension** that contains one or more **name** elements. You can define localized descriptions in a **descriptions** element within **extension** that contains one or more **description** elements.

The **name** and **description** elements require a locale attribute whose value is a valid ISO language code.

Example

This example of a **ws-extensions.xml** file defines an extension with separate names and descriptions for English and French.

```
<?xml version="1.0" encoding="UTF-8"?>

<extensions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="extensions.xsd">
  <extension id="endecaHome"
      defaultName="Endeca home page"
      defaultDescription="Visit the Endeca home page"
      url="http://www.endeca.com"
      role="admin">
    <names>
      <name locale="en">The Endeca Web site</name>
      <name locale="fr">La page d'accueil d'Endeca</name>
    </names>
    <descriptions>
      <description locale="en">Link to the Endeca Web site</description>
      <description locale="fr">Lien vers la page Web d’Endeca</description>
    </descriptions>
  </extension>
</extensions>
```

IAP Workbench checks for a name and description that matches the locale defined in the current installation of IAP Workbench. If no matching localized name or description is found, the defaultName and defaultDescription values are used.

Enabling extensions in IAP Workbench

To enable extensions in IAP Workbench:

1 Stop the Endeca HTTP service.

2 Navigate to **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX).

3 Open **ws-extensions.xml** in a text editor and add or modify extensions as necessary (see “Configuration of extensions in IAP Workbench” on page 65).

Note: *To enable or disable links to your extensions in the navigation menu and the launch page, see “Updating the IAP Workbench menu and launch page” on page 64.*

4 Save and close the file.

5 Start the Endeca HTTP service.

URL tokens and Workbench extensions

IAP Workbench can pass information to an extension through URL tokens in order to enable the extension to authenticate users, connect to the EAC Central Server, and maintain its state if a user navigates away from the extension and back again during the same session.

The following tokens are available to pass to extensions:

Token ID	Token description
${AUTH}	An MD5 hash value used to authenticate users coming from IAP Workbench. For more information on the authentication token, see "Token-based authentication for Workbench extensions" on page 70.
${EAC_APP}	The name of the application that the IAP Workbench user is logged in to.
${EAC_HOST}	The host running the EAC Central Server to which IAP Workbench is currently connected.
${EAC_PORT}	The port on the EAC host through which IAP Workbench and the EAC Central Server communicate.
${EXTENSION_ID}	The id of the extension as defined in **ws-extensions.xml**.
${LOCALE}	The locale of IAP Workbench; this is the value of the com.endeca.webstudio.locale property in **%ENDECA_CONF%\conf\webstudio.properties**.
${TS}	The time, in milliseconds since 00:00:00 UTC January 1, 1970, when the user navigates to the extension.
${USERNAME}	The username of the IAP Workbench user accessing the extension.
${WEBSTUDIO_SESSIONID}	The id of the user's current IAP Workbench session. The extension can use this in combination with the ${USERNAME} token to maintain the state of the extension throughout a single IAP Workbench session, for instance by storing the information in a cookie.

You use these tokens by specifying them in the **url** attribute of the extension definition in **%ENDECA_CONF%\conf\ws-extensions.xml**. The name of the URL parameter does not have to match the id of the token as listed in the preceding table.

For example, the following extension definition creates a URL that passes the EAC host, port, and application to the extension:

```
<?xml version="1.0" encoding="UTF-8"?>

<extensions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="extensions.xsd">
  <extension id="testExtension"
    defaultName="Test Extension"
    defaultDescription="Demonstrates extensions with tokens."
    url="http://www.example.com:8989/TestExtension/index.jsp?eac-host=
${EAC_HOST}&amp;eac-port=${EAC_PORT}&amp;eac-app=${EAC_APP}"
  </extension>
</extensions>
```

Note the use of the **&amp;** entity in the **url** attribute in place of the ampersand in the URL. In general, you should ensure that the **ws-extensions.xml** file validates against the provided schema before updating IAP Workbench with the new configuration.

Token-based authentication for Workbench extensions

You can enable extensions to authenticate users coming from IAP Workbench by including an authentication token in the URL. IAP Workbench calculates the value of the token by generating an MD5 hash from a portion of the URL and a shared secret. The portion of the URL that is used for the hash consists of everything after the host name and port, including the leading slash, but excluding the value of the AUTH token itself. The shared secret is a string that is specified in **ws-extensions.xml** and is also stored in the extension itself.

For example, the following **ws-extensions.xml** file defines an extension with a URL that uses the AUTH and TS tokens:

```
<?xml version="1.0" encoding="UTF-8"?>

<extensions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="extensions.xsd">
  <extension id="authExtension"
    defaultName="Authenticated Extension"
    defaultDescription="Demonstrates token-based authentication."
    url="http://localhost:8080/AuthExtension/index.jsp?timestamp=${TS}&amp;
auth=${AUTH}"
    role="admin"
    sharedSecret="secret!@#$%^*(987654321" />
</extensions>
```

In this case, the value of the authentication token is the hash of a string that looks similar to this:

```
/AuthExtension/index.jsp?timestamp=1189702462936&auth=secret!@#$%^*(987654321
```

The extension can verify that a user is coming from IAP Workbench by calculating the hash of the same string and comparing the result to the value of the AUTH token. This ensures that the user visiting the extension has logged in to IAP Workbench and has the role (if any) that is required to access the extension.

Because the AUTH token is based in part on the URL, it is recommended that you include the time stamp of the request to introduce some variation in the value of the token. The time stamp can also be used to filter out stale requests and limit the possibility of an eavesdropper reusing the same URL to gain access to the extension.

The following Java code shows how the extension defined in the preceding example can authenticate users from IAP Workbench:

```
// These values depend on what you defined in ws-extensions.xml
String extensionSecret="secret!@#$%^*(987654321";
final String authTokenParameterName = "auth";
final String timeStampParameterName = "timestamp";

// Set the tolerance, in milliseconds, before a request is considered too old
int allowedTimeStampSlackInMS = 5 * 60 * 1000;

// Calculate the hash of the substring of the URL and the shared secret
String url = request.getRequestURI() + "?" + request.getQueryString();
String findAuthToken = "&" + authTokenParameterName + "=";
url = url.substring(0, url.indexOf(findAuthToken) + findAuthToken.length());
String authCode = request.getParameter(authTokenParameterName);

MessageDigest md = MessageDigest.getInstance("MD5");
byte[] md5Hash = md.digest((url + extensionSecret).getBytes("UTF-8"));

StringBuffer hashCode = new StringBuffer();

for(int i : md5Hash)
{
  String str = Integer.toHexString(i+128);
  if (str.length() < 2)
  {
    str = "0" + str;
  }
  hashCode.append(str);
}

// Compare the hash to the value of the AUTH token
if (!hashCode.toString().equals(authCode))
{
  // Authentication fails because AUTH token did not match
}

// Compare the time stamp of the request to the current time stamp
long currentTime = new Date().getTime();
long ts = Long.parseLong(request.getParameter(timeStampParameterName));

if ( Math.abs(ts - currentTime) > allowedTimeStampSlackInMS)
{
  // Authentication fails because request is too old
}
```

The example extension places the AUTH token at the end of the URL, making it more convenient to build the substring of the URL for the hash.

However, the AUTH token can be in any position in the URL. For instance, the URL can be defined in **ws-extensions.xml** is as follows:

```
url="http://localhost:8080/AuthExtension/index.jsp?auth=${AUTH}&amp;
timestamp=${TS}"
```

This would result in a URL similar to this:

```
http://localhost:8080/AuthExtension/index.jsp?auth=dc40570f2e7111fbe1af820a85
4ca817&timestamp=1189702462936
```

The value of the authentication token would be the hash of a string similar to this:

```
/AuthExtension/index.jsp?auth=&timestamp=1189702462936secret!@#$%^*(987654321
```

In this case the code in the extension to remove the value of the authentication token from the URL would be more complex.

Theming extensions to match IAP Workbench

IAP Workbench provides a public cascading style sheet that includes the most common style elements in IAP Workbench. You can use the style sheet in your extension to give it a look and feel similar to that of the IAP Workbench interface.

To use the IAP Workbench public cascading style sheet:

- Add the following line within the **head** element of your HTML document:

  ```
  <link rel="stylesheet" type="text/css"
    href="http://hostname:8888/stylesheets/public.css"/>
  ```

  The host name is the name or IP address of the IAP Workbench server. Replace 8888 with the IAP Workbench port if it is not running on the default port.

For more information about the styles defined in the public style sheet, see the comments within the **public.css** file. The file can be viewed at the following URL on the IAP Workbench server:

```
http://hostname:port/stylesheets/public.css
```

The **public.css** file cannot be edited. If you want to specify additional styles or modify the default styles, create a separate style sheet and apply it to your application.

Troubleshooting Workbench extensions

The following sections provide troubleshooting information about Workbench extensions.

If the extension does not have a link in the navigation menu or launch page:

- Stop and restart the Endeca HTTP service. Changes to the XML configuration files for extensions, roles, and the navigation menu do not go into effect until the service is restarted.
- Ensure that you have the required IAP Workbench user role to access the extension.
- Ensure that a menu item for the extension is specified in **ws-mainMenu.xml** and that the **id** attribute matches the id of the extension as defined in **ws-extensions.xml**. Defining an extension in **ws-extensions.xml** does not automatically add a link to the navigation menu in IAP Workbench. For more information about customizing the IAP Workbench menu, see “Updating the IAP Workbench menu and launch page” on page 64.

  If you want an extension to have an entry on the launch page, specify **onLaunchPage="true"** in the **menuitem** element for the extension in **ws-mainMenu.xml**.
- If you have no applications defined in IAP Workbench, the only links that display in the navigation menu are for the EAC Admin Console and EAC Settings. To enable display of the full IAP Workbench menu, you must first provision an application.

If the link displays in the menu but the extension does not display when you click the link:

- Ensure that the URL for the extension specified in **ws-extensions.xml** is a valid HTTP or HTTPS URL. A Workbench extension must be a Web application running in a Web server.

If the IAP Workbench window does not display at all after updating

ws-extensions.xml:

There may be a problem with your XML configuration files that prevents IAP Workbench from starting up. The error messages in the IAP Workbench log can help you identify whether one of the following is the case:

- One or more of the XML configuration files is missing. The following files must be present in **%ENDECA_CONF%\conf** (on Windows) or **$ENDECA_CONF/conf** (on UNIX):

  - **ws-extensions.xml** and its associated schema, **extensions.xsd**
  - **ws-mainMenu.xml** and its associated schema, **mainMenu.xsd**
  - **ws-roles.xml** and its associated schema, **roles.xsd**

  The files are created in this location when you install Endeca. By default, the **ws-extensions.xml** and **ws-roles.xml** files define no extensions or additional roles. The **ws-mainMenu.xml** file controls the display of the navigation menu and launch page.

  If you have deleted one of these files, you can restore the default file by copying it from **%ENDECA_ROOT%\workspace_template\conf** (on Windows) or **$ENDECA_ROOT/workspace_template/conf** (on UNIX).

- One or more of the configuration files contains badly formed or invalid XML.

  Ensure that the configuration files contain well-formed XML. In particular, check that any ampersand that is used within an attribute value is specified as the **&amp;** entity.

  Use an XML tool to validate any configuration files that you have edited against the associated schema in **%ENDECA_CONF%\conf** (on Windows) and **$ENDECA_CONF/conf** (on UNIX).

Chapter 13

Transferring Endeca Implementations Between Environments

This chapter describes how to transfer your Endeca implementation from a staging environment that uses IAP Workbench to a production environment that uses IAP Workbench. Two methods are described: one uses the Endeca tools to manually transfer the implementation, and the other uses an **emgr_update** utility that allows you to script and automate transfers.

To improve the readability of the chapter, we assume you are transferring your Endeca implementation from a staging environment to a production environment. This need not be the case, however. You can use these procedures to transfer between any environments you choose.

This chapter includes the following sections:

- About transferring your front-end Web application
- Transferring implementations using the tools
- Transferring implementations using the emgr_update utility
- Transferring auto-generated and external dimension value ID assignments between environments
- Removing an application from Endeca IAP

About transferring your front-end Web application

This chapter focuses on transferring your instance configuration and MDEX Engine between environments. Depending on your environment and requirements, you may also have to transfer your front-end Web application to complete the move from one environment to another. From an Endeca perspective, all you have to do to transfer the front-end Web application is make sure the MDEX Engine hostname and port you are using in your **ENEConnection** object is correct for the new environment.

Note: *See the* Endeca Developer's Guide *for details on using the* **ENEConnection** *interface.*

Transferring implementations using the tools

You can use the Endeca tools to manually transfer from a staging environment that uses IAP Workbench to a production environment that uses IAP Workbench.

Retrieving the IAP Workbench instance configuration with Developer Studio

1. In your staging environment, start Developer Studio and create a new project.
2. From the Tools menu, choose IAP Workbench Settings.

   The IAP Workbench Settings dialog box appears.
3. Specify the hostname and port for IAP Workbench for this application. Make sure the hostname and port that are specified correspond with IAP Workbench whose information you want to retrieve.
4. In the same dialog box, select the application from the drop-down list, or make sure that the application name that is specified corresponds with the name of the application whose configuration you want to retrieve from IAP Workbench. Note that you can have instance configurations for more than one application created in IAP Workbench.

5. In the IAP Workbench toolbar, click Get Instance Configuration.

6. From the File menu, choose Save to save the project with the latest instance configuration.
7. Optionally, remove inactive dynamic business rules from the instance configuration.
8. Copy the instance configuration files from the saved project to the location in your production environment where the Endeca Application Controller expects them to be.
9. If you have used the load function in Developer Studio to load auto-generated or external dimensions, you should manually synchronize these Forge state files. For more information see "Transferring auto-generated and external dimension value ID assignments between environments."
10. Use the Application Controller to run a baseline update on the production system.

Transferring implementations using the emgr_update utility

Similar to the Endeca tools, the **emgr_update** utility lets you transfer your Endeca implementation from a staging environment that uses IAP Workbench to the production environment that uses IAP Workbench.

The primary benefit of the **emgr_update** utility is that it allows you to script and automate transfers between environments. Transferring an implementation from staging to production is a two-step process where you get the instance configuration from the staging environment and then set (update) the configuration in the production environment.

Note: *If you have used the load function in Developer Studio to load auto-generated or external dimensions, you should manually synchronize these Forge state files. For more information see "Transferring auto-generated and external dimension value ID assignments between environments."*

emgr_update syntax

The **emgr_update** is a utility that assists you in updating the instance configuration of a production system based on the changes made with the Endeca tools in a staging environment.

You run **emgr_update** from a command line. Open a command prompt or UNIX shell to run the program. The syntax for running **emgr_update** is:

```
emgr_update <parameters>
```

The following table describes the command line parameters you can use with **emgr_update**. You can specify only one **--action** operation for each invocation of the utility.

emgr_update parameter	Description
--host name:port	Specifies the host name and the port of a machine running IAP Workbench. If you are retrieving settings (using the **get** operation), this is the host name of the environment you are transferring *from*; if you are updating settings (using the **set** operation), this is the host name of the environment you are transferring *to*.
--action <op>	Specifies one of the actions, where **<op>** is one of the operations listed below.
--action get_all_settings	Retrieves all the instance configuration settings for a project you performed in the IAP Workbench in the staging environment, for their use in the production environment. Required parameters: **--dir, --prefix** Optional parameters: **--filter**

emgr_update parameter	Description
--action get_ws_settings	Retrieves only those instance configuration settings that can be modified in IAP Workbench (not all settings). These configuration settings include the following IAP Workbench features: dynamic business rules, keyword redirects, thesaurus entries, automatic phrases, stop words, and dimension ordering.
--action get_mdex_settings	Retrieves the instance configuration settings that were modified in IAP Workbench, and that do not require a baseline update to update the MDEX Engine. These configuration settings include the following IAP Workbench features: dynamic business rules, keyword redirects, thesaurus entries, automatic phrases. Required parameters: **--dir, --prefix** Optional parameters: **--filter**
--action set_post_forge_dims	Updates the IAP Workbench configuration with the post-Forge dimensions.
--action get_post_forge_dims	Retrieves the copy of IAP Workbench settings for the post-Forge dimensions. Typically, this operation can be used for debugging purposes.
--action update_mgr_settings	Updates a IAP Workbench production environment with instance configuration settings that were extracted from the IAP Workbench configuration in the staging environment.

emgr_update parameter	Description
--action remove_all_settings	Removes all the instance configuration files from IAP Workbench for the application that you specify with the **--app_name** parameter. Removing the instance configuration does not remove the associated provisioning information for an application.
--app_name <string>	Specifies the name of the application provisioned to the EAC Central Server.
Optional action parameters	
--dir <string>	Specifies the pathname of the directory where the instance configuration files are written to or read from. Required for all **--action** operations except for **set_post_forge_dims** and **remove_all_settings**.
--prefix <string>	Specifies the prefix used for the instance configuration files. This option is required for all **--action** operations except for **get_post_forge_dims** and **set_post_forge_dims**.
--filter *filter*	Filters out dynamic business rules that have a state of inactive. (A rule has the property **endeca.internal.workflow.state** set to INACTIVE.) This option can be used in conjunction with **get_all_settings** or **get_ws_settings** when retrieving an instance configuration. Removing inactive rules is not required but it is recommended. With the default rule filter in place, the MDEX Engine does not fire any rule whose state is inactive. In other words, you can transfer an instance configuration, including both active and inactive rules, and the MDEX Engine fires only active rules in reply to user queries.

emgr_update parameter	Description
--post_forge_file <string>	Specifies the pathname to the file that contains post-Forge dimensions. This option is required for the **set_post_forge_dims** operation.
Optional global parameters	
--stop_on_warnings	Stops the utility without asking you if the target directory is not empty before a **get** operation, or if it finds extra or missing files before an **update** operation.
--ignore_warnings	Continues running the utility if the target directory is not empty before a **get** operation, or continues if there are extra or missing files before an **update** operation.
--help	Displays the usage parameters for the utility.
--version	Displays the version number for the utility.

Here is an example of usage:

```
   emgr_update --host localhost:8888 --action get_ws_settings
--prefix wine --dir /apps/endeca/data/forge_input --app_name wine
```

By using the appropriate **--action** operations, you can use the **emgr_update** program to do the following tasks:

- Transfer the instance configuration files for a particular application of your choice from the staging environment to the production environment. After the transfer, you run a baseline update using your own EAC scripts. You have the option of transferring all instance configuration files, or transferring just the instance configuration files that IAP Workbench modified.
- Transfer the instance configuration for a particular application from one IAP Workbench environment to another.
- Remove instance configuration information for a specified application from the IAP Workbench configuration.

- Send the Forge dimensions to the IAP Workbench.

These operations are described in the following sections.

Using emgr_update to transfer from a IAP Workbench staging environment to a IAP Workbench production environment

This section describes how to transfer instance configuration files from a staging environment that uses IAP Workbench to a production environment that also uses IAP Workbench. Two scenarios are described:

- Transferring all instance configuration files for an Endeca project.
- Transferring only the instance configuration files that can be modified by IAP Workbench.

Transferring all instance configuration files

For this task, you move all the instance configuration files from the staging environment to the Forge input directory in the production environment. You then use the Endeca Application Controller to run a baseline update.

To transfer all configuration files to the production system:

1. In the staging environment, use Developer Studio and/or IAP Workbench to make changes to the project.
2. Run **emgr_update** with an **--action** of **get_all_settings**.
   - **a** For the **--host** parameter, specify the machine name and port for the staging IAP Workbench environment.
   - **b** For the **--dir** parameter, specify the Forge input directory in the production environment.
   - **c** For the --**app_name** parameter, specify the application name whose instance configuration you want to transfer.
   - **d** Use the **--filter** parameter, to remove inactive business rules.

   The following is a UNIX example.

```
emgr_update --host localhost:8888 --app_name My_application
--action get_all_settings --prefix wine --filter --dir
/apps/endeca/data/forge_input
```

3 If the destination directory is not empty, you will be prompted to continue. Answer **y**.

   When the utility finishes, all project configuration files (including the project and pipeline files) are copied to the production directory specified by the **--dir** parameter.

4 Use the Endeca Application Controller to run a baseline update on the production system.

The utility uses *prefix***.esp** as the name of the output Developer Studio project file (where *prefix* is whatever you specified with the **--prefix** parameter). If there is an existing project file in the production directory with another name, it is recommended that you change it to *prefix***.esp**.

Transferring only instance configuration files modified by IAP Workbench

In this task, you transfer files from the staging environment to the Forge input directory in the production environment. However, these files are the instance configuration files that can be modified by IAP Workbench. They are not the full set of instance configuration files. IAP Workbench can modify instance configuration files for any of the following features:

- Dynamic business rules
- Thesaurus entries
- Automatic phrases
- Stop words
- Dimension ordering

A subsequent baseline update uses the updated files for these features.

To transfer instance configuration files modified by IAP Workbench to a production system:

1 In the staging environment, use IAP Workbench to make any necessary instance configuration changes.

2 Run **emgr_update** with an **--action** of **get_ws_settings**.

   a For the **--host** parameter, specify the machine name and port for the IAP Workbench in the staging environment.

   b For the **--dir** parameter, specify the Forge input directory in the production environment.

   c For the --**app_name** parameter, specify the application name whose instance configuration you want to transfer.

   d Use the **--filter** parameter, to remove inactive dynamic business rules.

   The following is a Windows example:

```
    emgr_update.bat --host localhost:8888 --app_name My_application --action
get_ws_settings--prefix wine --filter --dir c:\endecaproduction\data\forge_input
```

3 If the destination directory is not empty, you will be prompted to continue. Answer **y**.

   When the utility finishes, the project files that IAP Workbench modified are copied to the production directory specified by the **--dir** parameter.

4 Use the Application Controller to run a baseline update on the production system.

Using emgr_update to transfer from one IAP Workbench environment to another

The process for transferring and deploying instance configuration files from one IAP Workbench environment to another, using the **emgr_update** utility is accomplished using one of the **emgr_update** utility's **--action** operations, and the entire process can be scripted.

To transfer and deploy all instance configuration files to the production system:

1. In the staging environment, use Developer Studio and/or IAP Workbench to make changes to the project.
2. Run **emgr_update** with an **--action** of **get_all_settings**.
   - **a** For the **--host** parameter, specify the machine name and port for the staging environment in IAP Workbench.
   - **b** For the **--dir** parameter, specify the Forge input directory in the production environment.
   - **c** For the --**app_name** parameter, specify the application name whose instance configuration you want to transfer.
   - **d** Use the **--filter** parameter, to remove inactive dynamic business rules.

   The following is a Windows example:

```
   emgr_update.bat --host localhost:8888 --app_name My_app --action
get_all_settings --prefix wine --filter --dir
c:\endecaproduction\data\forge_input
```

3. If the destination directory is not empty, you will be prompted to continue. Answer **y**.

   When the utility finishes, all project configuration files are copied to the production directory specified by the **--dir** parameter.
4. Run **emgr_update** with an **--action** of **update_mgr_settings**.
   - **a** For the **--host** parameter, specify the machine name and port for the production environment in IAP Workbench.
   - **b** For the **--dir** parameter, specify the directory that contains the project configuration files that will be used to update the production environment in IAP Workbench (typically, this will be the same directory that was used in step 2).
   - **c** For the --**app_name** parameter, specify the application name whose instance configuration you want to transfer.

   The following is a Windows example:

```
   emgr_update.bat --host localhost:8888 --app_name My_app--action
update_mgr_settings --prefix wine --dir c:\endecaproduction\data\forge_input
```

5 Use the Application Controller to run a baseline update on the production system.

Using emgr_update to remove instance configuration files from IAP Workbench

Deleting an application from EAC in IAP Workbench does not remove its instance configuration files. If you want to delete the instance configuration files for the application, you can use **emgr_update**.

To remove instance configuration files:

- Run **emgr_update** with an **--action** of **remove_all_settings**.
  - **a** For the **--host** parameter, specify the machine name and port for the staging environment in IAP Workbench.
  - **b** For the --**app_name** parameter, specify the application name whose instance configuration you want to remove.

  The following is a Windows example:

```
   emgr_update.bat --host localhost:8888 --app_name My_app --action
remove_all_settings --prefix My_prefix
```

The application's instance configuration files are removed from the IAP Workbench.

Using emgr_update to send the dimensions file produced by Forge to the IAP Workbench

Read this section only if you are not using an Application Controller default script for running the baseline update, and are using your own scripts for this purpose.

If you are using your own scripts for running the baseline update, then after you run Forge, you need to send the dimensions file produced by Forge to the IAP Workbench instance configuration for your application.

To send the dimensions file produced by Forge to the IAP Workbench, run emgr_update as follows:

- On the machine that has access to the output files of Forge (this is typically the machine on which you ran Forge), run **emgr_update** with an **action** of **set_post_forge_dims**:

  a For the **--host** parameter, specify the machine name and port for the environment in IAP Workbench.

  b For the --**app_name** parameter, specify the application name whose instance configuration you want to update with this information.

  c For the **--post_forge_file** parameter, specify the full pathname to the output file where Forge stores its dimensions.

```
    emgr_update.bat –-host localhost:8888 --action set_post_forge_dims –-app_name
wine –-post_forge_file
C:\sample_wine_data\data\partition0\forge_output\wine.dimensions.xml
```

Transferring auto-generated and external dimension value ID assignments between environments

If you have loaded any auto-generated or external dimension values into Developer Studio using the load function, the ID assignments are not stored in the instance configuration. Therefore, you should manually synchronize these files between implementation environments. Otherwise — unless your data is identical in each environment — IDs assigned by Forge may not match those assigned by Developer Studio. Mismatched ID assignments can affect project settings that depend on IDs stored in Developer Studio such as dimension value ordering, precedence rules, and business rules.

To transfer auto-generated and external dimension value ID assignments:

1 Navigate to the Forge state directory of the Endeca application instance that you use to modify dimension value ordering, precedence rules, and business rules. For example:

C:\Apps\staging\myapp\data\state

2 Locate and back up the Forge state files containing the auto-generated and external dimension value ID assignments. For example:

```
C:\Apps\staging\myapp\data\state\autogen_dimensions.xml.external.gz
C:\Apps\staging\myapp\data\state\autogen_dimensions.xml.gz
```

3 Copy the Forge state files.

4 Paste these files into the Forge state directory of the application instance to which you are transferring your implementation.

C:\Apps\production\myapp\data\state

5 Repeat this process for any additional implementation environments.

Your auto-generated and external dimension value IDs are now identical in both environments. You should repeat this process whenever you modify dimension value ordering, precedence rules, or business rules.

Removing an application from Endeca IAP

Removing an application in the EAC Admin Console removes provisioning information for an application but not instance configuration files. Running **remove_all_settings** in **emgr_update** removes instance configuration files but not provisioning information. To completely remove all information about an application, you perform both steps. If you do not perform both steps, you may store unnecessary or duplicate sets of files for an application.

To completely remove an application from the Endeca IAP:

1 In IAP Workbench, log in to the application you want to remove.

2 On the EAC Admin Console page, click Delete. Alternatively, you can also perform steps 1 and 2 using an EAC web services client. This removes the provisioning information.

3. Run **remove_all_settings** of **emgr_update** to delete the instance configuration files. For details, see “Using emgr_update to remove instance configuration files from IAP Workbench” on page 88.

Index