

**<CLIENT> Runbook**



www.<CLIENTSITE>.com

<CLIENT>

08/17/2014

Version <n>

Table of Contents

[Infrastructure 4](#_Toc401568624)

[Physical Servers 4](#_Toc401568625)

[How to add a physical server: 4](#_Toc401568626)

[How to remove a physical server: 4](#_Toc401568627)

[Setting up the Assembler Service 5](#_Toc401568628)

[Experience Manager Workbench Settings 5](#_Toc401568629)

[Assembler API Configuration 6](#_Toc401568630)

[How to set up the assembler service: 8](#_Toc401568631)

[How to remove the assembler service: 9](#_Toc401568632)

[Setting up the MDEX Engines 9](#_Toc401568633)

[Adding a new MDEX engine 9](#_Toc401568634)

[Remove a MDEX engine: 10](#_Toc401568635)

[Portability across Environments 11](#_Toc401568636)

[Modifying the Control Scripts 13](#_Toc401568637)

[Automate Data Feed 13](#_Toc401568638)

[Data Feed Location 14](#_Toc401568639)

[Archiving 14](#_Toc401568640)

[Data Checks 14](#_Toc401568641)

[Email Configuration 15](#_Toc401568642)

[IFCR Checks 16](#_Toc401568643)

[Modifying the Application Configuration Files 17](#_Toc401568644)

[Configuring the Assembler Files 17](#_Toc401568645)

[Supplemental Infrastructure Tasks 18](#_Toc401568646)

[Engine Load Balancer 18](#_Toc401568647)

[Load Balancing Scripts 19](#_Toc401568648)

[DNS Configuration 20](#_Toc401568649)

[Health Checks 20](#_Toc401568650)

[Indexer 20](#_Toc401568651)

[Live MDEX 20](#_Toc401568652)

[Live Web Services 21](#_Toc401568653)

[Baseline Update Process 21](#_Toc401568654)

[Forge 21](#_Toc401568655)

[Dgidx 21](#_Toc401568656)

[Dgraph 21](#_Toc401568657)

[Promote Content 21](#_Toc401568658)

[Troubleshooting Baseline Failures 22](#_Toc401568659)

[Forge 22](#_Toc401568660)

[Dgidx 22](#_Toc401568661)

[Dgraph 23](#_Toc401568662)

[Other 23](#_Toc401568663)

[Performance Tuning 23](#_Toc401568664)

[Reporting 25](#_Toc401568665)

[Staging/Production Instances 27](#_Toc401568666)

[Deployment Template 28](#_Toc401568667)

[Assembler Modifications 28](#_Toc401568668)

[Application Modifications 28](#_Toc401568669)

[GameStopStage 28](#_Toc401568670)

[GameStopProd 29](#_Toc401568671)

[Prod Push Script 29](#_Toc401568672)

[Pipeline Modifications for Environment Portability 30](#_Toc401568673)

[Infrastructure 31](#_Toc401568674)

[Indexing Server Installs 32](#_Toc401568675)

[Installing the MDEX Engine on Windows 33](#_Toc401568676)

[Installing Platform Services on Windows 33](#_Toc401568677)

[Installing Tools and Frameworks on Windows 34](#_Toc401568678)

[Installing CAS on Windows 35](#_Toc401568679)

[Installing Developer Studio 35](#_Toc401568680)

[Installing Python 35](#_Toc401568681)

[Installing Apache Tomcat (Live) 36](#_Toc401568682)

[Live MDEX/Webserver Installs 37](#_Toc401568683)

[Setting the Environment Variable 37](#_Toc401568684)

[Move GameStop app from Environment to Environment 38](#_Toc401568685)

[Move Experience Manager Configurations 39](#_Toc401568686)

[Move the Assembler App 40](#_Toc401568687)

[Config Scripts 41](#_Toc401568688)

[Custom Control Scripts 42](#_Toc401568689)

[GameStop Run (Partial Run) 42](#_Toc401568690)

[GameStop Mod 43](#_Toc401568691)

[Promote Content 43](#_Toc401568692)

[Export Site to Production 43](#_Toc401568693)

[Export Import Site 43](#_Toc401568694)

[Record Store Check (PratialCheck) 43](#_Toc401568695)

[Record Store Archive Prep (PartialPrep) 43](#_Toc401568696)

[Baseline Update Process 44](#_Toc401568697)

[Baseline Update 44](#_Toc401568698)

[Promote Content 44](#_Toc401568699)

[Partial Update 44](#_Toc401568700)

[Export Import Site 44](#_Toc401568701)

# Infrastructure

## Physical Servers

### How to add a physical server:

When scaling the GameStop application, it is recommended to add additional instances to handle the load. This can be done by adding multiple MDEX engines and Assembler Services.

Once the server has been created, the Endeca MDEX and Platform Services need to be installed. Please follow the install guides on the Oracle.com website to install the software. The MDEX has to be installed first and then Platform services.

### How to remove a physical server:

A server and the Endeca components can be removed from an application. Here are the steps:

1. Stop the MDEX running on the server. Here is the command (replace Dgraph1 with actual name
   1. runcommand.bat Dgraph1 stop
2. Remove any references in the AppConfig files to the host that is being removed. This should be done before any servers are permanently removed or there will be EAC errors about unreachable hosts.
3. Run a baseline update to make sure everything is working after removing any references to the removed host.
4. Stop Endeca services on the server to be removed
5. Uninstall Endeca on the server and the server can be removed

The Endeca Platform services service should be stopped on the server before removing any components

## Setting up the Assembler Service

### Experience Manager Workbench Settings

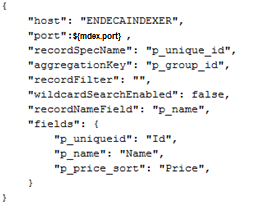
In order for experience manager to communicate with the engine, there are a couple files that need to be updated to reflect to correct configuration.

**dataservices.json**

This file is located in the “C:\Endeca\apps\<client>\config\editors\_config\services” directory for the <client>

Since the recommended configuration is for experience manager to connect to the authoring application, the file should look like this:

***<client>:***

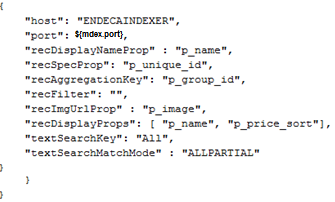


**endecaBrowserService.json**

This file is located in the “C:\Endeca\apps\<client>\config\editors\_config\services”  ”  directory.

Since the recommended configuration is for experience manager to connect to the authoring application, the file should look like this:

***<client>:***

******

The endeca browser is used when you are narrowing down results or using boost and bury in experience manager.

### Assembler API Configuration

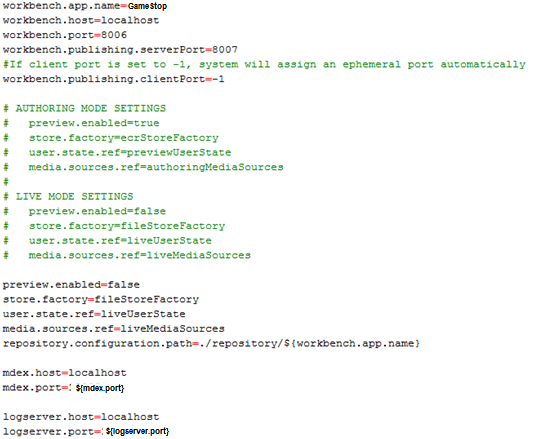
First, edit the assembler.properties file located in at:

C:\Endeca\ToolsAndFrameworks\11.0.0\reference\<client>-service\WEB-INF\assembler.properties

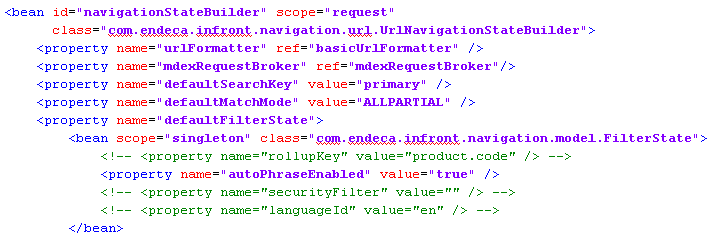
All variables will need to be updated to match the configuration as noted in the above sections. For mdex host and port, please reference the load balancer port and host if configured.

1. Change assembler.properties for workbench.host to use localhost
2. Add the following  command to the initialize\_services script:

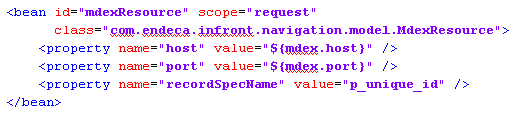
powershell -Command "(gc ..\web\_apps\<client>-service\WEB-INF\assembler.properties) -replace 'workbench.host=localhost', 'workbench.host=%ENDECA\_WORKBENCH\_HOST%' | Out-File  ..\web\_apps\gamestop-service\WEB-INF\assembler.properties -encoding "Default" "  
echo Finished updating assembler.properties file for gamestop-service application



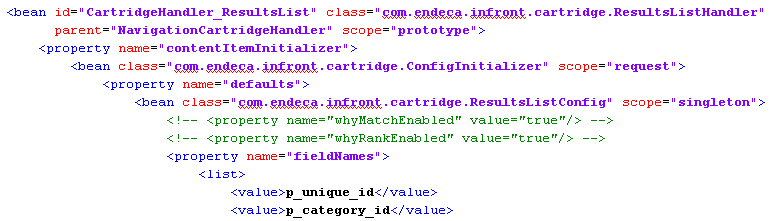
Next, to enable Web-service calls, if needed, we will update the assembler-context.xml file located in the same directory. First comment out the rollupKey property located in the navigationStateBuilder block:



Then, note the record spec (p\_unique\_id) in the mdexResource block:



Next, if needed, you can limit the properties returned in the ResultsList cartridge by adding or modifying the property names here:



Finally, add the service context file by creating a file called <client>-service.xml in the following directory:

C:\Endeca\ToolsAndFrameworks\11.0.0\server\workspace\conf\Standalone\localhost

The file will contain the path from the url and the service path location as seen below:

Once setup, restart the Tools service, and verify you are receiving back xml/json from the following URLS:

**GameStop:**

**Development:** [http://ENDECAINDEXER:8006/<client>-dev-service/json/pages/browse](http://ENDECAINDEXER:8006/%3cclient%3e-dev-service/json/pages/browse)

**Live:** [http://ENDECAINDEXER:8006/<client>-service/json/pages/browse](http://ENDECAINDEXER:8006/%3cclient%3e-service/json/pages/browse)

### How to set up the assembler service:

* Verify Apache Tomcat has been installed on the webserver.

Copy the <client>-service app to the new webserver(..\apache-tomcat-<version>\webapps\). The <client> app is located in the ToolsAndFrameworks directory on the indexing server.

Location: C:\Endeca\ToolsAndFrameworks\11.0.0\reference\<client>-service

* See to complete setup.

### How to remove the assembler service:

Here are the steps to remove the assembler service:

1. Stop the Apache Tomcat service from running
2. Uninstall Apache Tomcat

## Setting up the MDEX Engines

### Adding a new MDEX engine

* Verify MDEX is installed
* Verify Platform Services is installed

When you initially deploy a new application using the Deployment Template, it sets up an environment in which two Dgraphs are running on the same MDEX Engine server. One Dgraph is for the authoring application and the other Dgraph is for the live application. You can optionally change this configuration and add one or more MDEX Engine servers, with one or more Dgraphs, to either the authoring or live application.

This task describes how to add an MDEX Engine Server, and Dgraphs on that server, to a Live application by modifying settings in the LiveDgraphCluster.xml file. However, the task is essentially the same as adding an MDEX Engine server to an authoring application. In an authoring application, you modify AuthoringDgraphCluster.xml instead of LiveDgraphCluster.xml. Within each configuration file, the settings are the same to define host, dgraph, and dgraph-cluster options.

**To add MDEX Engine servers and Dgraphs to the servers:**

1. Navigate to <app dir>/config/script and open LiveDgraphCluster.xml in a text editor.

**2.** Locate the Live MDEX Hosts section of the file and add one or more host elements to represent an MDEX Engine server. Specify host name and EAC Agent port information for the new server.

This example adds a server named LiveServerB.CompanyName.com:

*<!--*

*################################################################*

*# Live MDEX Hosts - The machines used to host all MDEX processes*

*# for the 'live environment' MDEX cluster.*

*#*

*-->*

*<host id="LiveMDEXHostA" hostName="LiveServerA.CompanyName.com" port="8888"*

*/>*

*<host id="LiveMDEXHostB" hostName="LiveServerB.CompanyName.com" port="8888"*

*/>*

**3.** *Add the Dgraphs that run on the new server to the Dgraph Cluster block.*

*This example adds DgraphB1 and DgraphB2 to the new LiveServerB.CompanyName.com server:*

*<!--*

*################################################################*

*# Live Dgraph Cluster - The 'live environment' MDEX cluster.*

*#*

*-->*

*<dgraph-cluster id="LiveDgraphCluster" getDataInParallel="true" enabled="true">*

*<dgraph ref="DgraphA1" />*

*<dgraph ref="DgraphA2" />*

*<dgraph ref="DgraphB1" />*

*<dgraph ref="DgraphB2" />*

*</dgraph-cluster>*

**4.** Add a Dgraph process definition for each Dgraph on the new server.

This example adds DgraphB1 and DgraphB2 to the new LiveServerB.CompanyName.com server:

*...*

*<dgraph id="DgraphB1" host-id="LiveMDEXHostB" port="${mdex.port}"*

*post-startup-script="LiveDgraphPostStartup">*

*<properties>*

*<property name="restartGroup" value="1" />*

*<property name="DgraphContentGroup" value="Live" />*

*</properties>*

*<log-dir>./logs/dgraphs/DgraphB1</log-dir>*

*<input-dir>./data/dgraphs/DgraphB1/dgraph\_input</input-dir>*

*<update-dir>./data/dgraphs/DgraphB1/dgraph\_input/updates</update-dir>*

*</dgraph>*

*<dgraph id="DgraphB2" host-id="LiveMDEXHostB" port="${mdex.port}"*

*post-startup-script="LiveDgraphPostStartup">*

*<properties>*

*<property name="restartGroup" value="2" />*

*<property name="DgraphContentGroup" value="Live" />*

*</properties>*

*<log-dir>./logs/dgraphs/DgraphB2</log-dir>*

*<input-dir>./data/dgraphs/DgraphB2/dgraph\_input</input-dir>*

*<update-dir>./data/dgraphs/DgraphB2/dgraph\_input/updates</update-dir>*

*</dgraph>*

*...*

**5.** Save and close LiveDgraphCluster.xml.

### Remove a MDEX engine:

1a. Logging into the Workbench and Stopping the desired Dgraph **OR**

1b. Running the runcommand.sh script:

Ex. runcommand.bat shutdown <Dgraph>

2. Go into the LiveDgraphCluster.xml file and comment out the desired Dgraph from the dgraph cluster.

*<dgraph-cluster id="LiveDgraphCluster" getDataInParallel="true" enabled="true">*

*<!--<dgraph ref="DgraphA" />--> EXAMPLE*

*<dgraph ref="DgraphB" />*

*<dgraph ref="DgraphC" />*

*<dgraph ref="DgraphD" />*

*</dgraph-cluster>*

## Portability across Environments

The development, staging, and production environments often include servers with different operating systems and directory structures. Using forward slashes, relative path names and variables can ensure that application definitions propagated across environments work in the target environment.

**Use forward slashes**

To ensure that paths work in all locations, use forward slashes ("/") wherever possible in configuration files and scripts. Windows and Unix environments both work well with forward slashes. The only exceptions are platform-specific scripts, such as Windows .bat files or UNIX shell scripts.

**Use relative paths in the pipeline**

Using absolute paths in the Forge pipeline ties the application to a particular location in one environment. It is therefore better to use relative paths. In record-adapters, paths are relative to <app dir>/data/processing, which contains the content of <app dir>/data/incoming before the pipeline is run. Therefore any files from <app dir>/data/incoming can be referenced directly by name, without specifying a directory.

Java manipulator paths are relative to <app dir>. This applies both to the manipulator's class path and to any files the Java code may try to access. Since these file names are often given as pass-throughs, keep such pass-through values relative to <app dir>.

**Use variables in scripts**

You can specify relative paths in scripts, but keep in mind that scripts are not always invoked from the same directory. For example, there might be a script called crawl.sh in <app dir>/control. If this script uses paths relative to <app dir>/control, then it will only work correctly if it is invoked while stored in this directory. But if the script is invoked as/control/crawl.sh from <app dir>, then the paths will be incorrect and the script may fail.

To avoid this problem, scripts can use preset variables to refer to a known directory without presuming any particular absolute path. In Windows batch files, the variable%~dp0 resolves to the directory containing the script. In UNIX Bash scripts, the variable$0 resolves to the script name, and running the dirname command on it will give the script's directory. In BeanShell scripts for AppConfig.xml, the variable ${ENDECA\_PROJECT\_DIR} points to <app dir>. By using these variables, you can construct portable paths in location-independent scripts.

All the scripts generated by the Deployment Template use this technique and can serve as examples. For UNIX, the scripts generally start with the following lines:

WORKING\_DIR=`dirname ${0} 2>/dev/null`

. "${WORKING\_DIR}/../config/script/set\_environment.sh"

This sets the WORKING\_DIR variable to the directory containing the script being run, and then addresses the set\_environment.sh file in a different directory by making the path relative to WORKING\_DIR. Wherever the script is invoked, the path to set\_environment.sh will always resolve correctly.

For Windows, the scripts usually start with the following line:

call %~dp0..\config\script\set\_environment.bat

This addresses set\_environment.bat via a path relative to%~dp0. It ensures that the reference is correct wherever the script is invoked.

**Creating a custom file for environment-specific settings**

Most application configuration settings can be shared among all environments, but a few are environment-specific. These settings should be removed from the AppConfig.xml file and stored in a separate file. That way, each environment will have its own custom file of environment-specific settings that is not changed during synchronization.

Environment-specific settings typically include an application's name, file system paths, host addresses in the environment, and the definition of MDEX processes known as the Dgraph cluster.

To create a custom.xml file:

1. Edit the AppConfig.xml file generated by the Deployment Template to include a line similar to the following:

<spr:import resource="custom.xml" />

2. Move all environment-specific elements from AppConfig.xml to a new custom.xml file. These elements might include the <app>, <host>, <dgraph-cluster>, <dgraph>, and <logserver> elements.

3. Create a custom.xml file for each of the environments, with settings appropriate for that environment.

Here is a sample custom.xml file to use a general reference when creating your own file.

- <!--

########################################################################

# Global variables

-->

- <app appName="wine" eacHost="ConfigMig1" eacPort="8888" dataPrefix="wine"

sslEnabled="false" lockManager="LockManager">

<working-dir>${ENDECA\_PROJECT\_DIR}</working-dir>

<log-dir>./logs</log-dir>

</app>

- <!--

########################################################################

# Servers/hosts

#

# The "webstudio" host and its "webstudio-report-dir" directory use

# predefined names to inform Workbench where it should look for reports

# for this application.

#

-->

<host id="ITLHost" hostName="ConfigMig1" port="8888" />

<host id="MDEXHost" hostName="ConfigMig1" port="8888" />

- <host id="webstudio" hostName="ConfigMig1" port="8888">

- <directories>

<directory name="webstudio-report-dir">./reports</directory>

</directories>

</host>

- <!--

########################################################################

# Dgraph Cluster

#

-->

- <dgraph-cluster id="DgraphCluster" getDataInParallel="true">

<dgraph ref="Dgraph1" />

</dgraph-cluster>

- <!--

########################################################################

# Dgraphs

#

-->

- <dgraph id="Dgraph1" host-id="MDEXHost" port="${mdex.port}">

- <properties>

<property name="restartGroup" value="A" />

<property name="updateGroup" value="a" />

</properties>

<log-dir>./logs/dgraphs/Dgraph1</log-dir>

<input-dir>./data/dgraphs/Dgraph1/dgraph\_input</input-dir>

<update-dir>./data/dgraphs/Dgraph1/dgraph\_input/updates</update-dir>

</dgraph>

- <!--

########################################################################

# LogServer

#

-->

- <logserver id="LogServer" host-id="ITLHost" port="${mdex.port}">

- <properties>

<property name="numLogBackups" value="10" />

<property name="targetReportGenDir" value="./reports/input" />

<property name="targetReportGenHostId" value="ITLHost" />

</properties>

<log-dir>./logs/logservers/LogServer</log-dir>

<output-dir>./logs/logserver\_output</output-dir>

<startup-timeout>120</startup-timeout>

<gzip>false</gzip>

</logserver>

# Modifying the Control Scripts

## Automate Data Feed

The data feed export process should be scheduled to complete before the Endeca baseline runs.

The baseline can also be triggered if a file exists. This is also known as semaphore triggering. The process works by having the data export process drop off a file called “baseline.done” when the export process completes.

The baseline\_update.bat script can be called from a new script called “run\_complete\_refresh.bat”. This script can be scheduled to check for the “baseline.done” file every 15 minutes or so and if the file exists run the baseline update.

A scheduled task should be created to run every 15 minutes to. The task should execute the run\_complete\_refresh.bat script, which encompasses the logic to find the baseline.done file.

Basic Example

*if exist baseline.done (*

*call %~dp0load\_baseline\_test\_data.bat*

*call %~dp0baseline\_data\_ready\_flag.bat*

*del baseline.done*

*) else (*

*exit*

*)*

## Data Feed Location

The data feed should be loaded into a single directory. Thanx Media recommends using a script which pulls the data into the test\_data\baseline directory.

Script Location: load\_baseline\_test\_data.bat

*call %~dp0..\config\script\set\_environment.bat*

*copy* ***<data-directory>*** *%ENDECA\_PROJECT\_DIR%\test\_data\baseline\*

*copy %ENDECA\_PROJECT\_DIR%\test\_data\baseline\\* %ENDECA\_PROJECT\_DIR%\data\incoming*

*call %~dp0set\_baseline\_data\_ready\_flag.bat*

The %~dp0 (that’s a zero) variable when referenced within a Windows batch file will expand to the drive letter and path of that batch file.

## Archiving

After the feed has been copied to the test\_data\baseline directory, a script should archive the data feed when the run.py script runs. The script below writes the data files to the archive directory as a zip file, then deletes the files from the baseline directory.

Script Location: eodtools.py

*def archive\_baseline(self, delete=False):*

*backup\_date = datetime.today()*

*zip = zipfile.ZipFile(self.directory+'//archive/baseline-'+backup\_date.strftime("%Y%m%dT%H%M%S")+'.zip', 'w')*

*for name in glob.glob(self.directory+'//test\_data//baseline//\*'):*

*zip.write(name, os.path.basename(name), zipfile.ZIP\_DEFLATED)*

*if delete:*

*os.remove(name)*

## Data Checks

A script should be added to check if the data files exist as well as the validity of the data files before being indexed to the production engines. The script below checks if the data file exists and the size of the file. If the file passes, the script exits with a “True” status. If the file fails the check, the script exits with a “False” status

Script Location: run.py

*path = 'C:/Endeca/apps/GameStopUS/test\_data/baseline/'*

*log = 'C:/Endeca/apps/GameStopUS/logs/baseline\_update.out’*

*def sizeCheck(file,size):*

*try:*

*record = os.path.getsize(path+file)*

*print record*

*if record >= size:*

*checkR = True*

*else:*

*print "SEVERE small " + file*

*checkR = False*

*with open(log, 'a') as f:*

*f.write('SEVERE '+file+' File was too small. Baseline did not run.\n')*

*except WindowsError as Error:*

*checkR = False*

*print 'SEVERE Missing '+file+' File: Baseline Did Not Run!'*

*with open(log, 'a') as f:*

*f.write('SEVERE Missing ' +file+' Files!\n')*

*sizeCheck('items.txt',300000)*

## Email Configuration

A script should be added to the update process to send out an email if a baseline failure occurs. Baseline Failures consist of:

* The baseline failed to obtain the locks to run
* Failure in the Forge process
* Failure in the Dgidx process
* Failure in the Dgraph process

Script Location: eodtools.py and run.py

There are several pieces of the email notifications with functions that:

* generates the email notification
* runs the baseline update process and creates a log file
* check the status of the baseline update.

Email Notification Function

*def sendemail(self, from\_addr, to\_addr\_list, cc\_addr\_list,reply\_to\_addr,*

*failure,*

*smtpserver='smtp.pbs.org'):*

*header = 'From: %s\n' % from\_addr*

*header += 'reply-to: %s\n' % reply\_to\_addr*

*header += 'To: %s\n' % ','.join(to\_addr\_list)*

*header += 'Cc: %s\n' % ','.join(cc\_addr\_list)*

*baseline\_date = datetime.today()*

*update\_status = 'SUCCESS'*

*if failure:*

*update\_status = 'FAILURE'*

*subject = '%s [%s] Endeca Production Baseline Update Notification' % (update\_status, self.app\_name)*

*header += 'Subject: %s\n\n' % subject*

*message = """Dear %s,*

*This is a notification that an Endeca event has occurred.*

*Date: %s*

*Status: Baseline update %s*

*Account #: %s*

*Please review the baseline\_update.out log to determine where the error has occurred. This file can be found in the Endeca pbs folder (C:/Endeca/apps/pbs/logs)*

*""" % (self.app\_name, baseline\_date.strftime("%Y/%m/%d %I:%M:%S %p CDT"), update\_status.lower(), self.app\_id)*

*message = header + message*

*server = smtplib.SMTP(smtpserver)*

*server.starttls()*

*problems = server.sendmail(from\_addr, to\_addr\_list, message)*

*server.quit()*

*return problems*

Baseline Update

*def baseline\_update(self):*

*try:*

*print self.directory+'//control/baseline\_update'+self.postfix*

*subprocess.Popen(self.directory + '//control/baseline\_update'+self.postfix + ' > ' + self.directory + '//logs/baseline\_update.out').wait()*

*return True*

*except Exception:*

*print 'error running baseline\_update'+self.postfix*

*pass*

*def promote\_content(self):*

*try:*

*print self.directory+'//control/promote\_content'+self.postfix*

*subprocess.Popen(self.directory + '//control/promote\_content'+self.postfix + ' > ' + self.directory + '//logs/promote\_content.out').wait()*

*return True*

*except Exception:*

*print 'error running promote\_content'+self.postfix*

*pass*

Check for Errors

*def check\_erros(self):*

*log\_output = open(self.directory+'//logs/baseline\_update.out', 'r')*

*for line in log\_output.readlines():*

*if 'SEVERE' in line:*

*return True*

## IFCR Checks

A script should be added to check the status of the authoring assembler service before the promote\_content.bat script is executed. The promote\_content script promotes content and search configurations from the staging environment to the production environment.

The IFCR check will prevent an invalid configurations from being pushed to the live instances. The script should check for “refinementCrumbs” in the response (A valid response will always return “refinementCrumbs”.

Script Location:

*import urllib2*

*#Writes the the json source to a text file*

*site = 'http://<domain>/<client>/json/pages/browse'*

*status = open("status.txt", 'w')*

*status.write(urllib2.urlopen(site).read())*

*status.close()*

*#Checks if refinementCrumbs' is present in the status.txt file*

*if 'refinementCrumbs' in open('status.txt').read():*

*app.promote\_content()*

*else:*

*checkStatus = False*

*with open('C:/Endeca/apps/<client>/logs/baseline\_update.out', 'a') as f:*

*f.write('Authoring Assembler Service Invalid. EM configurations were not promoted to Live\n')*

# Modifying the Application Configuration Files

The standard script operations of the Deployment Template are sufficient to support the operational requirements of most projects. Some applications require customization to enable custom processing steps, script behavior, or even directory structure changes.

## Configuring the Assembler Files

1. Edit the LiveDgraphCluster.xml file in <app dir>/config/script to ensure the live Dgraph, the live cluster and post-startup script is correct for your environment.
2. Edit the LiveAppServerCluster.xml file in <app dir>/config/script to ensure that the application server clusters, the servers within each cluster, and the applications running on the servers are correct for your environment.
   1. For each server cluster, create an <app-server-cluster> element with an id attribute that corresponds to the cluster name. For example:

*<app-server-cluster id="LiveAppServerCluster">*

*</app-server-cluster>*

b) For each server within the cluster, create an <app-server> element with the following attributes:

* **id** — The name of the server.
* **hostName** — The DNS name or IP address of the server hosting the Assembler.
* **port** — The port on which the Assembler Web application is running.

For example:

*<app-server id="LiveDiscover" hostName="assemblerHost.example.com"*

*port="8006">*

*</app-server>*

c) For each application running on a given Assembler, create a <web-app> element with the following attributes:

* **id** — The name of the Assembler application.
* **contextPath** — The path to the application relative to the Assembler server.
* **sslEnabled** — Optionally, whether the application is SSL-enabled.

For example:

*<web-app id="DiscoverWebApp" contextPath="/discover" sslEnabled="true" />*

*<web-app id="DiscoverAsService" contextPath="/discoverAsService" />*

d) Add the <web-app> elements to their respective <app-server>s as referenced elements.

For example:

*<app-server id="LiveDiscover" hostName="assemblerHost" port="8006">*

*<web-app ref="DiscoverWebApp" />*

*<web-app ref="DiscoverAsService" />*

*</app-server>*

e) Add the <app-server> elements to their respective <app-server-cluster>s as referenced elements. For example:

*<app-server-cluster id="LiveAppServerCluster">*

*<app-server ref="LiveDiscover" />*

*</app-server-cluster>*

**3**. Edit the AssemblerConfig.xml file in <app dir>/config/script to ensure that it references theapplication server clusters that are correct for your environment.

**4**. Edit the WorkbenchConfig.xml file in <app dir>/config/script to ensure the Workbench Managerand IFCR components are correct for your environment.

**5**. Edit the UsageCollectionConfig.xml file in <app dir>/config/script to ensure that the Dgrap**h** clusters and application server clusters from which usage is collected are correct for your environment.

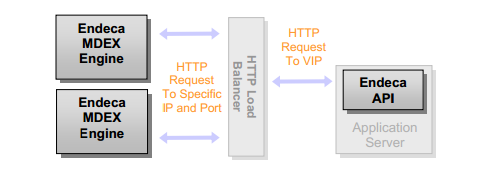
**6**. If necessary, edit the ReportGeneration.xml file in <app dir>/config/script. This file does notusually require any modifications.

# Supplemental Infrastructure Tasks

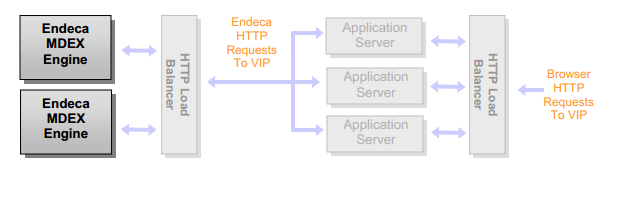
## Engine Load Balancer

The MDEX Engine functions very similarly to a web server in terms of network traffic: It simply accepts HTTP requests on a specified port, and returns results to the caller. This behavior allows for standard web load balancing techniques to be applied. In particular, all of these techniques will introduce a Virtual IP address, which will accept requests from the application server, and route the requests to the MDEX Engine it determines best suited to handling the request.

The MDEX Engine functions very similarly to a web server in terms of network traffic: It simply accepts HTTP requests on a specified port, and returns results to the caller. This behavior allows for standard web load balancing techniques to be applied. In particular, all of these techniques will introduce a Virtual IP address, which will accept requests from the application server, and route the requests to the MDEX Engine it determines best suited to handling the request.



It is important to realize that the load balancing scheme described above is no different than the solution most sites implement for balancing external traffic to application servers. The configuration process should therefore be familiar. In many cases, if enough ports are available, the same physical hardware can even be used, provided any firewalls do not restrict this loop-back.



## Load Balancing Scripts

<client> currently uses the DgraphPreShutdownScript /DgraphPostStartupScript scripts to communicate with their F5 load balancer during the baseline update. This ensures that the dgraphs will not be sent requests while they are down or starting. These scripts should be used going forward and are a part of Thanx Media’s best practices.

*<!--*

*########################################################################*

*# Scripts to distribute index to dgraph servers, then update dgraphs*

*# with the distributed index. This script can be called to update or*

*# refresh the index of the dgraph cluster in case a server fails, a*

*# new dgraph is added, or the index simply needs to be updated.*

*#*

*-->*

*<script id="DgraphPreShutdownScript">*

*<bean-shell-script>*

*<![CDATA[*

*id = invokingObject.getElementId();*

*hostname = invokingObject.getHost().getHostName();*

*port = invokingObject.getPort();*

*log.info("Removing dgraph with id " + id + " (host: " + hostname + ", port: " + port + ") from load balancer cluster.");*

*if (BigIPControl.DisablePoolMember("${gamestop.f5.host}", "${gamestop.f5.adminuser}", "${gamestop.f5.adminpassword}", "${gamestop.f5.pool}", hostname, port) == false)*

*Mailer.sendUrgentSimpleMsg("${gamestop.environment.name} - Endeca Baseline Warning: " + ITLHost.appName + " on " + ITLHost.hostName,*

*"Unable to remove dgraph with id " + id + " (host: " + hostname + ", port: " + port + ") from load balancer cluster.");*

*]]>*

*</bean-shell-script>*

*</script>*

*<script id="DgraphPostStartupScript">*

*<bean-shell-script>*

*<![CDATA[*

*id = invokingObject.getElementId();*

*hostname = invokingObject.getHost().getHostName();*

*port = invokingObject.getPort();*

*log.info("Adding dgraph with id " + id + " (host: " + hostname + ", port: " + port + ") to load balancer cluster.");*

*if (BigIPControl.EnablePoolMember("${gamestop.f5.host}", "${gamestop.f5.adminuser}", "${gamestop.f5.adminpassword}", "${gamestop.f5.pool}", hostname, port) == false)*

*Mailer.sendUrgentSimpleMsg("${gamestop.environment.name} - Endeca Baseline Warning: " + ITLHost.appName + " on " + ITLHost.hostName,*

*"Unable to add dgraph with id " + id + " (host: " + hostname + ", port: " + port + ") to load balancer cluster.");;*

*]]>*

*</bean-shell-script>*

*</script>*

 Assembler Web Service Load Balancer

A load balancer in front of the assembler services is highly recommended. Thanx Media recommends setting up an F5 hardware load balancer.

## DNS Configuration

After creating the web service load balancer, log into your DNS client and create the necessary entries for gamestop.com to point to the assembler load balancer.

## Health Checks

### Indexer

Create a node in your health monitoring system for the server which will host the ITL process. It should monitor the server’s CPU, network, load, memory, etc.

### Live MDEX

Create a node in your health monitoring system for each Production MDEX engine. The service should check the status with the admin?op=ping command. This will check if the MDEX engines are responding and can alert you if one engine is down for extended period of time. You can also add a check through the Load Balancer to also check the status of the Load Balancer.

A quick way of checking the availability of a Dgraph is by accessing the URL as described below. To check the whether a Dgraph is running:

* For a Dgraph: [http://DgraphServerNameOrIP:DgraphPort/admin?op=ping](http://dgraphservernameorip:DgraphPort/admin?op=ping)

The Dgraph quickly returns a lightweight HTML response page with the following content: dgraph host:port responding at date/time

### Live Web Services

# Baseline Update Process

## Forge

Forge is the data processing program that transforms your source data into standardized, tagged

Endeca records. Forge loads and processes one source record at a time, in sequential order. When Forge loads a source record, it transforms the record into a series of property/property value pairs. After a source record has been standardized, Forge maps the record’s source properties to dimensions and Endeca properties. After all the source records have been mapped, the Forge program writes its finished data.

The finished data consists of:

• The Endeca records along with their tagged dimension value IDs and Endeca properties.

• The names and IDs for each dimension and dimension value, along with any dimension hierarchy

**Directory**

data/forge\_output

## Dgidx

Dgidx is the indexing program that reads the tagged Endeca records that were prepared by Forge and creates the proprietary indices for the Endeca MDEX Engine.

**Directory**

data/dgidx\_output

## Dgraph

The Dgraph is the name of the process for the MDEX Engine, which is the query engine that provides the backbone for all Endeca solutions. The Dgraph uses proprietary data structures and algorithms that allow it to provide real-time responses to client requests. Because the Dgraph is key to every Endeca implementation, its performance is critical.

**Directory**

data/dgraphs

## Promote Content

Promotes Content pushes the Experience Manager/Workbench configurations from authoring to the live environments.

Experience Manager queries the MDEX Engine for record and dimension information that a content administrator can use to configure dynamic content. Examples include:

• specifying a navigation state as part of a location trigger

• selecting records or a navigation state for content spotlighting

• selecting records or dimension values for boost and bury

Workbench publishes configuration to the MDEX Engine, including:

• Rules

• Phrases

• Keyword redirects

• Thesaurus entries

# Troubleshooting Baseline Failures

## Forge

Location: C:\Endeca\apps\<client>\logs\forges\Forge

File: Forge.log

Example Forge Failure

*SEVERE: Batch component 'Forge' failed. Refer to component logs in C:\Endeca\apps\<client>\logs\forges.\logs\forges\Forge on host ITLHost.*

*Occurred while executing line 23 of valid BeanShell script:*

*[[*

*20|*

*21| // archive logs and run ITL*

*22| Forge.archiveLogDir();*

*23| Forge.run();*

*24| Dgidx.archiveLogDir();*

*25| Dgidx.run();*

*26|*

*]]*

Check the Forge.log file for more information on the failure.

## Dgidx

Location: C:\Endeca\apps\<client>\logs\forges\Dgidx

File: Dgidx.log

Example Dgidx Failure

*SEVERE: Batch component 'Dgidx' failed. Refer to component*

*logs in /usr/local/endeca/[version]/endeca/project/sample*

*/control/.././logs/dgidxs/Dgidx on host ITLHost.*

*Occurred while executing line 32 of valid BeanShell script:*

*[[*

*29| Forge.archiveLogDir();*

*30| Forge.run();*

*31| Dgidx.archiveLogDir();*

*32| Dgidx.run();*

*33|*

*34| // distributed index, update Dgraphs*

*35| DistributeIndexAndApply.run();*

*]]*

Check the Dgidx.log file for more information on the failure.

## Dgraph

Location: C:\Endeca\apps\<client>\logs\forges\Dgraph

File: DgraphA.log

Example Dgraph Failure

*SEVERE: Error communicating with EAC agent while starting utility 'copy\_index\_to\_host\_LiveMDEXHostA\_DgraphA'.*

*Occurred while executing line 7 of valid BeanShell script:*

*[[*

*4| AuthoringDgraphCluster.copyIndexToDgraphServers();*

*5| AuthoringDgraphCluster.applyIndex();*

*6| LiveDgraphCluster.cleanDirs();*

*7| LiveDgraphCluster.copyIndexToDgraphServers();*

*8| LiveDgraphCluster.applyIndex();*

*9|*

*10|*

*]]*

Check the DgraphA.log file for more information on the failure.

## Other

Failed to obtain lock

*INFO: Checking definition from AppConfig.xml against existing EAC provisioning.*

*INFO: Definition has not changed.*

*INFO: Starting baseline update script.*

*WARNING: Failed to obtain lock.*

Solution

1. Go to the .<AppDir>/control/ directory.
2. Run runcommand.bat LockManager releaseLock update\_lock

***OR***

1. Run eaccmd.bat remove-all-flags --app <your application>

# Performance Tuning

#### Using the Eneperf Tool

Eneperf is a performance testing tool that is included in your Endeca installation. This section describes how to use Eneperf.

#### About Eneperf

Eneperf is a performance, analytics and debugging tool that can measure throughput to help you identify system bottlenecks. Eneperf makes HTTP queries against the MDEX Engine (Dgraph) based on your MDEX Engine request logs and gathers the resulting statistics, without processing the results in any way.

Because Eneperf is lightweight, it has a very slight impact on performance. In most cases, it can be run on the same machine as the Dgraph being tested. It can also be run on a remote machine.

Eneperf drives a substantial load at the MDEX Engine and reveals how many operations per second the MDEX Engine responds with. Eneperf lets you measure both query latency and throughput. You specify the log file and specify to Eneperf how many times to run through it, as well as the number of client connections to simulate.

* eneperf –v <host> <port> <queryFile> <threads> <iterations>
* example - *eneperf –v gamestop.endecaondemand.net 80 DgraphA.out 1 1*

**Example of Eneperf ready queries (DgraphA.out):**

/graph?node=0&group=0&offset=0&nbins=15&attrs=TypeAheadAlpha+GAL\*|mode+MatchAll&filter=SearchWarehouse

/graph?node=0&group=0&offset=0&nbins=15&attrs=TypeAheadNumeric+08686113\*|mode+MatchAll&attrs=IsStockedInRSC+1|mode+MatchAll&filter=SearchWarehouse%3aM&irversion=601

/search?terms=SEAL&node=0&rank=0&offset=0&model=1000&irversion=601

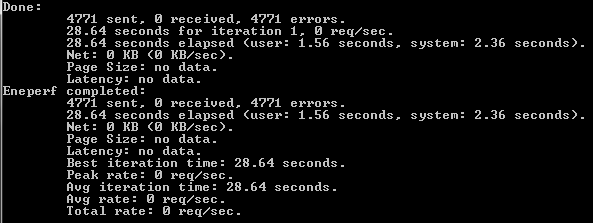
/graph?node=0&group=0&offset=0&nbins=15&attrs=TypeAheadAlpha+rad+rags\*|mode+MatchAll&attrs=IsStockedInRSC+1|mode+MatchAll&filter=SearchWarehouse%3aP&irversion=601

/search?terms=emergen&node=0&rank=0&offset=0&model=1000&irversion=601

/search?terms=SWIMMING+POOL+VACUUM+HEAD&node=0&rank=0&offset=0&model=1000&irversion=601

/search?terms=264253&node=0&rank=0&offset=0&model=1000&irversion=601

**Example of Eneperf Results:**



#### Identifying problems with resource usage by the application

Use the following recommendations to identify performance problems associated with resource usage.

Isolate performance testing for those parts of the application that specifically use the Endeca MDEX Engine from testing for other parts of the application. In other words, measure the performance of those parts of the application that use the Endeca MDEX Engine separately from the performance of those parts that use other software that may cause performance problems, such as a relational database. For example, if the latency is high, consider testing the interaction of the application with the database, if you are using one.

If you are sending a lot of requests to the front-end application and performance is slow but the MDEX Engine servers are idle, the front-end application and its resource usage is probably the issue. There are two possible fixes: you can reduce consumption of resources by the application by reviewing your coding practices for the front-end application, or add resources.

#### Recommendations for identifying network problems

Often the diagnosis of slow performance comes from a query load played against the front-end application. The front-end application, or the configuration of its application server, may be the reason for the poor performance.

Alternatively, the network may be the problem, although this is less likely. (In the case of a Dgraph, unlike an Agraph, it is unusual for the network to be the bottleneck.) To identify whether the network is a performance issue:

Compare Eneperf performance on the local host and a remote host. First, run Eneperf against the Dgraph on the Dgraph machine. Next, run the same Eneperf against the same Dgraph, but from the front-end machine (if possible), or somewhere on the other side of the network. If the difference is negligible, the network is not a problem. If Eneperf across the network is slow, you need to consider both the network itself and the application configuration.

Alternatively, you can run the Request Log Analyzer and compare the “Round-Trip Response Time” with the “Engine-Only Processing Time”. If “Round-Trip Response Time” is long but the “Engine-Only Processing Time” is short, this can indicate a network problem or a configuration of an application server for the front-end application.

# Reporting

**Log Server**

The Endeca Log Server is a stand-alone server that translates application logging requests into flat log files. These files are later streamed through the Report Generator and transformed into configurable reports.

The log server can be started with the following command or from the workbench:

* runcommand.bat LogServer start

Once the process completes the log server can be checked by going to this URL:

* http://<LOGSERVER\_HOST>:<LOGSERVER\_PORT>/stats

The files from the logserver are written to apps/<client>/logs/logserver\_output. We recommend checking the contents monthly and archiving unneeded logs. The logserver can also be configured to gzip the output. The example below from the AppConfig file named “ReportGeneration.xml” shows the default configuration option. Changing it to <gzip>true</gzip> enables compression of the logs.

<!--

########################################################################

# LogServer - Aggregates application request logging information to

# generate Workbench reports.

#

-->

<logserver id="LogServer" host-id="ReportGenerationHost" port="${mdex.port}">

<properties>

<property name="numLogBackups" value="10" />

<property name="targetReportGenDir" value="./reports/input" />

<property name="targetReportGenHostId" value="ReportGenerationHost" />

</properties>

<log-dir>./logs/logservers/LogServer</log-dir>

<output-dir>./logs/logserver\_output</output-dir>

<startup-timeout>120</startup-timeout>

**<gzip>false</gzip>**

</logserver>

**Reporting**

The Report Generator reads the log files created by the Log Server and creates XML-based or HTML-based reports that can be displayed on the View Reports tool in Workbench. These reports allow you to look at what has happened on your Web site on a daily or weekly basis. \

The reporting can be tweaked by changing the report setting located here:

**config\report\_templates\report\_settings.xml**

Here is a sample configuration:

**<report\_settings>**  
**<top\_n report\_item="top\_search\_keys" n="50, 100, 200, 300" initial\_value="50" />**  
**<top\_n report\_item="top\_search\_terms" n="50, 100, 200, 300" initial\_value="50" />**  
**<top\_n report\_item="top\_search\_terms\_w\_results" n="50, 100, 200, 300" initial\_value="50" />**  
**</report\_settings>**

**Modify the above with the values below as needed.**

Here is a list of IDs for report sections and the report items they contain,   
which you can use in <disable> and <top\_n> elements. (This list is for reference  
purposes only - changes to this list will have no effect.)

**req\_and\_sess**  
request\_and\_session\_summary  
num\_requests\_by\_hour  
num\_requests\_by\_date  
num\_unique\_sessions\_by\_hour  
num\_unique\_sessions\_by\_date  
requests\_by\_type  
num\_search\_only\_requests\_by\_num\_results  
num\_nav\_only\_requests\_by\_num\_results  
num\_search\_then\_nav\_requests\_by\_num\_results  
dimension\_values  
**conv\_rate\_summ**  
conversion\_rate  
conversion\_rate\_by\_hour  
conversion\_rate\_by\_date  
conversion\_rate\_by\_type  
conversion\_rate\_top\_by\_search\_terms  
conversion\_rate\_by\_top\_dimensions  
conversion\_rate\_by\_top\_merch\_rules  
**abtest**  
conversion\_rate\_by\_top\_abtest\_rules  
**search**  
top\_search\_keys  
top\_search\_terms  
top\_search\_terms\_w\_results  
**nav**  
top\_dimensions  
top\_dvals  
top\_dimension\_pairs  
top\_dval\_pairs  
top\_dimension\_triples  
top\_dval\_triples  
**recs**  
top\_records  
**noresults**   
spell\_correct\_alt\_sugg\_summary  
top\_search\_terms\_no\_results  
top\_autocorrected\_terms  
top\_search\_terms\_dym\_engaged  
**merch**  
business\_rules\_summary  
top\_merch\_rules  
top\_records\_selected\_through\_merch  
top\_dvals\_no\_merch  
top\_search\_terms\_no\_merch  
**dimsearch**   
dim\_search\_summary  
top\_dimension\_search\_results\_selected  
**sort**  
sorting\_summary  
top\_sort\_keys

# Staging/Production Instances

The <client> team requested information on how to have separate Staging and Production instances. The out of the box Deployment template combines Authoring and Production. Here are steps to keep them separate:

1. Staging: Baseline update
2. Staging: Site export
3. Prod: Site import of the Staging site
4. ~~Staging: Promote content (which points to the Prod authoring instance)~~
5. Prod: Baseline update
6. Prod: Promote content (which points to the Prod live instance)

## Deployment Template

In order to set up a staging to production instance, two apps will need to be deployed via the deployment template. Ex: <client>Stage and <client>Prod.

## Assembler Modifications

Create two apps in the \ToolsAndFrameworks\11.1.0\reference\ directory.

* <client>-stage-authoring-service – In the assembler.properties, set the mdex.port to the authoring port for the Staging instance

*mdex.port=${mdex.port}*

* <client>-prod-live-service – In the assembler.properties, set the mdex.port to the live port for for the Production instance.

*mdex.port=${mdex.port}*

Create two server entries in \11.1.0\server\workspace\conf\Standalone\localhost\

**<client>-stage-authoring-service**

*<Context*

*path="/<client>-stage-authoring-service"*

*docBase="${catalina.base}/../../reference/<client>-stage-authoring-service"*

*privileged="false">*

*</Context>*

**gamestop-prod-authoring-service**

*<Context*

*path="/<client>-prod-authoring-service"*

*docBase="${catalina.base}/../../reference/ <client>-prod-authoring-service"*

*privileged="false">*

*</Context>*

**gamestop-prod-live-service**

*<Context*

*path="/<client>-prod-live-service"*

*docBase="${catalina.base}/../../reference/ <client>-prod-live-service"*

*privileged="false">*

*</Context>*

## Application Modifications

### <client>Stage

**LiveAppServerCluster**

Set the context-path for the web app to <client>-prod-authoring-service

*<web-app id="DiscoverAssemblerApp1" contextPath="/<client>-prod-authoring-service " />*

**LiveDgraphCluster**

Set the Live Dgraph port to the Staging live port(which was set in the deployment template)

*<!--*

*########################################################################*

*# Live Dgraph Process Definitions*

*#*

*-->*

*<dgraph id="DgraphA1" host-id="LiveMDEXHostA" port="${mdex.port}"*

*post-startup-script="LiveDgraphPostStartup">*

*<properties>*

*<property name="restartGroup" value="1" />*

*<property name="DgraphContentGroup" value="Live" />*

*</properties>*

*<log-dir>./logs/dgraphs/DgraphA1</log-dir>*

*<input-dir>./data/dgraphs/DgraphA1/dgraph\_input</input-dir>*

*<update-dir>./data/dgraphs/DgraphA1/dgraph\_input/updates</update-dir>*

*</dgraph>*

### <client>Prod

**LiveAppServerCluster**

Set the context-path for the web app to <client>-prod-live-service

*<web-app id="DiscoverAssemblerApp1" contextPath="/<client>-prod-live-service " />*

**LiveDgraphCluster**

Set the Live Dgraph port to the Staging live port(which was set in the deployment template)

*<!--*

*########################################################################*

*# Live Dgraph Process Definitions*

*#*

*-->*

*<dgraph id="DgraphA1" host-id="LiveMDEXHostA" port="${mdex.port}"*

*post-startup-script="LiveDgraphPostStartup">*

*<properties>*

*<property name="restartGroup" value="1" />*

*<property name="DgraphContentGroup" value="Live" />*

*</properties>*

*<log-dir>./logs/dgraphs/DgraphA1</log-dir>*

*<input-dir>./data/dgraphs/DgraphA1/dgraph\_input</input-dir>*

*<update-dir>./data/dgraphs/DgraphA1/dgraph\_input/updates</update-dir>*

*</dgraph>*

## Prod Push Script

A python script will be used to do the processing to push the staging Experience Manager configurations to the production instance.

This script will run when the user is ready to view the Experience Manager Changes in the Production instance. It will first update the authoring engine and assembler service with the latest changes, then from there a Production baseline can be run to push the changes Live.

Code Example

*import os*

*import glob*

*import shutil*

*import subprocess*

*os.system(siteexport)*

*control = 'C:/Endeca/apps/discoverAuthoring/control/'*

*copyDir = 'C:/Endeca/apps/discoverLive/control/'*

*siteimport = 'C:/Endeca/apps/discoverLive/control/import\_site.bat '*

*siteexport = 'C:/Endeca/apps/discoverAuthoring/control/export\_site.bat '*

*promote = 'C:/Endeca/apps/discoverAuthoring/control/promote\_content.bat '*

*prepend = 'discoverAuthoring'*

*outfile = 'discoverAuthoringOut.xml'*

*finalfile = 'discoverAuthoring.xml'*

*for file in os.listdir(control):*

*if file.startswith(prepend):*

*shutil.copy2(control+file,copyDir+outfile)*

*with open(copyDir+finalfile, "wt") as fout:*

*with open(copyDir+outfile, "rt") as fin:*

*for line in fin:*

*fout.write(line.replace(prepend, 'discoverLive'))*

*os.remove(copyDir+outfile)*

*os.system(siteimport+copyDir+finalfile)*

*os.system(promote)*

## Pipeline Modifications for Environment Portability

The gamestop pipeline.epx file has been configured to pull content from a CAS record store. The record adapter has configurations set for host, port and other configurations

*<RECORD\_ADAPTER COL\_DELIMITER="" DIRECTION="INPUT" FILTER\_EMPTY\_PROPS="FALSE" FORMAT="JAVA\_ADAPTER" FRC\_PVAL\_IDX="FALSE" JAVA\_CLASSNAME="com.endeca.itl.recordstore.forge.RecordStoreSource" JAVA\_CLASSPATH="E:\Endeca\CAS\2.2.1\lib\recordstore-forge-adapter\recordstore-forge-adapter-2.2.1.jar" MULTI="FALSE" NAME="BazaarvoiceReview-en-US" PREFIX="" REC\_DELIMITER="" REQUIRE\_DATA="FALSE" ROW\_DELIMITER="" STATE="FALSE" URL="">*

*<COMMENT></COMMENT>*

***<PASS\_THROUGH NAME="HOST">casservice.gamestop.com<PASS\_THROUGH>***

***<PASS\_THROUGH NAME="PORT">8500</PASS\_THROUGH>***

***<PASS\_THROUGH NAME="INSTANCE\_NAME">BazaarvoiceReview-en-US-rs</PASS\_THROUGH>***

***<PASS\_THROUGH NAME="READ\_TYPE">BASELINE</PASS\_THROUGH>***

*</RECORD\_ADAPTER>*

If the pipeline is copied to an environment where the CAS record store was named differently, the forge process would fail. To prevent that from happening the pipeline.epx will be modified to change the PASS\_THROUGH values. The new values will be placeholder variables that will be changes to environment specific values when the baseline update runs.

*<RECORD\_ADAPTER COL\_DELIMITER="" DIRECTION="INPUT" FILTER\_EMPTY\_PROPS="FALSE" FORMAT="JAVA\_ADAPTER" FRC\_PVAL\_IDX="FALSE" JAVA\_CLASSNAME="com.endeca.itl.recordstore.forge.RecordStoreSource" JAVA\_CLASSPATH="E:\Endeca\CAS\2.2.1\lib\recordstore-forge-adapter\recordstore-forge-adapter-2.2.1.jar" MULTI="FALSE" NAME="BazaarvoiceReview-en-US" PREFIX="" REC\_DELIMITER="" REQUIRE\_DATA="FALSE" ROW\_DELIMITER="" STATE="FALSE" URL="">*

*<COMMENT></COMMENT>*

***<PASS\_THROUGH NAME="HOST">@@HOST@@<PASS\_THROUGH>***

*<PASS\_THROUGH NAME="PORT">8500</PASS\_THROUGH>*

*<PASS\_THROUGH NAME="INSTANCE\_NAME">BazaarvoiceReview-en-US-rs</PASS\_THROUGH>*

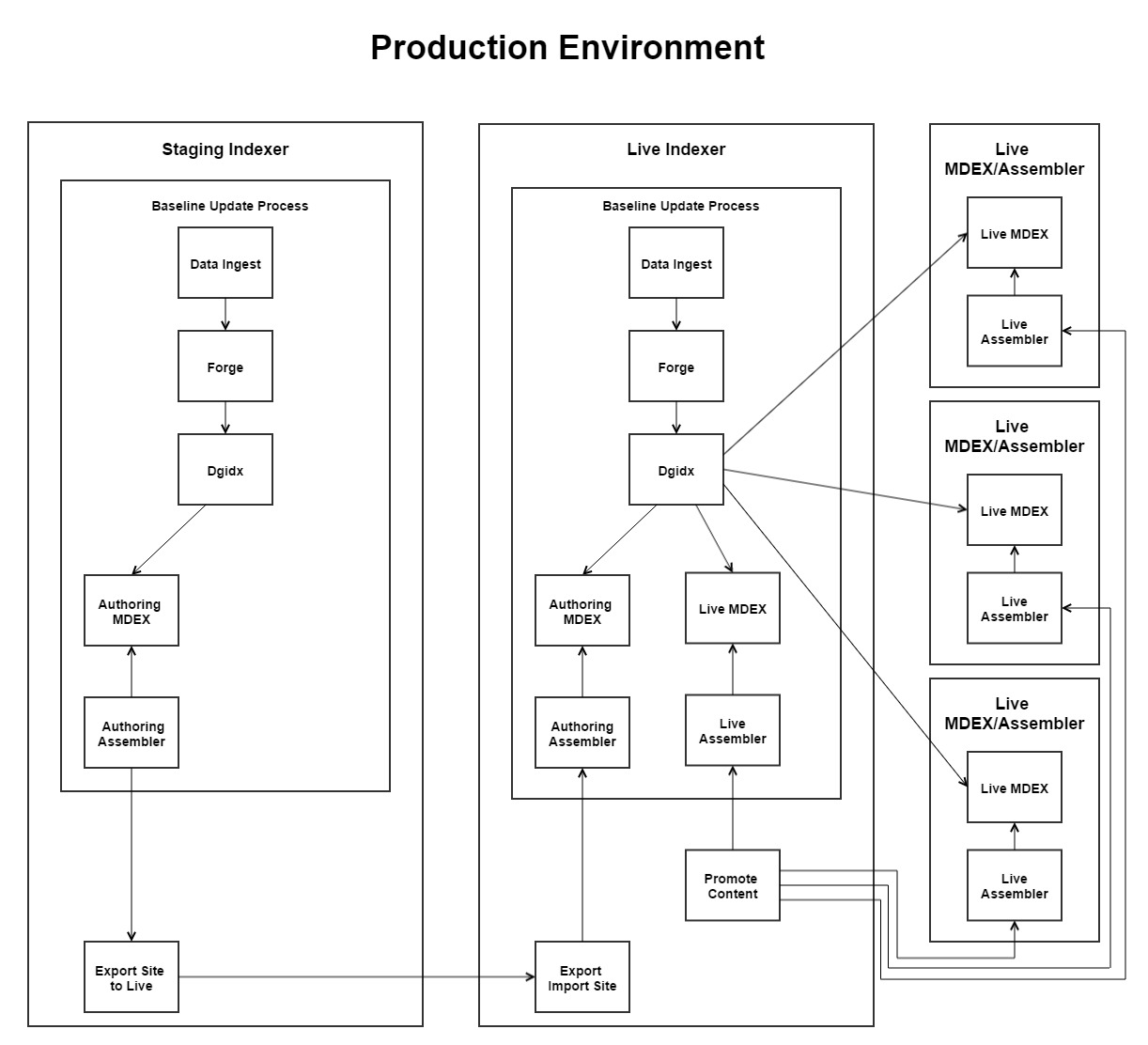
*<PASS\_THROUGH NAME="READ\_TYPE">BASELINE</PASS\_THROUGH>*

*</RECORD\_ADAPTER>*

A script should be created, that will read the environment variables for the CAS configurations for the current environment and change the placeholder variables in the pipeline.epx file.

# Infrastructure

# C:\Daniel\apps\gamestop\gamestopInfrastructure.jpg



# Indexing Server Installs

The below installations are required to implement an Endeca application. Install the following in order as they appear. Copy the install files from \\172.31.205.93\E$\Installation\EndecaInstalls

1. MDEX
2. Platform Services
3. ToolsAndFrameworks
4. CAS
5. Developer Studio
6. Python\*
7. Apache Tomcat\*\*

*Note: Be sure to change the installation to the*[*E:\*](file:///E:\)*drive.*

*\* Python will be used to run the Thanx Media Best Practice scripts.*

\*\* *Install on the Live Indexer & Live Webservers*

## Installing the MDEX Engine on Windows

Oracle recommends installing a machine-wide installation (step 6) assuming that administrator permissions are available for the user running the installer program.

**To install the Endeca MDEX Engine on Windows:**  
1. Extract the MDEX Engine package to a local directory.  
The name of the extracted installer file is mdex\_<version>\_x86\_64pc-win32.exe.  
3. Double-click the installer file mdex\_<version>\_x86\_64pc-win32.exe to start the wizard.  
4. Click Next to begin the installation process.  
5. In the Copyright and Legal screen, click Next.  
6. In the Select Program Folder screen, do the following:

• Accept the default value for Program Folder.  
• Select the Anyone who uses this computer (all users) button.  
• Click Next.

7. Select an installation location of E:\Endeca\MDEX\6.5.0 and click Next.  
8. Click Finish.

Installing Platform Services on Windows  
In this procedure, you have to specify a user name that runs the Endeca services. The user name should be the same admin user that ran the MDEX Engine installation. It does not need to be an endeca user. The user name cannot be null or have a blank password. At the end of the installation, you have to restart the machine.

If you have uninstalled Platform Services in the current session, restart your computer before you begin  
reinstalling Platform Services.  
Here again, accept all the installation defaults unless you must modify them. Do not install the Endeca Control System in the Custom Setup screen.

**To install the Endeca Platform Services on Windows:**  
1. Extract the Endeca Platform Services package to a local directory. The name of the extracted installer file is platformservices\_<version>\_x86\_64pc-win32.exe.  
3. Double-click the installer file to start the wizard.  
4. When the Endeca Platform Services Setup Wizard screen appears, click Next.  
5. Read the copyright information and click Next.  
6. In the Select Installation Type screen, select Anyone who uses this computer (all users) and click Next.  
7. In the Destination folder screen, select an installation location of **E:\Endeca\PlatformServices** installation directory and then click Next.  
8. In the Custom Setup screen, leave all the defaults selected and then click Next.

*Note: The Endeca Control System is not selected to install. This is Ok. You do not need it for new*

*applications. There is also a Endeca Reference Implementation that is not selected by default.*

9. In the Endeca Services Information screen, **enter the user name (endeca user with admin rights)**, domain name, and password to use when launching the Endeca HTTP Service and then click Next.  
10. In the Endeca Application Controller Service Information screen, do the following and then click Next:

• Accept the default EAC service port of 8888.  
• Accept the default EAC service shutdown port of 8090.  
• Specify an absolute path to the MDEX Engine root directory (for example, E:\Endeca\MDEX\6.5.0).

11. In the Ready to install the program screen, confirm the settings you selected in previous screens and then click Install.  
12. When the installation is complete, click Finish to exit the wizard.  
13. Click Yes to **restart** the computer.  
Restarting sets the Endeca environment variables (such as ENDECA\_ROOT) correctly and starts the Endeca  
HTTP Service.

Installing Tools and Frameworks on Windows  
**To install the Oracle Endeca Tools and Frameworks:**

1. Extract the Tools and Frameworks package to a local directory.

2. Navigate to the \cd\Disk1\install directory and double-click the **setup.exe** installer file to start the wizard. The Welcome screen displays.  
3. Click Next to begin the installation process.  
4. Read the License Agreement and click I accept the License Terms and Export Restrictions.  
5. If this is the first product that you have installed on this machine using the Oracle Universal Installer, the Specify Inventory directory screen appears. Specify the directory where the Oracle Universal Installer should place inventory files and directories. Oracle recommends that you accept the default path. The Installer uses the Oracle Inventory directory to store inventory information in files and subdirectories. This directory contains permanent and product specific files. Ensure that the files in this folder are not deleted or modified, as this would make patching or upgrading impossible. Click Next  
6. Choose the installation type, and click Next.  
The Complete Installation includes the reference application; the Minimal Installation does not.  
7. Specify a name and a path where you want to install Tools and Frameworks. Change the Name to ToolsAndFrameworks and Oracle recommends **E:\Endeca\ToolsAndFrameworks** as your path.

This is the path that Oracle refers to as your default path throughout the Oracle Endeca Commerce documentation. Click Next.  
8. Enter an admin password and confirm the password.  
The admin user is the default administrator user for Tools and Frameworks. Click Next  
9. When the Summary screen appears, click Install.  
10. When the End of Installation screen appears, click Exit.  
11. In Windows Explorer, navigate to the Tools and Frameworks installation directory.  
12. Install the Endeca Tools Service by double clicking server**\bin\install\_service.bat.**  
This creates the Endeca Tools Service and configures it to run under the current user profile. The user

running install\_service.bat and the Endeca Tools Service must have administrator privileges.  
13**. Start the** **Endeca Tools Service** by doing the following:

• Start the Microsoft Services console.

• Select the Endeca Tools Service from the list of services.

• Click Start Service.

16. Execute ToolsAndFrameworks/11.0.0/server**/bin/run.bat**

Installing CAS on Windows  
Here again, accept all the installation defaults unless you must modify them.  
To install CAS on Windows:  
1. Download the Content Acquisition System package from the Oracle Software Delivery Cloud.  
2. Extract the Content Acquisition System package to a local directory. The name of the extracted installer file is cas-<version>\_x86\_64pc-win32.exe.  
3. Double-click the installer file cas-<version>\_x86\_64pc-win32.exe to start the wizard.  
4. Click Next to begin the installation process.  
5. In the Important Information screen, read the copyright and then click Next.  
6. In the Custom Setup screen, select both program features and then click Next.  
7. In the Destination Folder screen, accept the default location of **E:\Endeca\CAS** and then click Next.  
8. In the Endeca CAS Service Information screen, specify the user name, password, and domain information for the admin user who will run the CAS Service and then click Next.  
9. In the CAS Server Information screen, accept the default values for the CAS Server port(8500) and CAS Server shutdown port (8506).  
10. In the Completing the Setup Wizard screen, click Next.  
The CAS Service starts automatically.

Installing Developer Studio  
Developer Studio is only available for Windows.  
To install Developer Studio:

1. Download the Endeca Developer Studio package from the Oracle Software Delivery Cloud.  
2. Extract the Endeca Developer Studio package to a local directory.  
The name of the extracted installer file is dstd\_version\_i86pc-win32.exe.  
3. Double-click the installer file: dstd\_version\_i86pc-win32.exe.  
4. Click Next to begin the installation wizard.  
5. On the Copyright and legal screen, click Next.  
6. In the License Agreement screen, select I accept the terms in the license agreement, then click Next.  
7. In the Destination Folder screen, either accept the default location or click Change and browse to the  
directory where you want to install the Endeca software. Change the location to (**E:\Endeca\DeveloperStudio**). When you have finished, click Next.  
8. In the Ready to Install the Program screen, click Install.  
9. When the installation is complete, click Finish.

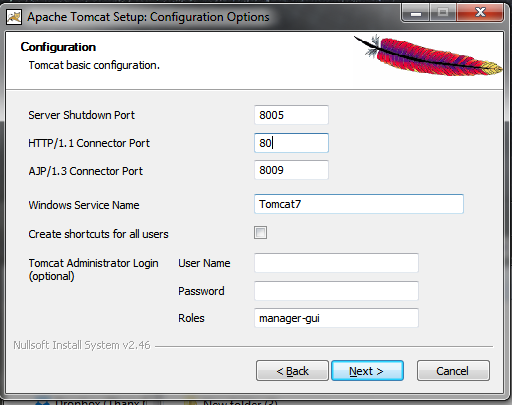
Reference <http://docs.oracle.com/cd/E51272_02/Common.110/pdf/GettingStarted.pdf> for full details on installation.

## Installing Python

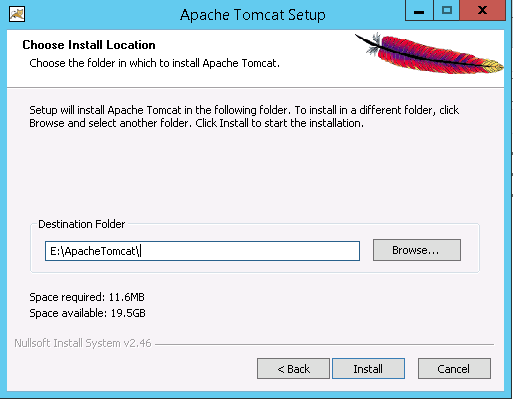
1. Run python-2.7.8.exe on the indexing server
2. Change the destination to **E:\Python27\**
3. Follow the on-screen install instructions
4. Click Finish
5. **Set the python environment variable**:
   * PATH = E:\Python27;

Installing Apache Tomcat (Live)

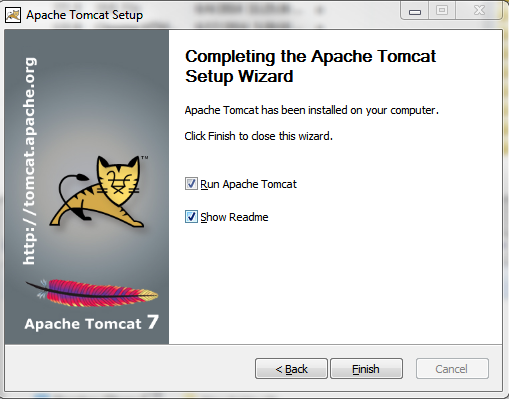
1. Run the Apache executable file.
2. Change the Connector Port to: 80



1. Change the Apache Tomcat destination path to E:\ApacheTomcat\



1. Click Finish (With “Run Apache Tomcat” checked off)



# Live MDEX/Webserver Installs

The Endeca software below is required to set up a Live production server. Copy the install files from \\172.31.205.93\E$\Installation\EndecaInstalls\.

1. MDEX (install same as above)
2. Platform Services (install same as above)

*Note: Server will need to be restarted*

Reference <http://docs.oracle.com/cd/E51272_02/Common.110/pdf/GettingStarted.pdf> for full details.

1. Apache-Tomcat-7.x.x (install same as above)

**Stats Page: http://<host>:<port>/<client>-service/admin?op=stats**

# Setting the Environment Variable

The GameStop Environment Variable will control which environment properties and set\_environment file is used to initialize and run the Endeca <client> app. The Environment Variables below need to be set for each indexing server if not set already.

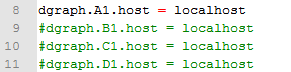
|  |  |  |
| --- | --- | --- |
| Environment | Environment Variable Name | Value |
| Dev Staging Indexer | ENDECA\_ENV | DEVSTAGE\_ |
| Dev Live Indexer | ENDECA\_ENV | DEVLIVE\_ |
| QA Staging Indexer | ENDECA\_ENV | QASTAGE\_ |
| QA Live Indexer | ENDECA\_ENV | QALIVE\_ |
| Production Staging Indexer | ENDECA\_ENV | PRODSTAGE\_ |
| Production Live Indexer | ENDECA\_ENV | PRODLIVE\_ |

**Note: For the examples below we will be referencing the QA Live servers and the Production Live servers**

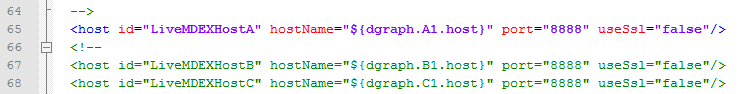
# Move <client> app from Environment to Environment

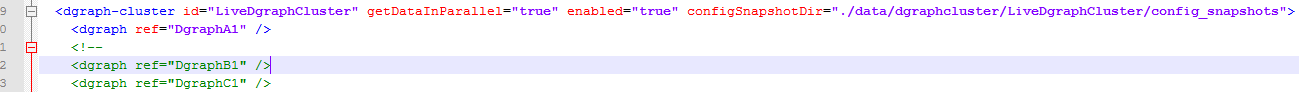
The GameStop app has been parameterized based on the environment variable set in the server settings. Moving the entire app to a new environment can be achieved by making a copy of the <client> app from the E:\Endeca\Apps\ directory and moving it to the new environment’s E:\Endeca\Apps\ directory. Once the app has been moved to the new environment the following steps are required to deploy the app.

1. Follow the steps within [**Move Experience Manager Configurations**](#_Move_Experience_Manager) section below.
2. On the Production Live Indexer uncomment and change “localhost” to the appropriate Dgraph host in the PRODLIVE\_environment.properties file.



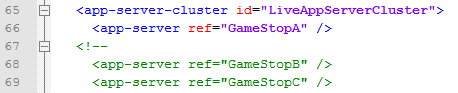
1. On the Production Live Indexer uncomment the corresponding Dgraphs in the PRODLIVE\_LiveDgraphCluster.xml

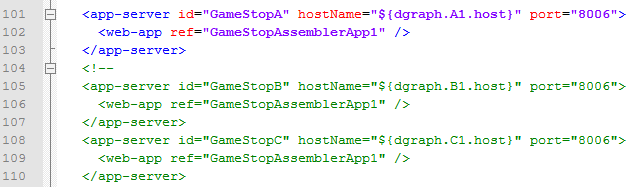






1. On the Production Live Indexer uncomment the corresponding PRODLIVE\_LiveAppServerCluster.xml



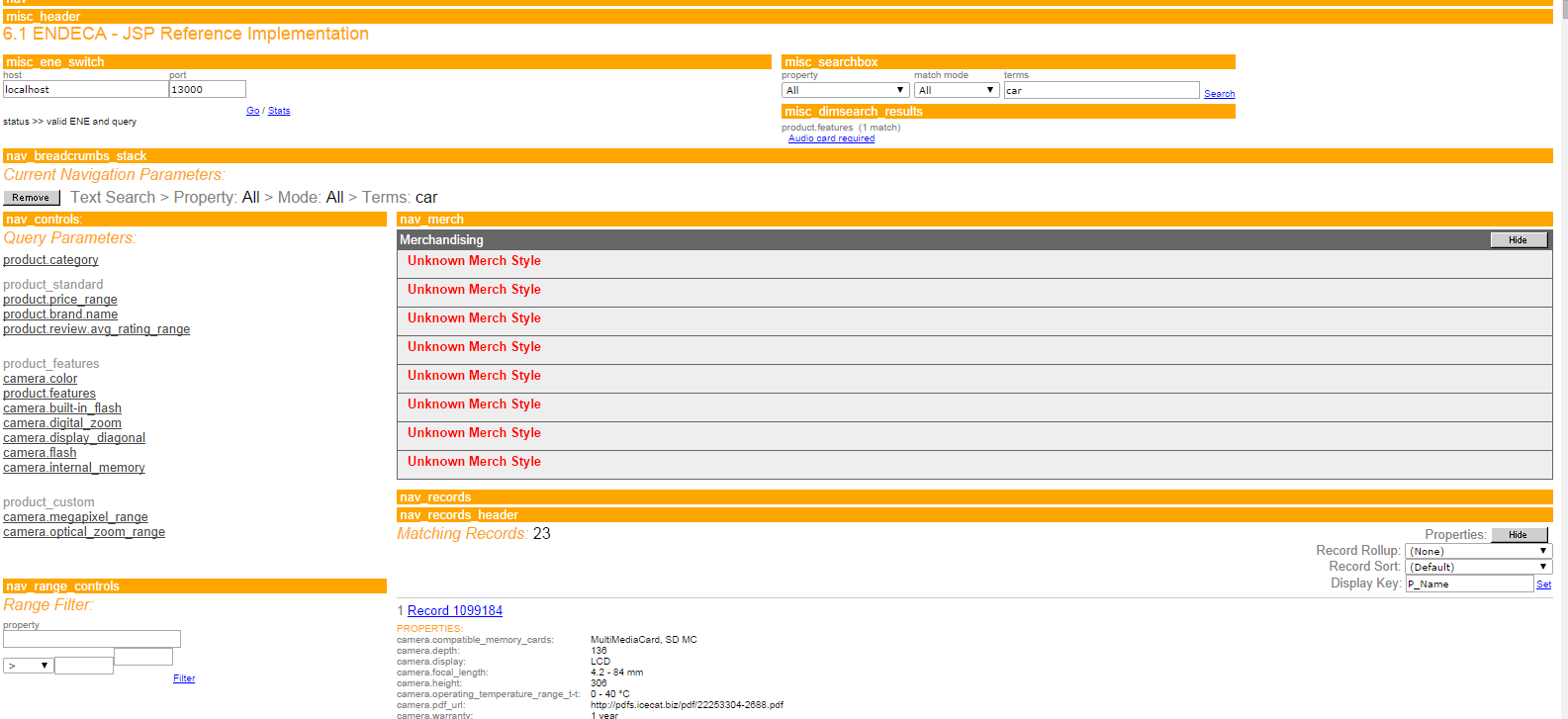


1. Confirm the snippet below is in the ..\control\initialize\_services.bat file. If not, follow the instructions in the [**Assembler API Configuration**](#_Assembler_API_Configuration) Section

powershell -Command "(gc ..\web\_apps\gamestop-service\WEB-INF\assembler.properties) -replace 'workbench.host=localhost', 'workbench.host=%ENDECA\_WORKBENCH\_HOST%' | Out-File  ..\web\_apps\gamestop-service\WEB-INF\assembler.properties -encoding "Default" "  
echo Finished updating assembler.properties file for gamestop-service application

1. Run ..\control\initialize\_services.bat
2. Run ..\control\run\_complete\_refresh.bat
3. Verify the reference apps are working ([http://localhost:8006/endeca\_jspref/controller.jsp?N=0&eneHost=<HOST>&enePort=<PORT](http://localhost:8006/endeca_jspref/controller.jsp?N=0&eneHost=%3cHOST%3e&enePort=%3cPORT)>)

Ex.



# Move Experience Manager Configurations

The Experience Manager configurations can be moved from one GameStop environment to another through a built in export script within the control directory.

**Run the export\_site.bat script**

When you’re moving to the new production environment the experience manager will have no configurations, so you will have to export from the working environment

1. Log into the QA server
2. Run E:\Endeca\Apps\GameStop\control\export\_site.bat <PATH>\<FILE>

(this will export the GameStop EM xml file to the desired directory with the specified file name) **Ex. export\_site.bat E:\Endeca\Apps\GameStop\GameStop.xml**

1. Copy the newly created **GameStop.xml** file to the Production Live Indexer’s control directory.
2. Log into the Production Live Indexer.
3. Run E:\Endeca\Apps\GameStop\control\import\_site.bat <PATH>\<FILE>

**Ex. import\_site.bat E:\Endeca\Apps\GameStop\GameStop.xml**

1. Run E:\Endeca\Apps\GameStop\control\promote\_content.bat from the production server to push the EM configurations to the live assembler service apps.
2. Verify the assembler service is returning a valid response (<http://localhost:8006/gamestop-service/json/pages/browse>

(NOTE: Check for “refinementCrumbs” in the response to verify the response is valid. Check for “error” in the response to recover any errors from the Experience Manager.)

Ex.



# Move the Assembler App

**Indexer to Indexer**

**Move the GameStop xml file from QA Live Indexer to Production Live Indexer**

1. Copy the gamestop-service-authoring.xml and gamestop-service.xml from the QA Live indexing server located:
   * E:\Endeca\ToolsAndFrameworks\11.0.0\server\workspace\conf\Standalone\

To the production staging indexer located

* + E:\Endeca\ToolsAndFrameworks\11.0.0\server\worksapce\conf\Standalone\

1. Restart Endeca Tools Service in Windows Services Control Panel on the production staging indexer
2. Verify the assembler service is returning a valid response (http://<HOST>:8006/gamestop-service/json/pages/browse on the production staging indexer.

**Live Indexer to Live Server**

**Prerequisite:**

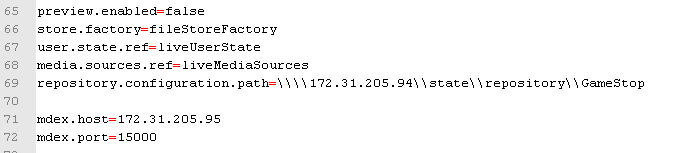
The Experience Manager repository on the indexing server will need to be made sharable for the live gamestop-service app to access on the Production Live Indexer.

1. Share the C:\Endeca\ToolsAndFrameworks\11.0.0\server\workspace\state directory.

**Migration:**

1. Copy the gamestop-service directory from QA Live Indexer (E:\Endeca\Apps\GameStop\webapps\gamestop-service to the Tomcat webapps directory on the live server (E:\ApacheTomcat\webapps\).
2. In the assembler.properties file located at: E:\ApacheTomcat\webapps\gamestop-service\WEB-INF\assembler.properties change the repository.configuration.path to reference the shared state directory on the indexer (prerequisite above).

**Ex. repository.configuration.path=\\\\172.31.205.94\\state\\repository\\GameStop**



1. Verify all the host and port references are correct.
2. Verify the assembler service is returning a valid response (http://<HOST>:80/gamestop-service/json/pages/browse

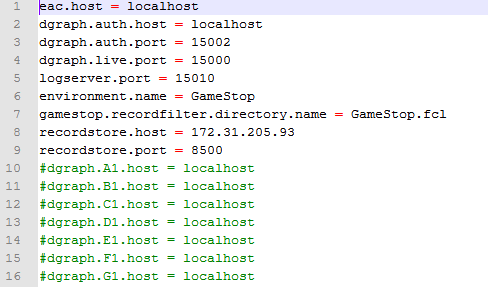
# Config Scripts

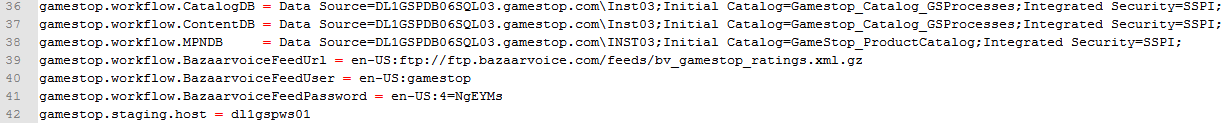
Within the ..\control\scripts\ directory each version of the assembler.properties and set\_environment.bat files will need to be configured for the correlating environment.

* **<ENDECA\_ENV>assembler.properties**
* **<ENDECA\_ENV>set\_environment.bat**

**PRODLIVE\_assembler.properties**

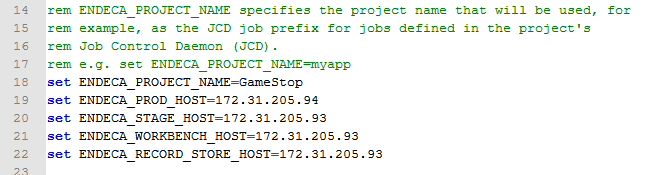
**Example:**

****

****

**PRODLIVE\_set\_environment.bat**

**Example:**

****

# Custom Control Scripts

## GameStop Run (Partial Run)

*Staging & Production Indexer*

The ..\control\**GameStopRun.py** script will be used to run a full baseline. The script will execute a series of functions to check the line count of the data, archive the data, sets the parameters for the pipeline files, execute the baseline update process, and send out an email notification if a failure in any of the processes occur.

## GameStop Mod

*Staging & Production Indexer*

The GameStop Mod script contains all the functions listed above.

## Promote Content

*Production Indexer*

Currently the promote content script is executed outside the GameStopRun.py script. The promote content script is only necessary on the Production Indexers. It is recommended that the GameStop team uses the ..\control\**promote\_content\_ifcrcheck.py** script to push experience manager changes to the live servers. There is logic in place to check the authoring assembler service is returning a valid response before pushing the configurations live. An email is sent out if the response it invalid.

## Export Site to Production

*Staging Indexer*

The ..\control\**exportSitetoProd.bat**  runs the export\_site.bat script which exports the authoring Experience Manager configuration file from staging to the GameStop control directory on the production indexer. This script is an entry in the Task Scheduler as ExportExperienceManagerConfigs. The task is triggered by the exportImportSite.bat script on the production indexer.

**The** **ExportExperienceManagerConfigs task will have to be added to all the staging indexers, use QA as a reference.**

## Export Import Site

*Production Indexer*

The ..\control\**exportImportSite.bat** script triggers the ExportExperienceManagerConfigs task on staging to export the staging EM configs to production. It then imports the file (GameStop.xml) into GameStop authoring and uses the promote\_content\_ifcrcheck.py script to push out to the live assembler apps.

## Record Store Check (PratialCheck)

*Staging & Production Indexer*

The ..\control\**record\_store\_check.bat** uses the CAS batch files to retrieve the number of records in the record store and writes it to a recordStore.size file in the log directory. The GameStopMod contains a function to compare the number of lines to a predetermined threshold, currently set to 75% of the original record stores.

## Record Store Archive Prep (PartialPrep)

*Staging & Production Indexer*

The ..\control\r**ecord\_store\_archive\_prep.bat** script uses the CAS batch files to write the record stores to a text file. The GameStopMod contains a function which compresses those files and moves them to the ..\archive directory.

# Baseline Update Process

## Baseline Update

*Staging & Production Indexer*

The baseline update process will be run through the GameStopRun.py script. A task should be created in the task scheduler to execute the GameStopRun.py.

## Promote Content

*Production Indexer*

The promote content process will only need to be run on production environment. A task can be created to execute the promote\_content\_ifcrcheck.py after the baseline update finishes or any time after.

## Partial Update

The partial update process will be run through the GameStopPartialRun.py script. A task should be created in the task scheduler to execute the GameStopPartialRun.py.

## Export Import Site

*Production Indexer*

A task to run the exportImportSite.bat can be created on the production indexer to automatically/manually push the staging Experience Manager configurations to the live production app.