Analysis Engine

Installation Guide

3.0.1-1

escenic

A CCI COMPANY

Table of Contents

1 Overview

1.1 Overview

Escenic Analysis Engine is a web statistics tool. This document describes how EAE works, how to install it on an application server, and how to administrate and configure it.

Escenic Analysis Engine is a module for logging and tracking statistical information based on the usage of Escenic web sites.

This document describes all aspects of Escenic Analysis Engine. Below is a summary of the content.

Introduction
: Explains what EAE is and how it works.

Installation
: Explains the installation procedure.

Getting started
: A tutorial on how to get started using this product, including topics like post-installation configuration options using the web administration interface, use of the query service and more.

Web administration
: A short introduction to the EAE web administration.

Installation tips
: Includes application and database server tips.

2 Introduction

2.1 What is Escenic Analysis Engine?

Escenic Analysis Engine (EAE) is a tool for logging and tracking statistical information based on visitors usage of Escenic web sites. EAE consists of a Logger module, for logging and storing data, and a Query Service module, an API framework for querying this data and generating real-time or historical reports.

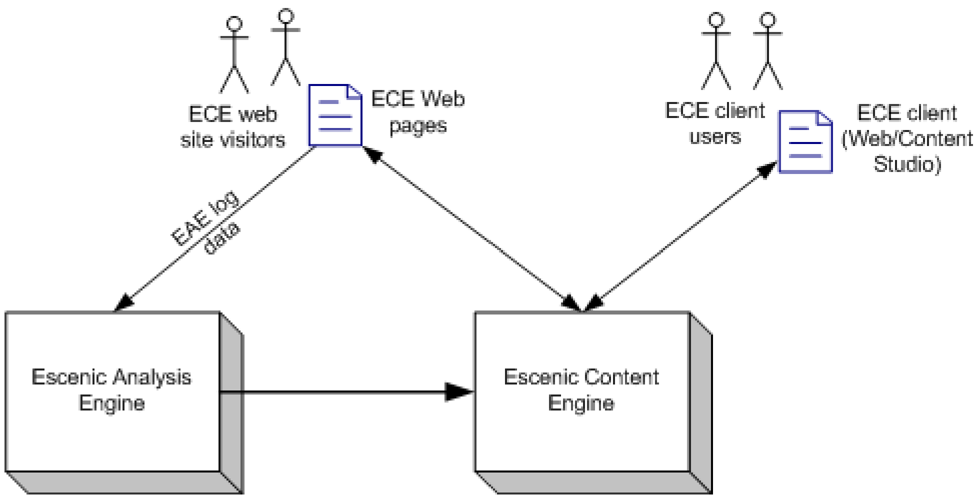
EAE currently implements the logging of page views, i.e., article and section frontpage views, and logging ad clicks and views. It has, however, a flexible architecture suitable for implementing logging and tracking of other types of statistical information, i.e., web site search statistics.

2.2 What purpose does Escenic Analysis Engine serve?

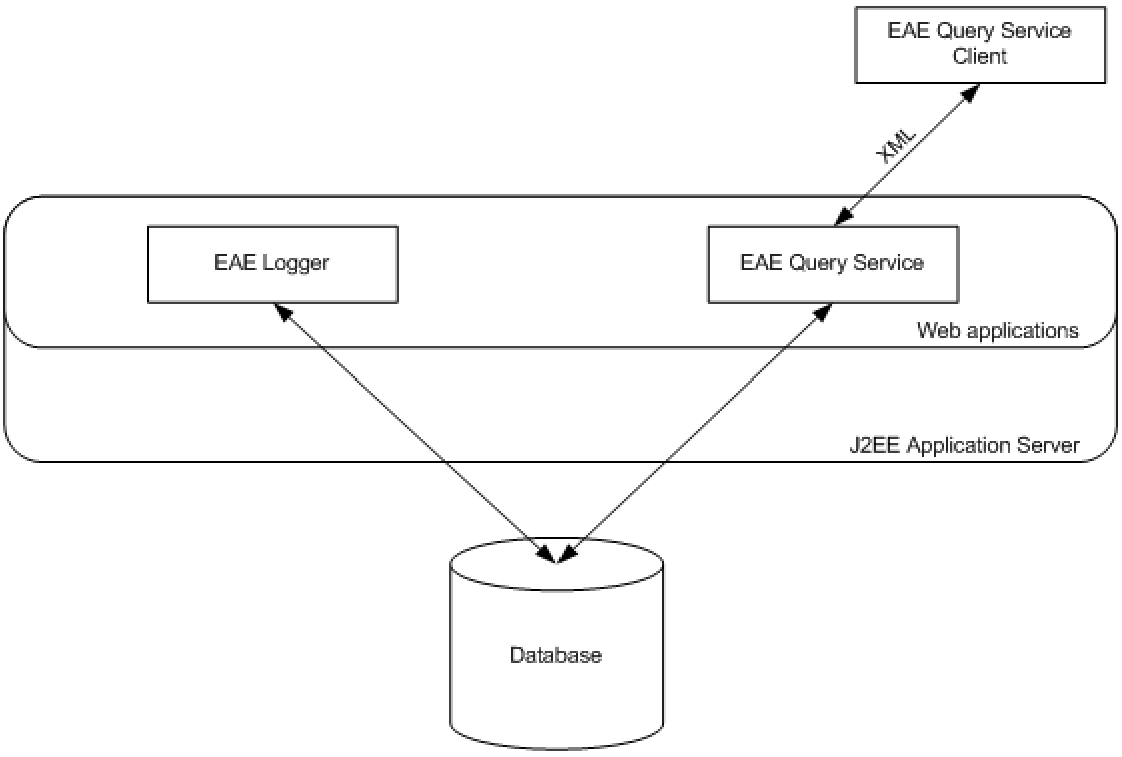
Escenic Analysis Engine has the purpose of providing real-time and short-term statistical information reflecting visitors use of Escenic web sites. This information can be analyzed by Escenic users (journalists, editors, etc.), i.e., to get an idea of what the current load on the site is and the popularity of the content they publish. Also, the visitors of the Escenic web site can be given some useful information, i.e., what the most popular articles are at the moment.

2.3 How does Escenic Analysis Engine work?

The figure below shows an architectural overview of Escenic Analysis Engine. The visitors of the Escenic web sites download pages from the ECE server. These pages contain some scripts (JavaScript) which will initiate a request to the EAE, i.e., uploading information about the page being viewed.



The Escenic Analysis Engine consists of three main components; EAE Logger, EAE Query Service, and EAE Query Service Client. The figure below provides a more detailed view of the Escenic Analysis Engine and illustrates the architectural composition of the main components.



Both Eae Logger and EAE Query Service are standalone J2EE web applications, and must be run on a J2EE application server. The Logger receives all the requests from the visitors of the web sites, handles the log information received with these requests and finally stores the handled log data in the database.

The Query Service is a Servlet used for retrieving reports from the EAE database, i.e., the most popular page views. Both queries and responses sent to and from the servlet are XML, and the protocol is defined by XML Schemas.

The EAE Query Service Client is a remote Java API for querying the EAE Query Service. It makes the communication with EAE Query Service transparent and hides all the XML processing logic. Hence, it simplifies the implementation of report modules.

2.4 What kind of statistical information does EAE support?

EAE Logger receives and stores page view information. The information stored with these page views includes object ID (i.e., article ID or section ID), object type (i.e., article or section), title, URL, context ID (i.e., section ID), time-slot (the time period in which the page view occurred), category (custom field), meta (custom field) and the number of page views that occurred during the time-slot.

The Query Service framework enables you to retrieve different reports based on the page view data stored. These are:

Total page views
: The total number of page views. Example: The total number of page views for the frontpage during last week.

Time distributed page views
: The number of page views distributed over time. Example: The total number of page views each hour yesterday.

Most popular page views
: The pages with the most page views. Example: The most popular articles in the sports section today.

For all the reports above, it's possible to filter the reports to specific parameters like object ID, types, etc.

By logging an unique session/user ID (i.e., session ID, user ID, MSISDN, etc.) to the meta field, the 'DistinctMetaCount' functionality provided by the Query Service can be used to keep track of unique number of sessions/users. These reports are similar as the page view reports listed above; Total DistinctMetaCount, Time distributed DistinctMetaCount and Most popular DistinctMetaCount. It's also possible to combine these reports so that they include both number of page views and 'unique sessions/users'.

3 Installation

This chapter contains general installation steps. For vendor specific details, refer to the appendix chapter 6, which provides tips and/or references to web sites containing more documentation/tips.

3.1 Pre-installation requirements

EAE release

You need an Escenic Analysis Engine release, **`analysis-engine.zip`**. If you do not have a release, please contact your Escenic representative.

Extract the release into a temporary folder. The folder should then contain **`lib/analysis-qs-client.jar`**, **`misc`** (folder), **`documentation`** (folder), **`wars/analysis-qs.war`**, and **`wars/analysis-logger.war`**.

Application server

An application server must be set up and running. The application server must be J2EE 1.5++ certified, running on JDK 1.5++

Database

You need a database server up and running. Supported databases are MySql (5.x), Oracle, MS Sql Server 2000 and Sybase.

Create a database instance for the EAE application and a user which has sufficent access to this database instance; http connect, select, insert, update, create tables, etc.

Due to security and performance issues, it's recommended that EAEs application and database servers are installed on a separate physical server, a part from other mission critical applications. In order to achieve even more performance, run the EAE database server on its own dedicated physical server, and distribute the Logger web application on as many physical servers necessary to handle the amount of concurrent requests during peak time. Refer to the section **Multiple Logger web applications** in the next chapter for more details.

The Query Service module utilizes JAXP 1.2 XML Schema validation. It's recommended that a XML library supporting JAXP 1.2 is installed on the application server. However, the application will work with older XML libraries, but in a non-validating mode.

3.2 Installing EAE Logger

3.2.1 Create database tables

This web application requires some database tables to be installed, if this hasn't been done already. The script for installing these tables is found in **`misc/database/`**. Use the script which filename reflects the database vendor you are using, i.e., for MySql databases, use the eae-mysql.sql script.

3.2.2 JDBC driver

Put the JDBC driver you need for accessing your database server into the application servers common classpath.

3.2.3 Create database connection pool for the Logger

EAE Logger requires a database connection pool. Please refer to your application server documentation on how to set up database connection pools. The database connection pool should allow approximately 4 active connections, 2 idle connections and the JNDI name could be something like `jdbc/eae-logger/logger`.

> Restart application server after creating the database connection pool.

3.2.4 Install Web Application Archive analysis-logger.war

The EAE Logger is a J2EE web application and is packaged as a WAR (web application archive) file. Install `analysis-logger.war` on your application server.

3.2.5 Basic configuration

Log into the EAE Logger admin by using an Internet browser and typing in the URL `http://server:port/analysis-logger/admin`. Using the menu, choose Options->General. Set the correct database type and data source name for the connection pool you just created. If your JNDI name for the connection pool was `jdbc/eae-logger/logger`, then the data source name will be `java:comp/env/jdbc/eae-logger/logger` (Tomcat) or `jdbc/eae-logger/logger` (Oracle 10g AS). Then apply these changes.

> See chapter 5 for short introduction to the Logger web administration.

> The username (default: admin) and password (default: secret) of the admin user is found in the EAE Logger web app `web.xml` file.

> Restart application server if changes were done to the database properties.

> Ensure UTF-8 support in application server. A filter is available in EAE Logger webapp which is disabled by default. Uncomment that portion in `web.xml` to enable the filter if you need to use `ServletRequest.setCharacterEncoding("UTF-8")`.

3.2.6 Verify installation

Log into the web administration for Logger and browse to Status->Database. Verify that the database status is OK.

3.3 Installing EAE Query Service

3.3.1 Create database connection pool for Query Service

EAE Query Service requires a database connection pool. Please refer to your application server documentation on how to set up database connection pools. The database connection pool should allow approximately 6 active connections, 2 idle connections and the JNDI name could be something like **`jdbc/eae-qs/qs`**.

> If this web application is running on a separate application server, make sure the JDBC driver is added to this application servers common classpath as well.

> Restart application server after creating this database connection pool.

3.3.2 Install Web Application Archive analysis-qs.war

The EAE Query Service is a J2EE web application and is packed as a WAR (web application archive) file. Install **`analysis-qs.war`** on your application server.

> This web application depends on the database tables created during EAE Logger installation.

3.3.3 Basic configuration

Log into the EAE Query Service admin by using an Internet browser and typing in the URL **`http://server:port/analysis-qs/admin`**. Using the menu, choose Options->General. Set the correct database type and data source name for the connection pool you just created. If your JNDI name for the connection pool was **`jdbc/eae-qs/qs`**, then the data source name will be **`java:comp/env/jdbc/eae-qs/qs`** (Tomcat) or **`jdbc/eae-qs/qs`** (Oracle 10g AS). Then apply these changes.

> See chapter 5 for short introduction to the Query Service web administration.

> The username (default: admin) and password (default: secret) of the admin user is found in the EAE Query Service web app **`web.xml`** file.

> Restart application server if any changes were made to the database properties.

3.3.4 Verify installation

Log into the Query Service web administration and browse to Status->Database. Verify that the database status is OK.

3.4 Installing EAE Query Service Client

Simply add the **`analysis-qs-client.jar`** to the Java classpath for the application using the EAE QS Client.

In order to make the EAE QS Client available for i.e. ECE template developers, add the jar file to the classpath for the ECE installation. However, since this jar file is part of the EAE Plugin, it will already be installed if the EAE Plugin has been installed on your system.

JavaDoc is found in **`documentation/javadoc/qs-client/`** .

4 Getting Started

This chapter will help you get started using Escenic Analysis Engine after a successful installation of its web application modules. If you're not familiar with the Logger and Query Service web administration tools, please read chapter 5 before continuing.

4.1 EAE Logger

This section starts off with a description on how to deploy client scripts required to commence the logging of page views. Then a guide to the available configuration options and services will be given, followed by a section on how to deploy multiple Logger web applications.

4.1.1 Enabling logging of page views

Logging of page views is done by placing a piece of JavaScript on each ECE page (article/section frontpage) downloaded by the visitors browsers. A common JSP template for generating the JavaScript for both article pages and section frontpages is found in the EAE distribution, **`misc/client-scripts/jsp/eaePageviewLoggerClient.jsp`**. Copy this client script to your publication(s). View the example below for details. As the Escenic templates may vary from site to site, it's not always evident where to best include this JSP. However, make sure the script appears on the bottom of the page, included in a common JSP file called from the article templates (art_*.jsp) and section templates (sec_*.jsp).

Including the EAE client JavaScript

Let's say that a common JSP template is found in **`{PUB_ROOT}/template/ver1-0/wireFrame/default.jsp`**. Then copy **`eaePageviewLoggerClient.jsp`** to the folder **`{PUB_ROOT}/template/ver1-0/eae/`**. Edit **`default.jsp`** and add the following at the bottom of this file.

```
<%-- Start: Including EAE client script --%>
<TEMPLATE:call file="../eae/eaePageviewLoggerClient.jsp" />
<%-- End: Including EAE client script --%>
```

When the script is installed, edit the root section parameters. In Web Studio section administration, add the following property telling the Logger script what the URL to the EAE Logger servlet is:

```
eae.logger.url=http://server:port/analysis-logger/Logger
```

4.1.2 Configuration options and services

There are several configuration options and services available for the EAE Logger module. These are managed through this modules web administration. After logging in to the web administration you will be able to see an **`Options`** menu item. Clicking on this will extend this item and show the **`General`** and the **`Page view`** options sub menu items.

General

The General options only includes some database options which were introduced during the installation of this module. The database options enables you to choose database type and the database source name, which is the JNDI path to the database connection pool that will be used by the Logger.

Page view

The page view Logger has some special services that can be managed from the Page view options page. These services are aggregation services and a maintenance service. The aggregation services will aggregate the page view tables data; from the original minute time-slots resolution to hour slots, or further, from hour slots resolution to day slots. The maintenance service will delete old records and clean up redundant records in the page view meta table.

The services are run periodically, defined by special cron expressions which you define. The cron expressions will say when and how often the services will be triggered. More on how these cron expressions are created is found on the Page view options page.

Clicking on the "start service" buttons along with correct cron expressions filled in the text fields right to these buttons, will start the aggregation services and the maintenance service. The aggregation services can be configured from **`logger.cfg`** placed in **`analysis-logger.war`**. After the war file is exploded in appserver, it will reside in **`analysis-logger/WEB-INF/config/logger.cfg`**. Please set the **`*.cron.expr`** appropriately so that it starts automatically when the webapp is loaded. If any expression is missing then that service will not start.

In order to see valid expressions please refer to the following URL. **`http://quartz.sourceforge.net/javadoc/org/quartz/CronTrigger.html`**

Please note that if the WAR/EAR file is redeployed, your configuration will be lost. To avoid this you should keep the configuration file **`logger.cfg`** in some other location, defined in a system property called **`com.escenic.eae.config`**. This location may either be in the file system or on the classpath. To keep **`logger.cfg`** somewhere in the file system, for example, you could specify **`-Dcom.escenic.eae.config=/some/location`** in your application server's start-up script.

When the EAE-Logger application starts, it will look for the configuration file in the location specified with **`com.escenic.eae.config`**, and if that fails it will default to **`analysis-logger/WEB-INF/config/logger.cfg`**.

4.1.3 Multiple Logger web applications

For Escenic sites with high traffic peaks, the number of parallel logging requests to the EAE Logger can become too many and performance issues will arise. In order to cope with these performance issues, EAE Logger can be deployed on multiple physical application servers.

When deploying multiple instances of EAE Logger, only one these can have the Page view services running; choose one as master and let the rest be slaves. This is important, avoiding potential interference problems if two or more Logger instances would start aggregation and maintenance services concurrently.

If your web infrastructure has a physical load balancer in front, this can be used to distribute the load over the available Logger servers. Otherwise, the application server might provide some means for load balancing. Another possibility is to implement a simple 'round robin' algorithm for the template script when it writes the URL to the Logger servlet in the JavaScript sent to the visitors browser.

4.2 EAE Query Service

This section includes a tutorial on how to use the Query Service. However, if you are going to build a Java module that will execute queries against the Query Service, it's highly recommended that the Query Service Client is used rather than coding logic for connecting to Query Service and handling XML. The Query Service Client is described in the next section.

4.2.1 Using the Query Service

The use of the Query Service involves the following steps:

1. **Connect to Query Service servlet**

   Connect to the servlet and open a stream for sending the XML query through.

2. **Send XML query**

   The XML should be sent using HTTP Post method and the parameter containing the XML must be named **`query`**.

   Please refer to XML Schema defining the query XML protocol, **`{EAE QS web app}/WEB-INF/eae-qs-query.xsd`**, when creating queries. Some examples are listed below.

   Time distributed page views query during Aug 22nd with interval set to 60 minutes

```
<eae-qs>
  <query>
    <pageview>
      <time-distributed>
        <timeframe>
          <period from="2004-08-22 00:00" to="2004-08-23 00:00" />
        </timeframe>
        <interval value="60" />
      </time-distributed>
    </pageview>
  </query>
</eae-qs>
```

3. **Receive XML response**

   Receive the XML response stream from the servlet, and redirect it to a XML parser.

4. **Parse XML response**

   Parse the XML response. When using Java it is recommended that a SAX parser is used. Please refer to XML Schema defining the response XML protocol, **`{EAE QS web app}/WEB-INF/eae-qs-query.xsd`**, when creating a parser. Some examples of responses are listed below.

   Time distributed page views response from Aug 22nd and to 29th Aug with interval set to 1440 minutes

```
<eae-qs>
  <response>
    <pageview>
      <time-distributed>
        <meta>
          <response-time value="215" />
        </meta>
        <timeslots>
          <timeslot from="2004-08-22 00:00" to="2004-08-23 00:00"
 pageviews="75467" />
```

```
          <timeslot from="2004-08-23 00:00" to="2004-08-24 00:00"
pageviews="71546" />
          <timeslot from="2004-08-24 00:00" to="2004-08-25 00:00"
pageviews="89063" />
          <timeslot from="2004-08-25 00:00" to="2004-08-26 00:00"
pageviews="67323" />
          <timeslot from="2004-08-26 00:00" to="2004-08-27 00:00"
pageviews="93443" />
          <timeslot from="2004-08-27 00:00" to="2004-08-28 00:00"
pageviews="56432" />
          <timeslot from="2004-08-28 00:00" to="2004-08-29 00:00"
pageviews="45663" />
        </timeslots>
      </time-distributed>
    </pageview>
  </response>
</eae-qs>
```

4.2.2 Choosing a Query Strategy

The Analysis Engine's "most popular" queries can sometimes take a long time to execute in a MySQL database. Whether or not this problem arises depends on the data set being queried, the parameters of the individual query, and on the variant of MySQL that is in use. To overcome this problem you can choose an optimization strategy for use with MySQL databases.

Three strategies are available:

`AvoidJoin`
: This is the default strategy. Two separate queries are run, and then merged in Java. This strategy should run with the same speed regardless of which variant of MySQL is being used, but it may in some cases be faster to use a database join.

`NormalJoin`
: A normal join operation is run and then the X most popular content items are selected from the result set. This method can incur a performance penalty from joining a large result set from which only a few results are selected. If your database is MySQL or Percona, this strategy is probably the best alternative to **`AvoidJoin`**.

`JoinInSubQuery`
: The first part of the query is executed in a sub-query. The results from this subquery are then limited before executing the join. This improves performance on some MySQL variants, but can dramatically worsen the performance on others. If your database is MariaDB, this strategy is probably the best alternative to **`AvoidJoin`**.

If the client requests more than 50 results to be returned (**`max > 50`**), the default **`AvoidJoin`** strategy will revert to **`NormalJoin`** in order to avoid a potentially large join operation that would consume the Java heap memory. In general, requesting more than 50 results is unnecessary and should be avoided.

If the default **`AvoidJoin`** strategy does not perform well for you, try switching to one of the other options by setting the **`mysqlJoinStrategy`** option in the query service's configuration file, **`analysis-qs/WEB-INF/config/qs.cfg`**:

```
# mysqlJoinStrategy can be set to one of AvoidJoin, JoinInSubQuery or NormalJoin
mysqlJoinStrategy=AvoidJoin
```

> Please note that if the WAR/EAR file is redeployed, your configuration will be lost. To avoid this you should keep the configuration file **`qs.cfg`** in some other location, defined in a system property called **`com.escenic.eae.config`**. This location may either be in the file system or on the classpath. To keep **`qs.cfg`** somewhere in the file system, for example, you could specify **`-Dcom.escenic.eae.config=/some/location`** in your application server's start-up script.
>
> When the Query Service starts, it will look for the configuration file in the location specified with **`com.escenic.eae.config`**, and if that fails it will default to **`analysis-qs/WEB-INF/config/qs.cfg`**.

4.3 EAE Query Service Client

The purpose of the Query Service Client is to make it simpler to create modules querying the Query Service, hiding TCP/IP communication and XML logic. In summary, the way this remote client works, is that you first create an instance of a query manager, i.e., for page view queries you create an instance of **`PageviewQueryManager`**, and tell this manager the URL to the Query Service servlet. Then you can call all the available query methods. These methods return special objects/Beans, wrapping the response from the Query Service.

Examples and documentation on this remote client API is available in the Query Service Client JavaDoc, found in **`documentation/javadoc/qs-client/`**.

Please note that for sites with high traffic peaks where the application servers receive too many requests, the Query Service may become too busy, and, it would take too long to respond to the Query Service Client. The client keeps waiting until it receives response from the Query Service which may cause performance issues to the sites using this service. To cope with such situations, one can define two system properties **`sun.net.client.defaultConnectTimeout`** (default -1) and **`sun.net.client.defaultReadTimeout`** (default -1) in the appserver start up script as a workaround. These properties specify the default connect and read timeout in milliseconds, respectively. If the timeout values expire before connection can be established to the Query Service or data is available to the Query Service Client, having these properties defined will cause a **`java.net.SocketTimeoutException`** to be raised.

> For more information about these system properties, please refer to http://java.sun.com/j2se/1.5.0/docs/guide/net/properties.html.

4.4 EAE Query Service Taglib

The EAE Query Service also provides a Taglib library which runs on top of the EAE Query Service Client API. Today, there is only one tag supplied with the EAE Taglib, called the 'eae:most-popular' tag. This tag enables you to easily create 'most popular' lists.

Using the Taglib requires that analysis-qs-client.jar is installed in the global classpath. If EAE Plugin is installed on the server(s), this jar is already in installed. Also, the analysis-taglib.jar must be placed in your publications **`WEB-INF/lib`** directory. Then, you are ready to start making the JSP's.

Every JSP using the taglib needs to declare it with a JSP directive:

```
<%@ taglib uri="http://www.escenic.com/taglib/escenic-eae" prefix="eae" %>
```

An example JSP is found in the EAE distribution JAR, under the misc/taglib-example.

4.4.1 eae:most-popular

Syntax

```
<eae:most-popular
   category="..."?
   collectionId="..."?
   from="..."?
   groupByPublication="..."?
   id="..."
   includeContextPubId="..."?
   includeRest="..."?
   includeSubSections="..."?
   max="..."?
   meta="..."?
   pubId="..."?
   sectionId="..."?
   sinceHoursAgo="..."?
   sinceMinutesAgo="..."?
   to="..."?
   type="..."?
   url="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
Used to identify 'com.escenic.analysis.qs.client.pageview.MostPopular'.

`collectionId`, no runtime expressions
Used to identify a java.util.Collection containing 'com.escenic.analysis.qs.client.pageview.MostPopularElement'.

`url`, mandatory
Type: String

This is the URL to the EAE query service, i.e. http://server:port/eae-qs/QueryService.

`from`
Type: java.util.Date

Defining the start of the time period.

Note: Cannot be used with sinceMinutesAgo/sinceHoursAgo attributes.

`to`
Type: java.util.Date

Defining the end of the time period.

Note: Cannot be used with sinceMinutesAgo/sinceHoursAgo attributes.

`sinceMinutesAgo`
Type: String

Defining how many minutes ago the time period starts (appending to sinceHoursAgo).

Note: Cannot be used with to/from attributes.

`sinceHoursAgo`
Type: String

Defining how many hours ago the time period starts (appending to sinceMinutesAgo).

Note: Cannot be used with to/from attributes.

`max`

Type: String

Default: 10

Maximum number of elements in the list.

`type`

Type: String (comma separated values) or String[], Default: (empty/null)

Limit the list to contain only the specified types, i.e. 'article' or 'section'.

`sectionId`

Type: Integer or String (comma separated values) or String[], Default: (empty/null)

Limit the list to contain elements only from the specified sections.

`includeSubSections`

By default, the list will be limited to contain only elements from the specified sections. However, setting this value to **`true`** will also include items from the subsections of the specified sections.

`pubId`

Type: Integer or String (comma separated values) or String[], Default: (empty/null)

Limit the list to contain elements only from the specified publications. Note the 'includeContextPubId' attribute. If that attribute is set to true, pubIds added with the pubId attribute will be appended to the context publication ID.

`includeContextPubId`

By default, if neo.xredsys.api.Publication is found in the **`request scope`**, the list will be limited to contain only elements from this Publication. However, setting this value to **`false`** will ignore the publication context.

`meta`

Type: String (comma separated values) or String[], Default: (empty/null)

Limit the list to contain only the specified metas.It may be empty but must be a **`String(comma separated)`** or a **`String[]`**. Absence of this attribute will include all metass.

`category`

Type: String (comma separated values) or String[], Default: (empty/null)

An array of category values which this query will be limited to. The attribute **`category`** may be empty but must be a **`String(comma separated)`** or a **`String[]`**. Absence of this attribute will include all categories .

`includeRest`

This attribute determines, Wether or not to include the remaining objects as one element. If set to true, this will add an aggregated element to the list (max + 1) representing all the remaining objects.By default the value of this attribute is **`false`**.

`groupByPublication`

This parameter is used to configure whether article will be grouped by publication or not when generating most popular reports.It defaults to **`true`**. If set to **`false`**, **`aricle`** and / or **`section`** will not be grouped across publications.Please note that when this parameter set **`false`**, accessing pubId property in **`'com.escenic.analysis.qs.client.pageview.MostPopularElement'`** will return 0 as article are not goruped considering publications in such case.

Scripting variable (id)

A scripting variable will be defined using the value of the `id` attribute as its name. The variable is of type `com.escenic.analysis.qs.client.pageview.MostPopular`.

Scripting variable (collectionId)

A scripting variable will be defined using the value of the `collectionId` attribute as its name. The variable is of type `java.util.Collection`.

5 Web administration

In this appendix, a short introduction to the EAE Logger and Query Service web administrations are given.

5.1 What are the EAE Logger and Query Service web administrations?

They are administration tools which enables you to manage the functions, options and services available, and in addition, view status information.

5.2 Where can I find the these web administrations?

Start an Internet browser. For entering the EAE Logger web administration, navigate to **`http://server:port/analysis-logger/admin`**, for entering EAE Query Service web administration, navigate to **`http://server:port/analysis-qs/admin`**.

Username and password for logging in to these web administrations, are found the their respective web applications web.xml file (in the root of the WEB-INF folder, default username/password is admin/secret).

> **Security note**
>
> The username and/or password should be altered from its default values defined in the **`web.xml`** file.

5.3 What kind of information and functions are available on the administration pages?

5.3.1 Logger admin

Status->Logger
The Logger status page gives information about the load on the system and the Logger database queue.

Status->Database
The Database status page gives information about the database server connection and the database tables used by this application.

Options->General
The General options page provides configuration options for the Logger application.

'Logger Database Connection' option enables admin to configure database type and data source name.

Options->Page view
The Page view options page is for configuring Pageview specific options.

'Hour Aggregation Service', 'Day Aggregation Service' and 'Maintenance Service' options are managed here; stopping and starting the services with a given cron expression defining how often these services will be triggered.

'Table names' option is used for telling the Logger what the Page view tables are called.

5.3.2 Query Service admin

Status->Query Service

The Query Service status page gives information about query cache size and the number of pending queries.

Status->Database

The Database status page gives information about the database server connection and the database tables used by this application.

Options->General

The General options page provides configuration options for the Query Service application.

'Query Service Database Connection' options enables admin to configure database type and data source name.

Options->Page view

The Page view options page is for configuring Pageview specific options.

'Table names' option is used for telling the Query Service what the Page view tables are called.

6 Installation tips

This appendix gives vendor specific tips on how to manage and/or find more information about the application and database servers supported.

6.1 Application server tips

6.1.1 Tomcat

Homepage

http://jakarata.apache.org/tomcat

Where to put the JDBC driver

`{TOMCAT_HOME}/common/lib/`

Creating database connection pool

There are two ways to do this, either by editing the server.xml configuration file found in `{TOMCAT_HOME}/conf/` directory, or by logging into tomcat web admin and adding the connection pool there, either as a global data source or as a local web app data source. The latter method is the simplest.

Before logging into the Tomcat web admin, you must configure and add a user with admin privileges in the file `{TOMCAT_HOME}/conf/tomcat-users.xml`. Below is an example.

```
<user username="user" password="password" roles="admin,manager" />
```

Start a web browser and enter `http://tomcat_server_host:port/admin` and then log in. When logged into the admin, navigate to Resources -> Data Sources using the menu on the left. The choose 'Create New Data Source' from the drop down menu on the right. Fill in everything but the 'Validation Query' field. See database vendor tips for details on the JDBC url and driver format.

How to install a Web Application Archive

Copy the WAR file to `{TOMCAT_HOME}/webapps/` and restart the application server.

6.1.2 Oracle AS 10g

Homepage

http://www.oracle.com

Where to put the JDBC driver

`{ORACLE.HOME}/j2ee/{OC4J_INSTANCE}/applib/`

> If you are using an Oracle database server, the JDBC driver is probably already available. However, there might be a mismatch between the Oracle JDBC driver that comes with the 9iAS and the Oracle database server. The correct JDBC driver is probably submitted with the database server, or it can be downloaded from Oracles website.

Creating database connection pool

There are two ways to do this, either by editing the `data-sources.xml` configuration file found in `{ORACLE.HOME}/j2ee/{OC4J_INSTANCE}/config/` directory, or by logging into the Oracle Enterprise Manager (default port 1810), and then add the connection pool there. The latter method is the simplest.

Start a web browser and enter URL to the Oracle Enterprise Manager, i.e. **`http://oracle.server:1810`** and then log in. When logged into the admin, navigate to the OC4J instance that will run the EAE web applications. Then choose 'Administration' followed by 'Data Sources'. Press the 'Create' button. Fill in the following.

Name
The name of this connection pool, i.e., EaeLogger or EaeQs.

Data Source Class
Use 'com.evermind.sql.DriverManagerDataSource' with non-Oracle databases, or a pure Oracle data source class if you are using Oracle database.

JDBC URL
The JDBC URL, i.e., Oracle JDBC URL 'jdbc:oracle:thin:@[host]:[port1521]:[SID]'.

JDBC Driver
The JDBC driver, i.e., Oracle JDBC driver 'oracle.jdbc.driver.OracleDriver'.

Username
Database username.

Use cleartext password (check this)
Database password.

Location, Transaction(XA) Location and EJB Location
Location attributes, i.e. 'jdbc/eae-logger/logger', 'jdbc/eae-logger/loggerXa' and 'jdbc/eae-logger/loggerEjb'.

Connection Retry Interval (seconds)
Set this i.e., to '1'.

Cached Connection Inactivity Timeout (seconds)
Set this to i.e. '30'.

Click the 'Apply' button and restart application server if necessary.

See database vendor tips for details on the JDBC url and driver format.

How to install a Web Application Archive
Start a web browser and enter URL to the Oracle Enterprise Manager, i.e. **`http://oracle.server:1810`** and then log in. When logged into the admin, navigate to the OC4J instance that will run the EAE web applications. Then choose 'Applications'. From this page you can deploy WAR files, just click the 'Deploy WAR file' button.

Telling the application server to allow context lookup from user threads
By default, Oracle 10g application server doesn't allow context lookup support from user-created threads. The EAE Logger application requires that this option is enabled. In order to enable this option, log into 10g iAS Enterprise Manager, choose the OC4J instance running the Logger application, choose Administration, then Server Properties. Add **`-Doc4j.userThreads=true`** to the Java Options.

Tuning the the Oracle (Apache) HTTP server connection handling
The client connection handling should be modified, closing each connection after one request. This is because a log request is not followed by multiple requests (css, images, etc) from the same client, which is the case for web page requests.

Log into the Enterprise Manager, choose HTTP_Server, then Administration and finally Server Properties. Under Client Connection Handling, uncheck the Allow Multiple Requests per Connection option, choose Apply and restart the HTTP server.

Telling the application server to use Xerces XML parser

Oracle 10g application server defaults to its own SAX parser implementation. EAE uses Xerces, and the Oracle application server instance must be configured to default to Xerces for EAE to work properly.

The first step is to copy a Xerces library jar from the EAE web application library folder to `{OC4J_Instance}/applib`.

The second and final step is to add the Java options – `Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl` and `-Dorg.xml.sax.driver=org.apache.xerces.parsers.SAXParser`. This is done using the Enterprise Manager, navigate to the OC4J instance, then choose 'Administration' followed by 'Server properties'. Add the above option to 'Java option'.

FYI, if you like to force the application server to use Xalan, add the Java option – `Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFacto` as well.

6.1.3 WebLogic 8.1

Please note that WebLogic requires repacking of the war files. This is described in the the section **How to install a Web Application Archive**.

Homepage

http://www.bea.com

Where to put the JDBC driver

`{WEBLOGIC.HOME}/weblogic81/lib/`

> WebLogic comes with a bunch of pre-installed JDBC drivers. A good rule here is to use the same JDBC driver as Escenic Content Engine if the same database server type is used. If not, it's recommended that you download the JDBC driver database server vendor homepage, ref database section below. Remember to remove any existing JDBC drivers that may conflict with the driver you add to the library repository.

Creating database connection pool

Log into the web logic administration console, and using the menu navigation tree, go to 'mydomain -> Services -> JDBC -> Connection Pools'. Then choose 'Configure a new JDBC Connection Pool...'.

Choose correct 'Database Type' and 'Database Driver'. You will probably use the 'Other' option. Choose 'Continue'.

Fill in the following:

Name

The name of this connection pool, i.e., 'EAE Logger Connection Pool' or 'EAE QS Connection Pool'.

Database Name

The name of the database to connect to, i.e., 'eae'.

Host Name

Host name for the database server, i.e., 'localhost'

Port
The port on the database server used to connect to the database, i.e. 1521.

Database User Name
The database account user name used in the physical database connection..

Password/Confirm password
The database account password used in the physical database connection

When the connection pool was successfully created, navigate to 'mydomain --> Services -> JDBC -> Data Sources' using the menu navigation tree. Then choose 'Configure a new JDBC Data Source'.

Choose correct 'Database Type' and 'Database Driver'. You will probably use the 'Other' option. Choose 'Continue'.

Fill in the following:

Name
The name of this data source, i.e., "EAE Logger Data Source" or "EAE QS Data Source".

JNDI Name
The JNDI path for the data source, i.e **`jdbc/eae-logger/logger`** or **`jdbc/eae-qs/qs`**.

Honor Global Transactions
Keep this checked.

Emulate Two-Phase Commit for non-XA Driver
Keep this unchecked.

Connection pool
Select the connection pool created for this data source.

When the data source is created, restart the application server.

See database vendor tips for details on the JDBC url and driver format.

How to install a Web Application Archive

Deploying the EAE WAR files on WebLogic, must follow the steps below. These steps involves unpacking the WAR files into web application directories, which then are deployed using the WebLogic console.

analysis-logger.war
Unpack the WAR file (**`jar xvf ../anlaysis-logger.war`**) to a folder, **`{ANALYSIS_LOGGER}`**, typically 'analysis-logger'. Move **`{ANALYSIS_LOGGER}`** to the WebLogic applications directory and then deploy this web application using the console.

analysis-qs.war
Unpack the WAR file (**`jar xvf ../analysis-qs.war`**) to a folder, **`{ANALYSIS_QS}`**, typically 'analysis-qs'. Move **`{ANALYSIS_QS}`** to the WebLogic applications directory and then deploy this web application using the console.

6.2 Database server tips

6.2.1 MySql

Homepage

http://www.mysql.com

Creating database and user

After installing MySQL, you must create an EAE database and an user with sufficient access rights to the database. Information on how to do this, is found at http://dev.mysql.com/doc/mysql/en/index.html.

JDBC driver

MySQL Connector/J found at http://www.mysql.com/products/connector/j/.

URL format is `jdbc:mysql://host:3306/database?autoReconnect=true`

Driver is `com.mysql.jdbc.Driver`.

6.2.2 Oracle

Homepage

http://www.oracle.com

Creating database and user

After installing Oracle, you must create an EAE database/SID and an user with sufficient access rights to the database.

JDBC driver

Oracle JDBC driver is typically distributed with the Oracle database server, or it can be downloaded from http://www.oracle.com. Make sure the JDBC driver corresponds to the database server version.

URL format is `jdbc:oracle:thin:@[host]:[port1521]:[SID]`

Driver is `oracle.jdbc.driver.OracleDriver`.

6.2.3 Sybase

Homepage

http://www.sybase.com

Creating database and user

After installing Sybase, you must create an EAE database, and an user with sufficient access rights to the database.

JDBC driver

Sybase JDBC driver can be downloaded from http://www.sybase.com. Make sure the JDBC driver corresponds to the database server version. An alternative driver is found at http://jtds.sf.net.

URL format for jTDS is `jdbc:jtds:sybase://[host]:[port4100];DatabaseName=[database name]`.

Driver for jTDS is `net.sourceforge.jtds.jdbc.Driver`.