An Oracle White Paper
June 2011

# MySQL- Oracle Enterprise Gateway Integration Guide

## Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

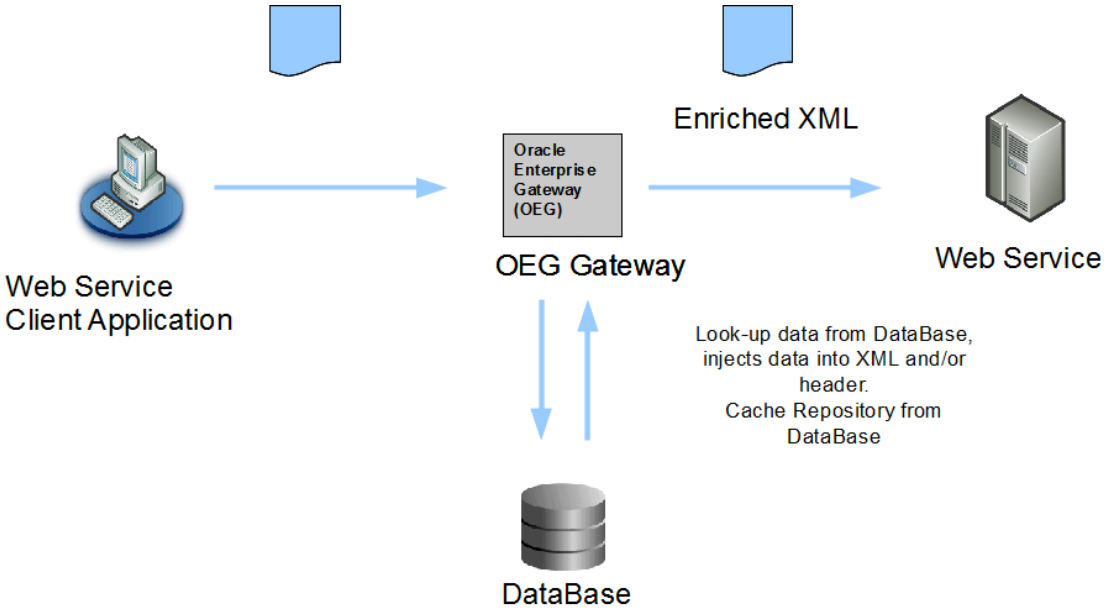
# 1 Introduction

## 1.1. Purpose

This document describes how to configure the Gateway to authenticate and authorize a user to access a Web Service based on the data stored in a MySQL database.

- The Gateway will be configured to authenticate a user located in the MySQL database.
- Upon successful authentication the Gateway will be configured to extract attributes/roles belonging to this user from a MySQL database.
- The user will be authorized based on the attributes/roles retrieved.
- The attribute can then be used in a SAML Authorization Assertion inserted into a message fro consumption by a downstream service.

Flow of request:



This guide applies to OEG Gateway software products, from version 11.1.1.x upwards.
In this guide the MySQL Database Server used is **MySQL SERVER 5.1.**

### 1.2. MySQL Architecture

MySQL is an open source relational database management system (RDBMS) that uses Structured Query Language (SQL), the most popular language for adding, accessing, and processing data in a database.

### 1.3. Setup Used for this Guide:

1. OEG Gateway 11.1.1.4.0
2. MySQL Server 5.1 for Windows

## 2 Configuring MySQL

### 2.1. The Database

For this guide a sample database populated by users with attributes will be created.
**NOTE:** Please refer to MySQL documentation for details information for installing and running MySQL on Windows and Linux platforms which is available from http://dev.mysql.com/ or www.mysql.com

To start MySQL Server and to create this database:

- Start the MySQL server on Windows using the command "**net start mysql**" (without the quotes) at the DOS prompt .The corresponding command to stop the MySQL server is "**net stop mysql**".
- Browse to All Programs -> My SQL Server 5.1 and click on MySQL Command Line Client
- Browse to the MySQL installation/bin directory and run "**mysql**" (without the quotes) at the DOS prompt.

- The prompt is changed to the **"mysql"** prompt.
- At the prompt enter the password for the admin user
- At the MySQL prompt type: create database employees; and press 'Return'
  *Example:* mysql> create database employees;
- The employee database should now be created.
- To see the list of databases that has be created type: show databases;
  *Example:* mysql> show databases;
- This will show the list of databases in MySQL
- To use the database that has just been created type: use employees;
  *Example:* mysql> use employees
- It should now state that the database 'employees is in use
- A table containing the rows with appropriate values would also need to be created. The table will be called 'employee_data'
- At the MySQL prompt either copy and paste the following command or type it out:

```
CREATE TABLE employee_data
(
emp_id int unsigned not null auto_increment primary key,
username varchar(20),
password varchar(20),
title varchar(30),
age int,
salary int,
perks int,
email varchar(60)
);
```

- Once created to view the table type: show tables;

  Example: mysql> show tables;

- This should display the list of tables associated with the current 'in use' database.

- To view information on the table type: describe employee_data;

  *Example:* describe employee_data

- The only thing left to do now is to populate the database with users and their respected attributes.

- The easiest and quickest way will be to copy and paste the following command in the MySQL prompt:

```
INSERT INTO employee_data (username, password, title, age,salary,
perks,email)
values ("Hubertf", "goodNews", "VP", 27, 120000, 40000,
"hubertf@planetexpress.com");
INSERT INTO employee_data (username, password, title, age,salary, perks,
email) values ("Johnd", "badNews", "Senior Programmer", 32, 120000, 25000,
"john_hagan@planetexpress.com");
INSERT INTO employee_data (username, password, title, age, salary, perks,
email) values ("Ganeshj", "greatNews", "Senior Programmer", 32, 110000,
20000, "g_pillai@planetexpress.com");
INSERT INTO employee_data (username, password, title, age, salary, perks,
email) values ("Anamikar", "seriousNews", "Web Designer", 27, 90000, 15000,
"ana@planetexpress.com");
```

- This will add the user Hubertf and 3 more users to the database.
- To view the data in employee_data:

  *Example:* mysql> select * from employee_data;
- The database is now ready for use.

**Summary of Database created:**
**Database Name:** Employees
**Table Name:** employee_data
**Column Values:** Username, Password, Title, Salary, Perks and Email Address.

# 3 Configuring the Gateway

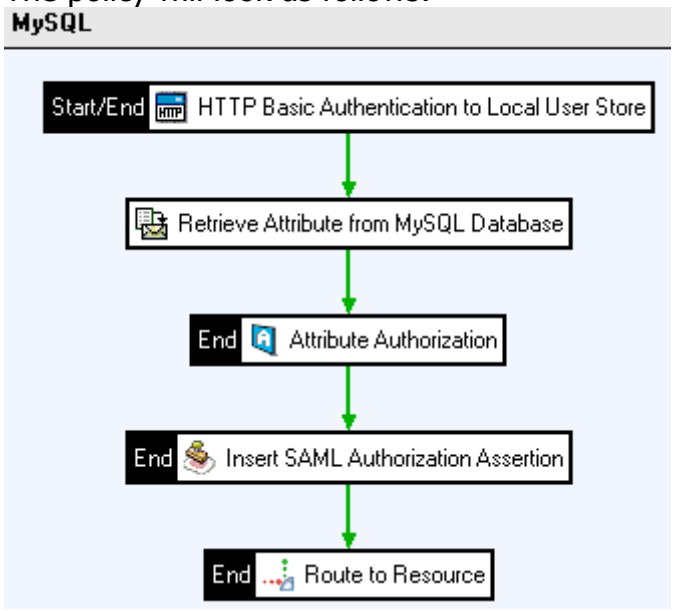
## 3.1. Create the policy

What the policy will do:

- Authenticate the user via the MySQL database.
- Retrieve the Attribute 'title' from the MySQL database.
- The user will be authorized by verifying that the title of the user is VP using the Attributes filter. If the title is not 'VP' in this case the policy will fail.

- The attribute will inserted as part of a SAML Authorization Assertion and forwarded to the resource (web service), in this case reflected back to the test client.

The policy will look as follows:




Before creating this policy it will be necessary to create a Database Connection and a Database Repository:

Creating the policy:

- Click on **External Connections** on the left hand side of **Policy Studio**
- Expand the **External Connections** Tree on the left hand side of **Policy Studio**
- Right Click on **Database Connections** and Click **Add a Database Connection**
- Name the connection 'Employees'
- **URL:** jdbc:mysql://ip_of_database_host:3306/Employees
- **Username:** root (assuming the admin username for MySQL is root)
- **Password:** Enter the password for the MySQL user

**Note:** If the Database and Gateway are not located on the same server it will be necessary to set the access privileges on the database Employees, to allow access for this Username & Password.

This is achieved using the mysql command below, where the user is 'root' and the appropriate password is supplied.
mysql>grant all privileges on employees.* to 'root'@'%' identified by '*password*'

- The rest of the values can be left default. Note: 'Time between Eviction (ms)' should be 10000
- Click on **'OK'**
- Click on **External Connections** on the left hand side of **Policy Studio**
- Right Click on **Database Repositories** and Click **Add a new repository**
- Name the repository 'MySQL'
- **Database Location:** Choose the previously created **Database Connection** 'Employees' from the drop down list
- Click on **'Add/Edit'** next to **'Database Query'**.
- In the **'Name'** field type 'Authentication'
- The query should look like this:

SELECT * FROM employee_data WHERE username = ${authentication.subject.id};
This query will take the user name from the HTTP header and return all the appropriate rows in the database

- **Statement Type:** Query
- **Table Structured With:** Attributes as column names
- Click on **'OK'**
- In the **'Authentication Repository'** window all options can be left as default under the **'Format password received from client'** section.
- Under the **'Query result processing'** section it should be configured as follows:
  - Password Column: Password

  Specify the name of the database table column that contains the user's password. The contents of this column will be compared to the password submitted by the user.
  - Password Type: Clear

  Depending on how the user's password has been stored in the database, select either "Clear Password" or "Digest Password" from the dropdown.

- Authorization Attribute Column: username (defines the name of the

By running the **Database Query**, all of the user's attributes are returned. Only the user's username and password are used for the actual authentication event. It is also possible to use one of the other user's attributes for authorization at a later stage in the policy. The additional "authorization attribute" should be either a username or an X.509 distinguished name (DName). The name of the column containing either the username or the DName should be entered here, but only if this value is required for authorization purposes.

- Authorization Attribute Format: User Name

The Gateway's authorization filters all operate on the basis of a username or DName. In other words, they all evaluate whether a user identified by a username or DName is allowed to access a specific resource. Select the appropriate format from the dropdown depending on what type of user credential is stored in the database table column entered above.

- Right Click on **Policies** in the tree on the left hand side of Policy Studio and click **Add Policy**
- Name the Policy "MySQL"
- Click on the Policy and drag a **"HTTP Basic" filter** located in the **"Authentication" group** of the filter palette located on the right hand side of Policy Studio
- Name of the filter can be left default or changed to any descriptive name.
- **Credential Format:** Select User Name from the drop down list
- **Repository Name:** Select the previously created **Database Repository** 'MySQL' from the drop down list.
- Click on **'Finish'**
- Next drag a **'Retrieve from or write to database' filter** from the **'Attributes' group**
- Name of the filter can be left default or changed to any descriptive name.

**Database tab:**

- **Database Location:** Select the **'Employees Database'** from the list as configured in the 'HTTP Basic' filter.
- Click on **'Add'** next to **'Database Queries'**

- For this guide we will retrieve the job title of the user, so the name of the query will be **'Get Title for user'**
- The query itself should look like follows:

SELECT title from employee_data where username = ${authentication.subject.id}

The query is looking for the title of the username that has been authenticated

- **Statement Type:** Query
- **Table Structured With:** Attributes as column names
- Click on **'OK'** and then click on **'Finish'**

**Advanced Tab:**

- **Associate Attributes with User ID:** Change this to: **authentication.subject.id** which now contains the user name as contained in the HTTP authentication header.
- The 'Retrieve from or write to Database' filter should now be configured correctly.
- The next filter to be configured in the circuit is the **'Attributes' filter** from the **'Authorization' group.**
- Click on **'Add'** to open the **'Add Attributes'** window.
- **Attributes Requirements**: Enter 'Title=VP' (without the quotes)
- Then click on **'OK'** and the click on **'Finish'**
- The next filter in the circuit is an **'Insert SAML Authorization Assertion' filter** which can be found in the **'Authorization' group.**
- **Expire In:** Set to a desired time frame
- **SOAP Actor/Role:** select **'Current Actor/Role Only'**
- **Issuer Name:** Select from the drop down list
- Under **'Advanced Options'** select **'Insert SAML Attribute'** statement
- For 'Resource' enter: ${http.request.uri} ( this is the attribute name for the URI on which the HTTP request was received by Gateway)
- For 'Action' any value can be entered. 'Allow' is used for this guide.
- Click on **'Finish'**
- The last filter in the circuit is the 'Reflect' filter to reflect the message back to the test client (in this case OEG Service Explorer).

- Drag the **'Reflect' filter** from the **'Utility' group** in the filter palette.
- Click on **'OK'**
- Make sure all filters are connected to one another with a green success path.
- The policy has now been configured.

### 3.2. Create a Failure Path for the Policy

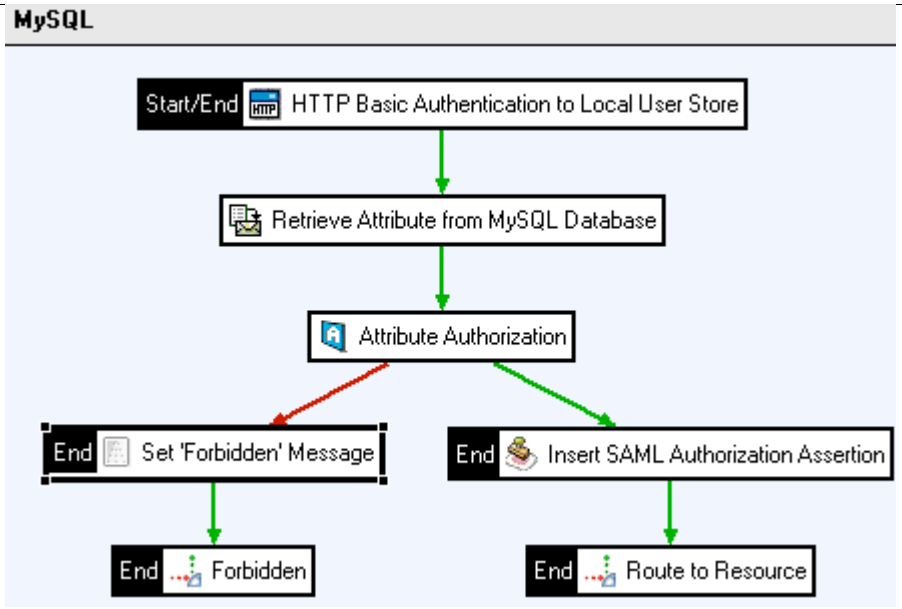
As the current policy will only Authorize users with the 'title' of 'VP', a failure path will be configured for any user who do not have 'VP' as a title and are trying to access the web service.

1. Drag a **'Set Message' filter** from the **'Conversion' group.**
2. For filter name use: Set 'Forbidden' Message
3. Content-Type: text/xml
4. Message Body:

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns:Response xmlns:ns="www.planetexpress">
      <ns:string>Forbidden</ns:string>
    </ns:Response>
  </soap:Body>
</soap:Envelope>
```

5. Click on **'OK'**
6. The next filter is a **'Reflect' filter** from the **'Utility' group**.
7. Name the filter 'Forbidden'
8. Change the **'HTTP Response code status'** to: **403** (indicating Forbidden resource)
9. Drag a red failure path from the 'Attribute Authorization filter to the 'Set Message' filter. Then connect the 'Set Message filter to the 'Reflect' filter titled 'Forbidden'.

The policy should now look like this:




### 3.3. Create a New Relative path to point to Policy

- Open **Policy Studio**
- Expand **Processes** and then **OEG Gateway**
- Right Click on **Default Services** and select "Add Relative Path"
- Name the Relative path as follows: /MySQL
- Map the path to the newly created policy titled "MySQL"
- Click **'OK'**

### 3.4. Ensure policies are updated on the Gateway

- Open the **Policy Studio**
- Click on **Settings**
- Select **Deploy** to ensure that the changes made are propagated to the live Gateway.

# 4 Test the Policy with OEG Service Explorer

A test message will now be sent through to test the policy.
The test SOAP message used for this guide is:

```
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns:Response xmlns:ns="www.planetexpress">
      <ns:string>access required</ns:string>
    </ns:Response>
  </soap:Body>
</soap:Envelope>
```

## 4.1 Set up a message in OEG Service Explorer

Open **OEG Service Explorer**
Load a message request
Click on **'Request Settings'** on the drop down list on the green **'Send Request'** button
Make sure that the URL is set correctly. In this case it will be http://localhost:8080/MySQL (where localhost is the IP of the Gateway)
Click on **OK**
Click on the **HTTP Authentication** tab
Select **HTTP Basic**
Enter the **Username** and **Password** of the user that will be Authenticated via LDAP
User Credentials:
User: hubertf
Password: goodNews
Click on **RUn**
The message would now have been sent

The response following a successful message should look as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  <wsse:Security
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
    <saml:Assertion
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="Id-0000011d28e482f6-0000000000235085-31"
IssueInstant="2008-10-23T08:49:03Z" Issuer="OU=VeriSign
Trust Network, OU=&quot;(c) 1998 VeriSign, Inc. - For
authorized use only&quot;, OU=Class 2 Public Primary
Certification Authority - G2, O=&quot;VeriSign,
Inc.&quot;, C=US" MajorVersion="1" MinorVersion="1">
      <saml:Conditions NotBefore="2008-10-23T08:49:03Z"
NotOnOrAfter="2008-10-23T09:49:03Z"/>
      <saml:AuthorizationDecisionStatement
Decision="Permit" Resource="/SalesData">
        <saml:Subject>
          <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">Hubertf</saml:NameIdentifier>
        </saml:Subject>
        <saml:Action>Allow</saml:Action>
      </saml:AuthorizationDecisionStatement>
      <saml:AttributeStatement>
        <saml:Subject>
          <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">Hubertf</saml:NameIdentifier>
        </saml:Subject>
```

```
        <saml:Attribute AttributeName="title"
AttributeNamespace="urn:vordel:attribute:1.0">
          <saml:AttributeValue>VP</saml:AttributeValue>
        </saml:Attribute>
      </saml:AttributeStatement>
    </saml:Assertion>
  </wsse:Security>
  </soap:Header>
  <soap:Body>
    <ns:Hello xmlns:ns="www.planetexpress.com">
      <ns:string>Check Attribute</ns:string>
    </ns:Hello>
  </soap:Body>
</soap:Envelope>
```

The trace looks as follows:

```
DEBUG   10:37:42:828 [03f4] incoming call on interface *:8080 from
127.0.0.1:207
0
DEBUG   10:37:42:828 [03f4] new connection 0C6508D8, settings source
incoming in
terface
DEBUG   10:37:42:828 [03f4] new server transaction 07A1CB64
DEBUG   10:37:42:828 [03f4] Incoming HTTP request: method=POST,
host=(unset), po
rt=(unset), path=/MySQL, query=(unset), version=1.1
DEBUG   10:37:42:828 [03f4] handle type text/xml with factory class
com.vordel.m
ime.XMLBody$Factory
DEBUG   10:37:42:828 [03f4] run circuit "MySQL"
DEBUG   10:37:42:828 [03f4] run filter [HTTP Basic Authentication to Local
User
Store] {
DEBUG   10:37:42:828 [03f4]     DatabaseRepository.checkCredentials: Check
user
```

```
via Database
DEBUG   10:37:42:828 [03f4]
DatabaseRepository.getQueryResultsFromCache. Key
=hubertf
DEBUG   10:37:42:828 [03f4]     WildcardedPreparedStatement: Running SQL
stateme
nt [SELECT * FROM employee_data WHERE username = ?;]
DEBUG   10:37:42:843 [03f4]     numActive connections AFTER
dataSource.getConnec
tion() is: 1
DEBUG   10:37:42:843 [03f4]     WildcardedPreparedStatement.getStatement:
param
1 is set to value [hubertf]
DEBUG   10:37:42:843 [03f4]     WildcardedPreparedStatement: The query
results a
re [{perks=40000, age=27, email=hubertf@planetexpress.com,
password=goodNews, em
p_id=1, salary=120000, title=VP, username=Hubertf}]
DEBUG   10:37:42:843 [03f4]
DatabaseRepository.addQueryResultsFromCache. Key
=hubertf
DEBUG   10:37:42:843 [03f4]     The password to be compared are client
passwd go
odNews and the db password goodNews
DEBUG   10:37:42:843 [03f4]     The AuthZ Credential set to message is
Hubertf w
ith format Username
DEBUG   10:37:42:843 [03f4]     UsernameAuthN.getResponse: Mapped
'hubertf' to '
Hubertf'. Format=Username
DEBUG   10:37:42:843 [03f4] } = 1
DEBUG   10:37:42:843 [03f4] run filter [Retrieve Attribute from MySQL
Database]
{
DEBUG   10:37:42:843 [03f4]     LookupHandler.process: userIdentity=Hubertf
DEBUG   10:37:42:843 [03f4]     LookupHandler.process: cacheKey=Hubertf
DEBUG   10:37:42:843 [03f4]     WildcardedPreparedStatement: Running SQL
stateme
nt [SELECT title from employee_data where username = ?
```

```
]
DEBUG   10:37:42:843 [03f4]     numActive connections AFTER
dataSource.getConnec
tion() is: 1
DEBUG   10:37:42:843 [03f4]     WildcardedPreparedStatement.getStatement:
param
1 is set to value [Hubertf]
DEBUG   10:37:42:843 [03f4]     WildcardedPreparedStatement: The query
results a
re [{title=VP}]
DEBUG   10:37:42:843 [03f4]     The retrieved attributes are {title=key=[title]
name=[title] values=[VP] namespace=[##nonamespace##]
namespaceForAssertion=[urn:
vordel:attribute:1.0] useForAssertion=[true]}
DEBUG   10:37:42:843 [03f4]     LookupHandler.process: Retrieved
attributes==>
1===> key=[title] name=[title] values=[VP]
namespace=[##nonamespace##] namespace
ForAssertion=[urn:vordel:attribute:1.0] useForAssertion=[true]
DEBUG   10:37:42:843 [03f4]     Copy user attribute [title] value=[VP] to
messag
e attribute [user.title.1]
DEBUG   10:37:42:843 [03f4] } = 1
DEBUG   10:37:42:843 [03f4] run filter [Attribute Authorization] {
DEBUG   10:37:42:843 [03f4] } = 1
DEBUG   10:37:42:843 [03f4] run filter [Insert SAML Authorization Assertion] {
DEBUG   10:37:42:843 [03f4]     SamlAuthZInsertProcessor.createAssertion:
Resour
ce=/SalesData
DEBUG   10:37:42:843 [03f4]     Assertion.setValidity: from=Thu Oct 23
10:37:42
BST 2008 until=Thu Oct 23 11:37:42 BST 2008
DEBUG   10:37:42:843 [03f4] } = 1
DEBUG   10:37:42:843 [03f4] run filter [Route to Resource] {
DEBUG   10:37:42:843 [03f4] } = 1
-------------------------------------------------------------------------------------------
-----------------
```

The following are the results of a message with user credentials not containing the attribute value as is configured in the 'Attribute Authorization' filter.
The message response should now look like this:


The trace output:

---

DEBUG 14:59:02:671 [0154] Incoming HTTP request: method=POST, host=(unset), po
rt=(unset), path=/MySQL, query=(unset), version=1.1
DEBUG 14:59:02:671 [0154] handle type text/xml with factory class com.vordel.m
ime.XMLBody$Factory
DEBUG 14:59:02:671 [0154] run circuit "MySQL"
DEBUG 14:59:02:671 [0154] run filter [HTTP Basic Authentication to Local User
Store] {
DEBUG 14:59:02:671 [0154] DatabaseRepository.checkCredentials: Check user
via Database
DEBUG 14:59:02:671 [0154] DatabaseRepository.getQueryResultsFromCache. Key
=Johnd
DEBUG 14:59:02:671 [0154] WildcardedPreparedStatement: Running SQL stateme
nt [SELECT * FROM employee_data WHERE username = ?;]
DEBUG 14:59:02:671 [0154] numActive connections AFTER dataSource.getConnec
tion() is: 1

```
DEBUG   14:59:02:671 [0154]     WildcardedPreparedStatement.getStatement:
param
1 is set to value [Johnd]
DEBUG   14:59:02:671 [0154]     WildcardedPreparedStatement: The query
results a
re [{perks=25000, age=32, email=john_hagan@planetexpress.com,
password=badNews,
emp_id=2, salary=120000, title=Senior Programmer, username=Johnd}]
DEBUG   14:59:02:671 [0154]
DatabaseRepository.addQueryResultsFromCache. Key
=Johnd
DEBUG   14:59:02:671 [0154]     The password to be compared are client
passwd ba
dNews and the db password badNews
DEBUG   14:59:02:671 [0154]     The AuthZ Credential set to message is
Johnd wit
h format Username
DEBUG   14:59:02:671 [0154]     UsernameAuthN.getResponse: Mapped
'Johnd' to 'Jo
hnd'. Format=Username
DEBUG   14:59:02:671 [0154] } = 1
DEBUG   14:59:02:671 [0154] run filter [Retrieve Attribute from MySQL
Database]
{
DEBUG   14:59:02:671 [0154]     LookupHandler.process: userIdentity=Johnd
DEBUG   14:59:02:671 [0154]     LookupHandler.process: cacheKey=Johnd
DEBUG   14:59:02:671 [0154]     WildcardedPreparedStatement: Running SQL
stateme
nt [SELECT title from employee_data where username = ?
]
DEBUG   14:59:02:671 [0154]     numActive connections AFTER
dataSource.getConnec
tion() is: 1
DEBUG   14:59:02:671 [0154]     WildcardedPreparedStatement.getStatement:
param
1 is set to value [Johnd]
DEBUG   14:59:02:671 [0154]     WildcardedPreparedStatement: The query
results a
re [{title=Senior Programmer}]
```

```
DEBUG   14:59:02:671 [0154]     The retrieved attributes are {title=key=[title]
name=[title] values=[Senior Programmer] namespace=[##nonamespace##]
namespaceFor
Assertion=[urn:vordel:attribute:1.0] useForAssertion=[true]}
DEBUG   14:59:02:671 [0154]     LookupHandler.process: Retrieved
attributes==>
1===> key=[title] name=[title] values=[Senior Programmer]
namespace=[##nonamespa
ce##] namespaceForAssertion=[urn:vordel:attribute:1.0]
useForAssertion=[true]
DEBUG   14:59:02:671 [0154]     Copy user attribute [title] value=[Senior
Progra
mmer] to message attribute [user.title.1]
DEBUG   14:59:02:671 [0154] } = 1
DEBUG   14:59:02:671 [0154] run filter [Attribute Authorization] {
DEBUG   14:59:02:671 [0154] } = 0
ERROR   14:59:02:671 [0154] The message [Id-0000011d2a004f0f-
0000000001b54362-22
] logged Failure at 10.23.2008 14:59:02,671 with log description: Attribute
base
d authorization failed
DEBUG   14:59:02:671 [0154] run filter [Set 'Not Authorized Resource'
Message] {
DEBUG   14:59:02:671 [0154]
ConversionProcessor.setConvertedMessage:The cont
entype of the converted message is text/xml
DEBUG   14:59:02:671 [0154]
ConversionProcessor.setConvertedMessage:the conv
erted message is  <soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
tance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.x
mlsoap.org/soap/envelope/">
  <soap:Body>
    <ns:Response xmlns:ns="www.planetexpress">
      <ns:string>Forbidden</ns:string>
    </ns:Response>
  </soap:Body>
</soap:Envelope>
```

```
DEBUG   14:59:02:671 [0154]     closing transaction input before sending
expecte
d '100' response
DEBUG   14:59:02:671 [0154]     handle type text/xml with factory class
com.vord
el.mime.XMLBody$Factory
DEBUG   14:59:02:671 [0154]     ConversionProcessor.setConvertedMessage:
coverte
d message is added to the the pipeline
DEBUG   14:59:02:671 [0154]     ChangeMessageProcessor.convert: finished
DEBUG   14:59:02:671 [0154] } = 1
DEBUG   14:59:02:671 [0154] run filter [Not Authorized] {
DEBUG   14:59:02:671 [0154] } = 1
```

The user was authenticated but not authorized to access the resource and followed the failure path to display the message above.

## 5 Conclusion

This document is a simplistic demonstration on how to setup the OEG Gateway to authenticate, retrieve and use attributes located in a MySQL database. This configuration can be part of a larger policy, including features such as XML threat detection and conditional routing, features which are out of scope in this document but are covered in other documents which can be obtained from Oracle at http://www.oraclel.com.

Oracle Enterprise Gateway
May 2011
Author:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment



**SOFTWARE. HARDWARE. COMPLETE.**