ORACLE® HYPERION ENTERPRISE PERFORMANCE MANAGEMENT SYSTEM

RELEASE 11.1.1.x

IMPLEMENTING A CUSTOM AUTHENTICATION MODULE

ORACLE
ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM

CONTENTS IN BRIEF

About EPM System Custom Authentication

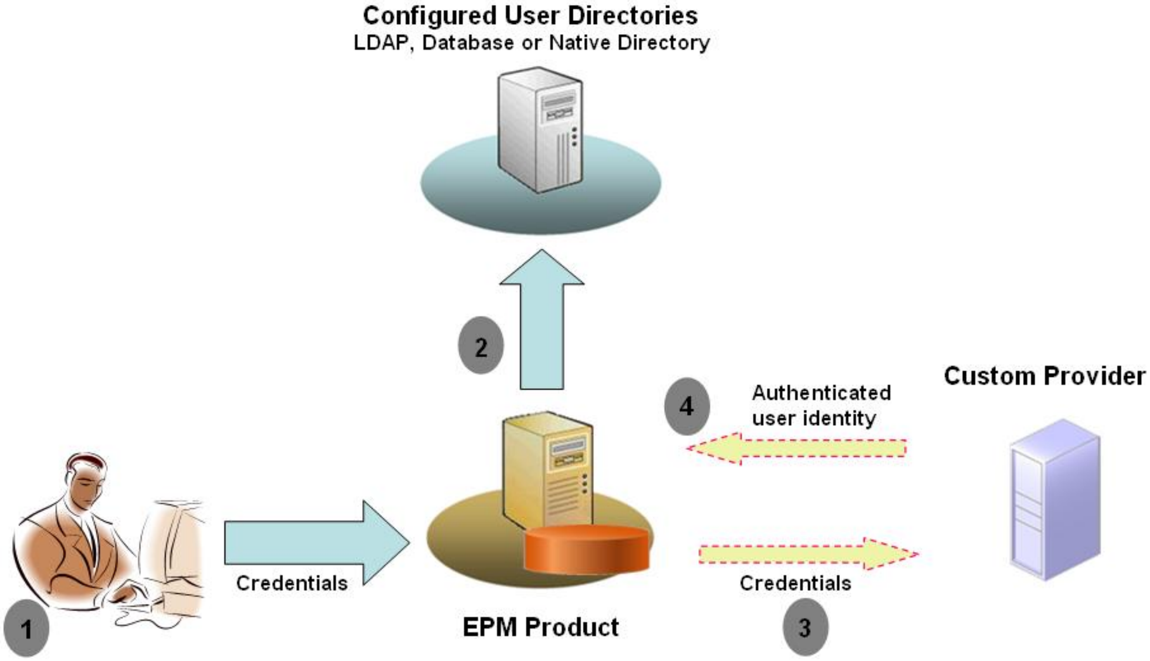
A custom authentication module is a Java module that a customer develops and implements to authenticate Oracle Hyperion Enterprise Performance Management System users. Generally, EPM System products use a login screen to capture the user name and password, which are used to authenticate users. Instead of using EPM System authentication, you can write a custom authentication module to authenticate users belonging to the user directory configured in an EPM System environment.

Implementing a custom authentication module does not involve modifying the EPM System products. You deploy the custom Java module on the server to seamlessly integrate your own authentication module with EPM System.

A custom authentication module may be used with both the thick clients (for example, Oracle Hyperion Smart View for Office, Fusion Edition, and Oracle Essbase Studio) and thin clients (for example, Oracle Enterprise Performance Management Workspace, Fusion Edition).

The custom authentication module uses the information a user enters when logging into an EPM System product. If enabled for a user directory, it authenticates users through the custom authentication module. On successfully authenticating the user, the custom authentication module returns the user name.

The following illustration presents a sample custom authentication scenario:


Configured User Directories
LDAP, Database or Native Directory
2
4
Authenticated user identity
Custom Provider
Credentials
Credentials
1
EPM Product
3

For example, you can use RSA SecurID infrastructure as the custom provider to ensure transparent strong authentication to EPM System clients products. The overview of steps:

1. The user enters credentials (generally, user name and password) to access an EPM System product. These credentials should uniquely identify the user to the provider used by the custom authentication module. For example, if you are using an RSA SecurID infrastructure to authenticate users, the user enters an RSA user ID and PIN (not an EPM System user ID and password).

2. Using the search order (see "Search Order" on page 4), EPM System cycles through configured user directories to locate the user.

    - If the current user directory is not configured for custom authentication, EPM System tries to locate and authenticate the user through EPM System authentication.
    - If the user directory is configured for custom authentication, EPM System delegates the authentication process to the custom module.

3. If EPM System delegates authentication to the custom module, the custom authentication module accepts the credentials and uses its own logic to direct user authentication against a custom provider; for example, RSA SecurID infrastructure.
4. If the custom authentication module authenticates the user against its provider, it returns the user name to EPM System, or it returns a Java exception.

    The user name returned by the custom authentication module must be identical to a user name in one of the user directories that is enabled for custom authentication.

    - If the custom authentication module returns a user name, EPM System locates the user in a user directory that is enabled for custom authentication. At this stage, EPM System does not search the user directories that are not configured for custom authentication.
    - If the custom authentication module throws an exception or returns a null user, EPM System continues to search for the user in the remaining user directories in the search order. If a user that matches the credentials is not found, EPM System displays an error.

Use Case Examples and Limitations

Custom authentication implementation scenarios include the following:

- Adding One-Time Password Support
- Performing authentication against a Resource Access Control Facility (RACF)
- Adding Simple Authentication and Security Layer (SASL) bind to LDAP-enabled user directories instead of simple LDAP binds

Authentication with challenge/response mechanism may not work well if you implement a custom authentication module. Custom messages thrown by the custom authentication module are not propagated to the clients. Because clients; for example, EPM Workspace, override the error message to display a generic message, the following scenarios are not valid:

- Two RSA SecurID PINs in a row
- Password variant with challenges, such as enter first, last and third characters of password

Custom Authentication Requirements

A custom authentication module is a Java module that a customer develops and implements to authenticate EPM System users. You can use a custom authentication module with thick clients (for example, Oracle Essbase Studio and Oracle Smart Space, Fusion Edition) and thin clients (for example, EPM Workspace).

To integrate with EPM System security, the custom authentication module must implement the public interface `CSSCustomAuthenticationIF`, defined in `com.hyperion.css` package as a part of the standard Oracle's Hyperion® Shared Services APIs. See http://download.oracle.com/docs/cd/E12825_01/epm.111/epm_security_api_11111/client/com/hyperion/css/CSSCustomAuthenticationIF.html.

Design and Coding Considerations

- "Search Order" on page 4
- "Custom Module Implementation" on page 8
- "User Directories and Custom Authentication Module" on page 8
- "`CSSCustomAuthenticationIF` Java Interface" on page 8

Search Order

In addition to Native Directory, multiple user directories can be configured in Shared Services. A default search order position is assigned to all configured user directories. You can modify the search order from Oracle's Hyperion® Shared Services Console. Excepting Native Directory, you can remove configured user directories from the search order. EPM System does not use the user directories that are not included in the search order. See the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*.

The search order determines the order in which EPM System cycles through the user directories to authenticate users. If the user is authenticated in a user directory, EPM System stops the search and returns the user. EPM System denies authentication and returns an error if the user cannot be authenticated against any user directories in the search order.

Impact of Custom Authentication on Search Order

The use of custom authentication affects how EPM System security interprets the search order.

If the custom authentication module returns a user name, EPM System locates the user only in a user directory that is enabled for custom authentication. At this stage, EPM System ignores user directories that are not configured for custom authentication.

Understanding the Custom Authentication Flow

The following use case scenarios are used to explore custom authentication flow:

- "Use Case Scenario 1" on page 5
- "Use Case Scenario 2" on page 6
- "Use Case Scenario 3" on page 7

Use Case Scenario 1

Table 1 details the EPM System user directory configuration and search order used in this scenario. This scenario assumes that the custom authentication module uses an RSA infrastructure to authenticate users.

Table 1 Setup for Scenario 1

User Directory Type and Name	Search Order	Custom Authentication	Sample User Names	Password[*]
Native Directory	1	Disabled	test_user_1 test_user_2 test_user_3	password
LDAP-Enabled SunONE_West	2	Disabled	test_ldap1 test_ldap_2 test_user_3 test_ldap_4	ldappassword
LDAP-Enabled SunONE_East	3	Enabled	test_ldap1 test_ldap_2 test_user_3	ldappassword on SunONE and RSA PIN in custom module

[*] For simplicity, it is assumed that all users use the same user directory password.

To initiate the authentication process, a user enters a user name and password in the login screen of an EPM System product.

In this scenario, the custom authentication module performs the following actions:

- Accepts a user name and RSA PIN as the user credentials
- Returns a user name in *username@providername* format; for example, test_ldap_2@SunONE_East to EPM System security.

Table 2 User interaction and results

User Name and Password	Authentication Result	Login User Directory
test_user_1/password	Success	Native Directory
test_user_3/password	Success	Native Directory
test_user_3/ldappassword	Success	SunONE_West (search order 2)[*]
test_user_3/RSA PIN	Success	SunONE_East (search order 3)[†]
test_ldap_2/ldappassword	Success	SunONE_West (search order 2)
test_ldap_4/RSA PIN	Failure	

User Name and Password	Authentication Result	Login User Directory
	EPM System displays an authentication error.[‡]	

[*] The custom authentication cannot authenticate this user because the user entered EPM System credentials. EPM System can identify this user only in a user directory that is not enabled for custom authentication. The user is not in Native Directory (search order number 1), but is identified in SunONE West (search order number 2).

[†] EPM System does not find this user in Native Directory (search order number 1) or SunONE West (search order number 2). The custom authentication module validates the user against RSA Server and returns `test_user_3@SunONE_EAST` to EPM System. EPM System locates the user in SunONE East (search order number 3), which is a custom authentication–enabled user directory.

[‡] Oracle recommends that all users authenticated by the custom module be present in a custom authentication–enabled user directory included in the search order. Login fails if the user name that is returned by the custom authentication module is not present in a custom authentication–enabled user directory included in the search order.

Use Case Scenario 2

Table 3 details the EPM System user directory configuration and search order used in this scenario. This scenario assumes that the custom authentication module uses an RSA infrastructure to authenticate users.

In this scenario, the custom authentication module performs the following actions:

- Accepts a user name and RSA PIN as the user credentials
- Returns a user name; for example, `test_ldap_2` to EPM System security.

Table 3 A sample search order

User Directory	Search Order	Custom Authentication	Sample User Names	Password[*]

Native Directory	1	Disabled	`test_user_1`
			`test_user_2`
			`test_user_3`
				`password`

LDAP-Enabled; for example, SunONE	2	Enabled	`test_ldap1`
			`test_ldap2`
			`test_user_3`
				`ldappassword` on SunONE and `RSA PIN` in custom module

[*] For simplicity, it is assumed that all users use the same user directory password.

To initiate the authentication process, a user enters a user name and password in the login screen of an EPM System product.

Table 4 User interaction and results

User Name and Password	Login Result	Login User Directory
`test_user_1/password`	Success	Native Directory
`test_user_3/password`	Success	Native Directory
`test_user_3/ldappassword`	Failure	SunONE[*]

User Name and Password	Login Result	Login User Directory
test_user_3/RSA PIN	Success	SunONE[†]

[*] Authentication of user against Native Directory fails because of password mismatch. Authentication of user using the custom authentication module fails because password used is not a valid RSA PIN. EPM System does not try to authenticate this user in SunONE (search order 2), because custom authentication settings override EPM System authentication in this directory.

[†] Authentication of user against Native Directory fails because of password mismatch. The custom authentication module authenticates the user and returns the user name test_user_3 to EPM System.

Use Case Scenario 3

Table 5 details the EPM System user directory configuration and search order used in this scenario. This scenario assumes that the custom authentication module uses an RSA infrastructure to authenticate users.

For clarity in scenarios such as this, Oracle recommends that your custom authentication module return the user name in username@providername format; for example, test_ldap_4@SunONE.

Table 5 A sample search order

User Directory	Search Order	Custom Authentication	Sample User Names	Password[*]
Native Directory	1	Enabled	test_user_1 test_user_2 test_user_3	password
LDAP-Enabled; for example, MSAD	2	Disabled	test_ldap1 test_ldap4 test_user_3	ldappassword
LDAP-Enabled; for example, SunONE	3	Enabled	test_ldap1 test_ldap4 test_user_3	ldappassword on SunONE and RSA PIN in custom module

[*] For simplicity, it is assumed that all users use the same user directory password.

To initiate the authentication process, a user enters a user name and password in the login screen of an EPM System product.

Table 6 User interaction and results

User Name and Password	Authentication Result	Login User Directory
test_user_1/password	Success	Native Directory
test_user_3/password	Success	Native Directory
test_user_3/ldappassword	Success	MSAD (search order 2)
test_ldap_4/ldappassword	Success	MSAD (search order 2)

User Name and Password	Authentication Result	Login User Directory
`test_ldap_4/RSA PIN`	Success	SunONE (search order 3)

Custom Module Implementation

Only one custom module is supported for an EPM System deployment. You can enable custom authentication for one or more user directories in the search order. See:

- "Updating User Directory Configuration" on page 18
- "Updating Security Options" on page 19

Your custom authentication module must implement the public interface `CSSCustomAuthenticationIF`, which is defined in `com.hyperion.css` package. This document assumes that you have a fully functional custom module that defines the logic for authenticating users against the user provider of your choice. After you develop and test a custom authentication module, you must implement it in EPM System environment. Implementation procedures involve:

- "Deploying the Custom Authentication Module" on page 13
- "Shared Services Procedures" on page 18
- "Testing Your Deployment" on page 19

User Directories and Custom Authentication Module

To use the custom authentication module, user directories that contain EPM System user and group information can be individually configured to delegate authentication to the custom module.

EPM System users who are authenticated using a custom module must be present in one of the user directories included in the search order (see "Search Order" on page 4). Also, the user directory must be configured to delegate authentication to the custom module.

The identity of the user in the custom provider (for example, `1357642` in RSA SecurID infrastructure) may be different from the user name in the user directory (for example, `jDoe` in Oracle Internet Directory) configured in Shared Services. After authenticating the user, the custom authentication module must return the user name `jDoe` to EPM System.

Note: As a best practice, Oracle recommends that the user name in the user directories configured in EPM System be identical to those available on the user directory used by the custom authentication module.

`CSSCustomAuthenticationIF` Java Interface

Your custom authentication module must use the `CSSCustomAuthenticationIF` Java interface to integrate with EPM System security framework.

Your custom module must return a user name string if custom authentication is successful or a an error message if authentication is not successful. For the authentication process to be completed, the user name returned by the custom authentication module must be present in one of the user directories included in Shared Services search order. EPM System security framework supports *username@providerName* format.

Note: Ensure that the user name that the custom authentication module returns does not contain an * (asterisk) because EPM System security framework interprets it as a wildcard character while searching for users.

See "Sample Code 1" on page 9 for CSSCustomAuthenticationIF interface signature.

Your custom authentication module can be a class file (expected default class name is CustomAuthenticationImpl) included in a Java archive (expected default archive name is customAuth.jar). The default package is com.hyperion.css.custom.

During runtime, if the implementing class is not found in the classpath, EPM System security framework initialization components scan for customAuth.jar to load the implementing class from HYPERION_HOME\common\CSS\9.5.0.0\lib.

For detailed information about the CSSCustomAuthenticationIF interface, see Security API documentation.

Authenticate Method

The authenticate method from CSSCustomAuthenticationIF supports custom authentication. The authenticate method accepts credentials (user name and password) that the user entered while trying to access the EPM System as input parameters. This method returns a string (user name) if custom authentication is successful. It throws a java.lang.Exception if authentication is not successful. The user name returned by the method should uniquely identify a user in one of the user directories included in Shared Services search order. EPM System security framework supports the *username@providerName* format.

See "Sample Code 1" on page 9 for authenticate method signature.

Note: To initialize resources; for example, a JDBC connection pool, use the class constructor. Doing so improves performance by not loading resources for every authentication.

Sample Code 1

The following code snippet is an empty implementation of the custom module.

```
package com.hyperion.css.custom;
import java.util.Map;
import com.hyperion.css.CSSCustomAuthenticationIF;
import org.apache.log4j.Logger; // imports Log4j's Logger
  public class CustomAuthenticationImpl
               implements CSSCustomAuthenticationIF {
    //Get the Logger to log exception or debug information
    //Log information is written to the Shared Services security log
    static Logger logger=Logger.getLogger

("com.hyperion.css.custom.CustomAuthenticationImpl");
      public String authenticate(Map context,String userName,
                                 String password) throws Exception{
        try{
          //Custom code to find and authenticate the user goes here.
          //The code should do the following:
          //if authentication succeeds:
                //set authenticationSuccessFlag = true
                //return authenticatedUserName
          // if authentication fails:
                //ensure debug is enabled using logger.isDebugEnabled()
                //log an authentication failure
                //throw authentication exception
        }
        catch (Exception e){
          //Custom code to handle authentication exception goes here
          //Create a new exception, set the root cause
          //Set any custom error message
          //Return the exception to the caller
        }
        return authenticatedUserName;
        }
  }
```

Input parameters are as follows:

- context: A map that contains key-value pair of locale information
- user name: An identifier that uniquely identifies the user to the user directory where the custom module authenticates the user. The user enters the value of this parameter while logging into an EPM System product.
- password: The password set for the user in the user directory where the custom module authenticates the user. The user enters the value of this parameter while logging into an EPM System product.

Sample Code 2

The following sample code demonstrates custom authentication of users using user name and password contained in a flat file. You must initialize user and password lists in the class constructor to make this work.

```
package com.hyperion.css.security;

import java.util.Map;
import java.util.HashMap;
import com.hyperion.css.CSSCustomAuthenticationIF;
import java.io.*;

public class CSSCustomAuthenticationImpl implements
CSSCustomAuthenticationIF{
  //get the Logger to log Exception or other info useful for debugging
  /*static Logger logger = Logger.getLogger
    ("com.hyperion.css.custom.CSSCustomAuthenticationImpl"); */
  static final String DATA_FILE = "datafile.txt";

/**
  * authenticate method includes the core implementation of the
  * Custom Authentication Mechanism. If custom authentication is
  * enabled for the provider, authentication operations
  * are delegated to this method. Upon successful authentication,
  * this method returns a valid user name, using which EPM System
  * retrieves the user from a custom authentication enabled provider.
  * User name can be returned in the format username@providerName,
  * where providerName indicates the name of the underlying provider
  * where the user is available. authenticate method can use other
  * private methods to access various core components of the
  * custom authentication module.

  *   @param context
  *   @param userName
  *   @param password
  *   @return
  *   @throws Exception
*/

Map users = null;

public CSSCustomAuthenticationImpl(){
  users = new HashMap();
  InputStream is = null;
  BufferedReader br = null;
  String line;
  String[] userDetails = null;
  String userKey = null;
  try{
     is = CSSCustomAuthenticationImpl.class.getResourceAsStream(DATA_FILE);
     br = new BufferedReader(new InputStreamReader(is));
     while(null != (line = br.readLine())){
        userDetails = line.split(":");
          if(userDetails != null && userDetails.length==3){
            userKey = userDetails[0]+ ":" + userDetails[1];
            users.put(userKey, userDetails[2]);
          }
      }
     }
  catch(Exception e){

        }
  finally{
     try{
        if(br != null) br.close();
        if(is != null) is.close();
     }
     catch(IOException ioe){
        ioe.printStackTrace();
     }
  }
}
```

```
/* Use this authenticate method snippet to return username from a flatfile
*/

public String authenticate(Map context,String userName, String
password)throws Exception{

  //UserName : user input for the userName
  //Password : user input for password
  //context  : Map, can be used to additional information required by
  //           the custom authentication module.

  String authenticatedUserKey = userName + ":" + password;

  if(users.get(authenticatedUserKey)!=null)
     return(String)users.get(authenticatedUserKey);
  else throw new Exception("Invalid User Credentials");
    }

/* Refer to this authenticate method snippet to return username in
   username@providername format */

public String authenticate(Map context,String userName, String
password)throws Exception{

  //UserName : user input for userName
  //Password : user input for password
  //context  : Map, can be used to additional information required by
  //           the custom authentication module.

  //Your code should uniquely identify the user in a custom provider and in
a configured
  //user directory in Shared Services. EPM Security expects you to append
the provider
  //name to the user name. Provider name must be identical to the name of a
custom
  //authentication-enabled user directory specified in Shared Services.

  //If invalid arguments, return null or throw exception with appropriate
message
  //set authenticationSuccessFlag = false

  String authenticatedUserKey = userName + ":" + password;
  if(users.get(authenticatedUserKey)!=null)
     String userNameStr = (new StringBuffer())
                           .append((String)users.get(authenticatedUserKey))
                           .append("@").append(PROVIDER_NAME).toString();
            return userNameStr;
  else throw new Exception("Invalid User Credentials");
    }
}
```

Data File for Sample Code 2

Ensure that the data file is named `datafile.txt`; which is the name used in the sample code, and that it is included in the Java archive that you create.

Use the following as the contents of the flat file that is used as the custom user directory to support the custom authentication module implemented by Sample Code 2 (see "Sample Code 2" on page 10).

```
xyz:password:admin
test1:password:test1@LDAP1
test1:password:test1
test1@LDAP1:password:test1@LDAP1
test1@1:password:test1
user1:Password2:user1@NTLM1
user1_1:Password2:user1
user3:Password3:user3
DS_User1:Password123:DS_User1@MSAD1
DS_User1:Password123:DS_User1
DS_User1@1:Password123:DS_User1
```

Use the following as the contents of the flat file that is used as the custom user directory if you plan to return user name in *username@providername* format.

```
xyz:password:admin
test1:password:test1
test1@1:password:test1
user1_1:Password2:user1
user3:Password3:user3
DS1_1G100U_User61_1:Password123:DS1_1G100U_User61
DS1_1G100U_User61_1@1:Password123:DS1_1G100U_User61
TUser:password:TUser
```

Deploying the Custom Authentication Module

You must deploy your custom authentication module into theEPM System product classpath to enable it to be loaded by the EPM System security libraries.

You must complete the deployment process simultaneously for all products, because it affects the authentication process of all EPM System products. Because EPM System products cannot be used during this process, Oracle recommends that you budget sufficient maintenance time before deploying a custom authentication module. Generally, the deployment process involves the following:

- Stopping EPM System products
- Adding custom authentication module to the classpath. See "Adding a Custom Authentication Module to Classpath" on page 14.
- Deploying EPM System products to the application server
- Updating user directory configurations using Shared Services Console. See:
  - "Updating User Directory Configuration" on page 18
  - "Updating Security Options" on page 19
- Starting EPM System products

Prerequisites

- A fully tested Java archive that contains custom authentication module binaries.
- Access to Shared Services as Shared Services administrator.

Adding a Custom Authentication Module to Classpath

To implement custom authentication module, you must add the custom authentication module Java archive to the EPM System product classpath. You must include the custom authentication module Java archive in the following locations:

- A common location on each EPM System host machines. See "Adding a Custom Authentication Module to Common Location on EPM System Host Machines" on page 14
- Classpath of EPM System products.

  For J2EE applications, update the enterprise archive (`.ear`) or Web archive (`.war`) and then redeploy it. See "Adding a Custom Authentication Module to Common Location on EPM System Host Machines" on page 14. For non-J2EE applications, you must perform product-specific procedures. See:

  - "Adding a Custom Module to the Essbase Classpath" on page 16
  - "Adding a Custom Module to the Financial Management Classpath" on page 17
  - "Adding a Custom Module to the Reporting and Analysis Core Services Classpath" on page 17

Note: Some procedures in this document use the `jar` command to create enterprise and Web archives. You must have a JDK installation to use the `jar` command. Also, the JDK installation path must be added tot he classpath of your system.

Note: After deploying a custom authentication module, review "Guidelines for Securing EPM System" in the *Oracle Hyperion Enterprise Performance Management System Security Administration Guide* for information on securing EPM System products, including the procedure to regenerate SSO Encryption Key.

Adding a Custom Authentication Module to Common Location on EPM System Host Machines

You must add the custom module Java archive to a common location on each machine that hosts an EPM System product.

➤ To add the custom authentication module to the common location:

1. **Stop all EPM System products.**
2. **On each EPM System product host machine, copy the custom authentication module Java archive into** *`HYPERION_HOME`*`/common/CSS/9.5.0.0/lib`.

Preparing EPM System J2EE Application Archives

You must perform this procedure for each EPM System product other than the following:

- Oracle Essbase
- Oracle's Hyperion Reporting and Analysis Core Services
- Oracle Hyperion Financial Management, Fusion Edition

Note: After adding a custom module to the classpath, you must perform additional procedures for Shared Services. See "Shared Services Procedures" on page 18.

➤ To add a custom authentication module to the classpath for EPM System J2EE applications:

1. **Ensure that EPM System products are not running.**
2. **Locate the EPM System J2EE application archive that you want to prepare for deployment. Generally, the archive is available in `HYPERION_HOME/products/PRODUCT_NAME/AppServer/InstallableApps/common`.**

   For example, Oracle Hyperion Planning, Fusion Edition enterprise archive (`HyperionPlanning.ear`) location is `HYPERION_HOME/products/Planning/AppServer/InstallableApps/common` and Shared Services Web archive (`interop.war`) is in `HYPERION_HOME/products/Foundation/AppServer/InstallableApps/common`.
3. **Create a backup copy of the application archive.**
4. **Create two temporary directories; for example, `temp1` and `temp2`, on your file system.**
5. **Copy the application archive into a temporary directory; for example, `temp1`.**
6. **Optional: Perform this step only if you are adding custom authentication module to the classpath in an enterprise archive; for example, `HyperionPlanning.ear`.**

   Using the `jar` command, extract the contents of the enterprise archive into the temporary directory (`temp1`), and delete the enterprise archive.
7. **From the temporary directory (`temp1`), copy the Web archive (for example, `HyperionPlanning.war`) to another temporary directory (`temp2`).**
8. **Using the `jar` command, extract the contents of the Web archive into the temporary directory (`temp2`), and delete the Web archive.**
9. **Copy the custom authentication Java archive (`.jar`) to the `WEB-INF/lib` directory within the temporary directory; for example, to `temp2/WEB-INF/lib`.**
10. **Recreate the enterprise archive or web archive for your product:**

    a. Using the `jar` command, create a Web archive; for example, `HyperionPlanning.war`, of the contents of the temporary directory (`temp2`). Ensure the following:

    - The Web archive name is identical to the original archive name; for example, `HyperionPlanning.war`
    - The directory structure is maintained in the Web archive.

Note: Perform the following steps only if you are creating an enterprise archive.

b. Copy the Web archive; for example, `HyperionPlanning.war`, to `temp1`, replacing the existing Web archive.

c. Using the `jar` command, create an enterprise archive; for example, `HyperionPlanning.ear`, of the contents of the temporary directory (`temp1`). Ensure the following:

- The enterprise archive name is identical to the original archive name; for example, `HyperionPlanning.ear`
- The directory structure is maintained in the enterprise archive.

11 **Copy the Web or Enterprise archive from the preceding step to** *`HYPERION_HOME`*`/products/`*`PRODUCT_NAME`*`/AppServer/InstallableApps/common` **directory, replacing the existing archive.**

For example, Oracle Hyperion Planning, Fusion Edition enterprise archive (`HyperionPlanning.ear`) should be copied to *`HYPERION_HOME`*`/products/Planning/AppServer/InstallableApps/common` and Shared Services Web archive (`interop.war`) should be copied to *`HYPERION_HOME`*`/products/Foundation/AppServer/InstallableApps/common`.

12 **Using Oracle's Hyperion Enterprise Performance Management System Configurator, deploy the application archive to the application server. See the *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide* for deployment instructions.**

Caution! To ensure that existing application data is preserved during the database configuration process, select the `Connect to a previously configured database` option while redeploying the application archive. Also, do not change any of the existing database connection parameters.

Note: After redeploying Shared Services Web application, you must perform additional procedures. See "Shared Services Procedures" on page 18.

13 **Start EPM System products.**

Adding a Custom Module to the Essbase Classpath

To enable custom authentication, you must add the custom authentication class file to the `css-9_5_0.jar` file that Essbase uses.

➤ To add a custom module to the Essbase classpath:

1 **Make sure that Essbase Server is shut down.**

2 **On the machine that hosts Essbase Server, create a backup copy of** *`HYPERION_HOME`*`/common/CSS/9.5.0.0/lib/css-9_5_0.jar`**.**

3 **Add the custom authentication module to existing** `css-9_5_0.jar`

a. Using the `jar` command, extract the contents of *`HYPERION_HOME`*`/common/CSS/9.5.0.0/lib/css-9_5_0.jar` into a temporary directory; for example, `temp1`.

b. Delete `css-9_5_0.jar` from the temporary directory.

c. Copy the custom authentication module class files into the temporary directory where you extracted the contents of `css-9_5_0.jar`.

   Be sure to copy the file into the directory appropriate for the package structure of your class file. If you used *`com.yourcompany.customauth`* package, make sure that your class is copied to the `com/yourcompany/customauth` directory within the temporary directory.

d. Using the `jar` command, recreate `css-9_5_0.jar` using the contents of the temporary directory. Make sure that the directory structure is maintained in `css-9_5_0.jar`.

4 **Copy `css-9_5_0.jar` from the preceding step to *`HYPERION_HOME`*`/common/CSS/9.5.0.0/lib`.**

5 **Start Essbase server.**

Adding a Custom Module to the Financial Management Classpath

To add a custom module to the Financial Management classpath, you must update Oracle's Hyperion Shared Services Registry with the location and name of the of the custom module.

➤ To add a custom module to the Financial Management classpath:

1 **Stop all Financial Management processes, especially `CASSecurity.exe`.**

2 **Make sure that the custom authentication module Java archive is available in *`HYPERION_HOME`*`/common/CSS/9.5.0.0/lib`.**

3 **Append the location of the custom module Java archive; for example, `%HYPERION_HOME%\COMMON\css\9.5.0.0\lib\customAuth.jar` to the Oracle Hyperion Financial Management, Fusion Edition value data of the following key. Be sure to use a semicolon (;) to separate the existing value data from the path that you append.**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial
Management\Server\Authentication\ClassPath
```

Adding a Custom Module to the Reporting and Analysis Core Services Classpath

Make sure that custom authentication module is copied into the EPM Workspace classpath. See "Adding a Custom Authentication Module to Classpath" on page 14.

➤ To enable Reporting and Analysis Core Services custom authentication:

1 **Make sure that EPM Workspace and Reporting and Analysis Core Services are shut down.**

2 **Make sure that the custom authentication module Java archive is available in *`HYPERION_HOME`*`/common/CSS/9.5.0.0/lib` on the machine where Reporting and Analysis Core Services is deployed.**

3 **Add the custom authentication Java archive to the classpath.**

a. Using a text editor, open *HYPERION_HOME*/common/workspacert/9.5.0.0/bin/workspace.app.

b. Add an entry such as the following for the custom authentication Java archive:

   ${home}/common/CSS/9.5.0.0/lib/CustAuth.jar, where CustAuth.jar indicates the name of the custom authentication module's Java archive.

c. Save and close workspace.app.

4 **Start Oracle's Hyperion Reporting and Analysis Core Services and EPM Workspace.**

Shared Services Procedures

- "Updating User Directory Configuration" on page 18
- "Updating Security Options" on page 19

Updating User Directory Configuration

By default, custom authentication is enabled for all user directories other than Native Directory. EPM System security automatically searches for a custom authentication module named com.hyperion.css.custom.CustomAuthenticationImpl in the classpath.

You can override the default behavior to disable custom authentication for specific external user directories or enable it for Native Directory.

If your custom authentication module does not use the default class name or package name, you must specify additional security settings to enable custom authentication.

You need to perform this procedure only under the following scenarios:

- To disable custom authentication for an external user directory included in the search order
- To enable custom authentication for Native Directory

➤ To update user directory configuration:

1 **Start Shared Services.**

2 **Log into Shared Services Console as a Shared Services administrator.**

3 Select **Administration**, and then **Configure User Directories.**

The Defined User Directories screen opens, listing all user directories, including Native Directory, that are already configured.

4 **Select the user directory for which you want to change the custom authentication setting.**

Note: EPM System uses only the user directories included in the search order.

5 Click **Edit.**

6 Select **Show Advanced Options.**

7 In **Custom Module**, perform an action:

- Deselect **Use Authentication Module** to disable custom module for the current user directory.
- Select **Use Authentication Module** to enable custom module for the current user directory.

8 Click **Finish.**

9 Repeat this procedure to update the configuration of the user directories for which you want to enable/ disable custom authentication.

Updating Security Options

By default, EPM System security automatically searches for a custom authentication module named `com.hyperion.css.custom.CustomAuthenticationImpl` in the classpath.

If you do not use the default name, you must update the security option to identify the custom module that EPM System security should use to authenticate users.

Note: Make sure the custom Java archive is in the classpath before performing the following procedure, because Shared Services dynamically looks for it.

➤ To update security options:

1 Log into Shared Services Console as a Oracle's Hyperion® Shared Services administrator.

2 Select **Administration**, and then **Configure User Directories.**

3 Select **Security Options.**

4 Select **Advanced Options.**

5 In **Authentication Module**, enter the fully qualified Java class name of the custom authentication module that should be used to authenticate users on all user directories for which custom authentication module is selected.

Caution! If you are enabling custom authentication for Oracle Essbase, you must specify the fully-qualified class name of the custom class even if you use the default class and package name.

6 Click **OK.**

Testing Your Deployment

If Native Directory is not configured for custom authentication, do not use Native Directory users to test custom authentication.

Note: It is your responsibility to identify and correct any issues with the custom authentication module. Oracle assumes that your custom module works flawlessly to map a user from the user directory used by the custom module to a user on a custom authentication-enabled user directory available in EPM System search order.

To test your deployment, log into EPM System using user credentials from the user directory; for example, an RSA SecurID infrastructure, used by the custom module. These credentials may be different from the EPM System credentials.

Your implementation can be considered successful if EPM System products allow you to access their resources. An error indicating that the user was not found is not always an indicator of an unsuccessful implementation. In such cases, verify that credentials you entered are present in the custom user store and a matching user is present in one of the custom authentication-enabled user directories in EPM System search order.

➤ To test custom authentication:

1 **Make sure that EPM System products are running.**

2 **Launch Oracle's Hyperion® Shared Services Console or Oracle Enterprise Performance Management Workspace, Fusion Edition.**

3 **Log in as a user defined on the user directory, used by the custom authentication module.**

   a. In **Username**, enter your user identifier; for example, an RSA User ID.

   b. In **Password**, enter a password; for example, an RSA PIN.

   c. Click **Login**.

4 **Verify that you can access EPM System product resources.**

Tips and Tricks

After the authentication steps, Oracle Hyperion Enterprise Performance Management System security checks the retrieved user name in a custom authentication-enabled user directory included in the search order before verifying user roles.

If you have configured the same user directory twice, one as custom authentication-enabled and the other as custom authentication-disabled, ensure that you provision the user from the right provider depending on whether the user logs in with an LDAP password or custom authentication credentials.

COPYRIGHT NOTICE

EPM System Implementing a Custom Authentication Module, 11.1.1

Authors: EPM Information Development Team

ORACLE
ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM